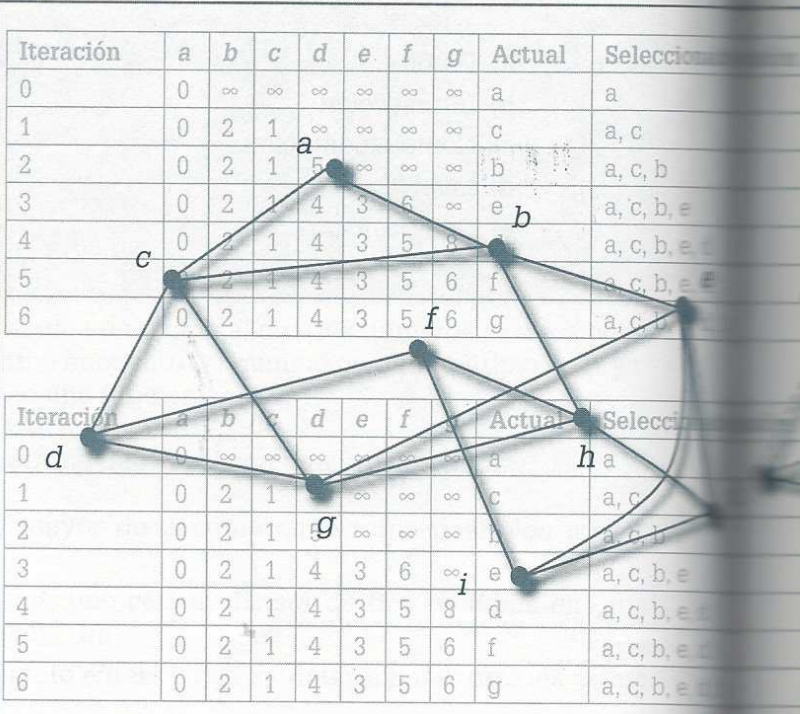
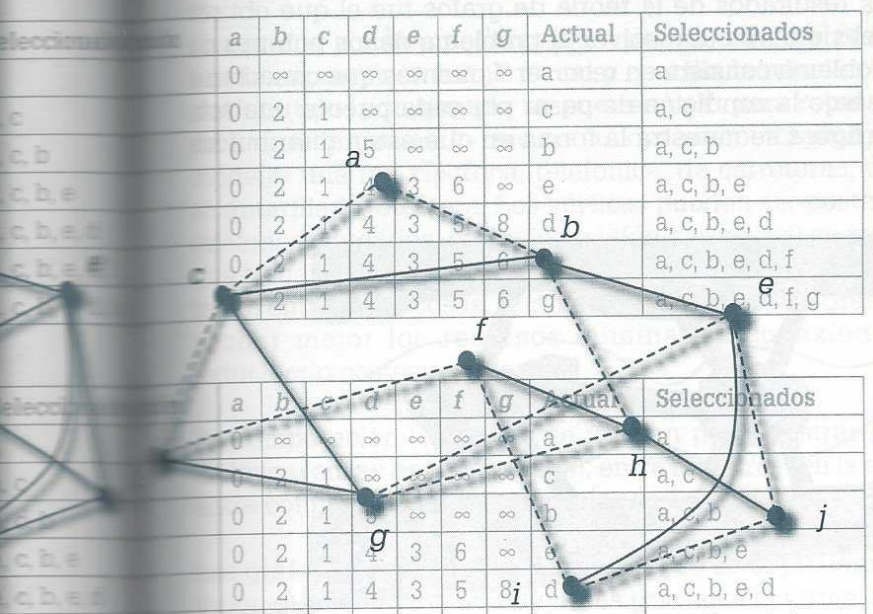
CAPÍTULO VII

Grafos



Iteración	a	b	c	d	e	f	g	Actual	Selección
0	0	∞	∞	∞	∞	∞	∞	a	a
1	0	2	1	∞	∞	∞	∞	c	a, c
2	0	2	1	5	∞	∞	∞	b	a, c, b
3	0	2	1	4	3	6	∞	e	a, c, b, e
4	0	2	1	4	3	5	8	d	a, c, b, e, d
5	0	2	1	4	3	5	6	f	a, c, b, e, d, f
6	0	2	1	4	3	5	6	g	a, c, b, e, d, f, g

Iteración	a	b	c	d	e	f	g	Actual	Selección
0	0	∞	∞	∞	∞	∞	∞	a	a
1	0	2	1	∞	∞	∞	∞	c	a, c
2	0	2	1	5	∞	∞	∞	b	a, c, b
3	0	2	1	4	3	6	∞	e	a, c, b, e
4	0	2	1	4	3	5	8	d	a, c, b, e, d
5	0	2	1	4	3	5	6	f	a, c, b, e, d, f
6	0	2	1	4	3	5	6	g	a, c, b, e, d, f, g



a	b	c	d	e	f	g	Actual	Seleccionados
0	∞	∞	∞	∞	∞	∞	a	a
0	2	1	∞	∞	∞	∞	c	a, c
0	2	1	5	∞	∞	∞	b	a, c, b
0	2	1	4	3	6	∞	e	a, c, b, e
0	2	1	4	3	5	8	d	a, c, b, e, d
0	2	1	4	3	5	6	f	a, c, b, e, d, f
0	2	1	4	3	5	6	g	a, c, b, e, d, f, g

a	b	c	d	e	f	g	Actual	Seleccionados
0	∞	∞	∞	∞	∞	∞	a	a
0	2	1	∞	∞	∞	∞	c	a, c
0	2	1	5	∞	∞	∞	b	a, c, b
0	2	1	4	3	6	∞	e	a, c, b, e
0	2	1	4	3	5	8	d	a, c, b, e, d
0	2	1	4	3	5	6	f	a, c, b, e, d, f
0	2	1	4	3	5	6	g	a, c, b, e, d, f, g

Si simplemente hace girar la rueda, es álgebra; pero si contiene una idea, es topología.

Solomon Lefschetz

Objetivos

- Aprender la simbología y nomenclatura propia de la teoría de grafos y distinguir las características de éstos.
- Aplicar la teoría de grafos a la solución de problemas que se pueden resolver con esta metología.
- Saber qué condiciones debe cumplir un grafo que se considera tiene circuitos importantes como Euler y Hamilton.
- Usar la técnica de coloración de grafos para iluminar un mapa con la cantidad mínima de colores sin que se junten dos colores iguales.

7.1 Introducción

Leonhard Euler
(1707-1783)

Fue un matemático y físico suizo que es considerado uno de los más grandes matemáticos de la historia, ya que realizó importantes descubrimientos en áreas tan diversas como el cálculo, la teoría de grafos, la geometría, el álgebra, la teoría de números, el cálculo de variaciones, la mecánica, la hidrodinámica, la óptica y la astronomía, y también introdujo gran parte de la moderna terminología del análisis matemático.

Hacia 1738 Euler quedó prácticamente ciego y a pesar de este problema su productividad intelectual no disminuyó gracias a su gran capacidad de cálculo mental y a su memoria fotográfica. A partir de 1741 vivió en Berlín y aquí escribió más de 380 artículos y publicó dos de sus principales obras: la *Introductio in analysin infinitorum*, un texto acerca de las funciones matemáticas publicado en 1748, y la *Institutiones calculi differentialis*, que se publicó en 1755 y que trata acerca del cálculo diferencial.

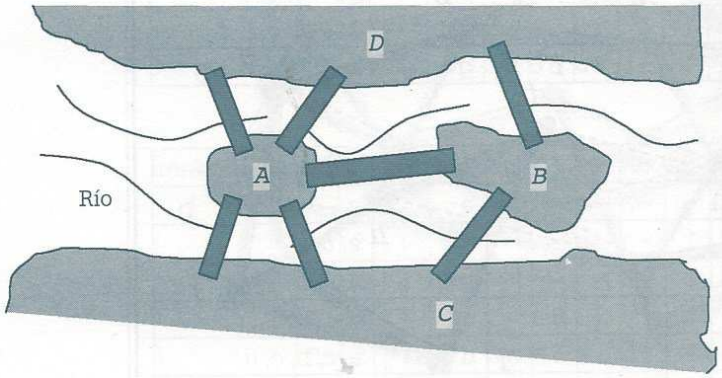
En 1735 Euler resolvió el problema conocido como el *problema de los puentes de Königsberg*, y con esta solución estableció lo que se considera como el primer teorema de la teoría de grafos así como el nacimiento de la topología.

Topología

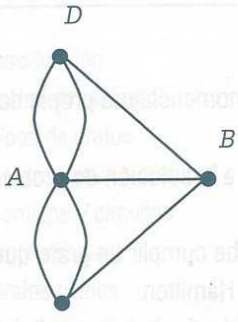
En términos coloquiales, la topología es un área de la matemática que estudia las propiedades de los objetos que no cambian cuando éstos se deforman o se estiran. Un ejemplo de este tipo de deformación es la que se puede hacer sobre una taza de goma para obtener una dona:

(continúa)

Uno de los primeros resultados de la teoría de grafos fue el que obtuvo Leonhard Euler en el siglo XVIII al resolver el problema de los puentes de Königsberg. Este problema consiste en recorrer 7 puentes que conectan 4 porciones de tierra, bajo la condición de pasar por cada puente una sola vez. En la siguiente figura se muestra la forma en que están distribuidos los puentes:



Euler representó este problema por medio de una figura como la siguiente:



y la llamó "grafo". A las porciones de tierra representadas por un punto las llamó "vértices", a los puentes representados por líneas les dio el nombre de "aristas" y al número de líneas que salen o entran a un vértice le llamó "orden del vértice", el cual más tarde se llamó valencia.

Después de analizar el problema, Euler llegó a la conclusión de que es imposible obtener un itinerario que salga de un vértice y regrese a él pasando por todas las aristas solamente una vez. Según Euler, si el vértice donde se inicia y termina el recorrido es el mismo, entonces dicho vértice debe ser de valencia par ya que por un puente se sale y por otro diferente se debe de regresar. Lo mismo ocurre con todos los demás vértices, ya

que debe estar contemplada la entrada y la salida, por lo tanto Euler estableció que "en un grafo, únicamente se puede establecer un ciclo que pase por todas las aristas sólo una vez si todos los vértices tienen valencia par".

Los grafos son representaciones de las redes, y por medio de ellos se puede expresar en forma visual y sencilla la relación entre elementos de distinto tipo, por ejemplo se pueden usar para representar la estructura de una empresa en lo que se conoce como "organigrama", o bien para modelar una red eléctrica, telefónica, de carreteras, de agua potable, de alcantarillado, etcétera. Los vértices pueden ser postes, transformadores, teléfonos, ciudades, centrales telefónicas, válvulas, registros, y las aristas que tienen relación entre esos vértices pueden ser cables, tubos y carreteras, entre otras cosas. Por medio de la teoría de grafos, se pueden aprovechar mejor los recursos eliminando conexiones redundantes y reduciendo costos y distancias.

En computación los grafos se utilizan para mostrar las relaciones entre archivos (en las bases de datos), entre registros (en la estructura de datos), entre computadoras y entre redes como lo hace la red internet.

Los grafos son relaciones, como las que se expusieron en el capítulo anterior, que resultan ser muy útiles gracias a la forma en la que se les puede representar ya que es más claro ver la relación entre dos elementos en un grafo que en una matriz o en un conjunto.

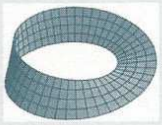
En este capítulo se exponen los conceptos fundamentales de los grafos y se presentan algunas aplicaciones de ellos en el área de la computación, sin embargo la teoría expuesta aquí se puede transportar a cualquier sistema en donde la intercomunicación es fundamental.

(continuación)

Bajo estas condiciones una taza de café y una dona son topológicamente equivalentes ya que deformando y estirando cualquiera de ellas se obtiene la otra como resultado de la transformación.

El nacimiento de la topología se suele ubicar en el año 1735, cuando Euler resolvió el problema de los puentes de Königsberg, solución que no sólo exhibe un enfoque totalmente topológico, sino que además aporta el primer invariante de la topología algebraica.

Hacia finales del siglo XVIII el punto de partida del desarrollo sistemático de la topología fue la definición rigurosa de conceptos fundamentales del análisis como función, continuidad, diferenciabilidad, así como el estudio de las nuevas geometrías no euclideanas y de objetos geométricos como la banda de Möbius (1858) que se muestra a continuación

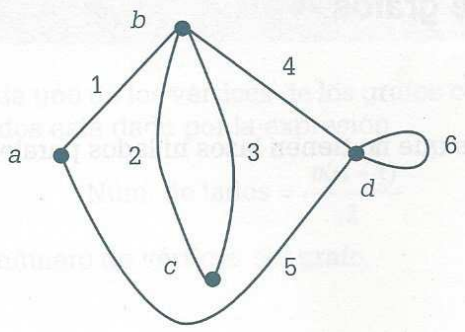


la cual fue descubierta en forma independiente por los matemáticos alemanes August Ferdinand Möbius y Johann Benedict Listing en 1858. Como se puede ver, esta banda se caracteriza por ser una superficie con un solo lado y un solo contorno además de ser un objeto no orientable.

7.2 Partes de un grafo

Un grafo (G) es un diagrama que consta de un conjunto de vértices (V) y un conjunto de lados (L).

Considérese el siguiente grafo:



A partir de esta figura se definen los siguientes elementos:

- ***Vértices (nodos)***

Se indican por medio de un pequeño círculo y se les asigna un número o letra. En el grafo anterior los vértices son V = {*a, b, c, d*}.

- ***Lados (ramas o aristas)***

Son las líneas que unen un vértice con otro y se les asigna una letra, un número o una combinación de ambos. En el grafo anterior los lados son L = {1, 2, 3, 4, 5, 6}.

- ***Lados paralelos***

Son aquellas aristas que tienen relación con un mismo par de vértices. En el grafo anterior los lados paralelos son: P = {2, 3}.

- ***Lazo***

Es aquella arista que sale de un vértice y regresa al mismo vértice. En el grafo anterior se tiene el lazo: A = {6}.

- ***Valencia de un vértice***

Es el número de lados que salen o entran a un vértice. En el grafo anterior las valencias de los vértices son:

Valencia (*a*) = 2
Valencia (*b*) = 4
Valencia (*c*) = 2
Valencia (*d*) = 3

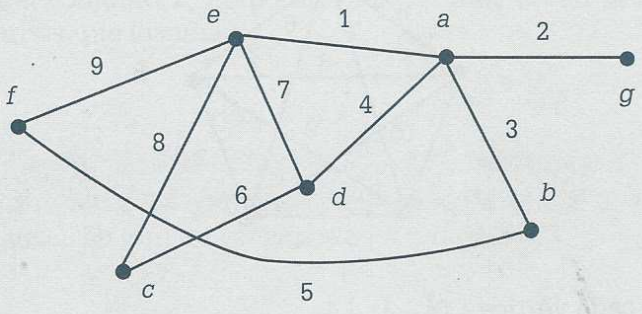
Hay que observar cómo en el caso del vértice *d* el lazo sólo se considera una vez, entrada o salida pero no ambos.

7.3 Tipos de grafos

- ***Grafos simples***

Son aquellos grafos que no tienen lazos ni lados paralelos.

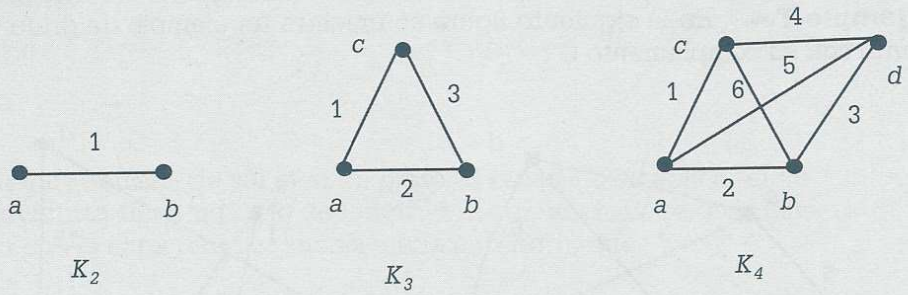
Ejemplo 7.1. En la siguiente figura se muestra un ejemplo de grafo simple.



- ***Grafo completo de n vértices (K_n)***

Es el grafo en donde cada vértice está relacionado con todos los demás, sin lazos ni lados paralelos. Se indica como K_n, en donde n es el número de vértices del grafo.

Ejemplo 7.2. En la siguiente figura se muestran tres ejemplos de grafos completos.

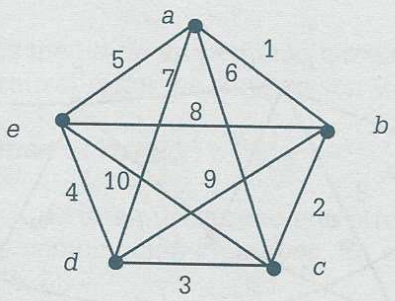


La valencia en cada uno de los vértices de los grafos completos es $(n-1)$, y el número de lados está dado por la expresión

$$\text{Núm. de lados} = \frac{n(n-1)}{2}$$

en donde n es el número de vértices del grafo.

Ejemplo 7.3. El grafo K_5 de la siguiente figura tiene



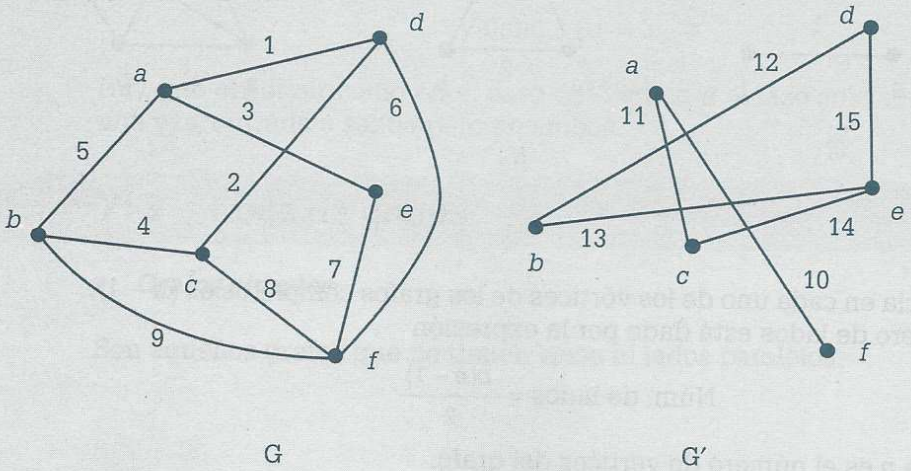
Valencia de cada vértice = (5 – 1) = 4

$$\text{Núm. de lados} = \frac{5(5-1)}{2} = 10$$

- ***Complemento de un grafo (G′)***

Es el grafo que le falta al grafo G, de forma que entre ambos forman un grafo completo de *n* vértices. Este grafo no tiene lazos ni ramas paralelas.

Ejemplo 7.4. En la siguiente figura se muestra un ejemplo de grafo G junto con su complemento G′:

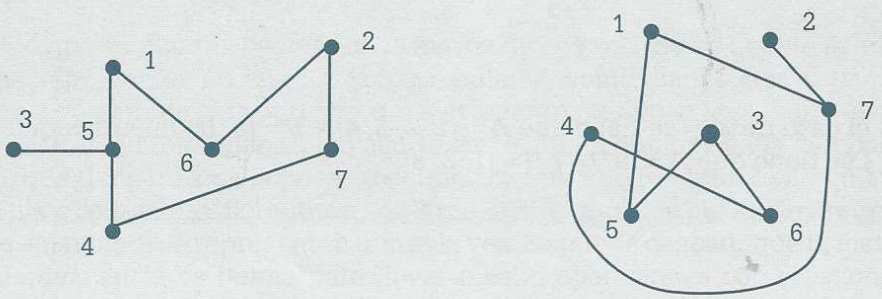


G G′

- ***Grafo bipartido***

Es el grafo que está compuesto por dos conjuntos de vértices, $A = \{a_1, a_2, a_3, \ldots, a_n\}$ y $B = \{b_1, b_2, \ldots, b_m\}$, en donde los elementos del conjunto A se relacionan con los del conjunto B, pero entre los vértices de un mismo conjunto no existe arista que los una.

Ejemplo 7.5. Sean los conjuntos de vértices $A = \{1, 2, 3, 4\}$ y $B = \{5, 6, 7\}$, con los cuales se forman los siguientes grafos:



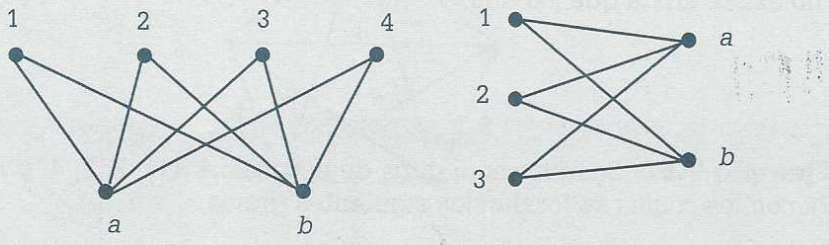
Estos dos grafos son bipartidos, ya que los elementos del conjunto A están relacionados con los del conjunto B, pero entre los elementos de un mismo conjunto no hay relación alguna.

Una forma muy sencilla de saber si un grafo es bipartido es aplicar el hecho de que nunca tiene un ciclo de longitud impar, además de que debe de cumplir con la característica mencionada anteriormente.

- ***Grafo bipartido completo ($K_{n, m}$)***

Es el grafo que está compuesto por dos conjuntos de vértices, uno de ellos $A = \{a_1, a_2, a_3, \ldots, a_n\}$ y otro $B = \{b_1, b_2, \ldots, b_m\}$, y en el que cada vértice de A está unido con todos los vértices de B, pero entre los vértices de un mismo conjunto no existe arista que los una. El grafo bipartido completo se indica como $K_{n, m}$.

Ejemplo 7.6. En la siguiente figura se muestran dos grafos bipartidos completos:



$K_{4,2}$ $K_{2,3}$

En el caso de $K_{4,2}$ se tiene que $A = \{1, 2, 3, 4\}$ y $B = \{a, b\}$, mientras que en $K_{2,3}$ se tiene que $A = \{a, b\}$ y $B = \{1, 2, 3\}$.

7.4 Representación matricial

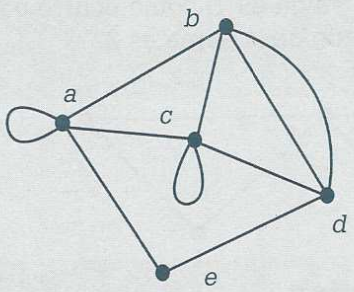
El uso de matrices para representar sistemas de ecuaciones, relaciones o grafos permite una rápida y clara manipulación de la información, así como el determinar algunas propiedades de los grafos que de otra manera serían más difíciles de obtener. Además de esto se tiene que en la computadora es más fácil el manejo de matrices, ya que se pueden tratar como arreglos o listas doblemente ligadas.

A continuación se describen las representaciones matriciales de los grafos.

- ***Matriz de adyacencia (M_a)***

Es una matriz cuadrada en la cual los vértices del grafo se indican como filas y como columnas: el orden de los vértices es el mismo que guardan las filas y las columnas de la matriz. Se coloca un 1 como elemento de la matriz cuando existe una relación entre uno y otro vértice, o bien un 0 cuando no exista relación alguna.

Ejemplo 7.7. A continuación se muestra un grafo y su matriz de adyacencia correspondiente:



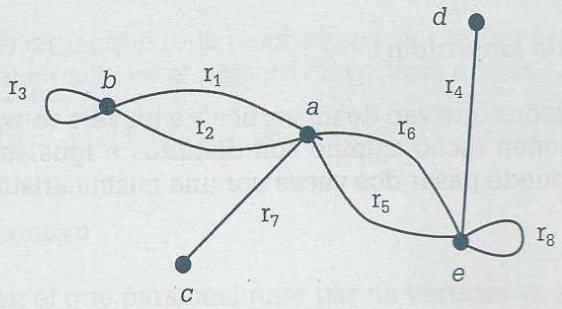
$$M_a = \begin{array}{c|ccccc} & a & b & c & d & e \\ \hline a & 1 & 1 & 1 & 0 & 1 \\ b & 1 & 0 & 1 & 1 & 0 \\ c & 1 & 1 & 1 & 1 & 0 \\ d & 0 & 1 & 1 & 0 & 1 \\ e & 1 & 0 & 0 & 1 & 0 \end{array}$$

Algo que se puede observar en la matriz de adyacencia es que no se pueden representar en ella los lados paralelos, como ocurre con el par de aristas que unen los nodos *b* y *d*. En esta matriz también la mayoría de las aristas están repetidas, como ocurre con la arista que une a los vértices *b* y *c* que tiene un 1 en la línea *b* columna *c*, pero que también tiene un 1 en la fila *c* columna *b*. Por último, los lazos, a diferencia de las aristas normales solamente se representan una sola vez. Se puede concluir que la matriz de adyacencia es buena para llevar a cabo operaciones con relaciones, pero que no permite registrar en ella toda la información del grafo.

- ***Matriz de incidencia (M_i)***

En esta matriz se colocan los vértices del grafo como filas y las aristas como columnas.

Ejemplo 7.8. Considérese el siguiente grafo junto con su matriz de incidencia correspondiente:



$$M_i = \begin{array}{c|cccccccc|c} & r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 & \\ \hline a & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 5 \\ b & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 3 \\ c & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ d & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ e & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 4 \\ \hline & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 1 & \end{array}$$

En esta matriz sí es posible representar lados paralelos, como ocurre con r_1, r_2, r_5 y r_6. Al sumar los elementos de cada una de las filas se obtiene la valencia de los vértices, y al sumar las columnas es posible distinguir cuando se trata de un lazo ya que su suma es 1, como ocurre con r_3 y r_8. Cuando no se trata de lazos, el resultado de la suma es 2.

7.5 Caminos y circuitos

En un grafo se puede recorrer la información de diferente manera, lo cual implica seguir distintas rutas para llegar de un nodo del grafo a otro. A continuación se definen varios conceptos relacionados con el recorrido de un grafo, y en el ejemplo 7.9 se ilustran éstos.

- ***Camino***

Es una sucesión de lados que van de un vértice *x* a un vértice *w* (dichos lados se pueden repetir).

- ***Circuito (ciclo)***

Es un camino del vértice *w* al vértice *w*, esto es, un camino que regresa al mismo vértice de donde salió.

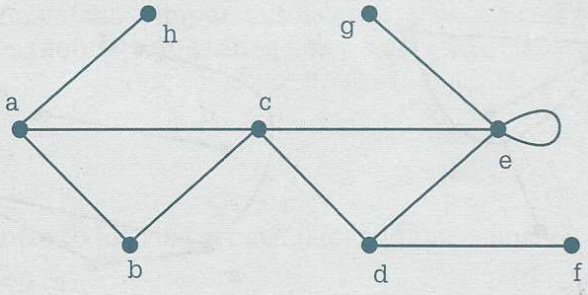
- ***Circuito simple de longitud n***

Es aquel camino del vértice *w* al vértice *w* que solamente tiene un ciclo en la ruta que sigue.

- ***Camino simple de longitud n***

Es una sucesión de lados que van de un vértice *x* a un vértice *w*, en donde los lados que componen dicho camino son distintos e iguales a *n*. Esto significa que no se puede pasar dos veces por una misma arista.

Ejemplo 7.9. Con relación al grafo



se tienen los recorridos que muestra la tabla con sus correspondientes características.

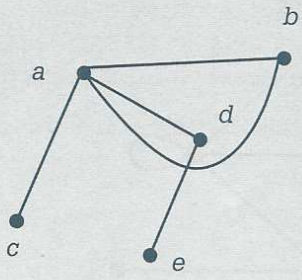
Recorrido	Camino	Camino simple de longitud n	Circuito	Circuito simple de longitud n
{a, b, c, e, d, f}	*	* L = 5		
{a, h, a, b, c}	*			
{c, e, e, d, c, b}	*	* L = 5		
{d, e, g, e, e, d}	*		*	
{e, e}	*		*	* L = 1
{h, a, b, c, a, h}	*		*	
{c, d, e, c}	*		*	* L = 3
{a, b, c, d, e, c}	*			
{a, h, a}	*		*	* L = 2
{b, a, c, d, f}	*	* L = 4		

Observar que todo recorrido es un camino y que la longitud del camino o del circuito es el número de vértices que se tocan menos 1.

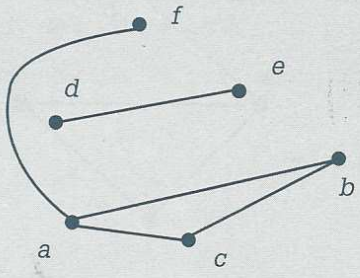
- ***Grafo conexo***

Es aquél en el que para cualquier par de vértices *w*, *x*, distintos entre sí, existe un trayecto para ir de *w* a *x*.

Ejemplo 7.10. Aquí se muestra un grafo conexo y uno no conexo.



Grafo conexo



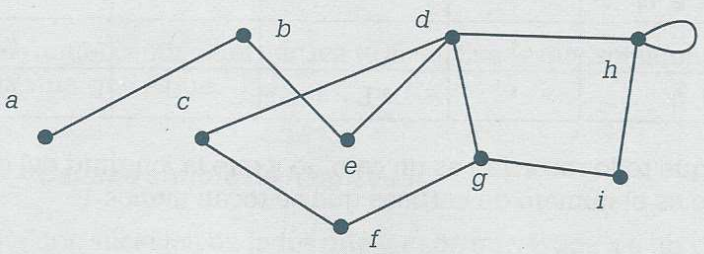
Grafo no conexo

En el grafo conexo (conectado) siempre existe un camino para ir de un vértice a otro, sin embargo en el grafo no conexo existen vértices que no están conectados y, por lo tanto, no se puede acceder a ellos. Así, en el grafo no conexo del ejemplo 7.10 no se puede tener un camino para ir del vértice *b* al *e*.

- ***Camino de Euler***

Es aquel camino que recorre todos los vértices pasando por todas las ramas solamente una vez.

Ejemplo 7.11. Considérese el siguiente grafo



Un camino de Euler es {*a, b, e, d, c, f, g, d, h, h, i, g*} o bien {*g, i, h, h, d, g, f, c, d, e, b, a*}.

Una característica importante de los grafos que tienen camino de Euler es que siempre comienza y termina en vértices que tienen valencia impar, por esta razón es imposible que en el grafo del ejemplo 7.11 un camino de Euler pueda comenzar en el vértice *f*. Por otro lado, si un grafo tiene más de dos vértices con valencia impar, entonces no puede tener un camino de Euler ya que es requisito que tenga dos y solamente dos vértices de valencia impar.

- ***Circuito de Euler***

Es aquel ciclo que recorre todos los vértices pasando por todos los lados solamente una vez.

Un grafo tiene un circuito de Euler si y sólo si es conexo y todos sus vértices tienen valencia par.

El siguiente algoritmo de Fleury permite determinar un circuito de Euler:

1) Verificar que el grafo sea conexo y que todos los vértices tengan valencia par. Si no cumple con estas condiciones entonces el grafo no tiene circuito de Euler y finalizar.

2) Si cumple con la condición anterior, seleccionar un vértice arbitrario para iniciar el recorrido.

3) Escoger una arista a partir del vértice actual. Esa arista seleccionada no debe ser "lado puente", a menos que no exista otra alternativa.

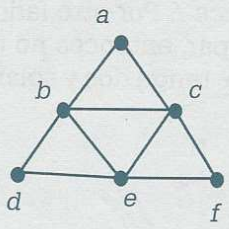
   Lado puente es aquella arista que si se elimina, los grafos pierden la propiedad de ser conexos.

4) Desconectar los vértices que están unidos por la arista seleccionada.

5) Si todos los vértices del grafo ya están desconectados, ya se tiene el circuito de Euler y finalizar. De otra manera continuar con el paso 3.

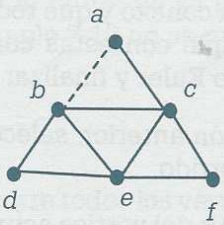
La firma del diablo es un juego que consiste en dibujar una figura sin levantar el lápiz del papel, partiendo de un punto y regresando nuevamente a él sin pasar dos veces por una misma arista. Este problema se puede resolver por medio del *circuito de Euler*.

Ejemplo 7.12. Determinar un circuito de Euler en el siguiente grafo.



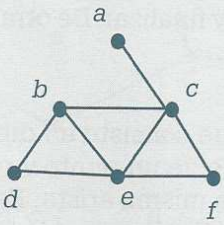
Solución

1) Se puede observar que, efectivamente, se trata de un grafo conexo y que todos sus vértices tienen valencia par.
2) Considérese que se inicia el recorrido en el vértice *a*.
3) Hay que escoger una arista a partir del vértice actual, y esa arista seleccionada no debe ser "lado puente" a menos que no exista otra alternativa.



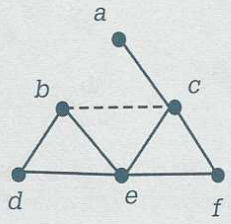
Se puede seleccionar cualquiera de las dos aristas (*a, b*) o (*a, c*), ya que ninguna es puente. Supóngase que en este caso se escoge (*a, b*), indicada con línea punteada.

4) Regístrese como parte del circuito de Euler dicha arista. Circuito de Euler: (*a, b*).
5) Desconéctense los vértices que están unidos por la arista seleccionada. Después de eliminar la arista se tiene el siguiente grafo

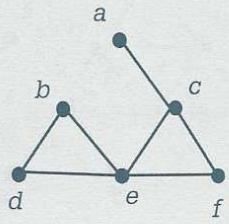


Como en este caso todavía no están desconectados todos los vértices del grafo, se continúa desde el paso 3, obteniéndose las siguientes posiciones:

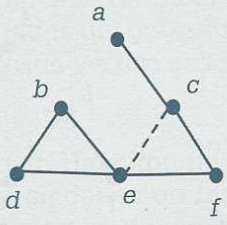
Como se ve en el siguiente grafo, del vértice actual *b* se puede seleccionar cualquiera de las aristas (*b*, *c*), (*b*, *e*) o (*b*, *d*), ya que ninguna es puente. Supóngase que se escoge (*b*, *c*) indicada con línea punteada. Circuito de Euler: (*a*, *b*, *c*).



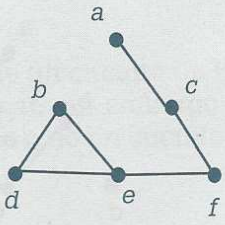
Eliminando la arista seleccionada se obtiene el siguiente grafo:



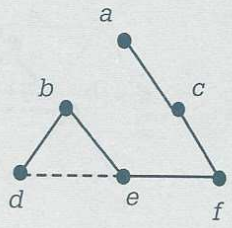
Del vértice actual *c* se puede seleccionar cualquiera de las aristas (*c*, *e*) o (*c*, *f*), ya que ninguna es puente. Supóngase que se escoge (*c*, *e*) como se indica. Circuito de Euler: (*a*, *b*, *c*, *e*).



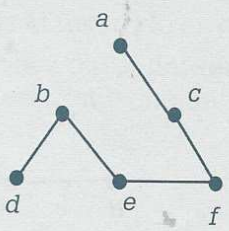
Eliminando la arista seleccionada se obtiene el siguiente grafo:



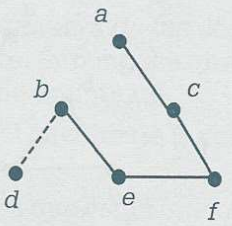
Del vértice actual *e* se puede seleccionar cualquiera de las aristas (*e*, *b*) o (*e*, *d*), pero no (*e*, *f*) porque se trata de un lado puente. Supóngase que se escoge (*e*, *d*) como se indica en el siguiente grafo. Circuito de Euler: (*a*, *b*, *c*, *e*, *d*).



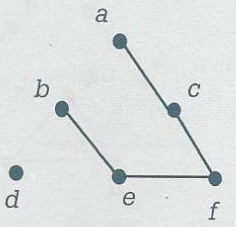
Eliminando la arista seleccionada se obtiene el siguiente grafo:



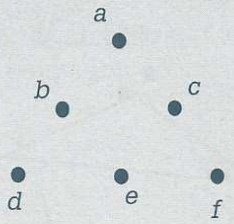
En el vértice actual *d* solamente está el lado puente (*d*, *b*), pero como ya no existe otra arista se selecciona ésta. Circuito de Euler: (*a*, *b*, *c*, *e*, *d*, *b*)



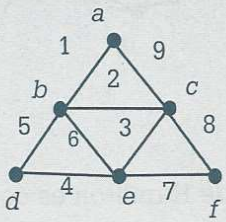
Eliminado la arista seleccionada se obtiene el siguiente grafo:



Se continúa así hasta que todos los vértices queden desconectados. Finalmente se obtiene el circuito de Euler: (*a, b, c, e, d, b, e, f, c, a*).



Como se mencionó, con el circuito de Euler se puede dar solución al juego de *La firma del diablo* al reconstruir el grafo, partiendo del nodo *a*, como lo muestra la numeración de las aristas:



- ***Circuito de Hamilton***

Se trata de un problema similar al del circuito de Euler, con la diferencia de que en lugar de pasar por todos los lados del grafo solamente una vez, en el circuito de Hamilton se pasa por cada vértice solamente una vez.

El problema surgió en el siglo XIX cuando Hamilton inventó un juego en donde estaban colocados nombres de ciudades en las esquinas de un dodecaedro. El juego consiste en iniciar en cualquier ciudad, viajar a lo largo de las aristas y visitar cada una de las ciudades exactamente una vez y regresar al punto de partida.

Respecto de un grafo se sabe que tiene un circuito de Euler si es conexo y todos sus vértices tienen valencia par, sin embargo no hay forma de saber con anticipación si un grafo tiene o no un circuito de Hamilton.

William Rowan Hamilton
(1805-1865)

Fue un matemático, físico y astrónomo irlandés que hizo importantes contribuciones al desarrollo de la óptica, la dinámica y el álgebra. Sin lugar a dudas su investigación mejor conocida es su descubrimiento de los cuaterniones, sin embargo su trabajo también ha sido muy importante en el desarrollo de la mecánica cuántica.

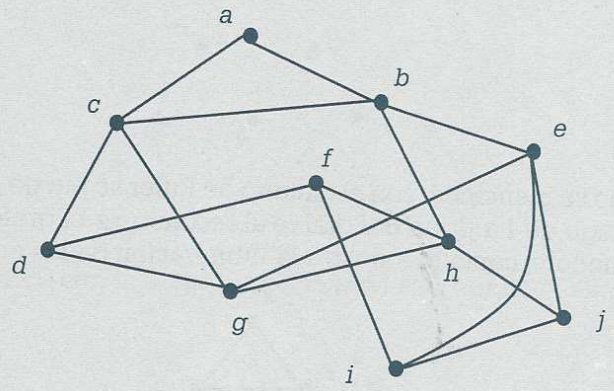
En relación con los cuaterniones se tiene que éstos se definen como el conjunto de números de la forma

$$H = \{a+bi+cj+dk \mid a, b, c, d \in R; \; i=j=k=ijk=-1\}$$

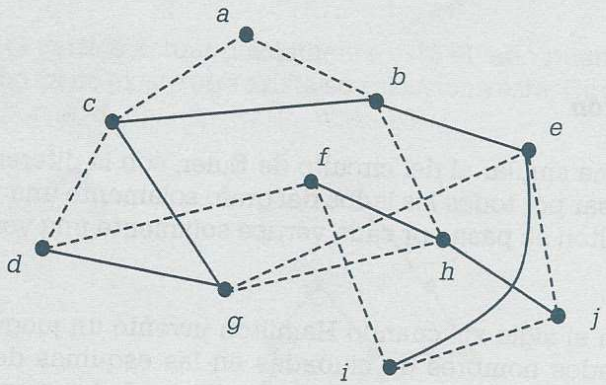
Usando la definición de suma y producto entre matrices con elementos complejos se definen la suma y el producto entre cuaterniones, y la estructura algebraica que se obtiene es la de un campo con producto no conmutativo, esto es, la de un anillo con división o campo asimétrico.

Además de su importancia intrínseca y de su utilidad en la teoría de números, los cuaterniones se aplican en electromagnetismo, teoría de la relatividad y mecánica cuántica.

Ejemplo 7.13. Determinar, si es posible, un circuito de Hamilton en el siguiente grafo.



Solución. El circuito de Hamilton es como se indica en el grafo.



Como se ve, el circuito de Hamilton es $\{a, b, h, g, e, j, i, f, d, c, a\}$, la línea punteada.

En general los algoritmos para obtener un circuito de Hamilton en un grafo tardan un tiempo exponencial, y están en función del número de vértices y aristas.

7.6 Isomorfismo

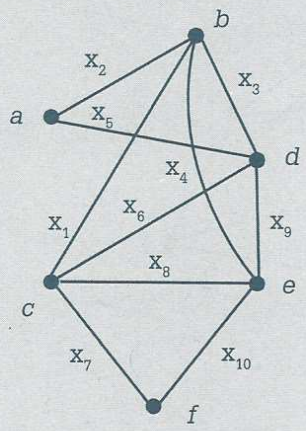
Se dice que dos grafos G_1 y G_2 son isomorfos cuando teniendo apariencia diferente realmente son iguales, porque coinciden en:

- El número de lados.
- El número de vértices.
- El conjunto de valencias.
- Ser o no conexos.
- El número de circuitos de longitud *n*.
- Tener o no circuito de Euler.

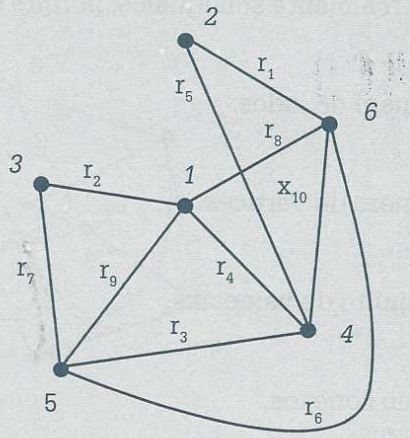
Esto implica que todos los vértices de G_1 tienen un vértice equivalente en G_2, y que todas las aristas del grafo G_1 tienen una arista equivalente en G_2. La consecuencia de esto es que con las propiedades de un vértice en G_1 como argumentos, y por medio de una función biyectiva *f*, se puede obtener un vértice en G_2 con las mismas propiedades. También teniendo las propiedades de un vértice en G_2 como argumentos, y por medio de una función biyectiva *g*, se puede obtener un vértice en G_1 con las mismas propiedades.

Por otro lado, se sabe que dos grafos G_1 y G_2 son isomorfos si y sólo si para alguna ordenación de vértices y sus aristas, sus matrices de incidencia son iguales.

Ejemplo 7.14. Determinar si los grafos siguientes G_1 y G_2 son isomorfos, haciendo coincidir sus matrices de incidencia, manteniendo una matriz estática y realizando intercambios de filas y/o columnas en la otra matriz.



G_1



G_2

Solución. Se tiene que las matrices de incidencia son:

$$M_{G_1} = \begin{array}{c|cccccccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} \\ \hline a & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ b & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ c & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ d & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ e & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ f & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}$$

$$M_{G_2} = \begin{array}{c|cccccccccc} & r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 & r_9 & r_{10} \\ \hline 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 5 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 6 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}$$

Considerando estática la matriz de G_1 y realizando intercambios en la de G_2 se tiene:

$$M_{G_2} = \begin{array}{c|cccccccccc} & r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 & r_9 & r_{10} \\ 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 4 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 6 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 5 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} = \begin{array}{c|cccccccccc} & r_1 & r_{10} & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 & r_9 & r_2 \\ 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 4 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 6 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}$$

$$M_{G_2} = \begin{array}{c|cccccccccc} & r_1 & r_{10} & r_4 & r_3 & r_5 & r_8 & r_2 & r_6 & r_9 & r_7 \\ 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 4 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 6 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} = \begin{array}{c|cccccccccc} & r_1 & r_4 & r_{10} & r_3 & r_5 & r_8 & r_7 & r_6 & r_9 & r_7 \\ 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 4 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 6 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}$$

$$M_{G_2} = \begin{array}{c|cccccccccc} & r_5 & r_4 & r_{10} & r_3 & r_1 & r_8 & r_2 & r_6 & r_9 & r_7 \\ 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 4 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} = \begin{array}{c|cccccccccc} & r_5 & r_4 & r_{10} & r_3 & r_1 & r_8 & r_2 & r_9 & r_6 & r_7 \\ 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 4 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}$$

$$M_{G_2} = \begin{array}{c|cccccccccc} & r_5 & r_4 & r_{10} & r_3 & r_1 & r_8 & r_2 & r_9 & r_6 & r_7 \\ 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 6 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} = \begin{array}{c|cccccccccc} & r_4 & r_5 & r_{10} & r_3 & r_1 & r_8 & r_2 & r_9 & r_6 & r_7 \\ 2 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}$$

Con lo anterior es posible comprobar que efectivamente G_1 y G_2 son isomorfos.

Es muy complicado demostrar que dos grafos son isomorfos haciendo intercambios de filas y/o columnas hasta que las matrices de incidencia coincidan, y lo es aún más a medida que aumenta el número de vértices y aristas, independientemente de si esto se hace en forma manual o en la computadora.

Debido a lo anterior, en lugar de hacer coincidir las matrices lo que se hace es una comparación de las propiedades principales de los grafos de forma que si coinciden en todas ellas se concluye que son isomorfos, pero si existe alguna diferencia (ya sea en el número de vértices, en el conjunto de valencias, o si uno de ellos es conexo y el otro no, o si uno tiene circuito de Euler y el otro no, o bien si uno de ellos tiene más circuitos de longitud *n* que el otro), esto es causa suficiente para determinar que los grafos no son isomorfos.

En la siguiente tabla se muestran las propiedades en donde debe haber coincidencia entre los grafos G_1 y G_2 anteriores para que se consideren isomorfos.

Propiedad	G_1	G_2	Observación
Núm. de vértices	6	6	
Núm. de lados	10	10	
Valencias	2, 4, 4, 4, 4, 2	4, 2, 2, 4, 4, 4	Coinciden en el mismo número de vértices de valencias 2 y 4.
Conexo	Sí	Sí	Ya que para cualquier par de vértices se puede encontrar un camino.
Camino de Euler	No	No	Ya que todos los vértices tienen valencia par.
Circuito de Euler	Sí	Sí	Ya que todos los vértices tienen valencia par y se trata de grafos conexos.
Circuitos de longitud *n*. (En este caso de longitud 3.)	6 a, b, d, a b, e, c, b b, d, c, b b, d, e, b c, d, e, c c, e, f, c	6 1, 3, 5, 1 1, 6, 4, 1 1, 4, 5, 1 1, 5, 6, 1 2, 4, 6, 2 4, 5, 6, 4	En lugar de tener longitud 3, se pudo ver cuántos circuitos tienen longitud 4. Pero en cualquier caso debe coincidir.

Es posible observar que en el caso de valencias, la coincidencia debe ser en el número de nodos de cierta valencia, y cuando se trata del número de circuitos de longitud n deben coincidir en el número y la longitud. Por ejemplo, si uno de ellos tiene un circuito de longitud 5 y el otro no, eso es causa suficiente para concluir que no son isomorfos. Las propiedades de "Camino de Euler" y "Circuito de Euler" son relativamente fáciles de determinar. Sin embargo, no es fácil determinar si tienen o no un circuito de Hamilton. Si es posible determinar de una forma relativamente fácil si dos grafos tienen o no circuito de Hamilton sería recomendable hacerlo, pero en caso contrario, con que sean iguales en las características que contiene la tabla anterior es suficiente.

7.7 Grafos planos

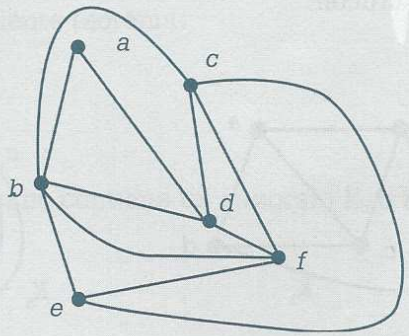
Un grafo plano es aquel que se puede dibujar en un solo plano y cuyas aristas no se cruzan entre sí.

Por otro lado, la ecuación de Euler

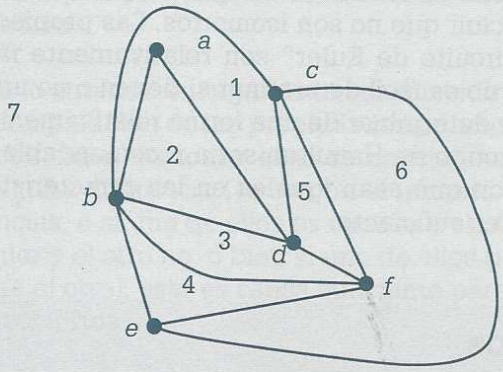
$$A = L - V + 2$$

en donde A = número de áreas, L = número de lados y V = número de vértices, es válida para un grafo plano y conexo.

Ejemplo 7.15. El siguiente grafo es un ejemplo de grafo plano y conexo:



Se consideran áreas a todas aquellas secciones cerradas {1, 2, 3, 4, 5, 6} y a la sección que rodea el grafo {7}, como se muestra en la siguiente figura.

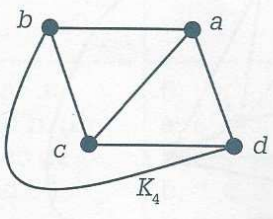


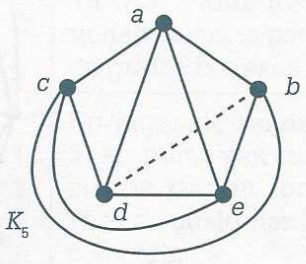
De aquí se tienen los valores A = 7, L = 11 y V = 6, los cuales satisfacen la fórmula de Euler: 7 = 11 – 6 + 2.

Otra propiedad importante de un grafo plano es que cada lado es frontera máximo de dos áreas. Así en el grafo del ejemplo 7.15 se tiene que el lado *c-f* es frontera de las áreas 5 y 6, y el lado *b-c* lo es de las áreas 1 y 7.

De acuerdo con lo anterior, si se tiene un grafo en el que la igualdad A = L – V + 2 no se cumple o bien uno de los lados es frontera de más de dos áreas, entonces con esto es más que suficiente para establecer que el grafo considerado no es plano.

Existen grafos importantes que se vieron anteriormente, por ejemplo K_4 es un grafo plano ya que se puede dibujar sin que sus aristas se crucen, pero K_5 es un grafo no plano ya que no hay forma en que por lo menos un par de aristas se crucen.

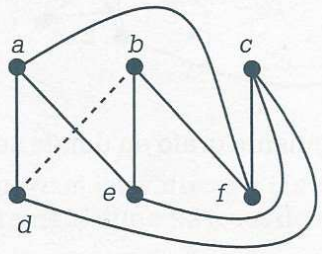




En el grafo K_4 es fácil observar que cada uno de los lados es frontera de máximo dos áreas y que la ecuación de Euler se cumple ya que A = 4, L = 6 y V = 4.

En relación con K_5 se observa que si el grafo se pudiera dibujar en forma plana, la línea punteada sería frontera de las áreas (*b, d, e, b*), (*a, b, d, a*) y (*a, b, d, c, a*). Incluso es complicado delimitar las áreas.

Otro grafo importante que no es plano, es el grafo bipartido completo $K_{3,3}$ que se muestra a continuación:



Aquí se observa que no es posible dibujar en forma plana un $K_{3,3}$.

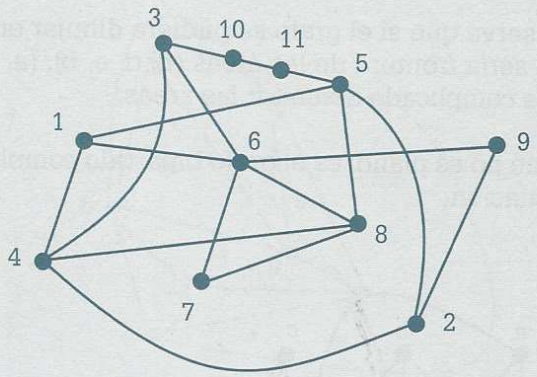
Entre más grande y complejo sea el grafo, es más difícil identificar las áreas y también es más complicado verificar si efectivamente cada uno de sus lados es frontera de máximo dos áreas adyacentes, por lo tanto, prácticamente es imposible determinar si la ecuación de Euler se cumple y en consecuencia es difícil determinar si el grafo es plano o no.

En lugar de determinar si la ecuación de Euler se cumple y si cada uno de los lados es frontera de máximo dos áreas, se utilizan $K_{3,3}$ y K_5 como patrones para demostrar que otros grafos más complejos no son planos. Esto es posible eliminando lados hasta encontrar oculto un grafo no plano $K_{3,3}$ o K_5 como lo establece el siguiente teorema:

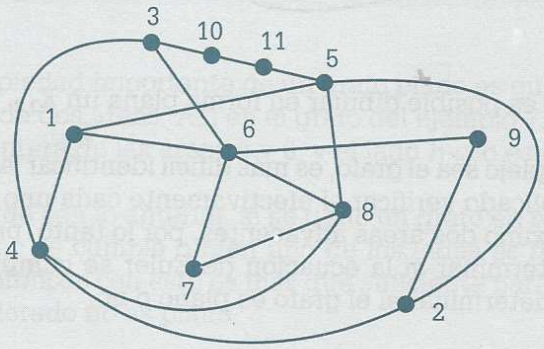
Teorema de Kuratowski

Un grafo G es plano si y sólo si no contiene un subgrafo K_5 o $K_{3,3}$.

Ejemplo 7.16. Si se desea dibujar en forma plana el siguiente grafo



una buena aproximación es el siguiente grafo en donde se han movido de lugar los lados 3-4, 5-2 y 4-8:

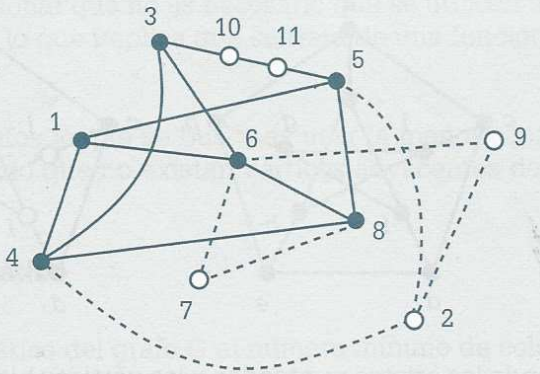


Sin embargo, el grafo aún no es plano ya que se cruzan dos pares de lados.

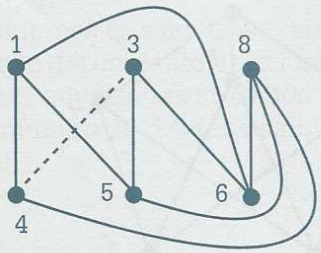
Es posible que cambiando de lugar los vértices (algo que también es permitido) se pueda lograr que sólo se crucen un par de lados, pero nunca se podrá dibujar en forma plana ya que el grafo contiene en su interior un grafo $K_{3,3}$ que se sabe que no es plano. Esto se puede observar si se eliminan los lados punteados 5-2, 6-9, 7-8, 2-4, 2-9 y 6-7, y los vértices 2, 7 y 9 de acuerdo con la siguiente figura.

Cuando se busca si G contiene un grafo no plano en su interior como $K_{3,3}$ y K_5 se eliminan los vértices de valencia dos, como fue el caso de los nodos 10 y 11 pero no necesariamente se elimina la arista o bien pueden eliminarse las aristas, como ocurre con los vértices 9 y 7. También es válido

eliminar todas las aristas y los vértices que permitan ver claramente que el grafo G contiene en su interior un grafo $K_{3,3}$ y K_5 y que por lo tanto no es plano como se muestra en la siguiente figura



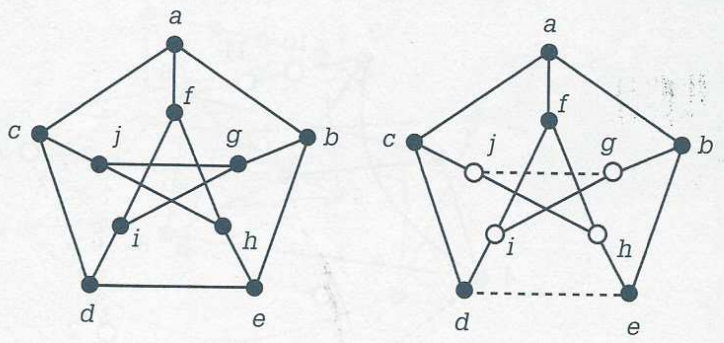
Dibujando el grafo con los vértices y las aristas que se conservan, se puede observar claramente que se trata de un grafo $K_{3,3}$, como se muestra a continuación:



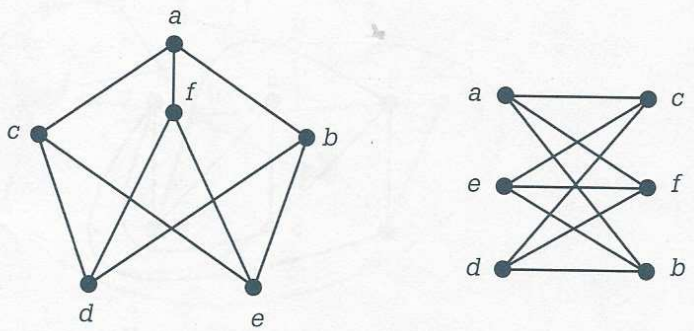
Es obvio que si un grafo tan pequeño como $K_{3,3}$ no es plano, menos lo va a ser uno más grande que contenga dentro de él a un $K_{3,3}$.

Hay que tener presente que para determinar si un grafo tiene en su interior un grafo K_5, éste deberá tener cinco o más vértices y cada uno de ellos una valencia de cuatro o más. Así mismo, para que un grafo pueda tener en su interior un $K_{3,3}$ como mínimo debe contar con seis vértices y cada uno de ellos al menos debe tener valencia tres, además de que en el K_5 todos los vértices están relacionados entre ellos y en el $K_{3,3}$ hay dos conjuntos de vértices en donde los de un conjunto están relacionados con los del otro y entre elementos de un mismo conjunto no existe lado que los una.

El siguiente grafo se conoce como de Petersen y se trata de un grafo no plano porque contiene dentro de él un subgrafo $K_{3,3}$ como se muestra a continuación



Eliminando las aristas punteadas y los vértices blancos se tiene el siguiente subgrafo $K_{3,3}$ que se sabe que no es plano, por lo tanto el grafo de Petersen tampoco es plano:



7.8 Coloración de grafos

Sea G(V,A) un grafo y sea C un conjunto de colores. La coloración de los vértices V del grafo usando un color del conjunto C se encuentra dada por la función

$$f: V \rightarrow C \text{ tal que } \forall\ v_1, v_2 \in V \text{ adyacentes}$$

$$f(v_1) \neq f(v_2)$$

Esto significa que cuando se lleva a cabo la coloración cada par de vértices adyacentes v_1 y v_2 del grafo deberán estar iluminados con un color diferente.

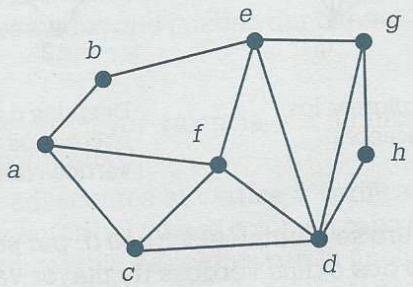
A cada vértice del conjunto V, la función f le asocia un color del conjunto C. Es importante mencionar que no es necesario que se utilicen todos los colores del conjunto C, lo que implica que se trata de una función que no es suprayectiva.

En la coloración de grafos lo que se busca es usar la menor cantidad de colores posible, cuidando que no existan vértices adyacentes del mismo color.

7.8.1 Número cromático

Se llama número cromático del grafo G al número mínimo de colores con que se puede colorear un grafo, cuidando que los vértices adyacentes no tengan el mismo color. El número cromático se indica de la siguiente como $\chi(G)$.

Ejemplo 7.17. Considérese que se desea iluminar el siguiente grafo G y que se dispone para ello del conjunto de colores C = {1, 2, 3, 4, 5}, teniendo en cuenta que vértices adyacentes no deben tener el mismo color y que se debe usar el menor número de colores posible para encontrar el número cromático.

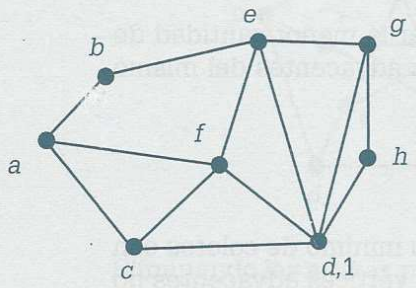


G

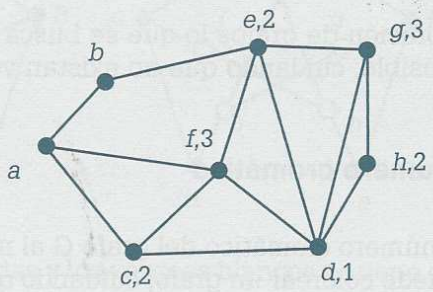
Solución

Se selecciona el vértice que tenga mayor valencia y se ilumina de un color de los disponibles en el conjunto C, después se colorean los vértices adyacentes a él con un color diferente, verificando que no se presenten

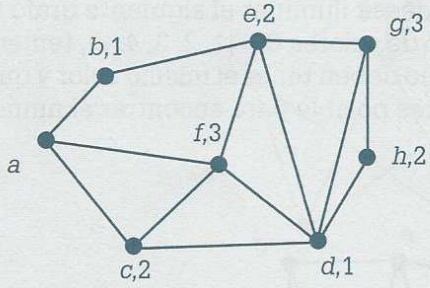
vértices adyacentes iluminados con un mismo color. Luego se toma el vértice que tenga mayor valencia de este grupo de vértices coloreados en segunda opción y se colorean sus vértices adyacentes, cuidando que no existan vértices adyacentes del mismo color y usando el menor número de colores posibles para iluminar el grafo. Lo que se obtiene son los siguientes estados del grafo:



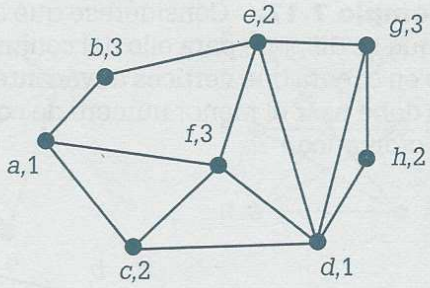
Después de colorear el vértice d con el color 1



Después de colorear los vértices adyacentes a d



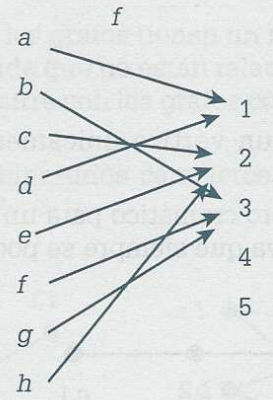
Después de colorear los vértices adyacentes faltantes a e



Después de hacer los ajustes necesarios y colorear los vértices faltantes

En este ejemplo primero se iluminó el vértice d, por ser el que tiene valencia mayor (en caso de dos o más vértices de mayor valencia se selecciona cualquiera de ellos). Después se colorearon los vértices adyacentes a él cuidando que no existan vértices con el mismo color, por ejemplo si el vértice c ya está iluminado con el color 2, el vértice f no puede iluminarse también con el color 2 o el color 1 porque es adyacente al vértice c y al vértice d. Nuevamente se seleccionó un vértice de mayor valencia que aún no tenía todos sus vértices adyacentes iluminados, en este caso el vértice e, y se coloreó el vértice adyacente faltante b con el color 1. Finalmente fue necesario realizar los ajustes necesarios usando la menor cantidad de

colores para lo cual fue necesario cambiar el color del vértice b al color 3 para que el vértice a fuera iluminado con el color 1. Se encontró que el número máximo de colores utilizado para iluminar el grafo G son 3, por lo tanto su número cromático es X(G)=3. Hay que observar que no fue necesario utilizar todos los colores del conjunto C de forma que la coloración usando la función f: V → C se puede representar de la siguiente manera



Aquí se observa que f: V → C no es una función suprayectiva, considerando que $\forall\ v_1, v_2 \in V$ adyacente $f(v1) \neq f(v_2)$.

La coloración de grafos es un problema NP-Computable y esto significa que no hay procedimientos eficientes para llevar a cabo esta tarea, sin embargo existen métodos aproximados que pueden dar buenos resultados. Uno de ellos es el siguiente:

1) Seleccionar el vértice de mayor valencia v e iluminarlo con un color cualquiera del conjunto C.
2) Colorear los vértices adyacentes al vértice v verificando que no existan vértices adyacentes del mismo color. En caso de ser necesario llevar a cabo intercambio de colores con la finalidad de usar la menor cantidad de ellos. Si ya están coloreados todos los vértices del grafo finalizar, en caso contrario continuar con el paso 3.
3) Seleccionar el vértice v de mayor valencia que ya esté coloreado y que todavía tenga vértices adyacentes sin colorear. Regresar al paso 2.

Se recomienda colorear del mismo color tantos vértices como sea posible e iluminar al mismo tiempo los vértices que compartan nodos vecinos.

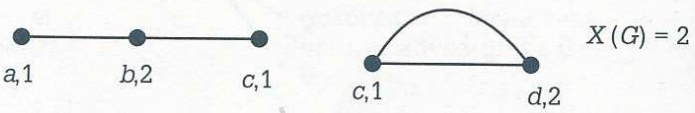
7.8.2 Características del número cromático

El número cromático posee las siguientes siete características fundamentales:

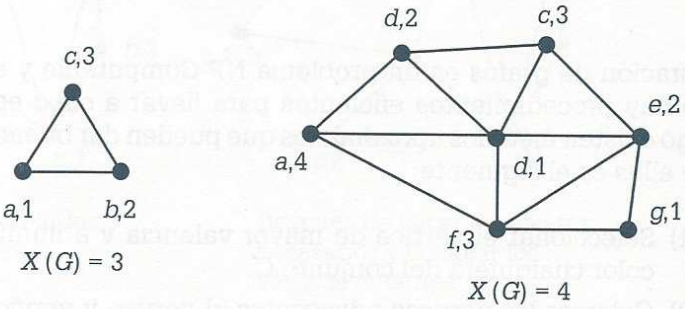
a) Un grafo G tiene número cromático X(G)=1 si y sólo si no tiene aristas:

ya que un vértice únicamente puede iluminarse de un solo color.

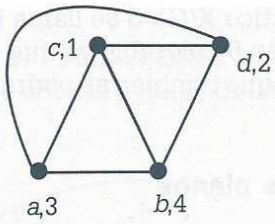
b) El número cromático para un camino o un ciclo de longitud 2 es X(G)=2 ya que siempre se podrán alternar los colores:



c) Si el grafo G tiene un ciclo de longitud impar entonces X(G) ≥ 3:



d) El número cromático del grafo completo K_n es $X(K_n)=n$, considerando que la característica de este tipo de grafo es que todos los vértices son adyacentes entre sí. Por jemplo K_4 tiene número cromático 4, porque es el menor número de colores con el que se puede iluminar, cuidando que vértices adyacentes no tengan el mismo color.

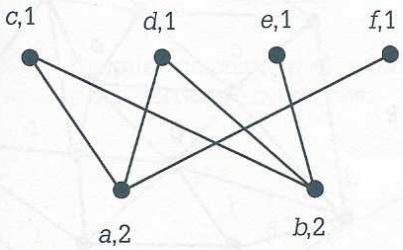


K_4

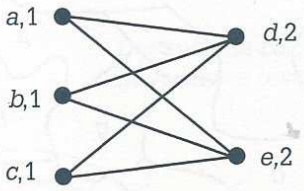
$X(K_4) = 4$

e) En general la mayoría de los grafos tienen un número cromático $X(G) \leq n$ porque se entiende que no están relacionados todos los vértices entre sí, como ocurre con los grafos completos de n vértices K_n.

f) Los grafos bipartidos o bipartidos completos ($K_{n,m}$), tienen un número cromático X(G)=2.

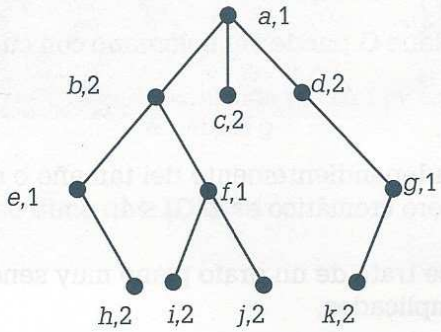


Grafo bipartido



Bipartido completo $K_{2,3}$

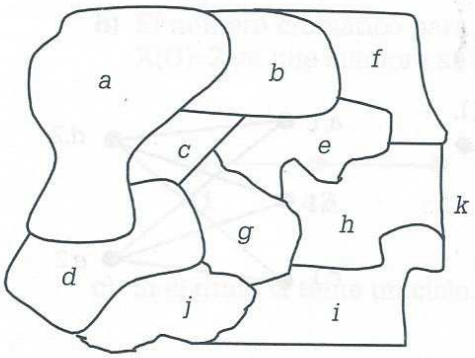
g) Todos árboles de cualquier orden tienen número cromático X(G) = 2 o bien se dice que son 2-colorables.



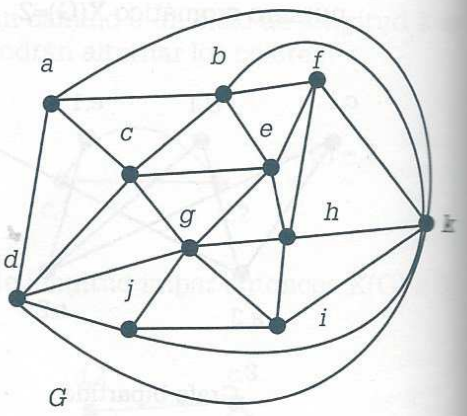
Un grafo G con número cromático X(G)=5 se llama 5-colorable y se infiere que si es 5-colorable es también 6-colorable ya que si un grafo G se puede iluminar con 5 colores es obvio que también se podrá colorear con 6, aunque su número cromático es 5.

7.8.3 Coloración de grafos planos

Todo mapa puede ser representado por un grafo plano, en donde cada parte del mapa representa un vértice. En el grafo también se debe incluir la parte que rodea al mapa como un vértice adicional, ya que es adyacente a varias partes del mapa. Dos partes del mapa que son vecinas se representan como vértices adyacentes en el grafo. A continuación se muestra un pequeño mapa con su correspondiente grafo plano.



Mapa



G

En relación con el coloreado de un grafo se tiene el siguiente teorema:

Teorema de los cuatro colores. (Appel y Haken)

Cualquier grafo plano G puede ser coloreado con cuatro colores diferentes.

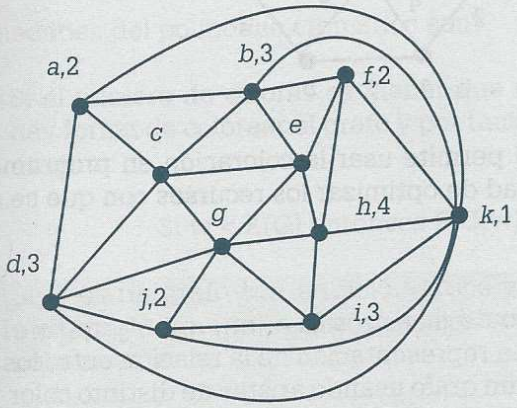
Esto significa que independientemente del tamaño o complicación de un grafo plano, su número cromático es $X(G) \leq 4$.

Será menor de 4 si se trata de un grafo plano muy sencillo y máximo será de 4 para grafos complicados.

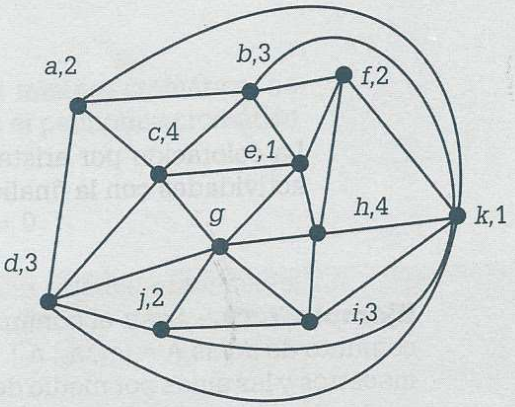
Problema de los cuatro colores

Este problema surgió a mediados del siglo XIX cuando Francis Guthrie (1831-1899), luego de colorear el mapa de Inglaterra con 4 colores, planteó la cuestión de si todos los grafos planos se podrían colorear con solamente 4 colores. El problema permaneció sin ser resuelto por más de cien años hasta que en 1976 Kenneth Appel y Wolfgang Haken demostraron con ayuda de una computadora que efectivamente todo grafo plano tiene un número cromático menor o igual a 4.

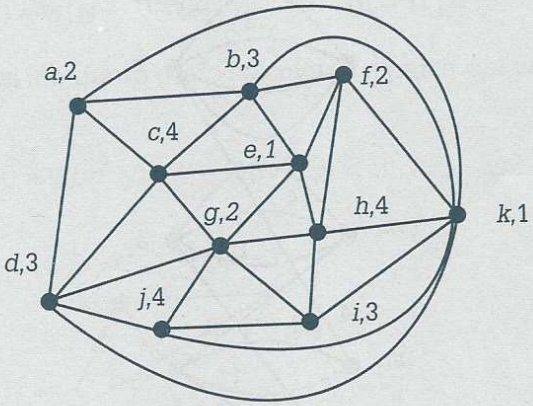
Ejemplo 7.18. Para colorear y obtener el número cromático del grafo plano G que se obtuvo a partir del mapa anterior, se procede de la siguiente manera:



Después de colorear el vértice k y sus vértices adyacentes.



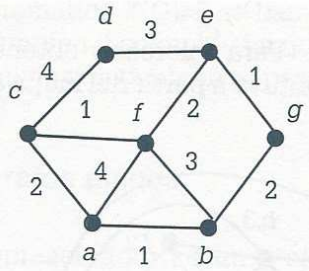
Después de colorear los vértices adyacentes restantes de b.



Cambiando de color j y coloreando el vértice g

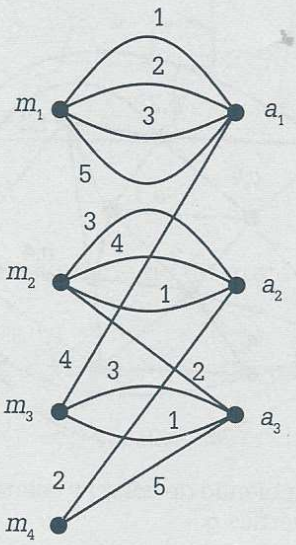
Por tanto se tiene que X(G) = 4.

También es posible la coloración de los grafos iluminando las aristas en lugar de los vértices. El siguiente grafo tiene iluminadas sus aristas con los colores {1, 2, 3} verificando que dos aristas de un mismo color no incidan sobre un mismo vértice.



La coloración por aristas permite usar la coloración en programación de actividades con la finalidad de optimizar los recursos con que se dispone.

Ejemplo 7.19. Sean el conjunto de maestros $M = \{m_1, m_2, m_3, m_4\}$ y el conjunto de aulas $A = \{a_1, a_2, a_3\}$. La representación de la relación entre los maestros y las aulas por medio de un grafo usando aristas de distinto color, en donde el color de la arista indica la hora a la cual se imparte la clase, es la siguiente:



Aquí se tiene que

1: Clase de 8:00-9:00.
2: Clase de 9:00-10:00
3: Clase de 10:00-11:00
4: Clase de 11:00-12:00
5: Clase de 12:00-13:00

7.8.4 Polinomio cromático

Al número de formas en que se puede colorear el grafo G usando w colores se le llama polinomio cromático de G y se denota como P(G,w).

Las propiedades del polinomio cromático son:

a) Si el número de colores es menor que el número cromático, no hay forma de colorear el grafo y por tanto el polinomio cromático es cero:

$$\text{Si } w < X(G) \text{ entonces } P(G,w) = 0$$

b) Si G es un grafo con un solo vértice y sin aristas, entonces el número de formas en que se puede colorear ese vértice con w colores es w:

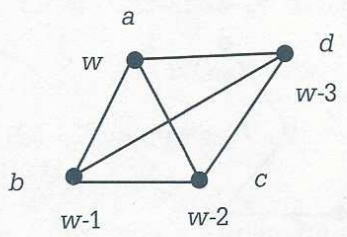
$$P(G,w) = w$$

c) Si G es un grafo completo de n vértices (K_n) entonces:

$$P(G,w) = P(K_n,w) = w(w-1)(w-2)\ldots[w-(n-1)] = w!$$

Por ejemplo, si el siguiente grafo se colorea a partir del vértice a se tiene que

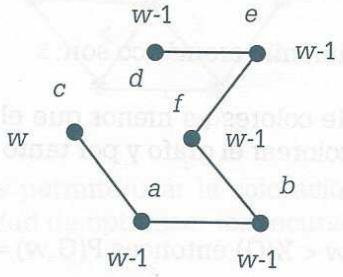
$$P(K_4,w) = w(w-1)(w-2)(w-3) = 4!$$



K_4

d) El número cromático para un camino simple es:

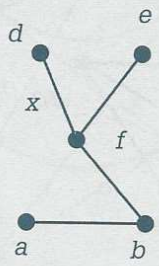
$$P(G,w) = w(w-1)(w-1)\ldots(w-1) = w(w-1)^{n-1}$$



e) Si G está integrado por subgrafos, es necesario descomponer el grafo en subgrafos cuyo polinomio cromático sea conocido. La descomposición se lleva a cabo eliminando aristas para obtener un subgrafo G_x y obtener de éste un subgrafo conocido que se llamará G'_x. En caso de ser necesario, se vuelve a descomponer G_x hasta obtener solamente subgrafos conocidos. La expresión que permite determinar el polinomio cromático de G es:

$$P(G,w) = P(G_x,w) - P(G'_x,w)$$

Ejemplo 7.20. Determinar el polinomio cromático del siguiente grafo G:

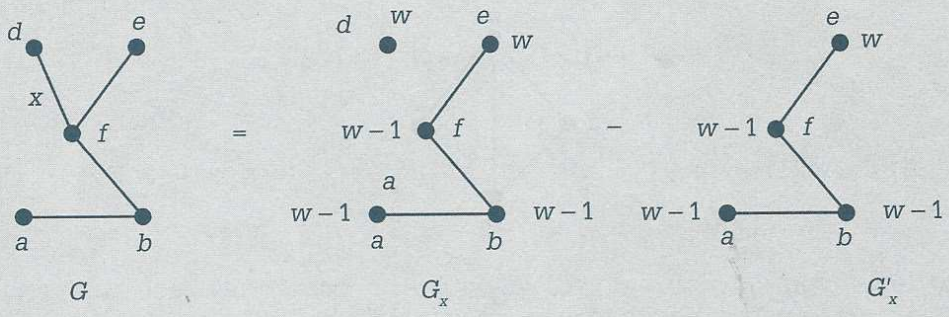


G

Solución

El polinomio cromático del grafo anterior no es conocido ya que no es un grafo integrado por un solo vértice sin aristas, tampoco es un camino simple, ni un grafo completo de n vértices K_n, por lo tanto se debe descomponer en subgrafos cuyo polinomio cromático sea conocido.

Considérese que se elimina la arista x que conecta a los vértices d y f para obtener el subgrafo G_x. Hay que observar también que G'_x es subgrafo de G_x.



Sustituyendo valores en

$$P(G,w) = P(G_x,w) - P(G'_x,w)$$

se tiene que

$$P(G,w) = w\,[w(w\text{-}1)(w-1)(w\text{-}1)] - w(w-1)(w-1)(w-1)$$

$$= w\,(w-1)(w-1)(w-1)(w-1) = w(w-1)^4$$

Con el polinomio cromático, también es posible obtener el número cromático de G. Hay que observar que para un valor de w = 1 se tiene que P(G,w) = 0 lo que quiere decir que no es posible colorear ninguna vez el grafo. Para w = 2 el número de formas en que se puede colorear G es:

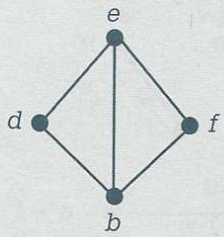
$$P(G,2) = 2(2-1)^4 = 2$$

Por lo tanto X(G) = 2 ya que es el menor valor de w con el que se puede colorear el grafo G.

Por otro lado, el número de formas que se puede colorear G con w = 5 es:

$$P(G,5) = 5(5-1)^4 = 1280$$

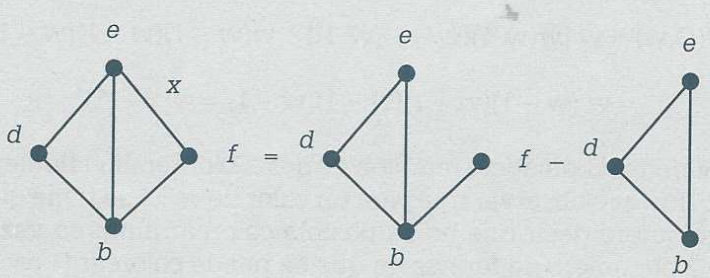
Ejemplo 7.21. Para el siguiente grafo G determinar el polinomio cromático P(G,w), el número cromático X(G) y el número de formas distintas en que se puede colorear dicho grafo si w = 5.



G

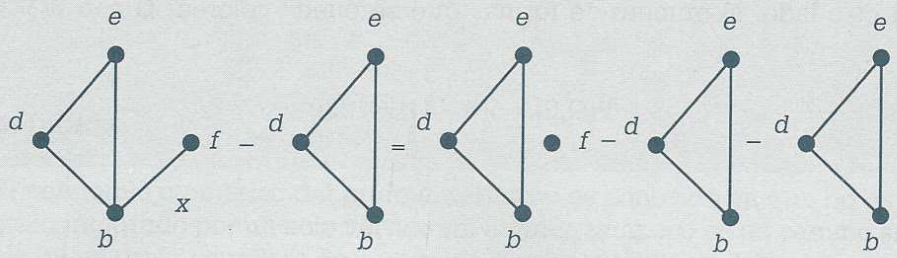
Solución

Debido a que G no es un grafo con el cual se pueda obtener directamente el polinomio cromático, es necesario descomponerlo en G_x y G'_x como se muestra en la figura.



G G_x G'_x

Como del subgrafo G_x tampoco es posible obtener su polinomio cromático, se descompone nuevamente:



G_x G'_x G_x G'_x G'_x

Hay que observar que ahora ya es posible obtener el polinomio cromático, considerando que G_x está integrado por un subgrafo completo de longitud 3 y un vértice sin aristas y que G'_x es dos veces un grafo completo K_3.

Se tiene que

$$P(G,w) = P(G_x,w) - (G'_x,w) - (G'_x,w)$$

Sustituyendo:

$$P(G,w) = w\,[w(w\text{-}1)(w-2)] - w(w-1)(w-2) - w(w-1)(w-2)$$

$$= w\,(w-1)(w-2)(w-1-1) = w(w-1)(w-2)^2$$

Hay que observar que con $w < 3$ no es posible colorear ninguna vez el grafo, por lo tanto $P(G,w) = 0$ y el valor de w mínimo con el cual se puede colorear G es $w = 3$ y por tanto el número cromático es $X(G) = 3$.

Si $w = 5$, entonces el número de maneras diferentes en que es posible colorear G es:

$$P(G,5) = 5(5-1)(5-2)^2 = 180$$

7.9 Aplicaciones de los grafos

Actualmente nada funciona de manera individual, por el contrario existe una relación entre los distintos elementos que integran un sistema, trabajo o equipo.

Puesto que los grafos permiten modelar todo aquello que está relacionado, es evidente que su aplicación es muy amplia y variada.

7.9.1 Reconocimiento de patrones mediante grafos de similaridad

Los grafos de similaridad permiten agrupar información con características semejantes. Esto implica formar subgrafos en donde los vértices de un subgrafo están relacionados entre sí, pero no tienen relación con los vértices del otro subgrafo, ya que no son similares. Una aplicación de este tipo de grafos se encuentra en el reconocimiento de patrones, en donde se agrupa información con propiedades muy parecidas de tal manera que se puedan detectar enfermedades como el cáncer, al agrupar conjuntos de células que comparten características similares. Otra aplicación se encuentra en la cartografía, en donde grupos de píxeles (pequeños cuadritos de una imagen) se agrupan porque son muy parecidos y por lo tanto se consideran similares.

En este tipo de grafo se debe definir una función que permita determinar la similaridad que existe entre los vértices, principalmente la distancia entre sus características. Una función muy usada para determinar la distancia es:

$$S(P_x - P_y) = \sum_{i=1}^{n} |P_{x,i} - P_{y,i}| = |P_{x,1} - P_{y,1}| + \\ + |P_{x,2} - P_{y,2}| + \ldots\ldots\ldots + |P_{x,n} - P_{y,m}|$$

en donde $P_{x,n}$ es la propiedad n del punto P_x y $P_{y,m}$ es la propiedad m del punto P_y. Con la función anterior es posible determinar la distancia que existe entre las propiedades de los vértices x, y. Un valor grande de $S(P_x - P_y)$ indica que no existe similaridad entre los vértices x, y mientras que un valor pequeño significa que son muy parecidos o similares.

Pero además de la función anterior, es necesario un valor referencial para discriminar la información que en este caso se llamará coeficiente de inferencia *C*. El valor de *C* se puede seleccionar de acuerdo con la experiencia que se tenga en el campo, o bien por medio de una prueba piloto. Si *C* es grande, la similaridad es poco exacta, sin embargo para un valor de *C* pequeño la similaridad es muy fuerte.

Una vez que se tiene $S(P_x - P_y)$ para todos los puntos (vértices) en cuestión, se forman los grafos similares por medio del siguiente criterio: si $S(P_x - P_y) \leq C$, se traza una arista entre los vértices P_x y P_y y se dice que P_x y P_y son similares. Los grafos similares forman un subgrafo y los no similares otro distinto, como se muestra en el siguiente ejemplo.

Ejemplo 7.22. La siguiente tabla muestra las características que tienen las diferentes partes de tejido de igual sección transversal de un análisis de mama, de acuerdo con "temperatura", "intensidad de color" e "inflamación".

Parte (P)	*Temperatura (T)*	*Color (Co)*	*Inflamación (I)*
1	39	48	7
2	37	42	6
3	40	47	8
4	36	41	5
5	39	49	9

Determinar los grafos de similaridad para un coeficiente de inferencia C = 5.

Solución

Lo primero que se debe de hacer es obtener la distancia entre los puntos P_x y P_y por medio de la siguiente función:

$$S(P_x - P_y) = \sum_{i=1}^{n} |P_{x,i} - P_{y,i}| = |P_{x,1} - P_{y,1}| + |P_{x,2} - P_{y,2}| + \ldots + |P_{x,n} - P_{y,m}|$$

Si x = y se tiene que

$$S(P_1 - P_1) = |P_{1,1} - P_{1,1}| + |P_{1,2} - P_{1,2}| + |P_{1,3} - P_{1,3}|$$
$$= |39 - 39| + |48 - 48| + |7 - 7| = 0$$

Se observa claramente que entre P_1 y P_1 existe una gran similaridad, es más, son iguales. Por lo tanto, no tiene caso obtener la similaridad entre puntos iguales, de forma que se da por hecho que son similares y solamente se registrarán al final en el grafo correspondiente.

Si $x \neq y$

$$S(P_1 - P_2) = |P_{1,1} - P_{2,1}| + |P_{1,2} - P_{2,2}| + |P_{1,3} - P_{2,3}|$$
$$= |39 - 37| + |48 - 42| + |7 - 6| = 9$$

$$S(P_1 - P_3) = |P_{1,1} - P_{3,1}| + |P_{1,2} - P_{3,2}| + |P_{1,3} - P_{3,3}|$$
$$= |39 - 40| + |48 - 47| + |7 - 8| = 3$$

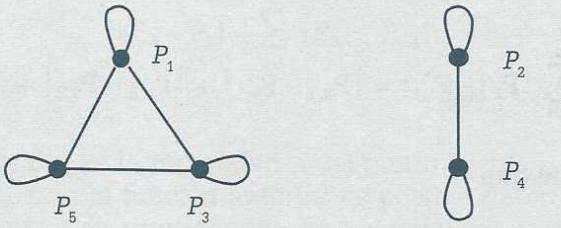
$$S(P_1 - P_4) = |P_{1,1} - P_{4,1}| + |P_{1,2} - P_{4,2}| + |P_{1,3} - P_{4,3}|$$
$$= |39 - 36| + |48 - 41| + |7 - 5| = 12$$

$$S(P_1 - P_5) = |P_{1,1} - P_{5,1}| + |P_{1,2} - P_{5,2}| + |P_{1,3} - P_{5,3}|$$
$$= |39 - 39| + |48 - 49| + |7 - 9| = 3$$

$$S(P_2 - P_3) = |P_{2,1} - P_{3,1}| + |P_{2,2} - P_{3,2}| + |P_{2,3} - P_{3,3}|$$
$$= |37 - 40| + |42 - 47| + |6 - 8| = 10$$

$$S(P_2 - P_4) = |37 - 36| + |42 - 41| + |6 - 5| = 3$$
$$S(P_2 - P_5) = |37 - 39| + |42 - 49| + |6 - 9| = 12$$
$$S(P_3 - P_4) = |40 - 36| + |47 - 41| + |8 - 5| = 13$$
$$S(P_3 - P_5) = |40 - 39| + |47 - 49| + |8 - 9| = 4$$
$$S(P_4 - P_5) = |36 - 39| + |41 - 49| + |5 - 9| = 15$$

Se considera que dos puntos son similares si $S(P_x - P_y) \leq C$ y en este caso se tiene que $S(P_x - P_y) \leq 5$, con lo cual se puede decir que el grafo está dividido en dos partes como se muestra a continuación:



Los lazos en cada uno de los vértices significan la similaridad entre ellos mismos. Con los grafos anteriores se puede considerar que las partes de tejido P_1, P_3, P_5 son similares así como P_2 y P_4 también lo son. Si se considera un mayor daño en las células cuyas características de temperatura, color e inflamación están más alejadas de los valores normales, se puede concluir que las áreas más dañadas son P_1, P_3 y P_5.

7.9.2 Determinación de la ruta más corta mediante grafos ponderados

En un grafo ponderado a las aristas se les asigna un valor al que se le llama *ponderación* y que podría representar la distancia que hay de un nodo a otro, o bien el costo de transportarse de una ciudad a otra.

Determinar la ruta más corta es un problema típico de la teoría de grafos, y consiste en encontrar el camino más corto para ir de una ciudad origen *w* a una ciudad destino *x*. Pueden existir distintas rutas para ir de un nodo a otro, pero el objetivo es encontrar la más corta o bien la más económica, si es que la ponderación representa un costo.

El método más utilizado para encontrar la ruta más corta de un nodo cualquiera *w* a cualquier nodo de la red, es por medio del algoritmo de Dijkstra el cual consta de los siguientes pasos:

1. Seleccionar la ciudad origen *a*.
2. Usar una matriz que tenga como columnas el número de *iteración*, una columna para cada nodo (*a, b, c, d,...*), la columna *Actual* que se utilizará para indicar el vértice que se seleccione en cada iteración y una columna *Seleccionados* para registrar los vértices que se van seleccionado en el proceso, como se muestra en la siguiente tabla:

Iteración	a	b	c	d	...	Actual	Seleccionados

3. Colocar en la matriz la distancia que existe de la ciudad origen a ella misma (cuando se trata de encontrar la distancia de una ciudad a ella misma considerar que es 0). A todas las demás columnas se les coloca ∞ como distancia.

Iteración	a	b	c	d	...	Actual	Seleccionados
0	0	∞	∞	∞			

4. Colocar en la columna *Actual* el vértice que tenga la distancia más corta de entre todos los nodos (es obvio que en esta primera iteración es el nodo origen). En la columna *Seleccionados* registrar dicho nodo escogido para ya no volverlo a elegir. (En nuestro caso registramos esta distancia en tipo bold, negro y subrayado.)

Iteración	a	b	c	d	...	Actual	Seleccionados
0	0	∞	∞	∞	...	a	a

5. Registrar en la columna de cada uno de los nodos la distancia más corta que resulta de sumar la *distancia registrada en el nodo actual + distancia a los vértices adyacentes a él*, y seleccionar la distancia más corta cuyo nodo aún no esté seleccionado de esa fila de la matriz (suponer que $d_1 > d_2$), por lo tanto, la matriz será:

Iteración	a	b	c	d	...	Actual	Seleccionados
0	0	∞	∞	∞	...	a	a
1	0	d_1	d_2	∞	...		

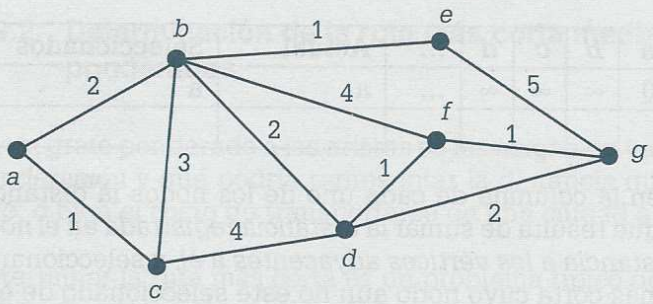
Si el nodo seleccionado tiene una distancia diferente de ∞, que es menor o igual a la que se obtiene de sumar la distancia registrada en la columna del nodo actual + la distancia de ese nodo actual a los nodos adyacentes a él, dejarla tal como está, en caso contrario cambiarla por la nueva suma.

6. Registrar en la columna *Actual* el vértice que tenga la distancia más corta de entre todos los nodos y que no haya sido seleccionado hasta ahora. Además de anotar en la columna *Seleccionados* dicho nodo para ya no volverlo a elegir.

Iteración	*a*	*b*	*c*	*d*	...	Actual	Seleccionados
0	0	∞	∞	∞	...	a	a
1	0	d_1	d_2	∞	...	c	a, c

7. Si ya están todos los vértices seleccionados, finalizar. En caso contrario regresar al paso 5.

Ejemplo 7.23. Considerar que el siguiente grafo se obtiene al observar la comunicación que existe entre las ciudades {*a*, *b*, *c*, *d*, *e*, *f*, *g*} y que la ponderación que tiene cada una de las aristas (carreteras) es la distancia que existe entre dos ciudades cualesquiera. Encontrar la ruta más corta para ir de la ciudad origen (*a*) a las ciudades restantes.



Solución

1. Seleccionar la ciudad origen (en este caso es el nodo *a*).
2. La matriz etiquetada es:

Iteración	*a*	*b*	*c*	*d*	*e*	*f*	*g*	Actual	Seleccionados

3. Después de colocar en la matriz la distancia que existe de la ciudad origen a ella misma y a todas las demás columnas ∞ se tiene que:

Iteración	*a*	*b*	*c*	*d*	*e*	*f*	*g*	Actual	Seleccionados
0	0	∞	∞	∞	∞	∞	∞		

4. Después de registrar en las columnas *Actual* y *Seleccionados* el vértice que tiene la distancia más corta de entre todos los nodos, la matriz queda como:

Iteración	a	b	c	d	e	f	g	Actual	Seleccionados
0	0	∞	∞	∞	∞	∞	∞	a	a

Hay que observar cómo la distancia del vértice elegido se distingue por el subrayado.

5. Una vez que se anotó en la columna de cada uno de los nodos (no seleccionados hasta ahora) la distancia más corta que resulta de sumar la *distancia registrada en el nodo actual* + *distancia a los vértices adyacentes a él* se tiene:

Iteración	a	b	c	d	e	f	g	Actual	Seleccionados
0	0	∞	∞	∞	∞	∞	∞	a	a
1	0	2	1	∞	∞	∞	∞		

6. Después de registrar en la columna *Actual* y agregar a la columna *Seleccionados* dicho nodo:

Iteración	a	b	c	d	e	f	g	Actual	Seleccionados
0	0	∞	∞	∞	∞	∞	∞	a	a
1	0	2	1	∞	∞	∞	∞	c	a, c

Repitiendo el paso 5 resulta:

Iteración	a	b	c	d	e	f	g	Actual	Seleccionados
0	0	∞	∞	∞	∞	∞	∞	a	a
1	0	2	1	∞	∞	∞	∞	c	a, c
2	0	2	1	5	∞	∞	∞		a, c

Repitiendo el paso 6 se obtiene:

Iteración	a	b	c	d	e	f	g	Actual	Seleccionados
0	0	∞	∞	∞	∞	∞	∞	a	a
1	0	2	1	∞	∞	∞	∞	c	a, c
2	0	2	1	5	∞	∞	∞	b	a, c, b

El proceso se repite hasta terminar con la siguiente matriz que muestra la distancia más corta que hay del nodo origen a todos los demás nodos del grafo:

Iteración	*a*	*b*	*c*	*d*	*e*	*f*	*g*	Actual	Seleccionados
0	0	∞	∞	∞	∞	∞	∞	a	a
1	0	2	1	∞	∞	∞	∞	c	a, c
2	0	2	1	5	∞	∞	∞	b	a, c, b
3	0	2	1	4	3	6	∞	e	a, c, b, e
4	0	2	1	4	3	5	8	d	a, c, b, e, d
5	0	2	1	4	3	5	6	f	a, c, b, e, d, f
6	0	2	1	4	3	5	6	g	a, c, b, e, d, f, g

Observar que en la iteración 3 en la columna *d* se registró una menor distancia, porque la suma del nodo actual en ese momento *b* más la distancia de ese nodo actual a *d* dan como resultado 2 + 2 = **4**, mismo que a su vez es menor que el **5** que tenía anteriormente la columna *d*, por tal razón se sustituye un valor mayor por uno menor.

Muchas de las aplicaciones de los grafos se pueden representar por medio de grafos ponderados, y por lo general siempre se desean optimizar los recursos reduciendo las distancias de tal manera que al encontrar la ruta más corta se disminuye el costo, tiempo y por supuesto distancia.

7.10 Resumen

Un grafo es un diagrama que consta de un conjunto de vértices y un conjunto de lados. Los nodos o vértices se indican por medio de un pequeño círculo y se les asigna un número o letra. Los lados o aristas son las líneas que unen un vértice con otro y se les asigna una letra, un número o una combinación de ambos. Cuando dos aristas unen a un mismo par de vértices, se les llama lados paralelos. Un lazo es aquella arista que sale de un vértice y regresa a él mismo. La valencia de un vértice es el número de aristas que salen o entran a un vértice.

Los tipos de grafos más comunes son:

- ***Grafo simple.*** Es aquel que no tiene lazos ni lados paralelos.

- ***Grafo completo de n vértices (K_n).*** Es aquel grafo en donde cada vértice está relacionado con todos los demás, sin lazos ni lados paralelos.

- ***Complemento de un grafo (G').*** Es aquél grafo que le falta al grafo G, para entre ambos formar un grafo completo de n vértices. Dicho grafo no tiene lazos ni lados paralelos.

- ***Grafo bipartido.*** Es aquel que está compuesto por dos conjuntos de vértices *A* y *B* en donde los vértices del conjunto *A* se relacionan con los del *B*, pero entre los vértices de un mismo conjunto no existe arista que los una.

- ***Grafo bipartido completo ($K_{n,m}$).*** Es aquel grafo que está compuesto por dos conjuntos de vértices, *A* y *B*, en donde cada vértice del conjunto A está unido con todos los vértices de *B*, pero entre los vértices de un mismo conjunto no existe arista que los una.

- ***Grafo conexo.*** Es aquél en el que para cualquier par de vértices w, x, distintos entre sí existe un camino para ir de w a x.

- ***Grafos isomorfos.*** Se dice que dos grafos G_1 y G_2 son isomorfos, cuando teniendo apariencia diferente realmente son iguales, porque tienen mismo número de lados, mismo número de vértices, mismo conjunto de valencias, ambos son o no conexos, ambos tienen el mismo número de circuitos de longitud n y ambos tienen o no circuito de Euler.

- ***Grafos de similaridad.*** Son aquellos que permiten agrupar información con características semejantes. Este tipo de grafos es útil en el reconocimiento de patrones, en donde se agrupa información con propiedades muy parecidas.

- ***Grafo plano.*** Es aquel que se puede dibujar en un solo plano y cuyas aristas no se cruzan entre sí. Euler estableció que la ecuación $A = L - V + 2$ se cumple para los grafos planos. Los grafos K_5 y $K_{3,3}$ son grafos no planos importantes y se utilizan como patrones para demostrar que otros más complejos no son planos.

- ***Grafos ponderados.*** Son aquellos en donde a las aristas se les asigna un valor al cual se le llama ponderación y que podría representar la distancia que hay de un nodo a otro, o bien el costo de transportarse de una ciudad a otra. Un problema típico de la teoría de grafos consiste en encontrar el camino más corto para ir de una ciudad origen (w) a una ciudad destino (x). Pueden existir distintas rutas para ir de un nodo a otro, pero el objetivo es encontrar el más corto o bien el más económico si es que la ponderación representa un costo. El método más utilizado para encontrar la ruta más corta es por medio del algoritmo de Dijkstra.

Un grafo se puede representar por medio de una *matriz de adyacencia* M_a (la cual es una matriz cuadrada en donde los vértices del grafo se indican como filas, pero también como columnas de la matriz) o bien por medio de una *matriz de incidencia* M_i (en la que los vértices del grafo se colocan como filas y las aristas como columnas).

La información de un grafo se puede recorrer de diferentes maneras y de acuerdo con sus características propias estos recorridos reciben respectivamente alguno de los siguientes nombres:

- ***Camino.*** Es una sucesión de lados que van de un vértice x a un vértice w.

- ***Circuito (Ciclo).*** Es un camino que regresa al mismo vértice de donde salió.

- ***Circuito simple de longitud n.*** Es aquel camino del vértice w al vértice w que solamente tiene un ciclo en la ruta que sigue.

- ***Camino simple de longitud n.*** Es una sucesión de lados que van de un vértice x a un vértice w, en donde los lados que componen dicho camino son distintos e iguales a n.

Algunos recorridos importantes en los grafos son los siguientes:

- ***Camino de Euler.*** Es aquel camino que recorre todos los vértices pasando por todas las ramas solamente una vez. Un camino de Euler deberá iniciar y terminar en vértices de valencia impar.

- ***Circuito de Euler.*** Es aquel ciclo que recorre todos los vértices pasando por todos los lados solamente una vez. Un grafo tiene circuito de Euler solamente si es conexo y todos su vértices tienen valencia par.

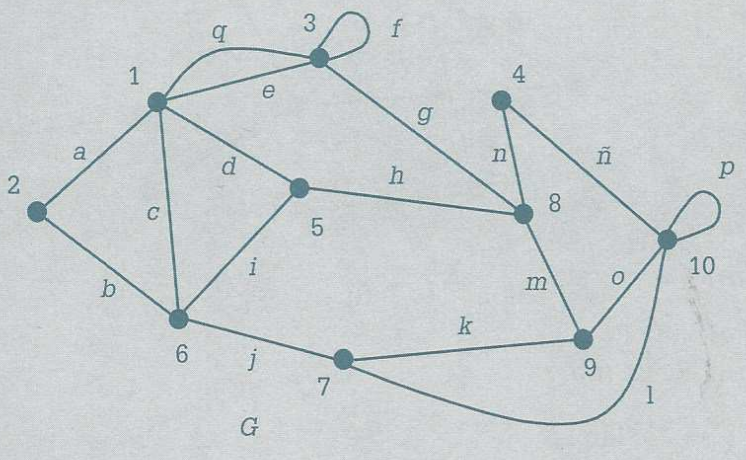
- ***Circuito de Hamilton.*** Es aquel circuito que pasa por cada vértice solamente una vez.

El número cromático de un grafo X(G) es el mínimo de colores necesarios para colorear dicho grafo de tal forma que vértices adyacentes no estén iluminados del mismo color. Todo grafo plano puede ser coloreado con máximo cuatro colores, de acuerdo con el teorema de Appel y Haken. Al número de formas en que se puede colorear el grafo G usando para ello w colores se llama polinomio cromático P(G,w).

El uso fundamental de los grafos son las redes carreteras, telefónicas, eléctricas, de agua potable, de alcantarillado, de computadoras, de cartografía y de distribución de tareas entre otras, y lo que siempre se busca es optimizar los recursos de dichas redes, reducir costos, disminuir distancias o aumentar la velocidad de comunicación. Por otro lado, los grafos permiten ilustrar estructuras químicas, organigramas de una empresa o algoritmos en el área de la computación.

7.11 Problemas

7.1 Considérese el siguiente grafo G:



a) ¿Es grafo simple?

b) ¿Es grafo K_n?

c) ¿Es grafo $K_{n,m}$?

d) ¿Es grafo conexo?

e) ¿Es grafo plano? ¿En caso de ser plano, se cumple la ecuación de Euler $A = L - V + 2$?

f) ¿Tiene un camino de Euler?

g) ¿Tiene un circuito de Euler?

h) ¿Tiene circuito de Hamilton?

i) Obtener:

- El conjunto de vértices (V).
- Conjunto de aristas (A).
- Conjunto de lazos (L).
- Conjunto de lados paralelos (P).

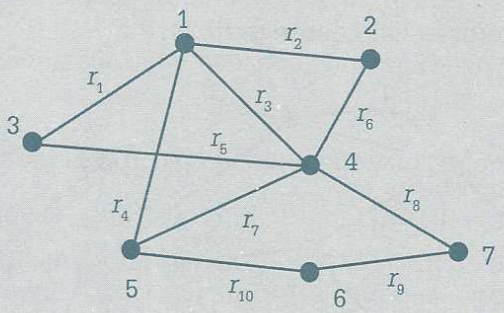
j) Obtener la matriz de adyacencia y la de incidencia.

k) ¿Cuál es la valencia de cada uno de los vértices?

l) Decir si los siguientes recorridos son caminos, camino simple, circuito, circuito simple:

- (1, 5, 8, 9, 7, 6, 1, 3).
- (5, 8, 4, 10, 10, 7, 6, 5).
- (1, 3, 3, 1).
- (4, 10, 9, 7, 6, 5, 1, 3, 8, 4).
- (2, 6, 5, 8, 9, 10).

7.2 Considérese el siguiente grafo G:



G

a) Es un grafo

- ¿Simple?
- ¿Conexo?

b) Obtener las matrices de adyacencia y de incidencia del grafo G.

c) ¿Tiene camino de Euler? Si es así, cuál es.

d) ¿Tiene circuito de Euler? Si es así, cuál es.

e) ¿Tiene circuito de Hamilton? Si es así, cuál es.

f) ¿Es plano? En caso de ser plano verificar que satisface la ecuación de Euler $A = L - V + 2$ y comprobar el resultado por medio de sus propiedades.

g) ¿Es un grafo K_n?

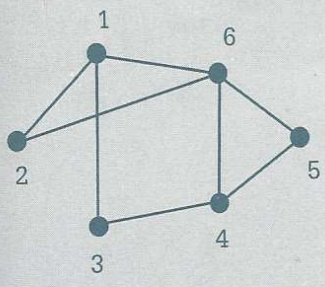
h) ¿Es un grafo $k_{n,m}$?

i) Obtener el complemento del grafo (G').

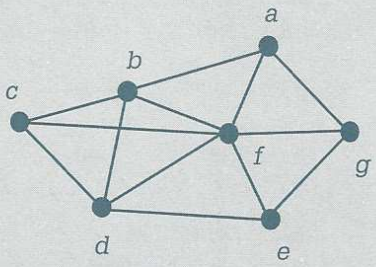
j) ¿Qué valencia debe tener cada uno de los vértices del grafo G para que se considere un grafo K_n, y cuál es el número total de aristas que debería tener?

7.3 En cada uno de los incisos obtener el complemento del grafo en caso de tratarse de un grafo simple, y en caso contrario explicar por qué no es grafo simple.

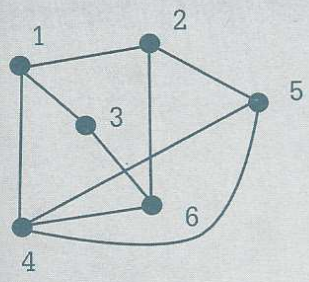
a)



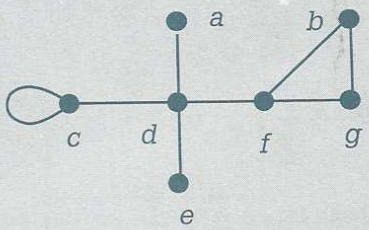
b)



c)

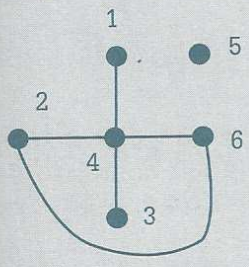


d)

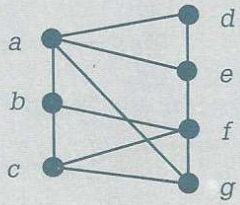


7.4 En cada uno de los siguientes incisos obtener el complemento del grafo en caso de ser un grafo simple, en caso contrario explicar por qué no es grafo simple.

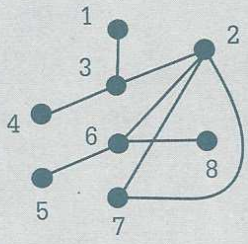
a)



b)



c)

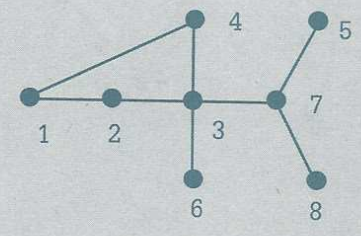


7.5 En relación con cada uno de los siguientes incisos determinar si el grafo es:

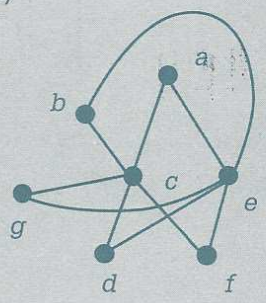
- Bipartido.
- Bipartido completo ($K_{n,m}$).
- Completo de n vértices (K_n).
- Ninguno de los anteriores.

En caso de ser bipartido o bipartido completo ¿cuáles son los elementos de los conjuntos? En caso de ser completo de n vértices ¿cuáles son los elementos del conjunto y cuál es el valor de n?

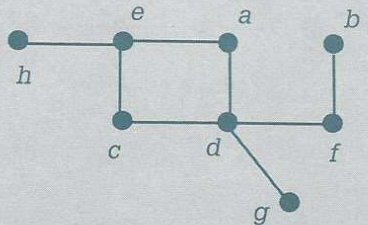
a)



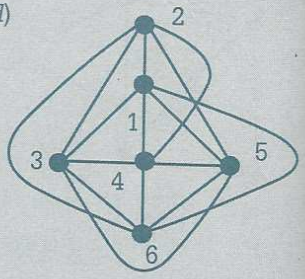
b)



c)



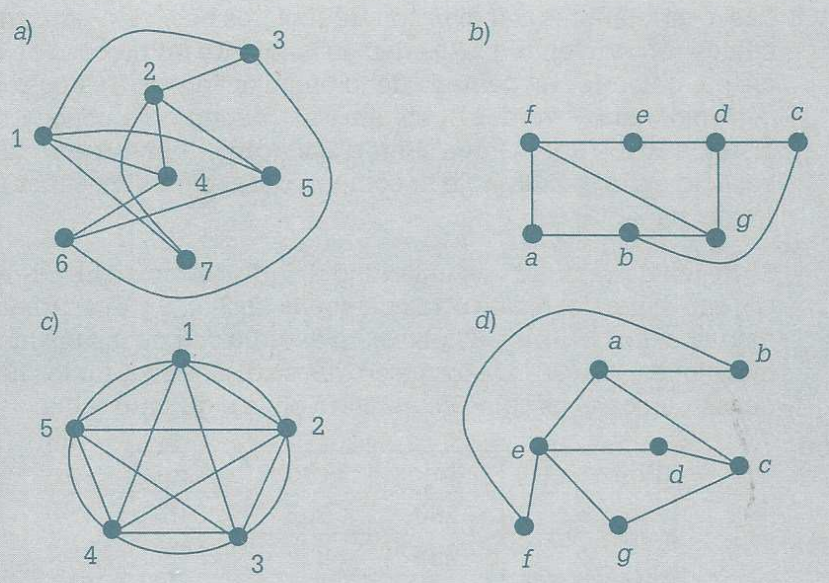
d)



7.6 Determinar si el grafo de cada uno de los siguientes incisos es:

- Bipartido.
- Bipartido completo ($K_{n,m}$).
- Completo de n vértices (K_n).
- Ninguno de los anteriores.

En caso de ser bipartido o bipartido completo ¿cuáles son los elementos de los conjuntos, y en caso de ser completo de n vértices cuáles son los elementos del conjunto y cuál es el valor de n?



7.7 Se desea enviar información a los usuarios de internet. Esto implica que si un usuario muestra interés en la información deportiva y cultural entonces sólo se le enviarán artículos de tipo cultural y deportivo, si muestra que sólo le interesa información científica entonces únicamente esa información se le enviará.

La información en internet está clasificada en artículos: culturales, deportivos y de carácter científico. Se realiza un registro mensual de las preferencias de los usuarios obteniéndose que en el mes anterior las preferencias fueron como se muestra en la siguiente tabla:

Usuario (U)	Culturales (C)	Deportivos (D)	Científicos (Ci)
1	10	2	0
2	2	14	1
3	0	15	2
4	3	9	7
5	8	1	1
6	1	12	4

a) Obtener los grafos de similaridad para un coeficiente de inferencia C = 6. ¿Qué interpretación se le puede dar a estos grafos?

b) ¿Cómo quedarían los grafos si C = 5? ¿Cuál es la interpretación de estos grafos?

7.8 Las computadoras trabajan con tres colores básicos a partir de los cuales construyen todos los demás mediante un proceso de mezcla por unidades de pantalla denominadas "pixels", estos colores son rojo, azul y verde. El sistema así definido se conoce como sistema RGB (Red, Green, Blue). Cada pixel tiene reservado un espacio en la memoria de la computadora, para almacenar la información.

Se desea determinar la similaridad del iris del ojo de una persona (1), con el iris del ojo de otras personas (2, 3, 4, 5 y 6) en una base de datos. Los bloques de pixels en estudio son de las mismas dimensiones, 30×30, y la forma en que se distribuye la información de las personas es como se muestra en la siguiente tabla:

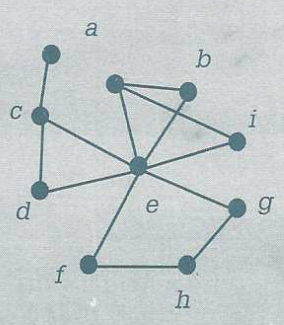
Persona (P)	Rojo	Verde	Azul
1	280	305	315
2	295	312	293
3	367	280	256
4	268	309	323
5	400	160	340

a) Obtener los grafos de similaridad para un coeficiente de inferencia C = 50. ¿Cuál es la interpretación de estos grafos?

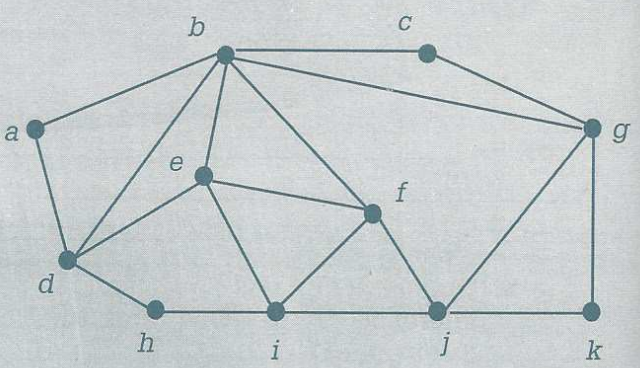
b) ¿Cómo quedarían los grafos si C = 30? Hacer una interpretación.

7.9 Establecer si cada uno de los siguientes grafos tiene camino de Euler y/o circuito de Euler.

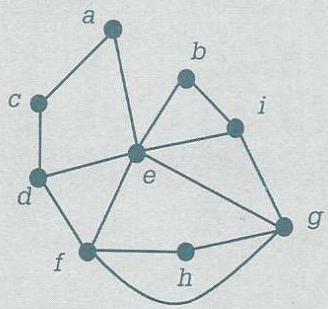
En caso afirmativo determinar dicho camino o circuito de Euler usando el algoritmo de Fleury. En caso negativo explicar el porqué.



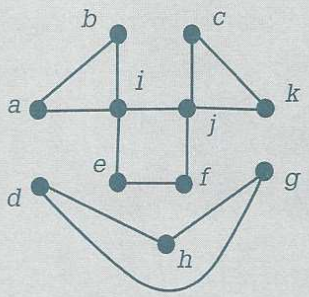
a)



b)



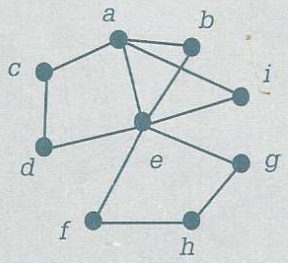
c)



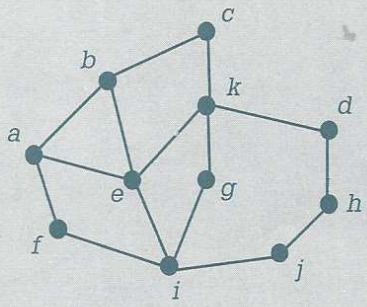
d)

7.10 Establecer si cada uno de los grafos siguientes tiene camino de Euler y/o circuito de Euler.

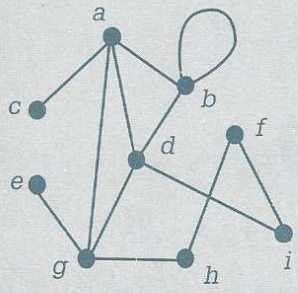
En caso afirmativo determinar dicho camino o circuito de Euler usando el algoritmo de Fleury. En caso negativo explicar el porqué.



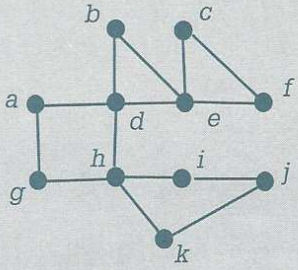
a)



b)

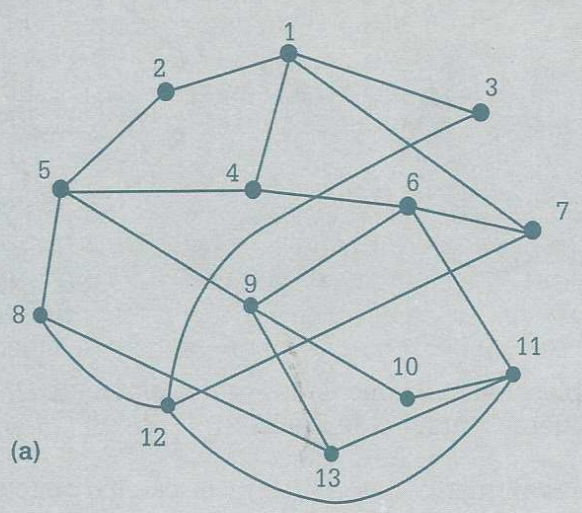


c)

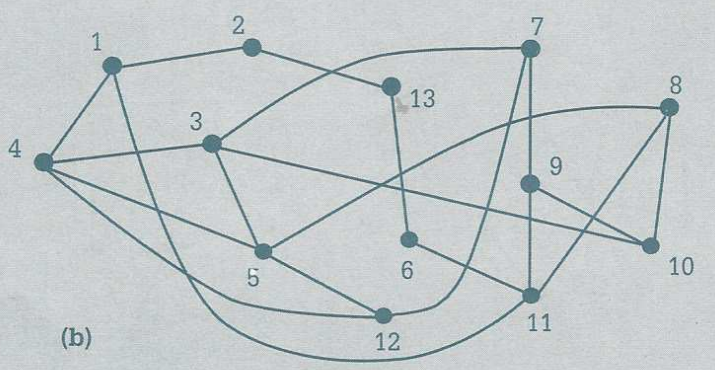


b)

7.11 Determinar si los siguientes grafos tienen circuito de Hamiltón:



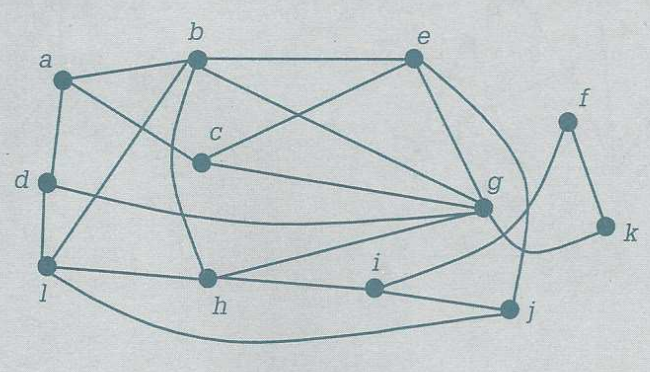
(a)



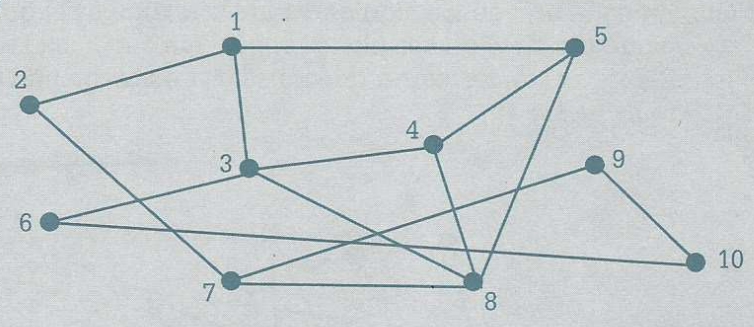
(b)

7.12 Determinar el circuito de Hamilton en cada uno de los siguientes grafos.

a)

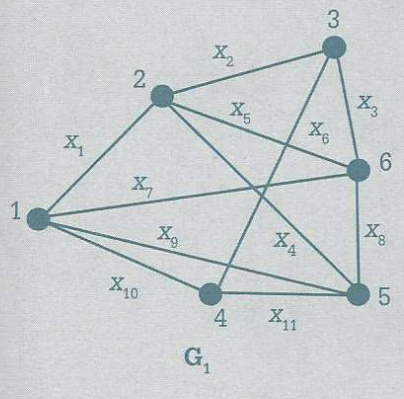


b)

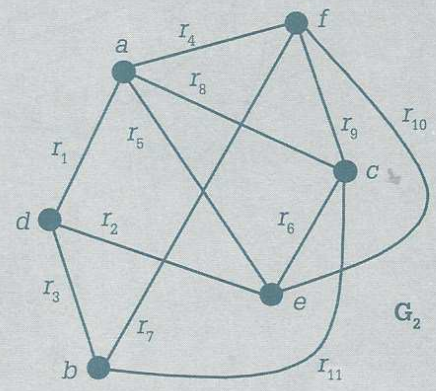


7.13 Determinar si los grafos siguientes G_1 y G_2 son isomorfos.

a) Por medio de las propiedades de G_1 y G_2.
b) Por medio de las matrices de incidencia.



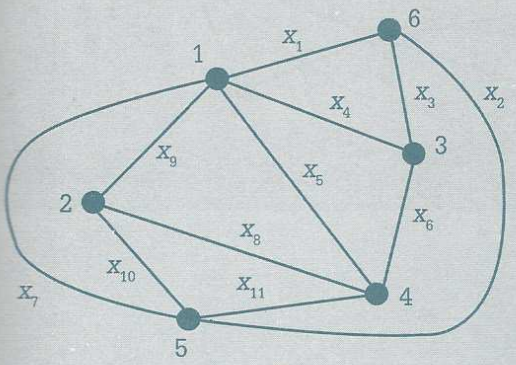
G_1



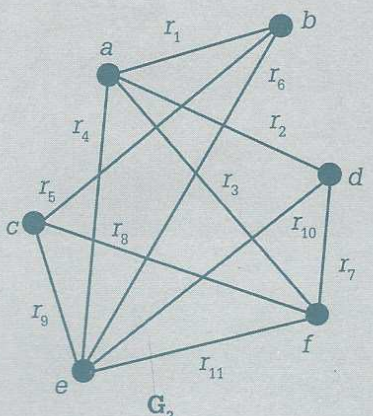
G_2

7.14 Determinar si los grafos siguientes G_1 y G_2 son isomorfos.

a) Por medio de las propiedades de G_1 y G_2.
b) Por medio de las matrices de incidencia.



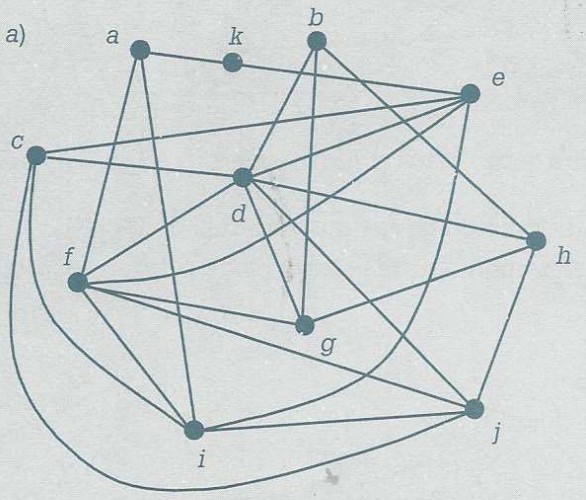
G_1



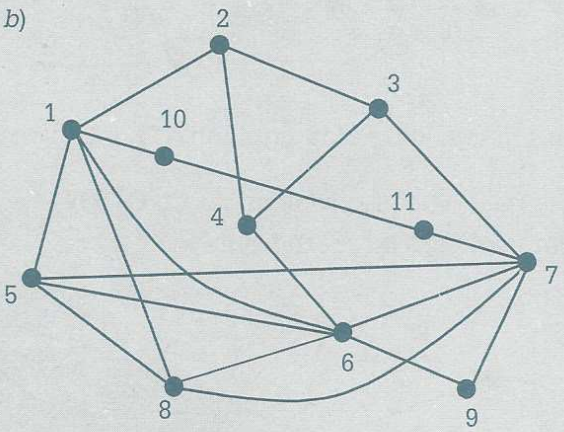
G_2

7.15 En cada uno de los siguientes incisos, dibujar el grafo en forma plana (en caso de ser plano probar que satisface la ecuación de Euler) o bien demostrar que no es plano obteniendo en él un grafo $K_{3,3}$ o K_5.

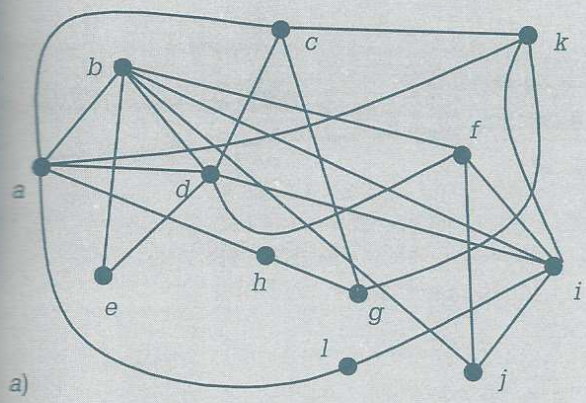
a)



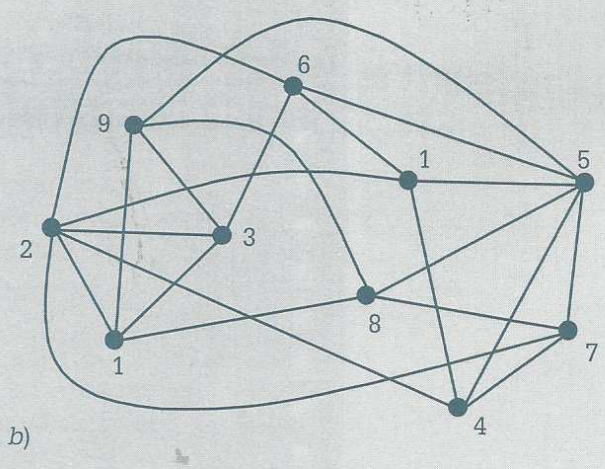
b)



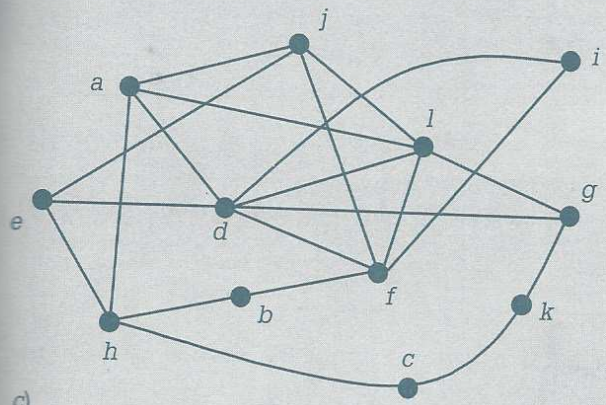
7.16 En cada uno de los siguientes incisos, dibujar el grafo en forma plana (en caso de ser plano mostrar que satisface la ecuación de Euler) o bien demostrar que no es plano obteniendo en él un grafo $K_{3,3}$ o K_5.



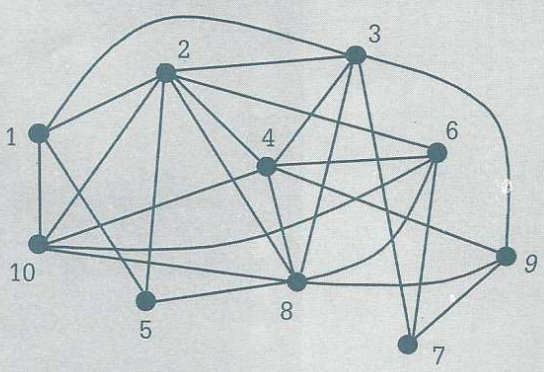
a)



b)



c)



d)

7.17 Dibujar un grafo plano conexo de 10 vértices y valencias 2, 2, 2, 3, 3, 4, 4, 4, 4, 6.

a) ¿Cuántos lados y cuántas caras tiene dicho grafo?

b) Probar que se cumple la ecuación de Euler.

7.18 Dibujar un grafo plano conexo de acuerdo a como se indica en cada inciso:

a) 9 vértices con valencias: 2, 3, 3, 4, 4, 4, 5, 5, 6.

b) 12 vértices con valencias: 2, 3, 3, 4, 4, 4, 4, 4, 5, 5, 6, 6.

Para cada uno de los incisos:

- Cuántos lados y cuántas caras tiene el grafo.
- Probar que se cumple la ecuación de Euler.

7.19 Colocar una E en la celda correspondiente de la siguiente tabla cuando el grafo $K_{n,m}$ tenga un circuito de Euler y colocar una H en la celda cuando el grafo $K_{n,m}$ tenga un circuito de Hamilton.

	1	2	3	4	5	6	...	m
1								
2								
3								
4								
5								
6								
.								
.								
n								

a) ¿Para qué valores de n y m un grafo $K_{n,m}$ tiene únicamente circuito de Euler?

b) ¿Para qué valores de n y m un grafo $K_{n,m}$ tiene únicamente un circuito de Hamilton?

c) ¿Para qué valores de n y m un grafo $K_{n,m}$ tiene un circuito de Euler y además un circuito de Hamilton?

7.20 Colocar en la siguiente tabla un asterisco en la fila E si el grafo K_n tiene un circuito de Euler, y colocar un asterisco en la fila H si el grafo K_n tiene un circuito de Hamilton.

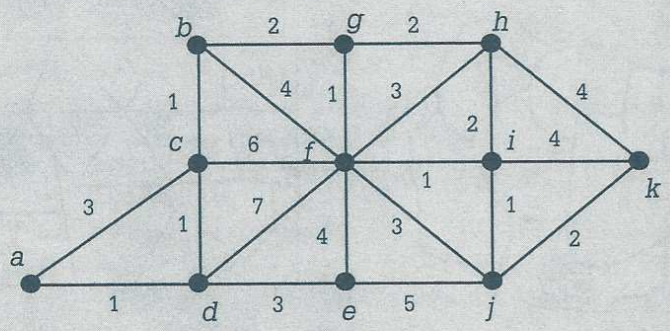
	1	2	3	4	5	6	7	...	n
E									
H									

a) ¿Para qué valores de n el grafo completo K_n no tiene circuito de Hamilton?

b) ¿Para qué valores de n el grafo K_n tiene circuito de Hamilton exclusivamente?

c) ¿Para qué valores de n el grafo K_n tiene un circuito de Euler y a la vez un circuito de Hamilton?

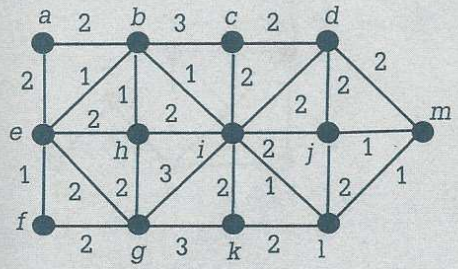
7.21 Determinar la ruta más corta en el siguiente grafo ponderado, para ir del nodo **a** hasta los nodos restantes del grafo, usando el algoritmo de Dijkstra.



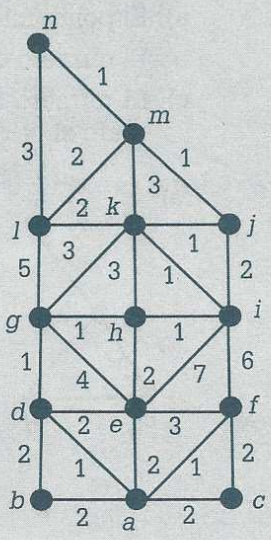
7.22 Determinar la ruta más corta en el siguiente grafo, usando el algoritmo de Dijkstra.

a) Para ir del nodo g a todos los demás en el inciso **a**.

b) Para ir del nodo c a todos los demás en el inciso **b**.

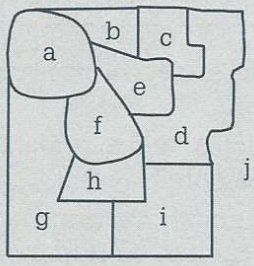


a)



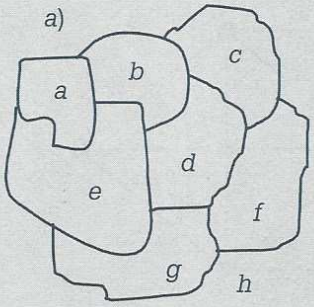
b)

7.23 Representar por medio de su grafo plano el siguiente mapa, colorear dicho grafo y obtener el número cromático X(G):

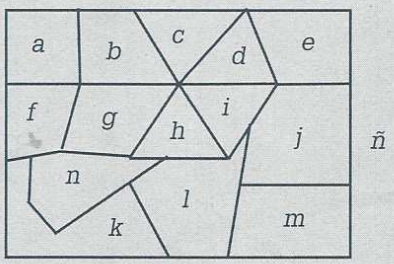


7.24 Representar por medio de grafos planos los mapas de cada uno de los incisos, colorear dichos grafos y obtener el número cromático X(G):

a)



b)



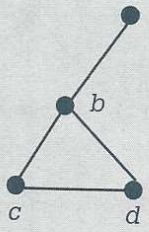
7.25 Para cada uno de los siguientes grafos obtener:

a) El polinomio cromático P(G,w).

b) El número cromático X(G).

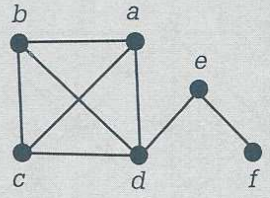
c) El número de maneras distintas que se puede coloreas cada grafo si w=6.

a)



G

b)



G

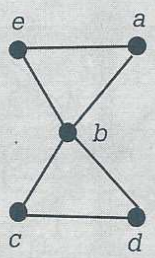
7.26 Para cada uno de los siguientes grafos obtener:

a) El polinomio cromático P(G,w).

b) El número cromático X(G).

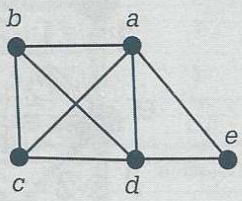
c) El número de maneras distintas en que se puede colorear cada grafo si w=6.

a)



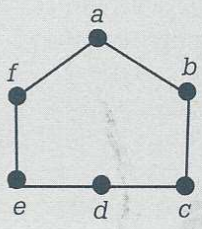
G

b)



G

c)

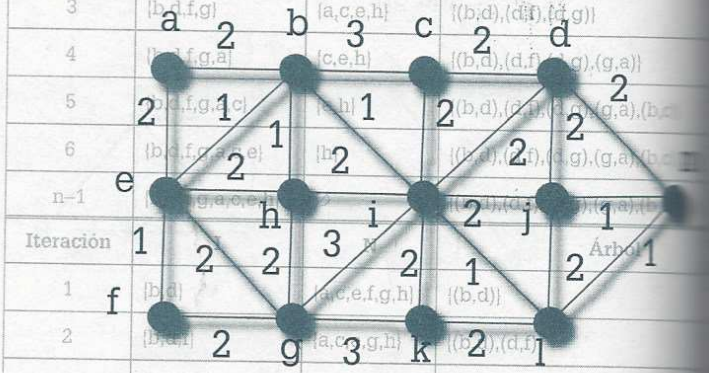


G

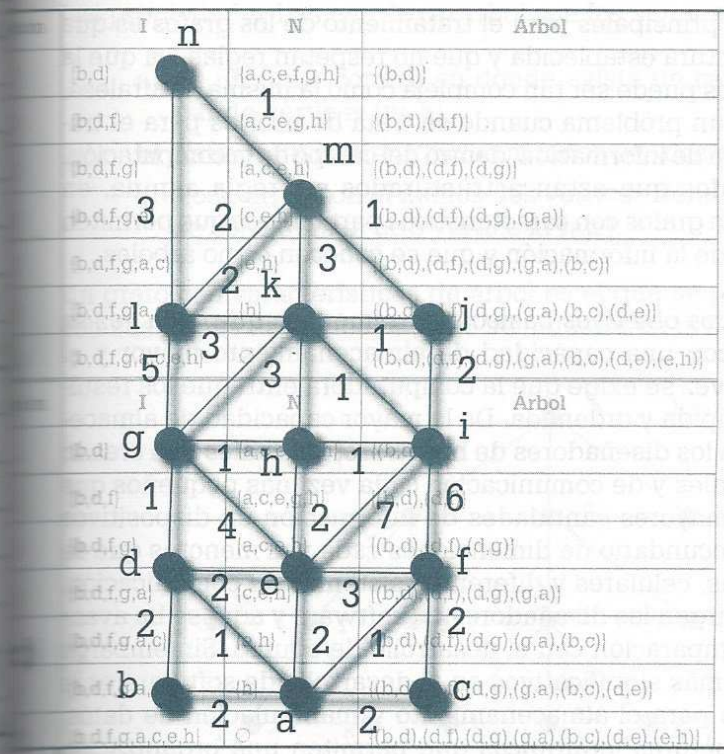
CAPÍTULO VIII

Árboles

Iteración	I	N	Árbol
1	{b,d}	{a,c,e,f,g,h}	{(b,d)}
2	{b,d,f}	{a,c,e,g,h}	{(b,d),(d,f)}
3	{b,d,f,g}	{a,c,e,h}	{(b,d),(d,f),(d,g)}
4	{b,d,f,g,a}	{c,e,h}	{(b,d),(d,f),(d,g),(g,a)}
5	{b,d,f,g,a,c}	{e,h}	{(b,d),(d,f),(d,g),(g,a),(b,c)}
6	{b,d,f,g,a,c,e}	{h}	{(b,d),(d,f),(d,g),(g,a),(b,c),(c,e)}
n–1	{b,d,f,g,a,c,e,h}	∅	{(b,d),(d,f),(d,g),(g,a),(b,c),(c,e),(e,h)}



b)



Con los árboles se representa la dependencia lógica entre la deci los atributos

Nicolas Bourbaki

Objetivos

- Establecer los principios fundamentales para abordar el tema de árboles.
- Reconocer los diferentes tipos de árboles, sus propiedades y aplicaciones.
- Conocer las características que debe tener un grafo para ser considerado como árbol y obtener un árbol a partir de un grafo.
- Aprovechar las propiedades de los árboles para modelar y resolver problemas específicos.
- Aplicar la estructura de árbol en la organización y procesamiento de información.

8.1 Introducción

Uno de los problemas principales para el tratamiento de los grafos es que no guardan una estructura establecida y que no respetan reglas, ya que la relación entre los nodos puede ser tan compleja como la misma naturaleza. Sin embargo, este es un problema cuando se trata de usarlos para el tratamiento y organización de información, dentro del campo de la computación. En lugar de usar grafos que están estructurados sin regla alguna, en computación se utilizan grafos con características particulares que permiten un mejor tratamiento de la información y que se conocen como árboles.

En computación hay dos objetivos básicos: el primero es que cada vez se desarrollen equipos con una capacidad de almacenamiento mayor y el segundo es que cada vez se exige que la computadora entregue los resultados en forma más rápida y ordenada. De la mayor capacidad de almacenamiento se encargan los diseñadores de hardware, los cuales han creado equipos computacionales y de comunicación cada vez más pequeños que permiten almacenar mayores cantidades de información en dispositivos de almacenamiento secundario de dimensiones cada vez menores que se usan en computadoras, celulares y diferentes sistemas de comunicación. De lo segundo se encargan los diseñadores de software y aquí se ha avanzado muy poco en comparación con el desarrollo de equipo. Sin embargo uno de los progresos más significativos en el desarrollo de software es la utilización de árboles para el almacenamiento y manipulación de datos. Los árboles son estructuras jerárquicas que permiten una organización ordenada de la información, de forma que cuando se requiera se pueda encontrar en forma rápida y precisa.

Una de las primeras aplicaciones de los árboles se presentó en 1847 cuando Gustav Kirchhoff los utilizó en la manipulación de redes eléctricas; otra aplicación importante la llevó a cabo Grace Harper en 1951 al utilizarlos en el manejo de expresiones matemáticas. En la actualidad los árboles se usan en la computación en los procesos de clasificación de información, bases de datos, codificación de información, estructuras de datos y reconocimiento de patrones.

Tomando en cuenta la teoría de grafos expuesta en el capítulo anterior, el concepto de árbol se puede definir en los siguientes términos:

Definición. Un árbol es un grafo conexo que no tiene ciclos, lazos ni lados paralelos.

8.2 Propiedades de los árboles

Las propiedades básicas de un árbol son las siguientes:

a) Es un grafo conexo en donde existe un camino entre cualquier par de vértices (w, x).
b) Este grafo no tiene ciclos ni lados paralelos.
c) Todo árbol con al menos dos vértices tiene al menos una hoja (si se considera al otro vértice la raíz).

Un grafo con características de árbol es el que se parece a un árbol real con sus ramas hacia abajo, como se muestra en la figura 8.1.

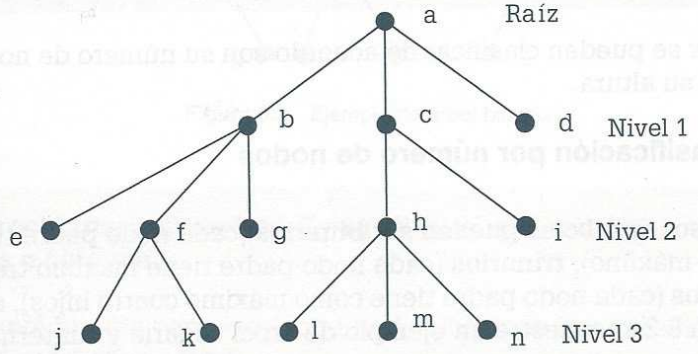


Figura 8.1 Ejemplo de árbol.

Los vértices de un árbol reciben el nombre de **nodos** y los lados de **ramas**. Un grafo está compuesto por niveles y el más alto de la jerarquía se llama **raíz**. La raíz tiene un nivel 0, los vértices inmediatamente debajo de la raíz tienen un nivel 1 y así sucesivamente. La altura o peso de un árbol es el valor de su nivel más bajo, por ejemplo en el árbol de la figura 8.1 la altura es 3.

Con excepción de la raíz, todo nodo está vinculado a otro de mayor nivel que recibe el nombre de **padre**, también cualquier nodo puede tener uno o más elementos relacionados en un nivel más bajo y a éstos se les llama **hijos**. En el árbol de la figura 8.1 la raíz es a y los hijos de la raíz son {b, c, d}. Como se ve, ningún hijo puede tener más de un padre.

A los elementos que están en las puntas de las ramas (es decir, que no tienen hijos) se les llama **hojas**. En el grafo de la figura 8.1 las hojas son {d, e, g, i, j, k, l, m, n}.

A todos los elementos colocados debajo de un nodo, independientemente de su nivel, se les llama **descendientes**. En el grafo de la figura 8.1 el nodo c tiene como descendientes a los nodos {h, i, l, m, n} y entre ellos se puede formar el subárbol {c, h, i, l, m, n}. En casos como éste se dice que un árbol puede estar integrado por varios subárboles.

Los elementos colocados en una misma línea de descendencia, antes de un nodo, se llaman **antecesores**. En la figura 8.1 los antecesores de k son {f, b, a}.

Por otro lado, se llaman **vértices internos** a todos aquellos que no son hojas. En la figura 8.1 {a, b, c, f, h} son vértices internos.

8.3 Tipos de árboles

Los árboles se pueden clasificar de acuerdo con su número de nodos y en función de su altura.

8.3.1 Clasificación por número de nodos

En este caso los árboles pueden ser **binarios** (cada nodo padre tiene uno o dos hijos máximo), **trinarios** (cada nodo padre tiene máximo tres hijos), **cuaternarios** (cada nodo padre tiene como máximo cuatro hijos), etcétera. En la figura 8.2 se muestra un ejemplo de árbol trinario y cuaternario.

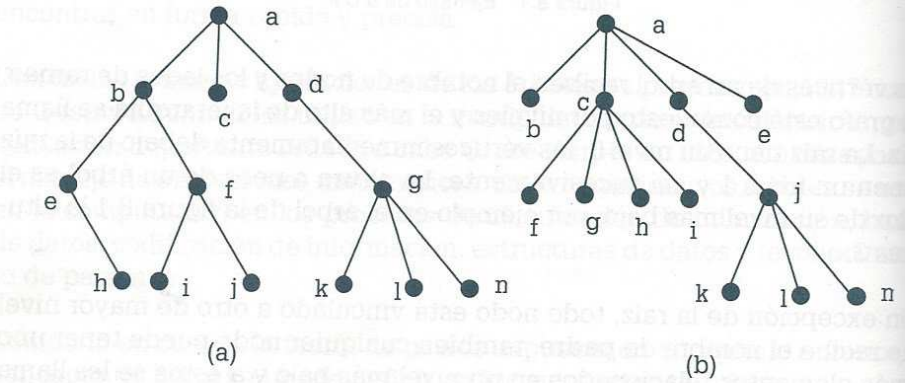


Figura 8.2 Dos tipos de árbol: (**a**) trinario; (**b**) cuaternario.

- **Árbol binario**. En este tipo de árbol cada nodo tiene como máximo dos hijos, esto es, el nodo puede tener dos ramas, una o ninguna, pero nunca puede tener más de dos. En la figura 8.3 se muestra un ejemplo de árbol binario.

Los árboles binarios son especialmente importantes en el área de la computación ya que por su naturaleza de tener solamente dos valores (0, 1), o bien falso o verdadero, son muy útiles en aplicaciones de sistemas digitales.

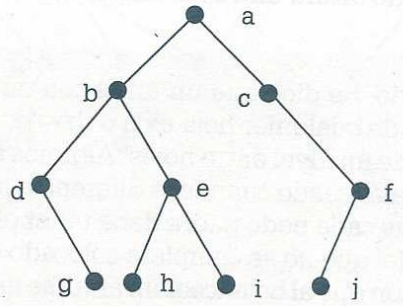


Figura 8.3 Ejemplo de árbol binario.

- **Árbol binario completo**. Es aquél en el que cada nodo tiene dos ramas o ninguna.

Un árbol binario completo con i nodos internos tiene (i + 1) hojas y (2i + 1) vértices en total.

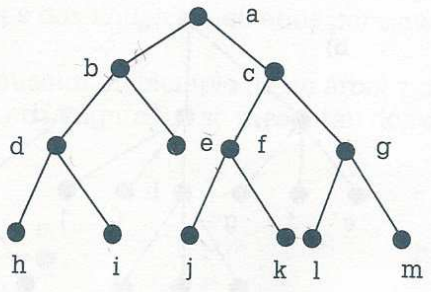


Figura 8.4 Ejemplo de árbol binario completo.

En el caso del árbol de la figura 8.4 los nodos internos son i = 6, por lo tanto tiene:

Hojas = i + 1 = 6 + 1 = 7

Total de vértices = 2i + 1 = 2 (6) + 1 = 13

Finalmente hay que destacar que los árboles completos trinarios, cuaternarios o con más hijos se usan para organizar información voluminosa.

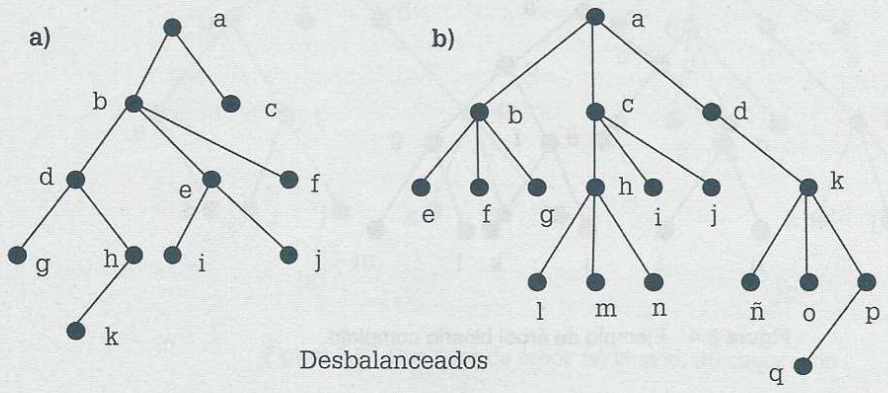
8.3.2 Clasificación por altura

De acuerdo con este criterio los árboles pueden ser **balanceados** (cuando la diferencia de altura entre sus ramas es máximo 1) y **desbalanceados** (cuando la diferencia de altura entre las ramas es mayor de 1).

- **Árbol balanceado**. Se dice que un árbol con una altura h está balanceado si el nivel de cualquier hoja es h o (h – 1), esto es, si hay una diferencia máxima de un nivel entre hojas. Algunos autores consideran que un árbol está balanceado cuando la diferencia máxima entre hojas es de 1, pero además cada nodo padre debe tener el mismo número de hijos, a excepción del que no se complete colocado en la parte baja del árbol. Es por esa razón que al balancear un árbol se debe indicar también el número de hijos que tendrá cada uno de los nodos.

  Para balancear un árbol con una cantidad constante de hijos de los nodos padres, se llenan empezando por la raíz y descendiendo con un avance de izquierda a derecha.

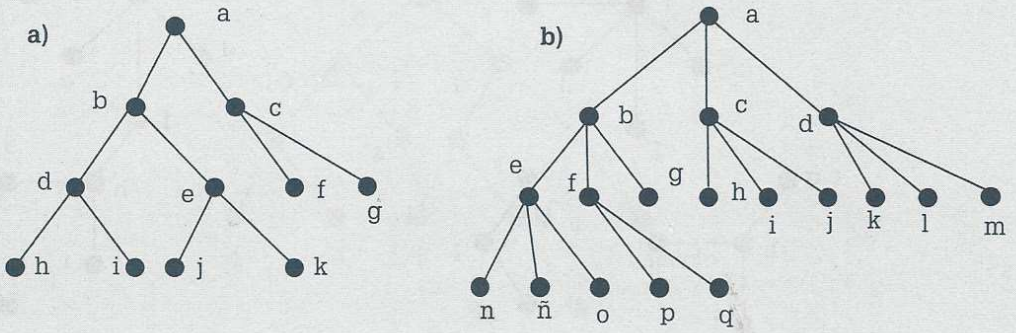
Ejemplo 8.1. Balancear como binario el árbol del inciso (a) y como trinario el del inciso (b).



Desbalanceados

Solución. Estos árboles son desbalanceados porque la diferencia de alturas entre hojas sobrepasa (h – 1). Por ejemplo, en el árbol del inciso (a) el nivel del vértice (c) es 1 mientras que la altura del árbol es h = 4 y ob-

viamente $1 \neq h - 1$. En la siguiente figura se tienen los mismos árboles balanceados por lo que la diferencia máxima entre hojas es de 1.



Balanceados

8.4 Bosques

Un **bosque** es un conjunto de árboles, en otras palabras un árbol es un bosque conectado.

De un árbol se pueden obtener varios subárboles, mismos que conforman un bosque. A su vez un árbol puede considerarse como un bosque conectado, sólo se debe tener en cuenta que el árbol más pequeño está integrado por cuando menos dos nodos conectados por una arista.

En la figura 8.5 se muestra un ejemplo de un árbol y de un grafo que no es árbol, mientras que en la figura 8.6 se presentan dos ejemplos de bosque.

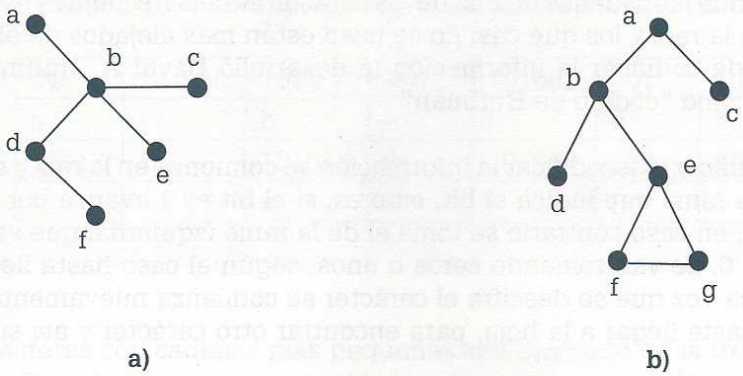


Figura 8.5 El grafo de (**a**) es un árbol ya que es conexo y no tiene ciclos, el de(**b**) no es árbol porque tiene ciclos.

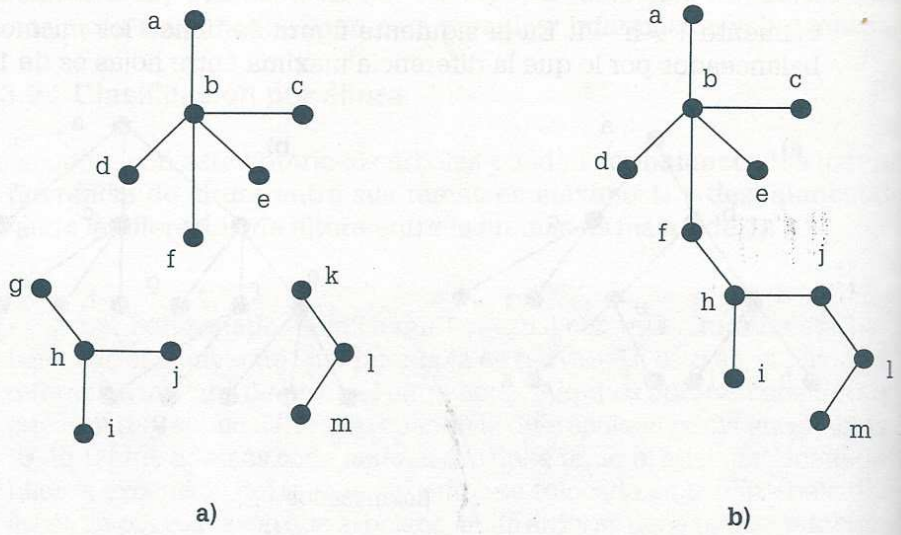


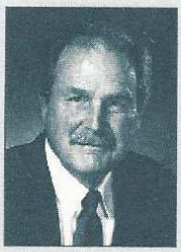
Figura 8.6 Considerando las tres partes que integran el grafo de (**a**) se tiene que éste no es árbol porque no es conexo, pero es bosque porque se compone de varios árboles. El grafo de (**b**) es árbol porque es conexo y no tiene ciclos, además de que es bosque porque es un árbol conectado.

8.5 Árboles con pesos

David Albert Huffman
(1925-1999)

Fue un ingeniero eléctrico estadounidense que hizo importantes contribuciones en el estudio de aparatos finitos, circuitos aleatorios, síntesis de procedimientos y diseño de señales, teoría de la información y codificación, diseños de señal para aplicaciones de radar y comunicaciones así como procedimientos de diseño para circuitos lógicos asíncronos; sin embargo es más conocido por su "código de Huffman", un sistema de compresión y codificación de longitud variable.

El código de Huffman puede ser usado en casi cualquier aplicación ya que es un sistema válido para la compresión y posterior transmisión de cualquier dato en formato digital, pudiendo aplicarse a faxes, módems, redes de computadoras y televisión.

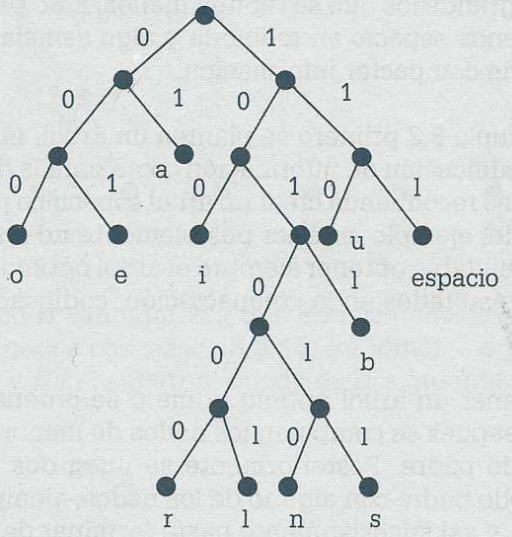


Para representar caracteres en el código ASCII se usan cadenas de 8 bits, sin embargo se puede aumentar la velocidad de procesamiento o bien aprovechar mejor la memoria de la computadora, mediante una compactación de la información, usando cadenas de diferente longitud. Las cadenas más pequeñas pueden representar a los caracteres que se presentan con más frecuencia, como son las vocales y las consonantes {b, c, d, f, m, n, p, r, s}.

Para codificar la información los bits se colocan en un árbol binario completo donde las cadenas de bits de los caracteres más frecuentes están más cerca de la raíz y los que casi no se usan están más alejados de ella. Esta técnica de codificar la información la desarrolló David A. Huffman y se conoce como "código de Huffman".

Para codificar o decodificar la información se comienza en la raíz y se avanza por la rama que indica el bit, esto es, si el bit es 1 avanza por la rama derecha, en caso contrario se toma el de la rama izquierda, que está marcado un 0. Se van tomando ceros o unos, según el caso hasta llegar a la hoja. Una vez que se descifra el carácter se comienza nuevamente desde la raíz hasta llegar a la hoja, para encontrar otro carácter y así sucesivamente.

Ejemplo 8.2. Usando el siguiente árbol



¿cuál es el significado de la cadena de caracteres?

01101000101100010100100110101111110111001010100110100010000010
1011

Solución. Comenzando por la raíz y tomando bits hasta llegar a la hoja, se tienen los códigos de los siguientes caracteres

000 = o	100 = i	101010 = n	111 = espacio
01 = a	101000 = r	101011 = s	1011 = b
001 = e	101001 = l	110 = u	

de forma que la decodificación del mensaje anterior es

01	101000	1011	000	101001	001	101011	111
a	r	b	o	l	e	s	espacio
1011	100	101010	01	101000	100	000	101011
b	i	n	a	r	i	o	s

Los caracteres con cadenas más pequeñas son producto de la frecuencia del uso. Cuando se desea compactar la información de un documento lo que se recomienda primero es darle una pasada para determinar el núme-

ro de veces que se presenta cada uno de los caracteres que integran dicho documento, posteriormente se elabora el árbol óptimo que permita codificar con menos bits los caracteres que se repiten más y con un número de bits más grande los que se repiten menos. Esto permite que la información ocupe menos espacio en memoria y algo semejante hace el software encargado de compactar información.

En el ejemplo 8.2 primero se plantea un árbol, mismo que se utiliza tanto para la codificación de información como para la decodificación de ésta, no obstante se recomienda crear un árbol específico para cada uno de los usos. El árbol del ejemplo anterior posiblemente no fue el óptimo, sin embargo es recomendable obtener siempre el árbol óptimo con la finalidad de lograr mejores resultados en la compactación, codificación, decodificación y velocidad.

Para obtener un árbol óptimo primero se ordenan los pesos de menor a mayor, después se combinan los nodos de menor peso y se coloca la suma en el nodo padre. Posteriormente se unen dos nuevos nodos, o bien el nuevo nodo padre con alguno de los nodos, siempre y cuando la suma sea la menor, y así sucesivamente hasta terminar de integrar todos los nodos al árbol óptimo.

El árbol binario óptimo se llama así porque su altura es mínima y los pesos o frecuencias están distribuidos de manera que los más pesados están más cercanos a la raíz y los menos pesados se encuentran más alejados de ella, además de que se trata de un árbol binario completo.

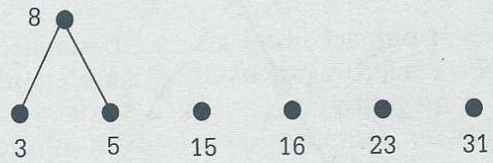
Ejemplo 8.3. Se tienen los caracteres y frecuencia de uso de cada carácter en la siguiente tabla. ¿Cuál es el árbol binario óptimo para el código de Huffman?

Carácter	Peso o frecuencia
d	15
e	23
m	5
a	31
t	3
s	16

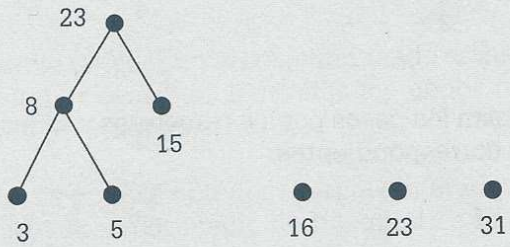
Descifrar con dicho árbol el siguiente mensaje:

00010100111010110000

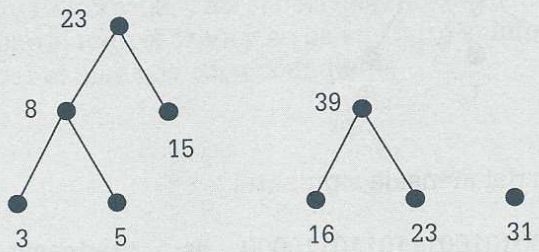
Solución. Para obtener el árbol óptimo, primero se ordenan los pesos de menor a mayor. Después se combinan los nodos de menor peso y se coloca la suma en el nodo padre:



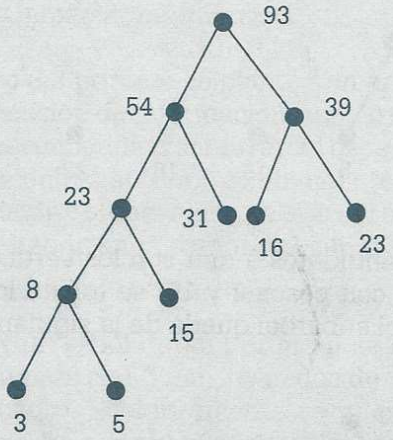
Los pares de nodos candidatos a unir son los vértices con pesos 15 y 16 además de los nodos con pesos 8 y 15; se toman los últimos porque su suma es menor y así el subárbol queda de la siguiente manera:



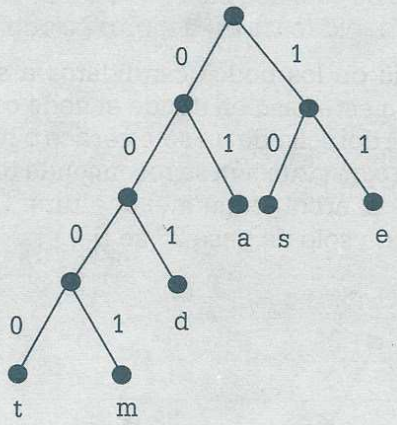
Algunas veces la suma de los nodos candidatos a ser combinados son iguales, como ocurre en este caso en donde el nodo con peso 16 se puede unir con el que está a la derecha de un solo vértice o bien al subárbol cuyo peso también es 23; en esta situación se recomienda unir los nodos nuevos con la finalidad de que el árbol tenga menor altura. Combinando el nodo con peso 16 y el que está solo de peso 23 se tiene:



Sumando ahora los nodos con pesos 23 y 31, finalmente se combinan los nodos con pesos 54 y 39 para obtener el árbol óptimo siguiente:



Por último se cambian los pesos por los caracteres y se les asignan los bits 0 y 1 en las ramas correspondientes:



La decodificación del mensaje siguiente:

00010100111010110110000 es **madeaset**

Es obvio que en la práctica los árboles son más grandes ya que seguramente se tienen todas las letras del alfabeto además de números y símbolos raros, lo que además hace que el número de bits de los caracteres menos frecuentes se incremente de manera considerable, circunstancia que es compensada cuando se tienen cadenas muy cortas de caracteres frecuentes como las vocales.

Por otro lado, no necesariamente deben ser letras las que se codifiquen, también pueden ser palabras de un lenguaje como C, Java o Pascal, en donde sólo se utilizan unas cuantas palabras para editar un programa y en donde palabras como printf() o write() son frecuentes, de forma que si se representan con cadenas de caracteres pequeñas los programas podrían ser compactados para ocupar menos memoria.

8.6 Árboles generadores

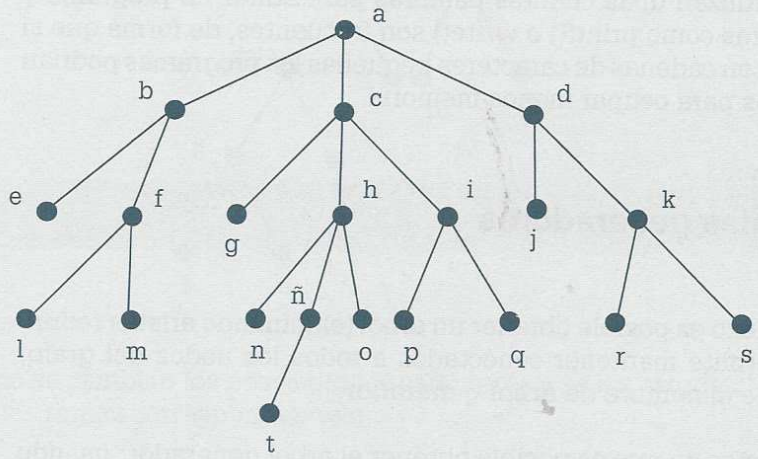
De un grafo conexo es posible obtener un árbol (eliminando aristas redundantes) que permite mantener conectados a todos los nodos del grafo; este árbol recibe el nombre de **árbol generador**.

Existen dos formas en que es posible obtener el árbol generador: usando **búsqueda en profundidad** o bien por medio de **búsqueda a lo ancho**. A continuación se exponen ambos procedimientos para luego aplicarlos en la obtención de un árbol generador en grafos.

8.6.1 Búsqueda a lo ancho

En este procedimiento se comienza en la raíz y después se examinan todos los hijos de la misma de izquierda a derecha. Si la información que se busca no se encuentra en ese nivel, se procede a buscar en el siguiente nivel también de izquierda a derecha, y así sucesivamente hasta encontrar la información. Si se recorrió todo el árbol y no se encontró la información buscada, se debe mandar el mensaje correspondiente.

Ejemplo 8.4. Considérese que en el siguiente árbol se está buscando la letra "n". Primero se busca en la raíz (a), después en todos los nodos dependientes de la raíz de izquierda a derecha (en este caso b, c, d) y como aún no se encuentra la información se procede a buscar en el siguiente nivel (e, f, g, h, i, j, k) y así sucesivamente de manera que el recorrido que se debe realizar para encontrar el nodo buscado es (a, b, c, d, e, f, g, h, i, j, k, l, m, n).



Si se recorre todo un árbol y no se encuentra la información es conveniente mandar el mensaje "información inexistente". La búsqueda a lo ancho es útil cuando los árboles están balanceados o tienen pocos niveles en relación a la información que contienen.

8.6.2 Búsqueda en profundidad

En este caso se comienza en el nodo raíz, después se busca en el hijo de la izquierda y si este nodo tiene hijos se continúa con el de la izquierda y así sucesivamente hasta llegar a la parte más baja del árbol. Si este nodo ya no tiene hijo izquierdo, se continúa con el hijo de la derecha (o el que se encuentra más a la izquierda en caso de no ser un árbol binario) hasta llegar a la hoja. Si no se ha encontrado la información, se recorre el camino andado hasta el nodo inmediato anterior que tenga hijos y cuyos hijos no hayan sido inspeccionados, dando preferencia al de más a la izquierda. Cuando se llega nuevamente a la hoja, se regresa hasta el nodo inmediato anterior que tenga hijos sin inspeccionar y así sucesivamente.

En la búsqueda en profundidad posiblemente se regrese hasta el nodo raíz (lo cual significa que se revisó toda la rama dependiente del hijo izquierdo

de la raíz). Si la raíz tiene hijos que aún no han sido inspeccionados se selecciona el nodo más cercano al hijo izquierdo y todos sus descendientes, dando preferencia a los nodos de la izquierda. Este procedimiento se lleva a cabo hasta encontrar la información o bien recorrer todo el árbol.

En el árbol del ejemplo 8.4 el recorrido para encontrar la letra "n" es:

- Comenzar en la raíz (a).
- Como tiene hijos, continuar con el de la izquierda (a, b).
- Como tiene hijos se sigue con el de la izquierda (a, b, e).

Como el nodo "e" ya es una hoja y no se ha encontrado la información buscada, se regresa al nodo antecesor "b" y se busca en el nodo inspeccionado "f" de tal manera que el recorrido es (a, b, e, f); continuando con el de la izquierda se llega al nodo "l" para tener el recorrido (a, b, e, f, l) y así se continúa hasta llegar a la información buscada. De esta forma el recorrido en profundidad para buscar información en el árbol hasta encontrar la letra "n" es: (a, b, e, f, l, m, c, g, h, n). El recorrido del árbol completo por medio de la búsqueda en profundidad es: (a, b, e, f, l, m, c, g, h, n, ñ, t, o, i, p, q, d, j, k, r, s).

Es importante mencionar que en este caso se está dando preferencia para buscar la información al nodo de la izquierda, pero podría ser al de la derecha; siempre y cuando se respete la forma de búsqueda en todos los casos, se estará hablando de una búsqueda en profundidad.

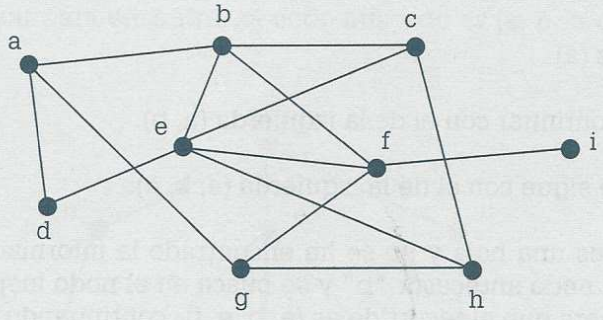
8.6.3 Obtención de árboles generadores

Un árbol es un grafo muy importante ya que permite la estructuración y tratamiento de la información de manera más sencilla. De todo grafo conexo se puede obtener un árbol que recibe el nombre de **árbol generador** y cuyas características son:

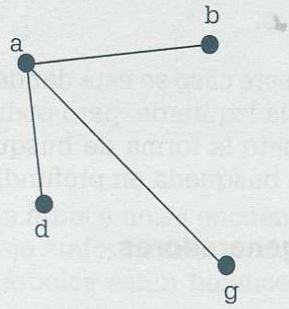
a) Es un grafo conexo.
b) Es un grafo que no tiene ciclos.

Es posible obtener un árbol generador por medio de búsqueda a lo ancho o bien por búsqueda en profundidad, sin embargo en un grafo cualquiera no se puede establecer niveles, ni tampoco se puede identificar el nodo de la izquierda como se realizó en el caso de árboles. Por lo tanto se recomienda establecer un orden de preferencia de búsqueda, mismo que puede ser en forma ascendente en caso de que los nodos sean números o bien alfabéticamente si los vértices se indican con letras.

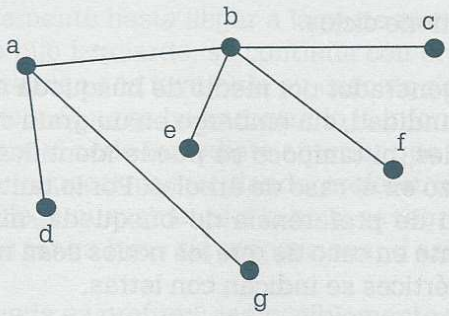
Ejemplo 8.5. Obtener un árbol generador a partir del siguiente grafo utilizando una búsqueda a lo ancho y una búsqueda en profundidad, partiendo del vértice (a) y considerando prioridad en la elección el orden alfabético.



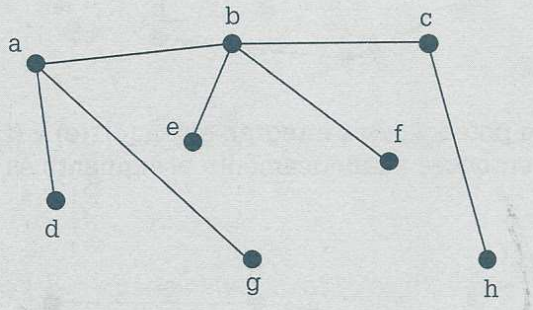
Por búsqueda a lo ancho. El árbol generador por medio de una búsqueda a lo ancho comienza en el vértice origen (a), después se integran los nodos adyacentes a él para obtener el siguiente grafo:



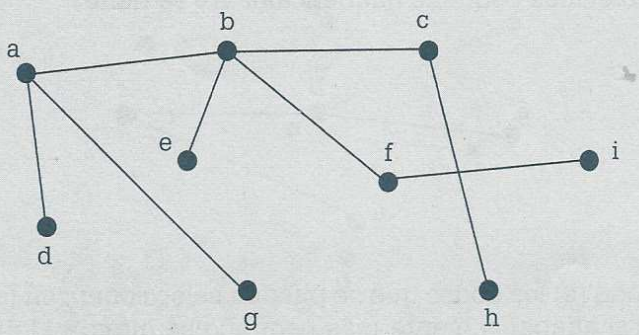
El nodo (a) ya no tiene nodos adyacentes a él que no hayan sido integrados y alfabéticamente el nodo (b) es el siguiente a inspeccionar, de forma que se anexan los nodos adyacentes a éste cuidando que no se formen ciclos con los nodos ya existentes y lo que se obtiene es la siguiente figura:



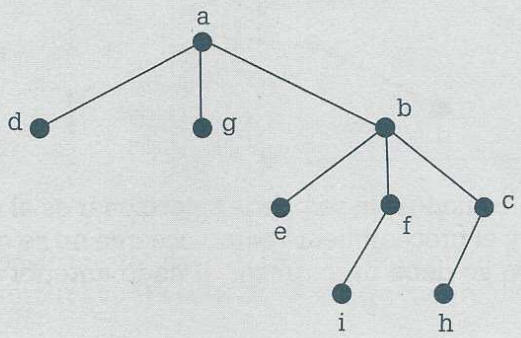
Los nodos (a) y (b) del grafo anterior ya no tienen nodos adyacentes sin inspeccionar, y de los nodos restantes el más pequeño alfabéticamente es el (c). Por lo tanto se integra al árbol generador el nodo (h):



Finalmente el árbol generador queda de la siguiente forma:

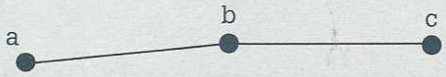


Se puede observar que se trata de un árbol porque es conexo y no tiene ciclos. Es un árbol "libre" porque no tiene raíz ya que cualquiera de sus nodos puede hacer las veces de raíz. Si se considera que la raíz es el vértice inicial (a) el árbol generador es el siguiente:

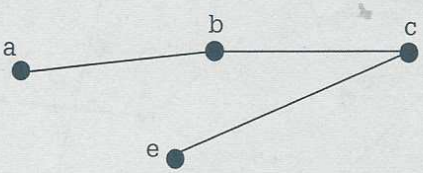


Por búsqueda en profundidad. Considerando el inicio del árbol generador en el nodo (a) y de los vértices adyacentes (b), (d) y (g), se selecciona el vértice (b) por ser el menor alfabéticamente y se integra al árbol generador como se muestra en la siguiente figura:

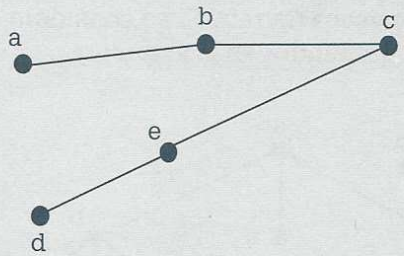
Ahora los nodos potenciales a integrarse son (c), (e) y (f), y como el nodo actual es el (b) entonces alfabéticamente el siguiente es (c), por lo que se tiene:



A partir del nodo (c), los nodos potenciales a integrar son (e) y (h) por lo que tomando el más pequeño alfabéticamente se tiene:

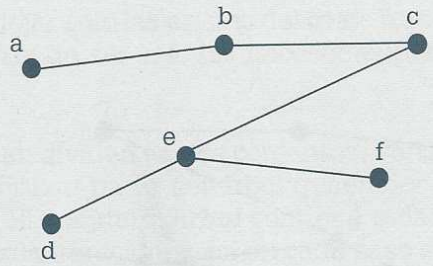


Luego del nodo (e) los nodos que se pueden seleccionar son (d), (f) y (h), y como (d) es el alfabéticamente más pequeño se integra al árbol generador:

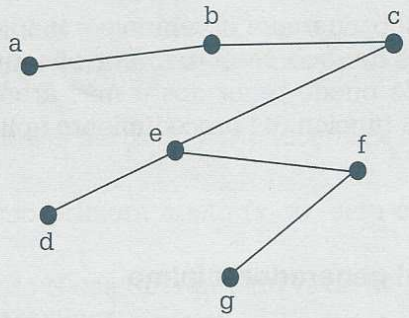


Del nodo (d) el único nodo que se puede seleccionar es el (a) pero con ello se forma un ciclo y el procedimiento establece que no se deben de formar ciclos, por lo tanto se debe de regresar al nodo anterior para ver si aún

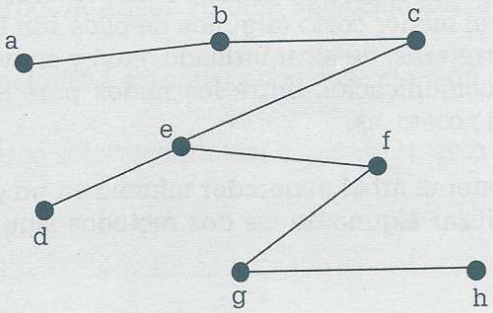
existen nodos que no han sido integrados. De esta forma se integra al árbol generador el nodo (f):



A partir del nodo (f) se puede seleccionar (g) o (i), y por la prioridad alfabética se selecciona (g):

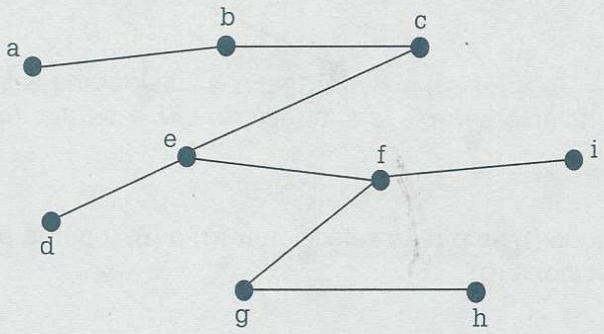


Desde el nodo (g) sólo se puede seleccionar (h) por lo que se tiene:



A partir de la posición del nodo (h) ya no es posible seleccionar ningún nodo, ya que tanto (c) y (e) están en el árbol y volverlos a seleccionar im-

plicaría la formación de ciclos. Por esta razón se regresa primero al nodo (g) y después al (f) de donde se puede seleccionar el nodo (i) que permite tener el siguiente árbol generador final, obtenido por medio de una búsqueda a profundidad partiendo del nodo (a) y con prioridad de selección en orden alfabético:



Se observa que el árbol generador obtenido por medio de búsqueda a profundidad es diferente del que se obtuvo por búsqueda a lo ancho. Esto muestra que un grafo puede tener dos o más árboles generadores, los cuales se obtienen en función del procedimiento aplicado.

8.6.4 Árbol generador mínimo

Se llama árbol generador mínimo de un grafo conexo a aquel que permite mantener unidos a todos los vértices y que no tiene ciclos, además de que es la forma más barata o corta ya que la trayectoria o costo es mínimo. Existen varios campos en donde es conveniente mantener funcionando los sistemas al menor costo (algunos de ellos son las redes telefónicas, eléctricas, carreteras, de alcantarillado, etc.) y en donde es conveniente mantener la comunicación entre los nodos pero sin aristas redundantes o demasiado costosas.

Para obtener el árbol generador mínimo en un grafo conexo con pesos se puede utilizar alguno de los dos métodos siguientes: el de Prim o el de Kruskal.

Método de Prim

Robert Prim desarrolló un método para obtener un árbol generador mínimo a partir de un grafo conexo etiquetado. Este método se puede poner en práctica aplicando un algoritmo que se utiliza cuando se desea eliminar conexiones redundantes y dejar sólo las aristas del grafo que tengan menor distancia o costo, además de ser sólo las necesarias para mantener conectados todos los nodos.

En este método los vértices se dividen en dos conjuntos: vértices integrados (I) —que son los que forman parte del árbol generador mínimo— y vértices no integrados (N). El conjunto ***Árbol*** contiene todas las aristas que se van integrando en cada iteración, y así en cada paso se agrega un vértice en el conjunto I mientras que el conjunto N es disminuido en uno. El algoritmo se puede resumir de la siguiente forma:

1) Se selecciona una arista de costo mínimo (s, x).
2) Se integran los vértices de esa arista al conjunto I = {s, x}, ya que serán los primeros nodos que conforman el árbol generador mínimo.
3) Todos los demás nodos del grafo pertenecen al conjunto N = N – I.
4) El Árbol tiene como primera arista (s, x), esto es, Árbol = {(s, x)}.
5) Mientras el conjunto N sea diferente de vacío ($N \neq \varnothing$) iterar. En caso contrario terminar.
6) Seleccionar nueva arista con costo mínimo (w, x). Una característica es que w debe estar contenido en I ($w \in I$) mientras que x debe estar en N ($x \in N$).
7) Se agrega al árbol la arista mínima seleccionada: Árbol = Árbol $\cup$ (w, x).
8) Se agrega al conjunto I el nuevo nodo seleccionado: I = I $\cup$ (x).
9) Se elimina el nodo seleccionado del conjunto N, esto es, N = N – (x).
10) Regresar al paso 5.

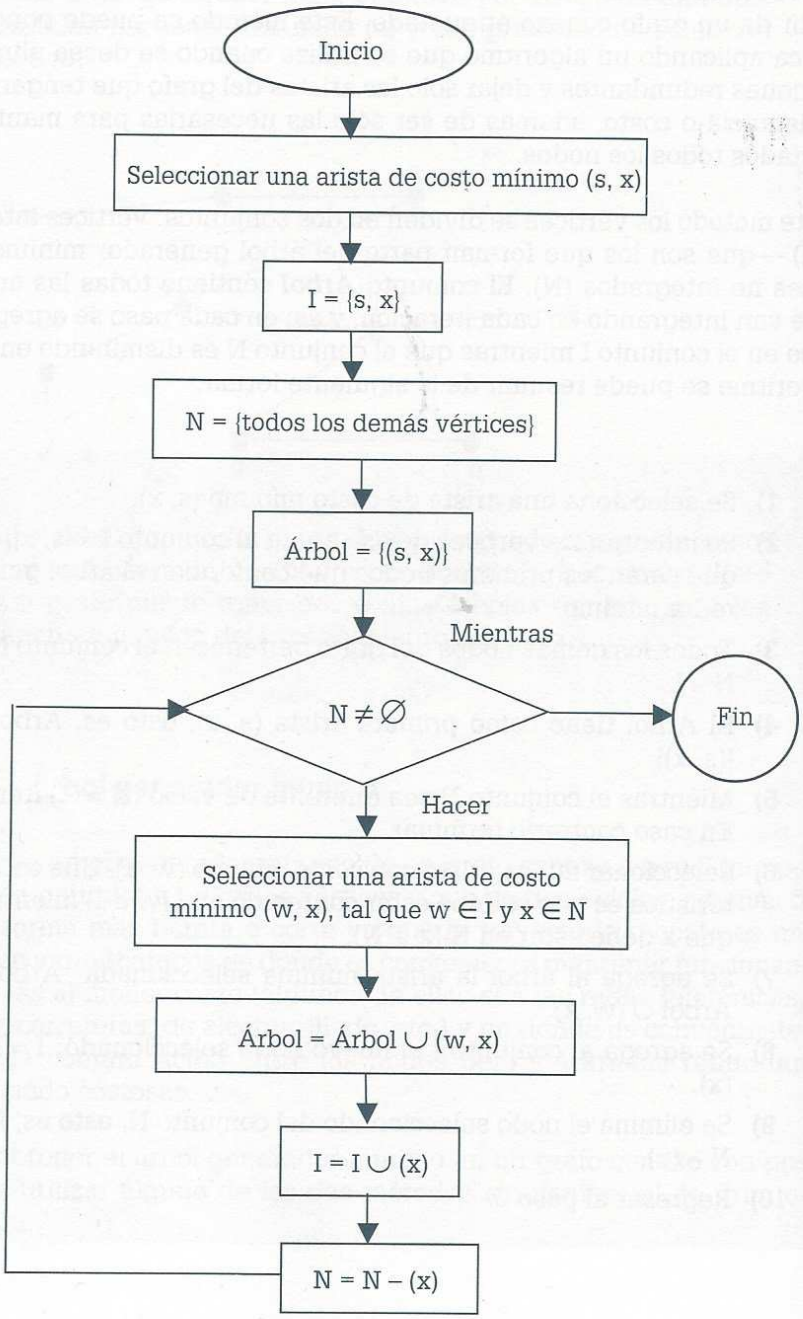
Robert Clay Prim
(1921)

Nació en 1921 en Sweetwater, Estados Unidos. En 1941 terminó la carrera de ingeniero electricista en la Universidad de Princeton, posteriormente en la misma Universidad obtuvo el grado de doctor en el área de matemáticas y fue investigador de 1948 a 1949.

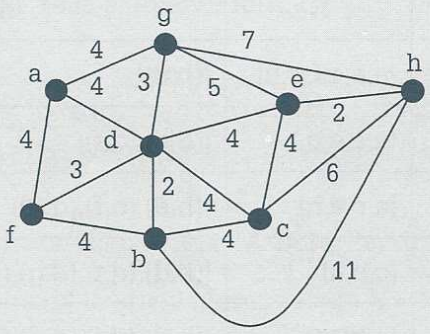
En plena segunda guerra mundial, Prim trabajó como ingeniero para *General Electric*. Desde 1944 hasta 1949 fue contratado por la *United States Naval Ordenance Lab* como ingeniero y más tarde como matemático. En los laboratorios *Bell* trabajó como director de investigación matemática desde 1958 hasta 1961 y ahí Prim desarrolló el conocido Algoritmo de Prim. Después de su estancia en los laboratorios *Bell*, Prim pasó a ser vicepresidente de investigación en *Sandia National Laboratorios*.

Durante su carrera en los laboratorios *Bell*, Robert Prim junto a su compañero Joseph Kruskal desarrolló dos algoritmos diferentes para encontrar los árboles generadores mínimos en un grafo ponderado.

Otra forma de expresar este algoritmo es por medio de un diagrama de flujo como el que se muestra a continuación:



Ejemplo 8.6. Determinar en el siguiente grafo el árbol generador mínimo aplicando el método de Prim.



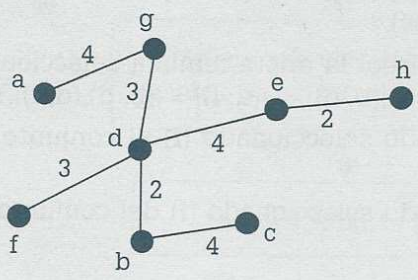
1) Supóngase que se selecciona la arista (b, d) ya que su costo es de 2.
2) I = {b, d}.
3) N = {a, c, e, f, g, h}.
4) Árbol = {(b, d)}.
5) Mientras (N ≠ ∅) iterar, en caso contrario terminar. En este caso N es diferente de vacío, por lo tanto se debe iterar.
6) Seleccionar nueva arista. Considérese que se selecciona (d, f) ya que su costo es de 3. No puede ser (e, h) porque un requisito es que uno de los vértices de la arista esté seleccionado (w ∈ I) y el otro no (x ∈ N). En nuestro caso (d ∈ I) y el otro no (f ∈ N).
7) Se agrega al árbol la arista mínima seleccionada (Árbol = Árbol ∪ {(d, f)} = {(b, d)} ∪ {(d, f)} = {(b, d),(d, f)}.)
8) Integrar el nodo seleccionado (f) al conjunto I = I ∪ (f) = {b, d, f}.
9) Eliminar el nodo seleccionado (f) del conjunto N = {a, c, e, g, h}.
10) Regresar al paso 5.

En la siguiente tabla se resume el resultado de cada una de las iteraciones:

Iteración	I	N	Árbol
1	{b,d}	{a,c,e,f,g,h}	{(b,d)}
2	{b,d,f}	{a,c,e,g,h}	{(b,d),(d,f)}
3	{b,d,f,g}	{a,c,e,h}	{(b,d),(d,f),(d,g)}
4	{b,d,f,g,a}	{c,e,h}	{(b,d),(d,f),(d,g),(g,a)}
5	{b,d,f,g,a,c}	{e,h}	{(b,d),(d,f),(d,g),(g,a),(b,c)}
6	{b,d,f,g,a,c,e}	{h}	{(b,d),(d,f),(d,g),(g,a),(b,c),(d,e)}
n–1	{b,d,f,g,a,c,e,h}	∅	{(b,d),(d,f),(d,g),(g,a),(b,c),(d,e),(e,h)}

Aquí n es el número de vértices que integran el grafo.

De esta forma, con el conjunto Árbol se puede obtener el siguiente árbol generador mínimo:



Cuando se tienen aristas de la misma longitud, como en el ejemplo anterior, es posible obtener más de un árbol generador mínimo, por lo tanto si en el momento de seleccionar la arista se opta por otra rama de igual longitud (o peso) entonces el árbol generador mínimo es diferente.

Método de Kruskal

Este método se destaca por integrar al árbol generador mínimo a aquellas aristas que tengan menor costo, cuidando siempre que no se formen ciclos. El algoritmo se puede resumir de la siguiente forma:

1) Ordenar los costos de las aristas del grafo en forma ascendente y colocar en el conjunto N las aristas, de acuerdo a este orden.
2) Se incluye la arista con menor costo Árbol = {(s, t)}.
3) Se resta del conjunto N la arista seleccionada N = N – (s, t).
4) Se registra en un contador el número de aristas incluidas, en este caso C = 1, ya que sólo se ha incluido una arista.
5) Mientras C ≤ (n – 1) iterar. En caso contrario finalizar.
6) Si Árbol ∪ (w, x) no forma ciclos, entonces
   Árbol = Árbol ∪ (w, x)
   C = C + 1
7) Regresar a paso 4.

Aquí se tiene que:

Árbol: conjunto que contiene las aristas que integran el árbol generador mínimo.

N: conjunto que contiene las aristas que aún no han sido seleccionadas.

C: variable para contar las aristas incluidas.

n: número de vértices que conforman el grafo.

(s, t): arista con menor costo que se integra al árbol generador.

(w, x): arista a integrarse al árbol generador, siempre y cuando con ella no se formen ciclos.

Joseph B. Kruskal

Fue investigador del Centro de Matemáticas de los Laboratorios Bell, y en 1956 descubrió el algoritmo para obtener un árbol generador mínimo a partir de un grafo ponderado: el árbol del costo total mínimo también llamado árbol recubridor euclídeo mínimo. Este problema es un problema típico de optimización combinatoria, y fue considerado originalmente por Otakar Boruvka (1926) mientras estudiaba la necesidad de electrificación rural en el sur de Moravia en Checoslovaquia.

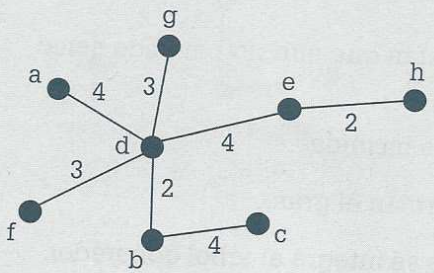
La aplicación típica de este problema es el diseño de redes telefónicas: una empresa con diferentes oficinas trata de trazar líneas de teléfono para conectarlas unas con otras. La compañía telefónica ofrece esta interconexión, pero presenta tarifas o costos diferentes por conectar cada par de oficinas. ¿Entonces cómo conectar las oficinas al mínimo costo total?

La formulación de este problema también ha sido aplicada para determinar soluciones en diversas áreas: diseño de redes de transporte, diseño de redes de telecomunicaciones (TV por cable, sistemas distribuidos), interpretación de datos climatológicos, visión artificial (análisis de imágenes), extracción de rasgos de parentesco, análisis de clusters y búsqueda de superestructuras de quasar, plegamiento de proteínas, reconocimiento de células cancerosas, y otros.

Ejemplo 8.7. La siguiente tabla muestra los resultados obtenidos en cada una de las iteraciones del algoritmo de Kruskal aplicado al caso del ejemplo 8.6.

C	Árbol	N
0	∅	{(b,d),(e,h),(d,g),(d,f),(a,d),(a,f),(a,g),(b,c),(b,f),(c,d),(d,e),(c,e),(e,g),(c,h),(g,h),(b,h)}
1	{(b,d)}	{(e,h),(d,g),(d,f),(a,d),(a,f),(a,g),(b,c),(b,f),(c,d),(d,e),(c,e),(e,g),(c,h),(g,h),(b,h)}
2	{(b,d),(e,h)}	{(d,g),(d,f),(a,d),(a,f),(a,g),(b,c),(b,f),(c,d),(d,e),(c,e),(e,g),(c,h),(g,h),(b,h)}
3	{(b,d),(e,h),(d,g)}	{(d,f),(a,d),(a,f),(a,g),(b,c),(b,f),(c,d),(d,e),(c,e),(e,g),(c,h),(g,h),(b,h)}
4	{(b,d),(e,h),(d,g),(d,f)}	{(a,d),(a,f),(a,g),(b,c),(b,f),(c,d),(d,e),(c,e),(e,g),(c,h),(g,h),(b,h)}
5	{(b,d),(e,h),(d,g),(d,f),(a,d)}	{(a,f),(a,g),(b,c),(b,f),(c,d),(d,e),(c,e),(e,g),(c,h),(g,h),(b,h)}
6	{(b,d),(e,h),(d,g),(d,f),(a,d),(b,c)}	{(a,f),(a,g),(b,f),(c,d),(d,e),(c,e),(e,g),(c,h),(g,h),(b,h)}
7	{(b,d),(e,h),(d,g),(d,f),(a,d),(b,c),(d,e)}	{(a,f),(a,g),(b,f),(c,d)(c,e),(e,g),(c,h),(g,h),(b,h)}

De esta forma el árbol generador mínimo queda de la siguiente manera:



En relación con los ejemplos 8.6 y 8.7 es necesario observar que el árbol generador mínimo obtenido por el método de Kruskal es diferente al que se obtuvo por el método de Prim, debido principalmente a características propias del método cuando se presentan aristas con la misma ponderación.

8.7 Recorrido de un árbol

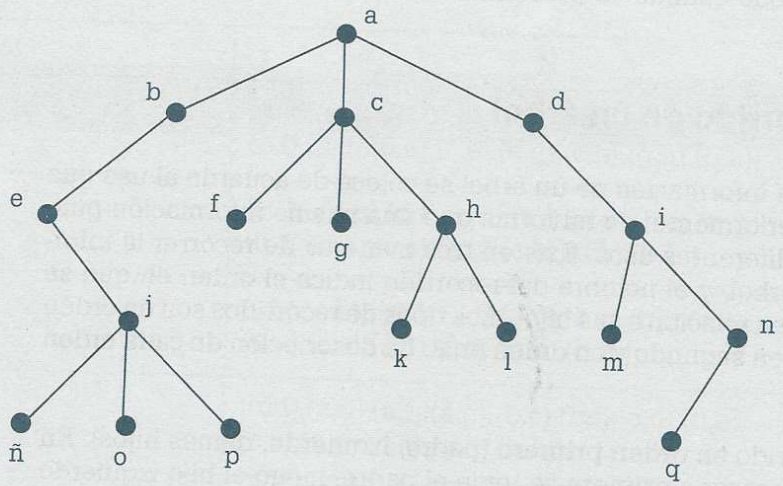
Por lo general la información de un árbol se coloca de acuerdo al uso que se le dará posteriormente, de tal forma que una misma información puede servir para diferentes usos. Existen tres maneras de recorrer la información de un árbol, y el nombre del recorrido indica el orden en que se coloca el padre en relación a sus hijos. Los tipos de recorridos son en orden primero, en orden segundo y en orden final. La descripción de cada orden es la siguiente:

a) Recorrido en **orden primero** (padre, izquierdo, demás hijos). En este recorrido **primero** se toma el padre, luego el hijo izquierdo y al final los demás hijos. Se comienza por la raíz, después se sigue por el nodo de la izquierda, si este nodo tiene hijos se sigue por el de la izquierda hasta llegar a la hoja. Si esta hoja tiene hermanos se toma el que está más cercano a ella (más a la izquierda). Después de que se termina con la rama izquierda, continúa con la rama más cercana a ella y así sucesivamente hasta terminar con el recorrido de todo el árbol.

b) Recorrido en **orden segundo** (izquierdo, padre, demás hijos). En este recorrido primero se toma el hijo izquierdo, **segundo** el padre y al final los demás hijos. Comienza con la hoja que se encuentra más a la izquierda del árbol, después se regresa al padre y posteriormente a todos los hermanos, después se regresa al padre de esta rama y con las ramas de éste (tomando siempre la que está más a la izquierda) y así sucesivamente hasta terminar el recorrido del árbol completo.

c) Recorrido en **orden final** (izquierdo, demás hijos, padre). En éste recorrido se toma primero el hijo izquierdo, después los demás hijos y al **final** el padre. Se comienza en la hoja que se encuentra más a la izquierda del árbol, después se continúa con los hermanos, si éstos tienen hijos primeramente recorre los hijos y hasta el final el padre, dando preferencia a los hijos de la izquierda y hasta el final el padre. En este tipo de recorrido lo último que se recorre es la raíz, ya que tienen preferencia los hijos sobre el padre.

Ejemplo 8.8. Considérese el siguiente árbol:



Entonces se tienen los siguientes recorridos.

- Primero: (a,b,e,j,ñ,o,p,c,f,g,h,k,l,d,i,m,n,q).
- Segundo: (e,ñ,j,o,p,b,a,f,c,g,k,h,l,d,m,i,q,n).
- Final: (ñ,o,p,j,e,b,f,g,k,l,h,c,m,q,n,i,d,a).

8.7.1 Recorridos en árboles etiquetados

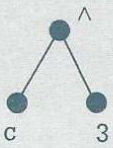
En el área de la computación los árboles etiquetados se usan para evaluar expresiones matemáticas, y las constantes o variables se ubican en las hojas mientras que los operadores (signos aritméticos o funciones) se colocan como nodos intermedios.

Ejemplo 8.9. Elaborar el árbol binario completo que representa la expresión:

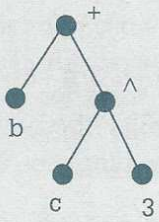
$$\frac{a(b + c^3)}{b - (c + d)e}$$

y determinar los recorridos en orden primero, segundo y final.

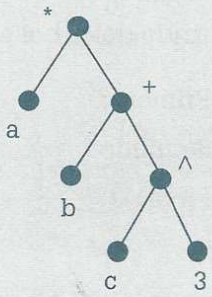
Solución. De acuerdo a la jerarquía de operación que se usa en computación el árbol se estructura de la siguiente forma:



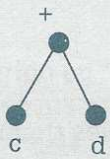
c^3



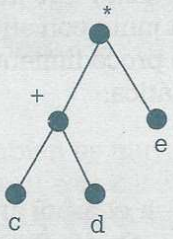
$b + c^3$



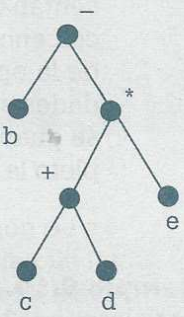
$a(b + c^3)$



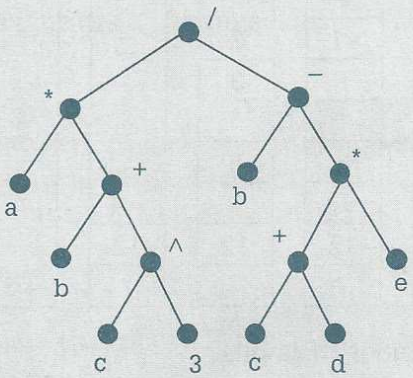
$c + d$



$(c + d)e$



$b - (c + d)e$



$$\frac{a(b + c^3)}{b - (c + d)e}$$

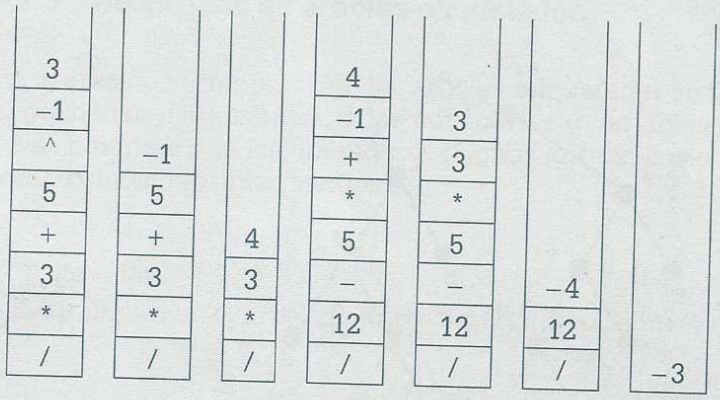
Hay que observar cómo crece el árbol de acuerdo con la literal de la ecuación: si la literal se encuentra a la derecha crece hacia la derecha y si se encuentra a la izquierda crece en ese mismo sentido.

El recorrido del árbol en orden primero, segundo y final es:

Primero:	/*a +b^c3 –b* +cde
Segundo:	a*b +c^3/b –c +d*e
Final:	abc3^ + *bcd +e*–/

El recorrido en "orden primero" también recibe el nombre de "notación polaca" y es una de las maneras más usadas para evaluar expresiones matemáticas dentro de la computación. La forma en que se realiza es colocando en una pila operadores aritméticos y variables, hasta que se presentan dos variables seguidas se sacan de la pila junto con el operador más cercano y se realiza la operación correspondiente, se guarda el resultado de la operación en la pila hasta que nuevamente se presenten dos cantidades seguidas, se sacan junto con su operador, se realiza la operación y se guarda en la pila. Este procedimiento se repite hasta evaluar por completo la expresión matemática.

Ejemplo 8.10. Supóngase que en el árbol anterior las variables tienen los valores a = 3, b = 5, c = –1, d = 4 y e = 3. Si el recorrido en orden primero es /*a + b^c3 –b* + cde, la forma en que se evalúa la expresión es:



Estados de la pila.

Se puede observar en primer lugar que se introducen en la pila todos los valores y signos aritméticos, hasta que se tengan dos variables (o cantida-

des seguidas), en este caso cuando están seguidos –1 y 3 que son los valores de c y su exponente. En el siguiente estado de la pila se observa que se sacó –1 y se elevó a la potencia 3, (–1)^3. Como nuevamente quedan dos valores juntos, se extraen de la pila y se les aplica el operador más cercano a ellos, "+", por lo que se obtiene 5 + (–1) = 4 y se vuelve a guardar el resultado en la pila.

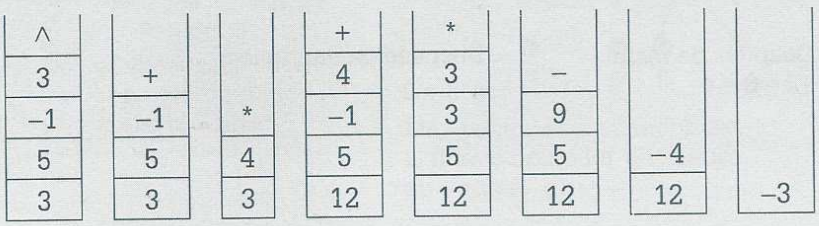
En el tercer estado de la pila nuevamente vuelven a quedar juntas dos cantidades, mismas que se extraen de la pila con su operador más cercano para obtener 3*4=12. Como ya no existen dos cantidades seguidas, es necesario introducir a la pila más valores.

Posteriormente se extraen de la pila –1 y 4 juntamente con el operador "+" para llevar a cabo la operación (–1 + 4 = 3) y así sucesivamente hasta obtener como resultado el –3 que está en el último estado de la pila, el cual es el mismo resultado obtenido al evaluar la expresión matemática:

$$\frac{a(b + c^3)}{b(c + d)e} = \frac{3(5 + (-1)^3)}{5 - (-1 + 4)^3} = -3$$

El recorrido en "orden final" también es muy útil en la evaluación de expresiones matemáticas, sólo que en este caso se introducen en la pila los valores de las variables hasta que llega un signo aritmético. En este momento se extraen de la pila el signo aritmético y los valores que están más cerca de él, se realiza la operación y se guarda el resultado. Cuando llega otro signo aritmético se vuelven a extraer dos valores juntamente con el signo para realizar la operación correspondiente y guardar el resultado. Estos pasos se llevan a cabo hasta terminar de evaluar la expresión matemática como se muestra a continuación.

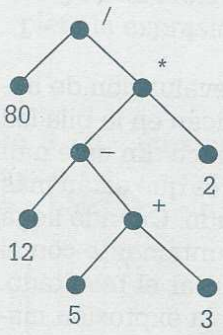
Ejemplo 8.11. Supóngase nuevamente que los valores son a = 3, b = 5, c = –1, d = 4 y e = 3. Si el recorrido en orden final es abc3^ + *bcd +e*–/, la forma en que se evalúa la expresión es:

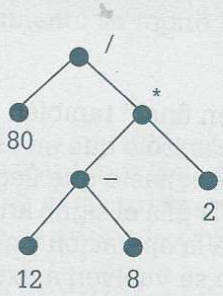


Primero se introducen valores a la pila hasta que se tenga un signo aritmético. Al llegar el signo "^" se extraen los valores más cercanos a él, se lleva a cabo la operación (–1)^ 3 = –1 y se guarda el resultado en la pila como lo muestra la figura.

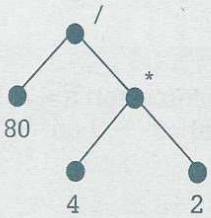
La siguiente información de acuerdo al orden final es el signo "+", como lo muestra la pila correspondiente, mismo que se saca de la pila junto con los valores que están cercanos a él para realizar la siguiente operación 5 + (–1) = 4. Se continúa de esta manera guardando la información en la pila y sacando las dos últimas cantidades guardadas, una vez que se presenta un signo aritmético, para llevar a cabo la evaluación correspondiente hasta obtener el resultado que se muestra en el último estado de la pila.

En el recorrido por "orden segundo" la evaluación de la información en el árbol debe ser por niveles y de izquierda a derecha. Por ejemplo, si los valores son a = 80, b = 12, a = 5, d = 3 y e = 2, primero se evalúan los niveles más bajos y si hay empate en cuanto al nivel tienen prioridad las operaciones que están más a la izquierda. Sustituyendo valores por variables la evaluación se lleva a cabo de la siguiente forma:

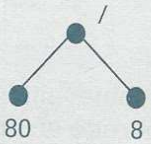




Después de sumar 5 + 3 = 8



Después de restar
12 – 8 = 4



Después de multiplicar
4*2 = 8

Finalmente
se divide
80/8 = 10

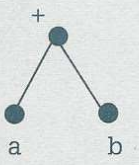
Cuando se evalúa una expresión matemática puede haber más de una hoja que tenga el mismo valor, así como también es frecuente que dos o más nodos intermedios tengan el mismo signo aritmético.

Ejemplo 8.12. Considérese que el recorrido de un árbol binario en orden final es ab + cd/ef – gb + *–*. Determinar:

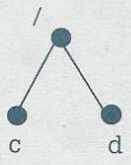
a) El árbol.

b) El recorrido en orden primero y segundo.

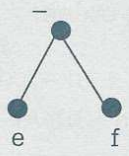
Solución. En el recorrido en orden final una operación se lleva a cabo hasta que se tienen las dos cantidades y el signo de la operación que se realizará. El procedimiento completo para estructurar el árbol es:



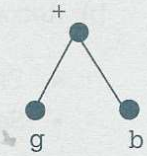
ab +



cd/

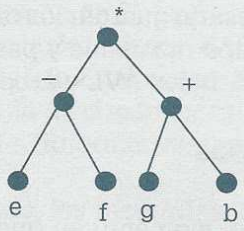


ef–

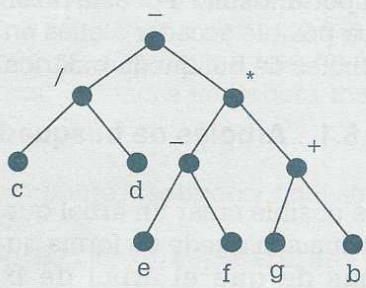


gb +

En todos los casos anteriores primero se deben de tener dos cantidades y la operación aritmética para de esa manera unir los nodos. La operación aritmética se lleva a cabo con las dos cantidades más cercanas a la izquierda del signo. Una vez que se realiza la operación el resultado se considera como una cantidad, misma que se puede tomar en el momento en que se presente un signo. Hasta ahora se lleva evaluado ab + cd/ef – gb + del recorrido en orden posterior.



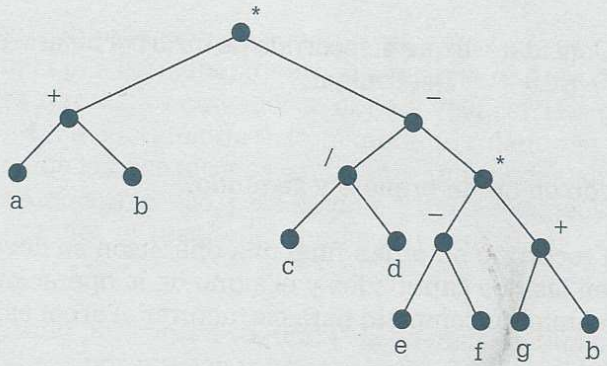
Dos últimos resultados con el nuevo operador aritmético ef – gb+*



Más recientes resultados con el nuevo operador aritmético ab + cd/ef – gb+* –

Finalmente se obtiene el siguiente árbol.

a)



b) Recorridos en orden:

- Primero: * + ab – /cd* – ef + gb
- Segundo: a + b*c/d – e – f*g + b

8.8 Búsquedas

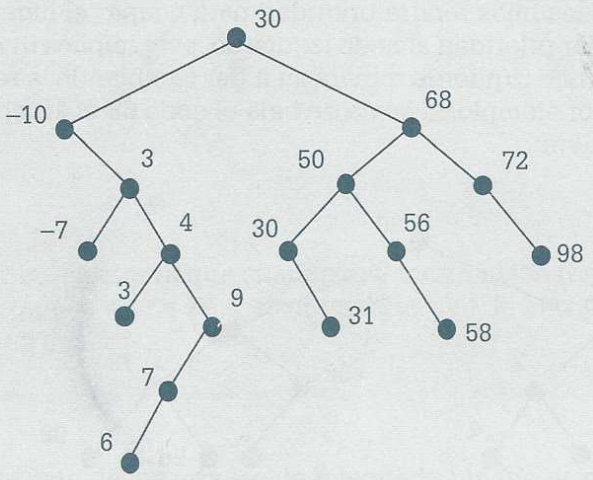
Se puede considerar que uno de los usos principales de la computadora es guardar la información para después recuperarla en el orden deseado y en forma rápida. Cuando la información es pequeña no hay ningún problema ya que el tiempo en que encuentra la información almacenada es relativamente pequeño, pero conforme ésta aumenta el tiempo de respuesta es importantísimo. Por esta razón es necesario guardar los datos de forma que sea posible acceder a ellos en un tiempo razonable y para esto se utilizan árboles de búsqueda binarios (ABB), árboles AVL y árboles B.

8.8.1 Árboles de búsqueda binarios

Es posible crear un árbol que desde el momento en que se captura la información quede de forma que sea relativamente fácil acceder a ella, además de que el árbol de búsqueda binario es sencillo de crear y manipular.

Ejemplo 8.13. Crear un árbol de búsqueda binario con la siguiente información: 30, –10, 3, 4, 9, 68, 50, 30, 3, 7, 6, 72, 98, –7, 56, 31, 58.

Solución. Se entiende que la raíz es el primer dato, en este caso el 30, y que los siguientes datos de colocarán a la izquierda si son menores que 30 o a la derecha si son mayores o iguales a 30. Lo mismo sucederá en los nodos restantes: el nuevo dato estará a la izquierda de un nodo cualquiera si es menor y a la derecha si es mayor o igual a él. Con estas reglas para estructurar el árbol se tendrá lo siguiente:



Una vez estructurado el árbol es posible obtener la información de sus nodos siguiendo las mismas reglas que se usaron para elaborarlo: se buscará a la izquierda si es menor que la información del nodo y a la derecha si es mayor o igual a él. Por ejemplo, si el dato a buscar es el número 56, dado que es mayor que 30 se busca a la derecha ya que se tiene la certeza de que todos los datos que están a la izquierda de 30 son menores que él. A la derecha de 30 está el 68, pero como el 56 es menor ahora se busca a la izquierda para llegar al 50 y como el 56 es mayor se busca a la derecha para localizarlo finalmente. Si en algún caso se llega a una hoja y no se encuentra la información que se busca, entonces se deberá mandar un mensaje de "información inexistente".

Por otro lado, los recorridos en orden primero, segundo y final son:

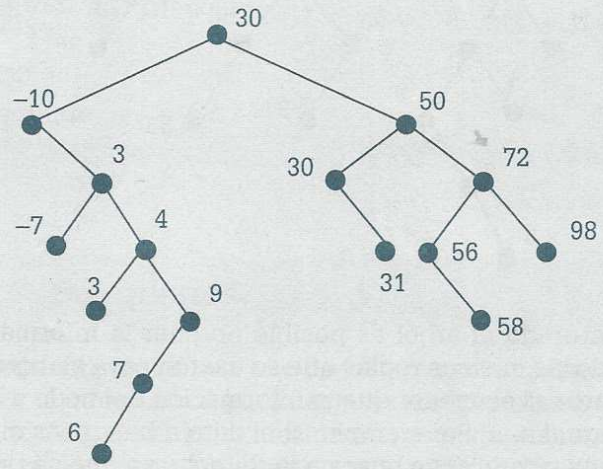
Primero: 30, –10, 3, –7, 4, 3, 9, 7, 6, 68, 50, 30, 31, 56, 58, 72, 98

Segundo: –10, –7, 3, 3, 4, 6, 7, 9, 30, 30, 31, 50, 56, 58, 68, 72, 98

Final: –7, 3, 6, 7, 9, 4, 3, –10, 31, 30, 58, 56, 50, 98, 72, 68, 30

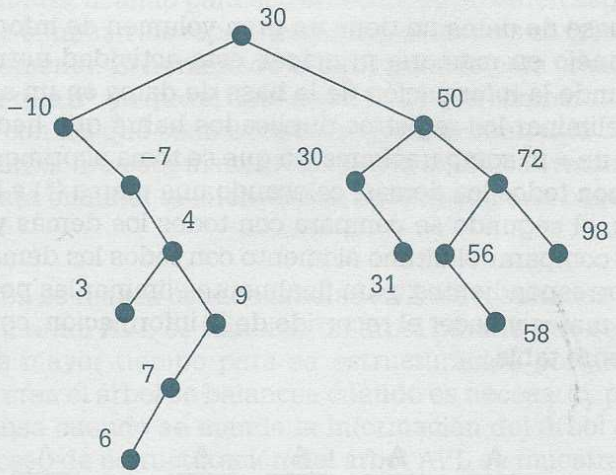
Hay que observar que el recorrido en orden segundo permite obtener la información del árbol ordenada en forma ascendente. Esa es una de las ventajas de acomodar la información en un árbol con estas características, ya que se puede encontrar un dato en forma relativamente fácil y además se puede obtener el total de la información del árbol en forma ordenada.

Considérese ahora que se desea dar de baja algunos elementos del árbol. Si el nodo que se da de baja es una hoja no hay ningún problema, solamente se quita el nodo y la demás información no se mueve de su lugar. Pero cuando el nodo es intermedio (o sea que tiene hijos) es recomendable establecer cuál de los hijos tendrá prioridad para ocupar el lugar del padre. Se acostumbra dar prioridad al nodo izquierdo, esto implica que al desaparecer el padre el hijo izquierdo tomará su lugar cambiando la reestructuración del árbol. Por ejemplo, si se da de baja el nodo 68 el árbol quedará de la siguiente manera



Se entiende que si el nodo que ocupa la posición del padre tiene hijo derecho, ese hijo derecho con toda su descendencia pasa a formar parte de la rama derecha como ocurre con el nodo 56 y su descendiente 58, que dependían del 50 y ahora dependen del 72.

Si ahora se da de baja al nodo 3 el árbol queda de la siguiente manera:



En este caso se observa que cuando hay información repetida, como es el nodo 3, el que se da de baja es el primero que se encuentra ya que está más cercano a la raíz.

Un problema importante de los árboles de búsqueda binario es que algunas veces crecen en forma descontrolada. Cuando el volumen de información es considerable el tiempo de búsqueda se incrementa, ya que se trata de árboles no balanceados, y en estos casos es conveniente utilizar árboles AVL o árboles B.

8.9 Aplicación de los árboles

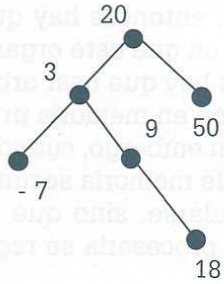
La estructura de árbol, independientemente de si se trata de árboles binarios, AVL o B, se usa principalmente para guardar la información organizada de tal manera que sea posible tener un rápido acceso a ella. La diferencia principal que permite decidir qué tipo de árbol usar depende de la forma en que está estructurada la información, pero sobre todo del volumen de la misma (si la información es poca, entonces hay que usar árboles binarios o AVL dependiendo de la forma en que esté organizada; si el volumen de información es grande entonces hay que usar árboles B) ya que cuando es posible manipular la información en memoria principal, los árboles binarios y AVL son recomendables. Sin embargo, cuando no se puede transferir la información completamente de memoria secundaria a memoria principal para posteriormente manipularse, sino que se trae parcialmente a memoria principal y después de procesarla se regresa al

dispositivo de almacenamiento secundario, entonces es recomendable la utilización de árboles B o B++. Por ejemplo, para eliminar información duplicada en una base de datos.

Si la base de datos no tiene un gran volumen de información y es posible su manejo en memoria principal, esta actividad normalmente se realiza colocando la información de la base de datos en un arreglo de forma que para eliminar los registros duplicados habrá que hacer en el peor de los casos $nn = n^2$ comparaciones, ya que se toma el primer elemento y se compara con todos los demás, colocando una marca (*) a los elementos duplicados, el segundo se compara con todos los demás y así sucesivamente hasta comparar el último elemento con todos los demás y colocar las marcas correspondientes, para finalmente eliminar las posiciones que tengan dicha marca y hacer el recorrido de la información, como se muestra en la siguiente tabla.

	A	A	A	A		A	Finalmente A
1	20	20	20	20		20	20
2	3	3	3	3		3	3
3	–7	–7	–7	–7		–7	–7
4	50	50	50	50		50	50
5	9	9	9	9		9	9
6	3	3	*	*		*	18
7	20	*	*	*		*	
8	–7	–7	–7	*		*	
9	18	18	18	18		18	
Iteración	1ª	2ª	3ª	4ª	.....	n	

Sin embargo, si esto mismo se lleva a cabo por medio de un árbol binario, el cual se crea con la información de la base de datos en el momento en que se manda la información a memoria principal, de tal manera que sólo se registran como nodos aquellos registros que no están duplicados y después se regresa nuevamente la información a la base de datos, entonces el número de comparaciones se reduce sustancialmente de tal forma que con la información anterior se tendría el siguiente árbol binario.



	A
1	20
2	3
3	–7
4	9
5	18
6	50

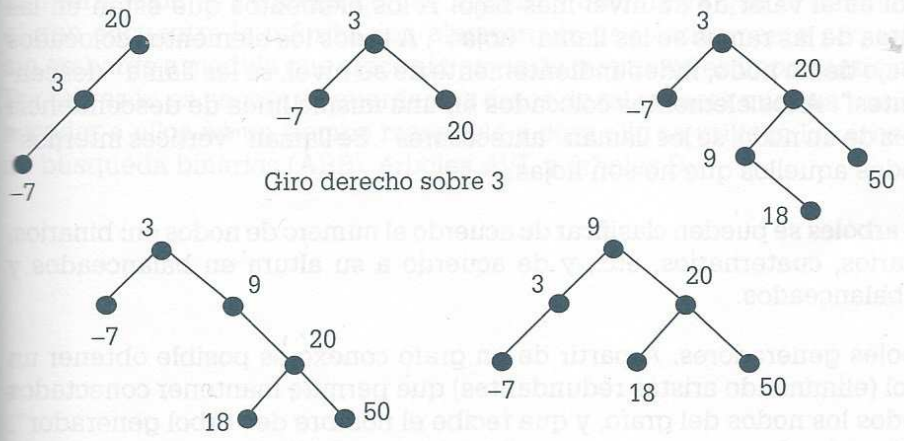
Primero

	A
1	–7
2	3
3	9
4	18
5	20
6	50

Segundo

Si después de crear el árbol se regresa la información a la base de datos en orden *primero*, la información conserva una estructura semejante a la que tenía, como se ve en la figura anterior, pero si se pasa la información del árbol a la base de datos, usando para ello el recorrido en orden *segundo*, entonces no sólo se eliminan los repetidos sino que además se colocan ordenados ascendentemente. El número de comparaciones para crear un árbol binario es menor que n^2, ya que el dato no se tiene que comparar con todos sino solamente con los que sean necesarios para encontrar la colocación del dato en el árbol. De esta manera se reduce significativamente el tiempo requerido para eliminar la información repetida en una base de datos.

Si esta misma operación se realiza usando árboles AVL los resultados son semejantes, ya que un árbol AVL es también un árbol binario que seguramente requerirá de mayor tiempo para su estructuración porque al mismo tiempo que se crea el árbol se balancea cuando es necesario, pero ese tiempo se compensa cuando se manda la información del árbol a la base de datos. El proceso de estructuración del árbol AVL se muestra en la siguiente figura.



Doble giro, uno derecho y luego izquierdo sobre 9

Hay que recordar que para buscar información en un árbol AVL el número de comparaciones en el peor de los casos es log n, sin embargo en un árbol binario que no es AVL el peor de los casos es n (cuando los hijos están alineados en un mismo lado con respecto al padre).

La diferencia entre los árboles B, AVL y binarios es que para usar los árboles B se requiere más trabajo para desarrollar y programar las operaciones de altas, bajas y cambios, pero una vez que ya se tienen programados los algoritmos, la velocidad de respuesta en los árboles B es significativamente menor ya que además de mantenerse balanceados como ocurre con los árboles AVL, la propiedad de tener información de más de un registro

de una base de datos en un mismo nodo permite el manejo de un mayor volumen de información. Al incrementarse el número de registros en un mismo nodo se reduce la altura del árbol, las operaciones para balancear el árbol decrecen y de esta manera se aumenta la eficiencia de los árboles B. Considérese también que los árboles B se utilizan cuando la memoria principal de la computadora no es lo suficientemente grande como para manejar toda la información de la base de datos, sino que se aprovechan las características de los discos rígidos (memoria secundaria) de tal manera que un nodo ocupa un bloque de disco con espacio, para n registros de la base de datos, procurando que la información quede en pistas o cilindros cercanos para evitar la pérdida de tiempo en la lectura y grabación de la información.

8.10 Resumen

Un árbol es un grafo conexo que no tiene ciclos, ni lazos, ni lados paralelos, además de que está compuesto por niveles y al más alto de la jerarquía se le llama "raíz". La raíz tiene un nivel 0, los vértices inmediatamente debajo de la raíz tienen un nivel 1 y así sucesivamente. La altura o peso de un árbol es el valor de su nivel más bajo. A los elementos que están en las puntas de las ramas se les llama "hojas". A todos los elementos colocados debajo de un nodo, independientemente de su nivel, se les llama "descendientes". A los elementos colocados en una misma línea de descendencia antes de un nodo, se les llaman "antecesores". Se llaman "vértices internos" a todos aquellos que no son hojas.

Los árboles se pueden clasificar de acuerdo al número de nodos en: binarios, trinarios, cuaternarios, etc., y de acuerdo a su altura en balanceados y desbalanceados.

Árboles generadores. A partir de un grafo conexo es posible obtener un árbol (eliminando aristas redundantes) que permite mantener conectados a todos los nodos del grafo, y que recibe el nombre de "árbol generador". Existen dos formas en que es posible obtener el árbol generador: usando búsqueda en profundidad, en donde se busca por ramas de arriba hacia abajo y de izquierda a derecha, o bien por medio de búsqueda a lo ancho en donde la búsqueda se hace por niveles.

Un árbol generador mínimo de un grafo es aquel que permite mantener unidos a todos los vértices y que no tiene ciclos, pero que además es la forma más barata o corta ya que la trayectoria o costo es mínimo. Existen varios campos en donde es conveniente mantener funcionando los sistemas al menor costo y algunos de ellos son las redes telefónicas, eléctricas, carreteras, de alcantarillado, etc., en donde es conveniente mantener la comunicación entre los nodos pero sin aristas redundantes o demasiado costosas. Para obtener el árbol generador mínimo en un grafo conexo con pesos es posible aplicar el método de Prim o bien el de Kruskal.

Existen tres maneras de recorrer la información de un árbol y el nombre del recorrido indica el orden en que se coloca el padre en relación a sus hijos. Los recorridos son:

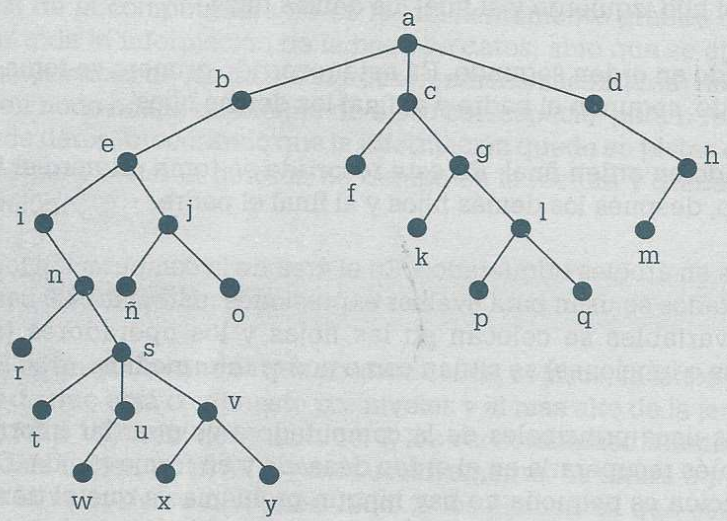
- Recorrido en orden primero. En este recorrido primero se toma el padre, luego el hijo izquierdo y al final los demás hijos.
- Recorrido en orden segundo. En este recorrido primero se toma el hijo izquierdo, segundo el padre y al final los demás hijos.
- Recorrido en orden final. En este recorrido se toma primero el hijo izquierdo, después los demás hijos y al final el padre.

Recorridos en árboles etiquetados. En el área de la computación los árboles etiquetados se usan para evaluar expresiones matemáticas. Las constantes o variables se colocan en las hojas y los operadores (signos aritméticos o funciones) se sitúan como nodos intermedios.

Uno de los usos principales de la computadora es guardar información para después recuperarla en el orden deseado y en forma rápida. Cuando la información es pequeña no hay ningún problema ya que el tiempo en el que encuentra la información almacenada es relativamente pequeño, sin embargo a medida que crece el tiempo de respuesta es importantísimo. Por tal razón es necesario guardar los datos de tal manera que sea posible acceder a ellos en un tiempo razonable y para ello se utilizan los árboles de búsqueda binarios (ABB), árboles AVL y árboles B.

8.11 Problemas

8.1 Considérese el siguiente árbol.

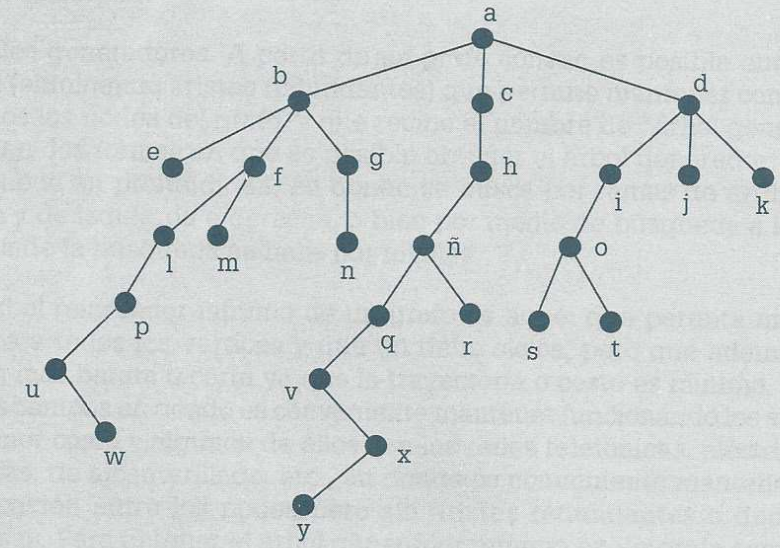


a) ¿Cuál es el recorrido en orden primero, segundo y final?

b) Balancear el árbol a trinario.

c) ¿Cuál es el recorrido en orden primero, segundo y final, ahora que está balanceado?

8.2 Considérese el siguiente árbol.



a) ¿Cuál es el recorrido en orden primero, segundo y final?

b) Balancear el árbol a binario.

c) ¿Cuál es el recorrido en orden primero, segundo y final, ahora que está balanceado?

8.3 Sea la siguiente expresión $\dfrac{(a + b) - bc}{\dfrac{a}{c} + \dfrac{dc}{b}}$. Determinar

a) El árbol binario que representa dicha expresión.

b) El recorrido en orden:

- Primero.
- Segundo.
- Final.

c) Si a = 6, b = 2, c = 3 y d = 4, ¿cuál es la forma en que se lleva a cabo la evaluación en orden primero y final?

d) Si los valores son los que se indican en el inciso (c), por medio de un árbol ilustrar la evaluación en orden segundo.

8.4 Considérense la expresión de cada uno de los incisos:

a) $\dfrac{\dfrac{a}{b - c} + dc}{c - \dfrac{d + a}{b}}$

b) $\dfrac{\dfrac{(a - bc)d}{e}}{\dfrac{(b + a)ce}{b} + f}$

c) $\dfrac{c + d}{a} - \dfrac{ca - bc}{\dfrac{e}{c + d}}$

d) $\dfrac{\dfrac{(e + d)a + ca - e}{b + f}}{\dfrac{a + b}{c} - \dfrac{adf}{e}}$

Determinar:

- El árbol binario que representa la expresión.
- El recorrido en orden primero, segundo y final.
- Si a = 8, b = 4, c = 2, d = 1, e = –2 y f = 6, ¿cuál es la forma en que se lleva a cabo la evaluación en orden primero y final en cada uno de los incisos?
- Si los valores son los que se indican anteriormente, por medio de un árbol ilustrar la evaluación en orden segundo.

8.5 Sea el recorrido en orden primero. +a–-*bcd/e+fg.

a) Construir el árbol binario.
b) ¿Cuál es el recorrido en orden: segundo y final?
c) ¿Cuál es la simulación de la evaluación, usando pilas para el recorrido primero y final?
d) ¿Cuál es la evaluación en orden segundo, ilustrándolo por medio de árboles? Con los valores donde sea necesario de a = 1, b = 2, c = 3, d = 5, e = 4, f = –3, g = 5.
e) ¿Cuál es la ecuación matemática que representa el árbol binario?

8.6 Sea el recorrido en orden:

a) Final: db*b–cab*/+a*d–
b) Final: ab+cd/ef–gh+*–*
c) Segundo: a+b–e/c–e+a/b*d
d) Segundo: a*d/f/g*h+e+b–c
e) Primero: /a+*+bc–ec/d–ba
f) Primero: *+/c*+abfde

En cada uno de los incisos realizar lo siguiente:

- Construir el árbol binario.
- Determinar el recorrido en orden segundo y final.
- Simular la evaluación usando pilas para el recorrido primero y final. Con los valores donde sea necesario de a = 3, b = 2, c = –1, d = 1, e = 4, f = 6, g = –3, h = 4.
- Ilustrar la evaluación en orden segundo, usando árboles.
- Determinar la ecuación que representa el árbol binario.

8.7 Con la siguiente información: (m, d, c, e, a, f, g, c, b, m, z, a):

a) Elaborar un árbol binario de búsqueda. Considerar que la información se coloca a la izquierda de un nodo determinado si es menor o igual al nodo y a la derecha exclusivamente si es mayor.

b) ¿Cuál es el recorrido en orden primero, segundo y final?

c) ¿Cómo queda el árbol si se dan de baja los nodos m y c, dando prioridad al nodo izquierdo sobre el derecho para ocupar el lugar del padre? Además dar de alta los nodos n, i y h.

d) ¿Cuál es el recorrido en orden segundo una vez que se han llevado a cabo los ajustes correspondientes?

8.8 Considérese la información siguiente:

a) 40, 12, –8, 0, 60, 35, 5, –1, 7, 23, 42, 70, 38

b) 0, 7, 17, –7, 4, –2, 60, 11, -5, 0, 8

c) 100, 3, 7, 5, –2, 60, 80, 115, –1, 0

d) 48, 32, 12, 27, 72, 85, 0, 1, 60, 70, 56, 49, 48

En relación con cada uno de los incisos:

- Elaborar un árbol binario de búsqueda. Considerar que la información se coloca a la izquierda de un nodo determinado si es menor o igual al nodo y a la derecha exclusivamente si es mayor.
- ¿Cuál es el recorrido en orden primero, segundo y final?
- ¿Cómo queda el árbol, si se hacen los siguientes movimientos en cada uno de los incisos? Concediendo prioridad al nodo izquierdo sobre el derecho para ocupar el lugar del padre cuando existe alguna baja.

a) Baja: 12; altas: 58, 3 y 6.

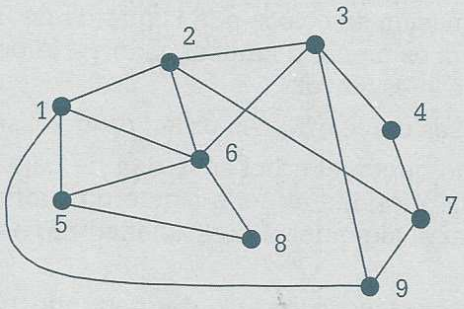
b) Bajas: 17 y –7; altas: 0, –2 y 18.

c) Bajas: –2 y 80; altas: 9 y 72.

d) Bajas: 48 y 12; altas: –9, 64 y 40.

- ¿Cuál es el recorrido en orden primero, después de haber hecho los ajustes?

8.9 Considérese el siguiente grafo:



Determinar el árbol generador, partiendo del nodo 1 y prioridad en orden ascendente. Por medio de búsqueda:

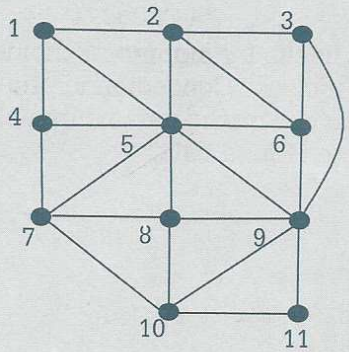
a) A lo ancho.

b) En profundidad.

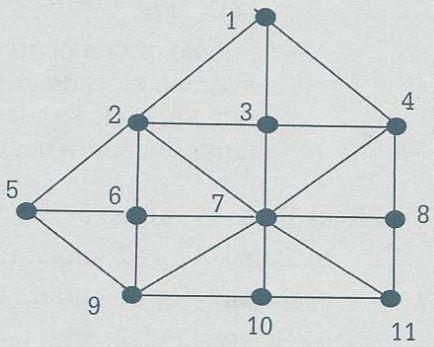
8.10 Determinar el árbol generador de los grafos de cada inciso, partiendo del nodo 1 y con prioridad de selección ascendente, usando para ello búsqueda.

- A lo ancho.
- En profundidad.

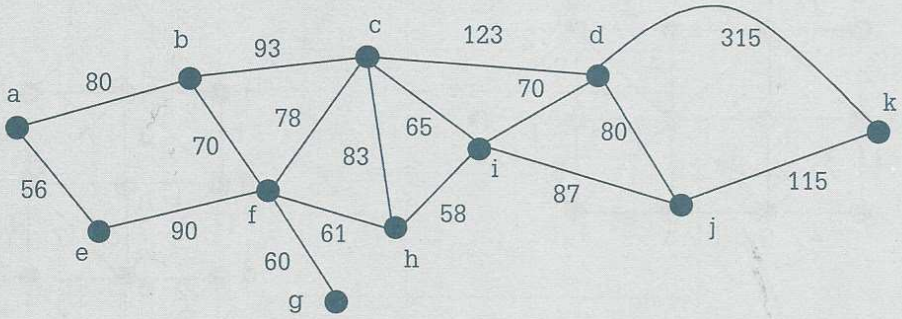
a)



b)



8.11 El siguiente grafo representa la red carretera entre las ciudades a, b, c, d, e, f, g, h, i, j y k, así como la longitud de cada una de las carreteras que unen las distintas ciudades:



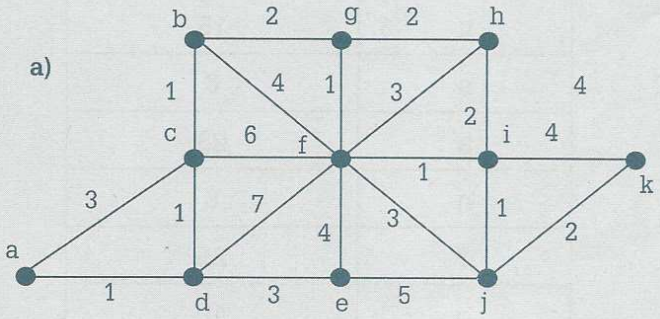
Determinar el árbol generador mínimo usando el método de:

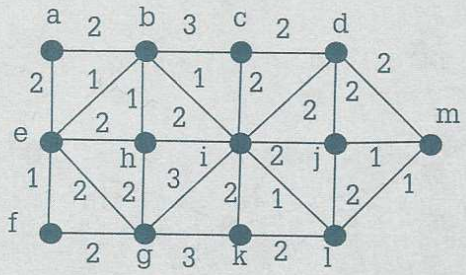
a) Prim.

b) Kruskal.

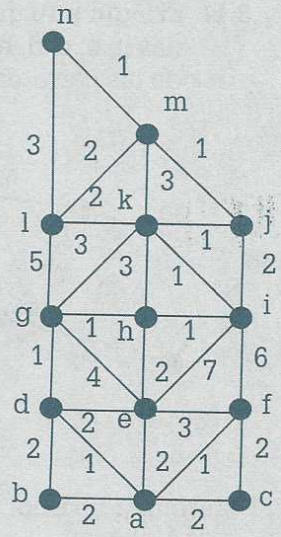
8.12 Determinar el árbol generador mínimo de los grafos de los incisos (a), (b) y (c) usando para ello los métodos de Prim y Kruskal. Si existen más árboles de expansión mínimos muestre adicionalmente uno para cada grafo.

a)





b)



c)

8.13 En un texto se encontró que la frecuencia de uso de cada uno de los caracteres es la siguiente.

Carácter	Peso o frecuencia
o	20
l	8
u	15
s	6
a	23
n	6
i	25
b	10
e	19

a) ¿Cuál es el árbol optimal para el código de Huffman?.

b) Con el árbol obtenido, codificar el mensaje: **solounabuenailusion**.

8.14 En un documento se encontró que la frecuencia con la que ocurren los caracteres es:

Carácter	Peso o frecuencia
a	80
e	72
f	55
g	28
h	19
i	78
l	33
m	43
o	69
s	36
u	48
espacio	47

a) ¿Cuál es el árbol optimal para el código de Huffman?

b) Con dicho árbol codificar el mensaje: **hola amigos**.