# Índice general

# Capítulo 1

# Teoría de grafos

## 1.1. Definiciones básicas

Un grafo es una estructura que permite estudiar relaciones entre objetos. A lo largo de este capítulo se mostrarán diversos ejemplos de situaciones donde los grafos permiten modelar los vínculos entre elementos: relación de amistad, de autoridad, de parentesco, entre un grupo de personas, relación de conectividad entre ciudades, terminales o dispositivos, etc.

La teoría de grafos provee herramientas (conceptos y algoritmos) para analizar grafos y obtener información útil de ellos. Por ejemplo, se podría analizar un grafo de conectividad de terminales para responder a la pregunta: ¿Cada terminal está conectada con todas las demás? O aún: ¿Cuál es el camino mínimo más largo para conectar dos terminales? ¿Existe alguna terminal de alta implicación en la conectividad de todo el sistema?

Formalmente, un GRAFO es un conjunto $V$ (no vacío) de vértices o nodos, $V = \{v_1, v_2, ..., v_n\}$ y un conjunto de aristas o arcos $E = \{e_1, e_2 ..., e_m\}$, donde cada arista es $e_k = \{v_i, v_j\}$, es decir, cada arista relaciona dos vértices (iguales o distintos). El par de vértices relacionados por una arista no es ordenado, entonces $e_k = \{v_i, v_j\} = \{v_j, v_i\}$.

Una arista que relaciona un vértice con sí mismo se denomina BUCLE o LAZO.

Dos vértices $v_i$ y $v_j$ son ADYACENTES si hay una arista que los relaciona, es decir, si $e_k = \{v_i, v_j\}$ es una arista del grafo. En tal caso, se dice que $e_k$ es INCIDENTE en $v_i$ y en $v_j$, y estos vértices son los EXTREMOS de la arista.

Cuando en un conjunto de vértices y aristas hay aristas repetidas (es decir, más de una arista para algún par de vértices $v_i$, $v_j$), se denomina MULTIGRAFO.

Un GRAFO DIRIGIDO es un conjunto $V$ (no vacío) de vértices o nodos, $V = \{v_1, v_2, ..., v_n\}$ y un conjunto de aristas dirigidas $E = \{e_1, e_2 ..., e_m\}$, tal que cada aristas relaciona un par ordenado de vértices: $e_k = (v_i, v_j)$. En este caso, como en el par importa el orden, entonces $e_k = (v_i, v_j) \neq (v_j, v_i)$.

Dada una arista dirigida $e_k = (v_i, v_j)$, se dice que $e_k$ es INCIDENTE en $v_i$ y en $v_j$, $e_k$ sale de $v_i$ y llega a $v_j$, $v_i$ es adyacente hacia $v_j$ y $v_j$ es adyacente desde $v_i$.

Si en un conjunto de vértices y aristas dirigidas hay aristas repetidas, se denomina MULTIGRAFO DIRIGIDO.

En un grafo (grafo dirigido) un VÉRTICE AISLADO es un vértice tal que no tiene

aristas incidentes en él.

El GRADO de un vértice $v_i$ en un grafo, denotado $gr(v_i)$ es el número de veces que aristas inciden en él (un bucle cuenta dos veces).

En un grafo dirigido, el GRADO DE ENTRADA de un vértice $v_i$, denotado $ge(v_i)$ es el número de aristas que llegan a él, y el GRADO DE SALIDA es el número de aristas que salen de él.

La cantidad de vértices se denota con $|V|$ (cardinalidad del conjunto $V$) y la cantidad de aristas se denota con $|E|$ (cardinalidad del conjunto $E$).

Existe una relación útil entre los grados de los vértices y el número de aristas:

**Proposición 1.1.**
*En todo grafo se cumple que* $2|E| = \sum_{v_i \in V} gr(v_i)$. ♢

**Proposición 1.2.**
*En todo grafo dirigido se cumple que* $|E| = \sum_{v_i \in V} ge(v_i) = \sum_{v_i \in V} gs(v_i)$. ♢

Un grafo es REGULAR si todos sus vértices tienen el mismo grado. Si el grado de cada vértice es k, se dice que es k-regular.

***Ejemplo* 1.1**
*En la figura 1.1a se muestra un grafo con* $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$ *y* $E = \{e_1, e_2, ..., e_{13}\}$, *con* $e_1 = \{v_1, v_2\}$, $e_2 = \{v_1, v_3\}$, $e_3 = \{v_2, v_3\}$, $e_4 = \{v_1, v_7\}$, $e_5 = \{v_7, v_8\}$, ..., $e_{12} = \{v_2, v_6\}$, $e_{13} = \{v_2, v_5\}$ *(el orden de las aristas y también de los vértices es totalmente arbitrario). Los vértices* $v_1$ *y* $v_2$ *tienen grado 5; los vértices* $v_3$, $v_4$, $v_6$, $v_7$, $v_9$ *tienen grado 2, los vértices* $v_5$ *y* $v_8$ *tienen grado 3, mientras que el vértice* $v_{10}$ *tiene grado 0. Este vértice es aislado.*

*En la parte b de la figura 1.1 se observa un multigrafo. Las aristas* $\{u_1, u_3\}$ *y* $\{u_2, u_6\}$ *están repetidas. Además, el grafo tiene un bucle en el vértice* $u_4$. *El grado de este vértice es 5.*

*Un grafo dirigido se muestra en la parte c de la figura. No es un multigrafo porque no hay aristas repetidas. Las aristas* $(v_2, v_5)$ *y* $(v_5, v_2)$ *son distintas, porque si bien relaciona el mismo par de vértices, las direcciones son distintas. Los grados de entrada y salida de algunos vértices son:* $gs(v_1) = 2$, $ge(v_1) = 0$, $gs(v_3) = 1$, $ge(v_3) = 3$, $gs(v_4) = 1$, $ge(v_4) = 1$, *etc.*

*Finalmente, la figura 1.1d muestra un multigrafo dirigido. La arista* $(u_2, u_4)$ *está repetida. Los grados de algunos vértices son:* $gs(u_1) = 2$, $ge(u_1) = 0$, $gs(u_4) = 1$, $ge(u_4) = 2$, *etc.* ■

***Ejemplo* 1.2**
*Los dos grafos de la figura 1.2 son regulares. El de la izquierda es 2-regular (cada uno de los 7 vértices tienen grado 2). El grafo de la derecha es 3-regular.* ■

La ubicación de los vértices y aristas en la representación gráfica de un grafo es totalmente arbitraria. Lo importante en el concepto de grafo, así como lo que se intenta visualizar en la representación gráfica, es la relación entre pares de vértices.

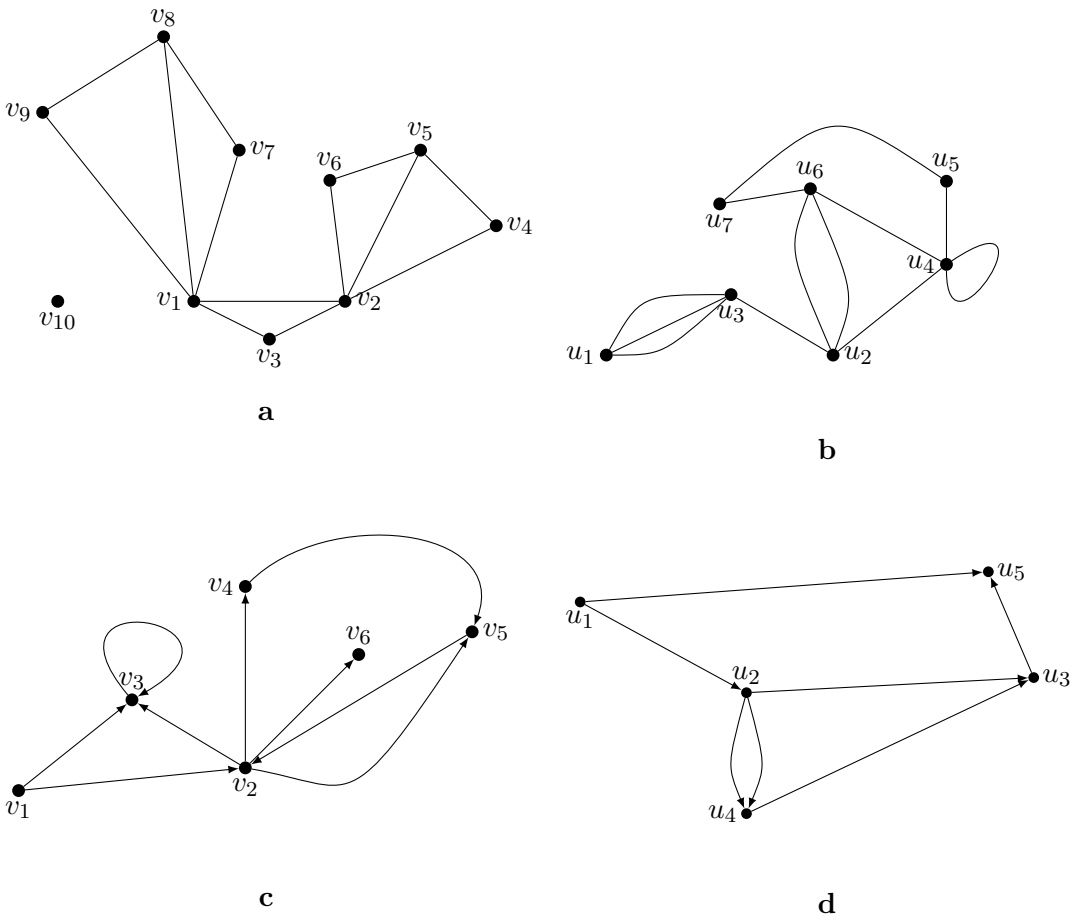



Figura 1.1: Grafos y multigrafos del ejemplo 1.1

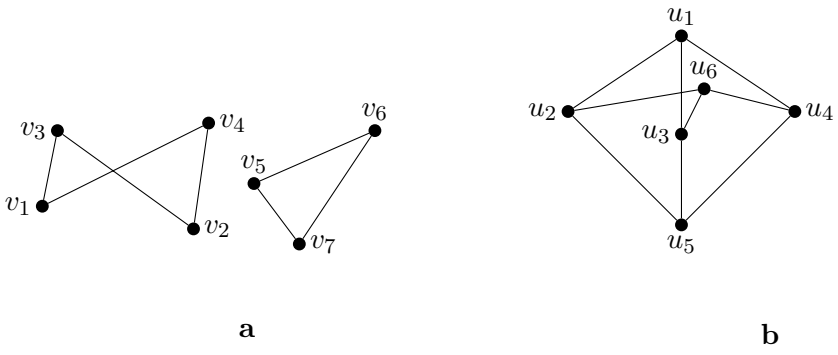


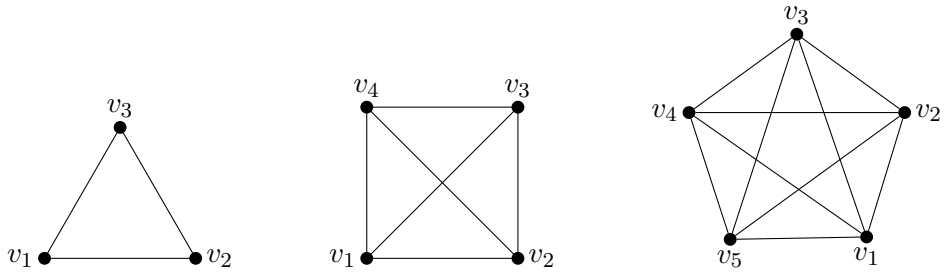

Figura 1.2: Grafos del ejemplo 1.2

## 1.2. Ejemplos de grafos

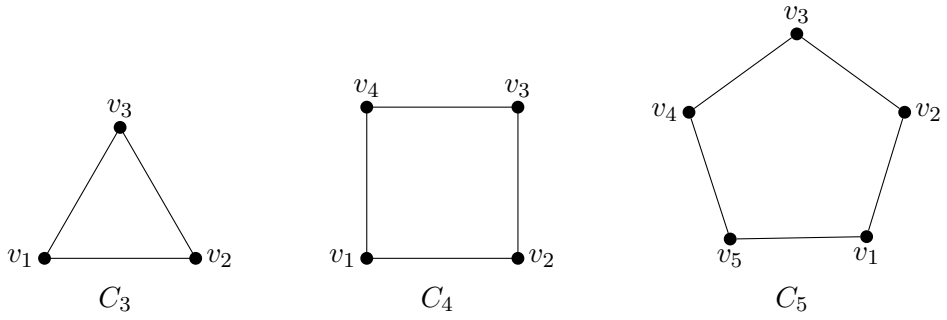
**a. Grafo completo.** Un grafo es COMPLETO si para todo par de vértices distintos $v_i$, $v_j$, existe una arista $e_k = \{v_i, v_j\}$ que los une. Es decir, si cada vértice es adyacente a todos los demás.

Si un grafo completo tiene $n$ vértices, es decir, $|V| = n$, entonces $|E| = n(n-1)/2$ y $gr(v_i) = n - 1$ para todo vértice $v_i$. Resulta que un grafo completo de $n$ vértices es (n-1)-regular.

Al grafo completo de $n$ vértices se lo suele denotar $K_n$.

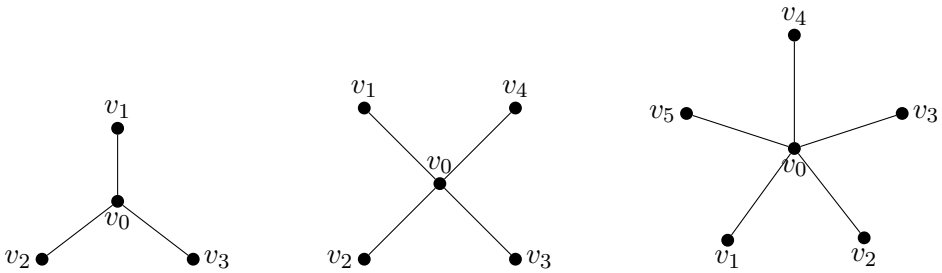



**b. Grafo ciclo**. Un grafo tiene estructura de ciclo si tiene vértices $v_1$, $v_2$, ..., $v_n$, tal que las aristas son $\{v_1, v_2\}$, $\{v_2, v_3\}$, ..., $\{v_n, v_1\}$. Al grafo ciclo de $n$ vértices se lo denota $C_n$. Además $|E| = n$ y $gr(v_i) = 2$ para todo vértice $v_i$. Todo grafo ciclo es 2-regular.

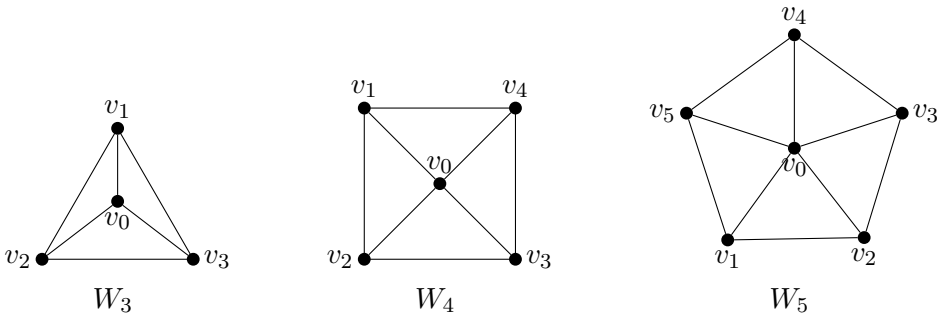



**c. Grafo estrella**. Un grafo estrella tiene un vértice central y $n$ vértices periféricos. Hay una arista entre el vértice central y cada vértice periférico. Se denota $S_n$.

En $S_n$, $|V| = n + 1$, $|E| = n$ y $gr(v_i) = 1$ si son vértices periféricos, y $gr(v_0) = n$ para el vértice central.



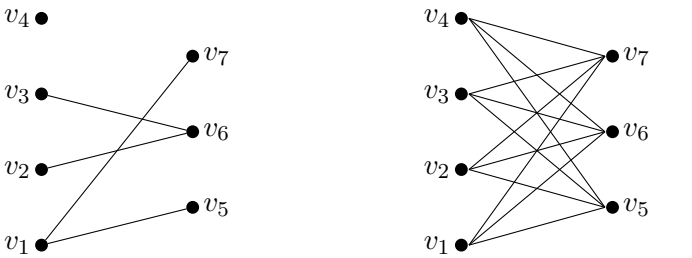

**d. Grafo rueda**. Un grafo rueda es un grafo ciclo, junto con un grafo estrella. Si el grafo rueda se origina con un ciclo de $n$ vértices, entonces $|V| = n + 1$ y se lo denota $W_n$. Además, $|E| = 2n$ y $gr(v_i) = 3$ si son vértices periféricos, y $gr(v_0) = n$ para el vértice central.




**e. Grafo lineal**. Un grafo lineal de $n$ vértices, denotado $L_n$ tiene vértices $v_1$, $v_2$, ..., $v_n$, tal que las aristas son $\{v_1, v_2\}$, $\{v_2, v_3\}$..., $\{v_{n-1}, v_n\}$. En este grafo, $|V| = n$, $|E| = n - 1$, $gr(v_i) = 2$ si $i = 2, 3, ..., n - 1$, y $gr(v_1) = gr(v_n) = 1$.


$L_n$

**f. Grafo bipartito**. Un grafo bipartito es un grafo tal que el conjunto de vértices puede partirse en dos subconjuntos disjuntos, $V_1$ y $V_2$, y las aristas unen vértices de $V_1$ con vértices de $V_2$. Un grafo bipartito completo es un grafo bipartito tal que hay una arista para cada par de vértices $\{v_i, v_j\}$ con $v_i \in V_1$ y $v_j \in V_2$. Si $|V_1| = n_1$ y $|V_2| = n_2$, el grafo bipartito completo se denota $K_{n_1,n_2}$ y tiene $|E| = n_1 \cdot n_2$ aristas y $|V| = n_1 + n_2$ vértices. Además, $gr(v_i) = n_2$ si $v_i \in V_1$ y $gr(v_i) = n_1$ si $v_i \in V_2$



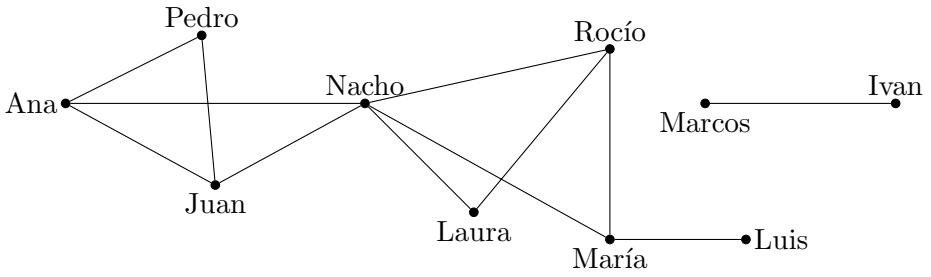
## 1.3. Grafos como modelos

### 1.3.1. Grafo de conocidos

Considere todos los alumnos de la carrera de Sistemas de UAI. Es posible que no todos se conozcan entre sí. ¿Pero es posible que cualquier alumno pueda acceder a conocer a cualquier otro mediante una cadena de conocidos? ¿Es posible que para cada par de alumnos, haya un conocido en común? ¿Quién es el más popular?

La situación puede modelarse con un grafo, y la respuesta a preguntas como las mencionadas pueden obtenerse con la teoría sobre grafos que se expondrá en este capítulo.

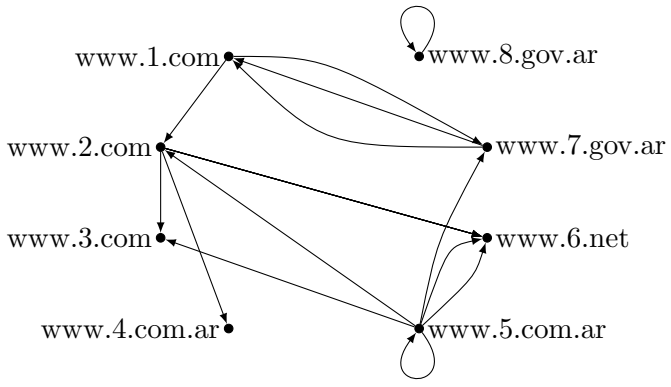
Se representa cada alumno con un vértice, y la relación *se conocen* se representa con una arista. Claramente, como la relación es simétrica (si A conoce a B, B conoce a A), las aristas son no dirigidas. En la figura siguiente se expone un pequeño grafo de conocidos.




### 1.3.2. Grafo de hipervínculos

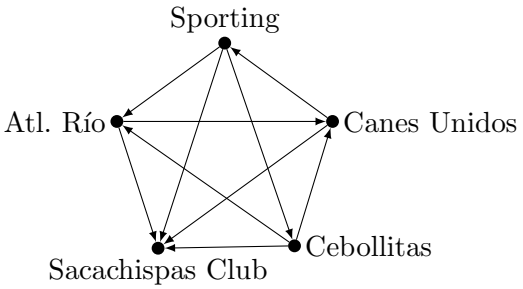
Considere la red de Internet. Se puede modelar considerando cada página como un vértice, y la relación entre páginas *tiene un hipervínculo a* se representa con aristas. En este caso la relación no es simétrica, por lo que necesariamente las aristas son dirigidas. Además, una misma página puede tener más de un enlace a otra. Y una página puede tener un hipervínculo a sí misma. Entonces, el modelo genera un multigrafo dirigido.

Con este modelo, pueden responderse preguntas como: ¿Es posible acceder a todas las páginas, comenzando por una determinada, siguiendo hipervínculos? ¿Cuál es la página más linkeada?



### 1.3.3. Grafo de torneo

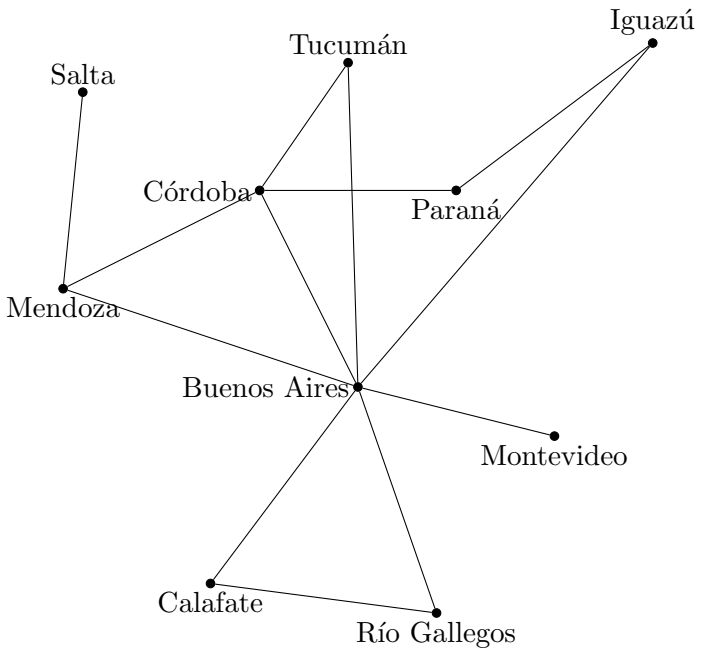
Considere un torneo entre $n$ equipos, en el que todos juegan contra todos y no hay empates. Los resultados se pueden representar con un grafo dirigido, como el de la figura para $n$=5, donde una arista desde un equipo A hacia otro B indica la relación *A le gana a B*.




Se pueden analizar cuestiones simples como: ¿Quién ganó (perdió) más veces? O cuestiones más complejas como por ejemplo: ¿Cómo se puede determinar el ganador del torneo? ¿Cómo se puede armar un ranking de los equipos, en base a los resultados?

### 1.3.4. Grafo de rutas aéreas

Considere una empresa de transporte aéreo que tiene como posibles destinos las ciudades de Buenos Aires, Paraná, Montevideo, Mendoza, Tucumán, Córdoba, Salta, Iguazú, Calafate y Río Gallegos. Los posibles vuelos entre ciudades pueden representarse en un grafo. Si vuela desde una ciudad A a otra ciudad B, también puede volar de B a A. Entonces el grafo es no dirigido.



El grafo permite analizar situaciones como: ¿Hay algún destino tal que si su aeropuerto está inoperante durante un tiempo, quedan ciudades desconectadas en este esquema de rutas? Claramente Buenos Aires lo es. Otro destino con esta característica es Mendoza, ya que si no se llega hasta allí, Salta queda desconectada.

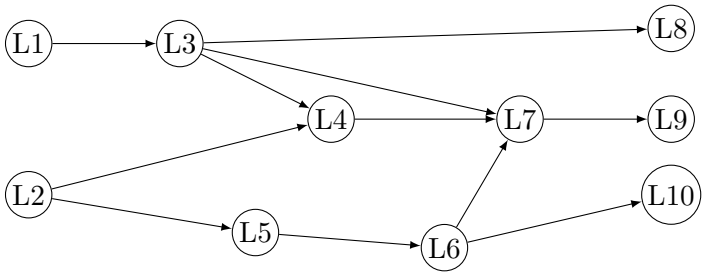
Por otra parte, si una persona en Salta quiere llegar a Iguazú, ¿cuál es la menor cantidad de vuelos que debe tomar?

### 1.3.5. Grafo de precedencias

Un programa informático ejecuta una serie de instrucciones. El programa puede ejecutarse más rápido si ciertas instrucciones se ejecutan simultáneamente. Sin embargo, hay cierto orden de precedencia entre las instrucciones, que imposibilita que todas puedan realizarse simultáneamente. Considere, por ejemplo, la serie de comandos:

L1≻ a:=1
L2≻ b:=0
L3≻ a:=a+1
L4≻ c:=a+b
L5≻ d:=b-2
L6≻ e:=2d
L7≻ d:=c-a
L8≻ Mostrar a
L9≻ Mostrar d
L10≻ Mostrar e

La línea 3 debe ejecutarse después de que la línea 1 haya sido ejecutada, el comando 9 debe ejecutarse luego de la instrucción 7, etc. Todas las precedencias se pueden mostrar en un grafo dirigido:



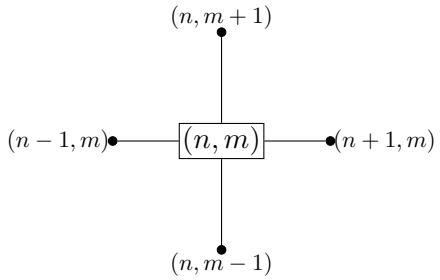

Analizando el grafo, pueden responderse cuestiones como: Si se usa ejecución simultánea, ¿cuánto tiempo se necesita para ejecutar todas las instrucciones? Si cada instrucción requiere 1 unidad de tiempo para llevarse a cabo, en el grafo dado se observa que se requieren como mínimo 4 unidades de tiempo para ejecutar las 10 instrucciones.

### 1.3.6. Grafo de adyacencias

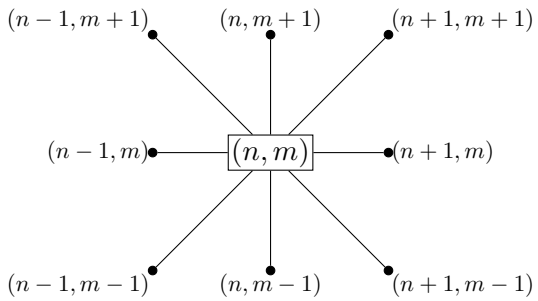
Una imagen digital es un conjunto de puntos $(n, m)$ con coordenadas enteras (pixeles). Para estudiar temas como componentes conexas de la imagen, borde, interior, etc., se define la noción de adyacencia: dos puntos $(n_1, m_1)$ y $(n_2, m_2)$ son adyacentes o vecinos

si $|n_1 - n_2| + |m_1 - m_2| = 1$. La relación de vecindad entre puntos se puede representar en un grafo: los vértices representan los puntos y las aristas representan la relación de adyacencia.

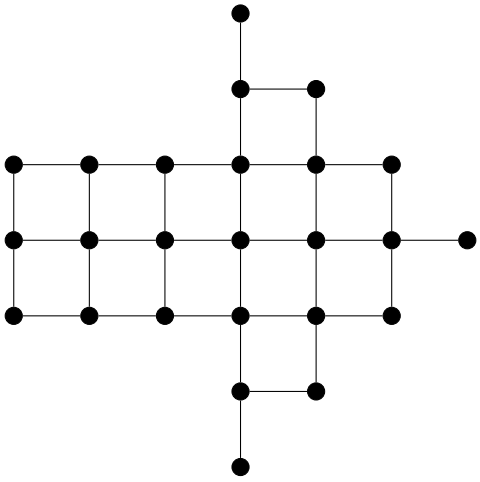
Con la definición de adyacencia dada, cada punto tiene 4 vecinos:




Es usual también definir la relación de adyacencia de la siguiente manera: dos puntos distintos $(n_1, m_1)$ y $(n_2, m_2)$ son adyacentes o vecinos si $|n_1 - n_2| \leqslant 1$ y $|m_1 - m_2| \leqslant 1$. De esta forma, cada punto tiene 8 vecinos:




El grafo a continuación representa la imagen digital de una flecha, usando la relación de adyacencia de 4 vecinos.



Se puede definir punto de borde como aquel cuyo vértice correspondiente tiene grado menor a 4, y los puntos interiores como aquellos cuyo vértice correspondiente tiene grado 4.

## 1.4. Más definiciones

Un CAMINO en un grafo es una sucesión de vértices del grafo tal que cada par de vértices consecutivos está relacionado por una arista. Suele llamarse CAMINO $w - v$ a un camino cuyo primer vértice es $w$, y cuyo último vértice es $v$.

En un grafo dirigido, un CAMINO DIRIGIDO es una sucesión de vértices del grafo tal que para cada vértice de la sucesión (excepto el último) existe una arista dirigida desde ese vértice al siguiente en la sucesión. Suele llamarse CAMINO DIRIGIDO $w - v$ a un camino cuyo primer vértice es $w$, y cuyo último vértice es $v$.

Dos vértices $w$ y $v$ están CONECTADOS si existe un *camino* $w - v$ entre ellos.

La LONGITUD del camino (camino dirigido) es la cantidad de aristas que usa.

Un camino $w - v$ (camino dirigido) es CERRADO si $w = v$.

Un RECORRIDO es un camino que no repite aristas.

Un CIRCUITO es un recorrido cerrado.

Un CAMINO SIMPLE es un camino que no repite vértices (excepto quizás el primero y el último en el caso de camino simple cerrado).

Un CICLO es un camino simple cerrado.

La DISTANCIA entre dos nodos $w$ y $v$ es la longitud del camino más corto entre $w$ y $v$.

***Ejemplo* 1.3**
*Considere el grafo de la figura 1.3. Hay varios caminos que conectan $v_1$ y $v_3$. Uno de ellos es $C_1 : v_1 - v_7 - v_5 - v_4 - v_3$; otro camino es $C_2 : v_1 - v_7 - v_8 - v_5 - v_7 - v_8 - v_2 - v_3$; y otro posible camino es $C_3 : v_1 - v_5 - v_6 - v_4 - v_5 - v_8 - v_2 - v_3$.*

*$C_1$ no repite aristas ni vértices, por lo tanto, es camino simple (todo camino simple además es recorrido). $C_2$ repite una arista, la arista $\{v_7, v_8\}$, por lo tanto, no es recorrido. $C_3$ no repite aristas, es un recorrido. Como el camino pasa dos veces por el vértice $v_5$ no es camino simple.*

*Caminos cerrados que pasen por $v_1$ son: $C_4 : v_1 - v_5 - v_6 - v_4 - v_5 - v_7 - v_1$. Éste es un circuito, ya que no repite aristas, es un recorrido cerrado. Pero no es ciclo, ya que repite un vértice.*

*Un ciclo que pase por $v_2$ es $C_5 : v_2 - v_3 - v_4 - v_5 - v_8 - v_2$.*

*La distancia entre los vértices se pueden buscar en este grafo por inspección: se analizan todos los caminos posibles, y se toma el de longitud menor. Así, se obtiene que la distancia de $v_1$ a $v_2$ es 3, de $v_1$ a $v_3$ es 3, de $v_1$ a $v_4$ es 2, de $v_1$ a $v_5$ es 1, de $v_1$ a $v_6$ es 2, $v_1$ a $v_7$ es 1, de $v_1$ a $v_8$ es 2.*

Un camino $w - v$ siempre contiene un camino simple de $w$ a $v$. Un camino simple en un grafo con $n$ vértices, tiene longitud menor o igual a $n - 1$.

Un grafo es CONEXO si para cada par de vértices, existe un camino entre ellos. Como todo camino contiene un camino simple, se puede también afirmar, equivalentemente, que un grafo es conexo si para cada par de vértices, existe un *camino simple* entre ellos. Entonces:

**Teorema 1.3.**
*Un grafo de $n$ vértices es conexo si y sólo si entre cualquier par de vértices existe un camino de longitud menor o igual a $n - 1$.* ♣

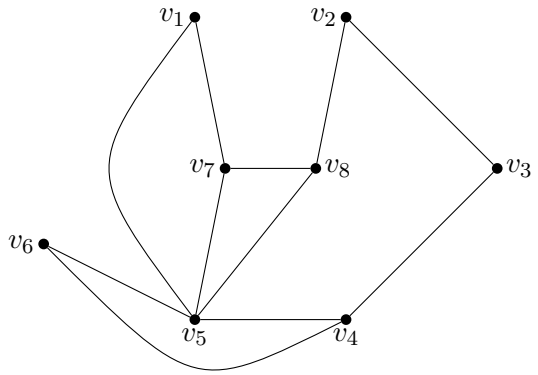



Figura 1.3: Grafo del ejemplo 1.3

Para un grafo dirigido, el concepto de conexión se refiere al grafo no dirigido que se obtiene de eliminar la dirección de las aristas: un grafo dirigido es conexo si y sólo si el grafo obtenido al suprimir la dirección de las aristas es conexo. Además, se dice que un grafo dirigido es FUERTEMENTE CONEXO si para cada par de vértices $w$ y $v$, existe un camino dirigido de $w$ a $v$ y un camino dirigido de $v$ a $w$.

***Ejemplo* 1.4**
*Los grafos de las figura 1.1a y 1.2b son conexos. El grafo de conocidos mostrado en la subsección 1.3.1 y el grafo de la figura 1.2a son no conexos.*

*El grafo de hipervínculos (grafo dirigido) de la subsección 1.3.2 no es conexo.*

*El grafo dirigido 1.1c y el grafo de torneo mostrado en la subsección 1.3.3 son conexos. Ninguno de estos dos es fuertemente conexo.* ■

Las COMPONENTES CONEXAS de un grafo son los subgrafos conexos maximales. Es decir, es un subconjunto de vértices tal que junto con las aristas entre ellos resulta un grafo conexo, y tal que no se pueda agregar ningún vértice más sin que pierda conexión. Cada vértice está en una y sólo una componente conexa.

Un grafo conexo tiene una sola componente conexa, que es el grafo mismo.

***Ejemplo* 1.5**
*El grafo de la figura 1.2a está formado por dos componentes conexas. En una componente conexa están los vértices $\{a, b, c, d\}$ y las aristas entre ellos, y en la otra están los vértices $\{e, f, g\}$ y las aristas entre ellos.* ■

Un vértice tal que al eliminarlo del grafo, junto con las aristas que inciden en él, aumenta la cantidad de componentes conexas del grafo, se denomina VÉRTICE DE ARTICULACIÓN. Un vértice de articulación se caracteriza por ser un vértice intermedio en todos los caminos que conectan a algún par de vértices.

En el grafo de rutas aéreas de la subsección 1.3.4, el vértice Buenos Aires es un vértice de articulación, y Mendoza es otro vértice de articulación. Nótese que todos los caminos que conectan, por ejemplo, Calafate y Tucumán, pasan por Buenos Aires, y todos los caminos que conectan Salta y Calafate pasan por Mendoza.

En el grafo de conocidos de la subsección 1.3.1, Nacho es un vértice de articulación, y María también lo es. El grafo completo $K_n$ no tiene vértice de articulación. En los grafos estrella, el vértice central es de articulación.

Un grafo no dirigido conexo sin lazos y sin vértices de articulación se denomina BICONEXO. Una COMPONENTE BICONEXA de un grafo es un subgrafo biconexo maximal. En un grafo biconexo, existen por lo menos dos caminos entre cualquier par de vértices.

El grafo ciclo $C_n$ es biconexo (si se elimina cualquier vértice permanece conexo), mientras que un grafo lineal $L_n$ no es biconexo (los vértices con grado 2 son vértices de articulación).

Interpretando los vertices del grafo como centros y las aristas como líneas de comunicación, los vértices de articulación del grafo representan los puntos donde el sistema es más vulnerable. Si no existen vértices de articulación, es más probable que este sistema no se desconecte con la interrupción de un centro de comunicación.

Una arista tal que al eliminarla aumenta la cantidad de componentes conexas se denomina ARISTA DE CORTE. Puede probarse que

**Teorema 1.4.**
*Una arista es de corte si y sólo si no participa de ningún ciclo.* ♣

En el grafo de conocidos, las aristas {María, Luis} y {Marcos, Iván} son aristas de corte. En el grafo de rutas aéreas, las aristas {Salta, Mendoza} y {Buenos Aires, Montevideo} son aristas de corte.

En un grafo sin vértices aislados, un RECORRIDO EULERIANO es un recorrido que pasa por cada arista exactamente una vez. Un CIRCUITO EULERIANO es un recorrido euleriano cerrado.

No todo grafo admite un recorrido (circuito) euleriano.

***Ejemplo* 1.6**
*El grafo de la figura 1.4a sí admite un recorrido euleriano. Un recorrido de ese tipo es* $v_5 - v_1 - v_2 - v_3 - v_4 - v_1 - v_3$. *Puede comprobarse que este grafo no admite circuito euleriano.*

*El grafo de la figura 1.4b no admite un recorrido euleriano. Se puede intentar construir un recorrido que pase por todas las aristas, y se comprobará fácilmente que es imposible.*■

Es sencillo determinar si un grafo admite un recorrido (circuito) euleriano, a partir de los siguientes resultados que aseguran la existencia de tal recorrido (circuito):

**Proposición 1.5.**
*Un grafo (o multigrafo) sin vértices aislados admite un circuito euleriano si y sólo si es conexo y todos sus vértices tienen grado par.* ♢

**Proposición 1.6.**
*Un grafo (o multigrafo) sin vértices aislados admite un recorrido euleriano (y no circuito euleriano) si y sólo si es conexo y tiene exactamente dos vértices de grado impar.*

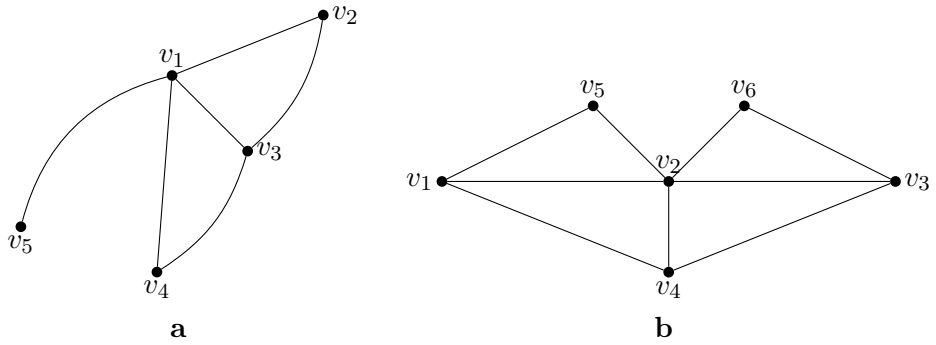



Figura 1.4: Grafos del ejemplo 1.6

El grafo de la figura 1.4a, que admite recorrido euleriano pero no circuito euleriano, tiene exactamente dos vértices de grado impar: $v_3$ y $v_5$.

En el grafo de la figura 1.4b no pudimos hallar un recorrido euleriano. Que no se haya podido encontrar, no implica que no existe; podría ser que la búsqueda no fue exhaustiva. Sin embargo, como tiene cuatro vértices de grafo impar, las proposiciones 1.6 y 1.5 permiten asegurar que no existe un recorrido (ni circuito) euleriano en ese grafo.

Para grafos dirigidos, recorrido (circuito) euleriano se define similarmente, requiriendo un recorrido (circuito) dirigido que pase por todas las aristas.

**Proposición 1.7.**
*Un grafo (o multigrafo) dirigido sin vértices aislados tiene un circuito euleriano si y sólo si $G$ es conexo y $gs(v_i) = ge(v_i)$ para cada vértice $v_i$.* ♢

***Ejemplo* 1.7**
*Considere el grafo de la figura 1.5. En cada vértice se verifica que el grado de entrada es igual al grado de salida. Por lo tanto, la proposición 1.7 asegura que se puede encontrar un circuito euleriano dirigido.*

*Éste podría ser* $v_1 - v_2 - v_6 - v_8 - v_5 - v_4 - v_1 - v_1 - v_7 - v_8 - v_9 - v_5 - v_3 - v_2 - v_7 - v_3 - v_7 - v_9 - v_6 - v_1$. ■

**Para pensar**.
¿Cuál debería ser la condición necesaria y suficiente para que un grafo o multigrafo dirigido admita recorrido euleriano?

Un algoritmo apropiado para hallar circuitos eulerianos se describe a continuación:

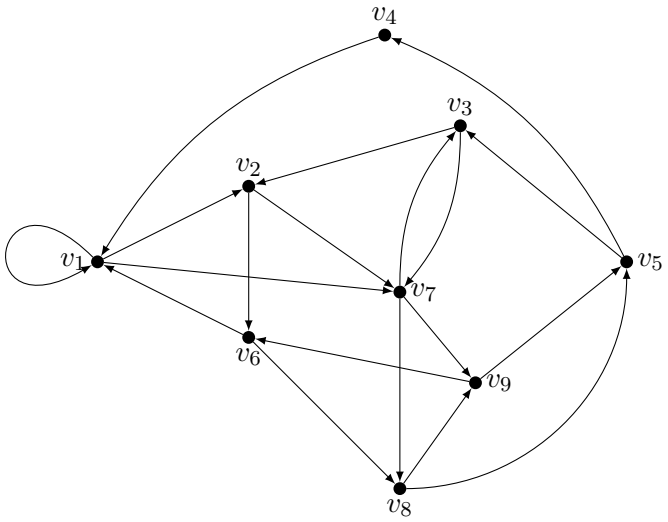



Figura 1.5: Grafo dirigido del ejemplo 1.7

**Algoritmo: circuitos eulerianos**
**Entrada**: Un grafo conexo con todos los vértices de grado par.
**Inicio**: Seleccionar un circuito $C$, comenzando en un vértice arbitrario. Sea $C = u_1 - u_2 - ... - u_k - u_1$
**Mientras** existan aristas en el grafo que no estén en C
Seleccionar una arista del grafo que no esté en C, que incida en algún vértice del circuito (sea $u_j$ tal vértice), y construir un subcircuito comenzando con esa arista, usando aristas que no estén en $C$. Sea $C_0 = w_1 - w_2 - ... - w_s$ el subcircuito hallado [1]. Actualizar $C = u_1 - u_2 - ... - u_{j-1} - w_1 - w_2 - ... - w_s - u_{j+1} - ... - u_k - u_1$
**Fin mientras**

Veamos un ejemplo de cómo se aplica el algoritmo al grafo de la figura 1.6a.

***Ejemplo* 1.8**
*Consideremos inicialmente el circuito:* $C = v_1 - v_3 - v_7 - v_8 - v_6 - v_2 - v_1$. *Este circuito se muestra en la figura 1.6b.*

*Como hay aristas no usadas (en línea punteada en la figura 1.6b), buscamos un circuito que use aristas punteadas, comenzando por una arista incidente en algún vértice de* $C$. *Una opción es comenzar con* $\{v_3, v_4\}$ *y obtener el circuito que se muestra en trazo grueso en la parte c de la figura. Entonces* $C_0 = v_3 - v_4 - v_5 - v_2 - v_9 - v_3$. *Luego, se actualiza* $C$*:*

$$C = v_1 - \underline{v_3 - v_4 - v_5 - v_2 - v_9} - v_3 - v_7 - v_8 - v_6 - v_2 - v_1$$

[1] Como $C_0$ es circuito, $w_1 = w_s$. Además, $w_1 = u_j$, porque así fue elegido

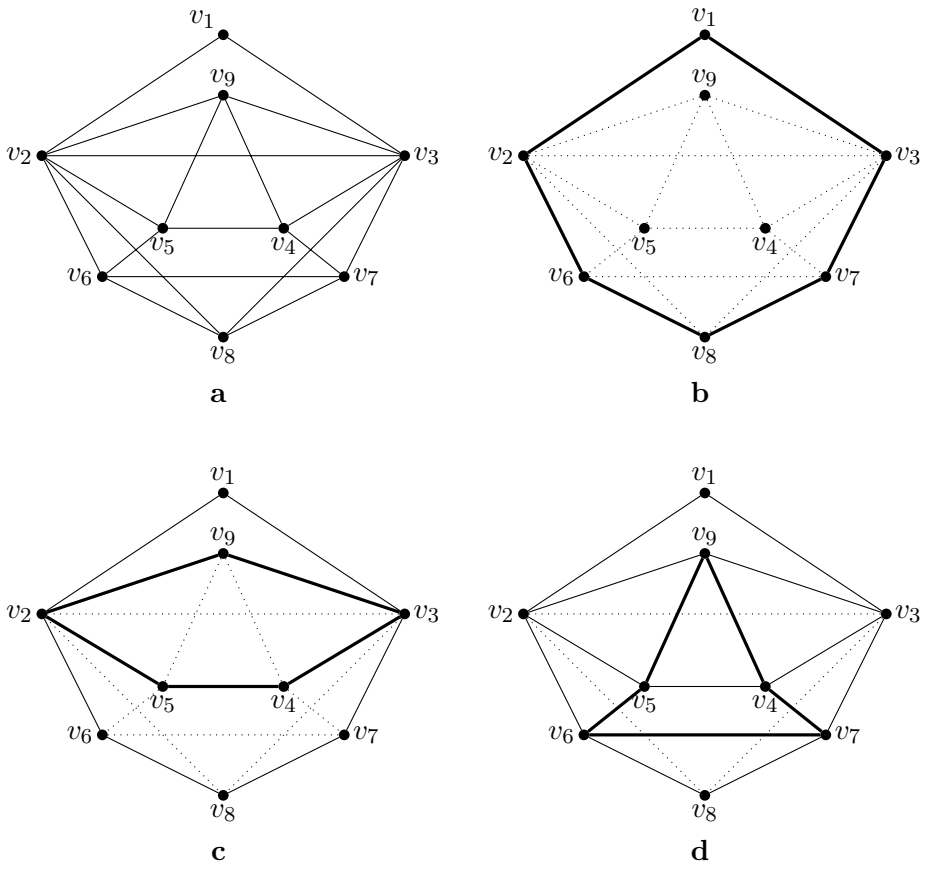



Figura 1.6: Ejemplo 1.8

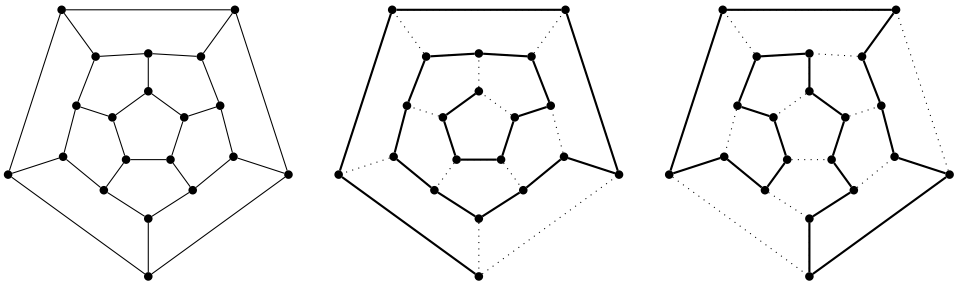



Figura 1.7: a- Grafo del ejemplo 1.9. b-Un camino hamiltoniano. c. Un ciclo hamiltoniano

*Aún quedan aristas sin usar. Entonces procedemos nuevamente: se busca un circuito que use aristas punteadas en la figura 1.6c. Por ejemplo, comenzando en* $v_9$ *con la arista* $\{v_9, v_4\}$ *obtenemos el nuevo circuito:* $C_0 = v_9 - v_4 - v_7 - v_6 - v_5 - v_9$ *que se muestra en trazo grueso en la figura 1.6d, y actualizamos el circuito* $C$*:*

$$C = v_1 - v_3 - v_4 - v_5 - v_2 - v_9 - \underline{v_4 - v_7 - v_6 - v_5 - v_9} - v_3 - v_7 - v_8 - v_6 - v_2 - v_1$$

*Quedan algunas aristas sin usar. Construimos un circuito que use esas aristas (punteadas en la figura 1.6d). Comenzando en* $v_2$*, es* $C_0 = v_2 - v_3 - v_8 - v_2$ *y actualizamos*

$$C = v_1 - v_3 - v_4 - v_5 - v_2 - v_9 - v_4 - v_7 - v_6 - v_5 - v_9 - v_3 - v_7 - v_8 - v_6 - v_2 - \underline{v_3 - v_8 - v_2} - v_1$$

*Ahora todas las aristas del grafo están en el circuito, por lo tanto, es un circuito euleriano.* ■

**Para pensar**.

a. ¿Cómo se aplicaría el algoritmo para hallar recorridos eulerianos en grafos que satisfacen las condiciones de la proposición 1.6?

b. ¿Qué condiciones debe cumplir un grafo para que un circuito euleriano en él sea un ciclo?

c. ¿Cuál es la salida del algoritmo si el grafo de entrada no es conexo?

Un CAMINO HAMILTONIANO es un camino que pasa por cada vértice exactamente una vez. Un CICLO HAMILTONIANO es un camino hamiltoniano cerrado.

***Ejemplo* 1.9**
*En la figura 1.7 se muestra un grafo, un camino hamiltoniano construido sobre ese grafo, y un ciclo hamiltoniano.* ■

Es claro que en un grafo de $n$ vértices, cualquier ciclo hamiltoniano tiene $n$ aristas, y un camino hamiltoniano tiene $n-1$ aristas. Necesariamente el camino hamiltoniano es un camino simple, ya que debe pasar sólo una vez por cada vértice.

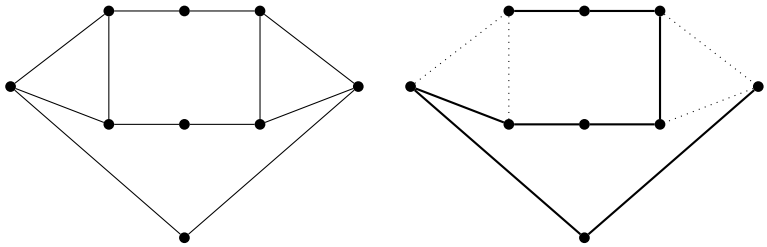


Figura 1.8: Grafo del ejemplo 1.10

***Ejemplo* 1.10**
*El grafo de la figura 1.8 no admite un ciclo hamiltoniano. Sin embargo, sí es posible trazar un camino hamiltoniano (marcado en la figura de la derecha).* ■

Dado un grafo, sobre todo si tiene gran tamaño, no es fácil determinar si admite o no un camino (ciclo) hamiltoniano. No existen condiciones necesarias y suficientes para la existencia de un camino (ciclo) hamiltoniano. Pero sí algunas condiciones necesarias, y algunas condiciones suficientes:

**Proposición 1.8.**
*Si un grafo admite un ciclo hamiltoniano, entonces es conexo, y* $gr(v_i) \geqslant 2$.

**Proposición 1.9.**
*Un grafo con* $n$ *vértices, y* $n \geqslant 3$, *que verifica que cada vértice tiene grado* $gr(v_i) \geqslant n/2$, *admite un ciclo hamiltoniano.*

***Ejemplo* 1.11**
*Se tienen las ocho secuencias de tres bits (ceros y unos). Se quiere ordenarlas de forma tal que cada secuencia difiera de la siguiente en sólo un bit; y que la última difiera de la primera en un sólo bit. Esto se conoce como* CÓDIGO GRAY.

*Un código Gray para secuencias de tres bits se puede obtener fácilmente si se ubican las ocho secuencias como etiquetas en los vértices de un grafo, cuyas aristas conectan vértices tal que sus etiquetas difieren en un sólo bit. El ordenamiento que se busca corresponde con un ciclo hamiltoniano en tal grafo. En la figura 1.9 se muestra el grafo descrito, llamado 3-cubo, y un ciclo hamiltoniano en él. Esto nos da el código Gray: 000- 001 - 101 - 111 - 011 - 010 - 110 - 100.*

*Imitando estas ideas para obtener un código Gray para secuencias de cuatro bits, se generaría un grafo de 16 vértices y 32 aristas, llamado 4-cubo. Hacer una representación gráfica de este grafo y buscar en él un ciclo hamiltoniano puede resultar engorroso. En lugar de eso, se puede razonar así: ordenemos primero las ocho secuencias de cuatro bits que comienzan con 0 (llamemos* $C_0$ *a este ordenamiento), luego las ocho secuencias de cuatro bits que comienzan con 1 (llamemos* $C_1$ *a esta lista). Posteriormente, se conectan estas dos ordenaciones, manteniendo la condición de que secuencias consecutivas difieran sólo en un bit.*

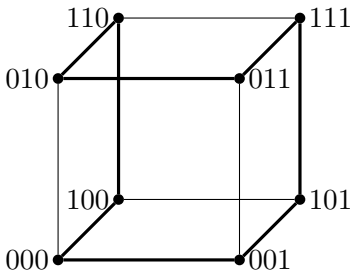



Figura 1.9: Grafo del ejemplo 1.11

*Ordenar las secuencias que comienzan con 0 (o con 1) es equivalente a ordenar las secuencias de tres bits. De lo hecho anteriormente, siguiendo el ciclo hamiltoniano en el 3-cubo se puede obtener el ordenamiento: $C_0$: 0000 - 0001 - 0101 - 0111 - 0011 - 0010 - 0110 - 0100, para las secuencias que comiencen con 0. Aquí se une con $C_1$. El final de $C_0$ debe enlazarse con el inicio de $C_1$, de forma que cambie sólo un bit, en este caso, el primer bit. Por lo tanto, $C_1$ debe comenzar con 1100. Además, el primer elemento de $C_0$ enlazarse con el último elemento de $C_1$. Así, $C_1$ debe terminar en 1000. Entonces, para establecer el orden en $C_1$ se sigue el ciclo del 3-cubo, pero en orden inverso al seguido para obtener $C_0$. Resulta: $C_1$: 1100 - 1110 - 1010 - 1011 - 1111 - 1101 - 1001 - 1000.*

*La secuencia obtenida al concatenar $C_0$ y $C_1$ es un ordenamiento de las secuencias de cuatro bits con las condiciones pedidas. Equivale a un ciclo hamiltoniano en el 4-cubo.*

*Este razonamiento se puede generalizar para secuencias de n bits, que equivalen a ciclos hamiltonianos en n-cubo. Así, se prueba que el grafo n-cubo, para cualquier n, admite un ciclo hamiltoniano.*

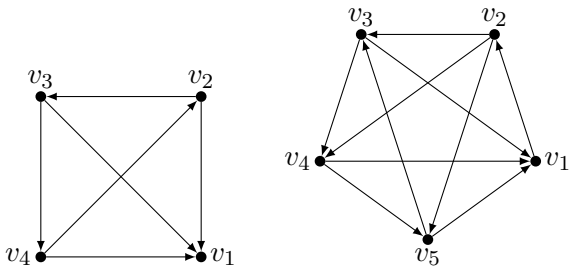
En grafos dirigidos, camino (ciclo) hamiltoniano se define como un camino (ciclo) dirigido que pasa por todos los vértices.

***Ejemplo* 1.12**
*Todo grafo torneo (grafo completo dirigido) admite un camino hamiltoniano. Por ejemplo, en el torneo de 4 vértices de la figura 1.10, un camino hamiltoniano es $v_2 - v_3 - v_4 - v_1$. Otro camino es $v_4 - v_2 - v_3 - v_1$. No es posible construir un ciclo hamiltoniano, ya que $gs(v_1) = 0$, y un ciclo debe poder salir de cada vértice.*

*En el torneo de 5 vértices de la figura 1.10 se tiene el camino hamiltoniano dirigido $v_1 - v_2 - v_5 - v_3 - v_4$, y otro camino posible es $v_4 - v_5 - v_1 - v_2 - v_3$. Este torneo también admite un ciclo hamiltoniano dirigido: $v_1 - v_2 - v_3 - v_4 - v_5 - v_1$.* ■

## 1.5. Representación de grafos

Hasta ahora hemos hablado de grafos relacionándolos directamente con sus representaciones gráficas. Sin embargo, a la hora de aplicar algoritmos para resolver problemas sobre grafos, implementados en una computadora, no son útiles las gráficas. Se requieren formas de representar grafos que se puedan introducir y almacenar en la memoria de una computadora. Hay varias opciones para representar un grafo o grafo dirigido.




**a** **b**

Figura 1.10: Torneos de 4 y 5 vértices

### 1.5.1. Listas de adyacencias

Dado un grafo no dirigido, para cada vértice $v_i$ se define una lista $L_{v_i}$ que contiene los vértices adyacentes a $v_i$.

Para grafos dirigidos, la lista de adyacencia $L_{v_i}$ de cada vértice, contiene los vértices adyacentes desde $v_i$.

### 1.5.2. Matriz de adyacencia

En un grafo con $n$ vértices, la matriz de adyacencia es una matriz cuadrada de $n \times n$, cuyos elementos se definen:

$$a_{ij} = \begin{cases} 1 & \text{si } v_i \text{ y } v_j \text{ son adyacentes} \\ 0 & \text{en otro caso} \end{cases}$$

Resulta una matriz binaria (sus elementos son 0 y 1) tal que aparece un 1 en la intersección de una fila y una columna correspondientes a vértices adyacentes.

Para grafos dirigidos, los elementos de la matriz de adyacencia se definen:

$$a_{ij} = \begin{cases} 1 & \text{si } v_j \text{ es adyacente desde } v_i \\ 0 & \text{en otro caso} \end{cases}$$

La matriz de adyacencia depende de un orden asignado a los vértices. Cambiando el orden, cambia la matriz (pero obviamente, la información contenida es la misma).

***Ejemplo* 1.13**
*La matriz de adyacencia del grafo de la figura 1.4b es:*

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

*La matriz de adyacencia del grafo dirigido de la subsección 1.3.3 (considerando el orden de los vértices: Atl Río, Sporting, Canes Unidos, Cebollitas, Sacachispas Club) es:*

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

La matriz de adyacencia de un grafo no dirigido es simétrica. Además, la suma de la fila $i$ (o columna $i$) da el grado del $i$-ésimo vértice (si no hay bucle en ese vértice). Los elementos en la diagonal representan bucles: el elemento $a_{ii}$ es 1 si y sólo si hay un bucle en el vértice $i$.

La matriz de adyacencia de un grafo dirigido no es simétrica en general. La suma de la fila $i$ da el grado de salida del vértice $i$-ésimo, y la suma de la columna $i$ da el grado de entrada del vértice $i$-ésimo.

Observe que la matriz de adyacencia $A$ de un grafo no dirigido tiene un 1 en la ubicación $ij$ si existe una arista entre los vértices $v_i$ y $v_j$. Es decir, tiene un 1 si existe un camino de longitud 1 entre esos vértices.

Al calcular $A^2 = A \cdot A$, el elemento $ij$ de la matriz resultante indica la cantidad de caminos de longitud 2 entre $v_i$ y $v_j$. En general, el elemento $ij$ de la matriz $A^m$ (producto de $m$ veces la matriz $A$) indica la cantidad de caminos de longitud $m$ entre los vértices $v_i$ y $v_j$. Un elemento cero en la posición $ij$ de $A^m$ indica que no existe un camino de longitud $m$ entre $v_i$ y $v_j$. Considere ahora la matriz $A^*$ definida como $A^* = \sum_{m=1}^{n-1} A^m$, donde $n$ es la cantidad de vértices del grafo. Como todas las componentes de las matrices son no negativas, un cero en la posición $ij$ de la matriz $A^*$ indica que todas las entradas en la posición $ij$ de las matrices $A$, $A^2$, $A^3$, ..., $A^{n-1}$ son cero. En tal caso, se deduce que no existe entre $v_i$ y $v_j$ un camino de longitud de 1, ni un camino de longitud 2, ni de longitud 3, ..., ni un camino de longitud $n-1$. Luego, se concluye que los vértices $v_i$ y $v_j$ no están conectados.

De esta manera, la matriz $A^*$ da información sobre la conexión del grafo. Un grafo es conexo si y sólo si la matriz $A^*$ tiene todos sus elementos no nulos .

***Ejemplo* 1.14**
*La matriz de adyacencia del grafo de la figura 1.11 es*

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

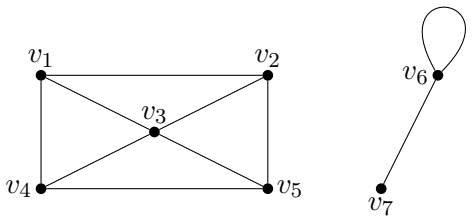



Figura 1.11: Grafo del ejemplo 1.14

*El cuadrado de esta matriz es:*

$$A \cdot A = A^2 = \begin{bmatrix} 3 & 1 & 2 & 1 & 3 & 0 & 0 \\ 1 & 3 & 2 & 3 & 1 & 0 & 0 \\ 2 & 2 & 4 & 2 & 2 & 0 & 0 \\ 1 & 3 & 2 & 3 & 1 & 0 & 0 \\ 3 & 1 & 2 & 1 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

*El elemento en la fila 2 y columna 4 es $a_{24} = 3$. Esto quiere decir que hay tres caminos de longitud 2 entre $v_2$ y $v_4$. Éstos son: $v_2 - v_1 - v_4$, $v_2 - v_3 - v_4$ y $v_2 - v_5 - v_4$. El elemento en la fila 6 y columna 6 es $a_{66} = 2$. Hay dos caminos de longitud 2 entre $v_6$ y $v_6$. Éstos son $v_6 - v_6 - v_6$ (este camino usa dos veces el bucle) y $v_6 - v_7 - v_6$. Los elementos 0 de la matriz indican que no existen caminos de longitud 2 entre los correspondientes vértices. Por ejemplo, entre $v_6$ y $v_1$, $v_7$ y $v_1$, etc. no existen caminos de longitud 2.*

*La matriz $A^*$ es*

$$A^* = \begin{bmatrix} 311 & 290 & 371 & 390 & 311 & 0 & 0 \\ 290 & 311 & 371 & 311 & 290 & 0 & 0 \\ 371 & 371 & 460 & 371 & 371 & 0 & 0 \\ 390 & 311 & 371 & 311 & 290 & 0 & 0 \\ 311 & 290 & 371 & 290 & 311 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 32 & 20 \\ 0 & 0 & 0 & 0 & 0 & 20 & 12 \end{bmatrix}$$

*Observando los ceros en esta matriz, de pueden determinar las componentes conexas del grafo. Se observa a partir de esta matriz que el vértice $v_6$ no está conectado con los vértices $v_1$, $v_2$, $v_3$, $v_4$ y $v_5$, mientras que sí está conectado al vértice $v_7$. El grafo tiene dos componentes conexas: una está formada por los vértices $v_1$, $v_2$, $v_3$, $v_4$ y $v_5$ y la otra componente conexa contiene los vértices $v_6$ y $v_7$.* ■

### 1.5.3. Matriz de incidencia

En un grafo con $n$ vértices y $m$ aristas, la matriz de incidencia es una matriz de $n \times m$, cuyos elementos se definen:

$$b_{ik} = \begin{cases} 1 & \text{si la arista } e_k \text{ es incidente en } v_i \\ 0 & \text{en otro caso} \end{cases}$$

Cada fila se asocia a un vértice, y cada columna a una arista. Los elementos 1 muestran la incidencia de aristas en vértices.

La matriz de incidencia de un grafo no dirigido es tal que en cada columna hay dos 1, si la columna no se corresponde con un bucle. En la columna correspondiente a un bucle, hay un solo 1. Es decir, la suma de los elementos de cada columna es siempre 2, salvo que la columna corresponda con un bucle. La suma de las filas es el grado de cada vértice, si el vértice no tiene un bucle. Un vértice aislado tiene asignada una fila de 0.

Para grafos dirigidos, los elementos de la matriz de incidencia se definen:

$$b_{ik} = \begin{cases} 1 & \text{si } v_i \text{ es el vértice final de la arista } e_k \\ -1 & \text{si } v_i \text{ es el vértice inicial de la arista } e_k \\ 0 & \text{en otro caso} \end{cases}$$

En este caso, no está bien definida la matriz si tiene un bucle dirigido. Por convención, se ignoran los bucles (es decir, no aparecen como columnas en la matriz de incidencia).

En cada columna de la matriz de incidencia de un grafo dirigido hay un 1 y un -1. La suma de cada fila da la diferencia entre el grado de entrada y el grado de salida del correspondiente vértice.

La matriz de incidencia depende de un orden asignado a los vértices y a las aristas. Cambiando el orden, cambia la matriz.

***Ejemplo* 1.15**
*La matriz de incidencia del grafo que se muestra en la figura 1.4b, con un determinado orden de las aristas, es:*

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

*La matriz de incidencia del grafo del torneo de la subsección 1.3.3 es*

$$A = \begin{bmatrix} 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

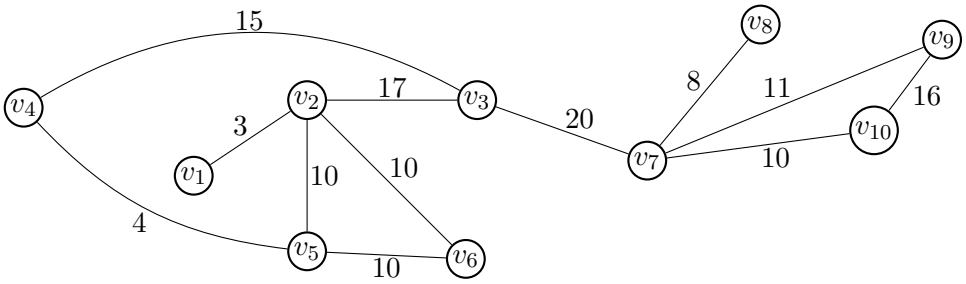



Figura 1.12: Grafo ponderado del ejemplo 1.16

## 1.6. Grafos ponderados

Un GRAFO PONDERADO, o GRAFO CON PESOS es un grafo cuyas aristas han sido ponderadas con pesos. El PESO de un grafo se define como la suma de los pesos de todas sus aristas. En las distintas aplicaciones, los pesos pueden representar distancias entre sitios representados por nodos, flujo máximo en conexiones, tiempo de conexión, costos de transporte, etc.

En un grafo ponderado, a cada camino se le asigna un peso, que es la suma de los pesos de las aristas que intervienen en el camino. Un problema interesante es, dado un grafo ponderado, hallar un camino entre dos vértices que tenga peso mínimo. Dicho camino se denomina CAMINO MÍNIMO. En un grafo ponderado, se llama DISTANCIA entre dos vértices al peso del camino mínimo que conecta esos vértices.

***Ejemplo* 1.16**
*Considere el grafo ponderado de la figura 1.12. El peso del grafo es 134. El camino* $v_1 - v_2 - v_6 - v_5 - v_4 - v_3 - v_7$ *tiene peso 62. La distancia entre* $v_1$ *y* $v_7$ *es 40 (el peso del camino mínimo* $v_1 - v_2 - v_3 - v_7$*). La distancia de* $v_5$ *a* $v_3$ *es 19 (el peso del camino mínimo* $v_5 - v_4 - v_3$*).* ■

Dado un grafo ponderado de $n$ vértices, los pesos de las aristas pueden darse en una MATRIZ DE PESOS $W$ de $n \times n$, tal que elemento $w_{ij}$ sea el peso de la arista que une $v_i$ y $v_j$.

La matriz de pesos del grafo de la figura 1.12 es

$$W = \begin{bmatrix} - & 3 & - & - & - & - & - & - & - & - \\ 3 & - & 17 & - & 10 & 10 & - & - & - & - \\ - & 17 & - & 15 & - & - & 20 & - & - & - \\ - & - & 15 & - & 4 & - & - & - & - & - \\ - & 10 & - & 4 & - & 10 & - & - & - & - \\ - & 10 & - & - & 10 & - & - & - & - & - \\ - & - & 20 & - & - & - & - & 8 & 11 & 10 \\ - & - & - & - & - & - & 8 & - & - & - \\ - & - & - & - & - & - & 11 & - & - & 16 \\ - & - & - & - & - & - & 10 & - & 16 & - \end{bmatrix}$$

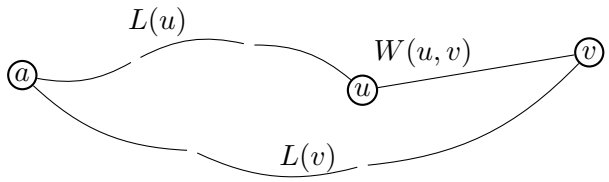
En muchas aplicaciones interesa determinar la longitud del camino más corto entre dos nodos. Por ejemplo, en el problema de rutas aéreas, ponderando las rutas según su distancia en kilómetros, se quiere conocer el camino más corto entre dos ciudades. En otro ejemplo, considérese el problema de una red de terminales, donde cada conexión entre terminales se pondera con el tiempo que insumen los datos en transmitirse de una terminal a la otra. Aquí interesa hallar la ruta entre dos terminales que haga que los datos se transfieran en el menor tiempo posible.

A continuación se describe un algoritmo que realiza un procedimiento para hallar la longitud del camino mínimo entre dos vértices, $a$ y $z$.

El algoritmo procede iterativamente, etiquetando los vértices del grafo con la distancia del camino más corto entre $a$ y ese vértice, tal que el camino use vértices en cierto conjunto $S$. La etiqueta del nodo $v$ se denota $L(v)$. El conjunto $S$ y las etiquetas se van actualizando en cada iteración.

Inicialmente el conjunto $S$ contiene sólo el vértice $a$, y la etiqueta $L(v)$ para cada $v$ es el peso de la arista $\{a, v\}$. Se selecciona el vértice $u$ (distinto de $a$) con mínima etiqueta, y se agrega este vértice en $S$. Para este vértice, la longitud del camino mínimo de $a$ hasta él es el valor de su etiqueta.

Luego se actualizan las demás etiquetas. Recordemos que en cada iteración, la etiqueta de cada vértice indica la longitud del camino mínimo que usa vértices que están en $S$. Para la actualización, se compara el valor de la etiqueta actual $L(v)$ de cada vértice $v$ con el valor de sumar la etiqueta de $u$ más el peso de la arista de $u$ a $v$, $W(u, v)$. Si esta suma es menor que $L(v)$, indica que existe un camino de $a$ a $v$, pasando por $u$, que tiene menor longitud que $L(v)$. Entonces, se cambia la etiqueta de $v$ por $L(u) + W(u, v)$ (véase el esquema debajo). Así, la etiqueta de $v$ cambia de valor si $L(v) > L(u) + W(u, v)$, y no cambia si $L(v) \leqslant L(u) + W(u, v)$. De modo que la actualización puede realizarse con $L(v) = min\{L(v), L(u) + W(u, v)\}$ .




Estos pasos se repiten: se selecciona un vértice de mínima etiqueta entre los que no están en $S$, se lo agrega a $S$ y se actualizan las etiquetas, hasta que $z$ esté en $S$. En este punto, la etiqueta de $z$ indica la longitud del camino mínimo entre $a$ y $z$.

Los pasos se describen en pseudocódigo a continuación.

**Algoritmo: longitud de camino mínimo**
**Entrada**: Un grafo conexo, con pesos positivos, y dos vértices distinguidos, $a$ y $z$. Sea $W(u, v)$ el peso de la arista $\{u, v\}$ (que será $\infty$ si la arista no existe en el grafo).
**Salida**: La longitud del camino mínimo de $a$ a $z$.
**Inicio**: Inicializar todas las etiquetas $L(v) := \infty$, $L(a) := 0$, $S := \emptyset$.
**Mientras** $z \notin S$
Seleccionar, entre todos los vértices que no estén en $S$, un vértice $u$ con mínima etiqueta $L(u)$. Agregarlo a $S$, $S := S \bigcup \{u\}$.
Para todos los vértices $v$ que no estén en S, actualizar las etiquetas: $L(v) := min\{L(v), L(u) + W(u, v)\}$.
**Fin**

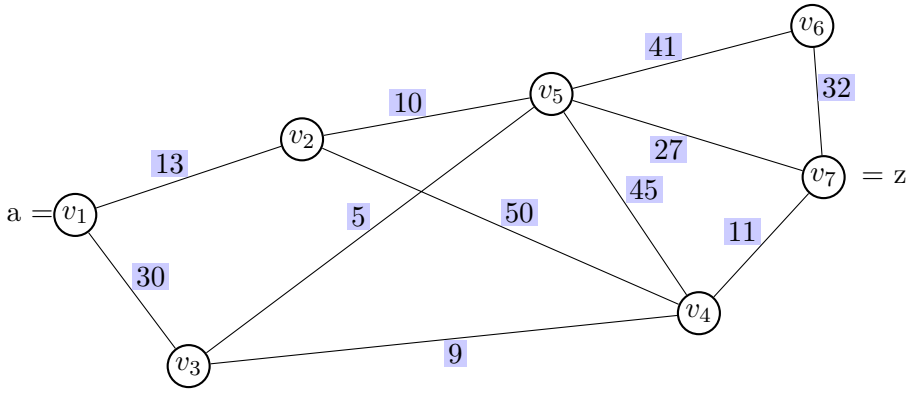
Antes de ver un ejemplo de aplicación del algoritmo, algunos comentarios.

En el inicio del algoritmo se indica inicializar las etiquetas con $\infty$. Esto en la práctica se lleva a cabo usando un número suficientemente alto para representar el infinito. Por ejemplo, un valor adecuado puede ser $n$ veces el doble del peso máximo de las aristas.

El algoritmo descrito es una simplificación del algoritmo conocido propuesto por Dijsktra, y que lleva su nombre. La versión completa de este algoritmo permite obtener la longitud del camino más corto, y el camino mínimo entre $a$ y todos los demás vértices. Si algunos pesos de las aristas son negativos el algoritmo de Dijsktra no se aplica. En este caso el problema se resuelve con el algoritmo de Bellman-Ford.

Otro algoritmo conocido para el problema del camino mínimo se debe a Floyd y Warshall (Algoritmo de Floyd-Warshall), que obtiene el camino mínimo entre todos los vértices.

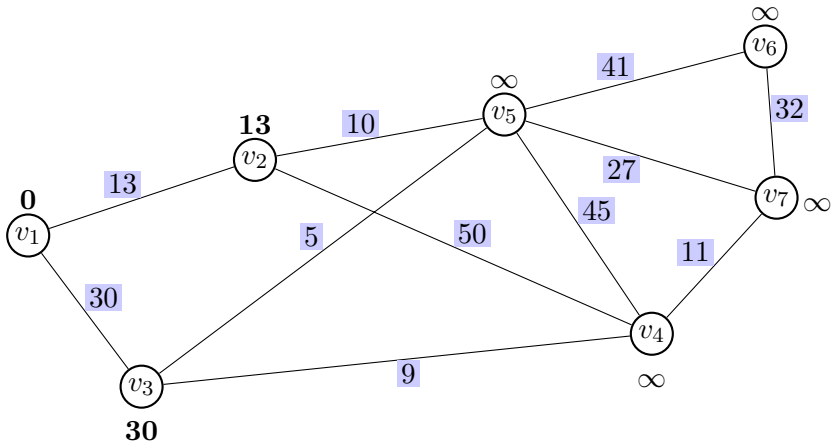
Veamos ahora sí cómo se aplica el algoritmo descrito al grafo que se muestra a continuación. Se quiere hallar la longitud del camino mínimo entre $v_1$ y $v_7$.




Se comienza etiquetando todos los nodos con $\infty$, excepto $v_1$, que etiquetamos con 0, e inicializamos $S = \emptyset$.

Dado que el vértice $z$ no está en $S$, procedemos: seleccionamos el vértice de menor etiqueta, que es $v_1$. Se agrega: $S = \{v_1\}$. Ahora, se actualizan las etiquetas de todos los vértices que no están en $S$. Como $L(v_2) = \infty$, $W(v_1, v_2) = 13$, la etiqueta de $v_2$ resulta: $L(v_2) = min\{L(v_2), L(v_1) + W(v_1, v_2)\} = min\{\infty, 0 + 13\} = 13$. Del mismo modo, la etiqueta de $v_3$ toma el valor: $L(v_3) = min\{L(v_3), L(v_1) + W(v_1, v_3)\} = min\{\infty, 0 + 30\} =$

30. Todos los otros vértices son tales que $W(v_1, v_j) = \infty$, por lo tanto, no van a cambiar sus etiquetas. El nuevo etiquetado se muestra en la figura siguiente.




Como $z$ no está en $S$, proseguimos. De los vértices que no están en $S$, quien tiene menor etiqueta es $v_2$. Entonces, $S = \{v_1, v_2\}$ y se actualizan las etiquetas de los vértices que no están en $S$:
$L(v_3) = min\{L(v_3), L(v_2) + W(v_2, v_3)\} = min\{30, 13 + \infty\} = 30$
$L(v_4) = min\{L(v_4), L(v_2) + W(v_2, v_4)\} = min\{\infty, 13 + 50\} = 63$
$L(v_5) = min\{L(v_5), L(v_2) + W(v_2, v_5)\} = min\{\infty, 13 + 10\} = 23$
$L(v_6) = min\{L(v_6), L(v_2) + W(v_2, v_6)\} = min\{\infty, 13 + \infty\} = \infty$
$L(v_7) = min\{L(v_7), L(v_2) + W(v_2, v_7)\} = min\{\infty, 13 + \infty\} = \infty$

Como $z$ no está en $S$, proseguimos. De los vértices que no están en $S$, el de etiqueta menor es $v_5$. Luego, $S = \{v_1, v_2, v_5\}$ y se actualizan las etiquetas de los vértices que no están en $S$:
$L(v_3) = min\{L(v_3), L(v_5) + W(v_5, v_3)\} = min\{30, 23 + 5\} = 28$
$L(v_4) = min\{L(v_4), L(v_5) + W(v_5, v_4)\} = min\{63, 23 + 45\} = 63$
$L(v_6) = min\{L(v_6), L(v_5) + W(v_5, v_6)\} = min\{\infty, 23 + 41\} = 64$
$L(v_7) = min\{L(v_7), L(v_5) + W(v_5, v_7)\} = min\{\infty, 23 + 27\} = 50$

El nuevo vértice a agregar es $v_3$, así $S = \{v_1, v_2, v_5, v_3\}$ y las nuevas etiquetas son:
$L(v_4) = min\{L(v_4), L(v_3) + W(v_3, v_4)\} = min\{63, 28 + 9\} = 37$
$L(v_6) = min\{L(v_6), L(v_3) + W(v_3, v_6)\} = min\{64, 28 + \infty\} = 64$
$L(v_7) = min\{L(v_7), L(v_3) + W(v_3, v_7)\} = min\{50, 28 + \infty\} = 50$

A continuación, consideramos $v_4$: $S = \{v_1, v_2, v_5, v_3, v_4\}$ y las etiquetas:
$L(v_6) = min\{L(v_6), L(v_4) + W(v_4, v_6)\} = min\{64, 37 + \infty\} = 64$
$L(v_7) = min\{L(v_7), L(v_4) + W(v_4, v_7)\} = min\{50, 37 + 11\} = 48$

El siguiente vértice con menor etiqueta es $v_7$: $S = \{v_1, v_2, v_5, v_3, v_4, v_7\}$ y se actualiza la etiqueta:
$L(v_6) = min\{L(v_6), L(v_7) + W(v_7, v_6)\} = min\{64, 48 + 32\} = 64$

Como el vértice $z$ está en $S$, termina el algoritmo. La etiqueta del vértice $z = v_7$ indica que la longitud del camino más corto de $a$ a $z$ es 48.

**Para pensar**

a. En términos prácticos, el último ciclo "mientras"del algoritmo se ejecuta inútilmente, porque allí no cambia la etiqueta de $z$. ¿Cómo se podría modificar el algoritmo para evitar esa ejecución inútil?

b. Al finalizar el algoritmo, no sólo se tiene la distancia más corta de $a$ a $z$, sino también la distancia más corta de $a$ a todos los vértices en $S$. ¿Cómo se debería modificar el algoritmo para que encuentre las distancias más cortas entre el vértice $a$ y todos los demás vértices del grafo?

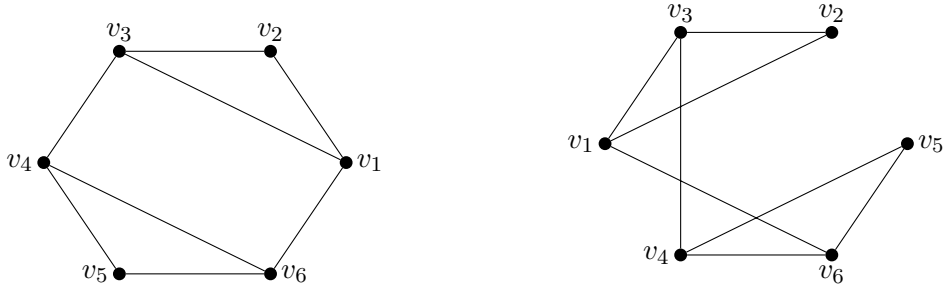
c. Tal como fue presentado el algoritmo, al finalizar se conoce la distancia de $a$ a $z$, pero no el camino que efectiviza esa distancia mínima. ¿Cómo podría ser modificado para que adicionalmente indique el camino más corto de $a$ a $z$?

d. ¿Cómo debería modificarse el algoritmo para ser aplicado a grafos dirigidos, con el fin de encontrar caminos dirigidos de longitud mínima?

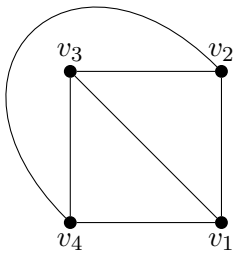
## 1.7. Grafos planos

Se dice que un grafo es PLANO si se lo puede representar gráficamente de modo tal que las aristas no se corten entre sí en puntos que no son los vértices. Tal representación se denomina representación plana o planar.

Obviamente, como todo grafo tiene muchas posibles formas de graficarse, un grafo plano podría tener representaciones con cortes de aristas. Por ejemplo, el grafo plano que se muestra a continuación se representa en forma planar a la izquierda, y en forma no planar a la derecha.

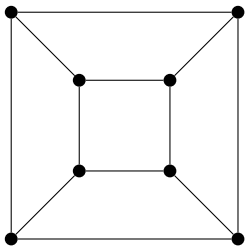



***Ejemplo* 1.17**
*El grafo $K_4$ es plano, como puede verse en la representación plana siguiente:*




■

***Ejemplo* 1.18**
*El grafo cubo $Q_3$ es plano, como se aprecia en la siguiente representación*



■

Los grafos $K_5$ y $K_{3,3}$ no son planos. Más adelante se darán justificaciones de esta afirmación.

La planaridad de grafos es importante en muchas aplicaciones prácticas. Por ejemplo, si se quiere conectar un conjunto de usuarios con un conjunto de servicios, utilizando cañerías subterráneas, estando todas las conexiones al mismo nivel, se debe pensar en un grafo plano. La situación con tres usuarios y tres servicios se puede estudiar mediante el grafo $K_{3,3}$. Como $K_{3,3}$ no es plano, no se pueden construir las conexiones al mismo nivel.

En el campo de la electrónica, un circuito electrónico es un conjunto de componentes interconectados con hilos conductores. Cuando se diseña un circuito electrónico debe atenderse a que las conexiones no se crucen. Nuevamente aquí es importante el concepto de grafo plano. Si el circuito que se quiere implementar no tiene una representación plana, será necesaria la utilización de puentes (cables aislados superpuestos).

La representación plana de un grafo conexo de $V$ vértices y $A$ aristas determina un conjunto de $R$ regiones. Entre estas cantidades se da la Fórmula de Euler:

**Proposición 1.10.**
*Toda representación plana de un grafo plano conexo con $V$ vértices y $A$ aristas divide el plano en $R = A - V + 2$ regiones.* ♢

***Ejemplo* 1.19**
*En el grafo $Q_3$, $V = 8$, $A = 12$. Toda representación plana divide el plano en $R = 12 - 8 + 2 = 6$ regiones. Esto puede comprobarse en el ejemplo 1.18.* ■

***Ejemplo* 1.20**
*Un grafo ciclo con $V$ vértices tiene $A = V$ aristas. Entonces, una representación planar divide el plano en $R = V - V + 2 = 2$ regiones.* ■

***Ejemplo* 1.21**
*Todo grafo estrella $S_n$ tiene $V = n + 1$ vértices, $A = n$ aristas. Entonces, una representación plana de este grafo divide el plano en $R = n - (n + 1) + 2 = 1$ región.* ■

Como corolarios de la proposición anterior, se pueden enunciar:

**Corolario 1.11.**
*En un grafo plano conexo sin bucles con $V \geqslant 3$ vértices y $A$ aristas, se verifica la desigualdad $A \leqslant 3V - 6$.* ♢

Esto implica, que si en un grafo conexo sin bucles, se verifica $A > 3V - 6$, entonces el grafo no es plano. Pero no implica que si se verifica $A \leqslant 3V - 6$, entonces el grafo debe ser plano.

***Ejemplo* 1.22**
*Apliquemos este corolario al grafo completo de 5 vértices. En este caso, $V = 5 \geqslant 3$, $A = 10$, es conexo y no tiene bubles. Además, $A = 10 > 3V - 6 = 15 - 6 = 9$. Por lo tanto, no puede ser plano.* ■

***Ejemplo* 1.23**
*El grafo bipartito $K_{3,3}$ tiene $A = 9$ aristas y $V = 6$ vértices. Y se verifica $A \leqslant 3V - 6$, ya que $9 \leqslant 3. \cdot 6 - 6 = 12$. Pero el grafo no es plano.* ■

El siguiente corolario se aplica al caso más restrictivo de grafos en los que no existan ciclos de longitud 3.

**Corolario 1.12.**
*En un grafo plano conexo sin bucles con $V \geqslant 3$ vértices y $A$ aristas, tal que no contenga ciclos de longitud 3, se verifica la desigualdad $A \leqslant 2V - 4$.* ♢

Este corolario implica que si en un grafo conexo sin bucles y sin ciclos de longitud 3 se verifica $A > 2V - 4$, entonces el grafo no es plano.

***Ejemplo* 1.24**
*El grafo bipartito $K_{3,3}$ no tiene ciclos de longitud 3. Además, es conexo y no tiene bucles. La cantidad de aristas es $A = 9$ y la cantidad de vértices es $V = 6$. Se cumple la desigualdad $A = 9 > 2V - 4 = 12 - 4 = 8$. Por el corolario anterior, este grafo no puede ser plano.* ■

Finalmente, el teorema siguiente da condiciones necesarias y suficientes para determinar la planaridad de un grafo.

**Teorema 1.13.**
*Teorema de Kuratowski.*
*Un grafo es plano si y sólo si no contiene un subgrafo homeomorfo a $K_5$ o a $K_{3,3}$.* ♣

Dos grafos son homeomorfos si uno se puede obtener del otro mediante subdivisiones elementales. Una subdivisión elemental consiste en reemplazar una arista $\{u_i, u_j\}$ por dos aristas y un vértice nuevo $v$, es decir, se agregan las aristas $\{u_i, v\}$ y $\{v, u_j\}$. En la figura 1.13 se observan dos grafos homeomorfos.

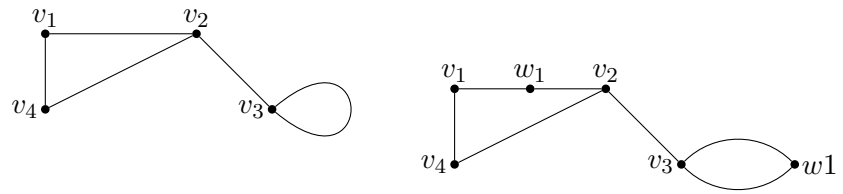



Figura 1.13: Grafos homeomorfos

***Ejemplo* 1.25**
*El grafo de la figura 1.14a se conoce como grafo de Petersen. En la parte b de la figura se observa un subgrafo del grafo de Petersen homeomorfo a $K_{3,3}$. Mediante cuatro subdivisiones elementales del grafo de la parte c de la figura se obtiene el subgrafo mostrado en la parte b.*

*Entonces, el teorema de Kuratowski asegura que el grafo de Petersen no es plano.*

## 1.8. Árboles

Un grafo se denomina ÁRBOL si es conexo y no contiene ciclos. Como un bucle es un ciclo de longitud 1, un árbol no tiene bucles.

Hay otras caracterizaciones de árboles, y se dan en el siguiente

**Teorema 1.14.**
*Las siguientes afirmaciones son equivalentes:*

- *Un grafo $G$ es un árbol.*
- *$G$ es un grafo conexo tal que si se elimina una arista, queda un grafo no conexo.*
- *Entre dos vértices cualesquiera en $G$, existe un único camino.*
- *$G$ no tiene ciclos, pero cuando se agrega una arista nueva, se genera un ciclo.*
- *Es conexo y $|V| = |E| + 1$.* ♣

***Ejemplo* 1.26**
*Consideremos el grafo de la figura siguiente. Nótese que es conexo, y tiene 8 nodos y 7 aristas. Entonces es un árbol. Además, al eliminar cualquier arista (por ejemplo, la arista $\{v_4, v_7\}$, el grafo resulta disconexo, ya que no existe camino que una $v_5$ con $v_1$. Una situación similar aparece si se elimina cualquiera de las otras aristas. Por otra parte, si agregamos una arista nueva, por ejemplo, $\{v_5, v_6\}$, aparece un ciclo: $v_4 - v_5 - v_6 - v_1 - v_8 - v_7 - v_4$. Del mismo modo, intentando agregar cualquier otra arista, se crea un ciclo.*■

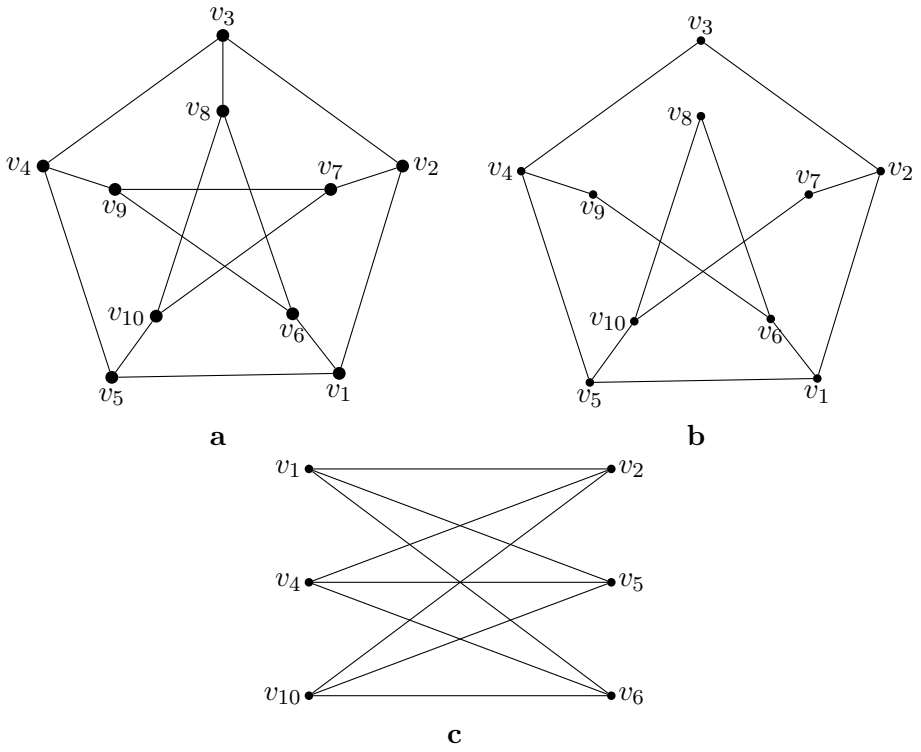
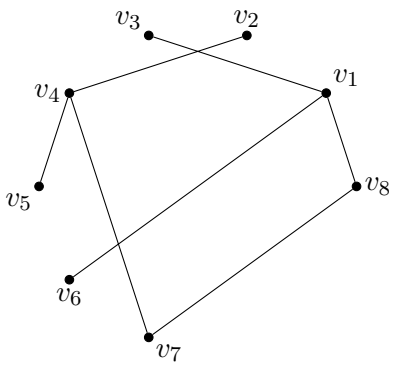
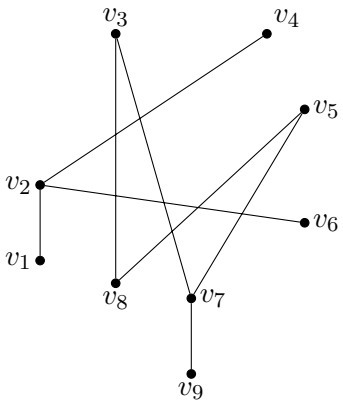



Figura 1.14: Grafo de Petersen y subgrafo homeomorfo a $K_{3,3}$




Grafo del ejemplo 1.26




Grafo del ejemplo 1.27

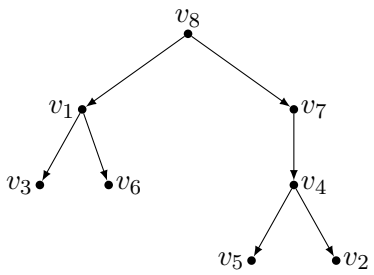
***Ejemplo* 1.27**
*Consideremos el grafo de la derecha en la figura anterior. Ese grafo tiene 9 nodos y 8 aristas. Sin embargo no es árbol. Por un lado porque tiene un ciclo:* $v_7 - v_5 - v_8 - v_3 - v_7$. *Pero además, es disconexo. Las componentes conexas tienen vértices* $\{v_4, v_2, v_1, v_6\}$ *y* $\{v_3, v_5, v_7, v_8, v_9\}$ *respectivamente.*

**Para pensar**
¿Cuáles son los vértices de articulación de un árbol? Si un árbol tiene V vértices, cuál es la cantidad máxima de vértices de articulación? ¿Cuál es la cantidad mínima de vértices de articulación?

Un grafo no conexo sin ciclos de llama BOSQUE. Cada componente conexa de un bosque es un árbol.

Un grafo dirigido es un árbol si el grafo no dirigido resultante de ignorar la dirección de las aristas es un árbol. Si además, existe un nodo $v_0$, llamado raíz, tal que $ge(v_0) = 0$, y $ge(v_i) = 1$ para todos los otros nodos, se denomina ÁRBOL CON RAÍZ. Cualquier árbol no dirigido puede considerarse como árbol con raíz, determinando uno de los vértices como raíz, e imponiendo direcciones a las aristas de modo que se alejen del vértice raíz.

Por ejemplo, en el árbol del ejemplo 1.26, tomando como raíz el vértice $v_8$, resulta el árbol con raíz que muestra en el siguiente gráfico.

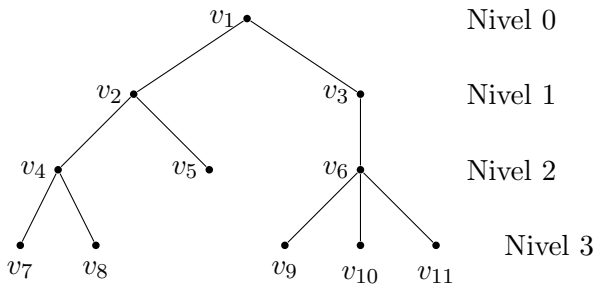



En un árbol con raíz, los nodos con $gs(v_i) = 0$ se denominan HOJAS. Todos los demás nodos se denominan NODOS DE RAMIFICACIÓN.

Si existe una arista dirigida $(v_i, v_j)$, entonces se dice que $v_i$ es el PADRE de $v_j$, y que $v_j$ es el HIJO de $v_i$.

Un árbol con raíz puede dividirse en niveles. En el nivel 0 se encuentra la raíz. En el nivel 1 los hijos de la raíz, en el nivel 2 todos los hijos de los nodos del nivel 1. En general, en el nivel $k$ se ubican los hijos de los nodos ubicados en el nivel $k-1$. La ALTURA de un árbol con raíz es el mayor nivel con nodos.

Los árboles con raíz suelen ser dibujados como el siguiente grafo:




Con esta ubicación de los nodos, no es necesario indicar la dirección de las aristas, ya que queda sobreentendido (se dirigen hacia abajo). En el árbol de la figura anterior, la raíz es $v_1$, las hojas son $v_5$, $v_7$, $v_8$, $v_9$, $v_{10}$ y $v_{11}$.

Un árbol con raíz se dice BINARIO si todos los nodos de ramificación tienen a lo sumo 2 hijos. En forma general, un árbol se dice M-ARIO si todos los nodos de ramificación tienen a lo sumo $m$ hijos. Un árbol M-ARIO COMPLETO es aquel tal que cada nodo tiene $m$ o ningún hijo. Un árbol con raíz de altura $h$ se dice BALANCEADO si todas sus hojas están en los niveles $h$ o $h-1$.

El árbol de la figura anterior es 3-ario (no completo). La figura 1.15 muestra árboles binarios.

**Teorema 1.15.**
*Un árbol m-ario completo con $R$ nodos de ramificación tiene $V = m \cdot R + 1$ vértices y $H = (m-1) \cdot R + 1$ hojas.* ♣

**Teorema 1.16.**
*Un árbol m-ario de altura $L$ tiene a lo sumo $m^L$ hojas.* ♣

**Corolario 1.17.**
*Si un árbol m-ario de altura $L$ tiene $H$ hojas, entonces $L \geqslant \lceil log_m H \rceil$. Si el árbol es balanceado completo, entonces $L = \lceil log_m H \rceil$.* ♢

***Ejemplo* 1.28**
*Considere un árbol binario con $V$ vértices. La altura máxima que puede tener es $V-1$, en el caso en que cada vértice de ramificación tenga un solo hijo, y el árbol tenga una sola hoja (resulta un grafo lineal). Y la altura mínima que puede alcanzar se obtiene si es completo balanceado. En tal caso, por el teorema 1.15, la cantidad de nodos de ramificación*

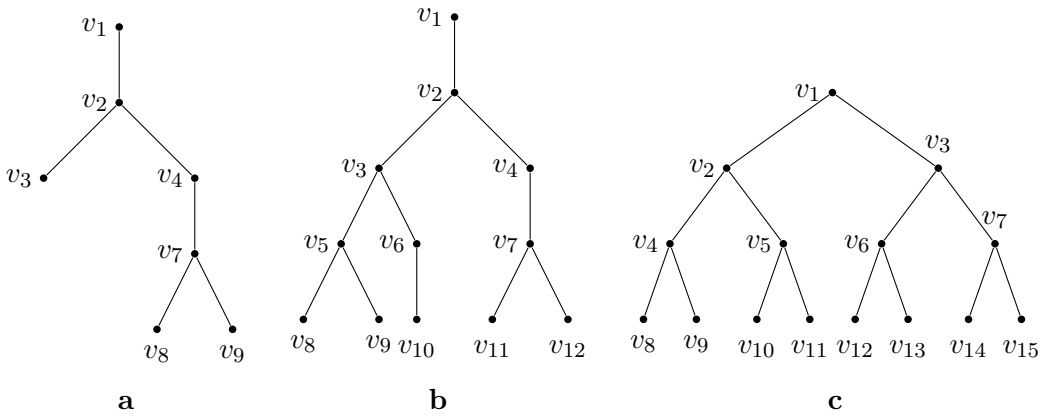



Figura 1.15: **a-** Árbol binario de altura 4, no balanceado. **b-** Árbol binario balanceado de altura 4. **c-**Árbol binario completo balanceado de altura 3.

*es* $R = (V-1)/m = (V-1)/2$ *(en árboles binarios, m=2). Y la cantidad de hojas es* $H = (m-1)\cdot R + 1 = (V-1)/2 + 1 = (V+1)/2$. *Ahora, por el último corolario, la altura del árbol completo balanceado es* $\lceil log_m H \rceil = \lceil log_2((V+1)/2) \rceil = \lceil log_2(V+1) - 1 \rceil$.

***Ejemplo* 1.29**

*En un árbol ternario completo con 34 vértices de ramificación (R=34), la cantidad total de vértices es* $V = 3\cdot 34 + 1 = 103$*, y la cantidad de hojas es el total de vértices menos los vértices de ramificación:* $H = V - R = 103 - 34 = 69$. ■

***Ejemplo* 1.30**

*Suponga que en una red social con 205 miembros se establece un sistema de comunicación donde un miembro determinado (emisor) envía un mensaje a otros cuatro miembros. Éstos retransmiten el mensaje a otros cuatro miembros, y así cada uno que recibe un mensaje lo retransmite a cuatro miembros, hasta que los 205 miembros hayan recibido el mensaje. La situacion se puede representar mediante un árbol cuaternario (o 4-ario) completo, donde la raíz es el emisor. El árbol debe tener 205 vértices, (205-1)/4=51 nodos de ramificación y 205-51=154 hojas. Los nodos de ramificación representan los miembros que transmiten el mensaje.*

*Podríamos preguntarnos: ¿cuántas veces se envía el mensaje? Para responder esta pregunta no es necesario aplicar lo aprendido sobre árboles. Cada vez que se transmite el mensaje, le llega a un nuevo miembro. Como todos los miembros, excepto el emisor, lo reciben individualmente, se transmite 204 veces.*

*Por otro lado, ahora sí mirando la estructura de árbol, cada transmisión de mensaje se asocia a una arista de éste. Y se sabe (visto en el inicio de esta sección) que la cantidad de aristas de un árbol es uno menos que la cantidad de vértices. Como el árbol tiene 205 vértices, tiene 204 aristas, por lo tanto, llegamos a la misma conclusión (se transmite 204 veces) con otro argumento.*

*Una pregunta más interesante es: si el tiempo que pasa entre que un miembro recibe el mensaje y lo retransmite es una hora, ¿cuál es el tiempo mínimo necesario para que todos*

*reciban el mensaje? Se está preguntando cuál es la altura mínima del árbol cuaternario. Esta altura mínima se alcanza cuando es un árbol balanceado. En tal caso, la altura es* $h = \lceil log_4 154 \rceil = \lceil 3{,}63 \rceil = 4$. *Es decir, se requieren, como mínimo, 4 horas para que todos reciban el mensaje.* ■

***Ejemplo* 1.31**
*Considere que se tienen 65 computadoras que deben conectarse a la red eléctrica mediante cables de tres salidas. Se desea saber cuántos cables de este tipo se necesitan. Un cable irá conectado a la toma de energía. Para hacer prolongaciones, un cable se puede conectar a una salida de otro cable. Así hechas las conexiones, resultará una estructura de árbol con raíz. La raíz será la toma de energía, las hojas serán las computadores, mientras que los nodos de ramificación serán las conexiones (la toma de energía o las salidas de los cables). Cada nodo de ramificación puede tener hasta tres ramas, por lo que es un árbol ternario. Tenemos:* $H = 65$ *hojas,* $m = 3$. *Por el teorema 1.15,* $65 = (3-1) \cdot R + 1$, *por lo que se tienen* $R = (65-1)/2 = 32$ *vértices de ramificación. O sea, se necesitan 32 cables.* ■

***Ejemplo* 1.32**
*En una situación como el ejemplo anterior, suponga que se tienen 40 cables de tres salidas. ¿Cuál es la máxima cantidad de computadoras que se pueden conectar?*

*De acuerdo al teorema 1.15, la cantidad de computadores (hojas) es* $H = (3-1) \cdot 40 = 80$.

*Ahora, si se tiene cantidad ilimitada de cables, pero se impone la restricción de que no pueden colocarse más de cuatro cables en serie, ¿cuántas computadoras como máximo se pueden conectar?*

*En este caso, el esquema de conexiones será un árbol ternario con raíz, de altura menor o igual a 4. Luego, por el teorema 1.16, para un árbol ternario de altura 4, la cantidad máxima de hojas es* $3^4 = 81$. *Esta es la cantidad máxima de computadoras que se pueden conectar con el esquema descrito.* ■

## 1.9. Árboles recubridores

En ciertas aplicaciones, dado un grafo se requiere conectar todos los vértices usando la menor cantidad de conexiones posibles. Como hemos visto, una estructura de vértices y aristas, conexa, con mínima cantidad de aristas es un árbol. En dichas aplicaciones se estaría buscando un subgrafo que sea árbol y que use todos los vértices.

Dado un grafo, un ÁRBOL RECUBRIDOR es un subgrafo que es un árbol y que contiene a todos los vértices del grafo dado.

Si el grafo dado no es conexo, no admite un árbol recubridor, ya que existe al menos un par de vértices que no pueden conectarse con aristas, por lo tanto tampoco se podrán conectar en el árbol recubridor. En cambio, todo grafo conexo admite un árbol recubridor.

***Ejemplo* 1.33**
*En la figura 1.16a se muestra un grafo con 10 vértices y 16 aristas. Un árbol recubridor debe incluir los 10 vértices y 9 aristas. En la parte b y c se muestran (en líneas gruesas) dos árboles recubridores de este grafo.* ■

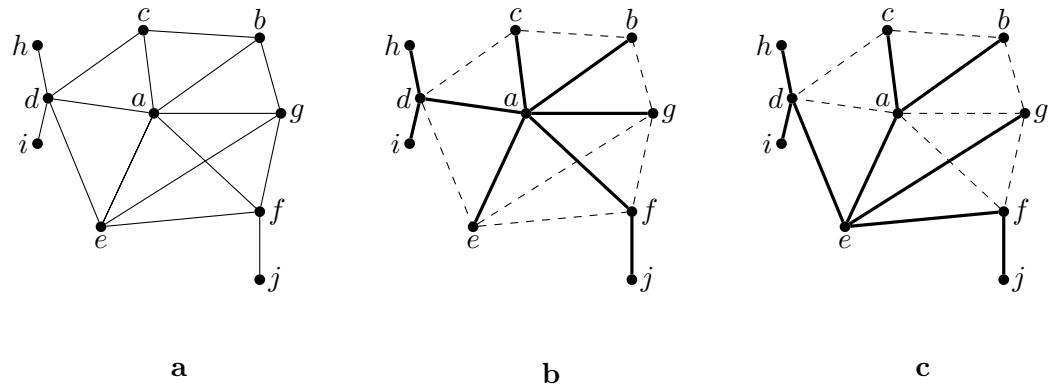



Figura 1.16: Grafo $G$ y dos árboles recubridores.

Una forma de obtener un árbol recubridor es eliminar aristas del grafo a fin de cortar los ciclos. Pero esta estrategia no es eficiente. Existen algoritmos más eficientes, conocidos como BÚSQUEDA EN PROFUNDIDAD y BÚSQUEDA EN ANCHO.

El algoritmo de búsqueda en profundidad procede así: a partir de un vértice inicial, construye un camino simple lo más largo posible, visitando nodos nuevos, pasando de uno a otro con aristas del grafo.

Si así se recorren todos los vértices del árbol, el camino hallado es un árbol recubridor. En caso contrario, sean $u$ y $v$ los últimos vértices del camino, en ese orden. El algoritmo entonces retrocede hasta $u$ y construye a partir de este vértice un camino simple lo más largo posible, visitando nodos no visitados previamente, pasando de uno a otro con aristas del grafo.

Cuando se llegue al último vértice de este camino (es decir, no se pueden agregar vértices no visitados), se retrocede al vértice anterior, y se busca un camino simple como antes.

En este procedimiento de avance y retroceso, en algún momento se llegará (retrocediendo) al vértice inicial. En este punto, los vértices del grafo, junto con las aristas usadas en los caminos hallados, constituye un árbol recubridor del grafo dado.

Los detalles del algoritmo se dan a continuación:

**Algoritmo: búsqueda en profundidad**
**Entrada**: Un grafo con vértices ordenados $v_1, v_2, ..., v_n$.
**Salida**: Un árbol recubridor: conjunto de vértices $\mathcal{V}$, conjunto de aristas $\mathcal{A}$.
**Inicio**:
***Paso 1***: Iniciar $v := v_1$, $\mathcal{V} := \{v_1\}$ y $\mathcal{A} := \emptyset$
***Paso 2***: Seleccionar el menor $i$ tal que $v_i$ es adyacente a $v$ en el grafo, y $v_i$ no está ya incluído en $\mathcal{V}$.
**Si** se encuentra tal vértice
Agregar el vértice y la arista correspondiente al árbol:
$\mathcal{V} := \mathcal{V} \cup \{v_i\}$ y $\mathcal{A} := \mathcal{A} \cup \{\{v, v_i\}\}$.
Asignar $v := v_i$
Ir al paso 2
**Si no**
Ir al paso 3
**Fin si**
***Paso 3***:
**Si** se llegó al vértice inicial ($v = v_1$)
El árbol recubridor es el formado por los vértices $\mathcal{V}$ y las aristas $\mathcal{A}$
**Si no**
Retroceder desde $v$: Si $u$ es el anterior a $v$ en el árbol, asignar $v := u$
Ir al paso 2
**Fin si**

***Ejemplo* 1.34**
*Consideremos el grafo de la figura 1.17a. Aplicaremos el algoritmo de búsqueda en profundidad para construir un árbol recubridor.*

*Paso 1:*
$\mathcal{V} = \{v_1\}$
$\mathcal{A} = \emptyset$.
$v = v_1$

*Paso 2: El primer adyacente a $v$ es $v_3$. Se actualiza:*
$\mathcal{V} = \{v_1, v_3\}$
$\mathcal{A} = \{\{v_1, v_3\}\}$
$v = v_3$

*Volvemos al paso 2:*

*El primer vecino de $v_3$ que no está en el árbol es $v_2$, entonces se agrega:*
$\mathcal{V} = \{v_1, v_3, v_2\}$
$\mathcal{A} = \{\{v_1, v_3\}, \{v_3, v_2\}\}$
$v = v_2$

*Volvemos al paso 2: Buscamos un adyacente a $v_2$ que no esté ya incluído en el árbol. Como $v_2$ no tiene vértices adyacentes que no estén en el árbol, vamos al paso 3.*

*Como no estamos en el vértice inicial, se retrocede al anterior: $v = v_3$. Pasamos a paso 2.*

*$v_3$ sí tiene un vecino que no está en el árbol ($v_4$), se agrega:*

$\mathcal{V} = \{v_1, v_3, v_2, v_4\}$
$\mathcal{A} = \{\{v_1, v_3\}, \{v_3, v_2\}, \{v_3, v_4\}\}$
$v = v_4$

*El primer vecino de $v_4$ no incluido aún en el árbol es $v_5$, se actualiza:*
$\mathcal{V} = \{v_1, v_3, v_2, v_4, v_5\}$
$\mathcal{A} = \{\{v_1, v_3\}, \{v_3, v_2\}, \{v_3, v_4\}, \{v_4, v_5\}\}$
$v = v_5$

*El siguiente vértice a agregar es $v_6$:*
$\mathcal{V} = \{v_1, v_3, v_2, v_4, v_5, v_6\}$
$\mathcal{A} = \{\{v_1, v_3\}, \{v_3, v_2\}, \{v_3, v_4\}, \{v_4, v_5\}, \{v_5, v_6\}\}$
$v = v_6$

*Luego se agrega $v_7$:*
$\mathcal{V} = \{v_1, v_3, v_2, v_4, v_5, v_6, v_7\}$
$\mathcal{A} = \{\{v_1, v_3\}, \{v_3, v_2\}, \{v_3, v_4\}, \{v_4, v_5\}, \{v_5, v_6\}, \{v_6, v_7\}\}$
$v = v_7$

*Como $v_7$ no tiene vecinos no incluidos en el árbol, vamos al paso 3.*

*No es el nodo inicial, entonces se retrocede al anterior: $v = v_6$.*

*$v_6$ tiene un vértice adyacente no incluido aún en el árbol, $v_8$. Actualizamos:*
$\mathcal{V} = \{v_1, v_3, v_2, v_4, v_5, v_6, v_7, v_8\}$
$\mathcal{A} = \{\{v_1, v_3\}, \{v_3, v_2\}, \{v_3, v_4\}, \{v_4, v_5\}, \{v_5, v_6\}, \{v_6, v_7\}, \{v_6, v_8\}\}$
$v = v_8$

*En los siguientes pasos se agregan las aristas $\{v_8, v_9\}$ y $\{v_9, v_{10}\}$. En $v_{10}$ se debe retroceder a $v_9$. De aquí, se avanza a $v_{11}$. Hasta aquí, el árbol es:*
$\mathcal{V} = \{v_1, v_3, v_2, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$
$\mathcal{A} = \{\{v_1, v_3\}, \{v_3, v_2\}, \{v_3, v_4\}, \{v_4, v_5\}, \{v_5, v_6\}, \{v_6, v_7\}, \{v_6, v_8\}, \{v_8, v_9\},$
$\{v_9, v_{10}\}, \{v_9, v_{11}\}\}$

*De $v_{11}$ hay que retroceder al anterior, $v_9$, y luego al vértice anterior, $v_8$. Así sucesivamente. Cada vez que se retrocede, no se encuentran nuevos vértices para agregar. El proceso de retroceso finaliza cuando se llega a $v_1$, el vértice inicial. Así, el algoritmo termina, obteniéndose el árbol recubridor que se muestra en la figura 1.17b.*

**Para pensar**:
1. Se ha dicho que un árbol no conexo no admite un árbol recubridor. ¿Qué sucede si la entrada del algoritmo anterior es un grafo no conexo?
2. Luego de que se agrega el último vértice al árbol, sólo se realizan operaciones de retroceso. Por lo tanto, el árbol está completamente construído cuando se agrega este último vértice, y las operaciones de retroceso se realizan sólo para conseguir el criterio de salida. ¿Cómo debería modificarse el algoritmo para que finalice al terminar de formar el árbol, sin las operaciones finales de retroceso?

El otro algoritmo que se describirá para generar árboles recubridores es el algoritmo de búsqueda a lo ancho. Este algortimo construye árboles que, en comparación con la búsqueda en profundidad, son de menor altura y más anchos, es decir, cada nodo tiene más hijos. Procede de la siguiente manera: a partir de un vértice inicial, agrega al árbol

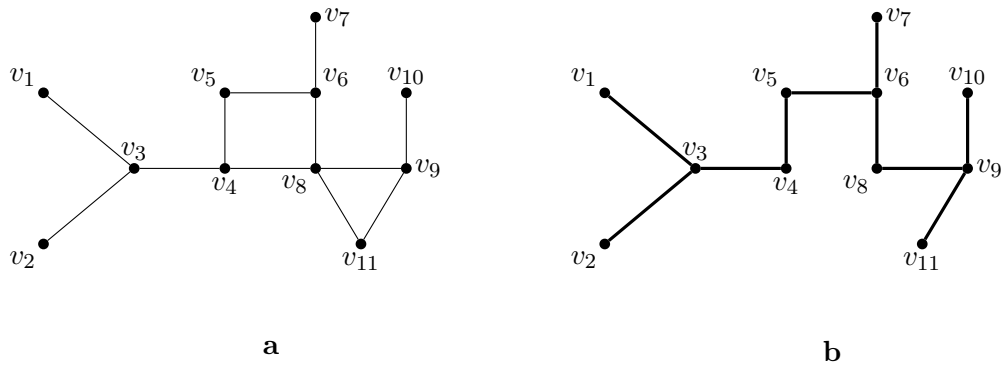



Figura 1.17: Grafo del ejemplo 1.34 y el árbol recubridor obtenido con la búsqueda en profundidad.

todos los vértices adyacentes a éste, junto con las aristas que los conectan. Los vértices agregados se guardan en una lista con estructura FIFO (cola).

Luego, se toma el primer vértice de la cola, se lo elimina de la misma, y se agregan al árbol todos los vértices adyacentes a él que no hayan sido incluídos aún en el árbol. Los vértices agregados al árbol, se suman a la cola. Esta operación de tomar el primer vértice y agregar a sus adyacentes al árbol se repite hasta que la lista quede vacía.

Los detalles del algoritmo se describen a continuación:

**Algoritmo: búsqueda a lo ancho**
**Entrada**: Un grafo con vértices ordenados $v_1, v_2, ..., v_n$.
**Salida**: Un árbol recubridor: conjunto de vértices $\mathcal{V}$, conjunto de aristas $\mathcal{A}$.
**Inicio**:
$Q := \{v_1\}$
$\mathcal{V} := \{v_1\}$
$\mathcal{A} := \emptyset$
**Mientras** haya vértices en $Q$
    Sea $u$ el primer elemento de $Q$.
    Eliminar $u$ de $Q$: $Q := Q - \{u\}$
    **Para** todos los $v$ adyacentes a $u$ que no están en $\mathcal{V} \cup Q$
        Agregar $v$: $\mathcal{V} := \mathcal{V} \cup \{v\}$
        Agregar la arista correspondiente al árbol: $\mathcal{A} := \mathcal{A} \cup \{\{u, v\}\}$
        Agregar $v$ a la cola: $Q := Q + \{v\}$
    **Fin para**
**Fin mientras**

***Ejemplo* 1.35**
*Apliquemos el algoritmo de búsqueda a lo ancho para hallar un árbol recubridor en el grafo de la figura 1.18a. Comenzando con el vértice $v_1$, agregamos sus vecinos al árbol, y los nuevos vértices a la cola:*

$\mathcal{V} = \{v_1, v_2, v_3, v_4\}$
$\mathcal{A} = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}\}$
$Q = \{v_2, v_3, v_4\}$

*En el siguiente paso, se quita el primer elemento de $Q$, $v_2$, y se agregan al árbol los vértices adyacentes a $v_2$ que aún no estén incluidos, ademas de agregarlos en $Q$:*
$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_8\}$
$\mathcal{A} = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_8\}\}$
$Q = \{v_3, v_4, v_8\}$

*A continuación, se toma el primer elemento de $Q$, $v_3$, se elimina de $Q$, y se consideran los vecinos de $v_3$ no incluidos. No hay ningún vértice con esas condiciones. Entonces el único cambio es en $Q$:*
$Q = \{v_4, v_8\}$

*En el paso siguiente, se toma el primer elemento de $Q$, $v_4$, se agregan nuevos vértices, adyacentes a $v_4$, y se actualiza el árbol y la cola:*
$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_8, v_5, v_7\}$
$\mathcal{A} = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_8\}, \{v_4, v_5\}, \{v_4, v_7\}\}$
$Q = \{v_8, v_5, v_7\}$

*A continuación, se toma $v_8$, y se agregan al árbol el vértice adyacente $v_9$:*
$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_8, v_5, v_7, v_9\}$
$\mathcal{A} = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_8\}, \{v_4, v_5\}, \{v_4, v_7\}, \{v_8, v_9\}\}$
$Q = \{v_5, v_7, v_9\}$

*Luego, se saca $v_5$ de $Q$ y se considera su adyacente $v_6$:*
$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_8, v_5, v_7, v_9, v_6\}$
$\mathcal{A} = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_8\}, \{v_4, v_5\}, \{v_4, v_7\}, \{v_8, v_9\}, \{v_5, v_6\}\}$
$Q = \{v_7, v_9, v_6\}$

*En el siguiente paso se elimina $v_7$ y se agrega el nuevo vértice $v_{10}$:*
$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_8, v_5, v_7, v_9, v_6, v_{10}\}$
$\mathcal{A} = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_8\}, \{v_4, v_5\}, \{v_4, v_7\}, \{v_8, v_9\}, \{v_5, v_6\}, \{v_7, v_{10}\}\}$
$Q = \{v_9, v_6, v_{10}\}$

*El primer elemento de $Q$ es $v_9$, que no tiene nuevos vértices para agregar al árbol. El siguiente es $v_6$, que es adyacente a $v_{11}$:*
$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_8, v_5, v_7, v_9, v_6, v_{10}, v_{11}\}$
$\mathcal{A} = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_8\}, \{v_4, v_5\}, \{v_4, v_7\}, \{v_8, v_9\}, \{v_5, v_6\}, \{v_7, v_{10}\}, \{v_6, v_{11}\}\}$
$Q = \{v_{10}, v_{11}\}$

*El primer elemento en la cola es $v_{10}$. No tiene nuevos vecinos, por lo tanto, se elimina de $Q$. Finalmente, queda $v_{11}$, que tampoco tiene nuevos vecinos. Se elimina de $Q$. Así, $Q$ queda vacío y el algoritmo termina.*

**Para pensar**:

a. ¿Qué sucede si la entrada del algoritmo anterior es un grafo no conexo?

b. Los últimos pasos del algoritmo, siempre se ocupan para eliminar uno a uno los vértices en la pila, el árbol recubridor ya estaba listo desde antes. ¿Cómo puede modificarse el algoritmo para detectar antes que el árbol ya está completo, evitando ejecutar los últimos ciclos?

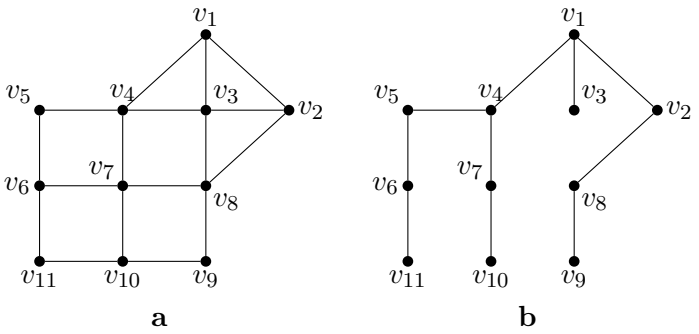



Figura 1.18: Grafo del ejemplo 1.35 y el árbol recubridor obtenido con la búsqueda en profundidad.

En los dos algoritmos descritos, el árbol generado depende del orden asignado a los vértices. Cambiando el orden de los mismos, el árbol generado será distinto.

## 1.10. Árbol recubridor minimal

En grafos ponderados, interesa en ciertas aplicaciones, hallar un árbol recubridor de peso mínimo. Tal árbol se denomina ÁRBOL RECUBRIDOR MINIMAL.

Por ejemplo, suponga que se tiene una red de computadoras y se requiere aumentar la seguridad en algunos enlaces, para formar una red de comunicación fiable que conecte todas las computadoras. La implementación de este aumento de seguridad supone un costo que se debe afrontar. Si se conoce el costo correspondiente a cada uno de los enlaces, interesa seleccionar enlaces de forma tal que todas las computadoras estén conectadas y que la suma de los costos sea mínimo.

La red existente puede verse como un grafo ponderado, y el costo de aumentar la seguridad en cada enlace es el peso de cada arista. Lo que se busca es un árbol recubridor minimal.

Otro ejemplo surge en la siguiente situación: se tienen un conjunto de ciudades, que están conectadas por rutas sin pavimentar. Se decide pavimentar algunas de estas rutas de forma tal que dos ciudades cualesquiera puedan conectarse por un camino pavimentado. Además, se quiere llevar esto a cabo con el menor costo.

Las ciudades pueden verse como vértices de un grafo, y las rutas como las aristas. Cada arista tiene un peso, dado por la longitud de ese tramo de ruta, o por el costo de pavimentar ese tramo. Se busca seleccionar un conjunto de aristas de peso mínimo, tal que ese conjunto de aristas, junto con todos los vértices, resulte un grafo conexo. El resultado será un árbol recubridor minimal.

Dado un grafo ponderado, un ÁRBOL RECUBRIDOR MAXIMAL es un árbol recubridor de peso máximo.

***Ejemplo* 1.36**
*En la figura 1.19a se muestra un grafo ponderado y tres árboles recubridores, con sus correspondientes pesos. El árbol de la parte c de la figura es minimal, y el de la parte d es*

*maximal.* ■

Describiremos dos algoritmos muy conocidos, para hallar árboles recubridores minimales. Estos algoritmos son voraces. Se llama así a los algoritmos que en cada paso realizan una elección óptima. Los árboles producidos por los algoritmos que detallaremos a continuación son globalmente óptimos.

**Algoritmo: Kruskal**
**Entrada**: Un grafo con pesos en las aristas.
**Salida**: Un árbol recubridor de peso mínimo.
**Inicio**:
$n :=$ cantidad de vértices en el grafo
Inicializar el árbol vacío.
**Para** $i$ de 1 a $n-1$ **hacer**
Seleccionar una arista de peso mínimo entre las que no se hayan agregado al árbol, tal que no forme un ciclo al agregarla.
Agregar la arista y sus extremos al árbol
**Fin para**

**Algoritmo: Prim**
**Entrada**: Un grafo con pesos en las aristas.
**Salida**: Un árbol recubridor de peso mínimo.
**Inicio**:
$n :=$ cantidad de vértices en el grafo
Inicializar el árbol con la arista de peso mínimo, y sus vértices extremos.
**Para** $i$ de 1 a $n-2$ **hacer**
Seleccionar una arista incidente en algún vértice del árbol, de peso mínimo entre las que no están en el árbol, tal que no forme un ciclo al agregarla.
Agregar la arista seleccionada y sus extremos al árbol.
**Fin para**

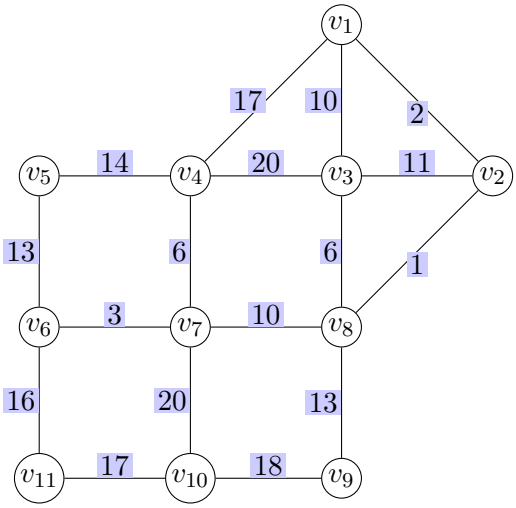
**Para pensar**.

a. Al finalizar, los algoritmos descritos dan un árbol minimal. ¿Cómo se debería modificar para que también informe el peso de ese árbol?

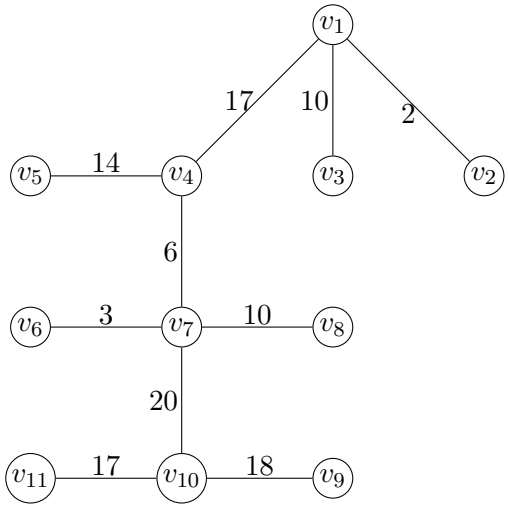
b. ¿Cómo se debería modificar el algoritmo para que halle un árbol recubridor maximal?

## 1.11. Problemas

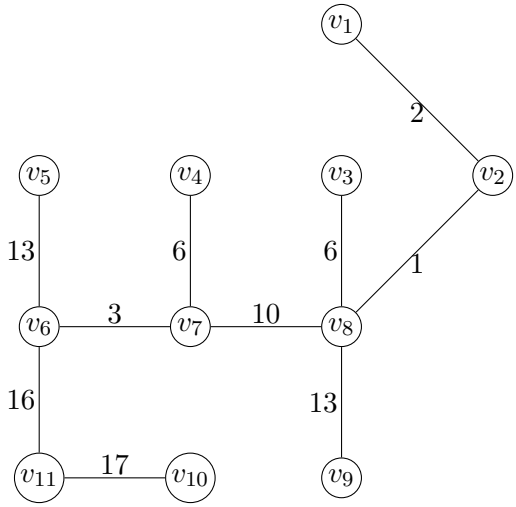
**Problema 1.1**
*Para cada uno de los grafos de la figura 1.20, indicar:*



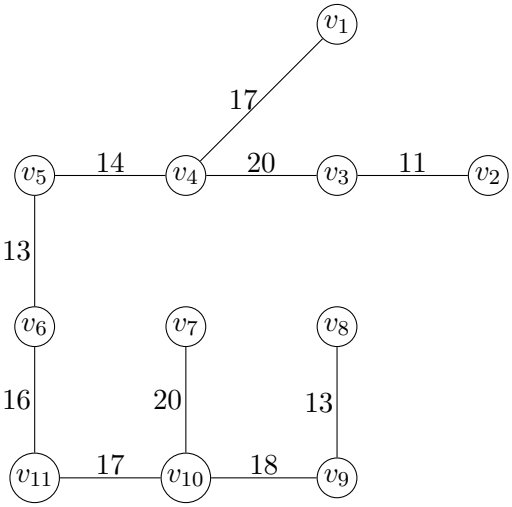

**a**. Grafo ponderado




**b**. Árbol recubridor de peso=117

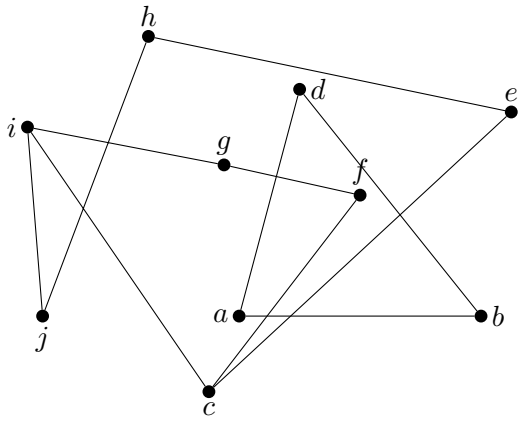



**c**. Árbol recubridor minimal de peso=87

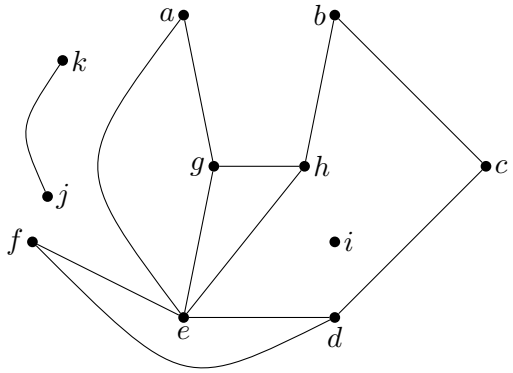



**d**. Árbol recubridor maximal de peso=159

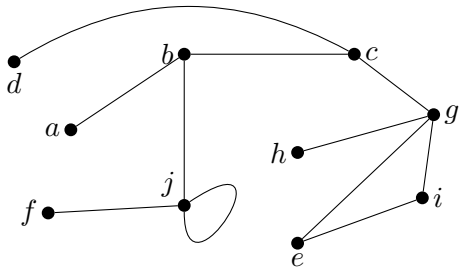
Figura 1.19: Grafo del ejemplo 1.36 y árboles recubridores.



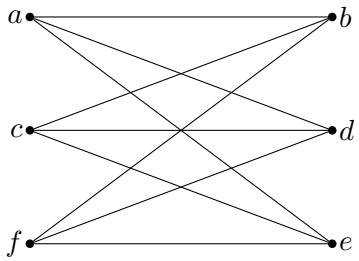

G1




G2




G3




G4

Figura 1.20: Grafos del problema 1.1

a. *El grado de cada vértice. Verificar la fórmula que relaciona los grados de vértices con el número de aristas.*

b. *Un ciclo en el grafo, (si hay).*

c. *Un camino cerrado que no sea ciclo (si hay).*

d. *Un recorrido de a a d, que no sea camino simple (si hay).*

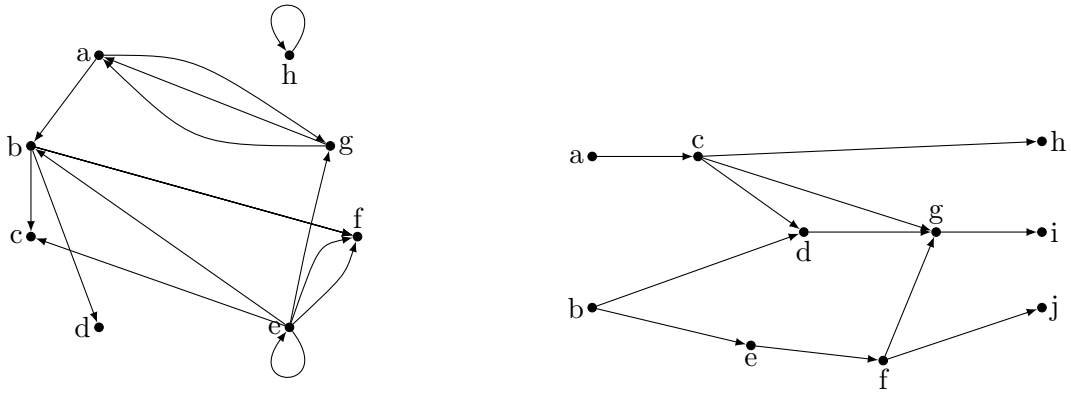
e. *Un ciclo de longitud par (si hay).*

f. *Todos los caminos simples de e a d.*

g. *La cantidad de componentes conexas.*

h. *La cantidad mínima de aristas que deben eliminarse para aumentar la cantidad de componentes conexas.*

i. *La cantidad máxima de aristas que pueden eliminarse manteniendo la cantidad de componentes conexas.*




G5 G6

Figura 1.21: Grafos y multigrafos dirigidos del problema 1.2

**Problema 1.2**

*Para cada uno de los grafos o multigrafos dirigidos de la figura 1.21, indicar:*

a. *El grado de entrada, y el grado de salida de cada vértice. Verificar la fórmula que relaciona los grados de entrada y salida de los vértices con el número de aristas.*

b. *Un ciclo dirigido en el grafo, (si hay).*

c. *Caminos dirigidos desde a los demás vértices.*

**Problema 1.3**

*Dada una colección de conjuntos, el grafo de intersección se define como el que tiene un vértice por cada conjunto, y una arista entre dos vértices que representan conjuntos de intersección no vacía. Representar el grafo de intersección de los conjuntos dados y escribir la matriz de adyacencia de cada uno.*

a. $A_1 = \{0, 1, 2, 3, 4, 5\}$; $A_2 = \{10, 11, 12\}$; $A_3 = \{1, 5, 10, 15, 20\}$; $A_4 = \{3, 12, 19\}$; $A_5 = \{0, 10, 20, 30, 40\}$; $A_6 = \mathbb{N} = \{$*números naturales*$\}$

b. $B_1 = \{n \in \mathbb{N} : n$ *es divisor de 4 mayor que 1*$\}$; $B_2 = \{n \in \mathbb{N} : n$ *es divisor de 6 mayor que 1*$\}$; $B_3 = \{n \in \mathbb{N} : n$ *es divisor de 11 mayor que 1*$\}$;$B_4 = \{n \in \mathbb{N} : n$ *es divisor de 9 mayor que 1*$\}$;

c. $C_1 = \{n \in \mathbb{N} : n$ *es múltiplo de 4*$\}$; $C_2 = \{n \in \mathbb{N} : n$ *es múltiplo de 6*$\}$; $C_3 = \{n \in \mathbb{N} : n$ *es múltiplo de 11*$\}$; $C_4 = \{n \in \mathbb{N} : n$ *es múltiplo de 9*$\}$;

**Problema 1.4**

*Indicar si existe un grafo regular con las siguientes características (dar un ejemplo o justificar la no existencia):*

a. *7 vértices y 7 aristas*

b. *7 vértices y 16 aristas*

c. *3 vértices, 6 aristas, sin bubles*

d. *4 vértices, 6 aristas, sin bucles*

**Problema 1.5**
*Indicar para qué valores de n es regular: $K_n$, $C_n$, $W_n$, $S_n$, $L_n$.*

**Problema 1.6**
*Escribir la matriz de adyacencia de los grafos: $K_6$ , $C_6$, $W_5$, $S_5$ , $L_6$.*

**Problema 1.7**
*Describir las matrices de adyacencia de $K_n$, $C_n$, $W_n$, $S_n$, $L_n$ dando el valor de cada componente $a_{ij}$ en función de $i$, $j$ y $n$.*

**Problema 1.8**
*Un n-cubo es un grafo en el que los vértices se etiquetan con n-uplas de ceros y unos. Una arista conecta dos vértices u y v si las etiquetas de u y v difieren exactamente en un símbolo.*

a. *Construir el 2-cubo.*

b. *Construir el 3-cubo.*

c. *¿Es conexo el n-cubo? Justificar.*

d. *Calcular el número de vértices y de aristas del n-cubo. Justificar.*

e. *Probar que el grafo n-cubo es bipartito. Graficar el 3-cubo de forma de mostrar que es bipartito.*

f. *¿Para qué n el n-cubo es regular? Justificar.*

g. *¿Puede existir un ciclo de longitud 3 en el n-cubo? Justificar. (Pensarlo desde el punto de vista de las etiquetas, ¿qué representa un ciclo de longitud 3?)*

h. *¿Puede existir un ciclo de longitud impar?*

i. *¿Para qué valores de n el n-cubo es plano?*

**Problema 1.9**
*¿Cómo se puede determinar la cantidad de aristas de un grafo a partir de su matriz de adyacencia? ¿Cómo se puede determinar el conjunto de vértices conectados con un vértice determinado, a partir de la matriz de adyacencia?*

**Problema 1.10**
*Si A es la matriz de adyacencia de un grafo, $A^k$ es una matriz cuyo elemento $(i, j)$ es el número de caminos de longitud k que hay en el grafo entre los vértices i y j. ¿Cómo se puede usar este resultado para probar si un grafo es conexo?*

**Problema 1.11**
*Indicar para qué valores de n los siguientes grafos admiten un circuito euleriano: $K_n$, $C_n$, $W_n$, $S_n$ , $L_n$.*

**Problema 1.12**
*Considere el grafo bipartito completo $K_{3,2}$, que consiste de 5 vértices: a, b, c, u y v; y las aristas unen a con u y v; b con u y v; y c con u y v.*

a. *¿Es un grafo regular? Justificar.*

b. *Hallar, si es posible, un ciclo de longitud 4. De no ser posible, justificar.*

c. *Hallar, si es posible, un ciclo de longitud 3. De no ser posible, justificar.*

d. *Hallar, si es posible, un ciclo de longitud 6. De no ser posible, justificar.*

e. *¿Es planar? Justificar.*

f. *Calcular la longitud del camino más corto entre cada par de vértices.*

**Problema 1.13**

*La distancia entre dos vértices u y v en un grafo es la longitud del camino mínimo entre u y v. Se define el diámetro de un grafo conexo G como la mayor distancia entre dos vértices del grafo; y se denota diam(G). Hallar diam($K_{n,m}$), diam($K_n$), diam($C_n$), diam($W_n$), diam($L_n$).*

**Problema 1.14**

*Considere el grafo dado por la siguiente matriz de adyacencia:*

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

*con vértices $v_1$, $v_2$, ..., $v_{12}$.*

a. *Determinar el grado de cada vértice.*

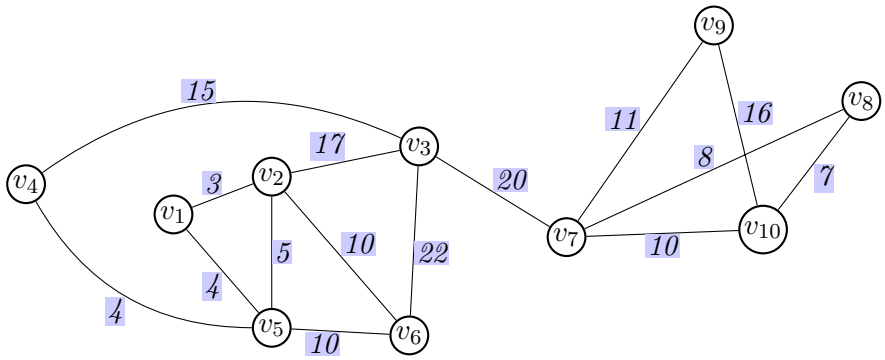
b. *Si es conexo, hallar un árbol recubridor. Si no es conexo, ¿cuál es el mínimo número de aristas que es necesario agregarle para que sea conexo? Justificar.*

c. *Hallar la distancia entre el vértice 1 los demás vértices.*

d. *Considerar el mismo grafo, agregando las aristas $\{v_2, v_5\}$ y $\{v_2, v_4\}$. Del grafo resultante, hallar un árbol recubridor. Considerando al árbol obtenido como un árbol con raíz en el vértice 1 ¿cuál es su altura?.*

**Problema 1.15**

*Dado el grafo ponderado que se muestra a continuación,*




*Obtener:*

a. *Un árbol recubridor minimal.*

b. *Todos los caminos simples de $v_1$ a $v_5$.*

c. *Un ciclo de longitud 5, si existe. Si tal ciclo no existe, justificar detalladamente.*

d. *La distancia de $v_2$ a los restantes vértices, aplicando el algoritmo de Dijkstra.*

**Problema 1.16**

*Dada la siguiente matriz de pesos de un grafo,*

$$\begin{bmatrix} 0 & 3 & 4 & 0 & 0 & 0 & 0 & 32 & 4 & 0 & 0 & 0 \\ 3 & 0 & 5 & 27 & 45 & 0 & 0 & 0 & 30 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 27 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 45 & 0 & 0 & 0 & 22 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 22 & 0 & 11 & 41 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11 & 0 & 0 & 0 & 0 & 0 & 0 \\ 32 & 0 & 0 & 0 & 0 & 41 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 30 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 19 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 19 & 9 & 0 \end{bmatrix}$$

*con vértices $v_1$, $v_2$, ..., $v_{12}$.*

*Responder:*

a. *¿Cómo puede determinarse si el grafo es conexo o no a partir de la matriz de pesos? (sin graficarlo)*

b. *¿Este grafo admite un árbol recubridor? Justificar.*

c. *Hallar un árbol recubridor minimal de cada componente conexa del grafo y dar el peso de cada uno.*

d. *Determinar una relación entre la cantidad de vértices de un grafo y la cantidad de componentes conexas del grafo y la cantidad de aristas en un bosque recubridor. Justificar.*

# Capítulo 2

# Relaciones de recurrencia

## 2.1. Definiciones básicas y ejemplos

Una sucesión de números reales es una lista infinita ordenada de reales: $a_0$, $a_1$, $a_2$, $a_3$, ... Cada elemento de la sucesión se denomina *término*, y se caracteriza por su valor (el número real $a_n$) y su posición en la lista ordenada ($n$).

Para cada entero no negativo $n$, se denota $a_n$ el n-ésimo término de la sucesión, $a_{n-1}$ es el término anterior a $a_n$ y $a_{n+1}$ es el siguiente. Obviamente, $a_{n-2}$ es el término anterior a $a_{n-1}$ y $a_n$ es el anterior al anterior a $a_{n+2}$, etc.

En ocasiones la sucesión comienza con $a_1$ (no existe el término $a_0$).

***Ejemplo* 2.1**
*Sabiendo que los primeros términos de una sucesión son:* $a_0 = 3$, $a_1 = 6$, $a_2 = 9$, $a_3 = 12$, *... , se puede inferir que el término n-ésimo de la sucesión es* $a_n = 3(n + 1)$.

*Por otro lado, se puede construir la sucesión observando que cada término* $a_n$ *se obtiene sumando 3 al anterior* $a_{n-1}$. *Esto es:* $a_n = 3 + a_{n-1}$. ■

***Ejemplo* 2.2**
*Se tiene una sucesión que comienza con los términos* $a_0 = 1$, $a_1 = 1/2$, $a_2 = 1/4$, $a_3 = 1/8$, *...*

*El término n-ésimo es* $a_n = 1/2^n$.

*También puede verse que cada término es 1/2 por el anterior, es decir,* $a_n = (1/2) \cdot a_{n-1}$. ■

***Ejemplo* 2.3**
*Otro ejemplo es la sucesión cuyos primeros términos son* $a_0 = 17/2$, $a_1 = 17/2$, $a_2 = 17/2$, $a_3 = 17/2$. *Es decir, cada término es igual al anterior, entonces* $a_n = a_{n-1}$, *o* $a_n = 17/2$. ■

***Ejemplo* 2.4**
*La sucesión cuyos primeros términos son* $a_0 = 7$, $a_1 = 9$, $a_2 = 11$, $a_3 = 13$ *satisface la condición que cada término es 2 más que el anterior,* $a_n = a_{n-1} + 2$.

*No es difícil obtener el término general explícitamente, que es* $a_n = 7 + 2n$. ■

***Ejemplo* 2.5**
*Considere la sucesión cuyos primeros términos son* $a_0 = 1$, $a_1 = 1$, $a_2 = 2$, $a_3 = 3$, $a_4 = 5$, $a_5 = 8$.

*Analizando estos primeros términos, podemos darnos cuenta de que a partir de $a_2$, cada término es la suma de los dos anteriores, es decir, $a_n = a_{n-1} + a_{n-2}$, para $n \geqslant 2$. O, equivalentemente, $a_{n+1} = a_n + a_{n-1}$, para $n \geqslant 1$.* ■

***Ejemplo* 2.6**
*Las bacterias de cierta colonia se reproducen por duplicación. Se sabe que al cabo de una hora todas se duplican. Es decir, la cantidad de bacterias que hay a una determinada hora es el doble de la cantidad de bacterias que había una hora antes.*

*Siendo $a_n$ la cantidad de bacterias en la hora $n$, se puede plantear que $a_n$ es el doble de $a_{n-1}$; es decir: $a_n = 2a_{n-1}$.* ■

***Ejemplo* 2.7**
*Un banco ofrece una tasa de interés mensual $p$ para los plazos fijos. Cada mes, los intereses acumulados se agregan al plazo fijo. Si en el mes $n$, el dinero acumulado en un plazo fijo es $C_n$, ese monto es la suma del dinero en el mes anterior ($C_{n-1}$) más el interés generado en ese mes ($p \cdot C_{n-1}$). Es decir, $C_n = C_{n-1} + p \cdot C_{n-1} = (1 + p) \cdot C_{n-1}$.* ■

En algunos de los ejemplos anteriores se logra describir sucesiones en forma explícita, dando una fórmula para el término $n$-ésimo que depende sólo de su posición $n$: $a_n = F(n)$. También se obtienen expresiones para $a_n$ que dependen de los términos anteriores, se definen en forma recurrente.

Dada una sucesión $a_0, a_1, a_2, ...$, una RELACIÓN DE RECURRENCIA (o simplemente recurrencia) para esa sucesión es una igualdad que relaciona un término de una sucesión con ciertos términos anteriores. Específicamente, una relación de recurrencia se puede escribir de la forma:

$$a_n = f(a_{n-1}, a_{n-2}, ..., a_{n-k})$$

válida para $n \geqslant k$. El ORDEN de una relación de recurrencia es la cantidad de términos anteriores que intervienen. En el caso general expuesto antes, el orden es $k$.

En los ejemplos anteriores han aparecido las relaciones de recurrencia: $a_n = 3 + a_{n-1}$, $a_n = (1/2) \cdot a_{n-1}$, $a_n = a_{n-1}$, $a_n = a_{n-1} + 2$, $a_n = 2a_{n-1}$, $C_n = (1 + p) \cdot C_{n-1}$, todas éstas de orden 1. Y la relación de recurrencia $a_n = a_{n-1} + a_{n-2}$ es de orden 2.

***Ejemplo* 2.8**
*La relación de recurrencia $a_n = 2a_{n-1} + 1$ tiene orden 1. La recurrencia $a_n = a_{n-1}.a_{n-3}$ tiene orden 3.*

*Notar que la primera recurrencia también se puede escribir: $a_{n+1} = 2a_n + 1$, ya que expresa lo mismo: un término, $a_{n+1}$, se obtiene multiplicando el anterior, $a_n$, por dos y sumando 1 al resultado.*

*Y la segunda recurrencia también puede escribirse $a_{n+1} = a_n.a_{n-2}$. A pesar de escribirla de otra forma, la recurrencia sigue siendo de orden 3: se necesitan 3 términos anteriores ($a_n$, $a_{n-1}$ y $a_{n-2}$) para definir el término $a_{n+1}$.* ■

Una SOLUCIÓN DE LA RELACIÓN DE RECURRENCIA es una sucesión que verifica la recurrencia. Existen infinitas soluciones de una recurrencia. Por ejemplo, en el ejemplo 2.4 se llegó a la recurrencia $a_n = a_{n-1} + 2$. La sucesión 2, 4, 6, 8, ... satisface la misma recurrencia, ya que cada término se obtiene sumando 2 al anterior. La misma recurrencia

también es satisfecha por la sucesión -5/3, 1/3, 7/3, 13/3,... Sin embargo, si especificamos el primer término, la solución es única.

En el ejemplo 2.5 se obtiene la recurrencia $a_n = a_{n-1} + a_{n-2}$ para la sucesión 1, 1, 2, 3, 5, 8, ... . Esa recurrencia también describe la sucesión 1, 0, 1, 1, 2, 3, ..., y también la sucesión 1, -1, 0, -1, -1, -2, -3, ... . En estas dos nuevas sucesiones, cada término (a partir del tercero) es igual a la suma de los dos términos anteriores. Además, en este caso, todas empiezan con 1. Por ser una recurrencia de orden dos, se necesita especificar los dos primeros términos para que la solución sea única.

En la relación de recurrencia general, $a_n = f(a_{n-1}, a_{n-2}, ..., a_{n-k})$, especificando los primeros $k$ términos, la solución es única. Es decir, existe una sola sucesión que verifica

$$\begin{gathered} a_n = f(a_{n-1}, a_{n-2}, ..., a_{n-k}) \quad n \geqslant k \\ a_0 = A_0 \\ a_1 = A_1 \\ ... \\ a_{k-1} = A_{k-1} \end{gathered}$$

donde $A_i$ son números reales dados. Las $k$ igualdades $a_i = A_i$ se denominan CONDICIONES INICIALES.

Nos ocuparemos en la sección siguiente de conocer métodos para resolver relaciones de recurrencia con condiciones iniciales. Veremos primero cómo resolver algunos ejemplos, y luego generalizaremos un método para un tipo especial de recurrencias.

***Ejemplo* 2.9**

*Considere la recurrencia $a_n = (a_{n-1})^2 + 1$, para $n \geqslant 1$, con condición inicial $a_0 = -1$. La recurrencia indica que cada término es el cuadrado del anterior más 1. Entonces,*

*para* $n = 1$, $a_1 = a_0^2 + 1 = (-1)^2 + 1 = 2$
*para* $n = 2$, $a_2 = a_1^2 + 1 = 2^2 + 1 = 5$
*para* $n = 3$, $a_3 = a_2^2 + 1 = 5^2 + 1 = 26$
*para* $n = 4$, $a_4 = a_3^2 + 1 = 26^2 + 1 = 677$
*...* ■

***Ejemplo* 2.10**

*Considere la recurrencia de orden 2 que indica que cada término es el anterior menos el anterior del anterior: $a_n = a_{n-1} - a_{n-2}$, para $n \geqslant 2$, con condiciones iniciales $a_0 = 0$, $a_1 = 1$. Entonces,*

*para* $n = 2$, $a_2 = a_1 - a_0 = 1 - 0 = 1$
*para* $n = 3$, $a_3 = a_2 - a_1 = 1 - 1 = 0$
*para* $n = 4$, $a_4 = a_3 - a_2 = 0 - 1 = -1$
*para* $n = 5$, $a_5 = a_4 - a_3 = -1 - 0 = -1$
*para* $n = 6$, $a_6 = a_5 - a_4 = -1 - (-1) = 0$
*para* $n = 7$, $a_7 = a_6 - a_5 = 0 - (-1) = 1$
*...*
*Aparentemente, en la sucesión se alternan repeticiones dobles de 1 y -1, intercaladas con repeticiones simples de 0.* ■

***Ejemplo* 2.11**
*Considere una sucesión donde cada término $a_n$ es el producto de su posición $n$ por el término anterior $a_{n-1}$. Es decir, satisface la recurrencia $a_n = n \cdot a_{n-1}$, $n \geqslant 1$ con condición inicial $a_0 = 1$. Los primeros términos son:*
*para* $n = 1$, $a_1 = 1\, a_0 = 1$
*para* $n = 2$, $a_2 = 2\, a_1 = 2{\cdot}\, 1 = 2$
*para* $n = 3$, $a_3 = 3\, a_2 = 3{\cdot}\, 2{\cdot}\, 1 = 6$
*para* $n = 4$, $a_4 = 4\, a_3 = 4{\cdot}\, 3{\cdot}\, 2{\cdot}\, 1 = 24$
*para* $n = 5$, $a_5 = 5\, a_4 = 5{\cdot}\, 4{\cdot}\, 3{\cdot}\, 2{\cdot}\, 1 = 120$

*Puede inferirse que la solución es $a_n = n!$, $n \geqslant 0$.*

***Ejemplo* 2.12**
*Dada la recurrencia $a_n = 2a_{n-1}$, $n \geqslant 1$, con condición inicial $a_0 = -1/3$, tenemos:*
*para* $n = 1$, $a_1 = 2a_0 = -2/3$
*para* $n = 2$, $a_2 = 2a_1 = -4/3$
*para* $n = 3$, $a_3 = 2a_2 = -8/3$
*para* $n = 4$, $a_4 = 2a_3 = -16/3$
*para* $n = 5$, $a_5 = 2a_4 = -32/3$

*Es fácil ver que la solución es $a_n = \left(-\frac{1}{3}\right) 2^n$, para todo $n \geqslant 0$.* ■

***Ejemplo* 2.13**
*En el problema de la inversión en el banco, $C_n = (1 + p) \cdot C_{n-1}$. Si el depósito inicial es de* 1000*, es decir, $C_0 = 1000$,*
*en el primer mes* $C_1 = (1 + p) \cdot C_0 = (1 + p) \cdot 1000$
*en el segundo mes* $C_2 = (1 + p) \cdot C_1 = (1 + p)^2 \cdot 1000$
*en el tercer mes* $C_3 = (1 + p) \cdot C_2 = (1 + p)^3 \cdot 1000$
*en el cuarto mes* $C_4 = (1 + p) \cdot C_3 = (1 + p)^4 \cdot 1000$

*Así, la solución puede expresarse: $C_n = (1 + p)^n \cdot C_0$.* ■

Una RECURRENCIA LINEAL CON COEFICIENTES CONSTANTES (RLCC) es de la forma

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + ... + c_k a_{n-k} + g(n)$$

donde $c_i$ son constantes, y $g(n)$ es una función que puede depender de $n$, pero no de los $a_n$. Si $g(n) = 0$, la recurrencia es HOMOGÉNEA.

***Ejemplo* 2.14**
*De las recurrencias vistas en los ejemplos anteriores, las que no son RLCC son: $a_n = n{\cdot}a_{n-1}$ porque el coeficiente que multiplica al término $n-1$ no es constante; $a_n = a_{n-1}.\, a_{n-3}$ porque aparecen dos términos de la sucesión multiplicándose; $a_n = (a_{n-1})^2 + 1$ porque aparece un término de la sucesión al cuadrado. Todas las demás son lineales con coeficientes constantes.* ■

## 2.2. Resolución de recurrencias lineales homogéneas

En esta sección veremos un método para hallar la solución de una relación de recurrencia lineal de coeficientes constantes homogéneas, comenzando por las de orden 1.

### 2.2.1. Recurrencias lineales de orden 1

La recurrencia lineal de coeficientes constantes homogénea de primer orden es de la forma

$$a_n = c \cdot a_{n-1},\ n \geqslant 1$$

siendo $c$ una constante dada. La solución general es de la forma $a_n = \alpha \cdot c^n$, donde $\alpha$ es una constante a determinar con la condición inicial. Específicamente, si la condición inicial es $a_0 = A$, debe cumplirse que $a_0 = \alpha \cdot c^0 = \alpha = A$. Por lo tanto, la solución de la recurrencia con la condición inicial dada es $a_n = A \cdot c^n$. Los ejemplos 2.12 y 2.13 muestran soluciones de este tipo.

***Ejemplo* 2.15**
*Considere la recurrencia* $10a_n + 4a_{n-1} = 0,\ n \geqslant 1$ *con condición inicial* $a_0 = 3$. *Despejando* $a_n$ *de la igualdad anterior, resulta* $a_n = -(4/10) \cdot a_{n-1}$. *Luego, la solución general es* $a_n = \alpha \cdot (-4/10)^n$. *El coeficiente* $\alpha$ *se obtiene haciendo cumplir la condición inicial para n=0:* $a_0 = \alpha \cdot (-4/10)^0 = 3$. *De aquí surge que* $\alpha = 3$.

*La solución es* $a_n = 3(-4/10)^n = 3(-1)^n 4^n/10^n$.

*El término* $a_4$ *es* $a_4 = 3(-4/10)^4 = 3 \cdot 256/10000 = 0{,}0768$. ■

Si la sucesión comienza desde $a_1$, la recurrencia puede estar dada de la siguiente forma: $a_n = c \cdot a_{n-1}$, $n \geqslant 2$, con condición inicial $a_1 = A$, la solución tiene la misma forma, $a_n = \alpha \cdot c^n$ y debe cumplirse: $a_1 = \alpha \cdot c^1 = \alpha \cdot c = A$, entonces $\alpha = A/c$. Por lo tanto, la solución general es $a_n = (A/c) \cdot c^n = A \cdot c^{n-1}$.

***Ejemplo* 2.16**
*Sea la recurrencia* $9a_n = a_{n-1}$ *para* $n \geqslant 2$, *con condición inicial* $a_1 = -1$. *Despejando* $a_n$ *resulta* $a_n = (1/9)a_{n-1}$. *Entonces la solución tiene la forma* $a_n = \alpha(1/9)^n = \alpha/9^n$.

*Debe cumplirse la condición inicial dada para n=1, así que:* $a_1 = \alpha/9^1 = -1$. *De aquí surge que* $\alpha = -9$. *Finalmente, la solución general es* $a_n = -9/9^n = -1/9^{n-1}$. ■

Debe notarse que la recurrencia también puede estar dada en la forma $a_{n+1} = c \cdot a_n$.

***Ejemplo* 2.17**
*Suponga que se tiene la recurrencia* $a_{n+1} = (3/5)a_n$ *para* $n \geqslant 0$, *con condición inicial* $a_0 = 1/5$. *La solución general puede plantearse:* $a_n = \alpha(3/5)^n$. *La condición inicial indica que* $a_0 = \alpha(3/5)^0 = 1/5$, *de donde* $\alpha = 1/5$. *La solución es:* $a_n = (1/5)(3/5)^n = 3^n/5^{n+1}$.

*La recurrencia dada expresa el término* $n+1$ *en función de un término anterior. No es correcto despejar* $a_n$, *llegar a* $a_n = (5/3)a_{n+1}$ *y dar la solución de la forma* $a_n = \alpha \cdot (5/3)^n$.

***Ejemplo* 2.18**
*Se realiza una experimentación sobre el tiempo que demora un determinado algoritmo en resolver un problema, conociendo el tamaño de la entrada (por ejemplo, si la entrada es un*

*vector de datos, el tamaño es la cantidad de datos, la dimensión del vector). Los resultados de la experimentación indican que si el tamaño de la entrada es 4, el algoritmo tarda 3 segundos, y si el tamaño es 5, consume 4.5 segundos. Además, se sabe que el tiempo $a_n$ que consume con una entrada n satisface una relación de RLCC homogénea de primer orden. Se desea predecir $a_n$ para cualquier n.*

*La recurrencia que rige la sucesión de tiempos es $a_n = c \cdot a_{n-1}$. Para $n = 5$, $a_5 = c \cdot a_4$, y entonces, de acuerdo a la experimentación, $4{,}5 = c \cdot 3$. Así surge que $c = 1{,}5$.*

*¿Cuánto tiempo consume el algoritmo si el tamaño de la entrada es 2? Se puede calcular, retrocediendo desde los datos, ya que se conoce la recurrencia: $a_n = 1{,}5 \cdot a_{n-1}$.*

*Con n=4: $a_4 = 1{,}5 \cdot a_3 \quad \rightarrow a_3 = 2$*

*Con n=3: $a_3 = 1{,}5 \cdot a_2 \quad \rightarrow a_2 = 4/3$.*

*La solución general es $a_n = A \cdot c^n = A \cdot 1{,}5^n$.*

*A debe ser $a_0$, pero no se conoce. Sin embargo, tomando $n = 4$, resulta $a_4 = 3 = A \cdot 1{,}5^4$, y $A = 3 \cdot 1{,}5^{(-4)}$. Así, $a_n = 3 \cdot 1{,}5^{n-4}$ es el tiempo que tarda el algoritmo para una entrada de tamaño n.* ■

### 2.2.2. Recurrencias lineales de orden 2

La recurrencia lineal homogénea con coeficientes constantes de segundo orden es de la forma

$$a_n = c_1\ a_{n-1} + c_2\ a_{n-2},\ n \geqslant 2$$

siendo $c_1$ y $c_2$ dos constantes dadas. Basados en lo que sucede en recurrencias de primer orden, se intuye que tiene soluciones de la forma $a_n = r^n$, para algún número $r$. ¿Existen valores $r$ tal que ésta sea la solución de la recurrencia de segundo orden? Si es así, $a_{n-1} = r^{n-1}$ y $a_{n-2} = r^{n-2}$. Reemplazando en la recurrencia:

$$\begin{array}{rcccc} a_n & = & c_1\ a_{n-1} & + & c_2\ a_{n-2} \\ r^n & = & c_1\ r^{n-1} & + & c_2\ r^{n-2} \end{array}$$

Dividiendo ambos miembros por $r^{n-2}$ queda:

$$r^2 = c_1\ r + c_2$$

La igualdad resultante es una ecuación cuadrática, llamada ECUACIÓN CARACTERÍSTICA de la recurrencia. De acuerdo al tipo de solución de la ecuación característica, se obtienen las soluciones de la recurrencia.

**Raíces distintas.** Si la ecuación cuadrática tiene dos raíces distintas, $r_1$ y $r_2$, la recurrencia tiene una solución que es la combinación lineal de las potencias de las dos raíces:

$$a_n = \alpha_1\ r_1^n + \alpha_2\ r_2^n$$

Los coeficientes $\alpha_1$ y $\alpha_2$ se obtienen usando las condiciones iniciales.

Si las condiciones iniciales son $a_0 = A$ y $a_1 = B$, entonces debe cumplirse:

$$\begin{array}{rcl} a_0 = \alpha_1\ r_1^0 + \alpha_2\ r_2^0 & = & A \\ a_1 = \alpha_1\ r_1^1 + \alpha_2\ r_2^1 & = & B \end{array}$$

Resolviendo ese sistema de dos ecuaciones lineales con 2 incógnitas ($\alpha_1$ y $\alpha_2$), se obtienen los valores de éstas. Veámoslo en ejemplos.

***Ejemplo* 2.19**

*Resolvamos la recurrencia* $a_n = 3a_{n-1} - 2a_{n-2}$, $n \geqslant 2$, *con condiciones iniciales* $a_0 = 1$; $a_1 = 0$. *Antes de hallar la solución general, veamos los siguientes términos de la sucesión:*

*para n=2,* $a_2 = 3a_1 - 2a_0 = 3 \cdot 0 - 2 \cdot 1 = -2$

*para n=3,* $a_3 = 3a_2 - 2a_1 = 3 \cdot (-2) - 2 \cdot 0 = -6$

*para n=4,* $a_4 = 3a_3 - 2a_2 = 3 \cdot (-6) - 2 \cdot (-2) = -14$

*para n=5,* $a_5 = 3a_4 - 2a_3 = 3 \cdot (-14) - 2 \cdot (-6) = -30$

*...*

*Veamos ahora la solución general. La ecuación característica es* $r^2 = 3r - 2$. *Las raíces son* $r_1 = 2$ *y* $r_2 = 1$. *La solución general es*

$$a_n = \alpha_1 2^n + \alpha_2 1^n$$

*Para hacer cumplir las condiciones iniciales, debe satisfacerse:*

$$\begin{aligned} a_0 = \alpha_1 2^0 + \alpha_2 1^0 &= 1 \\ a_1 = \alpha_1 2^1 + \alpha_2 1^1 &= 0 \end{aligned}$$

*La solución del sistema es* $\alpha_1 = -1$ *y* $\alpha_2 = 2$. *Entonces, la solución de la recurrencia es*

$$\begin{aligned} a_n &= -1 \cdot 2^n + 2 \cdot 1^n \\ &= -2^n + 2 \end{aligned}$$

*Para verificar, nótese que, por ejemplo,* $a_5 = -2^5 + 2 = -32 + 2 = -30$, *que coincide con el valor obtenido antes.* ■

***Ejemplo* 2.20**

*Dada la recurrencia de segundo orden* $a_n = (4/3)a_{n-2}$, *la ecuación característica es* $r^2 = 4/3$, *que tiene soluciones* $r_1 = 2/\sqrt{3}$ *y* $r_2 = -2/\sqrt{3}$. *La solución puede plantearse*

$$a_n = \alpha_1 \left(\frac{2}{\sqrt{3}}\right)^n + \alpha_2 \left(\frac{-2}{\sqrt{3}}\right)^n$$

*Si las condiciones iniciales son* $a_0 = -2$ *y* $a_1 = 0$, *los coeficientes se hallan resolviendo el sistema*

$$\begin{aligned} a_0 = \alpha_1 \left(\tfrac{2}{\sqrt{3}}\right)^0 + \alpha_2 \left(\tfrac{-2}{\sqrt{3}}\right)^0 &= \alpha_1 + \alpha_2 &= -2 \\ a_1 = \alpha_1 \left(\tfrac{2}{\sqrt{3}}\right)^1 + \alpha_2 \left(\tfrac{-2}{\sqrt{3}}\right)^1 &= \alpha_1 \tfrac{2}{\sqrt{3}} - \alpha_2 \tfrac{2}{\sqrt{3}} &= 0 \end{aligned}$$

*cuya solución es* $\alpha_1 = -1$ *y* $\alpha_2 = -1$.

*Entonces la solución de la recurrencia con las condiciones iniciales dadas es*

$$a_n = -\left(\frac{2}{\sqrt{3}}\right)^n - \left(\frac{-2}{\sqrt{3}}\right)^n$$

*La sucesión comienza con los términos* $a_0 = -2$, $a_1 = 0$, $a_2 = -8/3$, $a_3 = 0$, $a_4 = -32/9$, $a_5 = 0$. *Puede observarse que cuando n es impar,* $a_n = 0$, *y para n par de la forma* $n = 2j$, $a_n = -2\left(\frac{4}{3}\right)^j$ ■

***Ejemplo* 2.21**
*Resolvamos la recurrencia* $2a_n + a_{n-1} = a_{n-2}$, $n \geqslant 3$ *con condiciones iniciales* $a_1 = 2$, $a_2 = 1$. *Nótese que esta recurrencia describe una sucesión a partir de* $a_1$ *(no de* $a_0$*). Despejando* $a_n$, *la recurrencia se puede escribir* $a_n = -\frac{1}{2}a_{n-1} + \frac{1}{2}a_{n-2}$. *La ecuación característica es* $r^2 = -\frac{1}{2}r + \frac{1}{2}$, *o equivalentemente* $2r^2 + r - 1 = 0$. *Las raíces de la ecuación característica son* $r_1 = -1$ *y* $r_2 = \frac{1}{2}$. *La solución está dada entonces por*

$$a_n = \alpha_1(-1)^n + \alpha_2\left(\frac{1}{2}\right)^n$$

*Los coeficientes se determinan con las condiciones iniciales para* $n = 1$ *y* $n = 2$,

$$\begin{array}{rcl} a_1 & = & \alpha_1(-1)^1 + \alpha_2(\frac{1}{2})^1 = -\alpha_1 + \alpha_2/2 = 2 \\ a_2 & = & \alpha_1(-1)^2 + \alpha_2(\frac{1}{2})^2 = \alpha_1 + \alpha_2/4 = 1 \end{array}$$

*De aquí surgen los valores* $\alpha_1 = 0$ *y* $\alpha_2 = 4$. *Entonces, la solución de la recurrencia es*

$$a_n = 4\left(\frac{1}{2}\right)^n = \frac{4}{2^n}$$

***Ejemplo* 2.22**
*Consideremos la recurrencia* $a_{n+2} = 2a_{n+1} + 4a_n$, $n \geqslant 0$ *con condiciones iniciales* $a_0 = \frac{1}{2}$, $a_1 = \frac{1}{2}$. *La igualdad de la recurrencia dice: cada término* $(a_{n+2})$ *es igual al doble del término anterior* $(a_{n+1})$ *más cuatro veces el anterior al anterior* $(a_n)$. *La recurrencia es equivalente a*

$$a_n = 2a_{n-1} + 4a_{n-2} \quad n \geqslant 2$$

*La ecuación característica es* $r^2 = 2r + 4$, *o equivalentemente* $r^2 - 2r - 4 = 0$. *Las raíces de esta ecuación son* $r_1 = 1 + \sqrt{5}$ *y* $r_2 = 1 - \sqrt{5}$. *La solución está dada entonces por*

$$a_n = \alpha_1(1 + \sqrt{5})^n + \alpha_2(1 - \sqrt{5})^n$$

*Los coeficientes se determinan con las condiciones iniciales,*

$$\begin{array}{rcl} a_0 & = & \alpha_1(1 + \sqrt{5})^0 + \alpha_2(1 - \sqrt{5})^0 \\ a_1 & = & \alpha_1(1 + \sqrt{5})^1 + \alpha_2(1 - \sqrt{5})^1 \end{array} \begin{array}{rcl} & = & \alpha_1 + \alpha_2 \\ & = & \alpha_1(1 + \sqrt{5}) + \alpha_2(1 - \sqrt{5}) \end{array} \begin{array}{rcl} & = & \frac{1}{2} \\ & = & \frac{1}{2} \end{array}$$

*De aquí surgen los valores* $\alpha_1 = \frac{1}{4}$ *y* $\alpha_2 = \frac{1}{4}$. *Entonces, la solución de la recurrencia es*

$$a_n = \frac{1}{4}(1 + \sqrt{5})^n + \frac{1}{4}(1 - \sqrt{5})^n$$

***Ejemplo* 2.23**
*Se sabe que los primeros términos de una sucesión son* $a_1 = 1$, $a_2 = 2$, $a_3 = 1{,}2$, $a_4 = 1{,}28$. *También se sabe que la sucesión verifica una RLCC homogénea de orden 2. Se quiere hallar una expresión explícita para todos los términos de la sucesión.*

*La recurrencia es de la forma* $a_n = c_1a_{n-1} + c_2a_{n-2}$. *Haciendo* $n = 3$ *y* $n = 4$, *se llega a*

$$\begin{array}{rcl} a_3 = c_1a_2 + c_2a_1 & \rightarrow & 1{,}2 \\ a_4 = c_1a_3 + c_2a_2 & \rightarrow & 1{,}28 \end{array} \begin{array}{rcl} & = & 2c_1 + c_2 \\ & = & 1{,}2c_1 + 2c_2 \end{array}$$

*Estas dos ecuaciones nos permiten obtener los valores de los coeficientes de la recurrencia,* $c_1 = 0{,}4$, $c_2 = 0{,}4$.

*Para hallar la solución explícita de la recurrencia* $a_n = 0{,}4a_{n-1} + 0{,}4a_{n-2}$ *se calculan las raíces de la ecuación característica* $r^2 = 0{,}4r + 0{,}4$. *Las raíces son* $r_1 = 0{,}2 + 0{,}2\sqrt{11}$ *y* $r_1 = 0{,}2 - 0{,}2\sqrt{11}$.

*Se llega a que la solución es* $a_n = \alpha_1(0{,}2+0{,}2\sqrt{11})^n+\alpha_2(0{,}2-0{,}2\sqrt{11})^n$. *Los coeficientes* $\alpha_1$ *y* $\alpha_2$ *se pueden calcular reemplazando* $n$ *por 1 y por 2 en la solución general:*

$$\begin{aligned} a_1 = 1 &= \alpha_1(0{,}2 + 0{,}2\sqrt{11})^1 + \alpha_2(0{,}2 - 0{,}2\sqrt{11})^1 \\ a_2 = 2 &= \alpha_1(0{,}2 + 0{,}2\sqrt{11})^2 + \alpha_2(0{,}2 - 0{,}2\sqrt{11})^2 \end{aligned}$$

*Los valores de los coeficientes (aproximándolos con 4 cifras decimales) son* $\alpha_1 = 2{,}1508$ *y* $\alpha_2 = 1{,}8492$.

*Los términos de la sucesión, expresados en función de* $n$ *son* $a_n = 2{,}1508 \cdot (0{,}2 + 0{,}2\sqrt{11})^n + 1{,}8492 \cdot (0{,}2 - 0{,}2\sqrt{11})^n$, *para* $n \geqslant 1$. ■

***Ejemplo* 2.24**

*Se sabe que* $a_1 = 1$, $a_2 = -1$, $a_3 = 1/3$, $a_5 = 1/9$ *son términos de una sucesión que satisface una recurrencia lineal de segundo orden homogénea de la forma* $a_n = c_1a_{n-1} + c_2a_{n-2}$.

*Podríamos calcular* $c_1$ *y* $c_2$ *como en el ejemplo anterior. Pero en este caso no conocemos* $a_4$, *así que hay que hacer algunas cuentas más.*

*Reemplazando* $n$ *por 3, por 4 y por 5, se llega a:*

$$\begin{array}{lcccl} a_3 = c_1a_2 + c_2a_1 & \to & 1/3 & = & -c_1 + c_2 \\ a_4 = c_1a_3 + c_2a_2 & \to & a_4 & = & (1/3)c_1 - c_2 \\ a_5 = c_1a_4 + c_2a_3 & \to & 1/9 & = & a_4c_1 + (1/3)c_2 \end{array}$$

*Reemplazando* $a_4$ *en la última ecuación, resulta el sistema de ecuaciones*

$$\begin{array}{lcl} 1/3 & = & -c_1 + c_2 \\ 1/9 & = & (c_1/3 - c_2)c_1 + c_2/3 \end{array}$$

*De la primera ecuación surge que* $c_2 = 1/3 + c_1$, *y reemplazando en la segunda:* $1/9 = (c_1/3 - (1/3 + c_1))c_1 + 1/3(1/3 + c_1) = 1/9 - 2c_1^2/3$

*De aquí,* $c_1 = 0$ *y* $c_2 = 1/3$. *Luego, la recurrencia es* $a_n = (1/3)a_{n-2}$. ■

En caso de que las raíces sean complejas conjugadas, digamos $r_1 = a+ib$ y $r_2 = a-ib$, la solución tiene la misma forma:

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n = \alpha_1(a + ib)^n + \alpha_2(a - ib)^n$$

***Ejemplo* 2.25**

*Resolvamos la recurrencia* $4a_n + a_{n-2} = 0$, *que es equivalente a* $a_n = -\frac{1}{4}a_{n-2}$; *con condiciones iniciales* $a_0 = -1$, $a_1 = 1$.

*La ecuación característica es* $r^2 = -\frac{1}{4}$, *que tiene raíces* $r_1 = \frac{i}{2}$ *y* $r_2 = -\frac{i}{2}$.

*La solución de la recurrencia es*

$$a_n = \alpha_1 \left(\frac{i}{2}\right)^n + \alpha_2 \left(-\frac{i}{2}\right)^n$$

*los coeficientes $\alpha_1$ y $\alpha_2$ se calculan con las condiciones iniciales,*

$$\begin{array}{rcl} a_0 & = & \alpha_1 \left(\frac{i}{2}\right)^0 + \alpha_2 \left(-\frac{i}{2}\right)^0 \quad = \quad \alpha_1 + \alpha_2 = -1 \\ a_1 & = & \alpha_1 \left(\frac{i}{2}\right)^1 + \alpha_2 \left(-\frac{i}{2}\right)^1 \quad = \quad i\frac{\alpha_1}{2} - i\frac{\alpha_2}{2} = 1 \end{array}$$

*de donde surge que $\alpha_1 = -\frac{1+2i}{2}$ y $\alpha_2 = -\frac{1-2i}{2}$. Finalmente, la solución de la recurrencia es*

$$a_n = -\frac{1+2i}{2}\left(\frac{i}{2}\right)^n - \frac{1-2i}{2}\left(-\frac{i}{2}\right)^n$$

*Si estamos interesados, por ejemplo, en el término $a_6$ de la sucesión, se calcula de la siguiente manera,*

$$\begin{array}{rcl} a_6 & = & -\frac{1+2i}{2}\left(\frac{i}{2}\right)^6 - \frac{1-2i}{2}\left(-\frac{i}{2}\right)^6 \\ & = & -\frac{1+2i}{2}\left(\frac{-1}{64}\right) - \frac{1-2i}{2}\left(\frac{-1}{64}\right) \\ & = & \frac{1}{64} \end{array}$$

***Ejemplo* 2.26**
*La recurrencia del ejemplo 2.10 es $a_n = a_{n-1} - a_{n-2}$, que tiene ecuación característica $r^2 - r + 1 = 0$, cuyas raíces son complejas conjugadas: $r_1 = \frac{1}{2} + i\frac{\sqrt{3}}{2}$ y $r_2 = \frac{1}{2} - i\frac{\sqrt{3}}{2}$. Por lo tanto, la solución tiene la forma*

$$a_n = \alpha_1 \left(\frac{1}{2} + i\frac{\sqrt{3}}{2}\right)^n + \alpha_2 \left(\frac{1}{2} - i\frac{\sqrt{3}}{2}\right)^n$$

*Imponiendo las condiciones iniciales dadas en el ejemplo 2.10, se llega al sistema de ecuaciones para $\alpha_1$ y $\alpha_2$*

$$\begin{array}{rcl} a_0 & = & \alpha_1 \left(\frac{1}{2} + i\frac{\sqrt{3}}{2}\right)^0 + \alpha_2 \left(\frac{1}{2} - i\frac{\sqrt{3}}{2}\right)^0 \\ & = & \alpha_1 + \alpha_2 = 0 \\ a_1 & = & \alpha_1 \left(\frac{1}{2} + i\frac{\sqrt{3}}{2}\right)^1 + \alpha_2 \left(\frac{1}{2} - i\frac{\sqrt{3}}{2}\right)^1 \\ & = & \alpha_1 \left(\frac{1}{2} + i\frac{\sqrt{3}}{2}\right) + \alpha_2 \left(\frac{1}{2} - i\frac{\sqrt{3}}{2}\right) = 1 \end{array}$$

*de donde surgen los valores $\alpha_1 = -\frac{i}{\sqrt{3}}$ y $\alpha_2 = \frac{i}{\sqrt{3}}$. Luego, la solución de la recurrencia con las condiciones iniciales dadas es*

$$a_n = -\frac{i}{\sqrt{3}}\left(\frac{1}{2} + i\frac{\sqrt{3}}{2}\right)^n + \frac{i}{\sqrt{3}}\left(\frac{1}{2} - i\frac{\sqrt{3}}{2}\right)^n$$

*Verifiquemos que esta fórmula explícita nos da los valores de la sucesión correcta:*

$$a_0 = -\frac{i}{\sqrt{3}}\left(\frac{1}{2}+i\frac{\sqrt{3}}{2}\right)^0 + \frac{i}{\sqrt{3}}\left(\frac{1}{2}-i\frac{\sqrt{3}}{2}\right)^0 = 0$$
$$a_1 = -\frac{i}{\sqrt{3}}\left(\frac{1}{2}+i\frac{\sqrt{3}}{2}\right)^1 + \frac{i}{\sqrt{3}}\left(\frac{1}{2}-i\frac{\sqrt{3}}{2}\right)^1 = -\frac{i}{2\sqrt{3}} + \frac{1}{2} + \frac{i}{2\sqrt{3}} + \frac{1}{2} = 1$$
$$a_2 = -\frac{i}{\sqrt{3}}\left(\frac{1}{2}+i\frac{\sqrt{3}}{2}\right)^2 + \frac{i}{\sqrt{3}}\left(\frac{1}{2}-i\frac{\sqrt{3}}{2}\right)^2 = -\frac{i}{\sqrt{3}}\left(\left(\frac{-1}{2}+i\frac{\sqrt{3}}{2}\right)-\left(\frac{-1}{2}-i\frac{\sqrt{3}}{2}\right)\right) = 1$$
$$a_3 = -\frac{i}{\sqrt{3}}\left(\frac{1}{2}+i\frac{\sqrt{3}}{2}\right)^3 + \frac{i}{\sqrt{3}}\left(\frac{1}{2}-i\frac{\sqrt{3}}{2}\right)^3 = -\frac{i}{\sqrt{3}}\left(-1-(-1)\right) = 0$$
$$a_4 = -\frac{i}{\sqrt{3}}\left(\frac{1}{2}+i\frac{\sqrt{3}}{2}\right)^4 + \frac{i}{\sqrt{3}}\left(\frac{1}{2}-i\frac{\sqrt{3}}{2}\right)^4 = -\frac{i}{\sqrt{3}}\left(\left(\frac{-1}{2}-i\frac{\sqrt{3}}{2}\right)-\left(\frac{-1}{2}+i\frac{\sqrt{3}}{2}\right)\right) = -1$$
...

*Los valores hallados coinciden con los de la sucesión del ejemplo 2.10.* ■

**Raíces iguales.** Si la ecuación cuadrática tiene dos raíces iguales, llamémosla $r_1$, la recurrencia tiene una solución de la forma:

$$a_n = (\alpha_1 + n \cdot \alpha_2) \cdot r_1^n$$

Los coeficientes $\alpha_1$ y $\alpha_2$ se obtienen usando las condiciones iniciales.

Si las condiciones iniciales son $a_0 = A$ y $a_1 = B$, entonces debe cumplirse:

$$\begin{aligned} a_0 = (\alpha_1 + 0 \cdot \alpha_2) \cdot r_1^0 &= A \\ a_1 = (\alpha_1 + 1 \cdot \alpha_2) \cdot r_1^1 &= B \end{aligned}$$

Resolviendo ese sistema de dos ecuaciones lineales con 2 incógnitas ($\alpha_1$ y $\alpha_2$), se obtienen los valores de éstas.

***Ejemplo* 2.27**

*Dada la recurrencia $a_n = 6a_{n-1} - 9a_{n-2}$, $n \geqslant 2$, la ecuación característica correspondiente es $r^2 - 6r + 9 = 0$, que tiene una raíz doble $r_1 = 3$. La solución es $a_n = (\alpha_1 + n \cdot \alpha_2) \cdot 3^n$.*

*Teniendo condiciones iniciales $a_0 = -\frac{1}{5}$ y $a_1 = \frac{3}{5}$, los coeficientes se encuentran resolviendo el sistema:*

$$\begin{aligned} a_0 = (\alpha_1 + 0 \cdot \alpha_2) \cdot 3^0 &= \alpha_1 = -\tfrac{1}{5} \\ a_1 = (\alpha_1 + 1 \cdot \alpha_2) \cdot 3^1 &= 3\alpha_1 + 3\alpha_2 = \tfrac{3}{5} \end{aligned}$$

*De allí surge que $\alpha_1 = -\frac{1}{5}$ y $\alpha_1 = \frac{2}{5}$, y la solución de la recurrencia es $a_n = (-\frac{1}{5} + \frac{2}{5} \cdot n) \cdot 3^n$. Los primeros términos de la sucesión son:*

$$a_0 = (-\tfrac{1}{5} + \tfrac{2}{5} \cdot 0) \cdot 3^0 = -\tfrac{1}{5}$$
$$a_1 = (-\tfrac{1}{5} + \tfrac{2}{5} \cdot 1) \cdot 3^1 = -\tfrac{3}{5} + \tfrac{6}{5} = \tfrac{3}{5}$$
$$a_2 = (-\tfrac{1}{5} + \tfrac{2}{5} \cdot 2) \cdot 3^2 = -\tfrac{9}{5} + \tfrac{36}{5} = \tfrac{27}{5}$$
$$a_3 = (-\tfrac{1}{5} + \tfrac{2}{5} \cdot 3) \cdot 3^3 = -\tfrac{27}{5} + \tfrac{162}{5} = \tfrac{135}{5} = 27;$$ ■

***Ejemplo* 2.28**

*Dada la recurrencia $2a_{n+2} - 4a_{n+1} = -2a_n$, $n \geqslant 0$ la ecuación característica correspondiente es $2r^2 - 4r + 2 = 0$, o equivalentemente $r^2 - 2r + 1 = 0$ que tiene una raíz doble $r_1 = 1$. La solución es*

$$a_n = (\alpha_1 + n \cdot \alpha_2) \cdot 1^n = \alpha_1 + n \cdot \alpha_2$$

*Teniendo condiciones iniciales* $a_0 = 11$ *y* $a_1 = 7$*, los coeficientes se encuentran resolviendo el sistema:*

$$\begin{aligned} a_0 = \alpha_1 + 0 \cdot \alpha_2 &= 11 \\ a_1 = \alpha_1 + 1 \cdot \alpha_2 &= 7 \end{aligned}$$

*De allí surge que* $\alpha_1 = 11$ *y* $\alpha_1 = -4$*, y la solución de la recurrencia es* $a_n = 11 - 4n$*.*

*Los primeros términos de la sucesión son* $a_0 = 11$*,* $a_1 = 11 - 4 = 7$*,* $a_2 = 11 - 4 \cdot 2 = 3$*,* $a_3 = 11 - 4 \cdot 2 = -1$*,* $a_4 = 11 - 4 \cdot 4 = -5$*, etc.* ■

### 2.2.3. Orden superior

Una recurrencia lineal homogénea de orden $k$ tiene la forma $a_n = c_1 a_{n-1} + c_2 a_{n-2} + ... + c_k a_{n-k}$. Las soluciones serán una combinación lineal de soluciones del tipo $r^n$, donde $r$ es raíz de la ecuación característica $r^k = c_1 r^{k-1} + c_2 r^{k-2} + ... + c_k$. Si la ecuación característica tiene $k$ raíces distintas $r_1$, $r_2$, ..., $r_k$, la solución es

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + ... + \alpha_k r_k^n$$

donde los coeficientes $\alpha_i$ se obtienen de $k$ condiciones iniciales: $a_0 = A_0$, $a_1 = A_1$, ..., $a_{k-1} = A_{k-1}$.

Si una raíz $r$ tiene multiplicidad $m$, mayor que 1, en la solución se considera $r^n$ multiplicado por un polinomio de $n$, de orden $m - 1$.

## 2.3. Resolución de recurrencias lineales no homogéneas

Dada la recurrencia lineal no homogénea

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + ... + c_k a_{n-k} + g(n)$$

la recurrencia homogénea asociada es

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + ... + c_k a_{n-k}$$

es decir, la recurrencia homogénea que se obtiene de eliminar el término no homogéneo $g(n)$.

La solución de la recurrencia no homogénea es la suma de una solución particular $a_n^p$ más una solución de la recurrencia homogénea asociada $a_n^h$. La solución queda entonces

$$a_n = a_n^h + a_n^p$$

Ya hemos visto antes cómo obtener soluciones de recurrencias lineales, ahora nos centraremos en hallar una solución particular de la recurrencia lineal no homogénea.

Como solución particular se propone una sucesión $a_n^p$ que cumple la recurrencia. En general, esta solución particular tiene la misma forma funcional que el término no homogéneo. Es decir, si el término no homogéneo es $e^n$, la solución particular tiene la misma

forma, $a_n^p = A\,e^n$, donde $A$ es una constante a determinar. En la siguiente tabla se muestra la correspondencia entre el término no homogéneo y la solución particular para ciertos casos[1].

| $g(n)$ | $a_n^p$ |
|---|---|
| constante | constante |
| $n$ | $An + B$ |
| $n^2$ | $An^2 + Bn + C$ |
| $\operatorname{sen} n$ | $A \operatorname{sen} n + B \cos n$ |
| $t^n$ | $At^n$ |

***Ejemplo* 2.29**
*Sea la recurrencia* $a_n = 2a_{n-1} + 1$, $n \geqslant 1$, $a_0 = 0$. *El término no homogéneo es 1. Entonces se propone una solución particular constante,* $a_n^p = A$ . *La solución particular debe satisfacer la recurrencia. Como* $a_{n-1}^p = A$, *reemplazando en la recurrencia:*

$$\begin{aligned} a_n^p &= 2a_{n-1}^p + 1 \\ A &= 2A + 1 \end{aligned}$$

*de donde surge* $A = -1$.

*La solución de la recurrencia homogénea asociada es* $a_n^h = \alpha\, 2^n$. *Luego, la solución es* $a_n = a_n^h + a_n^p = \alpha 2^n - 1$. *Con la condición inicial,* $a_0 = \alpha 2^0 - 1 = 0$, *entonces* $\alpha = 1$, *por lo que*

$$a_n = 2^n - 1$$

***Ejemplo* 2.30**
*Sea la recurrencia lineal no homogénea* $a_n + 2a_{n-1} = 5 \cdot 3^n$, $n \geqslant 1$, *con condición inicial* $a_0 = -1$. *El término no homogéneo es* $5 \cdot 3^n$. *La recurrencia homogénea asociada es* $a_n + 2a_{n-1} = 0$. *La solución de ésta es* $a_n^h = \alpha \cdot (-2)^n$.

*Dado que el término no homogéneo tiene forma de potencia de 3, se propone* $a_n^p = A \cdot 3^n$ *como solución particular. El coeficiente A se calcula de tal modo que se cumpla la recurrencia. Entonces, se reemplaza la solución particular en la recurrencia, notando que* $a_{n-1}^p = A \cdot 3^{n-1}$. *Entonces, debe ser:*

$$\begin{aligned} a_n^p + 2a_{n-1}^p &= 5 \cdot 3^n \\ A \cdot 3^n + 2A \cdot 3^{n-1} &= 5 \cdot 3^n \end{aligned}$$

*Dividiendo por* $3^{n-1}$, *resulta:* $3A + 2A = 5 \cdot 3$, *de donde surge que* $A = 3$. *La solución particular queda* $a_n^p = 3 \cdot 3^n = 3^{n+1}$.

*Entonces, la solución de la recurrencia no homogénea dada es* $a_n = a_n^h + a_n^p = \alpha \cdot (-2)^n + 3^{n+1}$. *El coeficiente* $\alpha$ *se calcula con la condición inicial.*

$a_0 = a_0^h + a_0^p = \alpha \cdot (-2)^0 + 3^1 = \alpha + 3 = -1$, *entonces* $\alpha = -4$. *La solución resulta*

$$a_n = -4(-2)^n + 3^{n+1}$$

[1]La última fila de la tabla vale en el caso que $t$ no sea raíz de la ecuación característica. En otro caso, si $t$ es raíz de la ecuación característica con multiplicidad $m$, la solución particular propuesta debe ser un polinomio de $n$ de grado $m - 1$

***Ejemplo* 2.31**
*Considere la recurrencia lineal de segundo orden no homogénea*

$$a_n = 4a_{n-1} - a_{n-1} + 2n$$

*La recurrencia homogénea asociada es* $a_n = 4a_{n-1} - a_{n-1}$*, que tiene ecuación característica* $r^2 = 4r - 1$*, cuyas soluciones son* $r_1 = 2 + \sqrt{3}$ *y* $r_2 = 2 - \sqrt{3}$*. La solución de la recurrencia homogénea es* $a_n^h = \alpha_1(2 + \sqrt{3})^n + \alpha_2(2 - \sqrt{3})^n$*.*

*El término no homogéneo es* $2n$*. Por lo tanto, se propone una solución particular de la forma* $a_n^p = An + B$*. Entonces,* $a_{n-1}^p = A(n-1) + B$ *y* $a_{n-2}^p = A(n-2) + B$*. Reemplazando en la recurrencia:*

$$\begin{aligned} a_n^p &= 4a_{n-1}^p - a_{n-2}^p + 2n \\ An + B &= 4(A(n-1) + B) - (A(n-2) + B) + 2n \\ 0 &= (2A + 2)n - 2A + 2B \end{aligned}$$

*Esa igualdad debe cumplirse para todo* $n \geqslant 2$*, entonces debe ser* $2A + 2 = 0$ *y* $-2A + 2B = 0$ *, de donde se llega a que* $A = -1$*,* $B = -1$*. Así, la solución particular es* $a_n^p = -n - 1$*.*

*La solución general es*

$$a_n = a_n^h + a_n^p = \alpha_1(2 + \sqrt{3})^n + \alpha_2(2 - \sqrt{3})^n - n - 1$$

*Si las condiciones iniciales son* $a_0 = -1$ *y* $a_1 = 1$*, debe cumplirse:*

$$\begin{aligned} a_0 &= \alpha_1(2 + \sqrt{3})^0 + \alpha_2(2 - \sqrt{3})^0 - 0 - 1 = \alpha_1 + \alpha_2 - 1 = -1 \\ a_1 &= \alpha_1(2 + \sqrt{3})^1 + \alpha_2(2 - \sqrt{3})^1 - 1 - 1 = \alpha_1(2 + \sqrt{3}) + \alpha_2(2 - \sqrt{3}) - 2 = 1 \end{aligned}$$

*La solución de este sistema de ecuaciones es* $\alpha_1 = \sqrt{3}/2$ *y* $\alpha_2 = -\sqrt{3}/2$*. La solución de la recurrencia, con las condiciones iniciales dadas es*

$$a_n = \frac{\sqrt{3}}{2}(2 + \sqrt{3})^n - \frac{\sqrt{3}}{2}(2 - \sqrt{3})^n - n - 1$$

## 2.4. Problemas modelados con recurrencias

### 2.4.1. Modelos de crecimiento

Considere que se quiere estudiar la evolución de la cantidad de cuentas abiertas en una red social. Después de estudios estadísticos se sabe que cada año, el número de cuentas en la red crece un 12 % respecto del año anterior.

Si $a_n$ denota el número de cuentas en el año $n$, el dato es que $a_n = a_{n-1} + 0,12a_{n-1} = 1,12a_{n-1}$.

La solución de esta recurrencia es de la forma $a_n = \alpha \cdot (1{,}12)^n$. Si en el año 1 hay A cuentas abiertas, $a_1 = A = \alpha \cdot (1{,}12)^1$, de donde $\alpha = A/1,12$. (Nótese que en este caso no tiene sentido pensar en $a_0$, ya que se considera que la red comienza a funcionar en el año 1). Luego, $a_n = A \cdot (1,12)^{n-1}$.

Si se quiere determinar cuándo la red superará los mil usuarios, dado que el primer año tiene 96, se debe determinar $n$ tal que $a_n > 1000$, siendo $A = 96$. Entonces:

$$\begin{aligned} 96(1{,}12)^{n-1} &> 1000 \\ (1{,}12)^{n-1} &> \frac{1000}{96} \\ (n-1)\ln(1{,}12) &> \ln\left(\frac{1000}{96}\right) \\ n &> \ln\left(\frac{1000}{96}\right)/\ln(1{,}12) + 1 \end{aligned}$$

Por lo tanto, $n$ debe ser mayor que 21.678. Esto indica que durante el año 21, el número de usuarios superará los 1000.

Este es el resultado matemático. Sin embargo, se podría cuestionar si la predicción de crecimiento será válida luego de tanto tiempo, y si la recurrencia sigue valiendo para esos $n$. Pero eso es otro tema.

El modelo anterior predice que a medida que pasa el tiempo, la cantidad de cuentas crece y crece sin límite (es decir, tiende a $\infty$ cuando $n$ tiende a $\infty$), situación que no puede ser real.

Otro modelo de crecimiento de poblaciones es el conocido como *modelo logístico*. Éste supone que existe un máximo posible para el tamaño de la población (dado por limitaciones de espacio o recursos). Así, en cada etapa, la población $a_n$ será en cierta forma proporcional a la población en la etapa anterior ($a_{n-1}$) pero también proporcional a lo que falte para alcanzar el máximo ($M - a_{n-1}$), donde M es el máximo que puede alcanzar la población. El modelo puede escribirse

$$a_n = b \cdot a_{n-1} \cdot (M - a_{n-1})$$

o, equivalentemente, $a_n = b \cdot M \cdot a_{n-1} - b \cdot a_{n-1}^2$. Esta es una recurrencia no lineal, ya que un término de la serie está elevado al cuadrado.

### 2.4.2. Préstamo con pagos parciales

Suponga que se recibe un préstamo de \$C, que debe ser devuelto en cuotas fijas mensuales con una tasa de interés mensual de $I$. Las cuotas son de \$P.

Si $a_n$ es el dinero adeudado al finalizar el mes $n$, este monto es igual a lo adeudado el mes anterior, más los intereses generados, menos el pago realizado. Es decir:

$$\begin{aligned} a_n &= a_{n-1} + I \cdot a_{n-1} - P \\ &= (1+I) \cdot a_{n-1} - P \end{aligned}$$

Se obtiene una relación de recurrencia lineal no homogénea de primer orden. La solución es $a_n = a_n^h + a_n^p$, donde $a_n^h$ es la solución de la recurrencia homogénea asociada $a_n = (1+I) \cdot a_{n-1}$, y $a_n^p$ es una solución particular. En este caso, como el término no homogéneo es constante, se propone una solución particular constante.

Entonces $a_n^h = \alpha \cdot (1+I)^n$ y $a_n^p = A$. Debe cumplirse la recurrencia para la solución particular,

$$\begin{aligned} a_n^p &= (1+I)\cdot a_{n-1}^p - P \\ A &= (1+I)\cdot A - P \end{aligned}$$

de donde surge que $A = \frac{P}{I}$. Luego,

$$a_n = \alpha \cdot (1+I)^n + \frac{P}{I}$$

Usando la condición inicial $a_0 = C$ (porque al inicio de la inversión lo adeudado es el monto del préstamo), tenemos que $a_0 = C = \alpha \cdot (1+I)^0 + \frac{P}{I}$. Despejando, $\alpha = C - \frac{P}{I}$, y la solución queda

$$a_n = \left(C - \frac{P}{I}\right)(1+I)^n + \frac{P}{I}$$

¿En cuántos meses quedará saldada la deuda? Nos preguntamos hasta qué $n$ la deuda será positiva, es decir, $\left(C - \frac{P}{I}\right)(1+I)^n + \frac{P}{I} > 0$.

$$\begin{aligned} \left(C - \frac{P}{I}\right)(1+I)^n + \frac{P}{I} &> 0 \\ \frac{P}{I} &> \left(\frac{P}{I} - C\right)(1+I)^n \\ \frac{P}{I} \cdot \frac{1}{\left(\frac{P}{I} - C\right)} &> (1+I)^n \\ \ln\left(\frac{P}{P - I\cdot C}\right) &> n \cdot \ln(1+I) \\ \ln\left(\frac{P}{P - I\cdot C}\right) / \ln(1+I) &> n \end{aligned}$$

Si $C$=1000, $P$=70, $I = 0{,}05$, la desigualdad anterior es $25{,}67 > n$. Entonces, al mes 26 la deuda queda saldada.

Observe que debe cumplirse que $P - I \cdot C > 0$ y que $I > 0$. ¿Qué sucede si estas desigualdades no se cumplen?

### 2.4.3. Operaciones necesarias en ordenamiento por burbuja

Un algoritmo muy conocido para ordenar una lista L de $n$ números es

```
para i=1 a n-1
   para j=n a i+1
      si L(j)<L(j-1)
         aux=L(j-1)
         L(j-1)=L(j)
         L(j)=aux
      fin si
   fin para
fin para
```

El algoritmo consta de dos bucles. En el bucle *para* interno se comparan, comenzando desde el último, un término de la lista con el anterior. Si el anterior es mayor, se intercambian sus valores. En el primer paso del bucle exterior, el menor número de la lista "sube"(como una burbuja ascendiendo en un fluído) hasta la primera posición. En el segundo paso del ciclo exterior, el segundo menor número sube hasta la segunda posición, y así sucesivamente.

Estamos interesados en contar la cantidad de comparaciones que debe realizar el algoritmo para ordenar una lista de $n$ números. Ésta es una medida de la complejidad del algoritmo. Denotemos $a_n$ la cantidad de comparaciones que realiza el algoritmo para ordenar $n$ números. Se puede observar que en el primer bucle *para* se realizan $(n-1)$ comparaciones, y con ésto, el menor número toma su posición. Después el algoritmo ordena los restantes números, que son $(n-1)$ números. Para ésto, realiza $a_{n-1}$ comparaciones. Entonces, se deduce que debe cumplirse

$$a_n = a_{n-1} + n - 1, \quad n \geqslant 2$$

Si la lista tiene 1 elemento, el algoritmo la ordena en $a_1 = 0$ comparaciones. Los primeros términos de la sucesión así descrita son:
$a_1 = 0$
$a_2 = 0 + (2-1) = 1$
$a_3 = 1 + (3-1) = 1 + 2 = 3$
$a_4 = 3 + (4-1) = 1 + 2 + 3 = 6$
$a_5 = 6 + (5-1) = 1 + 2 + 3 + 4 = 10$
...
$a_n = 1 + 2 + 3 + ... + (n-1) = n(n-1)/2$

Entonces, $a_n = (n^2 - n)/2$. Luego, para ordenar, por ejemplo, una lista de 10 números se realizan $a_{10} = (10^2 - 10)/2 = 90/2 = 45$ comparaciones.

Esta medida de complejidad del algoritmo nos dice que requiere una cantidad de comparaciones del orden del cuadrado del tamaño de la entrada. Esto se denota $O(n^2)$.

Nótese que lo analizado anteriormente es independiente del orden de los números en la lista dada. Es decir, en una lista de 10 números, el algoritmo realiza 45 comparaciones en cualquier caso, aún si los 10 números son dados en forma ordenada.

### 2.4.4. Construcciones geométricas recursivas (fractales)

#### Conjunto de Cantor

Considere un proceso geométrico tal que, a partir de un segmento, realiza una operación recursivamente. La operación considerada es la siguiente: a cada segmento resultante en la etapa anterior, se lo divide en tres partes de igual longitud y se elimina la parte central.

Si el proceso comienza con un segmento de longitud 1, luego del primer paso se tienen dos segmentos cuyas longitudes suman 2/3 (véase figura 2.1).

En la etapa siguiente (etapa 2) se aplica la operación a los dos segmentos resultantes, se eliminan de cada uno su tercio central, es decir, se eliminan dos segmentos de longitud 1/9 cada uno. La suma de las longitudes de los segmentos que quedan es 2/3-2/9 = 4/9.


Figura 2.1: Construcciones geométricas recursivas: Conjunto de Cantor

Si en la etapa $n$ la suma de las longitudes de los segmentos es $a_n$, la transformación siguiente elimina 1/3 de cada segmento, entonces $a_{n+1} = a_n - \frac{1}{3}a_n = \frac{2}{3}a_n$. Si se comienza con un segmento de longitud 1 ($a_0 = 1$), entonces, en la etapa $n$ se tienen segmentos con una longitud total $a_n = \left(\frac{2}{3}\right)^n$.

Puede observarse que cuando $n$ tiene a $\infty$, la longitud total tiende a cero.

Después de infinitas etapas, se obtiene un conjunto de puntos, conocido como *conjunto de Cantor*. Este conjunto tiene asombrosas propiedades. Por un lado, existen infinitos puntos en el conjunto de Cantor. Obsérvese que los puntos 0, 1/3, 2/3, 1/9, 2/9, etc. nunca son eliminados, por lo tanto, después de infinitas operaciones, estos puntos siguen estando en el conjunto. Pero como ya se dijo, la longitud de este conjunto es 0. Por lo tanto, no contiene ningún segmento. Sin embargo, tiene tantos puntos como el conjunto de números reales.

Por otro lado, tiene la propiedad de autosimilitud: si miramos con lupa una parte del conjunto, será similar al conjunto entero. Es decir, una parte es similar al todo.

Otra característica interesante de mencionar es su dimensión. Se sabe que un segmento tiene dimensión 1, un punto tiene dimensión 0. El conjunto de Cantor no es un segmento (ni un conjunto de segmentos)y no es un punto (ni un conjunto finito de ellos). Entonces su dimensión no es 1 ni es 0. La dimensión [2] del conjunto de Cantor es $\frac{\ln 2}{\ln 3} \approx 0{,}6309$.

Las propiedades de autosimilitud y dimensión fraccionaria son características de los fractales. Éstos son objetos geométricos que escapan de la geometría euclidiana.

### Curva de Koch

Consideremos otro proceso geométrico similar. Se realiza en cada paso la siguiente transformación: a cada uno de los segmentos, se lo divide en tres segmentos de igual longitud, y el segmento central se lo reemplaza por dos segmentos de igual longitud, formando una *carpa* (figura 2.2).

[2] El concepto de dimensión que se aplica a estos objetos geométricos es la dimensión de Hausdorff

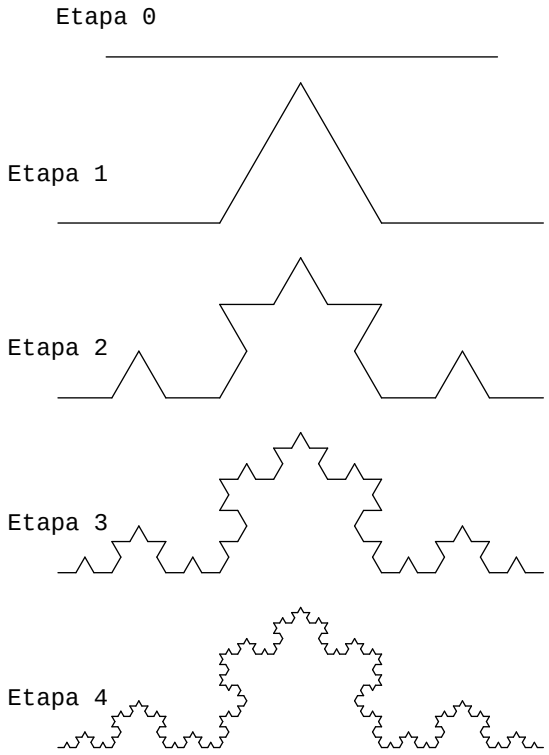



Figura 2.2: Construcciones geométricas recursivas: Curva de Koch

Aquí, de un segmento de una determinada longitud, luego de la transformación, quedan 4 segmentos de igual longitud, que es 1/3 de la longitud del segmento que los originó. Entonces, siendo $a_n$ la suma de las longitudes de todos los segmentos en la etapa n, se cumple la recurrencia $a_{n+1} = \frac{4}{3}a_n$. Comenzando con un segmento de longitud 1 (es decir, $a_0 = 1$), se obtiene la solución $a_n = \left(\frac{4}{3}\right)^n$, que indica la longitud de la curva resultante en el paso $n$.

Cuando $n$ tiende a $\infty$, la longitud de la curva tiende a $\infty$. La curva resultante en el límite se conoce con el nombre de *curva de Koch*. Es una curva acotada de longitud infinita. Ésta tiene la particularidad de ser una curva continua pero ninguno de sus puntos admite una tangente (no es suave en ningún punto).

La dimensión fractal de la curva de Koch es $\frac{\ln 4}{\ln 3} \approx 1{,}2619$. Intuitivamente, esto indica que cubre más que una curva, pero menos que una superficie.

Nótese que este objeto geométrico también tiene la propiedad de autosimilitud y dimensión fraccionaria.

### 2.4.5. Expresiones aritméticas válidas

Considere que se quiere contar la cantidad de expresiones aritméticas válidas que usan $n$ símbolos (dígitos o signos de operación). Para simplificar el análisis, consideremos que se tienen 10 dígitos (del 0 al 9) y dos signos de operación, + y *. Expresiones válidas son

45+2*3, 1+0+9*4+1; mientras que no son válidas las expresiones *87+1, 12*+9, etc.

Sea $a_n$ el número de expresiones aritméticas válidas con $n$ símbolos. Claramente $a_1 = 10$, ya que las expresiones válidas de un símbolo son los dígitos de 0 a 9. $a_2 = 100$, ya que las expresiones de dos símbolos son 00, 01, ..., 10, 11, ...98, 99 [3].

Analicemos las cadenas de tres símbolos. Si los dos últimos símbolos en una cadena de tres son dos dígitos, entonces tal expresión puede formarse a partir de una expresión de dos símbolos, agregándole un dígito de 0 a 9. El total de estas expresiones es $10\,a_2$. Si los dos últimos dígitos de una expresión aritmética de tres símbolos es un signo y un dígito, tal expresión puede obtenerse de una expresión de un símbolo, agregándole un signo (+,*) y un dígito (0,1,...,9). Entonces, la candidad de estas expresiones es $20\,a_1$. Finalmente, hemos obtenido que $a_3 = 10\,a_2 + 20\,a_1 = 10 \cdot 100 + 20 \cdot 10 = 1200$.

El análisis recién expuesto se puede extender al caso general de expresiones de $n$ símbolos. Las expresiones de $n$ símbolos que terminan en dos dígitos se pueden contar notando que se pueden obtener agregando un dígito a las expresiones de $n-1$ símbolos. Y las expresiones de $n$ símbolos que terminan en un signo y un dígito se pueden obtener agregando un signo y un dígito a las expresiones de $n-2$ símbolos. Luego, $a_n = 10a_{n-1} + 20a_{n-2}$, y esto vale para $n \geqslant 3$.

Resolvamos la recurrencia. La ecuación característica es $r^2 - 10r - 20 = 0$, cuyas raíces son $r_1 = 5 + 3\sqrt{5}$ y $r_2 = 5 - 3\sqrt{5}$. Por lo tanto, la solución tiene la forma $a_n = \alpha_1(5 + 3\sqrt{5})^n + \alpha_2(5 - 3\sqrt{5})^n$. Los coeficientes se obtienen con las condiciones iniciales para $n = 1$ y $n = 2$:

$$\begin{array}{rcrcl} a_1 & = & \alpha_1(5 + 3\sqrt{5}) + \alpha_2(5 - 3\sqrt{5}) & = & 10 \\ a_2 & = & \alpha_1(5 + 3\sqrt{5})^2 + \alpha_2(5 - 3\sqrt{5})^2 & = & 100 \end{array}$$

De aquí surge que $\alpha_1 = \sqrt{5}/3$ y $\alpha_2 = -\sqrt{5}/3$. Por lo tanto, el número de expresiones aritméticas válidas de $n$ símbolos es

$$a_n = \frac{\sqrt{5}}{3}(5 + 3\sqrt{5})^n - \frac{\sqrt{5}}{3}(5 - 3\sqrt{5})^n$$

(Sí, aunque parezca difícil de creer, para cada $n$ esa expresión es un número natural)

### 2.4.6. Conteo de cadenas binarias

Suponga que se quiere contar la cantidad de cadenas de 0 y 1 de longitud $n$ que no tienen dos 1 consecutivos. Sea $a_n$ la cantidad de tales cadenas.

Claramente $a_1 = 2$ (las cadenas de longitud 1 que no tiene dos 1 consecutivos son: 0 y 1); y $a_2 = 3$ (las cadenas de longitud 2 que no tienen dos 1 consecutivos son: 00, 01, 10).

Pensemos ahora cómo generar las cadenas de longitud 3 con la condición establecida. ¿Cómo se pueden obtener a partir de cadenas de menor longitud? Note que dada una cadena de longitud 2, podemos obtener cadenas de longitud 3 agregando un 0 ó un 1 al final. Al hacer eso, quedarían las cadenas: 000, 001, 010, 011, 100, 101. Pero la cadena 011

[3] Para ser rigurosos, habría que decir que no es válida una expresión de dos dígitos que comience con 0. Sin embargo, acá las contamos como válidas, ya que en otro caso la recurrencia sería de orden mayor, y el análisis para su deducción más complicado

no es admitida. ¿Por qué? Porque la cadena de longitud 2 que la genera termina con 1, al agregarle un 1 al final, aparecen dos 1 seguidos, lo que no es admitido. Entonces, las cadenas de longitud 3 que no tienen dos 1 seguidos se generan agregando un 0 ó un 1 a cadenas de longitud 2 que no terminen en 1, y agregando un 0 a las cadenas de longitud 2 que terminan en 1. ¿Cuántas son? Llamando $a_2^{(0)}$ a las cadenas de longitud 2 que terminan en 0, y $a_2^{(1)}$ a las cadenas de longitud 2 que terminan en 1, resulta $a_3 = 2a_2^{(0)} + a_2^{(1)}$. Dado que $a_2 = a_2^{(0)} + a_2^{(1)}$, entonces $a_3 = a_2^{(0)} + a_2$. Las cadenas de longitud 2 que terminan en 0 pueden obtenerse a partir de una cadena de longitud 1, agregando un 0 al final. Entonces, $a_2^{(0)} = a_1$. Luego, $a_3 = a_1 + a_2$.

Con un razonamiento similar para cadenas de longitud $n$, se obtiene que $a_n = a_{n-1} + a_{n-2}$ para $n \geqslant 3$.

La ecuación característica es $r^2 - r - 1 = 0$, que tiene por solución $r_1 = \frac{1+\sqrt{5}}{2}$ y $r_2 = \frac{1-\sqrt{5}}{2}$. La solución de la recurrencia tiene la forma $a_n = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right)^n$. Con las condiciones iniciales $a_1 = 2$ y $a_2 = 3$ se obtienen los valores $\alpha_1 = \frac{5+3\sqrt{5}}{10}$ y $\alpha_2 = \frac{5-3\sqrt{5}}{10}$, por lo tanto

$$a_n = \left(\frac{5+3\sqrt{5}}{10}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{5-3\sqrt{5}}{10}\right)\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Haciendo los cálculos necesarios de acuerdo a la expresión obtenida, obtenemos que $a_3 = 5$, $a_4 = 8$, $a_5 = 13$, $a_6 = 21$, etc.

### 2.4.7. Complejidad de un algoritmo recursivo

La sucesión de Fibonacci se define de la siguiente forma:
•$F_0 = 0$, $F_1 = 1$
•$F_n = F_{n-1} + F_{n-2}$ para $n \geqslant 2$

La recurrencia que la define es igual a la recurrencia que aparece en el modelo anterior, pero cambian las condiciones iniciales.

Una función recursiva para calcular el término n-ésimo de la sucesión de Fibonacci es:

```
función fib(n)
   si n=0, fib=0;
   si n=1, fib=1;
   si n>=2; fib=fib(n-1)+fib(n-2);
fin
```

Se quiere medir la complejidad de este algoritmo contando la cantidad de sumas necesarias para calcular $fib(n)$. Sea $a_n$ dicha cantidad. Si $n = 0$, no realiza suma alguna, entonces $a_0 = 0$. Si $n = 1$, igualmente, no realiza suma, $a_1 = 0$. Para $n$ mayores la función se llama así misma con los parámetros $n-1$ y $n-2$, y suma los resultados obtenidos. Entonces, la cantidad de sumas que realiza es la cantidad de sumas necesarias para calcular $fib(n-1)$, que son $a_{n-1}$, más la cantidad de sumas que ejecuta para calcular $fib(n-2)$, que son $a_{n-2}$, más 1.

Luego, $a_n = a_{n-1} + a_{n-2} + 1$. Es una relación de recurrencia de segundo orden lineal no homogénea. La recurrencia homogénea asociada es como la del modelo anterior, por lo tanto $a_n^h = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right)^n$. El término no homogéneo es constante, así que se propone una solución particular constante: $a_n^p = A$. Como ésta debe satisfacer la recurrencia, debe ser $A = A + A + 1$, de donde surge $A = -1$.

La solución tiene la forma $a_n = a_n^h + a_n^p = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right)^n - 1$. Los coeficientes se obtienen con las condiciones iniciales $a_0 = 0$ y $a_1 = 0$, dando lugar a la solución

$$a_n = \left(\frac{1+\sqrt{5}}{2\sqrt{5}}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2\sqrt{5}}\right)\left(\frac{1-\sqrt{5}}{2}\right)^n - 1$$

Notar que cuando $n$ se hace grande, $a_n$ es parecido a $\left(\frac{1+\sqrt{5}}{2\sqrt{5}}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n$, ya que el segundo término tiende a 0. Es decir, que la cantidad de sumas es proporcional a una potencia n-ésima de una constante. Esto se expresa diciendo que la complejidad del algoritmo es $\mathcal{O}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$. La complejidad exponencial de este algoritmo no lo hace muy atractivo en la práctica. De hecho, un algoritmo iterativo para calcular los términos de la serie de Fibonacci es mucho más eficiente, exhibiendo complejidad lineal ($\mathcal{O}(n)$).

## 2.5. Problemas

**Problema 2.1**
*Dar los primeros cinco términos de una sucesión que verifique la relación de recurrencia* $a_n = na_{n-1}$.

**Problema 2.2**
*Dar los primeros seis términos de una sucesión de términos positivos que verifique la relación de recurrencia* $a_n = a_{n-1}/a_{n-2}$.

**Problema 2.3**
*Dar los primeros cinco términos de una sucesión que verifique la relación de recurrencia* $a_n = a_{n-1} + n^2$.

**Problema 2.4**
*Dar los primeros cinco términos de una sucesión que verifique la relación de recurrencia* $a_n = ra_{n-1}$.

**Problema 2.5**
*Dar los primeros cinco términos de una sucesión que verifique la relación de recurrencia* $a_n = (n+1)a_{n-2}$.

**Problema 2.6**
*Dar los primeros seis términos de una sucesión que verifique la relación de recurrencia* $a_n = (n+1)a_{n-2}$ *tal que* $a_0 = 2$.

**Problema 2.7**
*Dar los primeros seis términos de una sucesión que verifique la relación de recurrencia* $a_n = na_{n-1}$ *tal que* $a_3 = 18$.

**Problema 2.8**
*Resolver las relaciones de recurrencia, con las condiciones iniciales dadas.*

*a.* $a_n - (2/3)a_{n-1} = 0$, $n \geqslant 1$, $a_0 = -1$.

*b.* $2a_{n+1} - 3a_n = 0$, $n \geqslant 0$, $a_0 = 1$.

*c.* $2a_{n+1} - 3a_n = 0$, $n \geqslant 0$, $a_0 = 2$.

*d.* $a_n - 5a_{n-1} + 6a_{n-2} = 0$, $n \geqslant 2$, $a_0 = 0$, $a_1 = 1$.

*e.* $a_{n+2} = 4a_{n+1} - 5a_n$, $n \geqslant 0$, $a_0 = -1$, $a_1 = 1$.

*f.* $a_n = 4a_{n-1} + 4a_{n-2}$, $n \geqslant 2$, $a_0 = 1$, $a_1 = -2$.

*g.* $9a_n - 4a_{n-2} = 0$, $n \geqslant 2$, $a_0 = 1$, $a_1 = 2$.

*h.* $a_{n+2} - 3a_n = 0$, $n \geqslant 0$, $a_0 = 0$, $a_1 = 1$.

*i.* $3a_{n+1} = a_{n-1} = 0$, $n \geqslant 1$, $a_0 = 1$, $a_1 = 0$.

**Problema 2.9**
*Dada la relación de recurrencia* $8a_n + 4a_{n-1} - 4a_{n-2} = 0$, $n \geqslant 2$, *indicar si las siguientes sucesiones pueden ser solución:*

*a.* $a_n = 3(-1)^n$

*b.* $a_n = 3(-1/2)^n + 1$

*c.* $a_n = 4(-1)^n + (1/2)^n$

*d.* $4a_n = -4 + (1/2)^n$

*En caso afirmativo, justificar e indicar cuáles serían las condiciones iniciales que deben imponerse para obtener dicha solución. En caso negativo, justificar.*

**Problema 2.10**
*Dada la relación de recurrencia* $a_{n+2} - a_n = 0$, *indicar si las siguientes sucesiones pueden ser solución:*

*a.* $a_n = 3(-1)^n$

*b.* $a_n = 3(-1/2)^n + 1$

*c.* $a_n = 7 + 2(-1)^n$

*d.* $a_n = (1/3)2^n$

*e.* $a_n = -8$

*En caso afirmativo, justificar e indicar cuáles serían las condiciones iniciales que deben imponerse para obtener dicha solución. En caso negativo, justificar.*

**Problema 2.11**
*El número de usuarios en un sitio web es de 1000 en un momento dado. Se sabe que el número de usuarios aumentará 25 % cada día. Plantear una relación de recurrencia para determinar el número de usuarios en el día n.*

**Problema 2.12**

*Una empresa de catering para eventos comienza sus actividades en enero de 2017. Durante el primer mes sus ingresos fueron 11000 pesos. Un estudio de mercado predice que los ingresos crecerán 23 % cada mes. Plantear una relación de recurrencia y las condiciones iniciales necesarias para determinar los ingresos en el mes n. Dar la solución de la recurrencia obtenida.*

**Problema 2.13**

*Una inversión de $100 iniciales recibe un interés de 10 % anual, capitalizado mensualmente. Plantear una relación de recurrencia para calcular el dinero acumulado al cabo de n meses.*

**Problema 2.14**

*Se sabe que la propagación de un virus informático en una red responde a la relación de recurrencia $a_n = (4/3)a_{n-1} - (1/3)a_{n-2}$, donde $a_n$ es la proporción de equipos infectados por el virus en la semana n. A la semana siguiente de haber lanzado el virus, se detectan 1/3 de los equipos infectados. Hallar la solución para $a_n$. ¿Cuál es la proporción de equipos infectados en la semana 3? ¿La proporción de infectados puede superar el 90 % en algún momento?*

**Problema 2.15**

*Sea $p_n$ la probabilidad de que se detecte al menos un caso de sarampión en la semana n-ésima de clases, luego de que se detecte el primer caso en la semana 1. Los registros históricos indican que $p_n = p_{n-1} - 0{,}25p_{n-2}$. Identificar las condiciones iniciales. ¿Cuál es la probabilidad de contagio en la semana 5ta? Resolver la relación de recurrencia.*

**Problema 2.16**

*Cierto canal de comunicación transmite mensajes que consisten en una sucesión dos tipos de señales, una que dura un microsegundo y otra que dura dos microsegundos. Determinar una relación de recurrencia para contar la cantidad de mensajes distintos de n microsegundos de duración, que se pueden transmitir. ¿Cuáles son las condiciones iniciales? ¿Cuántos mensajes se pueden transmitir en exactamente 10 microsegundos?*

**Problema 2.17**

*Cierto canal de comunicación transmite mensajes que consisten en una sucesión dos tipos de señales, una que dura dos microsegundo y otra que dura tres microsegundos. Determinar una relación de recurrencia para contar la cantidad de mensajes distintos de n microsegundos de duración, que se pueden transmitir. ¿De qué orden es la recurrencia? ¿Cuáles son las condiciones iniciales? ¿Cuántos mensajes se pueden transmitir en exactamente 10 microsegundos?*

**Problema 2.18**

*Hallar una relación de recurrencia para $a_n$, el número de formas de avanzar n metros dando pasos de 1 o 2 metros. Resolverla.*

**Problema 2.19**

** Hallar una relación de recurrencia para contar el número de sucesiones binarias de longitud n que no tienen ceros consecutivos.*

**Problema 2.20**

** Determinar una relación de recurrencia para el número de sucesiones de 0, 1 y 2 con un número par de ceros.*

**Problema 2.21**

*El tamaño (en cantidad de individuos) de cierta población en cada año responde a una relación de recurrencia homogénea lineal con coeficientes constantes de orden 1. Si se sabe que en 2015 dicha población tiene 1500 individuos, y que en 2016 su tamaño es 1650, hallar la cantidad de individuos en los años de 2010 a 2020.*

**Problema 2.22**

** El tamaño (en cantidad de individuos) de cierta población en cada año responde a una relación de recurrencia homogénea lineal con coeficientes constantes de orden 2. Se sabe que la población tenía en 2014, 1500 individuos, en 2015, 1650 individuos, en 2016 1800 individuos y en 2017, 1965. Encontrar la recurrencia que la modela, y hallar la cantidad de individuos en los años de 2010 a 2020.*

**Problema 2.23**

*Resolver las siguientes relaciones de recurrencias no homogéneas*

*a.* $a_n - 3a_{n-1} = 5 \cdot 7^n$, $n \geqslant 1$, $a_0 = 2$.

*b.* $a_{n+1} = a_n + 2^n$, $n \geqslant 0$, $a_0 = 0$.

*c.* $a_n = a_{n-1} + 3$, $n \geqslant 1$, $a0 = 1$.

*d.* $a_{n+1} + 2a_n + a_{n-1} = n$, $n \geqslant 1$, $a_0 = 1$, $a1 = -1$.

# Capítulo 3

# Aritmética entera

## 3.1. Números naturales y enteros

Si bien el lector estará familiarizado con los números naturales y los números enteros, se dará aquí una breve introducción para repasar sus propiedades más importantes.

Los números naturales son los que se utilizan para contar. El conjunto de los números naturales, denotado $\mathbb{N}$, contiene los números 1, 2, 3, 4, ...

La suma y producto de números naturales dan como resultado un natural. La resta y el cociente de naturales podría dar un resultado que no sea un número natural.

Todos los naturales son positivos, por ser mayores a 0. El opuesto de un número $n$, denotado $-n$, es $-n = (-1) \cdot n$. Los opuestos de los naturales son negativos, son menores a 0.

El conjunto de los números enteros, denotado $\mathbb{Z}$ está formado por todos los naturales, los números opuestos de los naturales y el 0. La suma, la resta y el producto de números enteros dan como resultado un entero. El cociente de enteros podría dar un número no entero. Esto justifica la definición de divisibilidad, que se desarrollará en la siguiente sección.

## 3.2. Divisibilidad

Dados dos números enteros $a$ y $b$, con $a \neq 0$, decimos que $a$ ES DIVISOR DE $b$ si existe un número entero $c$ tal que $b = a \cdot c$. En otras palabras, $a$ es divisor de $b$ si al dividir $b$ por $a$ se obtiene un número entero. También se usan las expresiones $a$ ES FACTOR DE $b$, $a$ DIVIDE A $b$, y $b$ ES MÚLTIPLO DE $a$.

Nótese que 0 no puede ser divisor de nadie, queda expresamente excluído de la definición.

***Ejemplo* 3.1**
*5 es divisor de 35, 21 es divisor de 84, -4 es divisor de 1024, 21 es divisor de -84.*
*1 es divisor de cualquier número entero. -1 es divisor de cualquier entero.*
*Todo número es divisor de 0.* ■

La relación de divisibilidad satisface las siguientes propiedades:

**Proposición 3.1.**

- ***a** divide a **a**, y divide a $-\boldsymbol{a}$.*
- *1 y -1 dividen a cualquier entero.*
- *Si a es divisor de b, entonces el valor absoluto de a es menor o igual al valor absoluto de b.*
- *Si* **a** *divide a* **b**, *entonces* **a** *divide a* $-\mathbf{b}$; $-\mathbf{a}$ *divide a* $-\mathbf{b}$ *y* $-\mathbf{a}$ *divide a* **b** .
- *Si* **a** *divide a dos números* **b** *y* **c**, *entonces* **a** *divide a* $\mathbf{b}+\mathbf{c}$.
- *Si* **a** *divide a un número* **b**, *entonces* **a** *divide a cualquier múltiplo de* **b**.
- *La relación de divisibilidad es transitiva: Si* **a** *divide a* **b** *y* **b** *divide a* **c**, *entonces* **a** *divide a* **c**.
- *Si* **a** *divide a dos números* **b** *y* **c**, *entonces divide a la suma de múltiplos de ellos.* $\diamondsuit$

Un número entero $p$ distinto de 1 y de -1 se denomina PRIMO si sus únicos divisores son $p$, $-p$, 1 y -1. Un número que no es primo, y es distinto de 1, 0 y -1, se denomina COMPUESTO. El número $p = 13$ es primo ya que sus únicos divisores son 13, -13, 1 y -1. El número $p = -43$ es primo, sus divisores son -43, 43, 1 y -1.

Los primeros primos positivos son 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, ...

Un número compuesto tiene al menos tres divisores positivos. Por ejemplo, el número 33 tiene como divisores positivos a los números 1, 3, 11, 33. El número 28 tiene como divisores positivos los números 1, 2, 4, 7, 14, 28.

¿Cuáles son los divisores de 54032? Este número es muy grande como para hacer la lista de divisores sin alguna herramienta algorítmica.

Una estrategia para hallar los divisores de un número entero $n$ es dividirlo sucesivamente por los enteros 2, 3, 4, 5, etc., y si la división da un número entero, el número por el que se dividió es un divisor de $n$. También el resultado de la división es un divisor de $n$. Por ejemplo, si tomamos $n = 96$, y dividimos por 4, el resultado es entero: 96/4=24. Entonces, 4 es divisor de 96, y 24 también lo es. ¿Hasta cuándo hay que hacer esas divisiones sucesivas para hallar todos los divisores? En principio, como todo divisor de $n$ es menor a $n$, podría pensarse que deben hacerse las divisiones hasta $n - 1$. Sin embargo, el procedimiento puede finalizar en $\sqrt{n}$, ya que en cada división que da un número entero, se obtienen dos números (divisor y cociente) que son divisores de $n$. Y de esos dos números, necesariamente uno es menor o igual a $\sqrt{n}$.

Así, un algoritmo para hallar todos los divisores positivos de $n$ es:

**Algoritmo: Divisores positivos**
**Entrada**: $n$, entero
**Salida**: lista $L$ de los divisores positivos de $n$
**Inicio**:
$L = \emptyset$
**Para** $i$ de 1 a $\lfloor\sqrt{|n|}\rfloor$ hacer
    **Si** $n/i$ es entero
        $L = L \bigcup \{i, n/i\}$
    **Fin Si**
**Fin Para**

**Para pensar**: ¿Cómo se puede modificar el algoritmo anterior para determinar si un número es primo o no?

**Teorema 3.2.**
***Teorema fundamental de la aritmética***
*Todo entero mayor que 1 se puede escribir como producto de números primos en forma única (salvo por el orden de los factores).* ♣

Tal representación como producto de números primos se conoce como FACTORIZACIÓN PRIMA o DESCOMPOSICIÓN EN FACTORES PRIMOS.

***Ejemplo* 3.2**
*Se dan como ejemplo algunas factorizaciones:*
$126 = 2 \cdot 3^2 \cdot 7$
$1032 = 2^3 \cdot 3 \cdot 43$
$641 = 641$, *ya que este número es primo*
$2048 = 2^{11}$ ■

Teniendo la factorización de un número es fácil conocer todos sus divisores. Pongamos como ejemplo el 126. Sabemos que $126 = 2.3^2.7$. Sus divisores positivos, con sus factorizaciones son:

| | | |
|---|---|---|
| $1 = 2^0 \cdot 3^0 \cdot 7^0$ | $3 = 2^0 \cdot 3^1 \cdot 7^0$ | $9 = 2^0 \cdot 3^2 \cdot 7^0$ |
| $2 = 2^1 \cdot 3^0 \cdot 7^0$ | $6 = 2^1 \cdot 3^1 \cdot 7^0$ | $18 = 2^1 \cdot 3^2 \cdot 7^0$ |
| $7 = 2^0 \cdot 3^0 \cdot 7^1$ | $21 = 2^0 \cdot 3^1 \cdot 7^1$ | $63 = 2^0 \cdot 3^2 \cdot 7^1$ |
| $14 = 2^1 \cdot 3^0 \cdot 7^1$ | $42 = 2^1 \cdot 3^1 \cdot 7^1$ | $126 = 2^1 \cdot 3^2 \cdot 7^1$ |

Puede verse que todos los divisores tienen los mismos factores primos que 126, y aparecen con potencias que son menores o iguales a las potencias con que aparecen en la factorización de 126.

Veamos otro ejemplo: $400 = 2^4 \cdot 5^2$. Sus divisores son:

| | | |
|---|---|---|
| $1 = 2^0 \cdot 5^0$ | $5 = 2^0 \cdot 5^1$ | $25 = 2^0 \cdot 5^2$ |
| $2 = 2^1 \cdot 5^0$ | $10 = 2^1 \cdot 5^1$ | $50 = 2^1 \cdot 5^2$ |
| $4 = 2^2 \cdot 5^0$ | $20 = 2^2 \cdot 5^1$ | $100 = 2^2 \cdot 5^2$ |
| $8 = 2^3 \cdot 5^0$ | $40 = 2^3 \cdot 5^1$ | $200 = 2^3 \cdot 5^2$ |
| $16 = 2^4 \cdot 5^0$ | $80 = 2^4 \cdot 5^1$ | $400 = 2^4 \cdot 5^2$ |

Obsérvese que todos los divisores positivos de 400 son de la forma $2^{e_1} \cdot 5^{e_2}$, donde $0 \leqslant e_1 \leqslant 4$ y $0 \leqslant e_2 \leqslant 2$. Es decir, $e_1$ puede tomar cinco valores y $e_2$ puede tomar tres valores. Luego, la cantidad de divisores es $5 \cdot 3 = 15$. Este resultado se generaliza en el siguiente teorema:

**Teorema 3.3.**
*Si la factorización de un número entero positivo $n$ es $n = p_1^{r_1} \cdot p_1^{r_2} \cdot ... \cdot p_k^{r_k}$, todos sus divisores positivos son de la forma $p_1^{e_1} \cdot p_1^{e_2} \cdot ... \cdot p_k^{e_k}$, donde $0 \leqslant e_i \leqslant r_i$ para todo $i = 1, 2, ..., k$. La cantidad de divisores es $(r_1 + 1) \cdot (r_2 + 1) \cdot ... \cdot (r_k + 1)$.* ♣

El MÁXIMO COMÚN DIVISOR entre dos números enteros $a$ y $b$, denotado $mcd(a, b)$ es un número entero positivo que divide a ambos, y es el mayor con esa propiedad. El MÍNIMO COMÚN MÚLTIPLO entre dos números enteros $a$ y $b$, denotado $mcm(a, b)$ es un número positivo que es múltiplo de ambos, y es el menor con esa propiedad.

***Ejemplo* 3.3**
*Busquemos el $mcd(45, 20)$. Los divisores de 45 son 1, 3, 5, 9, 15, 45; y los de 20 son 1, 2, 4, 5, 10, 20. Entonces, el mayor divisor de los dos es 5; $mcd(45, 20) = 5$. Y $mcm(45, 20) = 180$.*

*Consideremos los números 45 y 28. Los divisores de 28 son 1, 2, 4, 7, 14, 28. Entonces $mcd(45, 28) = 1$. Y el $mcm(45, 28) = 1260$.*

*Dados los números 40 y 150, los divisores de 40 son 1, 2, 4, 5, 8, 10, 20, 40, y los de 150 son 1, 2, 3, 5, 6, 10, 15, 25, 30, 50, 75, 150. Entonces $mcd(40, 150) = 10$.* ■

El método de hallar el *mcd* calculando primero todos los divisores de los dos números dados y viendo cuál es el mayor divisor de ambos, es poco eficiente, sobre todo cuando se opera con números grandes. Hay varias maneras mejores de hacerlo.

Miremos los resultados anteriores factorizando:
$mcd(45, 20) = mcd(3^2 \cdot 5^1, 2^2 \cdot 5^1) = 5 = 2^0 \cdot 3^0 \cdot 5^1$
$mcd(45, 28) = mcd(3^2 \cdot 5^1, 2^2 \cdot 7^1) = 1 = 2^0 \cdot 3^0 \cdot 5^0 \cdot 7^0$
$mcd(40, 150) = mcd(2^3 \cdot 5^1, 2^1 \cdot 3^1 \cdot 5^2) = 10 = 2^1 \cdot 3^0 \cdot 5^1$

Veamos otros ejemplos:

$mcd(8624, 252) = mcd(2^4 \cdot 7^2 \cdot 11^1, 2^2 \cdot 3^2 \cdot 7^1) = 28 = 2^2 \cdot 7^1$
$mcd(81675, 2750) = mcd(3^3 \cdot 5^2 \cdot 11^2, 2^1 \cdot 5^3 \cdot 11^1) = 275 = 5^2 \cdot 11^1$

De los ejemplos anteriores, es fácil ver que:

**Proposición 3.4.**
*Si dos números enteros positivos se factorizan $a = p_1^{e_1} \cdot p_2^{e_2} \cdot ... \cdot p_k^{e_k}$ y $b = p_1^{f_1} \cdot p_2^{f_2} \cdot ... \cdot p_k^{f_k}$, entonces $mcd(a, b) = p_1^{s_1} \cdot p_2^{s_2} \cdot ... \cdot p_k^{s_k}$, donde $s_i = \min(e_i, f_i)$ para todo $i = 1, 2, ..., k$.* ♢

Y además:

**Proposición 3.5.**
*Dados dos enteros positivos $a$ y $b$, $a \cdot b = mcd(a, b) \cdot mcm(a, b)$* ♢

Veamos, para terminar esta sección, algunas propiedades del *mcd*.

**Proposición 3.6.**

- $mcd(a, b) = mcd(b, a) = mcd(-a, b) = mcd(a, -b) = mcd(-a, -b)$
- $mcd(a, b) = a$ *si y sólo si* $a$ *divide a* $b$. ♢

***Ejemplo* 3.4**

$mcd(31, 1240) = 31$, *ya que* $31|1240$.
$mcd(-8624, 252) = 28$
$mcd(304, 0) = 304$, *ya que* $304|0$. ■

Dos números son COPRIMOS o primos relativos si el máximo común divisor de ellos es 1.

## 3.3. Números primos y sus aplicaciones

Los conceptos de divisibilidad y de factorización son conceptos de importancia relevante para seguridad informática y criptografía. En criptografía, se estudia cómo transmitir mensajes codificados de tal manera que sólo el destinatario (y no un tercero ajeno al sistema) pueda descifrarlo. Ciertos métodos de encriptación (codificación de mensajes) se basan en que la tarea de determinar si un entero es primo o no; o de factorizar un entero grande, y esta tarea requiere demasiados recursos computacionales.

Uno de los métodos de encriptación más difundidos, denominado RSA (Rivest-Shamir-Adleman), encripta el mensaje utilizando una clave pública que es producto de dos primos grandes. Si algún receptor indeseado pudiera factorizar la clave pública y conocer los dos primos grandes que la componen, podría descifrar el mensaje.

En la sección anterior se mostró un algoritmo para hallar los divisores positivos de un entero. El mismo algoritmo puede emplearse para determinar si un número $n$ es primo o no: se obtienen sus divisores, y si éstos son únicamente 1 y $n$, es primo. Es un algoritmo eficaz (lleva a cabo lo que se pretende) pero muy ineficiente. Si el número $n$ es demasiado grande (digamos, de 100 dígitos) la cantidad de pasos a realizar es cerca de $\sqrt{|n|}$, que es una cantidad de 50 dígitos. Demasiadas operaciones.

Si bien existen algoritmos para determinar la primalidad de un entero mucho más eficientes que el mencionado, el análisis de la primalidad de un entero es una tarea difícil. Tan difícil que se lo toma como desafío para probar nuevos procesadores y nuevos algoritmos.

Se sabe que hay infinitos números primos. Es decir, se pueden conseguir números primos mayores que cualquier número. Los primos conocidos más grandes son de la forma $2^p - 1$, con $p$ primo. Éstos se conocen como PRIMOS DE MERSENNE. En particular, si $p = 2$, $2^2 - 1 = 3$, si $p = 3$, $2^3 - 1 = 7$, si $p = 5$, $2^5 - 1 = 31$, si $p = 7$, $2^7 - 1 = 127$. Todos éstos son primos. Parecería que todo número de la forma $2^p - 1$, con $p$ primo, es primo. Pero esto no es cierto, y el contraejemplo surge con el siguiente primo: $p = 11$, $2^{11} - 1 = 2047 = 23 \cdot 89$, no es primo. Con $p$ =13, 17 y 19, $2^p - 1$ son primos, pero $2^{23} - 1 = 8388607 = 47 \cdot 178481$, no es primo.

Existe una comunidad virtual, conocida como GIMPS (Great Internet Mersenne Prime Search) que desde 1996 promueve la búsqueda de primos de Mersenne. En la práctica, son personas que ponen sus computadoras a disposición del grupo para ejecutar tests de primalidad.

El 41º primo de Mersenne, con $p$=24036583, fue hallado en 2006, y en 2011 se probó que no había otro menor sin descubrir.

El 47º primo de Mersenne, descubierto en 2008, es $2^{43112609}-1$ y tiene 12978189 dígitos. El descubridor de este número se hizo acreedor de un premio de U$S 100000 ofrecido por la Electronic Frontier Foundation, por ser el primer primo de más de 10 millones de dígitos. Este número está disponible en Internet, en un archivo de texto de 16,73 Mb.

A principios de 2013 se descubrió el primo de Mersenne 48º, $2^{57885161}-1$, que tiene más de 17 millones de dígitos. La prueba de primalidad tomó 39 días completos de cálculos en una computadora.

En enero de 2018 se descubrió un nuevo primo de Mersenne, el más grande conocido hasta el momento de escribir este libro. El número es $2^{77232917}-1$ y tiene más de 23 millones de dígitos.

El esfuerzo computacional y humano puesto a disposición de esta búsqueda tiene varias motivaciones. Además de obtener rarezas (estos números son números raros. ¡Sólo se conocen 50 hasta ahora!) y perseguir los premios ofrecidos [1] y la fama, las personas que trabajan en esto logran generar algoritmos para resolver más rapidamente operaciones intermedias (por ejemplo, el producto de dos enteros grandes). Además, los programas implementados en la búsqueda son usados para testear hardware: se ejecutan los programas (que usan intensivamente los procesadores) sobre primos conocidos, así el resultado es también conocido.

## 3.4. División entera (Algoritmo de Euclides)

Se había mencionado que el cociente de enteros no siempre da entero, por lo que la operación división no está bien definida en $\mathbb{Z}$. Tampoco es posible hablar de inversos entre los enteros [2].

Sin embargo, se puede definir la DIVISIÓN ENTERA, que da un cociente y un resto, enteros. Esta operación permite obtener un algoritmo para hallar el $mcd$ de dos enteros sin necesidad de factorizarlos. Además, da lugar al concepto de congruencia, que genera una partición del conjunto de enteros en subconjuntos, de acuerdo a su resto en la división por un entero.

Dados dos números enteros $a$ y $b$, con $b > 0$, existen enteros $q$ y $r$ únicos, con $0 \leqslant r < b$ tal que

$$a = q \cdot b + r$$

$q$ se denomina COCIENTE y $r$ es el RESTO de la división entera de $a$ (DIVIDENDO) por $b$ (DIVISOR).

[1] El premio ofrecido actualmente es de U$S 150000 para el descubridor del primer primo de Mersenne de 100 millones de dígitos.

[2] Dado un número $n$, su inverso es el número $j$ tal que $n \cdot j = 1$.

Por ejemplo:

| Dividendo (a) | Divisor (b) | División entera | Cociente (q) | Resto (r) |
|---|---|---|---|---|
| 123 | 4 | $\mathbf{123} = 30\cdot \mathbf{4} + 3$ | 30 | 3 |
| 11 | 13 | $\mathbf{11} = 0\cdot \mathbf{13} + 11$ | 0 | 11 |
| 204 | 6 | $\mathbf{204} = 34\cdot \mathbf{6} + 0$ | 34 | 0 |
| -11 | 13 | $\mathbf{-11} = (-1)\cdot \mathbf{13} + 2$ | -1 | 2 |
| -315 | 1 | $\mathbf{-315} = (-315)\cdot \mathbf{1} + 0$ | -31 | 0 |
| -100 | 7 | $\mathbf{-100} = (-15)\cdot \mathbf{7} + 5$ | -15 | 5 |

Nótese que el resto siempre debe ser mayor o igual a cero y menor que el divisor. Si bien es cierta la igualdad $-100 = -14 \cdot 7 - 2$, no es -2 el resto de dividir $-100$ por 7.

De la definición de divisor, se puede deducir que $b$ es divisor de $a$ si y sólo si el resto de la división entera de $a$ por $b$ es 0.

Veamos cómo la división entera puede darnos otras herramientas para hallar el *mcd* de dos números.

Suponga que el entero $c$ divide a $a$ y a $b$, con $a > b > 0$. Al hacer la división entera, $a = q \cdot b + r$, y $r = a - q \cdot b$. Entonces, $c$ también divide a $r$, ya que cualquier divisor común de $a$ y $b$ divide a una suma de múltiplos de de $a$ y $b$ (por la última propiedad de la proposición 3.1). Luego, el $mcd(a, b)$ divide a $r$.

En particular, $mcd(a, b) = mcd(b, r)$. Si $r = 0$, $mcd(b, r) = b$, por la segunda propiedad de la proposición 3.6, ya que $b$ divide a $r$ en ese caso. Estas ideas proveen un método para hallar el *mcd* de dos enteros. Consiste en realizar sucesivas divisiones enteras hasta tener un resto 0, de la siguiente manera:

$$
\begin{aligned}
\mathbf{a} &= q_1 \cdot \mathbf{b} + r_1 && 0 \leqslant r_1 < b \\
\mathbf{b} &= q_2 \cdot \mathbf{r_1} + r_2 && 0 \leqslant r_2 < r_1 \\
\mathbf{r_1} &= q_3 \cdot \mathbf{r_2} + r_3 && 0 \leqslant r_3 < r_2 \\
&\ldots \\
\mathbf{r_{k-2}} &= q_k \cdot \mathbf{r_{k-1}} + r_k && 0 \leqslant r_k < r_{k-1} \\
\mathbf{r_{k-1}} &= q_{k+1} \cdot \mathbf{r_k}
\end{aligned}
$$

En cada división entera, se divide el anterior divisor por el anterior resto. Como los sucesivos restos son números positivos cada vez más chicos, en algún momento se obtiene un resto 0. Si $r_k$ es el último resto distinto de cero, $r_k$ divide a $r_{k-1}$, entonces divide a $r_{k-2}$, ..., entonces divide a $b$, entonces divide a $a$. Luego, $r_k = mcd(a, b)$.

***Ejemplo* 3.5**
*Busquemos el* $mcd(1820, 231)$. *Haciendo las sucesivas divisiones:*

$$
\begin{aligned}
1820 &= 7 \cdot 231 + 203 \\
231 &= 1 \cdot 203 + 28 \\
203 &= 7 \cdot 28 + 7 \\
28 &= 4 \cdot 7
\end{aligned}
$$

*Entonces,* $mcd(1820, 231) = 7$. ■

***Ejemplo* 3.6**
*Hallemos el* $mcd(116, 40)$*:*

$$\begin{aligned} 116 &= 2 \cdot 40 + 36 \\ 40 &= 1 \cdot 36 + 4 \\ 36 &= 9 \cdot 4 \end{aligned}$$

*Entonces,* $mcd(116, 40) = 4$. ■

***Ejemplo* 3.7**
*Hallemos el* $mcd(5400, 96)$*:*

$$\begin{aligned} 5400 &= 56 \cdot 96 + 24 \\ 96 &= 4 \cdot 24 \end{aligned}$$

*Entonces,* $mcd(5400, 96) = 24$. ■

Este método es fácilmente programable. Varios lenguajes de programación tienen una función DIV para hallar el cociente de la división entera, y la función MOD para calcular el resto de la división. Se da el siguiente pseudocódigo:

**Algoritmo: mcd**
**Entrada**: dos enteros positivos $a$ y $b$, con $a > b$
**Salida**: el *mcd* de $a$ y $b$
dividendo=$a$;
divisor=$b$;
**Hacer**
    cociente= dividendo **div** divisor;
    resto=dividendo **mod** divisor;
    dividendo=divisor;
    divisor=resto;
**Hasta que** resto=0
**Salida**: dividendo

Aún si no se tiene la función DIV, cuando la operación división (/) opera sobre variables declaradas como enteros, da como resultado el cociente entero.

De todas formas, el cociente entero y el resto se pueden calcular usando las operaciones matemáticas básicas, y la función parte entera (floor):
cociente=floor(dividendo/divisor)
resto=dividendo-cociente·divisor.

### Aplicación: representación de enteros en alguna base

Una aplicación del algoritmo de Euclides es en la representación de un entero en alguna base. Comúnmente usamos notación decimal (en base 10) para escribir enteros. Los dígitos de la expresión decimal son los coeficientes de la expresión del número como suma de potencias de 10. Por ejemplo, el número 3781 se descompone como $3781 = 3 \cdot 10^3 + 7 \cdot 10^2 + 8 \cdot 10^1 + 1 \cdot 10^0$. Usando una base $b > 1$, un entero positivo $a$ se puede escribir

$$a = a_n b^n + a_{n-1} b^{n-1} + ... + a_1 b^1 + a_0 b^0$$

con $a_n \neq 0$ y $0 \leqslant a_i < b$ para $i = 0, 1, ..., n$. Entonces, la EXPRESIÓN DE $a$ EN LA BASE $b$ es $(a_n a_{n-1} ... a_1 a_0)_b$.

***Ejemplo* 3.8**
*El número cuya expresión en base 2 es* $(1001011)_2$ *es (en base 10)* $\mathrm{a} = 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 64 + 8 + 2 + 1 = 75$.

*En base 16 (llamada hexadecimal) se usan los dígitos 0, 1, 2, ..., 9, A, B, ..., F. Los dígitos de A a F representan 10, 11, ..., 15. El número cuya expresión en base 16 es* $(91A4F)_{16}$ *es* $9 \cdot 16^4 + 1 \cdot 16^3 + 10 \cdot 16^2 + 4 \cdot 16^1 + 15 \cdot 16^0 = 596784$. ■

La descomposición de $a$ en sumas de potencias de $b$ puede expresarse:

$$a = (((a_n b + a_{n-1})b + ... + a_2)b + a_1)b + a_0$$

De esta forma puede verse que
◇ al dividir $a$ por $b$, se obtiene cociente $q_0 = ((a_n b + a_{n-1})b + ... + a_2)b + a_1$, y resto $a_0$
◇ al dividir $q_0$ por $b$, se obtiene cociente $q_1 = (a_n b + a_{n-1})b + ... + a_2$, y resto $a_1$
◇ ...
◇ al dividir $q_{n-2}$ por $b$, se obtiene cociente $q_{n-1} = a_n$, y resto $a_{n-1}$

Esto da una forma de obtener la expresión de $a$ en base $b$, haciendo sucesivas divisiones enteras de los cocientes por $b$, hasta obtener un cociente menor a $b$. Los restos obtenidos en cada división, junto con el último cociente, son los dígitos de esa expresión.

***Ejemplo* 3.9**
*Escribamos el número* 4256 *en base* 4. *Haciendo las sucesivas divisiones:*

$$\begin{aligned} 4256 &= 1064 \cdot 4 + 0 \\ 1064 &= 266 \cdot 4 + 0 \\ 266 &= 66 \cdot 4 + 2 \\ 66 &= 16 \cdot 4 + 2 \\ 16 &= 4 \cdot 4 + 0 \\ 4 &= 1 \cdot 4 + 0 \end{aligned}$$

*Entonces,* $4256 = (1002200)_4$.
*El número* 540013 *en base* 16 *es:*

$$\begin{aligned} 540013 &= 33750 \cdot 16 + 13 \\ 33750 &= 2109 \cdot 16 + 6 \\ 2109 &= 131 \cdot 16 + 13 \\ 131 &= 8 \cdot 16 + 3 \end{aligned}$$

*Luego,* $540013 = (83D6D)_{16}$
*El número* 200 *en base* 2 *se calcula con las divisiones:*

$$\begin{aligned} 200 &= 100 \cdot 2 + 0 \\ 100 &= 50 \cdot 2 + 0 \\ 50 &= 25 \cdot 2 + 0 \\ 25 &= 12 \cdot 2 + 1 \\ 12 &= 6 \cdot 2 + 0 \\ 6 &= 3 \cdot 2 + 0 \\ 3 &= 1 \cdot 2 + 1 \end{aligned}$$

*Así,* $200 = (11001000)_2$ ■

## 3.5. Ecuaciones diofánticas

Otra de las aplicaciones del algoritmo de Euclides es en la búsqueda de soluciones de ecuaciones diofánticas. Veamos primero la definición de ecuaciones diofánticas, y luego analizaremos cómo se pueden obtener las soluciones (si existen).

Una ECUACIÓN DIOFÁNTICA LINEAL en dos variables es

$$a\,x + b\,y = c$$

donde $a$, $b$ y $c$ son números enteros dados, y $x$ e $y$ son las incógnitas, que deben tomar valores **enteros**. A los números $a$ y $b$ se los llama coeficientes de la ecuación, y el número $c$ es el término independiente.

Si $d$ es un divisor de $a$ y de $b$, que también divide a $c$, la ecuación diofántica $a\,x+b\,y = c$ es equivalente a la ecuación diofántica

$$\frac{a}{d}\,x + \frac{b}{d}\,y = \frac{c}{d}$$

Que sean ecuaciones equivalentes implica que tienen las mismas soluciones.

***Ejemplo* 3.10**
*Dada la ecuación diofántica $2x - 5y = 1$, algunas soluciones son $x = 3$, $y = 1$; $x = -2$, $y = -1$; $x = 13$, $y = 5$. ¿Habrá más soluciones?* ■

***Ejemplo* 3.11**
*Dada la ecuación diofántica $6x - 9y = -30$, algunas soluciones son $x = 10$, $y = 10$; $x = 40$, $y = 30$; $x = -20$, $y = -10$. Esta ecuación es equivalente a $2x - 3y = -10$, por lo que ambas ecuaciones tienen las mismas soluciones.* ■

***Ejemplo* 3.12**
*La ecuación diofántica $6x - 9y = 8$ no tiene solución. Veamos por qué: el término $6x$ es múltiplo de 3 si $x$ es entero, el término $9y$ es múltiplo de 3 si $y$ toma valores enteros, por lo tanto su resta $6x - 9y$ es múltiplo de 3 para cualesquiera valores enteros de $x$ y de $y$. Así, es imposible que sea igual a 8, ya que 8 no es múltiplo de 3.* ■

***Ejemplo* 3.13**
*La ecuación diofántica $45x+75y = 40$ no tiene solución. Por un lado, observemos que $45x$ y $75y$ son múltiplos de 5 (cuando $x$ e $y$ toman valores enteros), y por lo tanto $45x + 75y$ también es múltiplo de 5. Y el lado derecho, 40, también lo es. Pero esto no es suficiente para asegurar la existencia de soluciones. En este caso, $45x$ y $75y$ también son múltiplos de 3, por lo tanto su suma, $45x+75y$ también es múltiplo de 3. Esta suma entonces nunca podrá ser igual a 40, ya que éste no es múltiplo de 3.* ■

***Ejemplo* 3.14**
*¿Tiene soluciones la ecuación diofántica $45x + 75y = 90$? Sabemos que $45x + 75y$ es múltiplo de 5 y también es múltiplo de 3 (entonces es múltiplo de 15). Y también 90 es múltiplo de 15. ¿Esto me permite asegurar que la ecuación tiene solución? Veremos que sí. En este caso, una solución es $x = -3, y = 3$, otra solución es $x = 2, y = 0$. La ecuación dada es equivalente a $3x+5y = 6$, que se obtiene al dividir ambos miembros de la ecuación original por 15.* ■

***Ejemplo* 3.15**
*La ecuación diofántica* $42x + 105y = 30$ *no tiene solución. Por un lado, observemos que* $42x$ *y* $105y$ *son múltiplos de 3 (cuando* $x$ *e* $y$ *toman valores enteros), y el lado derecho, 30, también lo es. Pero además,* $42x$ *y* $105y$ *también son múltiplos de 7, por lo tanto su suma,* $42x + 105y$ *también es múltiplo de 7. Esta suma entonces nunca podrá ser igual a 30, porque 30 no es múltiplo de 7.* ■

La condición para asegurar la existencia de soluciones de una ecuación diofántica lineal se da en siguiente teorema.

**Teorema 3.7.**
*La ecuación diofántica* $a\,x + b\,y = c$ *tiene solución si y sólo si* $mcd(a, b)$ *divide a* $c$. ♣

Por eso, la ecuación del ejemplo 3.12 no tiene solución, ya que el $mcd(6, -9) = 3$ no divide a 8. La ecuación del ejemplo 3.11 tiene solución, porque el $mcd(6, -9) = 3$ divide a -30.

En el ejemplo 3.13, $mcd(45, 75) = 15$ y 15 no divide a 40. Pero sí divide a 90, por eso la ecuación del ejemplo 3.14 sí tiene solución. Y en el ejemplo 3.15, $mcd(42, 105) = 21$, y 21 no divide a 30.

Ya sabemos cómo determinar si una ecuación diofántica tiene solución y o no.

Ahora la siguiente cuestión a resolver es: ¿Cómo obtener las soluciones?

Por un lado, recordemos que $a\,x + b\,y = c$ es equivalente a $\frac{a}{d}\,x + \frac{b}{d}\,y = \frac{c}{d}$. Y ésta ecuación tiene coeficientes más chicos que la ecuación original, y podría resultar más fácil hallar una solución de la segunda ecuación, quizás por inspección, **a ojo**. Por ejemplo, la ecuación $44x - 121y = 1540$ es equivalente a $4x - 11y = 140$ y por inspección se puede obtener una solución: $x = -20, y = -20$.

Pero como no siempre es posible esta resolución **a ojo**, necesitamos un procedimiento más general.

Vamos a ir por pasos.

* Primero, y es el problema central, veremos cómo obtener **una** solución $(x_0, y_0)$ cuando el lado derecho de la ecuación es exactamente el *mcd* de los coeficientes.

* Luego veremos cómo obtener **una** solución $(x_0, y_0)$ cuando el lado derecho es un múltiplo del *mcd*.

* Finalmente, a partir de una solución particular, veremos cómo obtener todas las soluciones.

### 3.5.1. Ecuaciones diofánticas $a\,x + b\,y = mcd(a, b)$

Como se dijo anteriormente, el problema de hallar una solución en ecuaciones de este tipo es central. La situación se basa en el siguiente resultado.

**Teorema 3.8.**
*Dados dos enteros* $a$ *y* $b$, *existen enteros* $x$ *e* $y$ *tales que* $a\,x + b\,y = mcd(a, b)$ ♣

Esos enteros $x$ e $y$ mencionados en el teorema pueden obtenerse con el algoritmo de Euclides. Una vez obtenido el *mcd* de $a$ y $b$ con sucesivas divisiones enteras, se reem-

plazan hacia atrás los sucesivos restos obtenidos, hasta quedar el *mcd* expresado como combinación lineal de $a$ y $b$.

Veamos en ejemplos:

***Ejemplo* 3.16**

*En el ejemplo 3.5 se calculó el* $mcd(1820, 231)$ *con el algoritmo de Euclides, haciendo sucesivas divisiones. Éstas son:*

$$\begin{aligned} 1820 &= 7 \cdot 231 + 203 \\ 231 &= 1 \cdot 203 + 28 \\ 203 &= 7 \cdot 28 + 7 \\ 28 &= 4 \cdot 7 \end{aligned}$$

*Los sucesivos restos obtenidos son 203, 28, 7 y 0. El mcd es* 7*. Despejemos todos los restos (menos el 0) en las divisiones anteriores:*

$$\begin{aligned} 203 &= 1820 - 7 \cdot 231 \qquad \textbf{(a)} \\ 28 &= 231 - 1 \cdot 203 \qquad \textbf{(b)} \\ 7 &= 203 - 7 \cdot 28 \qquad \textbf{(c)} \end{aligned}$$

*Entonces* $7 = 203 - 7 \cdot 28$.

*Luego, en esa igualdad se reemplaza el resto anterior,* 28*, usando (b):*

$$7 = 203 - 7 \cdot 28 = 203 - 7 \cdot (231 - 1 \cdot 203) = 8 \cdot 203 - 7 \cdot 231$$

*A continuación, lo mismo con el resto anterior,* 203*: se reemplaza ese resto en la igualdad anterior, de acuerdo con (a). Resulta:*

$$7 = 8 \cdot (1820 - 7 \cdot 231) - 7 \cdot 231 = 8 \cdot 1820 - 63 \cdot 231$$

*Así, queda expresado el* 7 *como* 1820 *por un entero más* 231 *por un entero:*

$$7 = 8 \cdot 1820 - 63 \cdot 231$$

*Entonces, siendo* $a = 1820$ *y* $b = 231$*, los enteros* $x$ *e* $y$ *del teorema anterior son 8 y -63, respectivamente.*

*En otros términos, una solución de la ecuación diofántica* $1820x + 231y = 7$ *es* $x = 8$*,* $y = -63$*.* ■

***Ejemplo* 3.17**

*Hallemos el* $mcd(5400, 97)$ *y escribámoslo como combinación lineal de* 5400 *y* 97*:*

$$\begin{aligned} 5400 &= 55 \cdot 97 + 65 \\ 97 &= 1 \cdot 65 + 32 \\ 65 &= 2 \cdot 32 + 1 \\ 32 &= 32 \cdot 1 \end{aligned}$$

*Los restos obtenidos son 65, 32, 1 y 0. Entonces,* $mcd(5400, 97) = 1$ *Despejemos los restos:*

$$\begin{aligned} 65 &= 5400 - 55 \cdot 97 \qquad \textbf{(a)} \\ 32 &= 97 - 1 \cdot 65 \qquad \textbf{(b)} \\ 1 &= 65 - 2 \cdot 32 \qquad \textbf{(c)} \end{aligned}$$

*Entonces* $1 = 65 - 2 \cdot 32$. *Remplacemos en esa igualdad el resto 32 usando (b):*

$$1 = 65 - 2 \cdot (97 - 1 \cdot 65) = 3 \cdot 65 - 2 \cdot 97$$

*Luego, reemplazando el resto anterior, 65, usando (a):*

$$1 = 3 \cdot (5400 - 55 \cdot 97) - 2 \cdot 97 = 3 \cdot 5400 - 167 \cdot 97$$

*Se llegó a*

$$3 \cdot 5400 - 167 \cdot 97 = 1$$

*De aquí se obtiene que la ecuación diofántica* $5400x + 97y = 1$ *tiene una solución dada por* $x = 3$, $y = -167$.

*Además podemos deducir que la ecuación diofántica* $5400x - 97y = 1$ *tiene por solución* $x = 3$, $y = 167$ ■

***Ejemplo* 3.18**

*Hallemos el* $mcd(-3402, 114)$. *Como* $mcd(-3402, 114) = mcd(3402, 114)$, *hacemos las sucesivas divisiones enteras comenzando con* 3402 *dividido* 114*:*

$$\begin{array}{rcl} 3402 & = & 29 \cdot 114 + 96 \\ 114 & = & 1 \cdot 96 + 18 \\ 96 & = & 5 \cdot 18 + 6 \\ 18 & = & 3 \cdot 6 \end{array}$$

*Los restos son 96, 18, 6 y 0. Entonces,* $mcd(-3402, 114) = 6$.

*Despejando los restos se obtiene:*

$$\begin{array}{rcll} 96 & = & 3402 - 29 \cdot 114 & \textbf{\textit{(a)}} \\ 18 & = & 114 - 1 \cdot 96 & \textbf{\textit{(b)}} \\ 6 & = & 96 - 5 \cdot 18 & \textbf{\textit{(c)}} \end{array}$$

*Entonces* $6 = 96 - 5 \cdot 18$. *Reemplazando el resto anterior, 18, usando (b) se obtiene:*

$$6 = 96 - 5 \cdot (114 - 1 \cdot 96) = 6 \cdot 96 - 5 \cdot 114$$

*Luego, lo mismo con el resto anterior, 96:*

$$6 = 6 \cdot (3402 - 29 \cdot 114) - 5 \cdot 114 = 6 \cdot 3402 - 179 \cdot 114 = -6 \cdot (-3402) - 179 \cdot 114$$

*Se obtiene que*

$$-6 \cdot (-3402) - 179 \cdot 114 = 6$$

*Así, los coeficientes de la combinación lineal de* $-3402$ *y* 114 *para obtener el mcd* 6 *son* $-6$ *y* $-179$.

*La ecuación diofántica* $114x - 3402y = 6$ *tiene una solución* $x = -179$, $y = -6$. ■

### 3.5.2. Ecuaciones diofánticas $a\,x + b\,y = c$ con c múltiplo de $mcd(a, b)$

Llamemos $d = mcd(a, b)$. Supongamos que la ecuación a resolver es $a\,x + b\,y = c$ siendo $c$ un múltiplo de $d$, es decir, tenemos la ecuación $a\,x + b\,y = p\,d$, con $p = \frac{c}{d}$ entero.

De acuerdo a lo visto en la subsección anterior, se puede hallar una solución $(u_0, v_0)$ de la ecuación $a\,u + b\,v = d$, con $u_0$ y $v_0$ enteros. Entonces, $a\,u_0 + b\,v_0 = d$. Multiplicando por $p$ ambos miembros de esta igualdad, resulta:

$$p(a\ u_0 + b\ v_0) = p\,d = c$$

Que es equivalente a:

$$a\ (p\,u_0) + b\ (p\,v_0) = c$$

Esto indica que $x_0 = p\,u_0$ y $y_0 = p\,v_0$ es solución de la ecuación que queremos resolver, $a\,x + b\,y = c$.

Veamos algunos ejemplos:

***Ejemplo* 3.19**
*Considere la ecuación* $5400x + 97y = 2$. *En el ejemplo 3.17 se halló una solución para la combinación lineal de* 5400 *y* 97 *igualada al* $mcd(5400, 97)$, *que es* 1. *Esto es, se halló una solución para la ecuación* $5400u + 97v = 1$, *y es* $u_0 = 3$ *y* $v_0 = -167$, *ya que*

$$5400 \cdot 3 + 97 \cdot (-167) = 1$$

*Multiplicando ambos miembros* 2*:*

$$5400 \cdot (2 \cdot 3) + 97 \cdot (2 \cdot (-167)) = 2 \cdot 1$$

*Entonces, la solución de la ecuación diofántica dada en este ejemplo es* $x_0 = 2 \cdot 3 = 6$, $y_0 = 2 \cdot (-167) = -334$. ■

***Ejemplo* 3.20**
*Sea la ecuación* $3402x + 114y = 30$. *Del ejemplo 3.18 tenemos una solución para la ecuación* $3402u + 114v = mcd(3402, 114) = 6$, *y es* $u_0 = 6$ *y* $v_0 = -179$. *Por lo tanto, si queremos ahora una combinación lineal de* 3402 *y* 114 *igualada a* 30, *podríamos obtenerla multiplicando por* 5*:*

$$3402 \cdot (5 \cdot u_0) + 114 \cdot (5 \cdot v_0) = 5 \cdot 6$$

$$3402 \cdot 30 + 114 \cdot (-895) = 30$$

*Entonces, una solución de la ecuación dada es* $x_0 = 30$, $y_0 = -895$. ■

### 3.5.3. Solución general de una ecuación diofántica

Finalmente, obtenida **una** solución de una ecuación diofántica, se pueden obtener **todas** las soluciones.

La ecuación $ax + by = c$, considerando $x$ e $y$ variables reales continuas, se puede asociar a su representación gráfica en el plano cartesiano, que es una recta. Hallar las soluciones de una ecuación diofántica es equivalente a encontrar los puntos de esa recta que tienen coordenadas enteras.

Si $b \neq 0$, la ecuación de esta recta se puede escribir $y = -\frac{a}{b}x + \frac{c}{b}$. La pendiente es $-\frac{a}{b}$. Recordemos que la pendiente de una recta es el cociente entre el despazamiento vertical y el desplazamiento horizontal entre dos puntos. Si conocemos un punto $(x_0, y_0)$ de la recta, puede obtenerse otro punto desplazándose según la pendiente: avanzando hacia la derecha $b$ unidades y subiendo $-a$ unidades, llegando al punto $(x_0 + b, y_0 - a)$. Y si $(x_0, y_0)$ tiene coordenadas enteras, $(x_0 + b, y_0 - a)$ también tiene coordenadas enteras. Desplazándose de la misma manera a partir de este nuevo punto se llega a $(x_0 + 2b, y_0 - 2a)$. Y de aquí se puede obtener otro punto de coordenadas enteras, $(x_0 + 3b, y_0 - 3a)$. En general, si $(x_0, y_0)$ es un punto de coordenadas enteras en la recta $ax + by = c$, los puntos de la forma $(x_0 + n \cdot b, y_0 - n \cdot a)$, para cualquier $n$ entero, también tienen coordenadas enteras y están en la recta.

En la figura 3.1 se muestra la recta de ecuación $-2x + 5y = 11$. En este caso $a = -2$ y $b = 5$. Un punto en la recta, con coordenadas enteras es $(x_0, y_0) = (2, 3)$. Otro punto en la recta es $(x_0 + b, y_0 - a) = (2 + 5, 3 - (-2)) = (7, 5)$; y otro es $(7 + 5, 5 - (-2)) = (12, 7)$. La recta tiene infinitos puntos de coordenadas enteras, todos de la forma $(x, y) = (2 + n \cdot 5, 3 - n \cdot (-2))$, con $n$ entero.

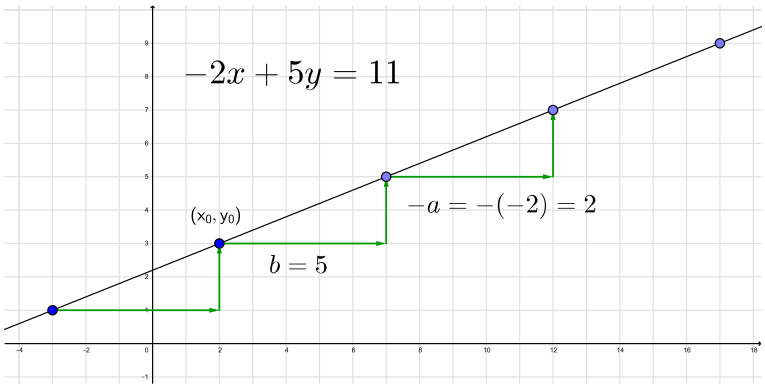



Figura 3.1: Interpretación gráfica de la ecuación diofántica $-2x + 5y = 11$

Consideremos ahora la ecuación diofántica $4x + 10y = 36$. Aquí, $a = 4$ y $b = 10$. Una solución de la ecuación es $(x_0, y_0) = (-6, 6)$. Este punto está en la recta correspondiente (véase figura 3.2). Todos los puntos de la forma $(x_0 + n \cdot b, y_0 - n \cdot a) = (-6 + 10n, 6 - 4n)$ son puntos de la recta con coordenadas enteras (y por lo tanto, soluciones de la ecuación diofántica dada). Algunos de los puntos de esa forma son $(4, 2)$, $(14, -2)$, $(24, -6)$, etc.

Pero hay otros puntos de coordenadas enteras en la recta que no son de esa forma. Nótese que la pendiente es $-\frac{a}{b} = -\frac{4}{10} = -\frac{2}{5}$. Esto indica que, a partir de un punto de la recta, avanzando 5 unidades hacia la derecha, y desplazándose 2 unidades hacia abajo, se obtiene otro punto de la recta. Así, como el punto $(-6, 6)$ está en la recta, también lo está el punto $(-6 + 5, 6 - 2) = (-1, 4)$.

Esto sucede porque $mcd(a, b) = mcd(4, 10) = 2$, y la ecuación dada es equivalente a la ecuación diofántica $2x + 5y = 18$. En esta nueva ecuación los coeficientes son $\frac{a}{mcd(a,b)} = \frac{4}{2} = 2$ y $\frac{b}{mcd(a,b)} = \frac{10}{2} = 5$.

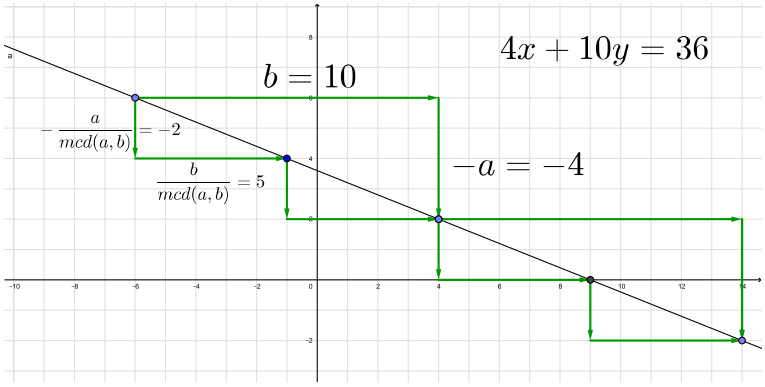



Figura 3.2: Interpretación gráfica de la ecuación diofántica $4x + 10y = 36$

En general, si $d = mcd(a, b)$ divide a $c$, la ecuación diofántica $ax+by = c$ es equivalente a $\frac{a}{d}x+\frac{b}{d}y = \frac{c}{d}$. Y todas las soluciones se pueden obtener a partir de una solución particular $(x_0, y_0)$, y son de la forma $(x_0 + n \cdot \frac{b}{d}, y_0 - n \cdot \frac{a}{d})$ con $n$ entero.

Comprobemos algebraicamente que $x = x_0 + n \cdot \frac{b}{d}$, $y = y_0 - n \cdot \frac{a}{d}$, con $n$ entero, son soluciones de la ecuación diofántica $ax+by = c$. Claramente $x$ e $y$ así definidos son enteros. Reemplazando los valores de $x$ e $y$ en la ecuación, resulta:

$$ax + by = a(x_0 + n \cdot \frac{b}{d}) + b(y_0 - n \cdot \frac{a}{d}) = a \cdot x_0 + n \cdot \frac{ab}{d} + b \cdot y_0 - n \cdot \frac{ba}{d} = a \cdot x_0 + b \cdot y_0 = c$$

**Proposición 3.9.**
*Si $(x_0, y_0)$ es una solución de la ecuación diofántica $a\,x+b\,y = c$, y $d = mcd(a, b)$ entonces todas las soluciones son de la forma*

$$x = x_0 + \frac{b}{d} \cdot n$$
$$y = y_0 - \frac{a}{d} \cdot n$$

*con $n$ entero.* ♢

Esta proposición por un lado, nos dice que si una ecuación diofántica tiene solución, tiene infinitas soluciones. Además, nos indica cómo obtener todas las soluciones.

***Ejemplo* 3.21**
*Considere la ecuación $5400x + 96y = 24$. En el problema 3.7, se obtuvo el $mcd(5400, 96)$:*

$$\begin{aligned} 5400 &= 56 \cdot 96 + 24 \\ 96 &= 4 \cdot 24 \end{aligned}$$

*Entonces, $mcd(5400, 96) = 24$. Como el mcd divide al término independiente de la ecuación, ésta tiene solución.*

*De la primera división realizada, vemos que $mcd(5400, 96) = 24 = 5400 - 59 \cdot 96$. De aquí se obtiene que la ecuación diofántica $5400x + 96y = 24$ admite la solución particular $x_0 = 1$ y $y_0 = -59$.*

*La solución general es*

$$\begin{aligned} x &= x_0 + \tfrac{96}{24} \cdot n = 1 + 4n \\ y &= y_0 - \tfrac{5400}{24} \cdot n = -59 - 225n \end{aligned}$$

*con $n$ entero.*

*Por ejemplo, con $n = 1$, se tiene la solución $x = 5$, $y = -284$; con $n = -1$ se tiene $x = -3$, $y = 166$, etc.* ■

***Ejemplo* 3.22**

*Consideremos la ecuación $116x + 40y = 4$.*

*Nótese que la ecuación dada es equivalente a $29x + 10y = 1$. Para esta ecuación una solución evidente es $x_0 = -1, y_0 = 3$. Luego, ésta también es solución de la ecuación original.*

*Si uno no tuviera habilidad para la resolución por inspección o por tanteo, se puede proceder de la siguiente manera:*

*Del problema 3.6 tenemos el $mcd(116, 40)$:*

$$\begin{aligned} 116 &= 2 \cdot 40 + 36 \\ 40 &= 1 \cdot 36 + 4 \\ 36 &= 9 \cdot 4 \end{aligned}$$

*Los restos obtenidos son 36, 4 y 0. Entonces, $mcd(116, 40) = 4$. Despejando los restos se obtiene:*

$$\begin{aligned} 36 &= 116 - 2 \cdot 40 \qquad \textbf{(a)} \\ 4 &= 40 - 1 \cdot 36 \qquad \textbf{(b)} \end{aligned}$$

*Entonces $4 = 40 - 1 \cdot 36$. Reemplazando el 36 de acuerdo a (a):*

$$4 = 40 - 1 \cdot (116 - 2 \cdot 40) = 40 - 1 \cdot 116 + 2 \cdot 40 = -1 \cdot 116 + 3 \cdot 40$$

*Ordenando:*

$$116 \cdot (-1) + 40 \cdot 3 = 4$$

*Así, una solución de la ecuación dada es $x_0 = -1$ y $y_0 = 3$. Y la solución general es*

$$\begin{aligned} x &= x_0 + \tfrac{40}{4} \cdot n = -1 + 10n \\ y &= y_0 - \tfrac{116}{4} \cdot n = 3 - 29n \end{aligned}$$

*Por ejemplo, son soluciones*
*$x = 9$, $y = -26$ (con n=1)*
*$x = -11$, $y = 32$ (con n=-1)*
*$x = 49$, $y = -142$ (con n=5)* ■

***Ejemplo* 3.23**

*Consideremos la ecuación $116x + 40y = 12$. Como 12 es divisible por $mcd(116, 40) = 4$, la ecuación tiene solución. En el ejemplo anterior vimos que*

$$116 \cdot (-1) + 40 \cdot 3 = 4$$

*Como queremos que el lado derecho sea 12, multiplicamos miembro a miembro por 3:*

$$\begin{aligned} 116 \cdot (-1) \cdot 3 + 40 \cdot 3 \cdot 3 &= 4 \cdot 3 \\ 116 \cdot (-3) + 40 \cdot 9 &= 12 \end{aligned}$$

*Entonces, una solución particular de la ecuación es* $x_0 = -3$ *y* $y_0 = 9$. *La solución general es*

$$\begin{aligned} x &= x_0 + \tfrac{40}{4} \cdot n = -3 + 10n \\ y &= y_0 - \tfrac{116}{4} \cdot n = 9 - 29n \end{aligned}$$

***Ejemplo* 3.24**

*Resolvamos la ecuación diofántica* $168x - 744y = 264$. *Los coeficientes son* 168 *y* $-744$, *debemos calcular el* $mcd(168, -744) = mcd(168, 744)$.

$$\begin{aligned} 744 &= 4 \cdot 168 + 72 \\ 168 &= 2 \cdot 72 + 24 \\ 72 &= 3 \cdot 24 \end{aligned}$$

*El mcd es* 24. *Como el término independiente de la ecuación (*264*) es divisible por* 24, *la ecuación tiene solución.*

*Por un lado, puede hallarse una solución considerando la ecuación equivalente obtenida al dividir ambos miembros por el mcd de los coeficientes:* $7x - 31y = 11$. *¿Se puede obtener una solución de esta ecuación por inspección? Esta vez no parece tan sencillo. Pero si pensamos en la ecuación* $7u - 31v = 1$, *se obtiene fácilmente la solución dada por* $u_0 = 9, v_0 = 2$. *Como queremos resolver la ecuación con el lado derecho igual a 11, una solución es* $x_0 = 11u_0 = 99, y_0 = 11v_0 = 22$.

*También puede hallarse una solución particular de la ecuación dada con las sucesivas divisiones del Algoritmo de Euclides, haciendo reemplazos hacia atrás. En primer lugar, despejemos los restos no nulos de las divisiones (que son 72 y 24):*

$$\begin{aligned} 72 &= 744 - 4 \cdot 168 \qquad \textbf{(a)} \\ 24 &= 168 - 2 \cdot 72 \qquad \textbf{(b)} \end{aligned}$$

*Entonces:* $mcd(168, -744) = 24 = 168 - 2 \cdot 72$

*Reemplazando el resto 72 en la igualdad anterior, usando (a):*

$$24 = 168 - 2 \cdot (744 - 4 \cdot 168) = 168 - 2 \cdot 744 + 8 \cdot 168 = 9 \cdot 168 - 2 \cdot 744$$

*Resulta:*

$$168 \cdot 9 - 744 \cdot 2 = 24$$

*Necesitamos que el lado derecho sea* 264, *entonces multiplicando término a término por* 11, *se llega a:*

$$\begin{gathered} 168 \cdot 9 \cdot 11 - 744 \cdot 2 \cdot 11 = 24 \cdot 11 \\ 168 \cdot 99 - 744 \cdot 22 = 264 \end{gathered}$$

*De aquí surge una solución de la ecuación:* $x_0 = 99$ *y* $y_0 = 22$.

*Y la solución general es*

$$\begin{aligned} x &= x_0 + \tfrac{(-744)}{24} \cdot n = 99 - 31n \\ y &= y_0 - \tfrac{168}{24} \cdot n = 22 - 7n \end{aligned}$$

*Algunas soluciones son:*

$x = 68$, $y = 15$ *(con* $n = 1$*)*
$x = 130$, $y = 29$ *(con* $n = -1$*)*
$x = 37$, $y = 8$ *(con* $n = 2$*)* ■

***Ejemplo* 3.25**
*Un dispositivo de memoria tiene capacidad de almacenar 2000 MB. Se quieren guardar archivos de video y archivos de imágenes. Cada video tiene un tamaño de 235 MB, y cada imagen tiene un tamaño de 10 Mb. ¿Cuántos archivos de cada tipo se puede almacenar, de modo que no quede espacio libre en la memoria?*

*La situación puede modelarse con la ecuación* $235x+10y = 2000$*, donde* $x$ *es la cantidad de archivos de video grabados en la memoria, e* $y$ *es la cantidad de imágenes guardadas.*

*Resolvamos esa ecuación diofántica. El mcd de los coeficientes es* $mcd(235, 10) = 5$*. Como* $2000$ *es divisible por* $5$*, la ecuación tiene solución.*

*Expresemos* $5$ *como combinación lineal de* $235$ *y* $10$*. Por inspección, obtenemos que* $235 \cdot 1 + 10 \cdot (-23) = 5$*. Multiplicando ambos miembros por* $400$*, resulta:*

$$235 \cdot 400 + 10 \cdot (-9200) = 2000$$

*Tenemos la solución particular* $x_0 = 400$ *y* $y_0 = -9200$*. La solución general de la ecuación es*

$$x = x_0 + \tfrac{10}{5} \cdot n = 400 + 2n$$
$$y = y_0 - \tfrac{235}{5} \cdot n = -9200 - 47n$$

*Como* $x$ *e* $y$ *representan cantidades, los valores que resuelven el problema deben ser positivos. Debe cumplirse que:*

$$\begin{array}{rcl} x & \geqslant & 0 \\ 400 + 2n & \geqslant & 0 \\ n & \geqslant & -200 \end{array} \qquad \begin{array}{rcl} y & \geqslant & 0 \\ -9200 - 47n & \geqslant & 0 \\ n & \leqslant & -9200/47 \approx -195{,}7 \end{array}$$

*Entonces, cada* $n$ *entre* $-200$ *y* $-196$*, da una solución del problema:*

$n = -200 \to x = 0, y = 200$*. La memoria se llena con 200 imágenes*
$n = -199 \to x = 2, y = 153$*. La memoria se llena con 2 videos y 153 imágenes*
$n = -198 \to x = 4, y = 106$*. La memoria se llena con 4 videos y 106 imágenes*
$n = -197 \to x = 6, y = 59$*. La memoria se llena con 6 videos y 59 imágenes*
$n = -196 \to x = 8, y = 12$*. La memoria se llena con 8 videos y 12 imágenes*

## 3.6. Congruencias

La división entera fundamenta la *aritmética modular*, que se ocupa de establecer relaciones de equivalencias entre los enteros, basadas en los restos en la división por cierto entero $m$.

### 3.6.1. Definiciones, propiedades y ejemplos

Dados dos enteros $a$ y $b$, se dice que $a$ ES CONGRUENTE A $b$ MÓDULO $m$ ($m$ entero positivo) y se denota

$$a \equiv b \ (mod\ m)$$

si ambos tienen el mismo resto en la división entera por $m$. Claramente, $a$ es congruente a $b$ módulo $m$ si y sólo si $b$ es congruente a $a$ módulo $m$. Entonces se puede decir que $a$ y $b$ son congruentes módulo $m$.

***Ejemplo* 3.26**

$$\begin{array}{rcl|rcl|rcl} 79 & \equiv & 3\ (mod\ 4) & 18 & \equiv & 10\ (mod\ 4) & -16 & \equiv & 4\ (mod\ 4) \\ 79 & \equiv & 15\ (mod\ 4) & -11 & \equiv & 1\ (mod\ 4) & -2 & \equiv & 2\ (mod\ 4) \\ 102 & \equiv & 52\ (mod\ 10) & -11 & \equiv & 1\ (mod\ 2) & -14 & \equiv & 0\ (mod\ 2) \\ 100 & \equiv & 1\ (mod\ 11) & -11 & \equiv & 9\ (mod\ 10) & 1 & \equiv & -10\ (mod\ 11) \end{array}$$

Si $a \equiv b\ (mod\ m)$, entonces, la división entera de $a$ por $m$ y de $b$ por $m$ tienen el mismo resto:

$$a = q_1 \cdot m + r$$
$$b = q_2 \cdot m + r$$

Restando estas dos igualdades resulta:

$$a - b = (q_1 - q_2) \cdot m$$

Es decir, $a$ y $b$ son congruentes módulo $m$ si y sólo si $a - b$ es múltiplo de $m$. Ésta es otra caracterización de la congruencia.

**Proposición 3.10.**
*$a$ y $b$ son congruentes módulo $m$ si y sólo si $a - b$ es divisible por $m$.* ♢

***Ejemplo* 3.27**

$570 \equiv 30\ (mod\ 54)$ *ya que* $570 - 30 = 540$*, es múltiplo de* 54
$1022 \equiv 557\ (mod\ 93)$ *ya que* $1022 - 557 = 465$*, es múltiplo de* 93 ■
$1022 \equiv -187\ (mod\ 93)$ *ya que* $1022 - (-187) = 1209$*, es múltiplo de* 93

Hay $m$ posibles restos en la división por $m$, y son: 0, 1, 2, ..., $m - 1$. Cualquier número entero tiene alguno de estos restos cuando se lo divide por $m$. Todos los enteros que tienen resto $i$ forman la CLASE DE CONGRUENCIA DE RESTO $i$ módulo $m$. La clase se denota $[i]$.

Todo entero pertenece a una y sólo una clase de congruencia módulo $m$. Entonces, el conjunto $\mathbb{Z}$ de los números enteros se puede particionar en $m$ clases, [0], [1], ..., $[m - 1]$. Esta partición de $\mathbb{Z}$ en clases de congruencias se denota $\mathbb{Z}_m$.

***Ejemplo* 3.28**
*Sea $m = 7$. Entonces $\mathbb{Z}_7$ está formado por las siguientes siete clases de congruencia:*

$[0] = \{... -14, -7, 0, 7, 14, 21, ...\}$ *(enteros de resto 0 en la división por 7)*
$[1] = \{... -13, -6, 1, 8, 15, 22, ...\}$ *(enteros de resto 1 en la división por 7)*
$[2] = \{... -12, -5, 2, 9, 16, 23, ...\}$ *(enteros de resto 2 en la división por 7)*
$[3] = \{... -11, -4, 3, 10, 17, 24, ...\}$ *(enteros de resto 3 en la división por 7)*
$[4] = \{... -10, -3, 4, 11, 18, 25, ...\}$ *(enteros de resto 4 en la división por 7)*
$[5] = \{... -9, -2, 5, 12, 19, 26, ...\}$ *(enteros de resto 5 en la división por 7)*
$[6] = \{... -8, -1, 6, 13, 20, 27, ...\}$ *(enteros de resto 6 en la división por 7)* ■

Las operaciones de la aritmética entera se comportan bien frente a la congruencia (por eso hablamos de aritmética modular):

**Proposición 3.11.**
*Sean $a$, $b$, $c$, $d$ y $m$ enteros, con $m > 0$ tales que $a \equiv b \ (mod\ m)$ y $c \equiv d \ (mod\ m)$. Entonces:*

- $a + c \equiv b + d \ (mod\ m)$ *(es decir, dos congruencias se pueden sumar miembro a miembro)*
- $a \cdot c \equiv b \cdot d \ (mod\ m)$ *(es decir, dos congruencias se pueden multiplicar miembro a miembro).* ♢

Cuando se realizan operaciones aritméticas módulo $m$, se puede emplear cualquier elemento de la clase de congruencia correspondiente. En particular, suele usarse el resto módulo $m$, es decir, el representante de la clase que sea mayor o igual a 0, y menor a $m$. Luego de la proposición 3.11 se daran algunos ejemplos.

Por ejemplo, si se quiere calcular el producto modular 2343·10642 módulo 7, se considera que $2343 \equiv 5 \ (mod\ 7)$ y $10642 \equiv 2 \ (mod\ 7)$, luego, $2343 \cdot 10645 \equiv 5 \cdot 2 \equiv 10 \equiv 3 \ (mod\ m)$.

Como corolario de esta proposición, dado que un número es congruente a sí mismo, (es decir, $c \equiv c \ (mod\ m)$), entonces en una congruencia se puede sumar un entero y se puede multiplicar por un entero cada miembro: Si $a \equiv b \ (mod\ m)$ entonces $a + c \equiv b + c \ (mod\ m)$ y $a \cdot c \equiv b \cdot c \ (mod\ m)$.

También como corolario de la proposición anterior, surge que se puede elevar a una potencia positiva $n$ cada lado de una congruencia. Es decir, si $a \equiv b \ (mod\ m)$ entonces $a^n \equiv b^n \ (mod\ m)$.

En general, no se puede dividir ambos miembros de una congruencia. Observe que $4 \equiv 2 \ (mod\ 2)$. Al dividir ambos miembros por 2, resulta una congruencia no válida: $2 \equiv 1 \ (mod\ 2)$. Esto es de esperarse cuando se divide por un número que no es coprimo con el módulo.

Sólo se puede dividir ambos miembros de una congruencia por un número que sea coprimo con el módulo:
Si $a \equiv b \ (mod\ m)$, $c|a$, $c|b$ y $mcd(c, m) = 1$ entonces $a/c \equiv b/c \ (mod\ m)$.

La siguiente proposición reúne otras propiedades importantes de las congruencias:

**Proposición 3.12.**

- *La relación de congruencia es transitiva. Es decir:*
  *Si $a \equiv b \ (mod\ m)$ y $b \equiv c \ (mod\ m)$, entonces $a \equiv c \ (mod\ m)$*
- *Un número es congruente a 0 módulo $m$ si y sólo si es múltiplo de $m$.*
- *Si $a \equiv b \ (mod\ m)$, entonces $a - m \equiv b \ (mod\ m)$.* ♢

***Ejemplo* 3.29**
*Hallemos la última cifra del número $7^{41}$. La última cifra es el resto de la división por 10. Entonces, usando las propiedades antes enunciadas:*

$7 \equiv 7\ (mod\ 10)$
$7^2 \equiv 7^2 \equiv 49 \equiv 9\ (mod\ 10) \quad \Rightarrow \quad 7^2 \equiv 9\ (mod\ 10)$
$(7^2)^2 \equiv 9^2 \equiv 81 \equiv 1\ (mod\ 10) \quad \Rightarrow \quad 7^4 \equiv 1\ (mod\ 10)$
$(7^4)^{10} \equiv 1^{10} \equiv 1\ (mod\ 10) \quad \Rightarrow \quad 7^{40} \equiv 1\ (mod\ 10)$
$7 \cdot (7^{40}) \equiv 7 \cdot 1 \equiv 7\ (10) \quad \Rightarrow \quad 7^{41} \equiv 7\ (mod\ 10)$

*Luego, la última cifra de* $7^{41}$ *es 7.* ■

***Ejemplo* 3.30**

*¿Cuál es el resto de dividir* $9^{123}$ *por* 11*? Empleando las propiedades de la proposición anterior:*

$9 \equiv 9\ (mod\ 11)$
$9^2 \equiv 81 \equiv 4\ (mod\ 11)$
$9 \cdot 9^2 \equiv 9 \cdot 4 \equiv 36 \equiv 3\ (mod\ 11) \quad \Rightarrow \quad 9^3 \equiv 3\ (mod\ 11)$
$9^3 \cdot 9^2 \equiv 3 \cdot 4 \equiv 12 \equiv 1\ (mod\ 11) \quad \Rightarrow \quad 9^5 \equiv 1\ (mod\ 11)$
$(9^5)^{24} \equiv 1^{24} \equiv 1\ (mod\ 11) \quad \Rightarrow \quad 9^{120} \equiv 1\ (mod\ 11)$
$9^{120} \cdot 9^3 \equiv 1 \cdot 3\ (mod\ 11) \quad \Rightarrow \quad 9^{123} \equiv 3\ (mod\ 11)$

*Entonces, el resto de dividir* $9^{123}$ *por* 11 *es* 3. ■

Un concepto central en congruencias es la noción de inverso modular:

El entero $j$ es INVERSO de $a$ módulo $m$ si

$$a \cdot j \equiv 1\ (mod\ m)$$

Claramente, si el entero $j$ es inverso del entero $a$ módulo $m$, entonces $a$ es inverso de $j$ módulo $m$.

***Ejemplo* 3.31**

*Hallemos inversos módulo* 5*:*

$1 \cdot 1 \equiv 1\ (mod\ 5) \Rightarrow 1$ *es inverso de* 1 *módulo* 5.
$2 \cdot 3 \equiv 1\ (mod\ 5) \Rightarrow 2$ *es inverso de* 3 *módulo* 5. *Y* 3 *es inverso de* 2 *módulo* 5.
$4 \cdot 4 \equiv 1\ (mod\ 5) \Rightarrow 4$ *es inverso de* 4 *módulo* 5.
*El* 0 *no tiene inverso.* ■

***Ejemplo* 3.32**

*Calculemos los inversos módulo 9*

$1 \cdot 1 \equiv 1\ (mod\ 9) \Rightarrow 1$ *es inverso de* 1 *módulo* 9.
$2 \cdot 5 \equiv 1\ (mod\ 9) \Rightarrow 2$ *es inverso de* 5 *módulo* 9. *Y* 5 *es inverso de* 2 *módulo* 9.
$4 \cdot 7 \equiv 1\ (mod\ 9) \Rightarrow 4$ *es inverso de* 7 *módulo* 9. *Y* 7 *es inverso de* 4 *módulo* 9.
$8 \cdot 8 \equiv 1\ (mod\ 9) \Rightarrow 8$ *es inverso de* 8 *módulo* 9.
*Los enteros* 0, 3 *y* 6 *no tienen inverso módulo* 9. ■

¿Por qué algunos números no tienen inverso en cierto módulo?

De la definición de inverso, se deduce que $a$ tiene inverso módulo $m$ si y sólo si existe un $j$ tal que $a \cdot j - 1$ es múltiplo de $m$. Es decir, $a \cdot j - 1 = m \cdot y$. O sea, $a$ tiene inverso módulo $m$ si y sólo si existen enteros $j$ e $y$ tal que

$$a \cdot j - m \cdot y = 1$$

Y esta ecuación diofántica tiene solución si y sólo si $mcd(a, m)$ divide a 1. Pero como el único divisor positivo de 1 es 1, se tiene el siguiente resultado:

**Proposición 3.13.**
*a tiene inverso módulo m si y sólo si* $mcd(a, m) = 1$. ♢

Volviendo al ejemplo 3.32, 0, 3 y 6 no tienen inversos módulo 9 porque tienen divisores comunes con 9 mayores a 1.

Nótese que si $j$ es un inverso de $a$ módulo $m$, $j + k \cdot m$ también es inverso de $a$ con cualquier entero $k$. Todos los inversos de $a$ pertenecen a la misma clase de restos, módulo $m$.

Si $a$ tiene inverso módulo $m$, tiene un único inverso positivo menor que $m$. Si a tal entero lo llamamos $r$, entonces cualquier inverso de $a$ pertenece a la clase de restos $[r]$ en $\mathbb{Z}_m$.

Se ha visto que 2 es inverso de 5 módulo 9. También son inversos de 5 los enteros 11 ($11 \cdot 5 \equiv 1\,(mod\,9)$), 20 ($20 \cdot 5 \equiv 1\,(mod\,9)$), 29, -7, etc.; todos pertenecientes a la clase de restos [2].

***Ejemplo* 3.33**
*Hallemos el inverso de* 81 *módulo* 250. *Es decir, debemos hallar un* $j$ *tal que*

$$81j \equiv 1\ (mod\ 250)$$

*Para ello, busquemos la solución de la ecuación diofántica* $81j + 250y = 1$. *Dividiendo sucesivamente:*

$$\begin{aligned} 250 &= 3 \cdot 81 + 7 \\ 81 &= 11 \cdot 7 + 4 \\ 7 &= 1 \cdot 4 + 3 \\ 4 &= 1 \cdot 3 + 1 \\ 3 &= 1 \cdot 3 \end{aligned}$$

*Los restos obtenidos son 7, 4, 3, 1 y 0. Despejando los restos no nulos se tiene:*

$$\begin{aligned} 7 &= 250 - 3 \cdot 81 && \textbf{(a)} \\ 4 &= 81 - 11 \cdot 7 && \textbf{(b)} \\ 3 &= 7 - 1 \cdot 4 && \textbf{(c)} \\ 1 &= 4 - 1 \cdot 3 && \textbf{(d)} \end{aligned}$$

*Entonces, de (d) se tiene* $1 = 4 - 1 \cdot 3$. *Reemplazando el resto anterior,* 3, *según (c) se obtiene:*

$$1 = 4 - 1 \cdot (7 - 1 \cdot 4) = 4 - 1 \cdot 7 + 1 \cdot 4 = 2 \cdot 4 - 1 \cdot 7$$

*Reemplazando* 4 *según (b):*

$$1 = 2 \cdot (81 - 11 \cdot 7) - 1 \cdot 7 = 2 \cdot 81 - 22 \cdot 7 - 1 \cdot 7 = 2 \cdot 81 - 23 \cdot 7$$

*Finalmente, reemplazando* 7 *según (a):*

$$1 = 2 \cdot 81 - 23 \cdot (250 - 3 \cdot 81) = 2 \cdot 81 - 23 \cdot 250 + 69 \cdot 81 = 71 \cdot 81 - 23 \cdot 250$$

*Se obtiene*

$$71 \cdot 81 - 23 \cdot 250 = 1$$

*Entonces,* $71 \cdot 81 \equiv 1\ (mod\ 250)$, *y* 71 *es inverso de* 81 *módulo* 250. *Más generalmente, los inversos de* 81 *son los enteros en la clase de restos* [71] *en* $\mathbb{Z}_{250}$, *que también se pueden escribir como* $j = 71 + 250k$, *con* $k$ *entero.* ■

***Ejemplo* 3.34**
*Hallemos el inverso de* 33 *módulo* 280*. Dividiendo sucesivamente:*

$$\begin{aligned} 280 &= 8 \cdot 33 + 16 \\ 33 &= 2 \cdot 16 + 1 \\ 16 &= 16 \cdot 1 \end{aligned}$$

*Reemplazando hacia atrás:*
$1 = 33 - 2 \cdot 16 = 33 - 2 \cdot (280 - 8 \cdot 33) = 17 \cdot 33 - 2 \cdot 280.$
*Se obtiene*

$$17 \cdot 33 - 2 \cdot 280 = 1$$

*Entonces,* $17 \cdot 33 \equiv 1$ *(mod* 280*), y* 17 *es inverso de* 33 *módulo* 280*. Todos los inversos de* 33 *se pueden obtener con* $j = 17 + 280k$*, con* $k$ *entero, o equivalentemente, decir que pertenecen a la clase de restos* $[17]$ *en* $\mathbb{Z}_{280}$*.* ■

***Ejemplo* 3.35**
*Hallemos el inverso de* 10 *módulo* 21*. Dividiendo sucesivamente:*

$$\begin{aligned} 21 &= 2 \cdot 10 + 1 \\ 10 &= 10 \cdot 1 \end{aligned}$$

*Reemplazando hacia atrás:*
$1 = 21 - 2 \cdot 10.$
*Entonces,* $(-2) \cdot 10 \equiv 1$ *(mod* 21*), y* $(-2)$ *es el inverso de* 10 *módulo* 21*. Como* $-2$ *pertenece a la clase de restos* $[19]$ *en* $\mathbb{Z}_{21}$*, ésta es la clase de restos de los inversos de 10.* ■

### 3.6.2. Aplicaciones de congruencias

**Funciones de dispersión.** Un problema habitual en programación consiste en almacenar en una tabla una gran cantidad de datos alfanuméricos, de tal forma que se puedan recuperar lo más rápidamente posible cuando se busque un dato.

Un modo de almacenamiento es, para cada dato $k$ que se quiere guardar, hacerlo en una dirección de memoria dada por una función $h(k)$. Esta función es llamada función de dispersión (o función hash). Si la función asigna la misma dirección a dos datos distintos, es decir, $h(k_1) = h(k_2)$ para $k_1 \neq k_2$, se dice que ocurre una colisión. Lo ideal sería que esta función sea fácil de calcular, y además, inyectiva [3], es decir, que no produzca colisiones. Esto no es tan fácil de lograr. En lugar de pedirle inyectividad, es suficiente con requerir que la probabilidad de que ocurran colisiones sea baja.

Una función de dispersión comúnmente usada es $h(k) \equiv k (mod\ m)$, donde $m$ es aproximadamente la cantidad de posiciones de memoria disponibles para el conjunto de datos, y $0 \leqslant h(k) < m$. Así, la imagen de esta función es el conjunto de direcciones 0, 1, ..., $m-1$.

Para poner un ejemplo, suponga que se quiere almacenar los registros de los alumnos de cierto curso mediante el DNI. Si el curso tiene a lo sumo 50 alumnos, se requieren como máximo 50 posiciones de memoria. Se podría usar la función $h(DNI) \equiv DNI (mod\ 50)$. Por ejemplo, h(33024887)=37, h(32914054)=4. Entonces el registro del alumno con DNI 33024887 se almacenará en la posición de memoria 37 y el registro del alumno con DNI 32914054 se almacenará en la posición de memoria 4.

[3] La inyectividad supone que si $k_1 \neq k_2$, entonces $h(k_1) \neq h(k_2)$

Esta función no es inyectiva (se pueden tener distintos datos, congruentes módulo $m$, y por lo tanto, la función le asigna la misma dirección de memoria), entonces, puede dar lugar a colisiones. La colisión se puede resolver asignando la dirección de la primera posición de memoria libre. La probabilidad de colisiones cuando la cantidad de registros es igual al módulo es muy alta. Una manera de disminuir la probabilidad de colisiones es usar un módulo mayor. Por ejemplo, para el registro de los 50 alumnos, usar módulo 100. Claro que esto requiere tener más posiciones de memoria reservadas, pero, como se dijo, tiene la ventaja de disminuir la probabilidad de colisiones.

**Eventos periódicos**. En nuestro sistema de medición del tiempo, usamos congruencias módulo 60 en los segundos y minutos, congruencias módulo 24 en horas, congruencias módulo 7 en semanas.

Suponga que un evento ocurre periódicamente, cada $y$ horas. Si el evento ocurre en algún momento a las $b$ horas, el evento siguiente tiene lugar $y + b$ $(mod\ 24)$ horas de un determinado día.

Por ejemplo, una fábrica implementa controles de fallas en una línea de producción cada 100 horas. En un determinado día, el control se realiza a las 10 hs. Para saber a qué hora se realizará el siguiente control, se suman las 100 horas que transcuren, y se calcula su resto módulo 24: $10 + 100 = 110 \equiv 14\ (mod\ 24)$. De modo que el siguiente control se realizará a las 14 hs del día correspondiente.

En otra situación, suponga que un día lunes se constituye un plazo fijo por 180 días. Interesa saber qué día de la semana vence tal plazo. Asumiendo que lunes es el día 1 de la semana, el martes es día 2, etc., se debe calcular el resto de 1+180 módulo 7. Esto es $1 + 180 = 181 \equiv 6\ (mod\ 7)$. Entonces, el plazo fijo vence el día 6 de la semana, esto es, el sábado de la semana correspondiente.

**Criterios de divisibilidad**. Las congruencias permiten obtener criterios de divisibilidad por ciertos divisores. Por ejemplo, comencemos con el criterio de divisibilidad por 3.

Dado un número entero positivo $a$ de $n$ dígitos, se puede escribir en su descomposición decimal de la siguiente forma: $a = a_n 10^n + a_{n-1}10^{n-1} + ... + a_1 10^1 + a_0 10^0$.

Veamos las siguientes congruencias:

$$\begin{array}{lcl} 10^0 \equiv 1\ (mod\ 3) & \Rightarrow & a_0 10^0 \equiv a_0\ (mod\ 3) \\ 10^1 \equiv 1\ (mod\ 3) & \Rightarrow & a_1 10^1 \equiv a_1\ (mod\ 3) \\ ... & \Rightarrow & ... \\ 10^{n-1} \equiv 1^{n-1} \equiv 1\ (mod\ 3) & \Rightarrow & a_{n-1}10^{n-1} \equiv a_{n-1}\ (mod\ 3) \\ 10^n \equiv 1^n \equiv 1\ (mod\ 3) & \Rightarrow & a_n 10^n \equiv a_n\ (mod\ 3) \end{array}$$

Sumando miembro a miembro las congruencias de la derecha:

$$a_n 10^n + a_{n-1}10^{n-1} + ... + a_1 10^1 + a_0 10^0 \equiv (a_n + a_{n-1} + ... + a_1 + a_0)\ \ (mod\ 3)$$

Ahora, el miembro izquierdo es $a$, y el derecho es la suma de los dígitos de $a$. De aquí surge el conocido criterio: *Un entero positivo es multiplo de 3 si y sólo si la suma de sus cifras es múltiplo de 3.*

El criterio de divisibilidad por 9 es similar.

Veamos finalmente el criterio de divisibilidad por 11, a partir de las congruencias:

$$\begin{array}{ll} 10^0 \equiv 1 \ (mod\ 11) & \Rightarrow \quad a_0 10^0 \equiv a_0 \ (mod\ 11) \\ 10^1 \equiv -1 \ (mod\ 11) & \Rightarrow \quad a_1 10^1 \equiv -a_1 \ (mod\ 11) \\ 10^2 \equiv (-1)\cdot(-1) \equiv 1 \ (mod\ 11) & \Rightarrow \quad a_2 10^2 \equiv a_2 \ (mod\ 11) \\ 10^3 = 10 \cdot 10^2 \equiv (-1)\cdot 1 \equiv -1 \ (mod\ 11) & \Rightarrow \quad a_3 10^3 \equiv -a_3 \ (mod\ 11) \\ ... & \Rightarrow \quad ... \\ 10^n \equiv (-1)^n \ (mod\ 11) & \Rightarrow \quad a_n 10^n \equiv (-1)^n a_n \ (mod\ 11) \end{array}$$

Sumando las congruencias de la derecha:

$$a_n 10^n + ... + a_3 10^3 + a_2 10^2 +_1 10^1 + a_0 10^0 \equiv ((-1)^n a_n + ... - a_3 + a_2 - a_1 + a_0) \ \ (mod\ 11)$$

De aquí surge el conocido criterio de divisibilidad por 11: *Un entero positivo es múltiplo de 11 si y sólo si la suma alternada de sus dígitos es múltiplo de 11.* La suma alternada comienza sumando el último dígito, restando el anteúltimo, etc.

**Números pseudoaleatorios** Quien tenga algo de conocimiento de programación seguramente conoce una función para generar números aleatorios. Puede llamarse rand, random, uniform, de acuerdo al lenguaje. Son funciones que devuelven números aleatorios, tomados de una distribución uniforme entre 0 y 1.

Pero... ¿cómo se generan? ¿Cuál es el algoritmo que permite obtener tales números?

Las secuencias de números generadas por algoritmos determinísticos no son verdaderamente aleatorias, sino pseudoaleatorias. El objetivo de un generador de números pseudoaleatorios es producir una secuencia de números entre 0 y 1 que tenga las propiedades ideales de distribución uniforme, es decir, una sucesión que parece aleatoria.

El más popular de los generadores de números aleatorios se conoce como *método lineal de congruencias*. Dados una semilla inicial $x_0$, un multiplicador $a$, un módulo $m$ y un desplazamiento $b$ (todos positivos), la secuencia de números aleatorios se genera con:

$$x_{n+1} \equiv a x_n + b \, (mod\, m)$$

Si $b = 0$, se llama generador multiplicativo, en caso contrario se llama mixto. Esto quiere decir que dado el anterior número aleatorio, $x_n$, se genera el siguiente tomando el resto de $ax_n + b$ dividido por $m$. Entonces, se generan números entre 0 y $m - 1$. Para ubicar los números entre 0 y 1, se normalizan: $X_n = x_n/m$. Por lo tanto, los números normalizados sólo pueden tomar los valores 0, $1/m$, $2/m$, $3/m$,..., $(m-1)/m$.

Claramente, la secuencia es cíclica. Esto es porque si se generan más de $m$ números, inevitablemente ocurre una repetición de algún valor. Y si $x_n = x_k$, entonces $x_{n+1} = x_{k+1}$, y entonces $x_{n+2} = x_{k+2}$, y $x_{n+3} = x_{k+3}$, etc...

Por ejemplo, si $x_0 = 0$, $a = 2$, $b = 3$, $m = 10$, se generan los números $x_1 = 3, x_2 = 9, x_3 = 1, x_4 = 5, x_5 = 3, x_6 = 9, x_7 = 1, ...$

El conjunto de parámetros dado genera una sucesión cíclica de longitud 4, ya que $x_1 = x_5$, $x_2 = x_6$, y en general, $x_n = x_{n+4}$.

En cambio, con $x_0 = 0$, $a = 1$, $b = 3$, $m = 10$, se obtienen $x_1 = 3, x_2 = 6, x_3 = 9, x_4 = 2, x_5 = 5, x_6 = 8, x_7 = 1, x_8 = 4, x_9 = 7, x_{10} = 0, x_{11} = 3, x_{12} = 6, ...$

En este caso, el ciclo tiene longitud 10, $x_n = x_{n+10}$.

Con los parámetros $x_0 = 0$, $a = 5$, $b = 3$, $m = 10$, se obtienen $x_1 = 3, x_2 = 8x_3 = 3, x_4 = 8, ...$

Ahora, el ciclo aparece en el segundo término, $x_n = x_{n+2}$.

Vemos con estos ejemplos que la longitud del ciclo depende fuertemente de los parámetros $a$, $b$, $m$ y $x_0$. Por supuesto, ya que el ciclo no puede evitarse, se quiere que el ciclo sea tan largo como sea posible. La longitud más larga que puede tener un ciclo es $m$. Cuando esto ocurre, se dice que tiene *período completo*.

Puede probarse que un generador de números pseudoaleatorios con el método lineal de congruencias es de período completo si y sólo si $m$ y $b$ son coprimos, $a - 1$ es divisible por los divisores primos de $m$ y si 4 divide a $m$, entonces 4 divide a $a - 1$. (es decir, $mcd(b, m) = 1$, $a - 1 \equiv 0\,(mod\ p)$, siendo $p$ divisor primo de $m$, y si $m \equiv 0\,(mod\ 4)$, entonces $a - 1 \equiv 0\,(mod\ 4)$).

Se desprende de esto que si $m$ es primo, el período completo ocurre sólo con $a = 1$.

Los números generados por la recurrencia anterior, se pueden expresar en forma cerrada:

$$x_n \equiv a^n x_+ b\frac{a^n - 1}{a - 1}(mod\, m)$$

**Algortimo de encriptación RSA**

En el sistema de encriptación conocido por la sigla RSA asigna una clave pública a cada sujeto, conformada por dos enteros que llamaremos $n$ y $e$; y una clave privada, un entero $d$. La clave pública es obviamente conocida por todos los interesados, y se emplea para encriptar los mensajes. La clave privada sólo es conocida por el sujeto que la posee, y es necesaria para desencriptar los mensajes recibidos. La clave $n$ es producto de dos números primos $p$ y $q$ (éstos deben ser grandes, para generar un producto $n$ del orden de entre 1024-2048 bits de longitud). La clave pública $e$ se elige como un entero coprimo con $(p - 1)(q - 1)$. La clave privada $d$ es el inverso de $e$ módulo $(p - 1)(q - 1)$.

¿Cómo funciona el algoritmo de encriptación y desencriptación?

Sea $M$ el mensaje original (traducido a entero). El mensaje encriptado $C$ se obtiene de

$$C \equiv M^e\,(mod\, n)$$

usando la clave pública $(e, n)$ del destinatario.

El receptor, al recibir el mensaje encriptado $C$, lo descifra mediante

$$M \equiv C^d\,(mod\, n)$$

donde $d$ es la clave privada del destinatario.

Cualquier receptor del mensaje codificado $C$, para desencriptarlo, necesita conocer la clave privada $d$. Todos saben que $d$ es el inverso de la clave pública $e$, pero para calcular $d$ se necesita conocer $p$ y $q$, que son los factores primos de $n$. No existe un algoritmo conocido que resuelva la factorización de un entero grande $n$ en un tiempo razonable, usando computadoras tradicionales.

¿Por qué funciona el algoritmo de desencriptación?

Al ser $d$ inverso de $e$, se tiene: $e \cdot d \equiv 1\,(mod\,(p-1)(q-1))$, o: $e \cdot d + (p-1)(q-1)y = 1$, con algún entero $y$.

Luego,

$$C^d = (M^e)^d = M^{e \cdot d} = M^{1+(p-1)(q-1)y} = M \cdot M^{(p-1)(q-1)y}$$

Por un resultado, conocido como Pequeño Teorema de Fermat, si $M$ no es múltiplo de $p$ ni de $q$ (lo que ocurre en la mayoría de los casos), sucede que $M^{p-1} \equiv 1 \ (mod \ p)$ y $M^{q-1} \equiv 1 \ (mod \, q)$.

Por lo tanto,

$$C^d = M \cdot (M^{p-1})^{(q-1)y} \equiv M \ (mod \, p)$$

$$C^d = M \cdot (M^{q-1})^{(p-1)y} \equiv M \ (mod \, q)$$

Y por el Teorema Chino del Resto (subsección 3.7.2), $M$ es la única solución, módulo $n$, del sistema de congruencias

$$C^d \equiv M \ (mod \, p)$$

$$C^d \equiv M \ (mod \, q)$$

Esas dos congruencias son equivalentes a (por ser $p$ y $q$ coprimos):

$$C^d \equiv M \ (mod \, n)$$

***Ejemplo* 3.36**

*Veamos un ejemplo (con enteros pequeños) de cómo funciona el algoritmo RSA.*

*Se quiere transmitir el mensaje GO ON, encriptándolo con el sistema RSA. El receptor tiene clave pública* $n = 77$ *(que es producto de dos primos,* $p = 7$ *y* $q = 11$*).*

*Así,* $(p-1)(q-1) = 6 \cdot 10 = 60$*. La clave pública e se toma como* $e = 13$ *(coprimo con 60).*

*Para enviar el mensaje, se debe traducir a enteros. Una forma de hacerlo es cambiando cada letra por su número de posición en el alfabeto. Entonces el mensaje es* $M =$ 07 16 16 14*. Luego se encripta, haciendo las operaciones:*

$$C_1 \equiv 07^{13} \ (mod \ 77)$$
$$C_2 \equiv 16^{13} \ (mod \ 77)$$
$$C_3 \equiv 16^{13} \ (mod \ 77)$$
$$C_4 \equiv 14^{13} \ (mod \ 77)$$

*Se obtiene (los detalles del cálculo se dejan al lector):*

$$C_1 = 7^{13} \equiv 35 \ (mod \, 77)$$

$$C_2 = 16^{13} \equiv 37 \ (mod \, 77)$$

$$C_3 = 16^{13} \equiv 37 \ (mod \, 77)$$

$$C_4 = 14^{13} \equiv 49 \ (mod \, 77)$$

*Entonces, el mensaje encriptado es* $C = 35 \ 37 \ 37 \, 49$*.*

*Para desencriptarlo se necesita conocer la clave secreta d, inverso de 13 módulo 60. Debe satisfacer:*

$$13\,d \equiv 1\ (mod\,60)$$

*En este caso,* $d = 37$.
*La recuperación del mensaje se obtiene de:*

$$M_1 \equiv 35^{37} \equiv 7\,(mod\ 77)$$
$$M_2 \equiv 37^{37} \equiv 16\,(mod\ 77)$$
$$M_3 \equiv 37^{37} \equiv 16\,(mod\ 77)$$
$$M_4 \equiv 49^{37} \equiv 14\,(mod\ 77)$$

*Con lo que se reconstruye correctamente el mensaje original:* $M = M_1\,M_2\,M_3\,M_4 = 7\,16\,16\,14$. ■

## 3.7. Ecuaciones en congruencias

### 3.7.1. Ecuaciones lineales

Una ecuación de la forma

$$a \cdot x \equiv b\ (mod\ m)$$

donde $a$ y $b$ son enteros, y $m$ es un entero positivo, es una ECUACIÓN LINEAL EN CONGRUENCIA. Resolver la ecuación en congruencia consiste en hallar los números enteros $x$ que hacen verdadera la congruencia.

***Ejemplo* 3.37**
*La ecuación en congruencia* $5x \equiv 2\ (mod\ 3)$ *se verifica para los valores* $x = -5,\ -2,\ 1,\ 4$, *etc. Este conjunto de enteros es la clase de restos* $[1]$ *en* $Z_3$.

*La ecuación* $3x \equiv 7\ (mod\ 11)$ *tiene como solución* $-5,\ 6,\ 17$, *etc., cualquier elemento en la clase de restos* $[6]$ *en* $Z_{11}$.

*La ecuación en congruencia* $33x \equiv 1\,(mod\,280)$ *tiene por solución los enteros en* $[17]$, *como se vio en el ejemplo 3.34.*

*La ecuación* $5x \equiv 2\ (mod\ 10)$ *no tiene solución. Nótese que* $5x$ *será siempre múltiplo de 5, y todos los múltiplos de 5 tiene resto 5 o 0 en la división por 10 (nunca tendrá resto 2).*

*Tampoco la ecuación* $6x \equiv 1\ (mod\ 33)$ *tiene solución. No puede ser que* $6x - 1$ *sea múltiplo de 33. En otras palabras, la ecuación diofántica* $6x - 1 = 33y$ *no tiene solución.* ■

Puede observarse de los ejemplos anteriores, que cuando una ecuación en congruencia $ax \equiv b\ (mod\ m)$ tiene solución, tiene infinitas soluciones, todas congruentes módulo $m$. En general, todas las soluciones serán congruentes módulo $\frac{m}{mcd(a,m)}$.

Llamaremos SOLUCIÓN PRINCIPAL a la/s solución/es $x_p$ tal que $0 \leqslant x_p < m$.

La SOLUCIÓN GENERAL es el conjunto de todas las soluciones de la ecuación en congruencia. Veremos que la solución general es el conjunto de los enteros $x = x_0 + k \cdot \frac{m}{d}$, $k \in \mathbb{Z}$, siendo $d = mcd(a, m)$ y $x_0$ cualquier solución particular.

Resolver una ecuación en congruencia $a \cdot x \equiv b \ (mod\ m)$ es hallar los $x$ tales que $a \cdot x - b$ sea múltiplo de $m$. Entonces, se deben obtener las soluciones de la ecuación diofántica $a \cdot x - m \cdot y = b \ (mod\, m)$.

A partir de los resultados teóricos acerca de la existencia y características de las soluciones de ecuaciones diofánticas, se obtiene :

**Teorema 3.14.**
*Dada la ecuación en congruencia $a \cdot x \equiv b \ (mod\, m)$, se verifica:*

- *La ecuación tiene solución si y sólo si el $mcd(a, m)$ divide a $b$.*
- *Si la ecuación tiene una solución $x_0$, tiene infinitas soluciones. La solución general es de la forma $x = x_0 + \frac{m}{d} \cdot k$, con $k$ entero, siendo $d = mcd(a, m)$.*
- *Si $d = mcd(a, m)$ divide a $b$, entonces la ecuación tiene las mismas soluciones que la ecuación $(a/d) \cdot x \equiv b/d \ (mod\ m/d)$* ♣

Veamos con unos ejemplos cómo se resuelven las ecuaciones en congruencias.

**Ecuaciones en congruencias con $mcd(a, m) = 1$.**

Si el $mcd(a, m) = 1$, la ecuación $a \cdot x \equiv b \ (mod\ m)$ siempre tiene solución, ya que 1 divide a todo entero $b$. En particular, $a$ tiene inverso módulo $m$. Es decir, existe un $j$ tal que $a \cdot j \equiv 1 \, (mod\, m)$.

Entonces, multiplicando por $b$ ambos miembros de la congruencia:

$$a \cdot (b \cdot j) \equiv b \, (mod\, m)$$

Esto indica que $x_0 = b \cdot j$ es una solución de la ecuación en congruencia.

La solución general es $x = x_0 + k \cdot m$, con $k$ entero. Todas las soluciones son congruentes con $x_0$ módulo $m$.

Para estos casos existe una única solución principal.

***Ejemplo* 3.38**
*Considere la ecuación $7x \equiv 1 \ (mod\ 9)$.*

*Como el $mcd(7, 9) = 1$ divide a 1, la ecuación tiene solución. Del algoritmo de Euclides obtenemos:*

$$\begin{aligned} 9 &= 1 \cdot 7 + 2 \\ 7 &= 3 \cdot 2 + 1 \end{aligned}$$

*Luego, $mcd(9, 7) = 1 = 7 - 3 \cdot 2 = 7 - 3 \cdot (9 - 1 \cdot 7) = -3 \cdot 9 + 4 \cdot 7$. Esto indica que 4 es inverso de 7 módulo 9, es decir, $7 \cdot 4 \equiv 1 \ (mod\ 9)$. Entonces la solución general es $x = 4 + 9k$, con $k$ entero.*

*La solución principal es 4. Se puede decir que la solución general es $[4]$ en $\mathbb{Z}_9$* ■

***Ejemplo* 3.39**
*Considere la ecuación $33x \equiv 38 \ (mod\ 280)$. Como $mcd(33, 280) = 1$ divide a 38, la ecuación tiene solución.*

*Busquemos primero un inverso de 33, es decir, un $j$ tal que $33j \equiv 1 \ (mod\ 280)$. En el ejemplo 3.34 obtuvimos que 17 es un inverso de 33 módulo 280. Entonces $j = 17$.*

*Si multiplicamos* $33 \cdot 17 \equiv 1\ (mod\ 280)$ *por* 38*, resulta:* $33 \cdot (17 \cdot 38) \equiv 1 \cdot 38\ (mod\ 280)$*, es decir,* $33 \cdot 646 \equiv 38\ (mod\ 280)$.

*Así, una solución particular es* $x_0 = 646$*, y la solución general es* $x = 646 + 280k$*, con* $k$ *entero. Algunas soluciones son* $-194$*,* 86*,* 366*, etc.*

*La solución principal es la solución positiva menor a* 280*. Es decir,*

$$\begin{array}{rcl} 0 \leqslant & x & < 280 \\ 0 \leqslant & 646 + 280k & < 280 \\ -646 \leqslant & 280k & < -366 \end{array}$$

*El valor buscado se obtiene con* $k = -2$*, y nos da la solución principal* $x_p = 86$*. Así, la solución general puede expresarse como* $[86]$ *en* $\mathbb{Z}_{280}$.

***Ejemplo* 3.40**

*Considere la ecuación en congruencia* $3x \equiv 7\ (mod\ 11)$*. Calculemos el inverso de* 3 *módulo* 11 *(si bien en este ejemplo es fácil obtener el inverso por inspección, mirando la congruencia* $3j \equiv 1\,(mod\,11)$*, usaremos el método algorítmico):*

$$\begin{array}{rcl} 11 & = & 3 \cdot 3 + 2 \\ 3 & = & 1 \cdot 2 + 1 \end{array}$$

$mcd(11, 37) = 1 = 3 - 1 \cdot 2 = 3 - 1 \cdot (11 - 3 \cdot 3) = -1 \cdot 11 + 4 \cdot 3$

*El inverso de* 3 *módulo* 11 *es* 4*:* $3 \cdot 4 \equiv 1\,(mod\,11)$*. Multiplicando por* 7 *ambos miembros, se llega a* $3 \cdot 28 \equiv 7\,(mod\,11)$*. Luego una solución particular es* $x_0 = 28$*, y la solución general es* $x = 28 + 11k$*, con* $k \in Z$.

*La solución principal es el valor de* $x$ *tal que:*

$$\begin{array}{rcl} 0 \leqslant & x & < 11 \\ 0 \leqslant & 28 + 11k & < 11 \\ -28 \leqslant & 11k & < -17 \end{array}$$

*Entonces,* $k = -2$ *para la solución principal,* $x_p = 28 - 2 \cdot 11 = 6$.

*Todas las soluciones están en la clase* $[6]$ *en* $\mathbb{Z}_{11}$ ■

***Ejemplo* 3.41**

*Considere la ecuación en congruencia* $81x \equiv 3\ (mod\ 256)$*. Para hallar la solución, primero calculemos el inverso de* 81 *módulo* 256*:*

$$\begin{array}{rcl} 256 & = & 3 \cdot 81 + 13 \\ 81 & = & 6 \cdot 13 + 3 \\ 13 & = & 4 \cdot 3 + 1 \end{array}$$

$mcd(256, 81) = 1 = 13 - 4 \cdot 3 = 13 - 4 \cdot (81 - 6 \cdot 13) = -4 \cdot 81 + 25 \cdot 13 = -4 \cdot 81 + 25 \cdot (256 - 3 \cdot 81) = 25 \cdot 256 - 79 \cdot 81$

*Entonces, el inverso de* 81 *es* $(-79)$*. Luego la solución es* $x = -79 \cdot 3 + 256k = -237 + 256k$*, con* $k \in Z$.

*La solución principal es el valor de* $x$ *tal que:*

$$\begin{array}{rcl} 0 \leqslant & x & < 256 \\ 0 \leqslant & -237 + 256k & < 256 \\ 237 \leqslant & 256k & < 493 \end{array}$$

*Entonces,* $k = 1$ *para la solución principal,* $x_p = -237 + 256 = 19$. ■

**Ecuaciones con** $mcd(a, m) > 1$**.**

Si $d = mcd(a, m) > 1$, la ecuación $a \cdot x \equiv b \ (mod\ m)$ tiene solución si y sólo si $d$ divide a $b$.

En tal caso, se puede resolver la ecuación equivalente $\frac{a}{d} \cdot x \equiv \frac{b}{d} \ (mod\ \frac{m}{d})$. Esta ecuación cae en el caso descrito anteriormente, ya que $mcd(\frac{a}{d}, \frac{m}{d}) = 1$, y se puede resolver como ya se explicó.

Otra forma de resolverlas es plantear la ecuación diofántica equivalente, que es $a \cdot x - b = m \cdot y$, o

$$a \cdot x - m \cdot y = b$$

y obtener los valores $x$ de las soluciones de esta ecuación.

En los casos descritos en este apartado, la solución principal no es única. Las ecuaciones tendrán $d$ soluciones principales.

***Ejemplo* 3.42**

*Resolvamos la ecuación* $8x \equiv 12 \ (mod\ 28)$*. Como* $mcd(28, 8) = 4$ *divide a* $12$*, tiene solución. Las soluciones son las mismas que las de la ecuación* $(\frac{8}{4})x \equiv \frac{12}{4} \ (mod\ \frac{28}{4})$*, es decir,* $2x \equiv 3 \ (mod\ 7)$*.*

*El inverso de* $2$ *módulo* $7$ *es* $4$*. Entonces, la solución de la ecuación en congruencia puede escribirse:* $x = 4 \cdot 3 + 7k = 12 + 7k$*.*

*Las soluciones principales de la ecuación dada son aquellas tales que*

$$\begin{array}{rcl} 0 \leqslant & x & < 28 \\ 0 \leqslant & 12 + 7k & < 28 \\ -12 \leqslant & 7k & < 16 \end{array}$$

*Entonces, las soluciones principales se obtienen con los valores de* $k = -1, 0, 1, 2$*, y son* $x_p = 5, 12, 19, 26$*.* ■

***Ejemplo* 3.43**

*Resolvamos la ecuación* $12x \equiv 60 \ (mod\ 54)$*. Como* $mcd(54, 12) = 6$ *divide a* $60$*, tiene solución. Las soluciones son las mismas que las de la ecuación* $(\frac{12}{6})x \equiv \frac{60}{6} \ (mod\ \frac{54}{6})$*, es decir,* $2x \equiv 10 \ (mod\ 9)$*.*

*El inverso de* $2$ *módulo* $9$ *es* $5$*, es decir:* $2 \cdot 5 \equiv 1 \ (mod\ 9)$*. Entonces, la solución de la ecuación puede escribirse:* $x = 5 \cdot 10 + 9k = 50 + 9k$*.*

*Las soluciones principales de la ecuación dada son aquellas tales que:*

$$\begin{array}{rcl} 0 \leqslant & x & < 54 \\ 0 \leqslant & 50 + 9k & < 54 \\ -50 \leqslant & 9k & < 4 \end{array}$$

*Entonces, las soluciones principales se obtienen con los valores de* $k = -5, -4, -3, -2, -1, 0$*, y son* $x_p = 5, 14, 23, 32, 41, 50$*.* ■

***Ejemplo* 3.44**

*Una empresa realiza control de calidad sobre su línea de producción cada 100 horas. Se enumeran los controles desde el inicio del proceso, comenzando con el control cero. Sabiendo que el control cero se realiza a las 0:00 am, se quiere determinar qué controles se realizan a las 8:00 am.*

*Esta situación se puede plantear con ecuaciones en congruencias módulo* $24$ *(horas del día). El control número* $x$ *se realiza cuando pasaron* $100x$ *horas desde el control cero.*

*Entonces, se trata de averiguar $x$ tal que* $100x \equiv 8\ (mod\ 24)$. *Como* $mcd(100, 24) = 4$ *divide a* 8, *hay solución. Ésta se encuentra resolviendo la congruencia equivalente* $(\frac{100}{4})x \equiv (\frac{8}{4})\ (mod\ \frac{24}{4})$, *esto es:* $25x \equiv 2\ (mod\ 6)$.

*Por inspección, se encuentra que* $x = 2$ *es solución. Luego, la solución general es* $x = 2 + 6k$. *Esto es, los controles 2, 8, 14, etc., se realizan a las 8:00 hs.*

*En la situación descrita es fácil ver que los controles caen únicamente a las 0:00 hs., 4:00 hs., 8:00 hs., 12:00 hs., 16:00 hs. y 20:00 hs. Si se quiere lograr que los controles caigan en todas las horas, algo hay que cambiar. Lo que se puede cambiar es la periodicidad. Supongamos que en lugar de controlar cada 100 horas, se hace cada* **a** *horas. Entonces el control $x$ cae a la hora* **b** *si* $\mathbf{a} \cdot x \equiv \mathbf{b}\ (mod\ 24)$. *Si se quiere elegir* **a** *tal que esta congruencia tenga solución para todo* **b** *entre 0 y 23, se debe pedir que* $mcd(\mathbf{a}, 24)$ *divida a todos los enteros entre 0 y 23. Esto es posible si y sólo si* $mcd(\mathbf{a}, 24) = 1$. *Entonces, para que haya controles a toda hora, éstos deben realizarse cada una cantidad de horas* **a** *que sea coprimo con 24. Por ejemplo,* **a** *puede ser 1, 5, 7, 11, 13, 17, 19, 23, 25, 29, ... 97, 101, 103, etc.*

*Tomando* $\mathrm{a} = 101$, *los controles que se realizan a la 1:00 hs son los $x$ tal que* $101x \equiv 1\ (mod\ 24)$, *esto es,* $x = 5 + 24k$, *con $k$ entero. Los controles que se realizan a las 2:00 hs. son los $x$ tal que* $101x \equiv 2\ (mod\ 24)$, *esto es,* $x = 10 + 24k$, *etc.*

### 3.7.2. Sistemas de ecuaciones lineales: Teorema chino del resto

Suponga que se quiere conocer un número entero, conociendo los restos de éste en la división por varios módulos. El siguiente teorema dice cuándo es posible conocer este número.

**Teorema 3.15.**
***Teorema chino del resto***

*Sean $m_1$, $m_2$, ..., $m_n$ enteros positivos, coprimos dos a dos. El sistema*

$$\begin{aligned} x &\equiv a_1\ (mod\ m_1) \\ x &\equiv a_2\ (mod\ m_2) \\ &... \\ x &\equiv a_n\ (mod\ m_n) \end{aligned}$$

*tiene solución única módulo* $m = m_1 \cdot m_2 \cdot ... \cdot m_n$. ♣

El teorema dice que siendo los $n$ módulos coprimos dos a dos, el sistema tiene una única solución $0 \leqslant x_0 < m$ y todas las soluciones son los elementos de la clase de restos $[x_0]$ en $Z_m$.

Daremos una forma explícita de hallar las soluciones. Sea $M_i = m/m_i$ para $i = 1, 2, ..., n$. Es decir, $M_i$ es el producto de todos los módulos, excepto $m_i$. Entonces $M_i$ y $m_i$ son coprimos, y la congruencia $M_i \cdot y \equiv 1\ (mod\ m_i)$ tiene solución. Sea $y_i$ la solución. Entonces $x = a_1 \cdot M_1 \cdot y_1 + a_2 \cdot M_2 \cdot y_2 + ... + a_n \cdot M_n \cdot y_n$ es solución del sistema de congruencias del teorema.

Veamos que efectivamente es solución de todas las congruencias. Como $M_k \equiv 0\ (mod\ m_j)$ si $j \neq k$, multiplicando esta congruencia por $a_k$ y $y_k$, resulta $a_k \cdot M_k \cdot y_k \equiv 0\ (mod\ m_j)$.

Por otro lado, $M_j \cdot y_j \equiv 1 \ (mod\ m_j)$, luego, multiplicando ésta por $a_j$, resulta $a_j \cdot M_j \cdot y_j \equiv a_j \ (mod\ m_j)$. Sumando todas esas congruencias:

$$a_1 \cdot M_1 \cdot y_1 + a_2 \cdot M_2 \cdot y_2 + ... + a_n \cdot M_n \cdot y_n \equiv a_j \ (mod\ m_j)$$

Veamos en un ejemplo cómo se aplica lo dicho.

***Ejemplo* 3.45**
*Hallemos $x$ tal que*

$$\begin{aligned} x &\equiv 2 \ (mod\ 5) \\ x &\equiv 0 \ (mod\ 2) \\ x &\equiv -1 \ (mod\ 7) \\ x &\equiv 2 \ (mod\ 3) \end{aligned}$$

*Los términos independientes (el lado derecho de las congruencias) son $a_1 = 2$, $a_2 = 0$, $a_3 = -1$, $a_4 = 2$. Los módulos de estas congruencias son $m_1 = 5$, $m_2 = 2$, $m_3 = 7$ y $m_4 = 3$, todos coprimos entre sí. Entonces el teorema asegura que existe una solución única, módulo $m = 5 \cdot 2 \cdot 7 \cdot 3 = 210$.*

*Para hallar la solución consideramos $M_1 = m/m_1 = 42$, $M_2 = m/m_2 = 105$, $M_3 = m/m_3 = 30$, y $M_4 = m/m_4 = 70$, y se resuelven independientemente las congruencias*

$$\begin{aligned} 42y_1 &\equiv 1 \ (mod\ 5) \\ 105y_2 &\equiv 1 \ (mod\ 2) \\ 30y_3 &\equiv 1 \ (mod\ 7) \\ 70y_4 &\equiv 1 \ (mod\ 3) \end{aligned}$$

*La primera de estas congruencias tiene una solución $y_1 = 3$, la segunda tiene solución $y_2 = 1$, una solución de la tercera es $y_3 = 4$ y una solución de la cuarta es $y_4 = 1$.*

*Luego, la solución del sistema de congruencias es $x = a_1 \cdot M_1 \cdot y_1 + a_2 \cdot M_2 \cdot y_2 + a_3 \cdot M_3 \cdot y_3 + a_4 \cdot M_4 \cdot y_4 = 2 \cdot 42 \cdot 3 + 0 \cdot 105 \cdot 1 - 1 \cdot 30 \cdot 4 + 2 \cdot 70 \cdot 1$, módulo 210. Es decir: $x = 252 - 120 + 140 = 272 \equiv 62 \ (mod\ 210)$.*

*Verifiquemos que ese valor de $x$ es solución de las congruencias dadas. El resto de 62 dividido 5 es 2, y el resto de 2 dividido 5 es 2; entonces se verifica la primera congruencia. El resto de 62 dividido 2 es 0, entonces se verifica la segunda congruencia. El resto de 62 dividido 7 es 6, y el resto de $-1$ dividido 7 es 6; entonces se verifica la tercera congruencia. Y el resto de 62 dividido 3 es 2, y el resto de 2 dividido 3 es 2; entonces se verifica la cuarta congruencia.* ■

***Ejemplo* 3.46**
*Si los alumnos de un curso de dividen en grupos de 3 personas, todos los grupos quedan completos (de 3 personas cada grupo). Si se dividen en grupos de 4 personas, uno de los grupos resulta con 2 alumnos. Y si se dividen en grupos de a 5, un grupo queda de 4 personas. Se quiere saber cuántos alumnos hay en ese curso.*

*El enunciado indica que el número $x$ de alumnos, cuando se lo divide por 3, queda un resto 0; si se lo divide por 4, queda resto 2 y si se lo divide por 5 tiene resto 4. Es decir, se satisfacen las congruencias:*

$$\begin{aligned} x &\equiv 0 \ (mod\ 3) \\ x &\equiv 2 \ (mod\ 4) \\ x &\equiv 4 \ (mod\ 5) \end{aligned}$$

*Esto puede resolverse, porque los tres módulos son coprimos dos a dos. Se calcula* $M = 3 \cdot 4 \cdot 5 = 60$, $M_1 = 60/3 = 20$, $M_2 = 60/4 = 15$, $M_3 = 60/5 = 12$.

*Se consideran las congruencias:*

$$\begin{aligned} 20y_1 &\equiv 1 \ (mod\ 3) \\ 15y_2 &\equiv 1 \ (mod\ 4) \\ 12y_3 &\equiv 1 \ (mod\ 5) \end{aligned}$$

*que tienen solución* $y_1 = 2$, $y_2 = 3$ *y* $y_3 = 3$.

*La solución de las conguencias que modelan el problema es* $x = 0{\cdot}20{\cdot}2{+}2{\cdot}15{\cdot}3{+}4{\cdot}12{\cdot}3 = 234 \equiv 54 \ (mod\ 60)$. *Es decir, la cantidad de alumnos en el curso puede ser* 54, *o* 114, *o* 174, *etc. En general, la cantidad de alumnos puede ser* $54 + 60k$. ■

***Ejemplo* 3.47**

*Un distribuidor importa equipos informáticos, que llegan en contenedores con 45 equipos cada uno. Envía la mercadería a los almacenes en cajas conteniendo 14 equipos cada uno, y la última caja sólo tiene 10. Sabiendo que la cantidad de equipos importados es más de 1000 y menos de 2000, se desea saber con exactitud cuántos equipos adquirió.*

*Si* $x$ *es la cantidad de equipos, debe cumplirse que* $x \equiv 0 \ (mod\ 45)$, *y* $x \equiv 10 \ (mod\ 14)$. *Como* 45 *y* 14 *son coprimos, tiene solución única módulo* $m = 45 \cdot 14 = 630$.

*Resolvemos las congruencias*

$$\begin{aligned} 14y_1 &\equiv 1 \quad (mod\ 45) \\ 45y_2 &\equiv 1 \quad (mod\ 14) \end{aligned}$$

*Las soluciones son* $y_1 = 29$ *y* $y_2 = 5$.

*Luego, la solución de las congruencias que modelan el problema es* $x = 0{\cdot}14{\cdot}29{+}10{\cdot}45{\cdot}5$, *módulo 630.*

*Así, la cantidad de equipos es alguno de los valores* $x = 2250 + 630n$, *con* $n$ *entero. Como el problema indica que* $1000 < x < 2000$, *debe ser* $1000 < 2250 + 630n < 2000$. *Luego:* $-1250/630 < n < 250/630$. *Entonces* $n$ *debe ser* $-1$, *y la cantidad de equipos importados es* $x = 2250 - 630 = 1620$.

Sea $m = m_1 \cdot m_2 \cdot ... \cdot m_n$, con todos los $m_i$ coprimos dos a dos.

El teorema chino del resto implica que todo número menor a $m$ se puede representar por sus restos en la división por los $m_i$, y esa representación la denotamos

$$x = [a_1 \ (mod\ m_1), a_2 \ (mod\ m_2), ..., a_n \ (mod\ m_n)]$$

Por ejemplo, si $m = 12 = 3{\cdot}4$, todos los números enteros de 0 a 11 se pueden representar por sus restos en la división por 3 y por 4. La representación por restos de estos números es:

| | | |
|---|---|---|
| $0 = [0 \ (mod\ 3), 0 \ (mod\ 4)]$ | $1 = [1 \ (mod\ 3), 1 \ (mod\ 4)]$ | $2 = [2 \ (mod\ 3), 2 \ (mod\ 4)]$ |
| $3 = [0 \ (mod\ 3), 3 \ (mod\ 4)]$ | $4 = [1 \ (mod\ 3), 0 \ (mod\ 4)]$ | $5 = [2 \ (mod\ 3), 1 \ (mod\ 4)]$ |
| $6 = [0 \ (mod\ 3), 2 \ (mod\ 4)]$ | $7 = [1 \ (mod\ 3), 3 \ (mod\ 4)]$ | $8 = [2 \ (mod\ 3), 0 \ (mod\ 4)]$ |
| $9 = [0 \ (mod\ 3), 1 \ (mod\ 4)]$ | $10 = [1 \ (mod\ 3), 2 \ (mod\ 4)]$ | $11 = [2 \ (mod\ 3), 3 \ (mod\ 4)]$ |

Una de las principales aplicaciones del teorema chino del resto, es en la implementación de operaciones aritméticas con números grandes realizadas por un procesador. La idea es tomar un módulo $m$ mayor que los números que se quieren operar, y factorizar $m$ en factores coprimos $m_1$, $m_2$, ..., $m_n$. Entonces, como cada número $x$ se representa en forma única por medio de sus restos, se realizan las operaciones en paralelo sobre dichos restos. Esto, además de la ventaja de poder trabajar en paralelo con los $n$ restos, permite realizar operaciones con números más grandes que lo que se puede representar.

Por ejemplo, si se tiene un hardware que sólo puede realizar aritmética entera sin signo con 4 bits, los únicos enteros que puede representar son 0, 1, ..., 15. Para hacer cálculos con números más grandes, se pueden considerar los enteros coprimos dos a dos $m_1 = 13$, $m_2 = 14$ y $m_3 = 15$, y trabajando con la representación de enteros mediante sus restos en la división por $m_1$, $m_2$ y $m_3$, se puede realizar cálculos aritméticos con enteros hasta $m = 13 \cdot 14 \cdot 15 = 2730$.

Si se quiere sumar 44 y 60, se los representa con sus restos:

$$44 = [5\ (mod\ 13), 2\ (mod\ 14), 14\ (mod\ 15)]$$

$$60 = [8\ (mod\ 13), 4\ (mod\ 14), 0\ (mod\ 15)]$$

y se suman componente a componente las representaciones, con los correspondientes módulos y se obtiene

$$44{+}60 = [5{+}8\ (mod\ 13), 2{+}4\ (mod\ 14), 14{+}0\ (mod\ 15)] = [0\ (mod\ 13), 6\ (mod\ 14), 14\ (mod\ 15)]$$

El número que tiene esta representación en restos módulo 13, módulo 14 y módulo 15 es el $x$ que satisface:

$$x \equiv 0\ (mod\ 13)$$
$$x \equiv 6\ (mod\ 14)$$
$$x \equiv 14\ (mod\ 15)$$

El valor de $x$ que satisface esas congruencias (módulo 2730) es $104 = 44 + 60$.

Si se quiere multiplicar 44 por 60, se multiplican componente a componente la representación en restos, y se obtiene

$$44{\cdot}60 = [5{\cdot}8\ (mod\ 13), 2{\cdot}4\ (mod\ 14), 14{\cdot}0\ (mod\ 15)] = [1\ (mod\ 13), 8\ (mod\ 14), 0\ (mod\ 15)]$$

que es la representación en restos de $2640 = 44 \cdot 60$.

## 3.8. Problemas

**Problema 3.1**

*Hallar la descomposición en factores primos de los números: 491891; 92400; 2541; 94017; 7!; 153. Indicar cuántos divisores tienen cada uno de esos números.*

**Problema 3.2**

*Aplicar el algoritmo de Euclides para hallar el máximo común divisor de 250 y 110 y escribirlo como combinación lineal de 250 y 110.*

**Problema 3.3**
*Aplicar el algoritmo de Euclides para hallar el máximo común divisor de 231 y 1820 y escribirlo como combinación lineal de 231 y 1820.*

**Problema 3.4**
*Hallar el máximo común divisor de 491891 y 92400; de 7! y 2541; de 94017 y 153.*

**Problema 3.5**
*Hallar el máximo común divisor de 2475 y 6615.*

**Problema 3.6**
*Hallar el máximo común divisor de 5544 y 1575.*

**Problema 3.7**
*Determinar si las siguientes ecuaciones diofánticas tienen solución. En caso de tenerla, dar la solución general*

*a.* $2x + 12y = 1$

*b.* $25x - 100y = 25$

*c.* $540x + 100y = 1201$

*d.* $33x + 17y = 1$

*e.* $87x - 14y = 3$

*f.* $2x + 12y = 2$

*g.* $-107x - 106y = 1$

*h.* $-107x - 106y = 13$

*i.* $126x + 819y = 21$

*j.* $33x + 17y = 3$

**Problema 3.8**
*Decir para qué valores de $c$ entero con $-10 \leqslant c \leqslant 10$, la ecuación diofántica $25x + 35y = c$ tiene solución.*

**Problema 3.9**
*Decir para qué valores de $c$ entero con $-10 \leqslant c \leqslant 10$, la ecuación diofántica $= c$ tiene solución.*

**Problema 3.10**
*Decir si las siguientes ecuaciones diofánticas tienen solución. En caso afirmativo, dar la solución general, y en caso negativo, justificar.*

*a.* $5544x + 1575y = 1$

*b.* $5544x - 1575y = 0$

*c.* $1575x + 5544y = 63$

*d.* $1575x - 5544y = 189$

**Problema 3.11**
*Decir si las siguientes ecuaciones diofánticas tienen solución. En caso afirmativo, dar la solución general, y en caso negativo, justificar.*

*a.* $2475x + 6615y = 5$

*b.* $2475x - 6615y = 0$

*c.* $6615x + 2475y = 45$

*d.* $6615x - 2475y = 55$

**Problema 3.12**
*Hallar todos los puntos de la recta de ecuación* $2475x+6615y = 45$ *que tengan coordenadas enteras y que estén en el primer cuadrante.*

**Problema 3.13**
*Hallar todos los puntos de la recta de ecuación* $2475x-6615y = 90$ *que tengan coordenadas enteras y que estén en el primer cuadrante.*

**Problema 3.14**
*Hallar todos los puntos de la recta de ecuación* $2475x - 6615y = 0$ *que tengan coordenadas enteras y que estén en el primer cuadrante.*

**Problema 3.15**
*Un operador de bolsa quiere comprar y/o vender acciones de dos compañías, A y B. Las acciones de cada una de esas empresas cotizan a \$55 y \$178 respectivamente. El operador quiere realizar transacciones a fin de contar finalmente con una diferencia de \$1000 a favor. ¿Es posible? ¿Es posible lograrlo vendiendo solamente? ¿Es posible lograrlo vendiendo acciones de la empresa A y comprando acciones de la empresa B?*

**Problema 3.16**
*Encontrar el inverso multiplicativo de los elementos invertibles de* $\mathbb{Z}_7$, $\mathbb{Z}_6$, $\mathbb{Z}_{11}$ *y* $\mathbb{Z}_{15}$.

**Problema 3.17**
*Hallar los divisores propios de cero en* $\mathbb{Z}_{11}$, $\mathbb{Z}_{15}$, *y en* $\mathbb{Z}_{24}$.

**Problema 3.18**
*¿Para qué* $m$ *es cierto que en* $\mathbb{Z}_m$ *no existen divisores propios de cero?*

**Problema 3.19**
*Hallar los inversos multiplicativos (si existen) de 13, 54 y 102 módulo 200.*

**Problema 3.20**
*Hallar los inversos multiplicativos (si existen) de 2, 10, 59 y 102 módulo 209.*

**Problema 3.21**
*Determinar el último dígito de* $13^{21}$.

**Problema 3.22**
*Dar la solución general de las siguientes ecuaciones en congruencias:*

*a.* $x \equiv 5 \,(mod\, 7)$

*b.* $2x \equiv 5\,(mod\,7)$

*c.* $3x \equiv 5\,(mod\,7)$

*d.* $4x \equiv 5\,(mod\,7)$

*e.* $5x \equiv 5\,(mod\,7)$

*f.* $6x \equiv 5\,(mod\,7)$

**Problema 3.23**
*Decir si las siguientes ecuaciones en congruencia tienen solución, y en caso de tenerla, hallar todas las soluciones.*

*a.* $2x + 3 \equiv 0\,(mod\,10)$

*b.* $-x + 10 \equiv 1\,(mod\,6)$

*c.* $12x \equiv 5\,(mod\,11)$

*d.* $33x \equiv 1\,(mod\,17)$

*e.* $3 \equiv 87x\,(mod\,14)$

*f.* $15x \equiv 30\,(mod\,55)$

*g.* $15x \equiv -5\,(mod\,55)$

*h.* $9x \equiv 2\,(mod\,36)$

*i.* $13x \equiv 0\,(mod\,2574)$

*j.* $107x \equiv 13\,(mod\,106)$

*k.* $106x \equiv 13\,(mod\,106)$

*l.* $x - 405 \equiv 0\,(mod\,110)$

**Problema 3.24**
*Decir si las siguientes ecuaciones en congruencia tienen solución, y en caso de tenerla, hallar todas las soluciones.*

*a.* $5544x \equiv 63\,(mod\,1575)$

*b.* $25x \equiv 25\,(mod\,100)$

*c.* $6615x \equiv 1\,(mod\,2475)$

*d.* $6615x - 45 \equiv 0\,(mod\,2475)$

**Problema 3.25**
*Dada la ecuación en congruencias* $23x + 14 \equiv 0\,(mod\,20)$*, decir cuál/es de estos conjuntos es solución (puede haber más de una respuesta):*

*a.* $x = -14 + 20n$

*b.* $\{..., -98, -84, -70, -56, ...\}$

*c.* $x = 7 - 20n$

*d.* $x = 2 - 20n$

*e.* $\{..., -38, -18, 2, 22, 42, ...\}$

*f.* $x = -98 + 20n$

*g.* $x = 2 - 10n$

**Problema 3.26**

*Plantear una ecuación en congruencias para hallar todos los múltiplos de 56 que terminen en 20. Resolverla.*

**Problema 3.27**

*Dada la ecuación en congruencias* $36x \equiv 1 \,(mod\, c)$*, hallar los valores enteros que puede tomar* $c$ *(sabiendo que* $40 \leqslant c \leqslant 50$*) para que la ecuación tenga solución.*

**Problema 3.28**

*El código ISBN (International Standard Book Number) para libros publicados hasta 2007 consiste en 10 dígitos* $x_1x_2x_3...x_{10}$*. El último dígito es en un símbolo de control (0, 1, 2, 3,... 9, X). El código debe verificar* $\sum_{i=1}^{10} i \cdot x_i \equiv 0 \,(mod\, 11)$ *(el X de la cifra de control se reemplaza por 10). Escribir un algoritmo para validar un número de ISBN.*

**Problema 3.29**

*Los primeros 9 dígitos del código ISBN del libro Matematica discreta y combinatoria de Ralph P. Grimaldi son 968444324. Hallar el dígito de control.*

**Problema 3.30**

*El código de barras es un número de 12 cifras* $x_1x_2...x_{12}$ *que identifica un producto, su fabricante y su país de origen más un dígito* $x_{13}$ *de control. Debe cumplirse que* $\sum_{i=1}^{6} x_{2i-1} + 3x_{2i} \equiv -x_{13} \,(mod\, 10)$*. Escribe un algoritmo para calcular el dígito de control, dado los primeros 12 números del código de barras de un artículo.*

**Problema 3.31**

*El código de barras de una lata de gaseosa es 54490000009x6, donde* $x$ *es un dígito que se ha borrado del envase. ¿Cuánto vale* $x$*?. ¿La solución es única?*

**Problema 3.32**

*Un sistema de codificación consiste en códigos de 6 dígitos (distintos de 0) que identifican un producto, y un séptimo dígito de verificación. Un código correcto* $abcdefg$ *debe verificar que* $a - b + 2c - 2d + e - f + 2g$ *tiene resto 1 en la división por 9. Se lee en un producto el codigo* $91327x3$*, donde* $x$ *es un dígito ilegible en la etiqueta del producto. Hallar el dígito perdido.*

**Problema 3.33**

*Un sistema de codificación consiste en códigos de 6 dígitos que identifican un producto, y un séptimo dígito de verificación. Un código correcto* $abcdefg$ *debe verificar que la suma* $a + 2b + 3c + 4d + 5e + 6f + 7g$ *tiene resto 1 en la división por 18. Se lee en un producto el código* $91023x3$*, donde* $x$ *es un dígito ilegible en la etiqueta del producto. ¿Puede ser un código válido? Si lo es, hallar el dígito perdido. Si no, explicar.*

**Problema 3.34**

*Resolver el sistema de congruencias:*

$$\begin{aligned} x &\equiv 5 \,(mod\, 7) \\ x &\equiv 0 \,(mod\, 6) \\ x &\equiv -1 \,(mod\, 5) \end{aligned}$$

**Problema 3.35**
*Hallar los enteros que tienen resto 1 al ser divididos por 12 y también tienen resto 1 al ser divididos por 7.*

**Problema 3.36**
*Hallar los enteros múltiplos de 14 que tienen resto 3 al ser divididos por 5.*

**Problema 3.37**
*Hallar los enteros $x$ tales que $x+1$ es múltiplo de 4 y $x-2$ es múltiplo de 5.*

# Capítulo 4

# Álgebra de Boole

## 4.1. Estructura del Álgebra de Boole

Muchas situaciones se pueden modelar con un sistema que toma dos valores. Por ejemplo, un interruptor que enciende o apaga un mecanismo tiene dos valores: *encendido, apagado*. Un sistema más complejo puede estar provisto de varios interruptores conectados en serie o en paralelo, que en su conjunto determinan el estado del mecanismo. Así, cada interruptor puede estar encendido o apagado, y la combinación de ellos, da el estado resultante del mecanismo: encendido o apagado.

Por otra parte, los dispositivos electrónicos reciben datos que están codificados con dos niveles de tensión: *baja* o *alta*, y el dispositivo tiene una salida que, dados los valores de las entradas, estará en un nivel de tensión alta o baja. Varios dispositivos pueden integrar un circuito, y dados los valores de los datos de entrada, el circuito tiene una salida, también codificada con dos niveles de tensión.

En lógica proposicional, cada proposición puede tomar dos valores de verdad: *verdadero - falso*. Las proposiciones pueden combinarse entre sí con operadores lógicos *y* (conjunción), *o* (disyunción), y *negación*, y la proposición combinada tendrá un valor de verdad que depende del valor de verdad de las proposiciones constituyentes.

Cuando se estudia teoría de conjuntos, dado un conjunto universal $\mathcal{U}$, y considerando el conjunto $\mathcal{P}(\mathcal{U})$ de partes de $\mathcal{U}$ (o subconjuntos de $\mathcal{U}$), se distinguen dos conjuntos especiales: el vacío y el universal $\mathcal{U}$. Y entre los conjuntos de $\mathcal{P}(\mathcal{U})$ se definen dos operaciones binarias: unión e intersección, y una operación unaria, el complemento. Y siempre la unión y la intersección de subconjuntos de $\mathcal{U}$ es un subconjunto de $\mathcal{U}$, así como el complemento de un subconjunto de $\mathcal{U}$.

En teoría de números, llamando $B$ al conjunto de divisores positivos de $n \in \mathbb{N}$, se distinguen dos valores en $B$: el 1 y el mismo $n$. Pueden definirse dos operaciones binarias en $B$, el $mcd(i, j)$ y el $mcm(i, j)$ para todo $i, j \in B$, y una operación unaria, una especie de complemento o inverso, dado por $n/j$, para cada $j \in B$. Se puede probar que estas operaciones son cerradas en $B$, es decir, el $mcd$ y el $mcm$ de dos elementos de $B$ está en $B$, y el complemento de un elemento de $B$ está en $B$.

De los ejemplos anteriores se pueden identificar las siguientes características en común: se tiene un conjunto donde se identifican dos valores (encendido-apagado, alta-baja, verdadero-

falso, vacío-universal, 1-n), y existen operaciones o combinaciones entre los elementos del conjunto. Específicamente, existen dos operaciones binarias, y una unaria, todas operaciones cerradas.

Lo que se está identificando es una estructura matemática subyacente en esas situaciones, conocida como *álgebra de Boole* .

Un ÁLGEBRA DE BOOLE es un conjunto $B$, con al menos dos elementos, denotados 0 y 1, en el que se han definido dos operaciones binarias: $+ : B \times B \to B$ (suma), y $\cdot : B \times B \to B$ (producto), y una operación unaria biyectiva: $^- : B \to B$ (complemento), que verifican las siguientes leyes, para todo $x$, $y$, $z$ de $B$:

| | |
|---|---|
| **Asociativa** | $(x + y) + z = x + (y + z)$<br>$(x \cdot y) \cdot z = x \cdot (y \cdot z)$ |
| **Conmutativa** | $x + y = y + x$<br>$x \cdot y = y \cdot x$ |
| **Distributiva** | $x \cdot (y + z) = x \cdot y + x \cdot z$<br>$x + y \cdot z = (x + y) \cdot (x + z)$ |
| **Elemento neutro** | $x + 0 = x$ (el 0 es elemento neutro en suma)<br>$x \cdot 1 = x$ (el 1 es elemento neutro en producto) |
| **Inverso** | $x + \bar{x} = 1$<br>$x \cdot \bar{x} = 0$ |

Todas las situaciones descritas al inicio de esta sección tienen la estructura de un álgebra de Boole, y verifican las leyes mencionadas.

En el conjunto de proposiciones, la operación suma se identifica con la disyunción *o*, la operación producto se identifica con la conjunción *y*, y el complemento con la negación. El valor $Verdadero$ es el 1 de esta estructura, y el valor $Falso$ es el 0. Puede probarse que en ese conjunto, con esas operaciones, se verifican las leyes enunciadas. Por ejemplo, la última ley: para toda proposición $P$, la proposición compuesta $Q = P$ *o* $(noP)$ siempre tiene valor de verdad $Verdadero$, y la proposición $R = P$ *y* $(noP)$ siempre tiene el valor de verdad $Falso$.

Asimismo, considerando el conjunto de divisores positivos de $n$, la operación suma se identifica con el $mcm$, la operación producto con el $mcd$ y la operación complemento de $j$ con $n/j$. En este caso, el neutro de la suma es 1, y el neutro del producto es $n$. Sea, por ejemplo, $n = 36$. Entonces, para cualquier divisor $j$ de 36, $j + 1 = mcm(j, 1) = j$ y $j \cdot 36 = mcd(j, 36) = j$. Además, con respecto a la propiedad de los inversos, se verifica que siendo $\bar{j} = 36/j$ el complemento de $j$, $j + \bar{j} = mcm(j, 36/j) = 36$ y $j \cdot \bar{j} = mcd(j, 36/j) = 1$. Por ejemplo, con $j = 9$ y $\bar{9} = 4$, $mcm(9, 4) = 36$ y $mcd(9, 4) = 1$.

En el caso de los subconjuntos de un conjunto universal $\mathcal{U}$, la suma se identifica con la unión, el producto con la intersección, y el complemento con la complementación de subconjuntos. Además, el vacío es el neutro para la suma (unión), ya que para cualquier subconjunto $S$ de $\mathcal{U}$, $S \bigcup \emptyset = S$, y el universal es el neutro para el producto (intersección),

ya que $S \bigcap \mathcal{U} = S$.

Es necesario comentar en este punto un detalle teórico. Por la propiedad de dualidad que se describirá más adelante, es indistinto llamar a una u otra operación *suma* o *producto*, siempre que quede claro cuál es el neutro de cada una. Por ejemplo, en el caso de los subconjuntos, se podría llamar suma a la intersección, con neutro dado por el universal, y llamar producto a la unión, siendo el neutro de este producto el vacío. Así definidas las operaciones, la estructura matemática sigue siendo un álgebra de Boole.

De las leyes dadas pueden deducirse otras propiedades, que se enuncian a continuación:

| | |
|---|---|
| **Doble complemento** | $\bar{\bar{x}} = x$ |
| **Leyes de De Morgan** | $\overline{x + y} = \bar{x} \cdot \bar{y}$ |
| | $\overline{x \cdot y} = \bar{x} + \bar{y}$ |
| **Idempotencia** | $x + x = x$ |
| | $x \cdot x = x$ |
| **Acotación** | $x + 1 = 1$ |
| | $x \cdot 0 = 0$ |
| **Absorción** | $x + x \cdot y = x$ |
| | $x \cdot (x + y) = x$ |

Por ejemplo, demostremos la propiedad de *idempotencia de la suma*:

| | |
|---|---|
| $x = x + 0$ | por ser 0 neutro de la suma |
| $= x + x \cdot \bar{x}$ | por ley de inversos |
| $= (x + x) \cdot (x + \bar{x})$ | por propiedad distributiva de suma |
| $= (x + x) \cdot 1$ | por ley de inversos |
| $= x + x$ | por ser 1 neutro del producto |

Y la propiedad de *idempotencia para el producto*:

| | |
|---|---|
| $x = x \cdot 1$ | por ser 1 neutro del producto |
| $= x \cdot (x + \bar{x})$ | por ley de inversos |
| $= x \cdot x + x \cdot \bar{x}$ | por propiedad distributiva del producto |
| $= x \cdot x + 0$ | por ley de inversos |
| $= x \cdot x$ | por ser 0 neutro de la suma |

Puede observarse que las demostraciones recién mostradas son muy similares, y usan las mismas propiedades básicas, en un caso para el producto y en el otro para la suma. Puede notarse también que todas las propiedades enunciadas (salvo la propiedad de doble complemento) aparecen de a pares, una para el producto, y otra para la suma. Y en cada par de propiedades si aparece un 0 en una, en la otra aparece un 1. Tales pares de propiedades se llaman *propiedades duales*.

Formalmente, si $P$ es una propiedad del álgebra de Boole, el DUAL DE $P$ es la propiedad que se obtiene al reemplazar en $P$ las ocurrencias de $+$ por $\cdot$, las ocurrencias de $\cdot$ por $+$,

y los 0 por 1, y los 1 por 0.

**Teorema 4.1.**
***Teorema de dualidad***

*Si $P$ es una proposición acerca del álgebra de Boole que se puede demostrar con las propiedades básicas u otras derivadas de éstas, entonces la proposición dual de $P$ también es una proposición válida en el álgebra de Boole.* ♣

Veamos ahora la demostración de las propiedades de acotación. Demostraremos las dos simultáneamente, ya que cada paso está basado en propiedades duales.

| **acotación en suma** | | **acotación en producto** |
|---|---|---|
| $x+1 = x+(x+\bar{x})$ | por propiedad de inversos | $x\cdot 0 = x\cdot(x\cdot\bar{x})$ |
| $= (x+x)+\bar{x}$ | por propiedad asociativa | $= (x\cdot x)\cdot\bar{x}$ |
| $= x+\bar{x}$ | por idempotencia | $= x\cdot\bar{x}$ |
| $= 1$ | por propiedad de inversos | $= 0$ |

Demostraremos ahora las dos leyes de De Morgan. Se trata de probar que $\bar{x}+\bar{y}$ es el complemento de $x\cdot y$; y que $\bar{x}\cdot\bar{y}$ es el complemento de $x+y$. Entonces, se probarán las dos leyes de los inversos para ambos casos, ya que el complemento es único, por ser una operación biyectiva.

| **De Morgan en suma** | | **De Morgan en producto** |
|---|---|---|
| $(x+y)+\bar{x}\cdot\bar{y} =$ | | $(x\cdot y)\cdot(\bar{x}+\bar{y}) =$ |
| $= (x+y+\bar{x})\cdot(x+y+\bar{y})$ | distributiva | $= (x\cdot y\cdot\bar{x} + x\cdot y\cdot\bar{y})$ |
| $= ((x+\bar{x})+y)\cdot(x+(y+\bar{y}))$ | conmutativa, asociativa | $= (x\cdot\bar{x})\cdot y + x\cdot(y\cdot\bar{y})$ |
| $= (1+y)\cdot(x+1)$ | propiedad de inversos | $= 0\cdot y + x\cdot 0$ |
| $= 1\cdot 1$ | propiedad de acotación | $= 0+0$ |
| $= 1$ | propiedad de neutro | $= 0$ |
| | | |
| $(x+y)\cdot\bar{x}\cdot\bar{y} =$ | | $(x\cdot y)+(\bar{x}+\bar{y}) =$ |
| $= (x\cdot\bar{x}\cdot\bar{y} + y\cdot\bar{x}\cdot\bar{y}$ | distributiva | $= (x+\bar{x}+\bar{y})\cdot(y+\bar{x}+\bar{y})$ |
| $= ((x\cdot\bar{x})\cdot y)+(x\cdot(y\cdot\bar{y}))$ | conmutativa, asociativa | $= ((x+\bar{x})+\bar{y})\cdot(\bar{x}+(y+\bar{y}))$ |
| $= (0\cdot y)+(x\cdot 0)$ | propiedad de inversos | $= (1+\bar{y})\cdot(\bar{x}+1)$ |
| $= 0+0$ | propiedad de acotación | $= 1\cdot 1$ |
| $= 0$ | propiedad de neutro | $= 1$ |

Luego, como $x+y$ y $\bar{x}\cdot\bar{y}$ satisfacen las propiedades de inversos, debe ser $\overline{x+y} = \bar{x}\cdot\bar{y}$. De igual modo, como $x\cdot y$ y $\bar{x}+\bar{y}$ satisfacen las propiedades de inversos, debe ser $\overline{x\cdot y} = \bar{x}+\bar{y}$.

Así como en producto de reales suele omitirse el signo $\cdot$, en este capítulo a partir de ahora se escribirá $xy$ para denotar $x\cdot y$.

Las operaciones booleanas tienen un orden de precedencia. El complemento precede al producto, y el producto precede a suma. Para alterar este orden, se requieren paréntesis. Por ejemplo, la expresión $xy+z$ indica que se suma el producto de $x$ por $y$ a $z$. Mientras que $x(y+z)$ denota el producto de $x$ por la suma de $y$ y $z$. En la expresión $\overline{xy}$ el complemento afecta al resultado del producto de $x$ por $y$ , mientras que en $\overline{x}y$, el complemento afecta

sólo a $x$, y la operación indicada es el producto del complemento de $x$ por $y$.

## 4.2. Funciones booleanas

Todos los ejemplos mencionados al inicio de esta sección son álgebras de Boole. Sin embargo este capítulo se centrará en el álgebra de Boole más simple, que es aquella en la que los elementos son dos: el 0 y el 1. Es decir, se considera $B = \{0, 1\}$.

En este conjunto, la operación suma se define de la siguiente manera:

$$0+0=0$$
$$0+1=1$$
$$1+0=1$$
$$1+1=1$$

La operación producto se define así:

$$0 \cdot 0=0$$
$$0 \cdot 1=0$$
$$1 \cdot 0=0$$
$$1 \cdot 1=1$$

Y el complemento, se define:

$$\bar{0} = 1$$
$$\bar{1} = 0$$

$B^n$ es el conjunto de n-uplas de elementos de $B$. Es decir, $B^n$ contiene vectores de la forma $(x_1, x_2, ..., x_n)$, tal que $x_i \in B$, para todo $i$.

Una VARIABLE BOOLEANA es una variable que toma valores en $B$. Es decir, $x$ es una variable booleana si puede tomar únicamente los valores 0 y 1.

Una FUNCIÓN BOOLEANA de orden $n$ es una función con dominio en $B^n$, que toma valores en $B$. Es decir, es una función $f : B^n \to B$. Para cada elemento $(x_1, x_2, ..., x_n)$ de $B^n$, la función toma el valor $f(x_1, x_2, ..., x_n)$ que puede ser 0 ó 1.

El dominio de una función booleana es finito, ya que en $B^n$ hay $2^n$ elementos. Entonces, una forma de describir una función booleana es dar los valores de $f$ para cada elemento del dominio. Esto puede organizarse en una tabla, como se muestra en el ejemplo siguiente.

***Ejemplo* 4.1**
*Una función de orden 2 es:*

| *x* | *y* | *f(x,y)* |
|---|---|---|
| *0* | *0* | *0* |
| *0* | *1* | *1* |
| *1* | *0* | *1* |
| *1* | *1* | *1* |

*(acá en lugar de escribir $x_1$ y $x_2$ para las variables booleanas, se usa $x$ e $y$, para facilidad de notación). Se puede ver que los valores de $f$ están dados por la suma de los valores de las dos variables, es decir, $f(x, y) = x + y$.*

*Otra función de orden 2 es:*

| $x$ | $y$ | $g(x,y)$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

*En este caso, los valores de $g$ son los opuestos de los valores de la $f$ anterior. Es decir, para cada par $(x, y)$, $g(x, y) = \overline{f(x, y)}$. Entonces, $g(x, y) = \overline{x + y}$.*

*La función $h$ dada por la tabla*

| $x$ | $y$ | $h(x,y)$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

*vale 1 si y sólo si las dos variables toman valores distintos. Esta función se la conoce como* o exclusiva, *y se denota $h(x, y) = x \oplus y$. En términos de las operaciones básicas, se puede escribir $h(x, y) = \bar{x}y + x\bar{y}$.*

*La siguiente función*

| $x$ | $y$ | $k(x,y)$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

■

*vale 1 si y sólo si las dos variables toman valores iguales. Es el complemento de la función anterior, $k(x, y) = \overline{h(x, y)} = \overline{x \oplus y}$. En términos de las operaciones básicas, se puede escribir $k(x, y) = xy + \bar{x}\bar{y}$.*

***Ejemplo* 4.2**
*Una función de orden 3 es*

| *x* | *y* | *z* | *f(x,y,z)* |
|---|---|---|---|
| *0* | *0* | *0* | *1* |
| *0* | *0* | *1* | *0* |
| *0* | *1* | *0* | *0* |
| *0* | *1* | *1* | *0* |
| *1* | *0* | *0* | *0* |
| *1* | *0* | *1* | *1* |
| *1* | *1* | *0* | *1* |
| *1* | *1* | *1* | *0* |

■

Si bien la tabla de valores de una función booleana es una buena herramienta cuando el número de variables es 2 ó 3, con más variables, la tabla adquiere tamaño demasiado grande. Por ejemplo, una función en $B^5$ requiere una tabla de 32 filas.

Afortunadamente, hay otras formas de denotar una función booleana, y es usando expresiones booleanas, que combinan las variables booleanas con las operaciones definidas en el álgebra: suma, producto, complemento. Más formalmente, las EXPRESIONES BOOLEANAS en las variables booleanas $x_1$, $x_2$, ..., $x_n$ se definen recursivamente como:

* 0,1,$x_1$, $x_2$, ..., $x_n$ son expresiones booleanas

* Si $E_1$ y $E_2$ son expresiones booleanas, entonces $\overline{E_1}$, $E_1E_2$ y $E_1 + E_2$ son expresiones booleanas.

***Ejemplo* 4.3**

*Considere la función $f(x, y, z)$ tal que para cada $(x, y, z)$, la función tome el valor de la suma del complemento de $y$ más $z$. Es decir, $f(x, y, z) = \bar{y} + z$.*

*Para conocer explícitamente los valores de $f$ en cada una de las 8 ternas booleanas, se arma la tabla de valores:*

| $x$ | $y$ | $z$ | $\bar{y}$ | $f(x, y, z) = \bar{y} + z$ |
|---|---|---|---|---|
| *0* | *0* | *0* | *1* | *1* |
| *0* | *0* | *1* | *1* | *1* |
| *0* | *1* | *0* | *0* | *0* |
| *0* | *1* | *1* | *0* | *1* |
| *1* | *0* | *0* | *1* | *1* |
| *1* | *0* | *1* | *1* | *1* |
| *1* | *1* | *0* | *0* | *0* |
| *1* | *1* | *1* | *0* | *1* |

■

***Ejemplo* 4.4**

*Una función booleana de orden 3 dada con una expresión booleana es $f(x, y, z) = x\overline{(y + z)}$.*

*Si se quiere conocer los valores de $f$ para cada elemento de su dominio, se construye la tabla de valores, siguiendo las operaciones que forman $f$.*

| $x$ | $y$ | $z$ | $y+z$ | $\overline{y+z}$ | $f(x,y,z) = x\overline{(y+z)}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

■

***Ejemplo* 4.5**
*Considere la función dada por* $f(w,x,y,z) = x(w+\bar{z}) + yz$. *Su tabla de valores es:*

| $w$ | $x$ | $y$ | $z$ | $w+\bar{z}$ | $x(w+\bar{z})$ | $yz$ | $f(w,x,y,z) = x(w+\bar{z}) + yz$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

■

***Ejemplo* 4.6**
*Una función muy sencilla de tres variables es* $f(x,y,z) = 0$. *Es la función que vale 0 siempre.*

*La función de tres variables* $f(x,y,z) = 1$ *vale 1 siempre.*

Dos funciones booleanas definidas en $B^n$ son iguales, si para cada n-upla de $B^n$, toman el mismo valor. Dos expresiones booleanas son equivalentes si representan funciones iguales. Por ejemplo, las expresiones $\overline{xy}$ y $\bar{x}+\bar{y}$ son equivalentes, así como las expresiones $(y+\bar{y})x+z$ y $x+z$.

Las mismas operaciones booleanas de suma, producto y complemento pueden aplicarse a funciones, para obtener otras funciones.

***Ejemplo* 4.7**
*Dadas las funciones $f$ y $g$ en la tabla, la última columna da los valores de una nueva función $h$ que toma el valor de la suma de los complementos de $f$ y $g$.*

| $x$ | $y$ | $z$ | $f$ | $g$ | $\bar{f}$ | $\bar{g}$ | $h(x,y,z) = \bar{f}(x,y,z) + \bar{g}(x,y,z)$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |

■

***Ejemplo* 4.8**
*Si se tienen dos funciones dadas con sus respectivas expresiones, $f(x,y) = x + \overline{(\bar{x}y)}y$, y $g(x,y) = (\bar{x}+y)x$, la función $h(x,y) = f(x,y)\overline{g}(x,y)$ se representa con la expresión booleana $h(x,y) = (x + \overline{(\bar{x}y)}y)\overline{((\bar{x}+y)x)}$.* ■

## 4.3. Simplificación de expresiones booleanas

Las expresiones booleanas para denotar una función de varias variables pueden resultar bastante complicadas. Por ejemplo, la función del ejemplo anterior, o $f(w,x,y,z) = (w\overline{(x+\bar{y}z)} + x)\overline{(z+yx)} + xy + \bar{w}$.

Para simplificar expresiones algebraicas se aplican las propiedades del álgebra de Boole. En esta sección veremos varios ejemplos que muestran cómo hacerlo.

***Ejemplo* 4.9**
*Simplifiquemos la función $f(x,y,z) = (x+\bar{z})yz + \overline{x+y}$. En la siguiente tabla se muestra la simplificación paso a paso, indicando las propiedades usadas.*

| | |
|---|---|
| $(x+\bar{z})yz + \overline{x+y}$ | |
| $xyz + \bar{z}yz + \overline{x+y}$ | *distributiva de producto* |
| $xyz + \bar{z}yz + \bar{x}\bar{y}$ | *De Morgan* |
| $xyz + y(z\bar{z}) + \bar{x}\bar{y}$ | *conmutativa y asociativa del producto* |
| $xyz + y0 + \bar{x}\bar{y}$ | *inversos en producto* |
| $xyz + 0 + \bar{x}\bar{y}$ | *acotación en producto* |
| $xyz + \bar{x}\bar{y}$ | *neutro de suma* |

■

***Ejemplo* 4.10**
*La expresión $x + y(\bar{z}+x)\bar{x}$ se puede simplificar con los siguientes pasos:*

| | |
|---|---|
| $x + y(\bar{z} + x)\bar{x}$ | |
| $x + (y\bar{z} + yx)\bar{x}$ | *distributiva del producto* |
| $x + y\bar{z}\bar{x} + yx\bar{x}$ | *distributiva del producto* |
| $x + y\bar{z}\bar{x} + y0$ | *inversos en producto* |
| $x + y\bar{z}\bar{x} + 0$ | *acotación en producto* |
| $x + y\bar{z}\bar{x}$ | *neutro en suma* |

■

***Ejemplo* 4.11**
*Dada la función* $h(x, y) = (x + \overline{(\bar{x}y)}\ y)(\overline{(\bar{x} + y)x})$*, una expresión simplificada se obtiene con los siguientes pasos:*

| | |
|---|---|
| $(x + \overline{(\bar{x}y)}\ y)(\overline{(\bar{x} + y)x})$ | |
| $(x + (\overline{\overline{x}} + \bar{y})\ y)(\overline{(\bar{x} + y)} + \bar{x})$ | *De Morgan* |
| $(x + (x + \bar{y})\ y)(\overline{\overline{x}}\bar{y} + \bar{x})$ | *doble complemento y De Morgan* |
| $(x + xy + \bar{y}y)(x\bar{y} + \bar{x})$ | *distributiva de producto y doble complemento* |
| $(x + \bar{y}y)(x\bar{y} + \bar{x})$ | *absorción* |
| $(x + 0)(x\bar{y} + \bar{x})$ | *inversos en producto* |
| $x(x\bar{y} + \bar{x})$ | *neutro en suma* |
| $xx\bar{y} + x\bar{x})$ | *distributiva de producto* |
| $x\bar{y} + 0$ | *idempotencia en producto y producto de inversos* |
| $x\bar{y}$ | *neutro en suma* |

■

***Ejemplo* 4.12**
*Simplifiquemos la expresión* $(w\overline{(x + \bar{y}z)} + x)\overline{(z + yx)} + xy + \bar{w}$.

| | |
|---|---|
| $(w\overline{(x + \bar{y}z)} + x)\overline{(z + yx)} + xy + \bar{w}$ | |
| $(w\bar{x}\overline{\overline{y}z} + x)\overline{z}\ \overline{yx} + xy + \bar{w}$ | *De Morgan* |
| $(w\bar{x}(\overline{\overline{y}} + \overline{z}) + x)\overline{z}\ (\overline{y} + \overline{x}) + xy + \bar{w}$ | *De Morgan* |
| $(w\bar{x}(y + \overline{z}) + x)\overline{z}\ (\overline{y} + \overline{x}) + xy + \bar{w}$ | *doble complemento* |
| $(w\bar{x}y + w\bar{x}\ \overline{z} + x)\ (\overline{z}\ \overline{y} + \overline{z}\ \overline{x}) + xy + \bar{w}$ | *distributiva de producto* |
| $(w\bar{x}y\overline{z}\,\overline{y} + w\bar{x}\,\overline{z}\,\overline{z}\,\overline{y} + x\overline{z}\,\overline{y} + w\bar{x}y\overline{z}\,\overline{x} + w\bar{x}\,\overline{z}\,\overline{z}\,\overline{x} + x\overline{z}\,\overline{x}) + xy + \bar{w}$ | *distributiva de producto* |
| $(w\bar{x}0\overline{z} + w\bar{x}\,\overline{z}\,\overline{y} + x\overline{z}\,\overline{y} + w\bar{x}y\overline{z} + w\bar{x}\,\overline{z} + 0\overline{z}) + xy + \bar{w}$ | *conmutativa, asociativa, idempotencia e inversos en producto* |
| $(0 + w\bar{x}\,\overline{z}\,\overline{y} + x\overline{z}\,\overline{y} + w\bar{x}y\overline{z}\ + w\bar{x}\,\overline{z} + 0) + xy + \bar{w}$ | *acotación en producto* |
| $(w\bar{x}\,\overline{z}\,\overline{y} + x\overline{z}\,\overline{y} + w\bar{x}y\overline{z}\ + w\bar{x}\,\overline{z}) + xy + \bar{w}$ | *neutro en suma* |
| $(w\bar{x}\,\overline{z}(\overline{y} + y + 1) + x\overline{z}\,\overline{y}) + xy + \bar{w}$ | *conmutativa y distributiva de producto* |
| $(w\bar{x}\,\overline{z}1 + x\overline{z}\,\overline{y}) + xy + \bar{w}$ | *acotación en suma* |
| $w\bar{x}\,\overline{z} + x\overline{z}\,\overline{y} + xy + \bar{w}$ | *neutro en producto* |

## 4.4. Formas normales

### 4.4.1. Forma normal disyuntiva

Se ha descrito cómo puede obtenerse la tabla de valores de una función dada su expresión algebraica. Ahora se describirá el proceso inverso, construir una expresión algebraica para una función dada con su tabla de valores.

Suponga que se tiene una función $f$ de tres variables que vale 1 únicamente si $x = 1, y = 1, z = 1$. Claramente, una expresión algebraica para $f$ es $f(x, y, z) = xyx$. Suponga que otra función booleana, $g$, es tal que vale 1 únicamente si $x = 1, y = 0, z = 1$. Una expresión para $g$ es $g(x, y, z) = x\bar{y}z$.

Ahora, sea $h$ una función que vale 1 cuando $x = 1, y = 1, z = 1$ o bien cuando $x = 1, y = 0, z = 1$. Entonces $h$ vale 1 cuando $f$ vale 1 o cuando $g$ vale 1. Es decir, $h = f+g$. Por lo tanto, una expresión algebraica para $h$ es $h(x, y, z) = f(x, y, z)+g(x, y, z) = xyz+x\bar{y}z$.

| $x$ | $y$ | $z$ | $f$ | $g$ | $h = f + g$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 |

***Ejemplo* 4.13**
*Dada la función $f$ cuyos valores se muestran en la siguiente tabla, se puede pensar que es suma de tres funciones, cada una de las cuales vale 1 una sola vez. Específicamente, sean $f_1$, $f_2$ y $f_3$ las funciones dadas en las últimas columnas de la tabla, de modo que $f = f_1 + f_2 + f_3$. Una expresión para $f_1$ es $f_1(x, y, z) = \bar{x}yz$; una expresión para $f_2$ es $f_2(x, y, x) = x\bar{y}z$ y una expresión para $f_3$ es $f_3(x, y, z) = xy\bar{z}$, Luego, $f$ puede escribirse $f(x, y, z) = \bar{x}yz + x\bar{y}z + xy\bar{z}$.*

| $x$ | $y$ | $z$ | $f$ | $f_1$ | $f_2$ | $f_3$ |
|---|---|---|---|---|---|---|
| *0* | *0* | *0* | *0* | *0* | *0* | *0* |
| *0* | *0* | *1* | *0* | *0* | *0* | *0* |
| *0* | *1* | *0* | *0* | *0* | *0* | *0* |
| *0* | *1* | *1* | *1* | *1* | *0* | *0* |
| *1* | *0* | *0* | *0* | *0* | *0* | *0* |
| *1* | *0* | *1* | *1* | *0* | *1* | *0* |
| *1* | *1* | *0* | *1* | *0* | *0* | *1* |
| *1* | *1* | *1* | *0* | *0* | *0* | *0* |

■

***Ejemplo* 4.14**
*Las funciones dadas en la tabla tienen como expresión booleana $f(x, y) = \bar{x}\,\bar{y}$, $g(x, y) = \bar{x}y$ y $h(x, y) = \bar{x}\,\bar{y} + \bar{x}y$ respectivamente.*

| $x$ | $y$ | $f$ | $g$ | $h$ |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |

■

En todos los ejemplos anteriores se obtuvo una expresión booleana que es suma de productos. Formalmente, la FORMA NORMAL DISYUNTIVA (FND) es suma de conjunciones fundamentales. Una CONJUNCIÓN FUNDAMENTAL es el producto de todas las variables, algunas posiblemente negadas. Cada conjunción fundamental de $n$ variables, vale 1 exactamente en una $n$-upla de variales booleanas. Por esto, también se suele llamar MINTÉRMINO a cada conjunción fundamental, ya que vale 1 la mínima cantidad de veces sin ser nula, es decir, vale 1 en un solo caso.

Entonces, cada término de una forma normal disyuntiva indica una $n$-upla donde la $f$ vale 1. Si la FND de una función booleana de $n$ variables tiene $k$ términos, $k$ debe ser menor o igual a $2^n$. Además, $f$ vale 1 en $k$ $n$-uplas, y vale 0 en $(2^n - k)$ $n$-uplas.

***Ejemplo* 4.15**

*La FND de una función de 4 variables es $f(w,x,y,z) = wxyz + w\bar{x}y\bar{z} + wx\bar{y}z$. Entonces $f$ vale 1 en 3 4-uplas, y vale 0 en 16-3=13 4-uplas. Específicamente, vale 1 cuando las variables $(w,x,y,z)$ toman los valores (1,1,1,1), (1,0,1,0), (1,1,0,1).* ■

***Ejemplo* 4.16**

*Sea $g$ una función de orden 5, cuya FND es $g(v,w,x,y,z) = \bar{v}wxyz + v\bar{w}xyz + vw\bar{x}yz + vwx\bar{y}z + vwxy\bar{z}$. La FND tiene 5 términos, indicando que la función vale 1 en 5 oportunidades, cuando las variables toman los valores (0,1,1,1,1), (1,0,1,1,1), (1,1,0,1,1), (1,1,1,0,1) y (1,1,1,1,0). En el dominio de la función hay $2^5 = 32$ 5-uplas, por lo tanto, la función vale 0 para 32-5=27 5-uplas.* ■

Considere ahora la situación de hallar la FND de una función conociendo alguna expresión booleana de ella, y sin construir la tabla. Como la FND consiste en sumas de productos de variables, habrá que manipular algebraicamente la expresión conocida de la $f$ para que queden sumas de productos de variables, aplicando las propiedades conocidas. Pero puede suceder que con ésto no queden conjunciones fundamentales, es decir, que en los productos no aparezcan todas las variables. Una forma de completar los productos es multiplicar por la suma de la variable faltante más su complemento. Es decir, si para una función de tres variables $x$, $y$, $z$, un producto resulta $xy$, multiplicar por $(z + \bar{z})$. Como esta suma es 1, y el 1 es neutro en el producto, la función no se altera. Veámoslo con un ejemplo.

***Ejemplo* 4.17**

*La FND de $f(x,y,z) = x(\bar{y} + xz)$ se obtiene haciendo:*

| | |
|---|---|
| $x(\bar{y} + xz)$ | |
| $x\bar{y} + xz$ | *distributiva del producto e idempotencia en producto* |
| $x\bar{y}(z + \bar{z}) + xz(y + \bar{y})$ | *se multiplica por suma de opuestos de variable faltante* |
| $x\bar{y}z + x\bar{y}\,\bar{z} + xyz + x\bar{y}z$ | *distributiva y conmutativa del producto* |
| $x\bar{y}z + x\bar{y}\,\bar{z} + xyz$ | *idempotencia en suma* |

*En el último paso se eliminan los términos repetidos (el primero y último término del penúltimo paso), por la propiedad de idempotencia en suma. La expresión obtenida es la FND de la función dada.* ■

***Ejemplo* 4.18**
*La FND de* $f(w,x,y,z) = \overline{wx}\,yz$ *se obtiene haciendo:*

| | |
|---|---|
| $\overline{wx}\,yz$ | |
| $(\bar{w}+\bar{x})\,yz$ | *De Morgan* |
| $\bar{w}yz + \bar{x}yz$ | *distributiva de producto* |
| $\bar{w}yz(x+\bar{x}) + \bar{x}yz(w+\bar{w})$ | *se multiplica por suma de opuestos de variable faltante* |
| $\bar{w}xyz + \bar{w}\,\bar{x}yz + w\bar{x}yz + \bar{w}\,\bar{x}yz$ | *distributiva y conmutativa* |
| $\bar{w}xyz + \bar{w}\,\bar{x}yz + w\bar{x}yz$ | *idempotencia en suma* |

*La última expresión es la FND de la función dada.* ■

En el caso de la función constantemente nula, se considera que la expresión $f(x_1, x_2, ..., x_n) = 0$ es su FND.

### 4.4.2. Forma normal conjuntiva

Considere una función booleana $f$ de 3 variables que vale 1 siempre, excepto cuando $x = 0, y = 0, z = 0$. La expresión $x + y + z$ vale 1 siempre, a menos que las tres variables valgan 0 simultáneamente. Entonces, ésta es una expresión válida para $f$, $f(x,y,z) = x + y + z$.

Considere ahora una función $g$ que vale 0 únicamente cuando $x = 0, y = 1, z = 0$. Una expresión válida para $g$ es $g(x,y,z) = x + \bar{y} + z$.

Ahora, sea $h$ una función que vale 0 cuando $x = 0, y = 0, z = 0$ o cuando $x = 0, y = 1, z = 0$. Recordando que el producto de dos expresiones booleanas vale 0 cuando alguna de las dos vale 0, podemos obtener una expresión para $h$ notando que $h = fg$, es decir, $h(x,y,z) = f(x,y,z)g(x,y,z) = (x+y+z)(x+\bar{y}+z)$.

***Ejemplo* 4.19**
*La función* $f$ *de la tabla admite la expresión* $f(x,y) = x + \bar{y}$. *Y la función* $g$ *admite la expresión* $g(x,y) = x + y$. *Notando que* $h = fg$, *se llega a que* $h(x,y) = (x+\bar{y})(x+y)$.

| $x$ | $y$ | $f$ | $g$ | $h$ |
|---|---|---|---|---|
| *0* | *0* | *1* | *0* | *0* |
| *0* | *1* | *0* | *1* | *0* |
| *1* | *0* | *1* | *1* | *1* |
| *1* | *1* | *1* | *1* | *1* |

■

***Ejemplo* 4.20**
*Las funciones dadas en la tabla admiten las expresiones* $f(x,y,z) = x + \bar{y} + \bar{z}$, $g(x,y,z) = (\bar{x} + \bar{y} + \bar{z})$. *Como puede observarse,* $h = fg$, *entonces* $h(x,y,z) = (x+\bar{y}+\bar{z})(\bar{x}+\bar{y}+\bar{z})$. *Finalmente,* $k(x,y,z) = x + y + z$, *y como* $j = kh = kfg$, $j(x,y,z) = (x+y+z)(x+\bar{y}+\bar{z})(\bar{x}+\bar{y}+\bar{z})$.

| $x$ | $y$ | $z$ | $f$ | $g$ | $h$ | $k$ | $j$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

■

En los últimos ejemplos se obtuvieron expresiones booleanas que son productos de sumas. Formalmente, la FORMA NORMAL CONJUNTIVA (FNC) es producto de disyunciones fundamentales. Una DISYUNCIÓN FUNDAMENTAL es la suma de todas las variables, algunas posiblemente negadas. Cada disyunción fundamental de $n$ variables, vale 0 exactamente en una $n$-upla de variables booleanas, y vale 1 en todas las restantes. Por esto, las disyunciones fundamentales también suelen llamarse MAXTÉRMINOS, ya que valen 1 la máxima cantidad de veces, sin ser idénticamente 1.

El producto de maxtérminos vale 0 si alguno de los factores vale 0. Entonces, cada factor de una forma normal conjuntiva indica una $n$-upla donde la $f$ vale 0. Si la FNC de una función booleana de $n$ variables tiene $k$ factores, $k$ debe ser menor o igual a $2^n$. Además, $f$ vale 0 en $k$ $n$-uplas, y vale 1 en $2^n - k$ $n$-uplas.

***Ejemplo* 4.21**
*La FNC de una función de tres variables es* $f(x,y,z) = (x + y + \bar{z})(\bar{x} + \bar{y} + z)(x + \bar{y} + \bar{z})$. *Entonces f vale 0 en tres ternas, que son (0,0,1), (1,1,0), (0,1,1). Y vale 1 en las 5 ternas restantes.* ■

***Ejemplo* 4.22**
*La FNC de una función de orden 4 es* $g(w,x,y,z) = (\bar{w} + x + y + z)(w + x + \bar{y} + \bar{z})$. *Entonces g vale 0 en las 4-uplas (1,0,0,0) y (0,0,1,1); y vale 1 en los restantes 14 casos.* ■

Se plantea el problema de obtener la FNC de una función, conociendo una expresión booleana de ella, y sin construir la tabla de valores. Esto puede hacerse manipulando la expresión conocida a fin de tenerla como producto de sumas. Puede ser que se llegue a una expresión donde cada factor no sea una disyunción fundamental, es decir, no aparezcan todas las variables en la suma. Dicho factor puede completarse sumándole el producto de la variable ausente por su complemento. Como el producto de una variable por su complemento siempre es 0, sumar 0 no altera el valor de la función.

***Ejemplo* 4.23**
*Dada la función* $f(x,y,z) = (x + y)\overline{xz}$, *la FNC puede obtenerse con los siguientes pasos:*

| | |
|---|---|
| $(x + y)\overline{xz}$ | |
| $(x + y)(\bar{x} + \bar{z})$ | *De Morgan* |
| $(x + y + z\bar{z})(\bar{x} + \bar{z})$ | *suma el producto de z por su complemento* |
| $(x + y + z\bar{z})(\bar{x} + \bar{z} + y\bar{y})$ | *suma el producto de y por su complemento* |
| $(x + y + z)(x + y + \bar{z})(\bar{x} + \bar{z} + y)(\bar{x} + \bar{z} + \bar{y})$ | *distributiva de suma* |

■

***Ejemplo* 4.24**
*La FNC de la función* $f(x,y,z) = x + yz$ *se obtiene haciendo:*

| | |
|---|---|
| $x + yz$ | |
| $(x+y)(x+z)$ | *distributiva de suma* |
| $(x+y+z\bar{z})(x+z)$ | *suma el producto de z por su complemento* |
| $(x+y+z\bar{z})(x+z+y\bar{y})$ | *suma el producto de y por su complemento* ■ |
| $(x+y+z)(x+y+\bar{z})(x+y+z)(x+\bar{y}+z)$ | *distributiva y conmutativa de suma* |
| $(x+y+z)(x+y+\bar{z})(x+\bar{y}+z)$ | *idempotencia en producto* |

***Ejemplo* 4.25**
*Dada la función* $f(w,x,y,z) = (w+x+z)(x+\bar{y}+z)(\bar{w}+y)$*, la FNC se obtiene completando la disyunción de cada factor:*
$w + x + z = w + x + z + y\bar{y} = (w + x + y + z)(w + x + \bar{y} + z)$
$x + \bar{y} + z = w\bar{w} + x + \bar{y} + z = (w + x + \bar{y} + z)(\bar{w} + x + \bar{y} + z)$
$\bar{w} + y = \bar{w} + y + x\bar{x} = (\bar{w} + x + y)(\bar{w} + \bar{x} + y) = (\bar{w} + x + y + z\bar{z})(\bar{w} + \bar{x} + y + z\bar{z})$
$= (\bar{w} + x + y + z)(\bar{w} + x + y + \bar{z})(\bar{w} + \bar{x} + y + z)(\bar{w} + \bar{x} + y + \bar{z})$

*La función f es el producto de las disyunciones fundamentales obtenidas. Aplicando la ley de idempotencia, queda la FNC de f con 7 factores:*

$f(w,x,y,z) = (w + x + y + z)(w + x + \bar{y} + z)(\bar{w} + x + \bar{y} + z)(\bar{w} + x + y + z)(\bar{w} + x + y + \bar{z})(\bar{w} + \bar{x} + y + z)(\bar{w} + \bar{x} + y + \bar{z})$.

*Esta función vale 0 en 7 4-uplas (que son: (0,0,0,0), (0,0,1,0), (1,0,1,0), (1,0,0,0), (1,0,0,1), (1,1,0,0), (1,1,0,1)), y vale 1 en las restantes 9 4-uplas.* ■

En el caso de la función constantemente igual a 1, se considera que la expresión $f(x_1, x_2, ..., x_n) = 1$ es su FNC.

## 4.5. Completitud funcional

Se ha visto que toda función booleana admite una expresión booleana que involucra las operaciones de suma, producto y complemento. Se dice que el conjunto de estas tres operaciones es *funcionalmente completo*. Surge la pregunta: ¿Es posible usar un conjunto menor de operaciones, tal que con ellas se pueda representar cualquier función booleana? La respuesta es sí. Se puede demostrar que toda expresión booleana que use suma, producto y complemento se puede reescribir usando solamente producto y complemento. Es decir, el conjunto de las operaciones producto y complemento es funcionalmente completo. Para comprobar esa afirmación basta escribir la operación suma en términos de las operaciones producto y complemento.

Por una de las leyes de De Morgan, $\overline{x+y} = \bar{x}\bar{y}$. Complementando ambos miembros, y por la ley de doble complemento se llega a $x + y = \overline{\bar{x}\bar{y}}$.

Entonces, por ejemplo, la expresión $x + zy\bar{w}$ se puede reescribir usando sólo producto y complemento de la siguiente manera: $\overline{\bar{x}\ \overline{zy\bar{w}}}$.

Por otro lado, también es posible usar solamente las operaciones suma y complemento, ya que se puede reescribir el producto en términos de estas operaciones.

Por la otra ley de De Morgan, $\overline{xy} = \bar{x} + \bar{y}$. Complementando ambos miembos, y por la ley de doble complemento se llega a $xy = \overline{\bar{x} + \bar{y}}$. Luego, en cualquier expresión booleana se

puede prescindir de la operación producto usando esa igualdad. Por ejemplo, la expresión $x\bar{y}+yz$ se puede reescribir usando sólo suma y complemento: $\overline{\bar{x}+\bar{\bar{y}}}+\overline{\bar{y}+\bar{z}} = \overline{\bar{x}+y}+\overline{\bar{y}+\bar{z}}$. Y la expresión $x(\bar{z}+y)$ se reescribe $\overline{\bar{x}+\overline{\bar{z}+y}}$.

Por lo dicho, vemos que es posible representar cualquier función booleana usando sólo dos operaciones. ¿Puede representarse cualquier función con sólo una operación booleana? De nuevo la respuesta es sí, pero tal operación no es una de las básicas.

Sea la operación NAND representada con el símbolo | y definida como $x|y = \overline{xy}$.

| $x$ | $y$ | $x\|y$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Luego, toda expresión boolena se puede representar usando esta única operación. Esto es en virtud de las identidades

$$x + y = (x|x)|(y|y)$$

$$xy = (x|y)|(x|y)$$

$$\bar{x} = x|x$$

que se pueden verificar fácilmente.

Existe otra operación con la que se puede escribir toda expresión booleana. Es la operación NOR, representada con $\downarrow$ y definida como $x \downarrow y = \overline{x + y}$.

| $x$ | $y$ | $x \downarrow y$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Luego, toda expresión boolena se puede representar usando esta única operación. Esto es en virtud de las identidades

$$x + y = (x \downarrow y) \downarrow (x \downarrow y)$$

$$xy = (x \downarrow x) \downarrow (y \downarrow y)$$

$$\bar{x} = x \downarrow x$$

## 4.6. Modelado con funciones booleanas

Como se ha visto, las funciones booleanas toman un valor 0 ó 1, de acuerdo a los valores de las variables de entrada. Esto permite modelar situaciones en las que se tenga que decidir entre dos alternativas, dado un conjunto de datos.

Por ejemplo, en un tribunal de examen, tres profesores evalúan a un alumno. Cada profesor toma una decisión personal de aprobarlo o no. La decisión final se toma por mayoría: si dos o los tres profesores deciden aprobarlo, la decisión final será aprobarlo.

La decisión personal de cada profesor se puede pensar como una variable boolena de entrada. Sean $x_1$, $x_2$, $x_3$ esas variables booleanas, y $x_i = 1$ si el profesor $i$ decide aprobar al examinado, y $x_i = 0$ en otro caso. Sea $f(x_1, x_2, x_3)$ la función que decide el resultado final del examen. Es decir, $f = 1$ si y sólo si dos o más variables de entrada valen 1. Es decir, $f$ vale 1 si $x_1$ y $x_2$ valen 1 (independientemente del valor de $x_3$), o si $x_1$ y $x_3$ valen 1 (independientemente del valor de $x_2$), o si $x_2$ y $x_3$ valen 1 (independientemente del valor de $x_1$), o si $x_1$, $x_2$ y $x_3$ valen 1. Entonces, una posible expresión booleana para $f$ es $f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3 + x_1x_2x_3$. Aplicando la propiedad de absorción, la expresion queda $f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3$.

***Ejemplo* 4.26**

*Dado un dispositivo con tres entradas booleanas, se quiere determinar una función que valga 1 cuando exactamente dos entradas valgan 1. Esto no es igual a la situación anterior, ya que ahora la función debe valer 0 si las tres entradas son 1. Esta función debe valer 1 si* $x_3 = 0$ *mientras que* $x_1 = 1$ *y* $x_2 = 1$*, o si* $x_2 = 0$ *mientras que* $x_1 = 1$ *y* $x_3 = 1$*, o si* $x_1 = 0$ *mientras que* $x_2 = 1$ *y* $x_3 = 1$*.*

*Así que en este caso, la expresión booleana de la función es* $f(x_1, x_2, x_3) = x_1x_2\bar{x}_3 + x_1\bar{x}_2\,x_3 + \bar{x}_1\,x_2x_3$*.* ■

***Ejemplo* 4.27**

*Dado el número binario wxyz, se quiere una función que indique si dicho número es la expresión binaria de una potencia de 2. Es decir, la función toma 4 entradas binarias, w, x, y, y z; y debe valer 1 si wxyz es la expresión binaria de 1, 2, 4, ó 8. Analizando la expresión binaria de las potencias de 2, vemos que se caracterizan por tener exactamente un 1. Entonces, la función buscada vale 1 si una y sólo una sola entrada vale 1.*

*Entonces:* $f(w, x, y, z) = \bar{w}\,\bar{x}\,\bar{y}\,z + \bar{w}\,\bar{x}\,y\,\bar{z} + \bar{w}\,x\,\bar{y}\,\bar{z} + w\,\bar{x}\,\bar{y}\,\bar{z}$*.* ■

***Ejemplo* 4.28**

*Se tienen tres interruptores que prenden y apagan una lámpara. Cuando los tres interruptores están cerrados, la lámpara está prendida. En cualquier momento, cambiando el estado de una de las llaves, la lámpara cambia su estado (de prendida a apagada, y viceversa). Se desea determinar la función que indique el estado de la lámpara.*

*Sea x, y, z las variables que determinan la posición de los interruptores (=1 si está cerrado, =0 si está abierto). Y sea f(x,y,z) la función que indica el estado de la lámpara (=1 si está prendida, =0 si está apagada). Entonces, según el enunciado,* $f(1, 1, 1) = 1$*. Si alguna de las llaves cambia de estado, la lámpara se apaga. Es decir, para las entradas (0,1,1), (1,0,1) y (1,1,0),* $f = 0$*.*

*Luego, si cambia nuevamente un interruptor, la luz se enciende. Es decir, para las entradas (0,0,1), (0,1,0) y (1,0,0),* $f = 1$*. Finalmente, si de estos tres casos se abre el interruptor cerrado se llega a la situación (0,0,0), en que la lámpara se apaga.*

*Entonces,* $f(x, y, z) = xyz + \bar{x}\,\bar{y}z + \bar{x}\,y\bar{z} + x\bar{y}\,\bar{z}$*.* ■

***Ejemplo* 4.29**

*Una máquina expendedora de boletos recibe monedas de 10 y de 25 centavos. El boleto cuesta 20 centavos. El usuario debe colocar las monedas en tres ranuras. Dos de las ra-*

*nuras (X y Y) admiten monedas de 10 centavos, y la tercera (Z) admite monedas de 25 centavos. Obviamente alguna ranura puede quedar vacía. Una vez introducido el dinero, el usuario debe accionar el botón de entrada. Se desea determinar una función que decida si la máquina debe entregar el boleto (si la cantidad de dinero ingresado es 20 centavos o más), y otra función que indique si debe entregar vuelto (si la cantidad de dinero ingresado es más de 20 centavos).*

*El boleto debe emitirse en todos los casos, salvo que no se haya introducido monedas (0 centavos), o se haya introducido sólo una moneda de 10 centavos. Sean $x$, $y$, y $z$ las variables que indican si se ha introducido moneda en la ranura correspondiente. Sea $f$ la función que indica si se entrega el boleto, entonces $f = 0$ si las variables toman los valores (0,0,0) (ninguna moneda) o (1,0,0) o (0,1,0) (sólo una moneda de 10 centavos). Entonces, $\overline{f(x,y,z)} = \bar{x}\,\bar{y}\,\bar{z} + x\,\bar{y}\,\bar{z} + \bar{x}\,y\bar{z}$. Luego, $f(x,y,z) = \overline{\bar{x}\,\bar{y}\,\bar{z} + x\,\bar{y}\,\bar{z} + \bar{x}\,y\bar{z}}$.*

*Esta expresión puede simplificarse, usando las leyes de Boole. La simplificación resultaría: $f(x,y,z) = \overline{\bar{x}\,\bar{y}\,\bar{z}}\;\overline{x\,\bar{y}\,\bar{z}}\;\overline{\bar{x}\,y\bar{z}} = (x+y+z)(\bar{x}+y+z)(x+\bar{y}+z) = (x\bar{x}+y+z)(x+\bar{y}+z) = (y+z)(x+\bar{y}+z) = y(x+\bar{y}) + z = xy + y\bar{y} + z = xy + z$.*

*Realizando un razonamiento distinto se puede llegar a la última expresión directamente. La máquina entrega boleto si el usuario coloca una moneda de 25 centavos (z=1), independientemente de las monedas de 10 centavos ingresadas; o si introduce dos monedas de 10 centavos (x=1, y=1), independientemente de si introdujo la moneda de 25 centavos. Entonces, la máquina entrega boleto si $xy + z$ es 1.*

*La segunda función deseada es la que determina si entrega vuelto o no. La máquina entrega vuelto si el usuario introdujo 25 centavos (independientemente de si introdujo monedas de 10 centavos). Entonces, llamando $g$ a esta función, $g(x,y,z) = z$.* ■

***Ejemplo* 4.30**

*Considere que se quiere transmitir una serie $n$ de bits. Es decir, se tiene una cadena $x_1x_2...x_n$ de $n$ dígitos binarios. En la transmisión podrían ocurrir errores, como que uno de los dígitos cambie de valor. Para detectar un error de este tipo, a la cadena de 0 y 1 se le agrega un dígito de control, de modo que la cadena de $n+1$ dígitos resultante tenga una cantidad par de 1. Esto es, si la cadena original de $n$ dígitos tiene una cantidad par de 1, el dígito de control es 0, para que la paridad se mantenga en la cadena ampliada. Y si la cadena original tiene una cantidad impar de 1, el dígito de control es 1, para cambiar la paridad, y la cadena ampliada tenga una cantidad par de 1.*

*Observando el caso sencillo de $n = 2$, llamando $x_1$ y $x_2$ a los dígitos de la cadena que se desea transmitir, y $f_2$ al dígito de control, se tiene la tabla siguiente:*

| $x_1$ | $x_2$ | $f_2(x_1, x_2)$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

*Esta función ya se ha visto, y su expresión algebraica es $f_2(x_1, x_2) = x_1 \oplus x_2$.*

*El caso de $n = 3$, se puede pensar en conexión con el caso anterior. Sea $f_3$ la función que da el dígito de control para las cadenas de longitud 3. En la tabla siguiente se dan los valores correspondientes.*

| $x_1$ | $x_2$ | $f_2$ | $x_3$ | $f_3(x_1, x_2, x_3)$ |
|---|---|---|---|---|
| *0* | *0* | *0* | *0* | *0* |
| | | | *1* | *1* |
| *0* | *1* | *1* | *0* | *1* |
| | | | *1* | *0* |
| *1* | *0* | *1* | *0* | *1* |
| | | | *1* | *0* |
| *1* | *1* | *0* | *0* | *0* |
| | | | *1* | *1* |

■

*Si $f_2 = 1$, en los dos primeros dígitos hay una cantidad impar de 1, y si $x_3 = 1$, la paridad cambia y $f_3 = 0$. Si $f_2 = 1$ y $x_3 = 0$, la paridad no cambia, y el código verificador $f_3$ es 1.*

*Si $f_2 = 0$, en los dos primeros dígitos hay una cantidad par de 1, y si $x_3 = 1$ la paridad cambia y $f_3 = 1$. Si $f_2 = 0$ y $x_3 = 0$, la paridad no cambia y $f_3 = 0$.*

*Se puede observar entonces que $f_3$ es 1 si y sólo si $f_2$ y $x_3$ tienen valores distintos.*

*Luego, puede pensarse en la expresión booleana $f_3(x_1, x_2, x_3) = f_2(x_1, x_2) \oplus x_3 = (x_1 \oplus x_2) \oplus x_3$.*

*A partir de este razonamiento, es fácil generalizar la situación para cualquier $n$. Siendo $f_n$ el código de verificación para una cadena de $n$ bits, se tiene*

$$f_n(x_1, x_2, ..., x_n) = ((((x_1 \oplus x_2) \oplus x_3) \oplus ...) \oplus x_{n-1}) \oplus x_n$$

***Ejemplo* 4.31**

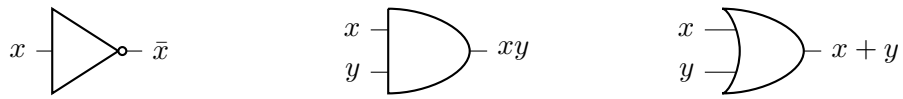
*Se quiere estudiar una función booleana que, dados dos números en binario $x_1x_2$ y $y_1y_2$ indique si el primero es mayor que el segundo. Es decir, la función debe valer 1 si $x_1x_2$ es mayor que el número $y_1y_2$, y 0 en otro caso. La función tiene cuatro variables de entrada, que son los dígitos de los números a considerar.*

*$f$ vale 1 o bien si el primer dígito del primer número es mayor que el primer dígito del segundo número, es decir: $x_1 = 1$ y $y_1 = 0$, o bien, si ambos dígitos son iguales y $x_2 = 1$ y $y_2 = 0$. Una expresión que vale 1 en el primer caso es $x_1\bar{y_1}$. Y una expresión que vale 1 en el segundo caso es $(\overline{x_1 \oplus y_1})x_2\bar{y_2}$. Recuérdese que la función $\oplus$ vale 1 si las dos variables tienen valores distintos, entonces $\overline{x_1 \oplus y_1}$ vale 1 las dos variables tienen valores iguales.*

*Finalmente, la función buscada puede escribirse $f(x_1, x_2, y_1, y_2) = x_1\bar{y_1} + (\overline{x_1 \oplus y_1})x_2\bar{y_2}$.*

## 4.7. Circuitos lógicos

Las funciones booleanas se pueden implementar en dispositivos digitales para procesar datos, a través de circuitos lógicos. Un CIRCUITO LÓGICO está formado por puertas lógicas y conexiones entre ellas. Las puertas lógicas representan las operaciones boolenas. Cada puerta tiene una o más entradas, y una salida. A través de las entradas y salidas se transfieren datos binarios 0-1. En la práctica, las entradas y salidas son cables y la información binaria que transportan se realiza con dos niveles de tensión: alto (1) o bajo (0).




**a**- NOT **b**- AND **c**- OR

Figura 4.1: Puertas lógicas

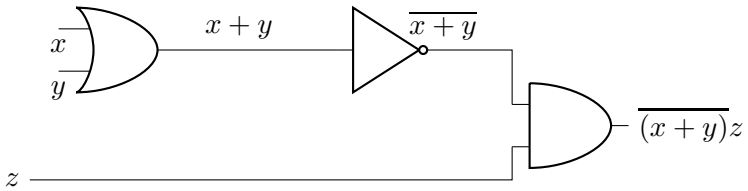



Figura 4.2: Circuito del ejemplo 4.32

En la figura 4.1 se representan las puertas básicas.

La primera se denomina INVERSOR o PUERTA NOT. Representa la operación complemento, tiene una única entrada y una salida. La salida siempre tiene el valor opuesto al de la entrada.

La PUERTA AND representa la operación producto. En principio se la considera como una puerta con dos entradas y una salida. La salida da el valor del producto de los valores de la entrada. En términos prácticos, el cable de salida tendrá un nivel de tensión alta si y sólo si las dos entradas están en nivel de tensión alto. Como el producto es una operación asociativa y conmutativa, en general, la puerta AND también puede tener múltiples entradas, para calcular el producto de múltiples variables.

La PUERTA OR representa la operación suma. Tiene dos entradas y una salida, que toma el valor de la suma de los valores de la entrada. El cable de salida estará en nivel de tensión alto si y sólo si al menos uno de los cables de entrada está en nivel de tensión alto. Como en el caso de la puerta AND, también puede considerarse la puerta OR con múltiples entradas.

En la representación gráfica de un circuito, las líneas a la izquierda de una puerta son las entradas, y la línea a la derecha es la salida.

Cualquier función booleana puede implementarse en un circuito lógico. Se darán algunos ejemplos.

***Ejemplo* 4.32**
*Sea la función* $f(x, y, z) = \overline{x + y}z$*. Su implementación en un circuito lógico puede realizarse usando una puerta OR, una puerta NOT o inversor para complementar la salida de la puerta OR, y una puerta AND, para hacer el producto de la salida del inversor por la entrada* $z$*. El circuito quedaría como se muestra en la figura 4.2.*

Dos circuitos son equivalentes si producen la misma salida para cada conjunto de valores de entrada. En otras palabras, dos circuitos son equivalentes cuando las expresiones

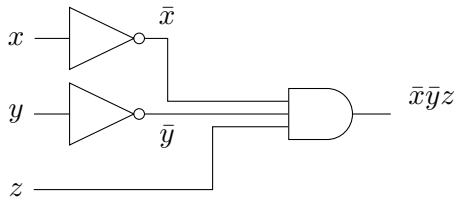



Figura 4.3: Circuito equivalente al de la figura 4.2

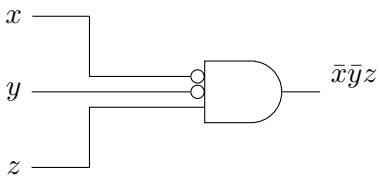



Figura 4.4: Circuito del ejemplo 4.32 usando puertas con más de dos entradas

booleanas que representan son equivalentes.

En el caso del circuito del ejemplo 4.32, nótese que una expresión equivalente para $f$ es $f(x, y, z) = \bar{x}\,\bar{y}z$. Entonces, usando una puerta AND con tres entradas, un circuito equivalente al anterior es el mostrado en la figura 4.3.

Dado que la operación inversión o complemento es unaria (una entrada, una salida), se suele usar una forma abreviada para graficarla en los circuitos. Cuando una entrada a una puerta OR o AND tiene un inversor inmediatamente antes, se denota con un círculo en la línea correspondiente a esa entrada.

Volviendo al ejemplo 4.32, usando puertas con más de dos entradas y la convención de notación para invertir las entradas, la función puede implementarse en el circuito de la figura 4.4.

***Ejemplo* 4.33**
*Consideremos ahora una función de cuatro entradas, cuya expresión booleana es* $f(w, x, y, z) = (\bar{x} + xz)(w + \overline{yz})$. *Un circuito que la implementa se muestra en la figura 4.5.*

Además de las puertas lógicas básicas, se pueden utilizar puertas que representan otras operaciones booleanas. Éstas son la PUERTA NAND (*not-and*), la PUERTA NOR (*not-or*), la PUERTA XOR (*o exclusiva*).

La puerta NAND tiene dos entradas y una salida. La salida representa la operación *producto negado*. Para las entradas $x$ y $y$, la salida es $\overline{xy}$. Esto es, el cable de salida tendrá nivel de tensión alta si y sólo si al menos una de las entradas tiene nivel de tensión bajo.

La puerta NOR también es una puerta de dos entradas y una salida, que representa la operación *suma negada*. Para las entradas $x$ y $y$, la salida es $\overline{x + y}$. Es decir, la salida tiene nivel de tensión alto si y sólo si las dos entradas tienen de nivel bajo.

Finalmente la puerta XOR tiene dos entradas y una salida que representa la operación

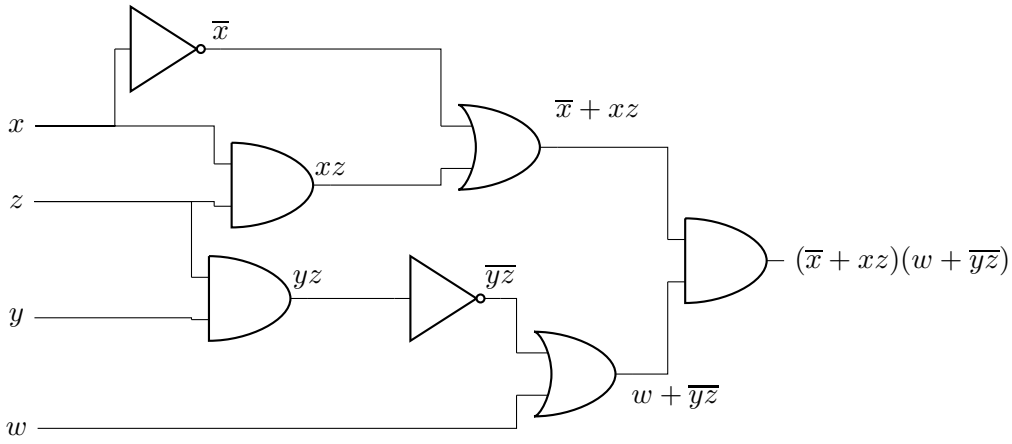
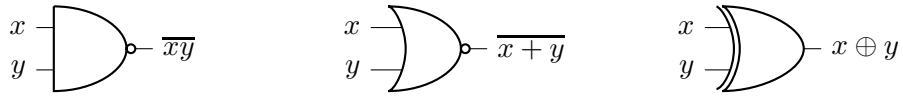



Figura 4.5: Circuito para la función del ejemplo 4.33




**a**- NAND **b**- NOR **c**- XOR

Figura 4.6: Puertas lógicas

*o exclusiva*, que es $x \oplus y = \bar{x}y + x\bar{y}$. La salida tiene nivel de tensión alto si y sólo si las entradas tienen niveles de tensión distintos. [1]

La representación gráfica de estas puertas se muestra en la figura 4.6.

***Ejemplo* 4.34**
*En el ejemplo 4.30 se obtuvo una expresión booleana para el generador del dígito de control de paridad. Usando (n-1) puertas XOR se puede implementar esta función (figura 4.7).*

***Ejemplo* 4.35**
*Dada la función booleana de la tabla, se quiere construir un circuito de puertas lógicas para implementarla.*

[1] Más generalmente, las puertas NAND, NOR y XOR admiten más de dos entradas.

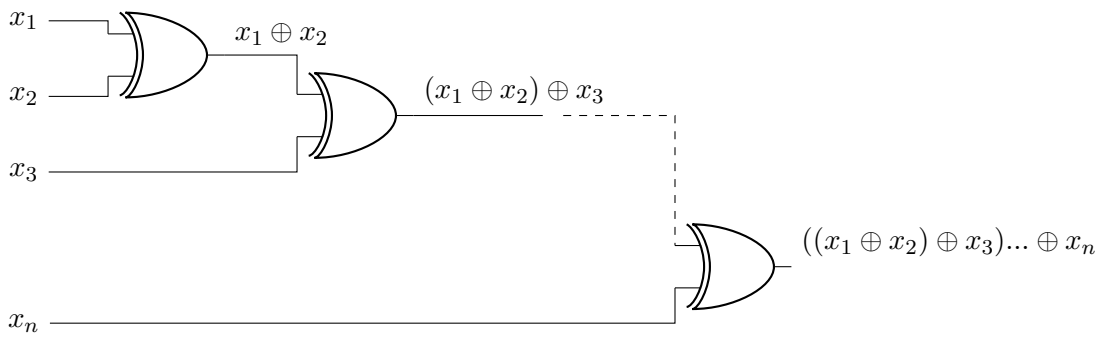



Figura 4.7: Circuito generador de código de paridad

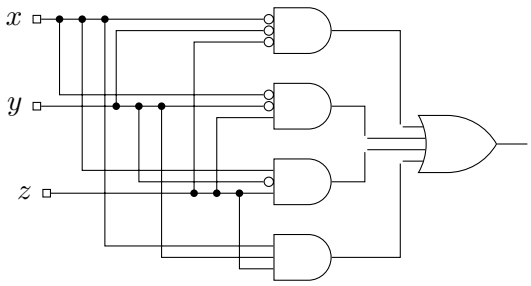



Figura 4.8: Circuito de la FND de la función del ejemplo 4.35

| $x$ | $y$ | $z$ | $f$ |
|---|---|---|---|
| *0* | *0* | *0* | *1* |
| *0* | *0* | *1* | *1* |
| *0* | *1* | *0* | *0* |
| *0* | *1* | *1* | *0* |
| *1* | *0* | *0* | *0* |
| *1* | *0* | *1* | *1* |
| *1* | *1* | *0* | *0* |
| *1* | *1* | *1* | *1* |

*Una expresión booleana para esta función es* $f(x,y,z) = \bar{x}\,\bar{y}\,\bar{z} + \bar{x}\,\bar{y}\,z + x\,\bar{y}\,z + xyz$. *Una implementación directa de esta función se muestra en la figura 4.8.*

*Los círculos negros representan bifurcaciones de las entradas, de modo que las entradas al circuito puedan ser usada como entrada a más de una puerta.*

*La FND de esta función puede simplificarse de la siguiente manera:* $f(x,y,z) = \bar{x}\,\bar{y}\,\bar{z} + \bar{x}\,\bar{y}\,z + x\,\bar{y}\,z + xyz = \bar{x}\,\bar{y}(\bar{z}+z) + xz(\bar{y}+y) = \bar{x}\,\bar{y} + xz$. *Con esta expresión se puede construir el circuito de la figura 4.9.*

*Otra expresión booleana posible para esta función es su FNC, dada por* $f(x,y,z) = (x + \bar{y} + z)(x + \bar{y} + \bar{z})(\bar{x} + y + z)(\bar{x} + \bar{y} + z)$. *De aquí se puede construir un circuito*

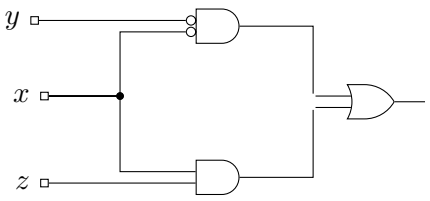
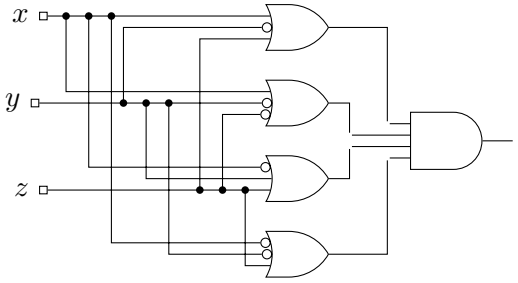



Figura 4.9: Circuito de la expresión simplificada del ejemplo 4.35




■

Figura 4.10: Circuito de la FNC de la función del ejemplo 4.35

*equivalente al anterior, con cuatro puertas OR de tres entradas y una puerta AND (figura 4.10).*

***Ejemplo* 4.36**
*Un circuito para el ejemplo de los tres interruptores (ejemplo 4.28) se muestra en la figura 4.11. Este circuito se construye directamente como una implementación de la FND de la función. Dicha expresión puede ser simplificada de la siguiente manera:* $f(x,y,z) = xyz + \bar{x}\,\bar{y}z + \bar{x}\,y\bar{z} + x\bar{y}\,\bar{z} = z(xy + \bar{x}\bar{y}) + \bar{z}(\bar{x}y + x\bar{y}) = z(\overline{x \oplus y}) + \bar{z}(x \oplus y)$*. Así, un circuito equivalente al de la figura 4.11 es el de la figura 4.12, que usa una puerta XOR, dos puertas AND y una puerta OR.* ■

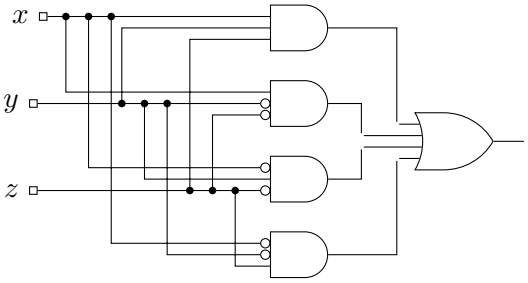



Figura 4.11: Circuito de la FND de la función del ejemplo 4.28

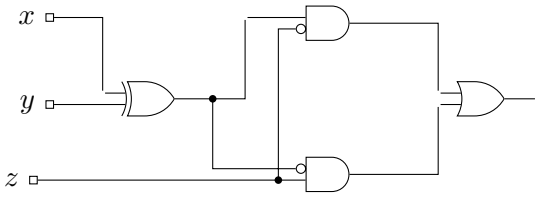



Figura 4.12: Circuito equivalente al de la figura 4.11

## 4.8. Complejidad y minimización de circuitos

Se ha visto que una función booleana puede ser implementada con diferentes circuitos, que usan distintos tipos y cantidad de puertas lógicas.

Una medida de la complejidad de un circuito es su PROFUNDIDAD. Esta característica se define por recursión. La profundidad de las entradas iniciales es 0. Para cada puerta con $n$ entradas cuyas profundidades son $p_1$, $p_2$, ..., $p_n$, la profundidad de la salida es $max(p_1, p_2, ..., p_n) + 1$, y es también la profundidad de esa puerta. La profundidad del circuito se define como el máximo de las profundidades de las puertas que lo componen. En otras palabras, la profundidad es la máxima cantidad de puertas que debe atravesar una señal de entrada para llegar a la salida.

Cada puerta por la que una señal debe pasar implica un retardo en el cálculo de la salida. Si bien en circuitos de poca profundidad este retardo es imperceptible en la práctica, en circuitos más complejos, puede ser de magnitud indeseable.

En el circuito del ejemplo 4.33, el inversor y las dos puertas AND que están más a la izquierda tienen profundidad 1. La puerta OR que está más a la izquierda y el otro inversor tienen profundidad 2. La otra puerta OR tiene profundidad 3, y la última puerta AND tiene profundidad 4. Entonces, el circuito tiene profundidad 4.

Ambos circuitos del ejemplo 4.35 tienen profundidad 2.

Como toda función booleana admite una expresión en su FND o su FNC, siempre es posible construir un circuito con profundidad 2 (como en ejemplo 4.35).

Pero otra medida de la complejidad es la cantidad total de puertas. Esto impacta en el costo de implementación del circuito. Entonces, si bien los circuitos construídos a partir de la FND o la FNC de una función tienen poca profundidad, en general resultan con muchas puertas.

Los circuitos obtenidos a partir de la forma normal de una función pueden simplificarse combinando sumandos o factores. En particular, si en una FND dos de los sumandos difieren en una sola variable, que aparece negada en uno y sin negar en el otro, se pueden combinar y obtener un solo sumando, con menos factores. Por ejemplo, si se tienen dos sumandos de la forma $xy\bar{z} + x\bar{y}\bar{z}$, eso es equivalente a $x\bar{z}(y + \bar{y}) = x\bar{z}$. Entonces los dos sumandos equivalen al único término $x\bar{z}$. Aplicando repetidas veces este procedimiento de simplificación en la FND de una función, se obtiene una expresión conocida como *suma minimal de productos*, que tiene la menor cantidad de sumandos, con la menor cantidad de factores.

***Ejemplo* 4.37**
*Dada la función cuya FND es* $f(w,x,y,z) = wx\bar{y}\bar{z} + wx\bar{y}z + wxyz + wxy\bar{z}$*, la expresión equivalente como suma minimal de productos se obtiene haciendo las simplificaciones:*

$$f(w,x,y,z) = wx\bar{y}(\bar{z}+z) + wxy(z+\bar{z}) = wx\bar{y} + wxy = wx(\bar{y}+y) = wx$$

*El circuito de puertas lógicas construído a partir de la FND tiene 4 puertas AND y una puerta OR, todas ellas de 4 entradas. En cambio, el circuito obtenido a partir de la suma minimal de productos resulta con sólo una puerta AND de 2 entradas.* ■

La clave de la simplificación es identificar pares de sumandos que difieran sólo en una variable (que aparece negada en un sumando, y sin negar en el otro). Esto puede hacerse por inspección, como en el ejemplo anterior. Sin embargo, existe un método gráfico para minimizar expresiones booleanas, conocido como DIAGRAMA DE KARNAUGH, que se aplica a funciones con no más de 6 variables (aunque la implementación para 5 y 6 variables es bastante complicada) y que organiza los términos de la FND para poder detectar fácilmente los pares de sumandos que difieren en una variable.

El diagrama de Karnaugh [2] para una función de $n$ variables es una tabla que tiene $2^n$ celdas. Cada columna y cada fila está asociada a un valor binario para determinadas variables. Luego, cada celda corresponde a un elemento de $B^n$, una $n-$upla de 0 y 1, que se obtiene de combinar los valores de las variable en la correspondiente fila y la correspondiente columna. Y a cada $n$-upla (o a cada celda) le corresponde un mintérmino. Una celda lleva un 1 si la función vale 1 en la $n$-upla correspondiente. En otras palabras, se coloca un 1 en las celdas del diagrama si el mintérmino correspondiente aparece en la FND de la función considerada.

Las celdas se ordenan de tal manera que las celdas adyacentes corresponden a mintérminos que difieren en sólo una variable.

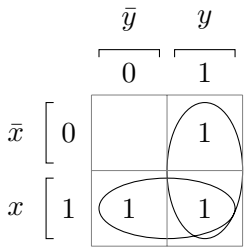
Para dos variables $x$ e $y$, el diagrama es una tabla de 4 celdas, dispuestas en un arreglo de $2 \times 2$. Cada una de las dos filas representa los posibles valores para $x$, y cada una de las dos columnas representa los posibles valores para $y$. Entonces, cada celda se corresponde con una combinación de 0 y 1 para las variables. En el diagrama que representa una determinada función, se coloca un 1 en la celda correspondiente al par binario en el que la función vale 1. Por ejemplo, el diagrama

| x \ y | 0 | 1 |
|---|---|---|
| 0 | | 1 |
| 1 | 1 | 1 |

representa la función de dos variables $x$ y $y$ que vale 1 cuando ($x$=0 y $y$=1) o cuando ($x$=1 y $y$=0) o cuando ($x$=1 y $y$=1). Es decir, representa la función $f(x,y) = \bar{x}y + x\bar{y} + xy$.

[2] También conocido como diagrama de Veitch o Veitch-Karnaugh, ya que Veitch lo propuso inicialmente, y luego Karnaugh lo modificó ligeramente y lo popularizó.

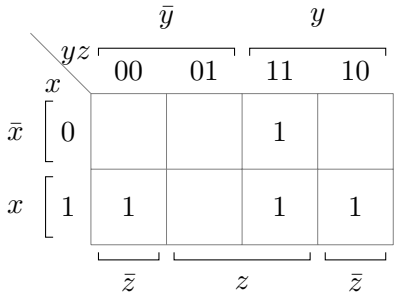
Dos celdas adyacentes en el diagrama representan términos que difieren en una sola variable. En el siguiente diagrama se han encerrado en elipses las celdas adyacentes que contienen 1. Además, se han etiquetado las filas y columnas con la variable correspondiente, negada en la fila/columna que vale 0, y sin negar donde vale 1.




Las dos celdas de la segunda fila representan las conjunciones fundamentales $x\bar{y}$ y $xy$, que difieren en la variable $y$. La suma de estas conjunciones se puede simplificar, resultando un único sumando: $x\bar{y} + xy = x(\bar{y} + y) = x$. Por otro lado, las dos celdas en la segunda columna corresponden a las conjunciones fundamentales $\bar{x}y$ y $xy$, que difieren sólo en $x$. Entonces, ambos términos se pueden simplificar y remplazarse por un único término $y$. Por lo tanto, la función correspondiente a ese diagrama es $f(x, y) = x + y$.

El diagrama para una función de tres variables $x$, $y$ y $z$ tiene 8 celdas, dispuestas en 2 filas y 4 columnas. Cada fila corresponde a un posible valor de $x$, y cada columna corresponde a una combinación de valores para $yz$, es decir, 00, 01, 10, 11. Pero se quiere que columnas adyacentes difieran sólo en una variable, entonces estos 4 valores se ordenan de la siguiente manera: 00, 01, 11, 10. En este caso, también se consideran adyacentes las columnas 1 y 4, ya que las celdas difieren en una sola variable.

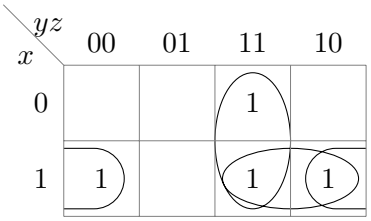
El diagrama




corresponde a la función $f(x, y, z) = \bar{x}yz + x\bar{y}\bar{z} + xyz + xy\bar{z}$. Se han etiquetado las filas con $\bar{x}$ y $x$ indicando los valores 0 y 1 de $x$ correspondientes a cada fila. Además, las dos primeras columnas corresponden a celdas donde $y$ vale 0, por eso fueron etiquetadas con $\bar{y}$. Las dos últimas columnas tienen celdas correspondientes a mintérminos que tienen $y$ sin negar. Por su parte, las columnas primera y cuarta corresponden a celdas cuyos mintérminos contienen $\bar{z}$, y así lo indican las etiquetas en la parte inferior de la tabla. Y las columnas segunda y tercera corresponde a $z$ sin negar.

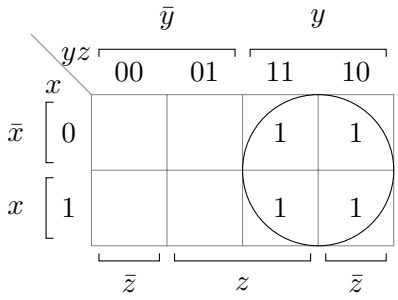
Para simplificar la FND de la función correspondiente se localizan las celdas adyacentes que tengan 1. Éstas son las dos de la tercera columna, las dos últimas celdas de la segunda

fila, y además, la primera y la última celda de la segunda fila. En el diagrama siguiente se marcan los pares de celdas adyacentes.




Entonces, para el primer par mencionado, correspondiente a las conjunciones $\bar{x}yz$ y $xyz$ (difieren en la variable $x$) se simplifica $yz$. El segundo par mencionado corresponde a conjunciones que difieren en $z$, por lo tanto se simplifica $xy$, y el último par difiere en el valor de $y$, y resulta simplificado $x\bar{z}$. Luego, la función se escribe como suma minimal de productos $f(x, y, z) = yz + xy + x\bar{z}$.

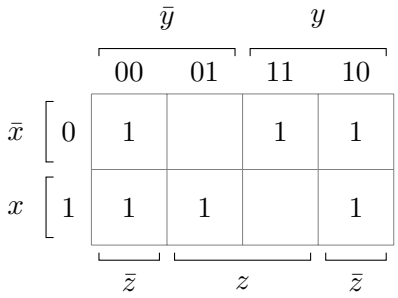
El siguiente diagrama de Karnaugh



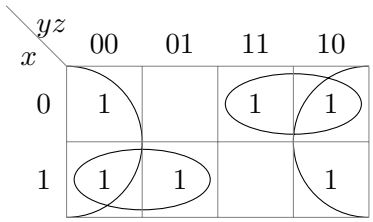

corresponde a la función $f(x, y, z) = \bar{x}yz + \bar{x}y\bar{z} + xyz + xy\bar{z}$. Esta FND se puede simplificar de la siguiente manera: $f(x, y, z) = \bar{x}y(z+\bar{z}) + xy(z+\bar{z}) = \bar{x}y + xy = y(\bar{x}+x) = y$. Entonces, la expresión booleana como suma minimal de productos es $y$. Nótese que los cuatro 1 de la tabla forman un bloque de $2 \times 2$, y corresponde a todas las conjunciones minimales en las que aparece $y$ sin negar, mientras que $x$ y $z$ aparecen en todas las combinaciones posibles (negadas y sin negar).

Esto muestra que también se pueden simplificar bloques de celdas de $2 \times 2$ que contengan 1.

En el siguiente ejemplo se muestra otra aplicación de esta simplificación por bloques.



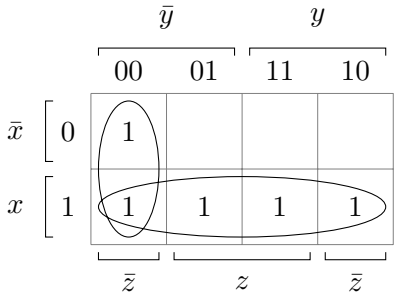

A continuación se marcan los bloques de celdas adyacentes para ese diagrama.




Se tiene un bloque de $2 \times 2$ celdas con 1, las dos de la primera columna y las dos de la cuarta columna. Esas 4 celdas corresponen a conjunciones fundamentales que continen $\bar{z}$, mientras que $x$ y $y$ aparecen en todas las combinaciones posibles. Entonces, esas 4 conjunciones se simplifica $\bar{z}$.

Pero además, hay un par de celdas adyacentes con 1, las dos primeras de la última fila, que se simplifica como $x\bar{y}$, y un segundo par de celdas adyacentes con 1, las dos últimas de la primera fila. Éstas se simplifican y resulta el producto $\bar{x}y$. Luego, la suma de productos minimales es $f(x, y, z) = \bar{z} + x\bar{y} + \bar{x}y$.

También es posible simplificar bloques de 4 celdas dispuestas en un arreglo de $1 \times 4$. Se muestra un ejemplo:




Las cuatro celdas de la segunda fila corresponden a las conjunciones $x\bar{y}\bar{z}$, $x\bar{y}z$, $xyz$ y $xy\bar{z}$. La suma de ellas se simplifica y resulta $x$.

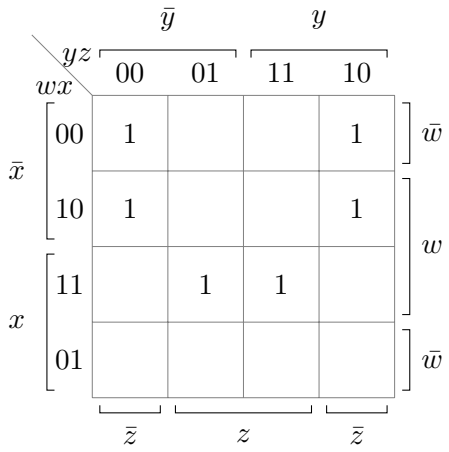
Además, en la tabla se encuentra el par de celdas adyacentes en la primera columna, correspondientes a las conjunciones $\bar{x}\bar{y}\bar{z}$ y $x\bar{y}\bar{z}$, cuya suma se simplifica $\bar{y}\bar{z}$. La expresión como suma minimal de productos es $f(x, y, z) = x + \bar{y}\bar{z}$.

Para funciones de cuatro variables $w$, $x$, $y$, $z$, se construye un diagrama de Karnaugh con 16 celdas, ubicadas en un arreglo de $4 \times 4$. Cada fila corresponde a una combinación de valores binarios para el par $wx$, y cada columna corresponde a una combinación de valores para el par $yz$, de modo que cada una de las 16 celdas corresponde a un mintérmino. Y tales combinaciones se ubican de modo que entre filas (columnas) adyacentes, difiere el valor de una sola variable. En este caso, también se consideran adyacentes la primera y la cuarta columna, y la primera y la cuarta fila.

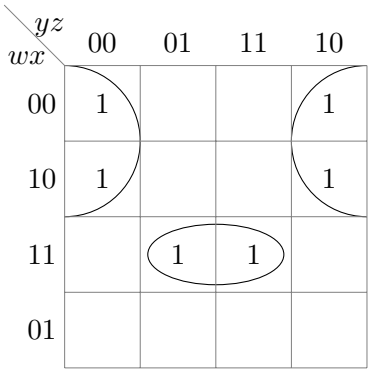
Las dos primeras columnas comparten el valor $y = 0$, se etiquetan con $\bar{y}$. Las dos últimas columnas correspondientes a $y = 1$ son etiquetadas con $y$. Las columnas primera y última se etiquetan con $\bar{z}$ (ya que esas columnas corresponden a las celdas con $z = 0$), y las columnas segunda y tercera se etiquetan con $z$.

Las dos primeras filas llevan la etiqueta $\bar{x}$, y las dos últimas filas se etiquetan con $x$. La primera y la última fila corresponden a las celdas con $w = 0$, por eso se etiquetan con $\bar{w}$, mientras que las filas segunda y tercera llevan la etiqueta $w$.

El diagrama




corresponde a la función $f(w, x, y, z) = \bar{w}\bar{x}\bar{y}\bar{z} + \bar{w}\bar{x}y\bar{z} + w\bar{x}\bar{y}\bar{z} + w\bar{x}y\bar{z} + wx\bar{y}z + wxyx$. Los bloques de celdas adyacentes se muestran en el siguiente diagrama:

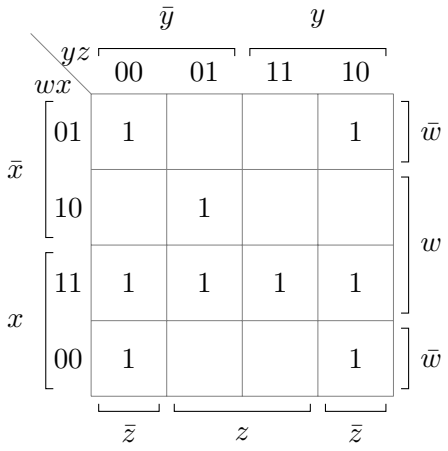



Los cuatro primeros términos corresponden a celdas adyacentes dispuestas en un bloque de $2 \times 2$. Todas las celdas de este bloque coindicen en los valores $x = 0$ y $z = 0$, mientras que las otras variables aparecen en todas las combinaciones posibles. Entonces la simplificación resultaría $\bar{x}\bar{z}$. Para verificarlo, los pasos son: $\bar{w}\bar{x}\bar{y}\bar{z} + \bar{w}\bar{x}y\bar{z} + w\bar{x}\bar{y}\bar{z} + w\bar{x}y\bar{z} = \bar{w}\bar{x}\bar{z}(\bar{y} + y) + w\bar{x}\bar{z}(\bar{y} + y) = \bar{w}\bar{x}\bar{z} + w\bar{x}\bar{z} = \bar{x}\bar{z}(\bar{w} + w) = \bar{x}\bar{z}$.

Además, el par de celdas que tienen 1 en la tercera fila comparten los valores $w = 1$, $x = 1$ y $z = 1$. Así que las dos conjunciones correspondientes se simplifican $wxz$.

Luego, la función como suma minimal de productos es $f(w, x, y, z) = \bar{x}\bar{z} + wxz$.

Otro ejemplo de 4 variables se muestra a continuación.



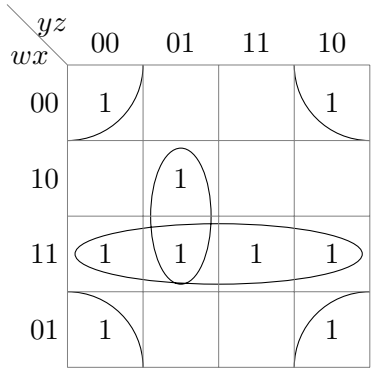

En la tercera fila hay un bloque de celdas con 1 de $1 \times 4$, que coinciden en los valores $w = 1$ y $x = 1$, y las otras variables alcanzan todas las combinaciones posibles. Por lo tanto, estos cuatro 1 se simpifican como: $wx$.

Las cuatro celdas de las esquinas son también adyacentes, y coinciden en los valores $w = 0$ y $z = 0$, mientras que las otras variables alcanzan todas las combinaciones posibles. Resulta el producto $\bar{w}\bar{z}$.

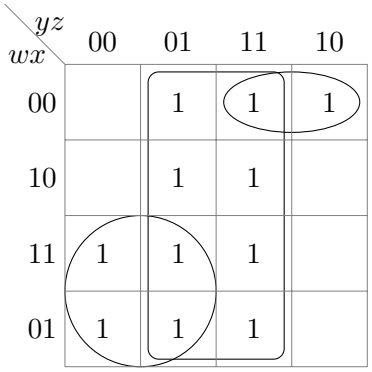
Finalmente, el par de 1 adyacentes en la segunda columna coinciden en $w = 1$, $y = 0$ y $z = 1$, mientras que $x$ toma los dos posibles valores. Entonces, esos 1 corresponden al producto $w\bar{y}z$.

La función como suma de productos minimales es $f(w, x, y, z) = wx + \bar{w}\bar{z} + w\bar{y}z$.

En el siguiente diagrama se muestran los bloques usados para minimizar la expresión.




También es posible hallar bloques de 8 celdas adyacentes, y al simplificarlas quedará un término con una sola variable (si las variables totales son 4). Por ejemplo, en el siguiente diagrama, se han marcado todos los bloques para simplificar.

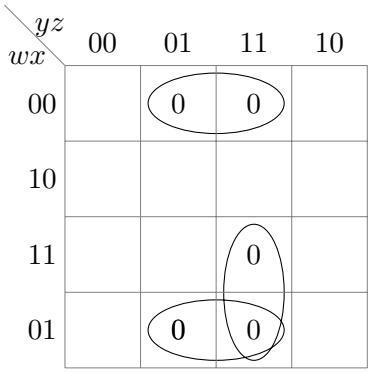



La función como suma minimal de productos es $f(w,x,y,z) = z + x\bar{y} + \bar{w}\bar{x}y$.

En el procedimiento de minimización basado en diagramas de Karnaugh, se deben identificar los bloques más grandes posibles, y cubrir todos los 1 del diagrama con la menor cantidad de bloques. Suele ocurrir que hay varias formas de cubrir los 1 con la misma cantidad de bloques. Esto indica que la suma minimal de productos no es única.

Para más de 4 variables el procedimiento se organiza en varias tablas. Se deben combinar y buscar adyacencias entre celdas de distintas tablas, además de las adyacencias del tipo de las ya vistas. No se desarrollará aquí el método para más variables, ya que es poco práctica su aplicación.

El procedimiento de minimización descrito también puede usarse para minimizar la FNC de una función, y expresarla como producto mimimal de sumas. Se comienza construyendo el diagrama, indicando en este caso las celdas donde la función vale 0 (correspondientes a los maxtérminos de la FNC). Luego se agrupan en bloques de 2, 4 u 8 celdas adyacentes con 0, que permiten simplificar los factores de la FNC. Para ejemplificar, considere la función $f(w,x,y,z) = (w+x+y+\bar{z})(w+x+\bar{y}+\bar{z})(\bar{w}+\bar{x}+\bar{y}+\bar{z})(w+\bar{x}+y+\bar{z})(w+\bar{x}+\bar{y}+\bar{z})$. En el siguiente diagrama se muestra la representación de esta función, así como los bloques que se usan para la simplificación.




Algebraicamente se puede simplificar de la siguiente manera:

$(w+x+y+\bar{z})(w+x+\bar{y}+\bar{z})(\bar{w}+\bar{x}+\bar{y}+\bar{z})$
$(w+\bar{x}+y+\bar{z})(w+\bar{x}+\bar{y}+\bar{z}) =$
$(w+x+y+\bar{z})(w+x+\bar{y}+\bar{z})(\bar{w}+\bar{x}+\bar{y}+\bar{z})$
$(w+\bar{x}+\bar{y}+\bar{z})(w+\bar{x}+y+\bar{z})(w+\bar{x}+\bar{y}+\bar{z}) =$ idempotencia y conmutativa en producto
$(w+x+y\bar{y}+\bar{z})(w\bar{w}+\bar{x}+\bar{y}+\bar{z})(w+\bar{x}+y\bar{y}+\bar{z}) =$ distributiva de suma
$(w+x+\bar{z})(\bar{x}+\bar{y}+\bar{z})(w+\bar{x}+\bar{z}) =$ ley de inversos y neutro en suma

Así, queda expresado como producto minimal de sumas.

## 4.9. Problemas

**Problema 4.1**
*Dar la tabla de valores de las siguientes funciones booleanas.*

a. $f(x,y) = x\,\bar{y}$
b. $f(x,y,z) = x + yz$
c. $f(w,x,y,z) = w\,\overline{\overline{x}}y\overline{z}$
d. $f(x,y) = xy + \overline{x}y + \overline{x}\,\overline{y} + x\overline{y}$

**Problema 4.2**
*Simplificar las siguientes expresiones booleanas, indicando qué leyes del álgebra de Boole se aplican.*

a. $f(x,y,z) = \overline{xy} + yz\overline{y} + z\overline{(x+y)}$
b. $g(w,x,y,z) = \overline{x}y(w+\overline{y}) + (z+\overline{z})x + x(\overline{x+y})$
c. $h(x,y,z) = (x+z)x + \overline{(y+\overline{z})x}$
d. $i(x,y,z) = (z+\overline{y})x + yz + z(\overline{z}+\overline{y})$
e. $j(x,y,z) = \overline{(x+y+\overline{z})} + x(\overline{y+z})$

**Problema 4.3**
*Dadas las funciones $f$ y $h$ del problema anterior, construir la tabla de valores de las funciones $f$, $h$, $fh$, $\overline{f}$ ,$hh$, $f+\overline{f}$, $h\overline{h}$ .*

**Problema 4.4**
*Hallar una expresión para las funciones booleanas de tres variables descritas por las siguientes condiciones:*

a. *vale 1 si $x = y = 0$ y $z = 0$, y vale 0 en otro caso.*
b. *vale 1 sólo cuando $x + y =1$ y $z = 0$.*
c. *vale 1 si $x$ o $y$ o $z$ es 1, y vale 0 en otro caso.*
d. *vale 0 si $x$ o $y$ son 0 mientras que $z$ vale 1; y vale 1 en otro caso.*

**Problema 4.5**
*Hallar una expresión para las funciones booleanas descritas por las siguientes condiciones:*

a. *Función de dos variables (x,y) que vale 1 si las variables toman valores distintos.*

b. *Función de 3 variables (x,y,z) que vale 1 únicamente si x e y tienen el mismo valor, y z tiene un valor distinto.*

c. *Función de 4 variables (w,x,y,z) que vale 1 sólo si las 4 variables valen 1.*

d. *Función de 4 variables (w,x,y,z) que vale 0 sólo si las 4 variables valen 1.*

e. *Función de 4 variables (w,x,y,z) que vale 1 sólo si $w + z = 1$ y $x + y$ es 0.*

f. *Función de 4 variables (w,x,y,z) que vale 1 sólo si el número binario wxyz representa un entero mayor o igual a 12.*

g. *Función de 4 variables (w, x, y, z) que vale 1 sólo si el número binario wxyz representa un entero mayor o igual a 9.*

h. *Función de 5 variables (v,w,x,y,z) que vale 0 sólo si las tres últimas no son todas 1.*

i. *Función de 5 variables (v,w,x,y,z) que vale 0 sólo si las tres primeras variables son 0.*

j. *Función de 5 variables (v,w,x,y,z) que vale 0 sólo si las cuatro primeras variables no son todas 1.*

**Problema 4.6**

*Determinar la forma normal conjuntiva y la forma normal disyuntiva de las siguientes funciones*

a. *las funciones del problema 1*

b. *las funciones del problema 2*

c. *la función booleana $F(x_1, x_2, ..., x_n)$ que vale 1 si al menos tres variables toman el valor 1.*

**Problema 4.7**

*Determinar la forma normal conjuntiva y la forma normal disyuntiva de las funciones $f$, $g$ y $h$, de orden 3, dadas en la siguiente tabla*

| $x$ | $y$ | $z$ | $f$ | $g$ | $h$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

**Problema 4.8**

*Si la función booleana $f : B^4 \to B$ se expresa en su forma normal conjuntiva (FNC) como $f(w, x, y, z) = (w + \bar{x} + y + z)(\bar{w} + x + \bar{y} + z)(w + x + \bar{y} + \bar{z})(w + x + y + z)$*

a. *¿En cuántas cuaternas $(w, x, y, z)$ la función vale 1?*

*b. ¿En cuántas cuaternas $(w, x, y, z)$ la función $ff$ vale 0?*

*c. ¿En cuántas cuaternas $(w, x, y, z)$ la función $f\overline{f}$ vale 1?*

*d. ¿Cuántos términos tiene la FND de $f$ ?*

*Justificar todas las respuestas*

**Problema 4.9**

*Construir los circuitos lógicos correspondientes a las funciones booleanas dadas en los problemas 1 y 2.*

**Problema 4.10**

*Simplifica las siguientes funciones booleanas dadas en su FND, para expresarla como suma minimal de productos.*

*a.* $f(x, y) = xy + x\overline{y} + \overline{x}y$

*b.* $f(x, y, z) = \overline{x}yz + \overline{xy}z + \overline{x}y\overline{z} + \overline{xyz}$

*c.* $f(x, y, z) = xy\overline{x} + xyz + x\overline{yz} + \overline{xyz}$

*d.* $f(w, x, y, z) = \overline{w}xyz + \overline{wx}yz + w\overline{x}yz + wxyz$

# Capítulo 5

# Autómatas, Lenguajes y Gramáticas

## 5.1. Alfabetos y lenguajes

### 5.1.1. Definiciones y ejemplos

Un ALFABETO es un conjunto no vacío finito de símbolos o caracteres. Se utiliza el símbolo $\Sigma$ para denotar un alfabeto.

Los alfabetos que más usaremos en los ejemplos son el alfabeto binario $\Sigma_1 = \{0, 1\}$, el conjunto de letras minúsculas $\Sigma_2 = \{a, b, ..., z\}$, el conjunto de todos los dígitos $\Sigma_3 = \{0, 1, 2, ..., 9\}$, $\Sigma_4$=conjunto de todos los caracteres ASCII.

Una CADENA (o palabra) es una secuencia finita de símbolos pertenecientes a un alfabeto. Por ejemplo, 1110011 es una cadena correspondiente al alfabeto $\Sigma_1$; `abracadabra` es una cadena con el alfabeto $\Sigma_2$; `hola@pepe.com` es una cadena en el alfabeto $\Sigma_4$.

Hay una cadena especial, llamada CADENA VACÍA, y denotada en este contexto con $\epsilon$, que es la que tiene cero símbolos. No es lo mismo que la cadena formada por el caracter `espacio en blanco`, ésta tiene un símbolo.

La longitud de una cadena es la cantidad de caracteres que usa. Dada una palabra $\mathbf{w}$, su longitud se denota $|\mathbf{w}|$. Por ejemplo, |1110011|=7, |abracadabra|=11; |hola@pepe.com|=13, $|\epsilon|$=0.

Siendo $\Sigma$ un alfabeto, el conjunto de cadenas de longitud $k$ con los caracteres de ese alfabeto se denota $\Sigma^k$. Por ejemplo, con el alfabeto $\Sigma = \{a, b\}$:

$\Sigma^1$= cadenas formadas por un solo caracter = $\{a, b\}$
$\Sigma^2$=cadenas formadas por dos caracteres = $\{aa, ab, ba, bb\}$
$\Sigma^3$=cadenas de tres caracteres = $\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

Para cualquier alfabeto, $\Sigma^0$ = cadenas con cero símbolos = $\{\epsilon\}$, esto es, un conjunto con una única cadena, la cadena vacía.

Como todo alfabeto es finito, el conjunto de palabras $\Sigma^k$ también es finito para cualquier $k$. Específicamente, si el alfabeto tiene $n$ símbolos, la cantidad de cadenas en $\Sigma^k$ es $n^k$.

El conjunto de todas las cadenas de cualquier longitud en un alfabeto se denota $\Sigma^*$, y se denomina CLAUSURA DE $\Sigma$. Es válida la identidad

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup ...$$

Nótese que siempre la cadena vacía está en $\Sigma^*$. Si se quieren considerar sólo las cadenas de longitud mayor a cero, se utiliza la notación $\Sigma^+$, que denota lo que se denomina CLAUSURA POSITIVA:

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup ...$$

Tanto $\Sigma^*$ como $\Sigma^+$ son conjuntos infinitos, cualquiera sea el alfabeto $\Sigma$.

Un LENGUAJE en este contexto es un conjunto de cadenas que usan símbolos de un determinado alfabeto. En términos conjuntistas, un lenguaje es un subconjunto de $\Sigma^*$. Por ejemplo, en el alfabeto $\Sigma = \{0, 1\}$, un lenguaje puede ser el conjunto de cadenas formadas por una cantidad finita de 0, seguida de una cantidad finita de 1. Es decir, $L_1 = \{0001, 000111, 01111, 0011, ...\}$. Otro lenguaje en el mismo alfabeto es el conjunto de cadenas de 0 y 1 alternados, $L_2 = \{01, 0101, 101, 010, 101010, ...\}$. Otro lenguaje que se puede definir sobre este alfabeto es el de las cadenas que tienen igual cantidad de 1 que de 0, $L_3 = \{01, 10, 0101, 1010, 0011, 1100, 110100, 110001, ...\}$.

Estos lenguajes son infinitos, pero bien podría considerarse un lenguaje finito, por ejemplo, el lenguaje de 4 palabras $L_4 = \{0, 1, 000, 110\}$.

Con el alfabeto $\Sigma = \{a, b, c, ..., z, ., ¿, ?, ¡, !, 0, 1, 2, \smile\}$ (donde $\smile$ denota el caracter `espacio en blanco`), se puede considerar el lenguaje de todas las cadenas de longitud menor a 4 que tengan sólo letras. Entonces algunas palabras de dicho lenguaje son *a*, *aba*, *cas*, *abl*, *dd*, *mme*, etc.

Con el alfabeto de los caracteres ASCII, un programa escrito en C puede considerarse como una cadena de caracteres. El conjunto de todos los programas en C forma un lenguaje.

Con el alfabeto $\Sigma = \{0, 1, 2, ..., 9, (, ), +, -, *, /\}$, se puede considerar el lenguaje de las expresiones aritméticas válidas. Así, las cadenas $32 - 58 * (23/1)$, $3 * 75 + 0$ pertenecen a este lenguaje, mientras que $/57$, $5 * (15 - 0$, $4 + *6$ no pertenecen al lenguaje.

Con el alfabeto de dos caracteres $\Sigma = \{(, )\}$, se puede definir el lenguaje de las cadenas de paréntesis balanceados. Forman parte de este lenguaje las cadenas ()(), ((())()), ()(()). No están en este lenguaje las cadenas ((), )(, ())(.

El conjunto de palabras reservadas en un lenguaje de programación es un lenguaje finito: $L = \{$begin, if, then, else, end, ..., for$\}$

Para todo alfabeto, hay tres lenguajes distintivos: el lenguaje vacío, que tiene ninguna cadena: $L = \emptyset$; el lenguaje que tiene sólo la cadena vacía: $L = \{\epsilon\}$, y el lenguaje que tiene todas las cadenas posibles: $L = \Sigma^*$.

En teoría de autómatas, que estudiaremos en este capítulo, un problema puede plantearse en términos de determinar si una cadena pertenece a cierto lenguaje o no.

La CONCATENACIÓN de dos cadenas **v** y **w** es la operación que consiste en formar una nueva palabra con una copia de **v** seguida de una copia de **w**. De esta forma se forma una palabra **vw**, de longitud igual a la suma de las longitudes de **v** y de **w**. Por ejemplo, si **v** = *pisa* y **w** = *papeles*, la concatenación de **v** y **w** da lugar a la palabra

$\mathbf{vw}$ = *pisapapeles*. Si se concatena la palabra vacía $\epsilon$ con cualquier palabra $\mathbf{w}$ resulta $\epsilon\mathbf{w} = \mathbf{w}$ y también $\mathbf{w}\epsilon = \mathbf{w}$.

Obviamente, esta operación no es conmutativa, en la concatenación es importante el orden. En el ejemplo anterior, concatenando $\mathbf{w}$ con $\mathbf{v}$ resulta la cadena $\mathbf{wv} = papelespisa \neq pisapapeles$.

La concatenación sí es asociativa. Teniendo tres cadenas, $\mathbf{u}$, $\mathbf{v}$ y $\mathbf{w}$, la concatenación de $\mathbf{u}$ con el resultado de concatenar $\mathbf{v}$ y $\mathbf{w}$ es el mismo resultado que se ontiene al concatenar la concatenación de $\mathbf{u}$ con $\mathbf{v}$, con $\mathbf{w}$. Es decir: $\mathbf{u}(\mathbf{vw}) = (\mathbf{uv})\mathbf{w} = \mathbf{uvw}$.

Dada una palabra $\mathbf{w}$ en un lenguaje, las potencias de la palabra son cadenas formadas por la concatenación de $\mathbf{w}$ con sí misma, Así, $\mathbf{w}^k = \mathbf{ww...w}$, concatenada k veces. Para ejemplificar, si $\mathbf{w} = 234$, $\mathbf{w}^2 = 234234$, $\mathbf{w}^3 = 234234234$, etc. Por convención, $\mathbf{w}^0 = \epsilon$.

Si $\mathbf{w} = 0$, entonces $\mathbf{w}^6 = 000000$. Y llamando $\mathbf{v} = 12$, $\mathbf{v}^2 = 1212$. Luego, concatenando ambas potencias: $\mathbf{w}^6\mathbf{v}^2 = 0000001212$.

Esto ofrece una herramienta para definir simbólicamente ciertos lenguajes. Se puede denotar $L = \{0^k1^h, h > k \geqslant 1\}$ al lenguaje formado por una cadena de uno o más 0, seguido de una cantidad mayor de 1. Entonces, 0011111, 0111, 0001111 son cadenas en este lenguaje, mientras que 111, 0011, 0001 no están en él.

El lenguaje $L = \{a^h *^2 b^k, h \geqslant 1, k \geqslant 0\}$ está formado por las cadenas de una o más $a$, seguida de dos asteriscos, concatenado con una cadena de cero o más b. Están en $L$ las cadenas *a***, *a**b*, *aaa**bbb*, *a**bbb*, etc.

El lenguaje $L = \{0^n1^n, n \geqslant 1\}$ está formado por cadenas de 0 concatenadas con una cadena de 1 que tenga la misma longitud que la anterior. Así, en $L$ están las cadenas 01, 0011, 000111, etc.

Dado un lenguaje $L$, la CLAUSURA de $L$, denotada $L^*$, es el lenguaje que contiene todas las palabras de $L$, la cadena vacía, y todas las palabras formadas por concatenación de palabras de $L^*$. Por ejemplo, siendo $L = \{0a, b1\}$, la clausura es $L^* = \{\epsilon, 0a, b1, 0ab1, 0a0a, 0ab10a, b1b1, b10a, b10a0a, ...\}$.

La CONCATENACIÓN de dos lenguajes es el conjunto de cadenas formadas al concatenar palabras de ambos lenguajes. Específicamente, la concatenación de $L_1$ y $L_2$, denotada $L_1L_2$ es $L_1L_2 = \{\mathbf{vw} : \mathbf{v} \in L_1, \mathbf{w} \in L_2\}$.

Así como la concatenación de palabras no es conmutativa, tampoco lo es la concatenación de lenguajes. Si $L_1 = \{a, bb, ab, aa\}$ y $L_2 = \{c, cc\}$, entonces $L_1L_2 = \{ac, bbc, abc, aac, acc, bbcc, abcc, aacc\}$, mientras que $L_2L_1 = \{ca, cbb, cab, caa, cca, ccbb, ccab, ccaa\}$.

### 5.1.2. Lenguajes regulares

Al estudiar autómatas finitos, veremos que están relacionados con una clase de lenguajes, llamados regulares. Éstos constituyen la clase más pequeña y más simple de lenguajes formales.

Un lenguaje $L$ es REGULAR si se cumple alguna de las siguientes condiciones:

- $L$ es finito
- $L$ es unión o concatenación de otros lenguajes regulares
- $L$ es la clausura de algún lenguaje regular

En la tabla siguiente se dan ejemplos de lenguajes regulares, así como la justificación de que así es.

| **Lenguaje** | **Justificación** |
|---|---|
| $L_1 = \{01, 10\}$ | Es finito |
| $L_2 = \{0\}$ | Es finito |
| $L_3 = \{0, 00, 000, 0000, ...\}$ = cadenas de ceros | Es la clausura de $L_2$ |
| $L_4 = \{001, 010, 0001, 0010, 00001, 00010, ...\}$ = cadenas de ceros seguida de 01 o de 10 | Es la concatenación de los lenguajes regulares $L_3$ y $L_1$ |
| $L_5$ = cadenas de A y B = $\{A, B\}^*$ | Es la clausura del lenguaje finito $\{A, B\}$ |
| $L_6$= cadenas de A y B que comiencen con A | Es la concatenación del lenguaje regular finito $\{A\}$ con el lenguaje regular $L_5$ |
| $L_7$ = cadenas de A y B que contienen la subcadena BBB | Es la concatenación de $L_5$ con el lenguaje finito {BBB}, concatenado con $L_5$ |
| $L_8$ = palabras reservadas en un lenguaje de programación | Es finito |

Obviamente, si se definen lenguajes regulares, es porque hay lenguajes que no lo son. Tal es el caso del conjunto de cadenas $\{0^n 1^n, n \geqslant 0\}$, donde cada cadena está formada por $n$ ceros seguida de $n$ unos. Tampoco es regular el lenguaje de las cadenas de paréntesis balanceados.

Otro lenguaje no regular es el conjunto de cadenas simétricas, es decir, que se leen igual al derecho y al revés. Por ejemplo, usando el alfabeto $\{A, B, ..., Z\}$, son cadenas en el lenguaje considerado NEUQUEN, ANITALAVALATINA, RECONOCER, ALLIVES-SEVILLA, etc.

## 5.2. Expresiones regulares

### 5.2.1. Definición

Existe una forma compacta de denotar un lenguaje regular, evitando describir en palabras las cadenas que forman parte del lenguaje, o enumerando todas las cadenas (cosa imposible en la mayoría de los lenguajes regulares, ya que en general tienen infinitas cadenas).

Una EXPRESIÓN REGULAR (ER) es una notación descriptiva compacta de un lenguaje que explicita un patrón o regularidad que cumplen las cadenas en el lenguaje. Denotemos con $\mathcal{L}(\mathbf{E})$ el lenguaje denotado por la expresión regular **E**. Las ER se definen constructivamente:

- $\epsilon$ y $\emptyset$ son expresiones regulares, que denotan los lenguajes $\mathcal{L}(\epsilon) = \{\epsilon\}$ y $\mathcal{L}(\emptyset) = \{\}$.

- Si $a$ es un símbolo del alfabeto, $\mathbf{a}$ es una expresión regular y denota el lenguaje formado por la palabra $a$: $\mathcal{L}(\mathbf{a}) = \{a\}$.
- Si $\mathbf{E_1}$ y $\mathbf{E_2}$ son dos expresiones regulares, $\mathbf{E_1} + \mathbf{E_2}$ es una expresión regular y denota el lenguaje unión de los lenguajes representados por $\mathbf{E_1}$ y $\mathbf{E_2}$: $\mathcal{L}(\mathbf{E_1} + \mathbf{E_2}) = \mathcal{L}(\mathbf{E_1}) \cup \mathcal{L}(\mathbf{E_2})$.
- Si $\mathbf{E_1}$ y $\mathbf{E_2}$ son dos expresiones regulares, $\mathbf{E_1E_2}$ es una expresión regular y denota la concatenación de los lenguajes representados por $\mathbf{E_1}$ y $\mathbf{E_2}$: $\mathcal{L}(\mathbf{E_1E_2}) = \mathcal{L}(\mathbf{E_1})\mathcal{L}(\mathbf{E_2})$.
- Si $\mathbf{E}$ es una expresión regular, $\mathbf{E}^*$ es una expresión regular que representa la clausura del lenguaje representado por $\mathbf{E}$: $\mathcal{L}(\mathbf{E}^*) = (\mathcal{L}(\mathbf{E}))^*$.

Así, la ER $(\mathbf{a} + \mathbf{b})^*$ denota la clausura del lenguaje formado por las cadenas $a$ y $b$. Es decir, denota el conjunto de cadenas de caracteres $a$ y $b$ (incluída la cadena vacía $\epsilon$, que está en toda clausura). Y la ER $\mathbf{a}(\mathbf{a} + \mathbf{b})^*$ denota la concatenación del lenguaje $\{a\}$, con la clausura del lenguaje formado por las cadenas $a$ y $b$. Es decir, es el conjunto de cadenas de $a$ y $b$ que comienzan con $a$.

La ER $(\mathbf{x} + \mathbf{y})^*\mathbf{yy}(\mathbf{x} + \mathbf{y})^*$ denota el lenguaje que es concatenación de la clausura del lenguaje $\{x, y\}$, con el lenguaje $\{yy\}$ y la clausura del lenguaje $\{x, y\}$. Es decir, denota el conjunto de cadenas de $x$ e $y$ que contienen la subcadena $yy$. Son cadenas de este lenguaje $xxxyyy$, $xyxyyyxx$, $yy$, etc. Nótese que la cadena vacía $\epsilon$ está en $\{x, y\}^*$, por lo tanto, concatenando $\epsilon$, $yy$ y $\epsilon$ resulta la cadena $yy$.

La ER $(\mathbf{10})^*$ denota la clausura del lenguaje formado por la concatenación de 1 y 0. Así, resulta el conjunto de cadenas de 0 y 1 formadas por repeticiones de 10. Están en este lenguaje las cadenas $\epsilon$, 10, 1010, 101010, etc.

En las operaciones con ER hay un orden de precedencia. La clausura se aplica primero, luego la concatenación, luego la suma. Entonces, la ER $\mathbf{10}^*$ denota las cadenas formadas por concatenación de la cadena 1 con una cadena en la clausura de $\{0\}$, es decir, una cadena de 0. Son cadenas en este lenguaje 1, 10, 100, 1000, etc. Para cambiar el orden de precedencia natural se usan paréntesis.

Por otro lado, la ER $\mathbf{a} + \mathbf{bc}$ denota el lenguaje formado por la cadena $a$ y la cadena $bc$, mientras que $(\mathbf{a} + \mathbf{b})\mathbf{c}$ denota las cadenas formadas por concatenación de las cadenas $a$ o $b$ con la cadena $c$. Es decir, contiene las palabras $ac$ y $bc$.

### 5.2.2. Equivalencia de expresiones regulares

Dos ER son equivalentes si representan el mismo lenguaje. La equivalencia de ER puede identificarse conociendo las siguientes propiedades algebraicas.

Siendo $\mathbf{E_1}$, $\mathbf{E_2}$ y $\mathbf{E_3}$ expresiones regulares, se satisfacen:

- La suma de ER es conmutativa: $\mathbf{E_1} + \mathbf{E_2} = \mathbf{E_2} + \mathbf{E_1}$
- La suma de ER es asociativa: $(\mathbf{E_1} + \mathbf{E_2}) + \mathbf{E_3} = \mathbf{E_1} + (\mathbf{E_2} + \mathbf{E_3})$
- La suma es idempotente: $\mathbf{E_1} + \mathbf{E_1} = \mathbf{E_1}$
- $\emptyset$ es el neutro de la suma: $\mathbf{E_1} + \emptyset = \mathbf{E_1}$
- La concatenación de ER no es conmutativa. En general, $\mathbf{E_1E_2} \neq \mathbf{E_2E_1}$

- La concatenación de ER es asociativa $(\mathbf{E_1E_2})\mathbf{E_3} = \mathbf{E_1}(\mathbf{E_2E_3})$
- $\epsilon$ es el neutro de la concatenación: $\epsilon\mathbf{E_1} = \mathbf{E_1}\epsilon = \mathbf{E_1}$
- $\emptyset\mathbf{E_1} = \mathbf{E_1}\emptyset = \emptyset$
- La concatenación es distributiva respecto de la suma: $\mathbf{E_1}(\mathbf{E_2} + \mathbf{E_3}) = \mathbf{E_1E_2} + \mathbf{E_1E_3}$ y $(\mathbf{E_1} + \mathbf{E_2})\mathbf{E_3} = \mathbf{E_1E_3} + \mathbf{E_2E_3}$
- $\mathbf{E_1}^* = \mathbf{E_1}^*\mathbf{E_1}^* = (\mathbf{E_1}^*)^*$
- $\mathbf{E_1}^* = \epsilon + \mathbf{E_1}\mathbf{E_1}^* = \epsilon + \mathbf{E_1}^+$
- $\epsilon^* = \epsilon$, $\emptyset^* = \epsilon$

En virtud de esas propiedades, se pueden transformar y simplificar ER. Por ejemplo:

- $(\mathbf{1}+\mathbf{2})^*\mathbf{1}(\mathbf{0}+\mathbf{1}) + (\mathbf{1}+\mathbf{2})^*\mathbf{2}(\mathbf{0}+\mathbf{1}) = (\mathbf{1}+\mathbf{2})^*(\mathbf{1}(\mathbf{0}+\mathbf{1}) + \mathbf{2}(\mathbf{0}+\mathbf{1})) = (\mathbf{1}+\mathbf{2})^*(\mathbf{1}+\mathbf{2})(\mathbf{0}+\mathbf{1})$
- $(\mathbf{a}+\mathbf{b}) + \mathbf{c}(\mathbf{a}+\mathbf{b}) = \epsilon(\mathbf{a}+\mathbf{b}) + \mathbf{c}(\mathbf{a}+\mathbf{b}) = (\epsilon+\mathbf{c})(\mathbf{a}+\mathbf{b})$
- $(\mathbf{m}+\mathbf{n})^*\mathbf{n} + (\epsilon + (\mathbf{m}+\mathbf{n})^*)(\mathbf{n}+\mathbf{o}) = (\mathbf{m}+\mathbf{n})^*\mathbf{n} + (\mathbf{m}+\mathbf{n})^*(\mathbf{n}+\mathbf{o}) = (\mathbf{m}+\mathbf{n})^*(\mathbf{n} + (\mathbf{n}+\mathbf{o})) = (\mathbf{m}+\mathbf{n})^*(\mathbf{n}+\mathbf{o})$

Vale la pena explicitar que la clausura no es distributiva respecto de la suma. Es decir, las ER $(\mathbf{a}+\mathbf{b})^*$ y $\mathbf{a}^* + \mathbf{b}^*$ no son equivalentes. La primera indica el lenguaje de cadenas de $a$ y $b$, mientras que la segunda representa el lenguaje formado por las cadenas que usan sólo $a$ o sólo $b$.

### 5.2.3. Ejemplos

***Ejemplo* 5.1**
*Vamos a construir una ER para denotar el lenguaje de las cadenas de 0 y 1 alternados. En este lenguaje están las cadenas 1010, 01010, 101, 010, 10, 0, 1, etc.*

*En principio se puede plantear la ER* $(\mathbf{01})^*$*. Pero todas las cadenas en* $\mathcal{L}((\mathbf{01})^*)$ *comienzan con 0 y terminan con 1. Es decir, falta considerar las cadenas que empiezan con 1 y terminan con 0, y las que empiezan y terminan con 1 o con 0. Considerando todos los casos, se puede plantear la ER:* $(\mathbf{01})^* + \mathbf{1}(\mathbf{01})^* + (\mathbf{01})^*\mathbf{0} + \mathbf{0}(\mathbf{01})^*\mathbf{0}$*. Usando equivalencias de ER, se puede escribir:*

$$(\mathbf{1}+\epsilon)(\mathbf{01})^*(\mathbf{0}+\epsilon)$$

***Ejemplo* 5.2**
*Considere el lenguaje de cadenas de caracteres a, b y c que no tengan dos o más b seguidas, y que no terminen en b. Para construir una ER que represente este lenguaje, debe tenerse en cuenta que cada vez que aparece una b, debe haber otro caracter después. Puede pensarse que las cadenas en este lenguaje están formadas por repeticiones de las cadenas a, c, bc y ba. Entonces, una ER adecuada es* $(\mathbf{a}+\mathbf{c}+\mathbf{ba}+\mathbf{bc})^*$*, que representa la clausura del lenguaje* $\{a, c, ba, bc\}$*. Esa ER también se puede escribir*

$$(\mathbf{a}+\mathbf{c}+\mathbf{b}(\mathbf{a}+\mathbf{c}))^*$$

*Similarmente, el lenguaje de las cadenas de a, b, c que no tienen dos o más b seguidas y que no empiezan con b se representa con*

$$(\mathbf{a}+\mathbf{c}+\mathbf{ab}+\mathbf{cb})^* = (\mathbf{a}+\mathbf{c}+(\mathbf{a}+\mathbf{c})\mathbf{b})^*$$

***Ejemplo* 5.3**
*Sea L el conjunto de cadenas de 0 y 1 tales que el tercer símbolo es 1. La regularidad de las cadenas en este lenguaje es que tienen un primer caracter 0 ó 1, un segundo caracter 0 ó 1, el tercer caracter 1, y luego cualquier cadena (posiblemente vacía) de 0 y 1. Entonces, una ER es*

$$(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1})\mathbf{1}(\mathbf{0}+\mathbf{1})^*$$

***Ejemplo* 5.4**
*La ER* $(\mathbf{A}+\mathbf{B})^*\mathbf{A}(\mathbf{A}+\mathbf{B})$ *representa el lenguaje de cadenas de A y B tales que el anteúltimo caracter es A.* ■

***Ejemplo* 5.5**
*Considere el lenguaje de las cadenas en el alfabeto* $\{a, b\}$ *con una cantidad par de a. Para construir una expresión regular, se debe tener en cuenta que en las cadenas de este lenguaje las a se pueden agrupar en grupos de a dos (no necesariamente consecutivas). Entonces se puede pensar que cada cadena es concatenación de subcadenas, cada una de ellas conteniendo dos a, y cualquier cantidad de b. Entonces, una ER para este lenguaje es*

$$(\mathbf{b}^*\mathbf{ab}^*\mathbf{ab}^*)^*$$

*Esta expresión regular admite la cadena vacía. Eso es correcto, ya que la cadena vacía tiene 0 caracteres a, y el 0 es un número par.* ■

***Ejemplo* 5.6**
*Sea* $L_1$ *el lenguaje de las palabras que usan los caracteres a, b y c, tales que tienen la subcadena bb. Una ER para este lenguaje es*

$$(\mathbf{a}+\mathbf{b}+\mathbf{c})^*\mathbf{bb}(\mathbf{a}+\mathbf{b}+\mathbf{c})^*$$

*Ahora, sea* $L_2$ *el lenguaje de las cadenas de a, b y c, en las que aparece una sola vez la subcadena bb, y no tienen repeticiones de 3 o más b seguidas. La ER anterior no es adecuada, ya que al concatenar* $(\mathbf{a}+\mathbf{b}+\mathbf{c})^*$ *con* $\mathbf{bb}$*, quedan admitidas palabras que se obtienen de concatenar, por ejemplo ab con bb, lo que genera la cadena abbb, que no está en* $L_2$*.*

*Entonces, las cadenas en* $L_2$ *se pueden obtener concatenando una subcadena de a, b, y c que no termine en b (posiblemente la cadena vacía), con la subcadena bb, concatenado con otra cadena de a, b, y c que no empiece con b (posiblemente la cadena vacía). Entonces, en vista del ejemplo 5.2, una ER adecuada es*

$$(\mathbf{a}+\mathbf{c}+\mathbf{b}(\mathbf{a}+\mathbf{c}))^*\mathbf{bb}(\mathbf{a}+\mathbf{c}+(\mathbf{a}+\mathbf{c})\mathbf{b})^*$$

***Ejemplo* 5.7**
*Una ER que describe los números decimales es*

$$(- + \epsilon)(\mathbf{0-9})(\mathbf{0-9})^*.(\mathbf{0-9})(\mathbf{0-9})^*$$

*La notación* $(\mathbf{0-9})$ *indica* $(\mathbf{0+1+2+...+9})$.

*La primera parte de la ER posibilita que el número decimal tenga signo - o no. Luego sigue una cadena de 1 o más dígitos, seguido de un punto, seguido de 1 o más dígitos.* ■

***Ejemplo* 5.8**
*Considere el conjunto de domicilios. Por simplicidad, suponemos que todas las domicilios constan de nombre de calle, formado por una o más palabras que usan sólo letras, y número de casa (uno o más dígitos). Todas las palabras que forman el nombre de calle comienzan con letra mayúscula, y están separadas por un espacio.*

*Una ER para este conjunto de cadenas es*

$$(\mathbf{A-Z})(\mathbf{a-z})^* \smile ((\mathbf{A-Z})(\mathbf{a-z})^* \smile)^*(\mathbf{0-9})(\mathbf{0-9})^*$$

*La notación* $(\mathbf{A-Z})$ *es una abreviación de* $(\mathbf{A+B+C+...+Y+Z})$, *y* $\mathbf{a-z}$ *denota* $\mathbf{a+b+c+...+y+z}$. ■

Una extensión de la definición de ER incluye la operación potenciación de ER, como caso finito de la clausura. Si $\mathbf{E}$ es una ER y $n$ es un número natural, $\mathbf{E^n}$ también es una ER que representa el lenguaje obtenido al concatenar $n$ palabras del lenguaje representado por $\mathbf{E}$.

También se usa $\mathbf{E}^+$ para denotar la clausura del lenguaje de $\mathbf{E}$ concatenado con ese mismo lenguaje, es decir: $\mathbf{E}^+ = \mathbf{E}^*\mathbf{E} = \mathbf{E}\mathbf{E}^*$.

Con esta convención, suponiendo que los domicilios siempre tienen 4 dígitos, la ER del ejemplo 5.8 se puede escribir $((\mathbf{A-Z})(\mathbf{a-z})^* \smile)^+(\mathbf{0-9})^\mathbf{4}$. Más generalmente, admitiendo domicilios con 1, 2, 3, ó 4 dígitos, la ER es
$((\mathbf{A-Z})(\mathbf{a-z})^* \smile)^+((\mathbf{0-9}) + (\mathbf{0-9})^\mathbf{2} + (\mathbf{0-9})^\mathbf{3} + (\mathbf{0-9})^\mathbf{4})$

***Ejemplo* 5.9**
*Una ER para el lenguaje de ciertas direcciones de emails puede plantearse de la siguiente forma:*

$$(\mathbf{A-Z+a-z+0-9})^+@(\mathbf{A-Z+a-z})^+.(\mathbf{A-Z+a-z})^+.(\mathbf{A-Z+a-z})^+$$

## 5.3. Autómatas

### 5.3.1. Autómatas finitos deterministas

Un autómata es una máquina o dispositivo con capacidad de computación. Es decir, un dispositivo que puede recibir información (codificada con cierto alfabeto) y dar información o una salida referida a ella. Su estudio es importante en ciencias de la computación, ya que permite estudiar en forma teórica qué es computable y qué no. En esta sección se definirá un tipo particular de autómatas, conocidos como autómatas finitos.

Algunas de las aplicaciones de los autómatas finitos son

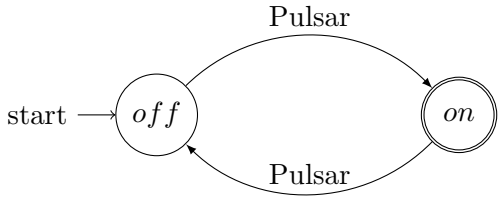



Figura 5.1: Esquema del interruptor del ejemplo 5.10

- Dado el código de un programa (codificado como cadena de símbolos o caracteres), decidir si está bien formulado, si pertenece a determinado lenguaje de programación.
- Reconocimiento de lenguajes: determinar si una cadena pertenece o no a un determinado lenguaje.
- Construcción y análisis de software para verificación de circuitos digitales.
- Analizador léxico de un compilador, que separa el código de un programa en palabras reservadas, identificadores, signos de puntuación, etc.
- Buscadores de páginas, texto o patrones.
- Software para comprobar sistemas con un número finito de estados (protocolos).

Más adelante se definirá formalmente autómata finito, pero antes se describirán ejemplos sencillos.

***Ejemplo* 5.10**
*Un interruptor tiene dos posiciones o estados: encendido y apagado. Inicialmente, digamos que está apagado. El interruptor entiende una única entrada externa: el hecho de que se pulse el botón del dispositivo. Estando en el estado apagado, si recibe una pulsación, cambia de estado, pasa a estar en el estado encendido. Y si está en este estado y recibe una pulsación, cambia al estado apagado. La situación se la puede representar con el esquema de la figura 5.1.*

*Si se quiere encender el dispositivo, luego de cierta cantidad de pulsaciones el interruptor debe estar en el estado encendido. Así, si el interruptor recibe, por ejemplo, la entrada* `Pulsar` *o la entrada* `Pulsar Pulsar Pulsar`*, quedará en el estado encendido.*

***Ejemplo* 5.11**
*Suponga que se tiene un analizador léxico de un compilador, que busca la palabra reservada* ***while****. El analizador lee uno a uno los caracteres del código fuente. Es decir, la información que recibe del exterior está codificada en caracteres ASCII. Si en algún momento lee el caracter* ***w****, el analizador debe recordarlo, ya que puede ser el comienzo de la palabra buscada. Si el caracter siguiente es* ***h****, debe recordar que hasta el momento leyó* ***wh****, ya que puede ser parte de la palabra buscada. En cambio, si después de* ***w****, lee cualquier otro caracter, distinto de* ***h****, quiere decir que no comenzó a leer la palabra buscada.*

*De la misma forma, a medida que lee caracteres, debe recordar si los últimos caracteres leídos pueden formar parte de la palabra buscada. Esto hasta leer los 5 caracteres* ***w****,* ***h****,* ***i****,* ***l****, y* ***e*** *consecutivamente, o terminar de leer el código y no haber encontrado la palabra.*

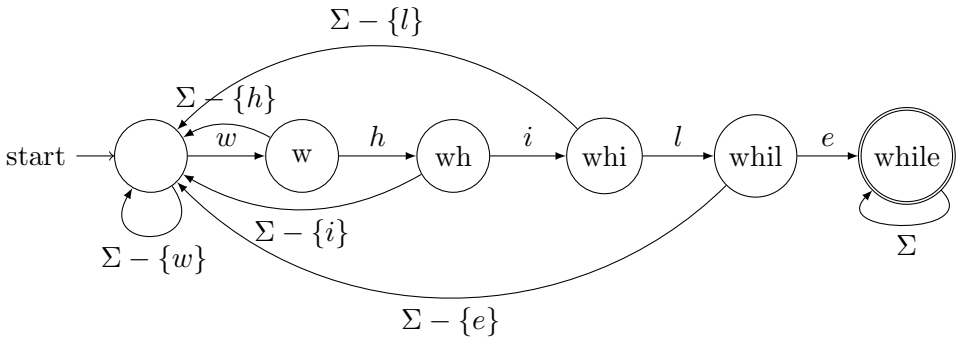



Figura 5.2: Esquema del autómata buscador de **while** (Ejemplo 5.11)

*En cada momento, el analizador se encuentra en un estado, y cada estado puede corresponder a una parte de la palabra **while** leída hasta ese momento. Al leer un nuevo caracter, según cuál sea, cambiará su estado.*

*Esta situación se puede representar en el diagrama de la figura 5.2. En este grafo los nodos son los estados, y los arcos están etiquetados con caracteres, indicando el cambio de estado al leer el correspondiente caracter. El nodo más a la izquierda es el nodo inicial, corresponde al estado que se encuentra el analizador antes de leer algún caracter. Y el nodo más a la derecha es el estado final. Si el analizador llega a ese estado, es porque encontró la palabra **while** en el texto.*

***Ejemplo* 5.12**
*Una máquina expendedora de gaseosas recibe monedas de \$1 y \$2. Una lata de gaseosa cuesta \$4. A medida que la máquina recibe monedas debe ir recordando la suma total para decidir si luego de una determinada entrada entrega o no el producto. Entonces podemos decir que la máquina puede encontrarse en el estado $q_0$ si no recibió ninguna moneda, en el estado $q_1$ si recibió un total de \$1, en el estado $q_2$ si hasta el momento recibió \$2, en el estado $q_3$ si recibió \$3 en total, o en el estado $q_{+4}$ si recibió \$4 o más. Nótese que no es importante si recibe más o exactamente \$4, ya que no se analiza en este modelo la entrega de vuelto.*

*Si el autómata está en el estado $q_0$ y recibe \$1, pasará al estado $q_1$, y si recibe una moneda de \$2 pasará al estado $q_2$. Así en cada uno de los estados: recibiendo una u otra moneda, cambiará su estado para recordar la nueva situación.*

*$q_0$ es el estado inicial, en el que se encuentra antes de empezar a recibir monedas, y $q_{+4}$ es el estado final o de aceptación: si llega a este estado, la entrada de monedas se acepta y la máquina entrega el producto.*

*La situación puede representarse en el gráfico de la figura 5.3. Como en el ejemplo anterior, los estados se representan con nodos en el grafo, y las transiciones entre estados mediante arcos. Los arcos están etiquetados de acuerdo a la entrada que provoca ese cambio de estado.* ■

Con los ejemplos motivadores descritos, se presenta la definición formal de autómata finito determinista.

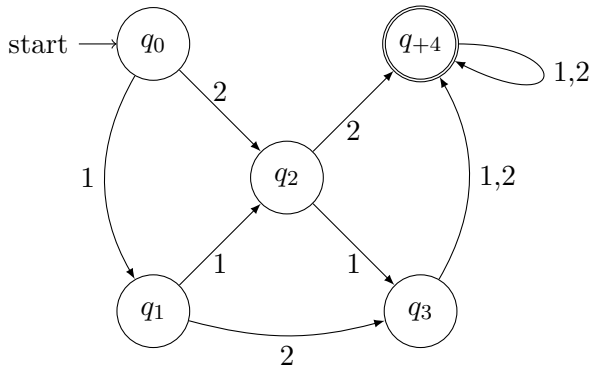



Figura 5.3: Esquema del autómata del ejemplo 5.12

Un AUTÓMATA FINITO DETERMINISTA (AFD) consta de

- Un conjunto finito de estados (Q),
- un alfabeto de entrada $\Sigma$ (caracteres que puede leer el autómata),
- una función de transición $\delta$: para cada estado $q$ y cada caracter $a$ del alfabeto de entrada, la función de transición $\delta(q, a)$ indica a qué estado pasa el autómata si estando en el estado $q$ lee el caracter $a$,
- un estado inicial,
- uno o más estados de aceptación.

Puede pensarse que la cadena de entrada está escrita en una celda semiinfinita, de modo que el primer caracter de la cadena está escrito en la primera celda. El autómata lee uno a uno el contenido de las celdas, desde el primero hasta el último caracter de la cadena de entrada. El funcionamiento de un AFD consiste en ir pasando de un estado a otro a medida que lee los caracteres. Estos pasajes de estado están dados por la función de transición. Al terminar de leer una palabra, el autómata se encuentra en algún estado. Si ese estado es uno de los estados de aceptación (indicados en los grafos como nodos con círculo doble), se dice que el autómata acepta la cadena. Por ejemplo, en el ejemplo de la máquina expendedora de gaseosas, el único estado de aceptación es $q_{+4}$. La cadena 22 es aceptada por el autómata, así como las cadenas 11111, 2111, 212. No son cadenas aceptadas 111, 21, 12 (el autómata termina en $q_3$), 2, 11 (el autómata termina en el estado $q_2$), 1 (termina en $q_1$), y la cadena vacía (termina en $q_0$).

El LENGUAJE ACEPTADO por un autómata es el conjunto de cadenas aceptadas por éste. También se dice que es el lenguaje que RECONOCE el autómata.

El tipo de lenguaje que puede reconocer un AFD es el conjunto de lenguajes regulares. Esto es: el lenguaje aceptado por un AFD es un lenguaje regular; y viceversa: dado un lenguaje regular, existe un AFD que lo reconoce. Esto implica que un lenguaje que no sea regular no puede ser reconocido por un AFD. Se verán más adelante autómatas más generales, capaces de aceptar lenguajes no regulares.

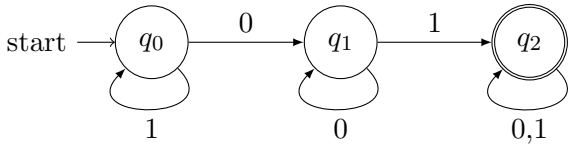



Figura 5.4: Grafo de transción para el AFD del ejemplo 5.13

El adjetivo determinista se refiere a que el funcionamiento del autómata es determinista, para cada caracter de entrada, existe un único estado al que el autómata puede llegar partiendo del estado actual. En cada momento se encuentra en un único estado y esto está determinado únicamente por la cadena de caracteres leídos.

El AFD no tiene dispositivo de memoria. El único medio para recordar lo que leyó hasta el momento es el conjunto de estados. Por eso se dice que es una máquina de memoria limitada.

Un autómata se puede representar con un grafo como en los ejemplos anteriores, llamado DIAGRAMA DE TRANSICIÓN. Otra forma de representarlo es a través de una TABLA DE TRANSICIÓN, que tiene una fila por estado y una columna por caracter del alfabeto de entrada. En la celda intersección de la fila correspondiente al estado $q$ y la columna correspondiente a un caracter $a$ se coloca la tansición $\delta(q, a)$. La fila del estado inicial se marca con $\rightarrow$, y las filas correspondientes a los estados de aceptación se marcan con *.

Para el ejemplo 5.12, la tabla de transición es

| | 1 | 2 |
|---|---|---|
| $\rightarrow q_0$ | $q_1$ | $q_2$ |
| $q_1$ | $q_2$ | $q_3$ |
| $q_2$ | $q_3$ | $q_{+4}$ |
| $q_3$ | $q_{+4}$ | $q_{+4}$ |
| $*q_{+4}$ | $q_{+4}$ | $q_{+4}$ |

***Ejemplo* 5.13**
*Sea el lenguaje* $L = \{\mathbf{w}01\mathbf{v}, \mathbf{w} \in \{0,1\}^*, \mathbf{v} \in \{0,1\}^*\}$. *Es decir, el conjunto de las cadenas de 0 y 1 que contienen la secuencia 01. Son palabras en este lenguaje 111101, 01, 0101010, 00000110, mientras que las cadenas 111110, 0000, 11,* $\epsilon$ *no pertenecen a* $L$. *Un AFD que acepta ese lenguaje se muestra en la figura 5.4. Dado que existe un AFD que reconoce este lenguaje, es regular. Una ER para este lenguaje es* $(\mathbf{0}+\mathbf{1})^*\mathbf{01}(\mathbf{0}+\mathbf{1}^*)$.

*Nótese que si la entrada es 111101, 01, 0101010, 00000110 el AFD termina en el estado* $q_3$ *que es estado de aceptación. Mientras que si la entrada es 111110 ó 0000 el AFD termina en el estado* $q_1$, *y si la entrada es 11 o* $\epsilon$ *el AFD termina en el estado* $q_0$, *y ninguno de estos estados es de aceptación.* ■

***Ejemplo* 5.14**
*Considere el lenguaje de las cadenas de A y B que tienen una cantidad impar de A.*

*Un autómata para reconocer este lenguaje debe recordar si la cantidad de A leídas hasta el momento es par o impar. Se puede diseñar con dos estados:* $q_0$, *donde se encuentra el*

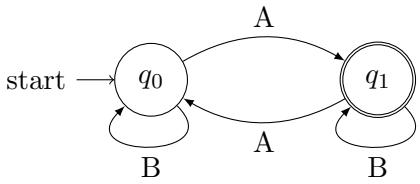



Figura 5.5: Grafo de transción para el AFD del ejemplo 5.14

*autómata cuando la cantidad de A leídas hasta el momento es par, y $q_1$, donde se encuentra el autómata cuando la cantidad de A leídas hasta el momento es impar. Entonces, $q_1$ es el único estado de aceptación.*

*Un AFD que acepta el lenguaje considerado se muestra en la figura 5.5.*

*Este lenguaje también es regular. ¿Qué regularidad tienen las palabras de este lenguaje? Las A en cada cadena se pueden agrupar de a dos (no necesariamente consecutivas), y queda una sola sin estar en un par. Y entre las A puede haber o no caracteres B. Entonces, una ER posible es:* $\mathbf{B}^*\mathbf{A}\mathbf{B}^*(\mathbf{A}\mathbf{B}^*\mathbf{A}\mathbf{B}^*)^*$. ■

***Ejemplo* 5.15**

*Sea L el lenguaje de las cadenas de A y B que tienen una cantidad par de A y una cantidad par de B.*

*Un autómata que reconozca este lenguaje puede encontrarse en la situación de haber leído cantidad par de A y de B, cantidad par de A e impar de B, cantidad impar de A y de B, cantidad impar de A y par de B. Entonces, se necesitan 4 estados, que se denotan $q_0$, $q_1$, $q_2$ y $q_3$ respectivamente. En este caso, el estado de aceptación es $q_0$.*

*La tabla de transición para este autómata es*

| | $A$ | $B$ |
|---|---|---|
| $^* \rightarrow q_0$ | $q_3$ | $q_1$ |
| $q_1$ | $q_2$ | $q_0$ |
| $q_2$ | $q_1$ | $q_3$ |
| $q_3$ | $q_0$ | $q_2$ |

■

*En la figura 5.6 se muestra el grafo de transición.*

*Si la entrada del autómata es AABBAA, el autómata pasa por los estados $q_0$, $q_3$, $q_0$, $q_1$, $q_0$, $q_3$, $q_0$, terminando en el estado de aceptación. Por lo tanto, esta cadena es aceptada. Si la entrada es BBBB, el autómara pasa por los estados $q_0$, $q_1$, $q_0$, $q_1$, $q_0$, y también es aceptada. Si la entrada es $\epsilon$, el AFD que inicialmente está en $q_0$, permanece allí, porque no lee ningún caracter, por lo tanto no hay transición. Entonces, la cadena vacía es aceptada.* [1]

***Ejemplo* 5.16**

*Considere la tabla de transición dada a continuación.*

[1] La cadena vacía tiene cero A y cero B. Y cero es un número par, entonces esta cadena satisface la definición del lenguaje, y pertenece a él.

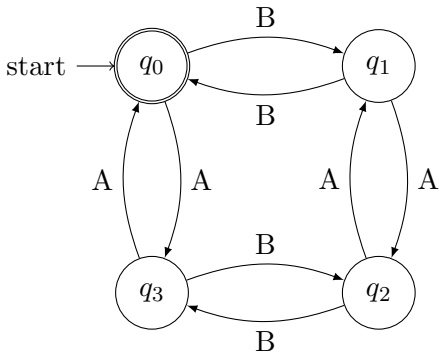



Figura 5.6: Grafo de transción para el AFD del ejemplo 5.15

| | $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|---|
| $^*\rightarrow q_0$ | $q_2$ | $q_1$ | $q_2$ | $q_2$ |
| $^*q_1$ | $q_2$ | $q_2$ | $q_0$ | $q_0$ |
| $q_2$ | $q_2$ | $q_2$ | $q_2$ | $q_2$ |

■

*Se quiere identificar el lenguaje aceptado por el AFD que tiene la función de transición de la tabla. Una forma fácil de hacerlo es dibujar el grafo de transiciones e identificar qué cadenas llevan al autómata a uno de los estados de aceptación.*

*En la figura 5.7 se muestra el diagrama de transición correspondiente. Se pueden analizar distintas cadenas de prueba, para ver cómo se comporta el autómata. Por ejemplo, dadas las cadenas ABC, DDCD, AADBBBD, BCD, BBABD, BB, al ser leídas por el autómata, éste termina en el estado $q_2$, que no es de aceptación. Por lo tanto, estas cadenas no están en el lenguaje reconocido por el AFD.*

*Por otra parte sí acepta las cadenas BC, BCBDBC, BDBC (el autómata termina en $q_0$) y B, BDBDB, BCB, BCBDB (el autómata termina en $q_1$). ¿Qué características definen a las cadenas aceptadas? No tienen el caracter A, comienzan con B, y después de cada B, si no es el último caracter, debe aparecer una C o una D. Y después de cada C o D (si no son el último caracter) debe aparecer una B.*

*La descripción algebraica de este lenguaje con una ER es $(\mathbf{BC} + \mathbf{BD})^*(\epsilon + \mathbf{B})$. Esto indica que cada cadena en el lenguaje aceptado es concatenación de BC y BD (entre cero y varias veces), y concatenada con una B final, o con la cadena vacía.*

***Ejemplo* 5.17**

*Se define el lenguaje $L_1$ de las cadenas binarias que comienzan con 00. En la figura 5.8a se muestra el autómata que reconoce este lenguaje. Los estados $q_0$, $q_1$, $q_2$ y $q_3$ representan la situación de no haber leído ningún caracter, haber recibido un 0 al inicio, haber recibido dos 0 consecutivos al inicio, y haber recibido algo distinto a dos 0 al inicio, respectivamente. Obviamente, el estado de aceptación es $q_2$. Luego de haber detectado los dos 0 iniciales, y estar en el estado $q_2$ se puede seguir leyendo cualquier caracter, y el autómata permanece en el mismo estado.*

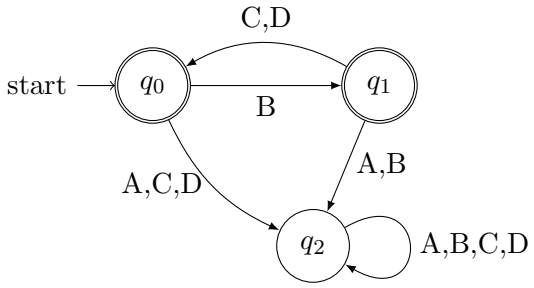



Figura 5.7: AFD del ejemplo 5.16

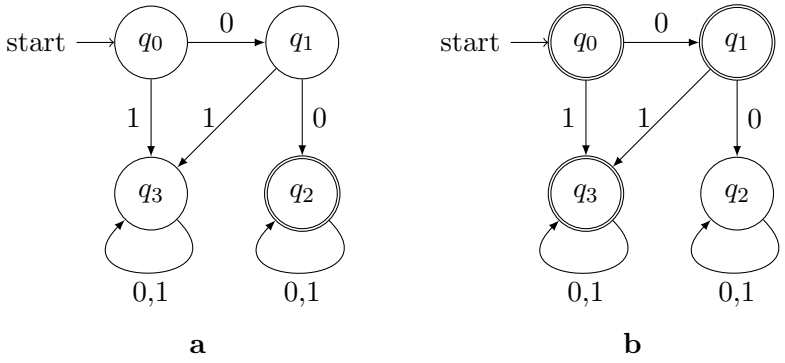



Figura 5.8: Autómatas del ejemplo 5.17

*Considere ahora el diseño de un autómata que acepte el lenguaje $L_2$ de las cadenas binarias que no comiencen con 00. Nótese que el lenguaje considerado es exactamente el complemento del anterior. Es decir una cadena binaria está en $L_2$ si y sólo si no está en $L_1$. El autómata mostrado en la figura 5.8a reconoce palabras que comienzan con 00 si al terminar de leerlas se encuentra en $q_2$. Si al leer una cadena de 0 y 1 el autómata termina en otro estado, es porque no comienza con 00. Luego, esa cadena está en $L_2$. Entonces, un autómata para reconocer $L_2$ es similar al anterior, pero los estados de aceptación deben ser todos menos $q_2$. Tal autómata se muestra en la figura 5.8b.* ■

Este ejemplo muestra que si se conoce un AFD que acepte un lenguaje $L$, para construir un AFD que acepte el lenguaje $L^c = \Sigma^* - L$, basta con tomar como estados de aceptación a los que antes no lo eran, y viceversa. Esto se conoce como *diseño por complemento*.

Para un mismo lenguaje se pueden construir autómatas distintos. Se dice que dos autómatas son equivalentes si aceptan exactamente el mismo lenguaje.

### 5.3.2. Autómata finito no determinista

Un AUTÓMATA FINITO NO DETERMINISTA (AFN) está formado por

- Un conjunto finito de estados (Q),

- un alfabeto de entrada $\Sigma$ (caracteres que puede leer el autómata),
- una relación de transición $\delta$: dado un estado $q$ y un caracter $a$ del alfabeto de entrada, la transición $\delta(q, a)$ indica a qué estados podría pasar el autómata si estando en el estado $q$ lee el caracter $a$; también puede haber transiciones para un estado $q$ y la palabra vacía $\epsilon$: la transción $\delta(q, \epsilon)$ indica a qué estado podría pasar el autómata desde el estado $q$, sin necesidad de consumir un caracter de la entrada (éstas se denominan transiciones espontáneas),
- un estado inicial,
- uno o más estados de aceptación.

De acuerdo a esta definición, la diferencia entre AFD y AFN es que las transiciones en este último están dadas por una relación (no función): podría haber más de un posible estado al que pasa el autómata en un momento determinado (es decir, para un $q$ y un caracter $a$, $\delta(q, a)$ podría ser más de un estado). Así, en cada momento el AFN puede estar en más de un estado. Por otra parte, puede no haber transición desde determinado estado con determinado caracter (es decir, $\delta(q, a)$ puede ser vacío para algún $q$ y algún $a$), y puede haber transiciones espontáneas (cambios de estado sin leer caracteres de la entrada).

Claramente, esta definición es más abarcativa, y de hecho, todo AFD es un AFN.

Para el caso de AFN, el lenguaje aceptado es el conjunto de cadenas tales que el autómata puede terminar en un estado de aceptación cuando las lee. Como el AFN puede estar simultáneamente en varios estados, al terminar de leer una cadena podría estar en un conjunto de estados. Tal cadena será aceptada por el autómata si este conjunto de estados contiene algún estado de aceptación.

Por ejemplo, considere el AFN mostrado en la figura 5.9.

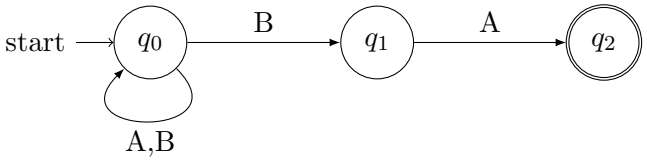
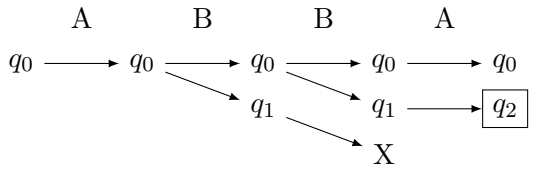



Figura 5.9: Autómata no determinista

El autómata no es determinista ya que existen dos transiciones desde el estado $q_0$ con el caracter B. Esto es, cuando el autómata, estando en el estado $q_0$ lee un caracter B, puede pasar al estado $q_1$ o quedarse en el estado $q_0$. Entonces no está determinado el estado siguiente.

Por otro lado, no hay transición desde el estado $q_1$ con el caracter B. Y finalmente, no hay ninguna transición desde el estado $q_2$.

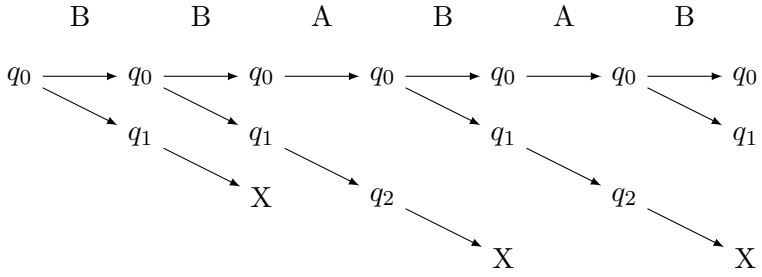
Para analizar cómo se comporta el autómata, se puede simular su funcionamiento para algunas cadenas. Suponga que la cadena de entrada es ABBA. En el gráfico siguiente se muestra el paso de un estado a otro a medida que lee los 4 caracteres.




Estando en $q_0$ lee el primer caracter A, queda en el estado $q_0$. Luego, cuando recibe el caracter B, puede pasar al estado $q_1$ o quedarse en $q_0$. Esto se indica con una bifurcación en la trayectoria de estados. Para cada uno de estos posibles estados, se indica a qué estado pasa al leer el siguiente caracter, la segunda B. Estando en $q_1$, al leer la B, esa trayectoria se termina, se bloquea, dado que no hay transición para este caso. Esto se indica con una X en tal trayectoria.

De esta forma, se van abriendo y posiblemente bloqueando trayectorias hasta que lee todos los caracteres. Los estados finales de todas las trayectorias son $q_0$ y $q_2$. Como uno de estos estados es de aceptación ($q_2$), la cadena es aceptada.

Veamos el funcionamiento del autómata con la cadena BBABAB. Se representa en el siguiente gráfico:




Al terminar de leer la cadena, las trayectorias no bloqueadas llegan al estado $q_0$ o al estado $q_1$. Como ninguno de éstos es estado de aceptación, la palabra no se acepta, no pertenece al lenguaje reconocido por el AFN.

¿Qué características tiene el lenguaje aceptado? Observe que para llegar al estado de aceptación, debe leer los caracteres BA desde $q_0$, y no debe aparecer otro caracter después. Entonces, las cadenas aceptadas son aquellas que terminen con BA. El lenguaje es $(\mathbf{A} + \mathbf{B})^*\mathbf{BA}$.

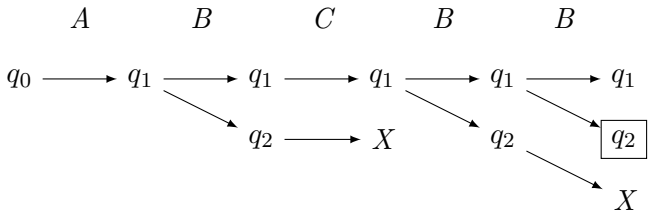
Veamos otros ejemplos de AFN.

***Ejemplo* 5.18**

*Consideremos el lenguaje de las cadenas de A, B y C que comiencen con A y terminen con B. Es muy fácil construir un AFN que reconozca este lenguaje, un posible diseño se observa en la figura 5.10.*

*El autómata es no determinista porque no existe transición desde $q_0$ con los caracteres B o C, no existen transiciones desde $q_2$ con ningún caracter, y existen dos transiciones desde $q_1$ con el caracter B.*

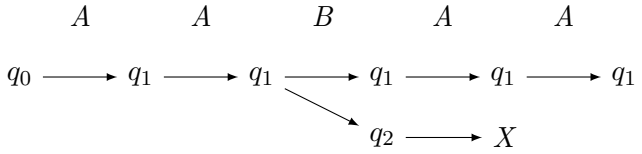
*Si la entrada a ese AFN es la cadena ABCBB, el comportamiento es como se muestra en el gráfico:*




*Al terminar de leer la cadena, las trayectorias no bloqueadas llegan a $q_1$ o $q_2$. Como $q_2$ es estado de aceptación, la cadena es aceptada.*

*Si, en cambio, la cadena de entrada es BBACB, el autómata se bloquea al intentar leer el primer caracter B, estando en el estado inicial $q_0$, ya que no hay una transición definida desde allí con el caracter B.*

*Con la cadena de entrada AABAA, el autómata funciona como muestra el diagrama:*




*Al terminar de leer la cadena, el AFN puede encontrarse únicamente en $q_1$, que no es estado de aceptación. Entonces, la cadena no es aceptada.* ■

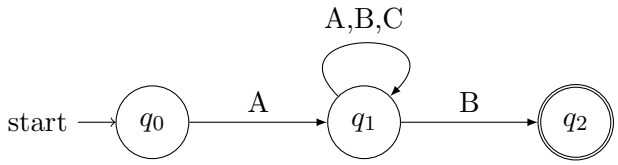



Figura 5.10: AFN del ejemplo 5.18

***Ejemplo* 5.19**
*Dado el alfabeto $\Sigma = \{0, 1, ..., 9, +, -, .\}$, se puede definir el lenguaje de los números decimales. Una cadena que representa un número decimal puede comenzar con un signo +, un signo -, o puede no tener signo inicial. A continuación sigue una cadena de dígitos (parte entera), un punto, y otra cadena de dígitos (parte decimal). Se puede suponer que alguna de estas cadenas de dígitos puede ser vacía, pero no ambas. Entonces, son cadenas válidas +3.234, -0.001, .45, -23., 98.1.*

*El autómata no determinista de la figura 5.11 reconoce este lenguaje.*

*Veamos el funcionamiento cuando la cadena de entrada es 98.1. Se muestra en el siguiente diagrama:*

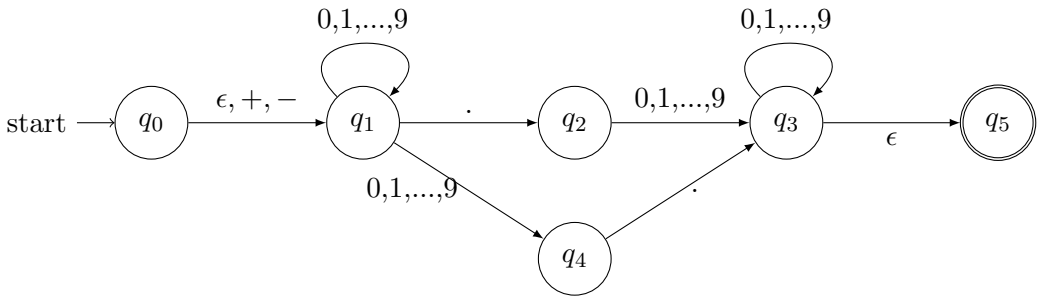
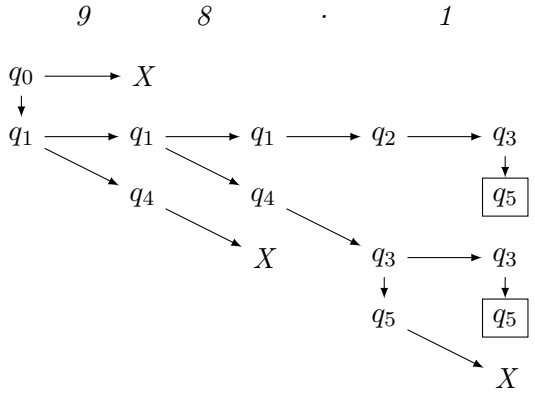


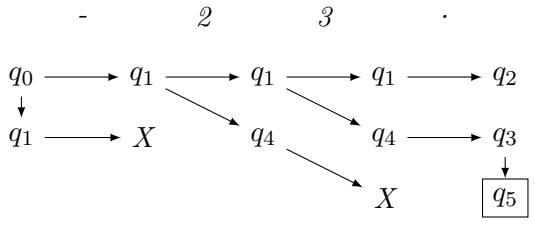

Figura 5.11: AFN que reconoce números decimales



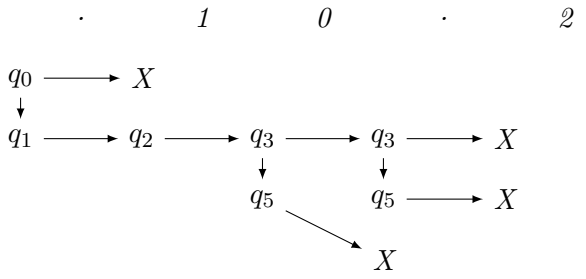

*La transición de $q_0$ a $q_1$ se produce sin consumir ningún caracter de la entrada. Esto es debido a que esa transición se produce con la cadena vacía $\epsilon$. También hay transiciones con la cadena vacía de $q_3$ a $q_5$.*

*Al terminar de leer la cadena de entrada, el AFN puede estar en el estado $q_3$ o en el estado $q_5$. Como uno de éstos es estado de aceptación, la cadena se acepta.*

*Si la cadena de entrada es -23., el funcionamiento es como se muestra en la figura:*




*Consideremos la cadena .10.2, que no pertenece al lenguaje considerado. El comportamiento del autómata con esta entrada es ilustrado en el siguiente gráfico:*




■

*Cada vez que el autómata esté en $q_0$ o en $q_3$ puede pasar espontáneamente (sin consumir caracteres de la entrada) al estado $q_1$ y $q_5$ respectivamente, por la transición con $\epsilon$.*

*El autómata no termina de leer la entrada, porque todas las trayectorias posibles están bloqueadas luego de leer el segundo punto. Entonces, el autómata no acepta esta palabra.*

*Una ER para este lenguaje es:*

$$(\epsilon + \underline{+} + \underline{-})(\mathbf{0}-\mathbf{9})^*.(\mathbf{0}-\mathbf{9})(\mathbf{0}-\mathbf{9})^* + (\epsilon + \underline{+} + \underline{-})(\mathbf{0}-\mathbf{9})(\mathbf{0}-\mathbf{9})^*.(\mathbf{0}-\mathbf{9})^*$$

*Se usó la notación $\underline{+}$ y $\underline{-}$ para simbolizar los caracteres + y - del alfabeto de entrada, y no confundirlo con el + de la ER.*

***Ejemplo* 5.20**

*Sean los alfabetos $\Sigma_1 = \{A, B, ..., Z, a, b, ..., z, \smile,$ á, é, í, ó, ú $\}$ y $\Sigma_2 =$ { ¿,?,!,¡,@,(,),.}. La unión $\Sigma = \Sigma_1 \cup \Sigma_2$ es el alfabeto de entrada del autómata de la figura 5.12. Es un autómata no determinista porque desde $q_0$ hay dos transiciones con el caracter $\smile$, desde $q_1$ no hay transiciones para casi todos los caracteres, no hay transiciones desde $q_3$, etc.*

*Las siguientes son cadenas aceptadas:*

*Hey$\smile$¿estás?*

*$\smile$¿Cuál$\smile$es?*

*Hola$\smile$Juan$\smile$¿Cómo$\smile$estás?$\smile$Vamos*

*Podría pensarse en principio que este AFN acepta las cadenas que contengan una pregunta bien formulada. Esto no es correcto, porque si bien las cadenas que contienen una pregunta son aceptadas, también son aceptadas otras cadenas que no son de esa forma, por ejemplo: $\smile$¿?aaa, b$\smile$¿sje?¡.*

*El lenguaje aceptado es regular, y se puede representar con la ER:* $\mathbf{E}^* \smile ¿\ \mathbf{E_1}^*?\mathbf{E}^*$

*donde* $\mathbf{E_1} = (\mathbf{A}-\mathbf{Z}+\mathbf{a}-\mathbf{z}+ \smile + \textbf{á-ú})$ *y* $\mathbf{E} = \mathbf{E_1} + ¿ + ? + ¡ + ! + @ + ( + ) + .$ ■

No se demostrará aquí, pero puede probarse que dado un AFN, puede construirse un AFD que acepte exactamente el mismo lenguaje. Existe un procedimiento algorítmico para tal construcción. Entonces, cabe preguntarse cuál es la ventaja de introducir no determinismo, si no se amplía el conjunto de lenguajes reconocibles. La ventaja es que desde el punto de vista del diseño de estos autómatas, es más fácil construir un AFN para un determinado lenguaje que un AFD, y resulta un autómata más simple.

Con el procedimiento algorítmico para obtener un AFD a partir de un AFN, resulta un autómata determinista que, en general tiene muchos más estados. En el peor de los casos, partiendo de un AFN con $n$ estados, el AFD equivalente tiene $2^n$ estados.

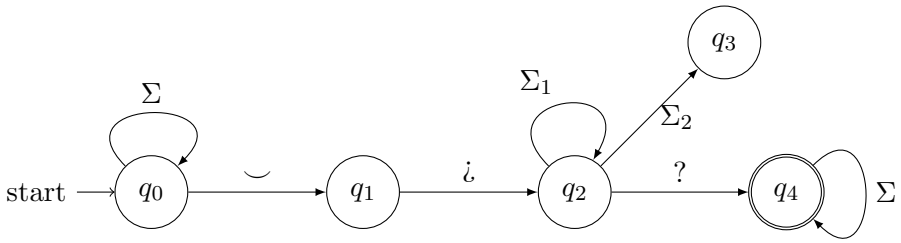



Figura 5.12: AFN del ejemplo 5.20

Lo dicho hasta aquí implica que todo lenguaje aceptado por un AFN es aceptado también por un AFD. Entonces, el tipo de lenguaje aceptado por la clase de autómatas no deterministas es el de lenguajes regulares.

El funcionamiento de los autómatas se puede implementar algorítmicamente con un programa que codifique la función (relación) de transición, y simule el cambio de estado. Decidir si una cadena de longitud $n$ es aceptada o no por un AFD puede hacerse en tiempo lineal de orden $n$.

El no determinismo en la simulación de AFN implica explorar varios caminos posibles a fin de analizar si alguno llega a un estado de aceptación. Esto es, se debe explorar en profundidad un árbol de altura $n$ (para una cadena de longitud $n$). Esto requiere un tiempo exponencial de orden $n$.

### 5.3.3. Máquinas con salida

Los autómatas definidos hasta ahora sólo dan una salida *si* o *no* acerca de la pertenencia de una cadena a un lenguaje. Hay otro tipo de autómatas que dan una salida diferente a *si* o *no*. Son llamados AUTÓMATAS FINITOS CON SALIDA, o MÁQUINA DE ESTADO FINITO. La salida es un caracter perteneciente a cierto alfabeto (puede ser igual o no al alfabeto de entrada), y la salida se puede producir al realizar una transición, o estando en un estado.

El concepto de máquina de estado finito es más general que el de autómata finito. De hecho, los AFD estudiados anteriormente pueden verse como máquina de estado finito, siendo la salida 0 y 1. Una salida 0 corresponde a estados que no son de aceptación, y una salida 1 corresponde a estados de aceptación.

#### Máquina de Mealy

En las máquinas de Mealy la salida depende de la transición. La representación gráfica es con el grafo de transiciones, y en este caso la etiqueta de los arcos son de la forma $a/w$ donde $a$ es el caracter que lee de entrada, y $w$ es el caracter de salida.

En la figura 5.13 se muestra un ejemplo de una máquina de Mealy que da como salida la cadena de entrada invertida. El funcionamiento al leer la cadena 0010111 se muestra en el esquema siguiente.

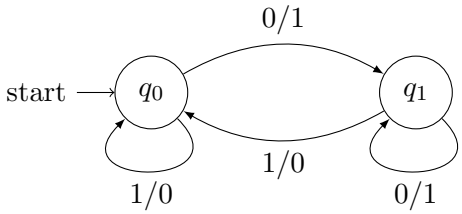



Figura 5.13: Máquina de Mealy que invierte la cadena de entrada

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Estado** | $q_0$ | | $q_1$ | | $q_1$ | | $q_0$ | | $q_1$ | | $q_0$ | | $q_0$ | $q_0$ |
| **Caracter** | | 0 | | 0 | | 1 | | 0 | | 1 | | 1 | | 1 |
| **Salida** | | 1 | | 1 | | 0 | | 1 | | 0 | | 0 | | 0 |

***Ejemplo* 5.21**
*Considere una máquina que detecta secuencias bbb en cadenas en* $\{a, b\}^*$.

*Dada una cadena, la máquina localiza secuencias de tres b seguidas. Para la cadena aabbbaabbabbbaa, la salida sería* 000010000000100*, donde los 1 corresponden al último caracter de la secuencia bbb localizada. Las secuencias pueden solaparse, como en el caso de la cadena bbbbabbbbb. Para esta cadena, la salida debería ser* 0011000111*. El 1 en la tercera posición indica la secuencia bbb que se ubica en las posiciones 1, 2 y 3; el 1 en la cuarta posición indica la secuencia bbb que se ubica en las posiciones 2, 3 y 4; etc.*

*La máquina debe recordar los últimos caracteres leídos. En particular, debe recordar si el último caracter fue una a, o si lo último que se leyó fue una secuencia de una b, o una secuencia de dos b, o una secuencia de tres b. Estas situaciones dan lugar a cuatro estados de la máquina,* $q_0$*,* $q_1$*,* $q_2$ *y* $q_3$*, respectivamente. Las transiciones que lleven al estado* $q_3$ *darán una salida 1, mientras que las otras darán salida 0.*

*En la figura 5.14 se muestra el diagrama de transiciones para esta máquina.*

La tabla de transición para máquinas con salida debe incluir la información de la salida, además del cambio de estado. Esto puede hacerse agregando columnas correspondientes a la función de salida, por cada caracter de entrada. La tabla para el ejemplo 5.21 es

| | Transición | | Salida | |
|---|---|---|---|---|
| | a | b | a | b |
| $\rightarrow q_0$ | $q_0$ | $q_1$ | 0 | 0 |
| $q_1$ | $q_0$ | $q_2$ | 0 | 0 |
| $q_2$ | $q_0$ | $q_3$ | 0 | 1 |
| $q_3$ | $q_0$ | $q_3$ | 0 | 1 |

***Ejemplo* 5.22**
*Un tipo importante de autómata finito con salida es la llamada máquina de k-retardo, siendo k un número natural. Esta máquina toma la entrada* $a_1a_2a_3...a_{n-1}a_n$ *y da como salida la cadena* $0 \ldots 0a_1a_2...a_{n-k}$*, es decir, devuelve k ceros, y luego da los caracteres de entrada, retrasados k posiciones. En particular, para* $k = 1$*, la salida es* $0a_1a_2...a_{n-1}$.

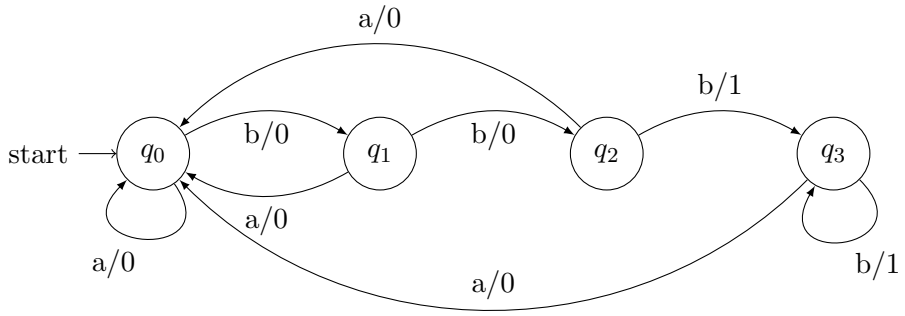



Figura 5.14: Máquina reconocedora de *bbb*

*Para $k = 1$, la máquina debe recordar cuál fue el anterior caracter leído. Si el alfabeto de entrada es $\{+, *\}$, necesita un estado para representar que el último caracter leído es +, y un estado para representar que el último caracter leído es *. Además, un estado inicial para representar que no se ha leído caracter hasta el momento. Estos estados se denotan $q_+$, $q_*$ y $q_0$ respectivamente. Entonces, cada vez que se lee un +, debe ir a $q_+$ y cada vez que se lee el caracter *, debe ir a $q_*$. Además, cuando se sale de $q_+$ con una transición, la máquina debe dar como salida +. Y cuando una transición salga del estado $q_*$, debe dar como salida *.*

*La figura 5.15 muestra el diagrama de transición. Analizando el diagrama, se puede simular el comportamiento de la máquina. Por ejemplo, para la cadena ++*+***+, la secuencia de estados y la salida se muestra a continuación:*

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ***Estado*** | $q_0$ | $q_+$ | $q_+$ | $q_*$ | $q_+$ | $q_*$ | $q_*$ | $q_*$ | $q_+$ | |
| ***Caracter*** | | + | + | * | + | * | * | * | + | |
| ***Salida*** | | *0* | + | + | * | + | * | * | * | |

*La tabla para esta máquina es*

| | *Transición* | | *Salida* | |
|---|---|---|---|---|
| | + | * | + | * |
| $\rightarrow q_0$ | $q_+$ | $q_*$ | *0* | *0* |
| $q_+$ | $q_+$ | $q_*$ | + | + |
| $q_*$ | $q_+$ | $q_*$ | * | * |

■

### Máquinas de Moore

En las máquinas de Moore la salida depende del estado en que se encuentra el autómata. Cada vez que se produce una transición, el autómata da como salida una cadena o símbolo asociado al estado al que llega.

Una manera de representarlo es con un grafo de transiciones, como antes, pero ahora en cada estado se indica la cadena o símbolo de salida.

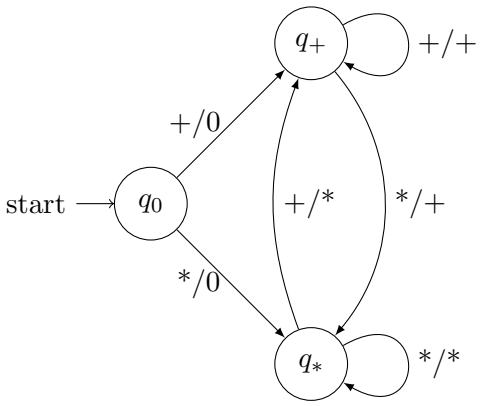



Figura 5.15: Máquina de 1-retardo

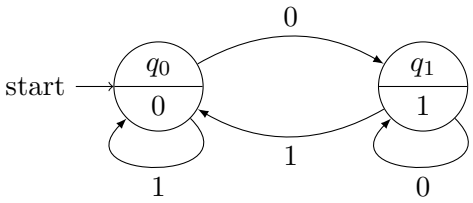



Figura 5.16: Máquina de Moore que invierte la cadena de entrada

Por ejemplo, considere el autómata de Moore de la figura 5.16. Éste recibe una cadena de 0 y 1. Cada vez que llega al estado $q_0$, da como salida un 0, y cada vez que llega al estado $q_1$ da como salida un 1.

Para analizar su comportamiento, se puede simular el paso de estados y la salida en cada paso. Al leer la cadena, 0010111, la sucesión de estados en los que se encuentra y la salida se muestran en la tabla siguiente:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Estado** | $q_0$ | $q_1$ | $q_1$ | $q_0$ | $q_1$ | $q_0$ | $q_0$ | $q_0$ |
| **Salida** | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| **Caracter** | | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

Nótese que el autómata devuelve como salida la cadena de entrada invertida, con un 0 adicional al inicio.

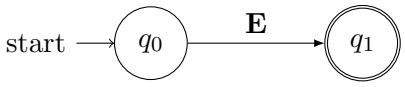
## 5.4. Expresiones regulares y autómatas

Las expresiones regulares describen lenguajes regulares algebraicamente, y los autómatas representan lenguajes procedimentalmente. Pero esencialmente el conjunto de ER y el

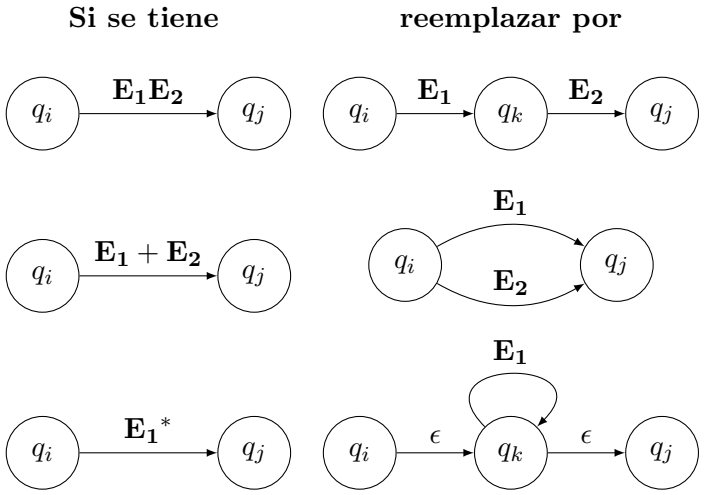
conjunto de AFN representan exactamente el mismo conjunto de lenguajes: el de los lenguajes regulares. En los ejemplos de autómatas hemos analizado el lenguaje aceptado en cada caso, y construido una ER para él. En esta sección veremos el prodecimiento inverso: dada una ER, construir un AFN que reconozca el lenguaje representado por esa ER.

La conversión de una ER en un AFN consiste en sucesivos pasos que transforman gradualmente la ER en AFN usando gráficos auxiliares, llamados gráficos de conversión. Estos gráficos son AFN que en general usan ER como etiquetas en sus arcos de transición.

Inicialmente, dada la ER **E** se plantea un gráfico inicial que es:



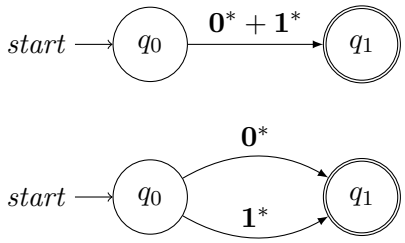

A partir de este gráfico inicial, la idea es ir transformándolo para obtener un AFN cuyas etiquetas en los arcos son caracteres del alfabeto o la palabra vacía. En cada paso se descompone/n la/s ER que etiquetan los arcos según las operaciones que las componen (suma, concatenación, clausura). Las transformaciones se dan en la tabla siguiente:

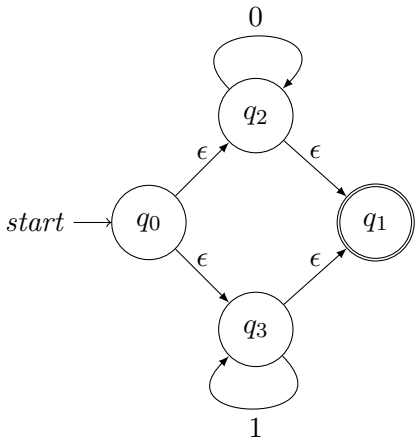



El AFN obtenido con estas transformaciones puede no ser el óptimo en términos de tamaño y/o complejidad. Es un autómata válido que reconoce el mismo lenguaje representado por la ER, que luego puede minimizarse, si se desea simplificarlo.

***Ejemplo* 5.23**

*Considere la ER* $\mathbf{0}^* + \mathbf{1}^*$*, que representa el lenguaje de las cadenas que son una secuencia de 0 o una secuencia de 1. Los pasos para obtener un AFN que acepte ese lenguaje son:*



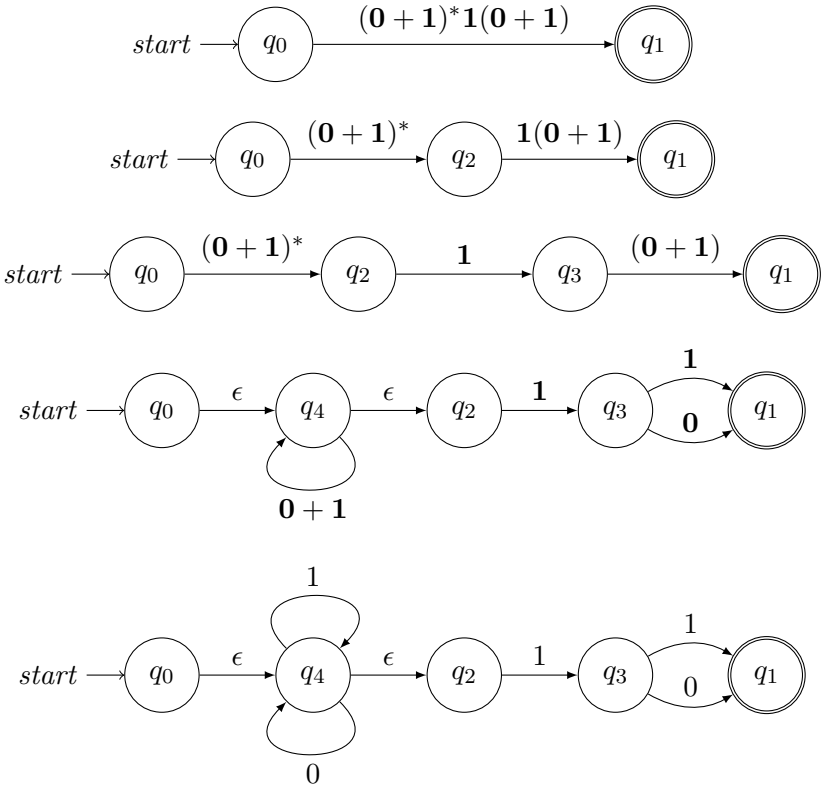



■

***Ejemplo* 5.24**

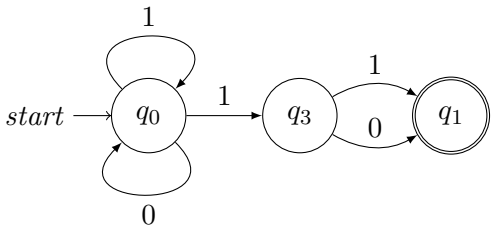
*Se propone construir un autómata finito que acepte el lenguaje representado por la ER* $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$*. Este lenguaje contiene las cadenas de 0 y 1 tales que el anteúltimo caracter es un 1.*

*El AFN correspondiente se obtiene con los siguientes gráficos de conversión.*




*El último gráfico es el diagrama de transición del autómata buscado.*

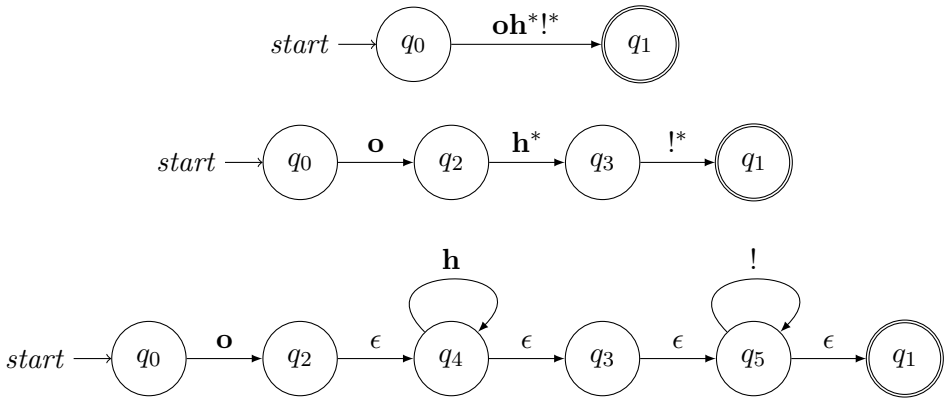
*Nótese que, eliminando las transiciones espontáneas (con etiquetas* $\epsilon$*), se obtiene el autómata equivalente:*



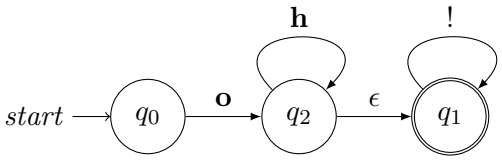

■

***Ejemplo* 5.25**
*La ER* $\mathbf{oh}^*!^*$ *representa el lenguaje de cadenas formadas por una o, una cadena (posiblemente vacía) de h y una cadena (posiblemente vacía) de !. No confundir con* $(\mathbf{oh})^*!^*$*, que representa el lenguaje de repeticiones de oh, seguido de repeticiones de !.*

*La ER da lugar al AFN que se obtiene con los siguientes gráficos de conversión.*




*Este AFN es equivalente al autómata más simple mostrado a continuación:*




■

Si bien existen técnicas de minimización de autómatas, que aplican reglas específicas, no se desarrollarán en este libro. Las simplificaciones realizadas en los dos últimos ejemplos surgen de la observación de las transiciones espontáneas, y no aplican ninguna regla general.

## 5.5. Gramáticas

### 5.5.1. Definición

Los lenguajes formales también pueden representarse con una gramática formal. Una gramática es básicamente, un conjunto de reglas de formación de cadenas válidas en un lenguaje. Una regla es una expresión de la forma $\alpha \to \beta$, donde $\alpha$ y $\beta$ son cadenas de

símbolos del alfabeto considerado (llamados símbolos terminales), o variables (símbolos no terminales).

La gramática del idioma español nos da reglas para la formación de frases válidas. Por ejemplo, podemos pensar en las siguientes reglas para formar frases:

frase → sujeto + predicado
sujeto → artículo+sustantivo+adjetivo
sujeto → artículo+sustantivo
predicado → verbo
predicado → verbo+adverbio
artículo → el
artículo → la
sustantivo → libro
sustantivo → película
adjetivo → nuevo
verbo → aburre
verbo → entretiene

Estas reglas indican que una frase está formada por sujeto y predicado. Sujeto puede estar formado por artículo-sustantivo-adjetivo o artículo-sustantivo. Predicado puede formarse con un verbo o con un verbo seguido de adverbio. Un artículo es `el` o `la`, sustantivo puede ser `libro` o `película`, adjetivo es `nuevo` y verbo es `entretiene` o `aburre`. En este ejemplo, frase, sujeto, predicado, artículo, sustantivo, adjetivo, verbo y adverbio son símbolos no terminales o variables (también llamadas categorías sintácticas), mientras que `el`, `la`, `libro`, `película`, `nuevo`, `aburre` y `entretiene` son símbolos terminales.

La aplicación de la regla $\alpha \to \beta$ a la cadena $\mathbf{u}\alpha\mathbf{v}$, genera la cadena $\mathbf{u}\beta\mathbf{v}$. Es regla de reemplazo: si aparece la cadena $\alpha$ que está a la izquierda de la regla, se reemplaza por la cadena de la derecha $\beta$.

Con las reglas dadas para la construcción de frases en español, comenzando con $frase$, se puede generar la siguiente frase válida:

frase
sujeto+predicado
artículo + sustantivo + predicado
artículo + sustantivo + verbo
el + libro + entretiene

Formalmente, una GRAMÁTICA consta de un alfabeto de constantes o símbolos terminales $\Sigma$, un conjunto de variables o símbolos no terminales $V$, un símbolo inicial $S \in V$, y un conjunto de reglas de la forma $\alpha \to \beta$, donde $\alpha$ y $\beta$ son cadenas de símbolos terminales y/o no terminales, y además, $\alpha$ debe contener al menos un símbolo no terminal.

Para evitar confusión, se usarán letras mayúsculas para las variables y letras minúsculas, dígitos o símbolos para las constantes. Las letras en negrita representan cadenas.

Se dice que la cadena $\mathbf{u}\beta\mathbf{v}$ DERIVA de $\mathbf{u}\alpha\mathbf{v}$ si existe una regla de la forma $\alpha \to \beta$. Se denota $\mathbf{u}\alpha\mathbf{v} \Rightarrow \mathbf{u}\beta\mathbf{v}$. Una cadena $\mathbf{w} \in \Sigma^*$ (es decir, formada solamente por símbolos terminales) ES DERIVABLE a partir de una gramática si existen cadenas $\mathbf{a_1}$, $\mathbf{a_2}$, ..., $\mathbf{a_n}$,

tales que $\mathbf{a_i}$ derive de $\mathbf{a_{i-1}}$ para i=2, 3, ..., n, $\mathbf{a_1}$ derive de $S$ (el símbolo inicial de la gramática) y $\mathbf{w}$ derive de $\mathbf{a_n}$. En notación compacta: $S \Rightarrow \mathbf{a_1} \Rightarrow \mathbf{a_2} \Rightarrow ... \Rightarrow \mathbf{a_n} \Rightarrow \mathbf{w}$.

El LENGUAJE GENERADO POR UNA GRAMÁTICA es el conjunto de cadenas derivables a partir de las reglas de la gramática.

Las gramáticas se clasifican de acuerdo al tipo de cadenas a la izquierda y derecha de las reglas. Esta clasificación de gramáticas impone una jerarquía en el tipo de lenguaje que cada una puede generar. La clasificación de lenguajes que se dará fue propuesta por Chomsky.

- La gramática de tipo 0 no tiene restricciones en las reglas. Generan el tipo de lenguajes denominado RECURSIVAMENTE ENUMERABLES.
- La gramática de tipo 1 tiene reglas de la forma $\mathbf{u}\alpha\mathbf{v} \rightarrow \mathbf{u}\beta\mathbf{v}$. Generan lenguajes denominados DEPENDIENTES DEL CONTEXTO.
- Una gramática de tipo 2 tiene reglas $\alpha \rightarrow \beta$ tal que $\alpha$ es un único elemento no terminal. Estas gramáticas generan lenguajes INDEPENDIENTES DEL CONTEXTO.
- Una gramática de tipo 3 tiene reglas de la forma $N \rightarrow aM$ o $N \rightarrow a$, donde $N$ y $M$ son símbolos no terminales, y $a$ es un símbolo terminal. Estas gramáticas describen los lenguajes REGULARES.

En la clasificación dada, se imponen condiciones más restrictivas a las reglas de las gramáticas de mayor tipo. Así, toda gramática de tipo 3 es también de tipo 2. De este modo, todo lenguaje regular es también independiente del contexto. Pero, por supuesto, hay lenguajes independientes del contexto que no son regulares.

***Ejemplo* 5.26**
*Una gramática para generar el lenguaje* $\mathbf{0^n1^n}$ *tiene las reglas:*

*1:* $S \rightarrow 0S1$
*2:* $S \rightarrow \epsilon$

*Veamos ejemplos de cadenas derivadas con estas reglas.*

| ***Cadena*** | ***Derivación*** |
|---|---|
| *0011* | $S \Rightarrow_1 0S1 \Rightarrow_1 00S11 \Rightarrow_2 0011$ |
| *000111* | $S \Rightarrow_1 0S1 \Rightarrow_1 00S11 \Rightarrow_1 000S111 \Rightarrow_2 000111$ |

*(El subíndice en las flechas de derivación indica la regla usada en ese paso).*

*Debido a la forma de las reglas gramaticales que lo generan, este lenguaje es un lenguaje independiente del contexto.* ■

***Ejemplo* 5.27**
*Considere la gramática de tipo 2 que tiene alfabeto de constantes* $\Sigma = \{(, )\}$*, variable S, y reglas:*

*1:* $S \rightarrow SS$
*2:* $S \rightarrow (S)$
*3:* $S \rightarrow ()$

*Con estas reglas, se pueden derivar las cadenas de paréntesis bien balanceados (lenguaje independiente del contexto). Algunas derivaciones son:*

| ***Cadena*** | ***Derivación*** |
|---|---|
| *()((()))* | $S \Rightarrow_1 SS \Rightarrow_2 S(S) \Rightarrow_2 S((S)) \Rightarrow_3 S((())) \Rightarrow_3 ()((()))$ |
| *(((())())())* | $S \Rightarrow_2 (S) \Rightarrow_1 (SS) \Rightarrow_3 (S()) \Rightarrow_2 ((S)()) \Rightarrow_1 ((SS)()) \Rightarrow_3$ $((S())()) \Rightarrow_3 (((())())())$ |
| *(())* | $S \Rightarrow_2 (S) \Rightarrow_3 (())$ |

■

Muchas veces en una gramática hay varias reglas que tienen la misma cadena a la izquierda. Una notación compacta para el conjunto de reglas $X \to Y$, $X \to Z$ y $X \to T$ es $X \to Y|Z|T$.

***Ejemplo* 5.28**
*En ciertos lenguajes de programación, los identificadores son cadenas de caracteres alfanuméricos que deben comenzar con una letra. El conjunto de todos los identificadores válidos se pueden generar con las siguientes reglas:*

*1:* $IDENTIF \to LETRA$
*2:* $IDENTIF \to IDENTIF\ DIGITO$
*3:* $IDENTIF \to IDENTIF\ LETRA$
*4:* $LETRA \to a|b|c...y|z$
*5:* $DIGITO \to 0|1|2..,8|9$

*IDENTIF, LETRA, DIGITO son símbolos no terminales, y a, b, c,..., z, 0, 1, ..., 9 son símbolos terminales.*

*Con estas reglas se pueden derivar los identificadores válidos. Por ejemplo, para derivar el identificador* ***punto0****, se utilizan las siguientes derivaciones:* $IDENTIF \Rightarrow_2 IDENTIF\ DIGITO \Rightarrow_5 IDENTIF\ 0 \Rightarrow_3 IDENTIF\ LETRA\ \mathbf{0} \Rightarrow_4 IDENTIF\ \mathbf{o0} \Rightarrow_3 IDENTIF\ LETRA\ \mathbf{o0} \Rightarrow_4 IDENTIF\ \mathbf{to0} \Rightarrow_3 IDENTIF\ LETRA\ \mathbf{to0} \Rightarrow_4 IDENTIF\ \mathbf{nto0} \Rightarrow_3 IDENTIF\ LETRA\ \mathbf{nto0} \Rightarrow_4 IDENTIF\ \mathbf{unto0} \Rightarrow_1 LETRA\ \mathbf{unto0} \Rightarrow_4 \mathbf{punto0}$ ■

### 5.5.2. Gramáticas regulares

Las gramáticas regulares son las del tipo 3. Como se dijo, generan los lenguajes regulares. Surge entonces aquí una nueva forma de describir un lenguaje regular, además de las expresiones regulares y los AFD/AFN.

***Ejemplo* 5.29**
*Considere la gramática regular con símbolos no terminales S y A, y símbolos terminales 0 y 1, cuyas reglas son:*

*1:* $S \to 1A$
*2:a,b,c* $A \to 0A|1A|1$

*En el paso inicial de toda derivación se debe aplicar la regla 1, que transforma el símbolo inicial S en 1A. Las otras reglas permiten modificar esta cadena, reemplazando la variable A por 0A, 1A o 1. Es decir, el 1 que apareció antes de A en la primer derivación,*

*permanecerá inalterado en todo el proceso. Esto indica que todas las cadenas del lenguaje generado por estas reglas comienzan con 1.*

*Toda derivación termina cuando ya no hay variables en la cadena que se está generando. En este caso, la derivación termina únicamente con la aplicación de la regla 2.c. Entonces, toda cadena derivada terminará con 1.*

*Resumiendo, todas las palabras del lenguaje generado por esta gramática son cadenas de 0 y 1 que comienzan y terminan con 1. En ER se escribe:* $\mathbf{1(0+1)^*1}$ ■

***Ejemplo* 5.30**

*Considere la gramática con símbolos no terminales S, A y B, símbolos terminales* $\Sigma = \{0,1\}$*, y las siguientes reglas:*

$$\begin{array}{ll} \textit{1:a,b,c} & S \to 0A|1B|1 \\ \textit{2:a,b} & A \to 0A|1B \\ \textit{3:a,b,c,d} & B \to 0B|1B|0|1 \end{array}$$

*Estas reglas permiten obtener las siguientes cadenas:*

| ***Cadena*** | ***Derivación*** |
|---|---|
| *0010* | $S \Rightarrow_{1a} 0A \Rightarrow_{2a} 00A \Rightarrow_{2b} 001B \Rightarrow_{3c} 0010$ |
| *01011* | $S \Rightarrow_{1a} 0A \Rightarrow_{2b} 01B \Rightarrow_{3a} 010B \Rightarrow_{3b} 0101B \Rightarrow_{3d} 01011$ |
| *000110* | $S \Rightarrow_{1a} 0A \Rightarrow_{2a} 00A \Rightarrow_{2a} 000A \Rightarrow_{2b} 0001B \Rightarrow_{3b} 00011B \Rightarrow_{3c} 000110$ |
| *1* | $S \Rightarrow_{1c} 1$ |
| *1101* | $S \Rightarrow_{1b} 1B \Rightarrow_{3b} 11B \Rightarrow_{3a} 110B \Rightarrow_{3d} 1101$ |

*El lenguaje generado es* $\mathbf{0^*1(0+1)^*}$*. Está formado por cadenas de 0 y 1 que consisten en una cadena de 0 (posiblemente vacía), seguido de un 1, seguido de una cadena de 0 y 1.*

*La variable A indica que se están agregando 0 antes del primer 1. Cuando aparece el primer 1, se usa la variable B. Ésta indica que ya se generó el primer 1, y que luego puede aparecer cualquier caracter 0 ó 1 hasta el final de la palabra generada.* ■

***Ejemplo* 5.31**

*Se propone hallar una gramática con símbolos constantes* $\Sigma = \{0,1,2\}$*, que genere el lenguaje de las palabras que contengan la subcadena 22.*

*Las variables en una gramática denotan determinadas situaciones o estados. Podemos pensar en este caso en las siguientes variables:*

*S: representa la situación de no haber producido ningún 2.*

*X: representa la situación de haber producido un 2.*

*Y: representa la situación de ya haber generado la subcadena 22.*

*Las reglas para esta gramática, con las variables así definidas, son:*

$$\begin{array}{ll} \textit{1:} & S \to 0S|1S|2X \\ \textit{2:} & X \to 0S|1S|2Y|2 \\ \textit{3:} & Y \to 0Y|1Y|2Y|0|1|2 \end{array}$$

*La variable S puede reemplazarse por 0S o 1S. En ambos casos, sigue estando la variable S porque aún no se recibió un 2. Pero S también puede reemplazarse por 2X. En esta regla se agrega un 2 a la cadena que se está derivando, y por esto aparece la variable X.*

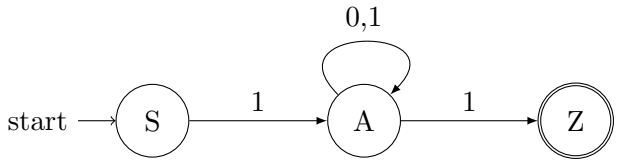



Figura 5.17: AFN para la gramática del ejemplo 5.29

*X puede reemplazarse por 0S o 1S. En ambos casos, al agregarse un caracter que no es 2 a la cadena siendo derivada, se debe volver a la variable S. Pero si se agrega un 2, aparece la variable Y, que denota que ya se tiene la subcadena 22.*

*Una vez que se obtiene la variable Y en la derivación, ya no aparecen S o X, la cadena se termina de derivar cuando Y se reemplaza por un único símbolo terminal.* ■

### 5.5.3. Conversión entre gramáticas regulares y AF

Dado que las gramáticas regulares generan los mismos lenguajes que reconocen los autómatas finitos, existe una equivalencia entre estas dos formas de describir un lenguaje. De hecho, existe un procedimiento para, dada una gramática regular, generar un autómata finito que reconozca el lenguaje generado, y viceversa.

El procedimiento consiste en asociar los símbolos no terminales de la gramática a los estados del autómata, más un estado de aceptación $Z$. Las transiciones surgen de las reglas: dada una regla de la forma $A \to xB$, ésta se asocia a una transición del estado $A$ al estado $B$, con el caracter (etiqueta en el arco) $x$. Y las reglas de la forma $A \to x$, se asocian a una transición desde el estado $A$ al estado de aceptación $Z$, con el caracter (etiqueta) $x$.

***Ejemplo* 5.32**
*Se construirá un AFN para el lenguaje generado por la gramática del ejemplo 5.29. Las variables son S y A. Entonces el autómata tendrá dos estados correspondientes a estas variables, que los llamaremos con los mismos nombres, S y A, más un estado de aceptación Z. La primera regla indica una transición del estado S al estado A con el caracter 1. Las otras reglas indican las transiciones de A a A con los caracteres 0 y 1, y una transición de A a Z con el caracter 1. El AFN se muestra en la figura 5.17.*

*Pueden hacerse simulaciones del funcionamiento del AFN para verificar que efectivamente acepta las cadenas que empiezan y terminan con 1, y sólo esas. Se deja esta comprobación al lector.* ■

***Ejemplo* 5.33**
*Dada la gramática del ejemplo 5.30, un AFN que acepte el lenguaje generado por ella tiene tres estados correspondientes a las variables S, A y B respectivamente, y un estado de aceptación Z. Siguiendo las reglas de la gramática, desde el estado S salen transiciones hacia A con el caracter 0, hacia B con el caracter 1 y hacia Z con el caracter 1. Desde el estado A salen transiciones hacia A con caracter 0 y hacia B con caracter 1. Desde el estado B las transiciones son hacia B con etiquetas 0 y 1, y hacia Z con etiquetas 0 y 1.*

*El diagrama del AFN se observa en la figura 5.18.*

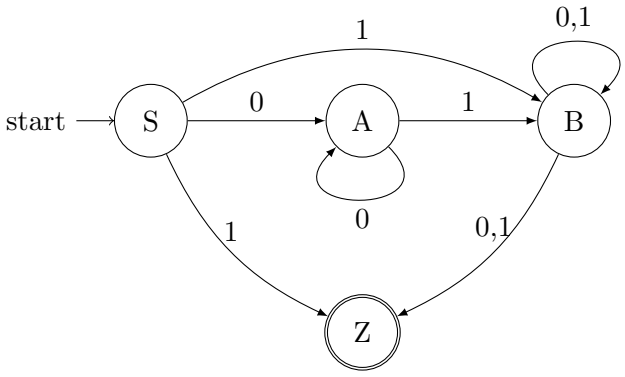



Figura 5.18: AFN para la gramática del ejemplo 5.30

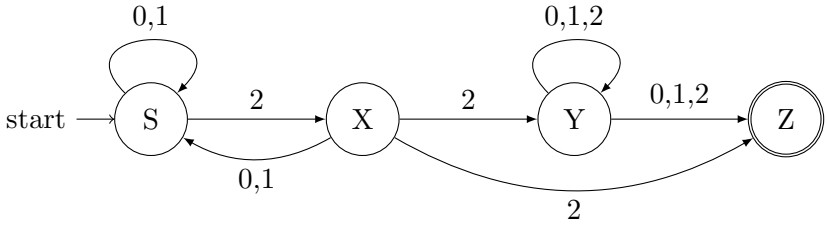



Figura 5.19: AFN para el lenguaje de cadenas de 0,1,2 que contienen la subcadena 22 (ejemplo 5.31)

***Ejemplo* 5.34**
*Considere la gramática del ejemplo 5.31, que genera el lenguaje de cadenas que tienen la subcadena 22, es decir, el lenguaje* $(\mathbf{0}+\mathbf{1}+\mathbf{2})^*\mathbf{22}(\mathbf{0}+\mathbf{1}+\mathbf{2})^*$*. A esta altura es fácil obtener el AFN que acepte el mismo lenguaje, y se muestra en la figura 5.19.* ■

Para convertir un AFD en una gramática que genere el lenguaje aceptado, el proceso es similar. La gramática tendrá una variable por cada estado, y la transición del estado $q_i$ al estado $q_j$ con el caracter $x$ se convierte en la regla $Q_i \to xQ_j$, donde $Q_i$ y $Q_j$ son las variables correspondientes a los estados mencionados. Además, para cada estado de aceptación $q_f$, si existe una transición de $q_k$ al estado $q_f$ con caracter z, se convierte en la regla $Q_k \to z$ (este tipo de reglas gramaticales, que no tienen variables a la derecha, son las que dan por terminada la derivación de una cadena).

***Ejemplo* 5.35**
*Considere el AFD de la figura 5.20. Este autómata reconoce las cadenas del alfabeto* $\{+, *\}$ *tienen tres * seguidos.*

*La gramática equivalente tiene cuatro variables* $Q_0$*,* $Q_1$*,* $Q_2$ *y* $Q_3$*. Y las reglas, siguiendo las transiciones, son:*

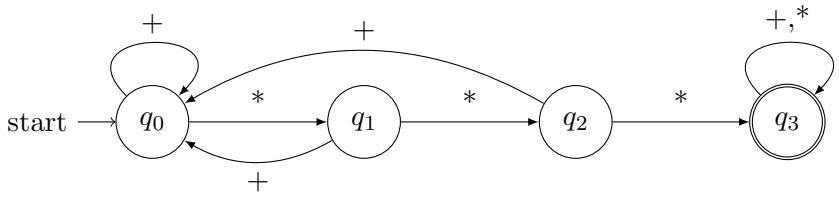



Figura 5.20: AFD del ejemplo 5.35

$$
\begin{aligned}
Q_0 &\to *Q_1 | + Q_0 \\
Q_1 &\to +Q_0 | * Q_2 \\
Q_2 &\to +Q_0 | * Q_3 \\
Q_3 &\to +Q_3 | * Q_3 | * | +
\end{aligned}
$$

■

***Ejemplo* 5.36**
*El autómata de la figura 5.8a reconoce el lenguaje de cadenas binarias que comienzan con 00. Las reglas para una gramática que genere ese lenguaje son:*

$$
\begin{aligned}
Q_0 &\to 0Q_1 | 1Q_3 \\
Q_1 &\to 0Q_2 | 1Q_3 \\
Q_2 &\to 0Q_2 | 1Q_2 | 0 | 1 \\
Q_3 &\to 0Q_3 | 1Q_3
\end{aligned}
$$

■

### 5.5.4. Gramáticas independientes de contexto

Una gramática de tipo 2 también se denominan gramática independiente de contexto (GIC). En una GIC las reglas son tales que a la izquierda sólo tienen una variable.

Este tipo de gramática es importante porque permite describir la sintaxis de los lenguajes de programación. Existe una forma mecánica de transformar la descripción de una sintaxis dada por una GIC en un analizador sintáctico. Si bien hay partes de un lenguaje de programación que tienen estructura de lenguaje regular (palabras reservadas, identificadores válidos, etc); hay aspectos que escapan de esta caracterización. Por ejemplo, los paréntesis (o llaves, o corchetes) balanceados. O la estructura de las sentencias condicionales if y if - else. Por otra parte, los lenguajes de marcado (como HTML) que utilizan algunas etiquetas de apertura y cierre (<x> y </x>) para determinar el formato del contenido entre las etiquetas; o etiquetas sólo de apertura (<x>). Y esta estructura no es regular, sino que forma un lenguaje independiente de contexto.

***Ejemplo* 5.37**
*Se da un ejemplo de una GIC para generar expresiones artiméticas válidas con números naturales, con sumas y producto:*

*1:* $S \to I$
*2:* $S \to S + S$
*3:* $S \to S \cdot S$
*4:* $S \to (S)$
*5:* $I \to I0|I1|...|I9$
*6:* $I \to 0|1|...|9$

*La variable S representa una expresión, y la variable I indica un número. Las constantes o símbolos terminales son los dígitos 0,1,...,9, y los signos +, ·, ( y ). La expresión aritmética* $(23 + 4) \cdot 0$ *se deriva con los siguientes pasos:*

$S \Rightarrow_3 S \cdot S \Rightarrow_4 (S) \cdot S \Rightarrow_2 (S + S) \cdot S \Rightarrow_1 (S + S) \cdot I \Rightarrow_6 (S + S) \cdot 0 \Rightarrow_1 (I + S) \cdot 0 \Rightarrow_1 (I + I) \cdot 0 \Rightarrow_5 (I3 + I) \cdot 0 \Rightarrow_6 (23 + I) \cdot 0 \Rightarrow_6 (23 + 4) \cdot 0$ ■

***Ejemplo* 5.38**
*Considere el lenguaje de las cadenas palíndromas, aquellas que se leen igual al derecho y al revés. Para simplificar, considere que los caracteres del alfabeto son a y b.*

*Las reglas de una gramática para este lenguaje pueden formar las cadenas agregando el mismo caracter adelante y atrás de la cadena. Es decir,* $S \to aSa$ *y* $S \to bSb$. *Estas dos reglas no son suficientes, porque se necesita al menos una regla que no deje variables, como por ejemplo* $S \to \epsilon$. *Con estas tres reglas se pueden generar todos los palíndromos de longitud par. Se necesitan reglas que permitan formar los palíndromos de longitud impar, y éstas son* $S \to a$ *(si el caracter central es a) y* $S \to b$ *(si el caracter central es b). Ahora sí, la gramática está completa, y consta de las reglas:*

*1:* $S \to aSa$
*2:* $S \to bSb$
*3:* $S \to \epsilon$ ■
*4:* $S \to a$
*5:* $S \to b$

***Ejemplo* 5.39**
*Una GIC para generar programas con sentencias* if *y* if else *se puede plantear de la siguiente manera. Sean las variables o caracteres no terminales* $V = \{ S, STAT \}$, *y las constantes o caracteres terminales* $\Sigma = \{$ *if, then, else, cond, otras* $\}$ *y las reglas:*

$S \to STAT$
$STAT \to if\ cond\ then\ STAT$
$STAT \to if\ cond\ then\ STAT\ else\ STAT$ ■
$STAT \to STAT\ STAT$
$STAT \to otras$

***Ejemplo* 5.40**
*Una versión muy simplificada de la estructura del lenguaje HTML puede ser descrita con las reglas:*

*1:* $Doc \rightarrow \epsilon \mid Elem\ Doc$
*2:* $Elem \rightarrow Texto \mid <EM>Doc</EM> \mid <P>Doc \mid <OL>Lista</OL>$
*3:* $Lista \rightarrow \epsilon \mid ElemLista\ Lista$
*4:* $ElemLista \rightarrow <LI>Doc$
*5:* $Texto \rightarrow \epsilon \mid CarTexto$
*6:* $Car \rightarrow a|b|...|z|A|B|...|Z|...$

■

A continuación se muestra una derivación de un sencillísimo documento html:
Doc $\Rightarrow_1$ Elem Doc $\Rightarrow_1$ ... $\Rightarrow_1$
Elem Elem Elem $\Rightarrow_2$
Elem <EM>Doc</EM> Elem $\Rightarrow_1$ ...$\Rightarrow_1$
Elem <EM>Elem</EM> Elem $\Rightarrow_2$ ... $\Rightarrow_2$
Texto <EM>Texto</EM> <OL>Lista</OL> $\Rightarrow_3$ ... $\Rightarrow_3$
Texto <EM>Texto</EM> <OL>ElemLista ElemLista </OL> $\Rightarrow_4$ ... $\Rightarrow_4$
Texto <EM>Texto</EM> <OL> <LI>Doc <LI>Doc </OL> $\Rightarrow_1$ ... $\Rightarrow_1$
Texto <EM>Texto</EM> <OL> <LI>Elem <LI>Elem </OL> $\Rightarrow_2$ ... $\Rightarrow_2$
Texto <EM>Texto</EM> <OL> <LI>Texto <LI>Texto </OL> $\Rightarrow_{5,6}$...$\Rightarrow_{5,6}$
No <EM>Olvidar</EM> <OL> <LI>Lustrar zapatos <LI>Ajustar corbata </OL>

Se usó la notación $\Rightarrow_i$...$\Rightarrow_i$ para denotar más de un paso de aplicación de la regla i.

### 5.5.5. Árboles de derivación

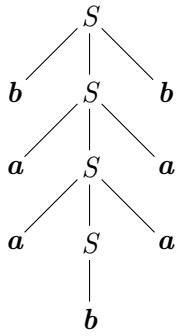
Las derivaciones de cadenas usando reglas de una gramática pueden representarse en un grafo con estructura de árbol. Los árboles de derivación se utilizan en los compiladores para representar la estructura sintáctica del código fuente de un programa. Esta representación facilita la traducción a código ejecutable, acción que puede ser desarrollada por funciones recursivas.

Los árboles de derivación de una palabra para una gramática tienen las siguientes características:

- La raíz está etiquetada con el símbolo inicial (S)
- Cada nodo de ramificación está etiquetado con una variable o símbolo no terminal
- Cada hoja tiene por etiqueta una constante o símbolo terminal o la cadena vacía $\epsilon$.
- Si un nodo de ramificación tiene etiqueta $A$, y sus hijos tienen etiquetas $x_1$, $x_2$, ..., $x_m$ (enumerados de izquierda a derecha), donde $x_i$ puede ser terminal o no terminal, entonces $A \rightarrow x_1x_2...x_m$ es una regla de la gramática.

El árbol representa la cadena de símbolos terminales obtenida al concatenar las hojas de izquierda a derecha.

***Ejemplo* 5.41**
*La cadena baabaab es un palíndromo que puede generarse con la gramática del ejemplo 5.38. El árbol de derivación de esta cadena es:*

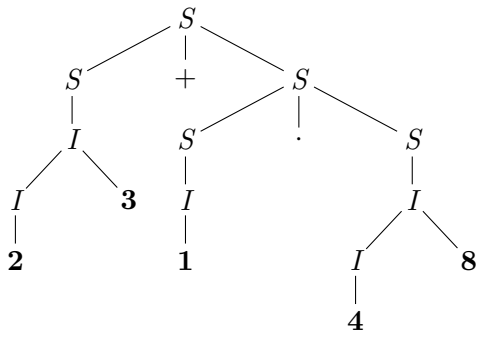



*La primera ramificación representa la aplicación de la regla $S \to bSb$, las dos siguientes corresponden a la regla $S \to aSa$, y la última ramificación se corresponde con la regla $S \to b$.*

*Leyendo las hojas del árbol de izquierda a derecha se obtiene baabaab.* ■

***Ejemplo* 5.42**

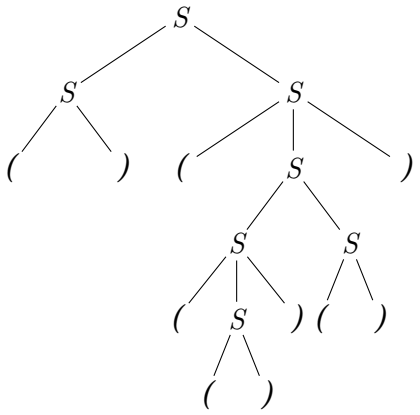
*La gramática del ejemplo 5.37 permite generar la cadena 23+1·48. El árbol de derivación para esta cadena es:*




■

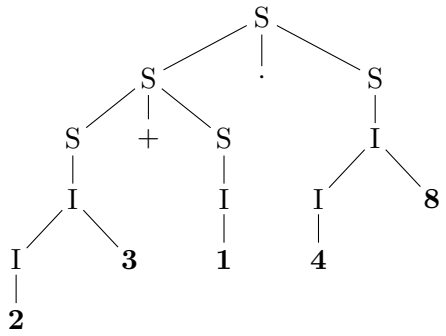
***Ejemplo* 5.43**

*La gramática dada en el ejemplo 5.27 genera las cadenas de paréntesis balanceados. Las reglas permiten derivar la cadena ()((())()). Su árbol de derivación es:*




■

Lamentablemente, el árbol de derivación no es único para una cadena dada, depende del orden en que se apliquen las reglas. Por ejemplo, la cadena 23+1·48, generada con la gramática del ejemplo 5.37, admite el árbol de derivación mostrado en el ejemplo 5.42 y también admite el siguiente árbol:




Cuando esto ocurre se dice que la gramática es ambigua. Existen técnicas para eliminar la ambigüedad de una gramática. Pero aún así hay lenguajes independientes de contexto para los que no existe una gramática no ambigua.

## 5.6. Autómatas a pila

Las gramáticas independientes de contexto permiten generar lenguajes no regulares. Un lenguaje es regular si y sólo si es aceptado por un AFD, por lo que ciertos lenguajes generados por GIC no pueden reconocerse con un AFD.

La limitación de los AFD es que la única forma de recordar lo que recibió hasta un determinado momento es el estado en que se encuentra. Esto es un obstáculo para reconocer, por ejemplo, el lenguaje de las cadenas palíndromas, ya que hasta leer la primera mitad de la palabra, debe recordar todos los caracteres leídos para compararlo con la segunda mitad.

Una forma de pensar un autómata más potente, es añadirle un dispositivo de memoria, llamado pila. La pila tiene la capacidad de almacenar caracteres (que pueden ser caracteres distintos a los del alfabeto de entrada). A los autómatas con este dispositivo de memoria se los llama AUTÓMATAS A PILA (AP).

La pila funciona de manera que el último caracter ingresado es el primero en salir (LIFO: last-in-first-out). Las transiciones de este tipo de autómatas, además de indicar el paso de estado con cada caracter leído, indica el cambio en el contenido de la pila: qué caracteres se agregan a la pila y qué caracteres se eliminan de ella. La notación usada es la siguiente: como etiqueta en cada transición se usa $a/\alpha/\beta$, donde $a$ es el caracter leído, $\alpha$ es lo que se elimina del tope de la pila, y $\beta$ es lo que se añade a la pila (cualquiera de estas tres entidades puedan ser la cadena vacía, $\epsilon$).

Al inicio, la pila está vacía y el autómata se encuentra en el estado inicial. A medida que lee los caracteres de la cadena de entrada, cambia de estado según lo indican las transiciones, y modifica el contenido de la pila, también siguiendo las transiciones. Una palabra es aceptada por un AP si el autómata puede terminar de leerla, y en tal caso, se encuentra en un estado final y además, la pila está vacía.

***Ejemplo* 5.44**
*Un AP para el lenguaje* $\{0^n 1^n, n \geqslant 1\}$ *es el que se muestra en la figura 5.21. El AP tiene dos estados.* $q_0$ *representa la situación de estar leyendo caracteres 0, y* $q_1$ *representa la situación de leer caracteres 1.*

*Al inicio, el AP se encuentra en* $q_0$*. Si recibe un 0, no saca nada de la pila, y agrega un caracter 0 a la misma. Si estando en* $q_0$ *recibe un 1, pasa al estado* $q_1$*, y elimina un 0 del tope de la pila. Mientras esté en* $q_1$ *y siga recibiendo caracteres 1, va eliminado caracteres 0 de la pila. Si estando en* $q_1$ *recibe un 0, no lo puede leer (no hay transición definida para este caso), y el autómata se detiene sin haber podido terminar la palabra. En cambio, mientras reciba 1, y haya caracteres 0 para sacar de la pila, el autómata continúa en ese estado. Si llega al final de la palabra y la pila esta vacía (se consumieron todos los 0 agregados mientras estaba en* $q_0$*), la cadena es aceptada, porque* $q_1$ *es estado de aceptación.*

*Estudiemos el funcionamiento del AP para la cadena 000111. El cambio de estado y el contenido de la pila (entre paréntesis, al lado del estado) se muestran en el siguiente diagrama:*

$$q_0(\epsilon) \xrightarrow{0} q_0(0) \xrightarrow{0} q_0(00) \xrightarrow{0} q_0(000) \xrightarrow{1} q_1(00) \xrightarrow{1} q_1(0) \xrightarrow{1} q_1(\epsilon)$$

*Se observa que al terminar de leer la cadena, está en el estado de aceptación, y la pila se encuentra vacía. Por lo tanto, esta cadena es aceptada por el AP.*

*Si la cadena de entrada es 00111, el AP funciona como lo muestra el diagrama:*

$$q_0(\epsilon) \xrightarrow{0} q_0(0) \xrightarrow{0} q_0(00) \xrightarrow{1} q_1(0) \xrightarrow{1} q_1(\epsilon) \xrightarrow{1} X$$

*Al leer el segundo 1, la pila queda vacía, y no puede ejecutar la siguiente transición al leer el tercer 1, que implicaría sacar un 0 de la pila. El autómata no puede terminar de leer la cadena, luego, no es aceptada.*

*Con la cadena 010110, el autómata ejecuta las transiciones indicadas en el diagrama:*

$$q_0(\epsilon) \xrightarrow{0} q_0(0) \xrightarrow{1} q_1(\epsilon) \xrightarrow{0} X$$

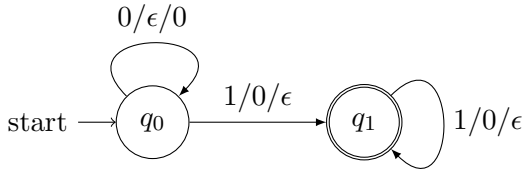



Figura 5.21: AP del ejemplo 5.44

*Nuevamente, en este caso, no puede terminar de leer la palabra, porque no hay transición definida desde $q_1$ con el caracter 0.*

*Y si la cadena de entrada es 00011:*

$$q_0(\epsilon) \xrightarrow{0} q_0(0) \xrightarrow{0} q_0(00) \xrightarrow{0} q_0(000) \xrightarrow{1} q_1(00) \xrightarrow{1} q_0(0)$$

■

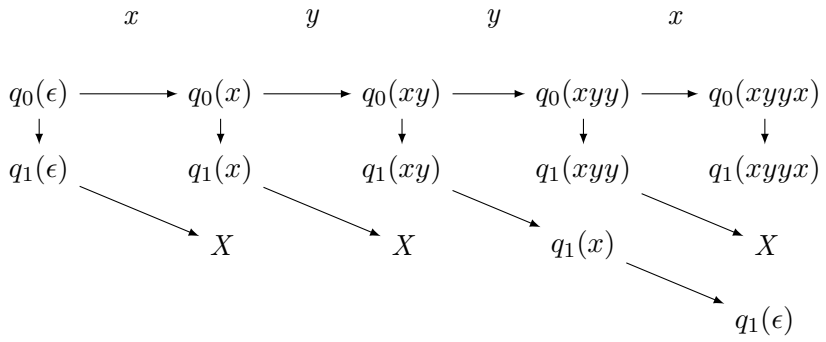
*El autómata termina de leer la cadena, se encuentra en un estado de aceptación, pero la pila no está vacía, entonces no la acepta.*

***Ejemplo* 5.45**

*Un AP que reconozca el lenguaje de los palíndromos de longitud par, con los caracteres x e y se muestra en la figura 5.22.*

*Cada vez que recibe un caracter en la primera mitad de la palabra, lo guarda en el tope de la pila. La transición que cambia de estado sin consumir caracter se supone que se ejecuta en la mitad de la palabra. Al recibir un caracter de la segunda mitad de la cadena, elimina el caracter del tope de la pila, que debe ser el mismo caracter recibido.*

*Para la cadena xyyx, el funcionamiento es como sigue:*




*Hay tres trayectorias que se bloquean porque la transición correspondiente indica eliminar un determinado caracter de la lista, pero este caracter no está disponible en el tope de la pila. Estos bloqueos se representan con las X.*

*Al terminar de leer la palabra existen tres trayectorias no bloqueadas, una termina en el estado $q_0$, que no es de aceptación, y la otra termina en el estado $q_1$, que es estado de aceptación, pero la pila no está vacía, y la tercera trayectoria termina en el estado de aceptación $q_1$ y además la pila se encuentra vacía en este punto. Por lo tanto, la cadena es aceptada por el AP.*

*Para la cadena xyx, el funcionamiento es como sigue:*

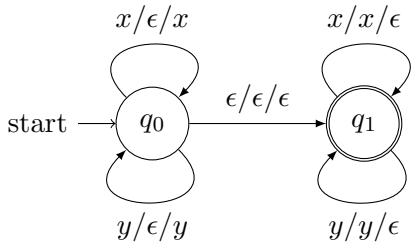
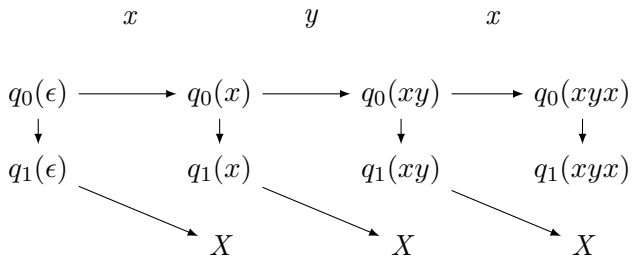



Figura 5.22: AP del ejemplo 5.45




*Al terminar de leer la cadena, las dos trayectorias no bloquedas muestran que la pila no está vacía. Por lo tanto, la cadena no es aceptada, no pertenece al lenguaje de palíndromos de longitud par.* ■

Un lenguaje reconocido por un AP es necesariamente independiente de contexto. Y viceversa, dado un lenguaje independiente de contexto, existe un AP que lo reconoce.

Además, todo AF es un AP, ya que se puede pensar que tiene una pila a la que no se le agregan ni quitan símbolos.

## 5.7. Máquina de Turing

Se han estudiado en las secciones anteriores los lenguajes regulares, y lenguajes independientes de contexto. Por supuesto, existen lenguajes que no son independientes de contexto. Por ejemplo, el conjunto de las cadenas de la forma $0^n1^n2^n$, para $n \geqslant 1$. El lenguaje de las cadenas de esa forma no puede ser reconocido por un autómata a pila (tampoco por un autómata finito!). Podría pensarse en una máquina más poderosa que los AP, agregándole alguna capacidad adicional.

Turing propuso un modelo de máquina abstracta que resultó ser muy importante en el estudio de los alcances de la computabilidad, es decir, en decidir qué es capaz de hacer una computadora y qué no. Los problemas que no pueden resolverse utilizando la Máquina de Turing se denominan problemas indecidibles. Si bien la máquina de Turing es un modelo abstracto y sería muy ineficiente construirla para resolver problemas, es un modelo muy útil en el estudio de las capacidades de la computación, ya que se ha comprobado que es capaz de modelar el funcionamiento de cualquier dispositivo de computación.

Cabe mencionar que en este contexto, resolver un problema es equivalente a determinar la pertenencia de una cadena a un lenguaje. Por ejemplo, pensando en el problema de decidir si un grafo es conexo o no, puede definirse a $L$ como el lenguaje de las codificaciones de todos los grafos conexos. Luego, para un grafo particular, con codificación $\mathbf{w}$, se debe determinar si $\mathbf{w} \in L$. Otro ejemplo es el problema de determinar si un determinado programa se detiene o no con determinada entrada. En este caso, se puede plantear como la cuestión de pertenencia al lenguaje $L$ de las cadenas $\mathbf{wI}$, donde $\mathbf{w}$ es la codificación de un programa $P$, e $\mathbf{I}$ es la codificación de la entrada $I$, tal que el programa $P$ se detiene con la entrada $I$.

La teoría de los problemas indecidibles estudia la existencia de tales problemas, además de determinar los límites de la capacidad de la programación. También esta teoría estudia el tipo de problemas conocidos como intratables: son aquellos que si bien son decidibles, no se conoce un algoritmo que pueda resolverlo en tiempo razonable.

La máquina de Turing (MT) es un autómata finito que posee una cinta de longitud infinita, dividida en celdas, sobre la que se pueden escribir y leer datos. Tiene un cabezal que apunta a una celda de la cinta, y puede leer su contenido y escribir en ella. La operación de la MT consiste en leer un caracter de la cinta, hacer una transición de estado, escribir sobre la cinta y mover el cabezal a la derecha o izquierda una posición (la transición de estado puede consistir en permanecer en el mismo estado, y el caracter escrito sobre la cinta puede ser el mismo caracter leído, con lo que no se cambia el contenido de la celda). Puede ser que para algún estado y algún caracter leído, no haya transición definida. En este caso, la máquina se detiene.

Además del alfabeto de entrada, una MT tiene un alfabeto de cinta, que es el conjunto de caracteres que se pueden escribir en ella. Siempre el alfabeto de entrada es un subconjunto del alfabeto de cinta.

Inicialmente la cadena de entrada está escrita en la cinta. Las celdas de la cinta no ocupadas por caracteres de la palabra de entrada, están ocupadas con espacio en blanco (que es un caracter del alfabeto de cinta, pero no del alfabeto de entrada). El cabezal de la MT al inicio apunta al caracter más a la izquierda de la palabra de entrada.

Una cadena de entrada es aceptada por una MT si y sólo si la máquina termina en un estado de aceptación. Las MT reconocen los lenguajes recursivamente enumerables, que son lenguajes que pueden ser generados por una gramática de tipo 0.

***Ejemplo* 5.46**
*Se describirá en este ejemplo una MT que reconoce el lenguaje* $\{0^n1^n, n \geqslant 1\}$. *La cadena de entrada está escrita en la cinta, y el cabezal apunta al primer caracter. La máquina funcionará de la siguiente manera: se ubica en el 0 más a la izquierda, lo reemplaza por el caracter X, y moviéndose a la derecha, busca el primer 1 y lo reemplaza por Y. Luego se mueve hacia la izquierda y busca el 0 más a la izquierda (que estará a la derecha de las X escritas en la cinta). Lo reemplaza por X, y continúa iterando como antes.*

*En cada momento, la cinta contiene una cadena formada por una secuencia de X, seguida de una secuencia de 0, seguida de una secuencia de Y, seguida de una secuencia de 0 y 1. Si la cadena de entrada es de la forma* $0^n1^n$, *en cada momento, la cinta tendrá escrita una cadena de la forma* $X^k0^{n-k}Y^k1^{n-k}$ *o* $X^k0^{n-k}Y^{k-1}1^{n-k+1}$.

*La máquina tiene 4 estados:* $q_0$, *que identifica la situación de estar leyendo el 0 más*

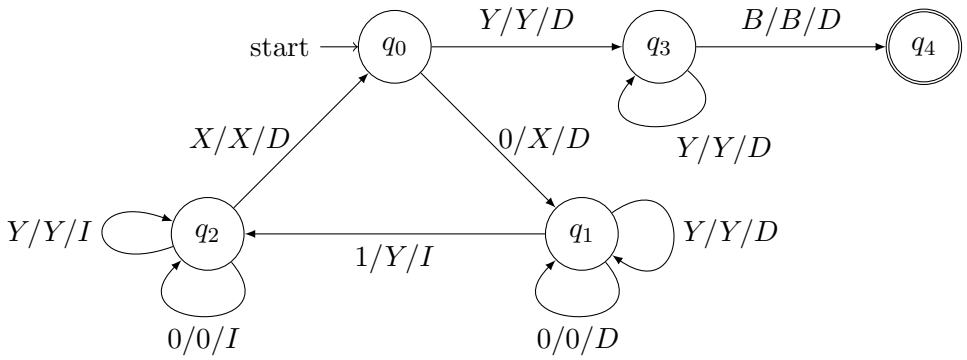



Figura 5.23: Máquina de Turing para el lenguaje $0^n1^n$

*a la izquierda; $q_1$, que representa la situación de estar moviéndose a la derecha en busca del primer 1; $q_2$ identifica movimientos a la izquierda en busca del primer 0; $q_3$, donde se encuentra la máquina si inmediatamente después de las X escritas se encontró una Y, y $q_4$ que es el estado de aceptación, al que se llega si se logra tener en la cinta una cadena de la forma $X^nY^n$, seguido del caracter espacio en blanco (B).*

*Las transiciones, escritura en cinta, y movimiento del cabezal se representan en la figura 5.23. Las etiquetas en las transiciones son de la forma $a/\alpha/M$, donde a es el caracter leído, $\alpha$ es el caracter escrito, y $M$ es el movimiento del cabezal, que será D o I, según se mueva una posición a la derecha o a la izquierda.*

Con este ejemplo como modelo, no es difícil construir una MT que reconozca el lenguaje $0^n1^n2^n$. Podría pensarse en el mismo funcionamiento: buscar el 0 más a la izquierda y cambiarlo por X, moverse a la derecha y buscar el 1 más a la izquierda y cambiarlo por Y, moverse a la derecha y buscar el 2 más a la izquierda y cambiarla por Z. Luego volver hacia la izquierda hasta encontrar una X, y repetir el proceso.

Como se sabe, no existe un AF o AP que reconozca este lenguaje.

Además de reconocer lenguajes, las MT también se aplican al cálculo de funciones. El resultado escrito en la cinta puede pensarse como el resultado de un cálculo, la salida de una función.

Por ejemplo, la suma de dos números naturales escritos en representación unaria (el número $n$ se representa con una cadena de $n$ unos) puede pensarse como una función de dos entradas. Una posible MT para implementarla podría funcionar de la siguiente manera: la cadena de entrada representa los dos números a sumar, $n$ y $m$, que se escriben en la cinta como una cadena de $n$ unos, seguida de un cero, seguida de una cadena de $m$ unos. La máquina, comenzando a la izquierda de la entrada, puede borrar el primer 1, moverse hasta el 0, y sobreescribir un 1 en esa posición. El resultado es una cadena de $n + m$ unos.

Toda función que se puede implementar en una máquina de Turing se dice que es computable. Existen funciones no computables.

Por ejemplo, sea $B(n)$ una función que indica el número máximo de unos que una MT con $n$ estados puede escribir sobre una cinta inicialmente en blanco. Se sabe que $B(2) = 4$,

$B(3) = 6$, $B(4) = 13$. Sin embargo, se desconoce $B(n)$ para $n \geqslant 5$. No existe una MT que pueda calcular esta función.

El diseño de MT que implemente funciones puede resultar muy complicado, aún para funciones simples. Sin embargo, la potencia de este modelo está en la respuesta a la pregunta de qué se puede computar y qué no, más que la construcción en sí misma.

Se han propuesto diseños alternativos de MT para aumentar su capacidad de cómputo. Por ejemplo, se han estudiado MT con más de una cinta, más de un cabezal, una cinta como un arreglo bidimensional, etc. Pero todos estos estudios determinaron que ninguna de estas ideas consigue aumentar la capacidad de la MT definida aquí, es decir, que todas las variantes del modelo son equivalentes. Se conoce como Tesis de Church-Turing a la afirmación (no demostrada pero universalmente aceptada) de que todo lo que pueda hacer un dispositivo de computación, puede ser hecho por alguna MT.

Se ha mencionado que una MT acepta una cadena si termina en un estado de aceptación. Existe otra idea de aceptación, y es el de FUNCIONAMIENTO FINITO. O, en otras palabras, que la MT pare en algún momento, por ejemplo, llegando a un estado $q$, apuntando a un caracter $a$, siendo que no está definida la transición para este caso.

Siempre es posible hacer que una MT pare cuando acepta una palabra, por ejemplo, si no hay transiciones desde el estado de aceptación. Pero puede ser que la máquina no se detenga si no acepta la palabra, y quede ciclando infinitamente. Los lenguajes reconocidos por MT que siempre se detienen, ya sea que acepten o no la cadena analizada, se denominan RECURSIVOS.

Se ha dicho que un lenguaje aceptado por una MT entra en la tipificación de lenguajes recursivamente enumerables. Esta clasificación no abarca a todos los lenguajes, ya que hay lenguajes que no son aceptados por una MT. Como en este contexto, problemas y lenguajes puede verse como lo mismo, estamos diciendo que hay problemas indecidibles. Daremos un ejemplo de un problema indecidible, es decir, para el cual no existe una MT que pueda resolverlo.

Todas las MT se pueden codificar con cadenas binarias. Es decir, la descripción de la función de transición se puede hacer con cadenas de 0 y 1. Dada una MT $A$, llamaremos $\mathbf{w_A}$ a su codificación. Y consideremos las máquinas cuyo alfabeto de entrada es $\{0, 1\}$. El problema de decidir si una MT $A$ acepta la cadena $\mathbf{w_A}$ es indecidible. Es decir, no existe una MT que resuelva esa cuestión para toda MT de alfabeto binario. En términos de lenguajes, se está considerando el lenguaje $L_{noRE} = \{\mathbf{w} : \mathbf{w}$ no es aceptada por la MT codificada con $\mathbf{w}\}$, consistente en las codificaciones de máquinas que no son aceptadas por la misma máquina. Este lenguaje no es recursivamente enumerable: no existe MT que acepte exactamente las cadenas en $L_{noRE}$.

Si existiera tal máquina, llamémosla $T$. Sea $\mathbf{w_T}$ su codificación.

Si $\mathbf{w_T} \in L_{noRE}$, es aceptada por $T$, ya que esta máquina acepta las cadenas en el lenguaje $L_{noRE}$. Pero, por definición de $L_{noRE}$, $\mathbf{w_T}$ no es aceptada por la máquina que representa, es decir, no es aceptada por $T$, y se llega a una contradicción.

Si $\mathbf{w_T} \notin L_{noRE}$, no es aceptada por $T$, ya que $T$ acepta únicamente las cadenas en $L_{noRE}$. Por definición de $L_{noRE}$, como $\mathbf{w_T}$ no está en ese conjunto, es aceptada por la máquina que representa, $T$. Nuevamente se llega a una contradicción.

Las contradicciones surgen de suponer que la máquina $T$ existe. Entonces, no existe máquina que reconozca $L_{noRE}$ y entonces no es un lenguaje recursivamente enumerable.

Para finalizar, se presenta una tabla resumen de los tipos de lenguajes, tipos de gramáticas y tipos de máquinas correspondientes.

| **Máquina** | **Lenguaje que acepta** | **Gramática que lo genera** |
|---|---|---|
| Autómatas finitos | Regulares | Regulares (Tipo 3) |
| Autómatas a pila | Independiente de contexto | Independiente de contexto (Tipo 2) |
| Autómatas linealmente acotados | Dependientes de contexto | Dependientes de contexto (Tipo 1) |
| Máquinas de Turing que se detienen | Recursivos | |
| Máquinas de Turing | Recursivamente enumerables | No restringidas (Tipo 0) |

## 5.8. Problemas

**Problema 5.1**
*Sean* $\Sigma_1 = \{a, b, c\}$, $\Sigma_2 = \{0, 1, 2\}$ *y* $\Sigma_3 = \Sigma_1 \cup \Sigma_2$ *alfabetos; y considere los lenguajes* $L_1 = \{0^i 2^j : i, j \in \mathbb{Z}, i, j \geqslant 1\}$, $L_2 = \{a^i b^j : i, j \in \mathbb{Z}, i, j \geqslant 1\}$; $L_3 = \{0^i 2^j : i, j \in \mathbb{Z}, 1 \leqslant i, j \leqslant 3\}$; $L_4 = \{a0i^2 j^b : i, j \in \mathbb{Z}, 1 \leqslant i, j \leqslant 3\}$ *y* $L_5 = \{a, c, 0, 1, 2\}$.

*Dar cuatro cadenas de cada uno de los lenguajes.*

**Problema 5.2**
*Decir cuáles de las siguientes afirmaciones sobre los lenguajes definidos en el problema anterior son verdaderas.*

*a.* $L_1$ *es un lenguaje sobre* $\Sigma_2$

*b.* $L_2$ *está contenido en la clausura de* $\Sigma_1$

*c.* $L_3$ *es un subconjunto de* $L_1$

*d.* $L_4$ *es un subconjunto de* $L_1$

*e.* $L_5$ *es un lenguaje sobre* $\Sigma_3$

*f.* $L_5$ *es finito*

*g.* $L_1$ *es finito*

*h.* $L_3$ *es finito*

**Problema 5.3**
*Dado el alfabeto* $\Sigma = \{a, c, l, n, d\}$ *y los lenguajes* $L_1 = \{ala, c, ca\}$ *y* $L_2 = \{ana, ada, da\}$, *dar los lenguajes* $L_1 L_2$ *y* $L_1 \cup L_2$.

*¿Cuál es la relación entre la cantidad de palabras en* $L_1$, *en* $L_2$, *en* $L_1 L_2$ *y en* $L_1 \cup L_2$*?*

**Problema 5.4**
*Dar un ejemplo de dos lenguajes* $L_1$ *y* $L_2$ *tales que* $|L_1 L_2| = |L_1||L_2|$ *y* $|L_1 \cup L_2| = |L_1| + |L_2|$.

**Problema 5.5**
*Dado el alfabeto* $\Sigma = \{0, 1\}$, *describa las cadenas de los lenguajes*

*a.* $0^*1(01)^*1$

*b.* $001^*0$

*c.* $10^*$

*d.* $(0+1)0$

*e.* $0^*1^30$

**Problema 5.6**
*Dado el alfabeto* $\Sigma = \{0,1\}$*, determine si la cadena* 010001 *pertenece a algun/os de los siguientes lenguajes:* $01(00)^*1$*;* $010^*1$*;* $(01) + 0^*(01)$*;* $(010001)^+$

**Problema 5.7**
*Dado el alfabeto* $\Sigma = \{0,1\}$*, dé expresiones regulares para describir los lenguajes*

*a. que tengan exactamente dos 0*

*b. que terminen con 101*

*c. que comiencen con 0 y terminen en 1*

*d. que contienen una cantidad par de 0*

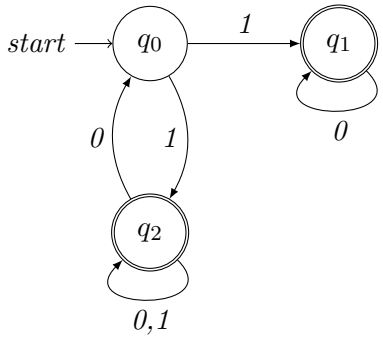
**Problema 5.8**
*Para el alfabeto* $\{0,1,2\}$*, obtenga una expresión regular para el lenguaje formado por las palabras que contienen exactamente una vez dos 1 consecutivos.*

**Problema 5.9**
*Diseñar autómatas finitos correspondientes a cada uno de los lenguajes descritos por las expresiones regulares del problema 5.5.*

**Problema 5.10**
*Diseñar autómatas finitos correspondientes a cada uno de los lenguajes descritos por las expresiones regulares del problema 5.7.*

**Problema 5.11**
*Diseñar un autómata finito que acepte el lenguaje descrito por la expresión regular del problema 5.8.*

**Problema 5.12**
*Obtener la expresión regular para el autómata finito de la figura:*



**Problema 5.13**
*Generar una gramática libre de contexto que genere las palabras binarias que comienzan con 1.*

**Problema 5.14**
*Proponer una gramática libre de contexto para producir el lenguaje* $\{a^i b^j, i > j\}$.

**Problema 5.15**
*Proponer una gramática libre de contexto para producir el lenguaje de paréntesis, corchetes y llaves balanceados.*

**Problema 5.16**
*En un lenguaje de programación, cada aparición de un else debe ir precedida por un if. Y cada if no puede corresponder a más de un else. Por otro lado, puede aparecer un if sin un else.*

*Entonces, en un programa en ese lenguaje pueden aparecer secuencias como if ... if ... else ... if ...; pero no es correcta la secuencia if ... else ... else ... if ... else ....*

*Proponer una gramática para generar las secuencias de if y else válidas .*

**Problema 5.17**
*Diseñar un autómata a pila para aceptar el lenguaje de los palíndromos con un número par de símbolos.*

**Problema 5.18**
*Diseñar un autómata a pila para aceptar el lenguaje formado por las palabras de* $\{0,1\}^*$ *que tenga la misma cantidad de unos que de ceros.*

**Problema 5.19**
*Si se quiere diseñar un autómata para buscar ciertas palabras claves en un texto , ¿qué tipo de autómata es más apropiado?*

**Problema 5.20**
*Suponga que se requiere un analizador léxico para detectar direcciones (nombre de calle + número de casa) en páginas web. Construye una expresión regular para el analizador.*