SEP SEIT DGIT

INSTITUTO TECNOLÓGICO DE NUEVO LAREDO

DEPTO. DE SISTEMAS Y COMPUTACIÓN


# "GRAFOS"

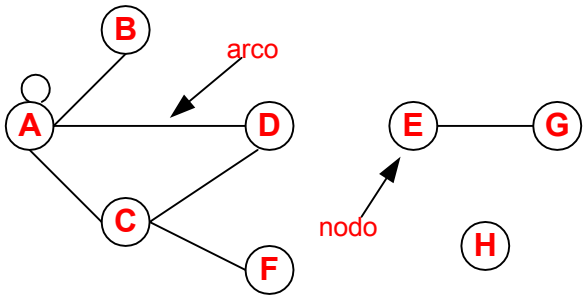
Estructura de Datos

Lipschutz, Seymour

***CAPÍTULO 8***

## GRAFOS

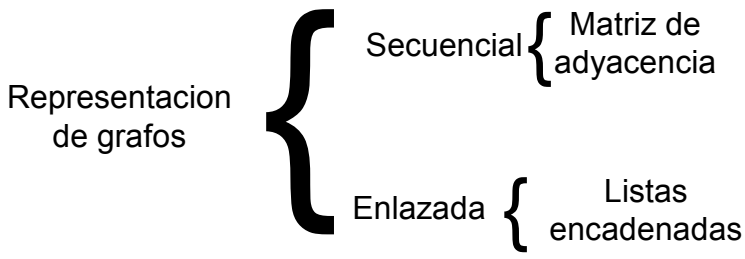
Un grafo está formado por un conjunto de nodos(o vértices) y un conjunto de arcos. Cada arco en un grafo se especifica por un par de nodos.
El conjunto de nodos es {A, B, C, D, F, G, H} y el conjunto de arcos {(A, B), (A, D), (A, C), (C, D), (C, F), (E, G), (A, A)} para el siguiente grafo



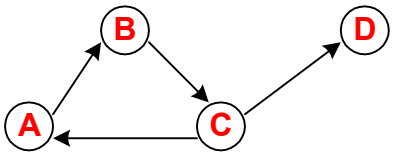

Si los pares de nodos en los arcos dirigidos, el grafo se denomina grafo directo, dirigido o dígrafo.

## TERMINOLOGÍA

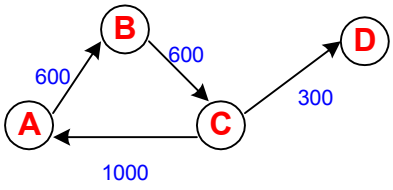
*.-Al número de nodos del grafo se le llama orden del grafo.
*.-Un grafo nulo es un grafo de orden 0 (cero).
*.-Dos nodos son adyacentes si hay un arco que los une.
*.-En un grafo dirigido, si A es adyacente de B, no necesariamente B es adyacente de A
*.-Camino es una secuencia de uno o mas arcos que conectan dos nodos.
*.-Un grafo se denomina conectado cuando existe siempre un camino que une dos nodos cualesquiera y desconectado en caso contrario.
*.-Un grafo es completo cuando cada nodo esta conectado con todos y cada uno de los nodos restantes.
*.-El camino de un nodo así mismo se llama ciclo.



## MATRIZ DE ADYACENCIA

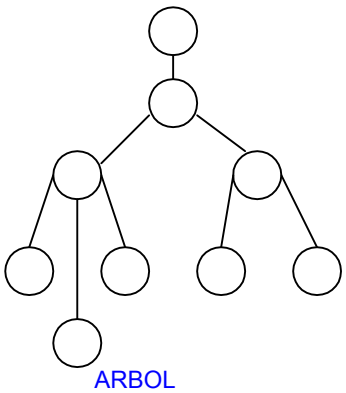


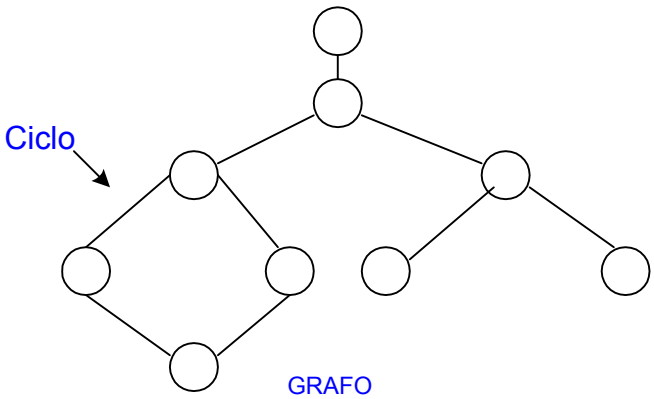

| | A | B | C | D |
|---|---|---|---|---|
| A | 0 | 1 | 0 | 0 |
| B | 0 | 0 | 1 | 0 |
| C | 1 | 0 | 0 | 1 |
| D | 0 | 0 | 0 | 0 |




| | A | B | C | D |
|---|---|---|---|---|
| A | 0 | 600 | 1000 | 0 |
| B | 600 | 0 | 600 | 0 |
| C | 1000 | 600 | 0 | 300 |
| D | 0 | 0 | 300 | 0 |

*.-Un grafo sin ciclos es un árbol.
*.-El entregado de un nodo indica el número de arcos que llegan a se nodo.
*.-El fuera de grado de un nodo indica el número de arcos que salen de él.
*.-Un grafo de N vértices o nodos es un árbol si cumple las siguientes condiciones
a) Tiene N-1 arcos
b) Existe una trayectoria entre cada par de nodos.
c) Esta mínimamente conectado.




*.-Un grafo esta etiquetado si sus arcos tiene valores asignados. Si este valor es numérico se dice que el grafo tiene peso.

## GRAFOS Y SUS APLICACIONES

## INTRODUCCIÓN

Este capítulo profundiza en otra estructura de datos no lineal: el grafo. Como hemos hecho con otras estructuras de datos. Discutiremos la representación de los grafos en memoria y presentaremos varias operaciones y algoritmos sobre ellos. En particular discutiremos la búsqueda en anchura y la búsqueda en profundidad para nuestros grafos. También se repasaran ciertas aplicaciones de los grafos, incluyendo la ordenación topológica.

## 8.2 TERMINOLOGÍA DE TEORÍA DE GRAFOS

Esta sección recoge alguna de la terminología principal asociada con la teoría de grafos por tanto advertimos al lector de que nuestras definiciones pueden ser ligeramente diferentes de las definiciones usadas en otros textos de estructuras de datos y teoría de grafos.

## GRAFOS Y MULTIGRAFOS

Un grafo G consiste en dos cosas:
(1) Un conjunto V de elementos llamados nodos (o puntos o *vértices*)
(2) Un conjunto E de aristas tales que cada arista *e* de E esta identificada por un único (desordenado) par [*u,v*] de nodos de V, denotado por *e*-[*v,u*].

A veces denotamos un grafo escribiendo G=(V,E)
Suponga que *e* =[*u,v*]. entonces los nodos *u* y *v* se llaman extremos de *e*, y *u* y *v* se dice *que son nodos adyacentes o vecinos.* El grado de un nodo *u*, escrito grad(*u*), es el número de artistas que contienen a *u*.si grad(*u*)= 0, o sea, si u no pertenece a ninguna arista---entonces se dice que u es un nodo aislado.
Un camino P de longitud *n* desde un nodo *u* se define como la secuencia de *n* +1 nodos.

$$P=(v_{0i}, v_{1i}, v_{2i}, \ldots v_m)$$

Tal que $u = v_{0i}$, $v_1$, es adyacente a $v_{i-1}$ para i=1,2,…*n* ; y $v_n = v$ . El camino P se dice que es cerrado si $v_0 = v_n$ .El camino P se dice que es simple si todos los nodos son distintos, a excepción de $v_0$ que puede ser igual a $v_n$ ; es decir , P es simple si los nodos $v_0, v_1 \ldots\ldots\ldots v_{n-1} \ldots$ son distintos y los nodos $v_1, v_2 \ldots\ldots\ldots v_n$ son distintos. Un ciclo es un camino simple cerrado de longitud 3 o mas . Un ciclo de longitud *k* se llama *k*-ciclo.

Un grafo G se dice que es conexo si existe un camino entre cualesquiera dos de sus nodos.

Mostraremos en el problema 8.18 que si existe un camino P desde un nodo *u* a un nodo *v*, entonces eliminando las aristas innecesarias, se puede obtener un camino simple Q de *u* a *v*, de acuerdo con esto podemos establecer la siguiente proposición.

Proposición 8.1.-Un grafo G es conexo si y sólo si existe un camino simple entre cualesquiera dos nodos de G.
Se dice que un grafo G es completo si cada nodo *u* de G es adyacente a todos los demás nodos de G claramente, un grafo así es conexo, un grafo completo de *n* nodos tendrá $n(n-1)2$ aristas.

Un grafo conexo T sin ciclos se llama grafo árbol o árbol libre o simplemente árbol. Esto significa en particular, que existe un único camino simple P entre cada dos nodos *u* y *e* de T (problema 8.18) mas aún, si T es un árbol de finito de *m* nodos, entonces T tendrá *m* -1 aristas (problema 8.20).

Un grafo G se dice que esta etiquetado si sus aristas tiene datos asignados. En particular, se dice que G tiene peso si cada arista e de G tiene asignado un valor numérico no negativo $w(e)$ llamado peso o longitud de *e.* En ese caso, a cada camino P de G se le asigna un peso o una longitud que es la suma de lo pesos de las aristas que constituyen el camino P. si no se da otra información sobre pesos, asumiremos que cada grafo G tiene peso pero de forma que los pesos $w(e)$ de cada arista *e* de G es igual a 1.
La definición de un grafo puede ser generalizada si permitimos lo siguiente:

(1) .- Aristas múltiples. Dos aristas e y e distintas se llama aristas múltiples si conectan los mismos extremos, o sea, si $e[u,v]$ y $e=[u.v]$.
(2) .- Bucles. Una arista e se llama bucle si tiene extremos idénticos, o sea, si $e=[u,v]$.

Tal generalización M se llama *multigrafo*. En otras palabras la definición de un grafo normalmente no permite ni aristas múltiples ni bucles.
Un multigrafo M se dice que es finito si tiene un numero finito de nodos y de aristas. Observe que un grafo G con un número finito de nodos debe automáticamente un número finito de aristas y por tanto debe ser finito; pero esto no es necesariamente cierto para un multigrafo M, ya que M puede tener múltiples aristas. A menos que se indique lo contrario, los grafos y multigrafos de este texto serán finitos.

EJEMPLO 8.1

(*a*) La figura 8-1(*a*) es un dibujo de un grafo conexo con cinco nodos—A,B,C,D y E— y seis aristas ;

*[A,B],[B,C], [C,D],[D,E], [A,E], [C,E], [A,C]*

Hay dos caminos simples de longitud 2 entre B y E ; (B, A, E)y (B, C, E). Hay un solo camino simple de longitud dos desde B hasta D:(B, C, D). Anotamos que (B, A, D) no es un camino, ya que [A, D] no es una arista . Hay dos 4-ciclos en el grafo:

[A, B, C, E, A] y [A, C, D, E, A]

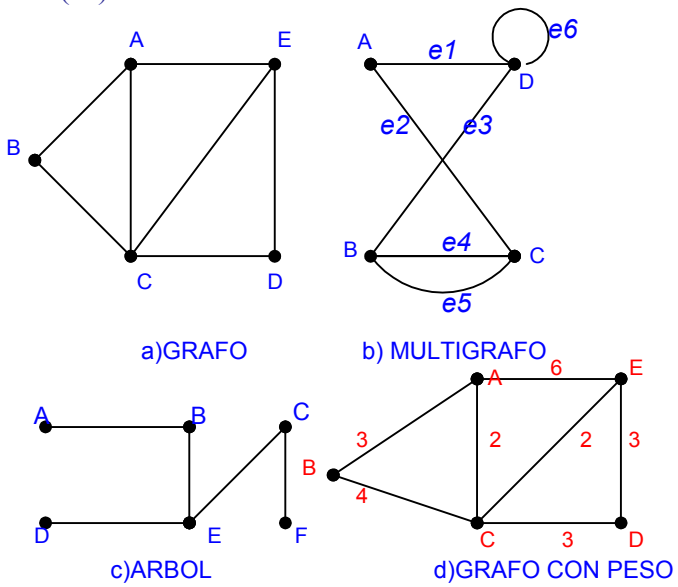
Note que grad(A)=3, ya que A pertenece a tres aristas. Similarmente, grad(C)=4 y

grad(D)=2.

(b) La figura 8.1 (b) no es un grafo sino un multigrafo. La razón es que tiene aristas múltiples ------*ei*=[B, C] y *e*5=[B, C]--- y tiene un bucle *e*6=[D, D]. La definición de un grafo normalmente no permite aristas múltiples ni bucles.

(c) La figura 8-1(c) es un grafo árbol con *m*=6 nodos y, consecuentemente, *m*-1=5 aristas. El lector puede verificar que hay un único camino simple entre cada dos nodos del grafo árbol.

(d) La figura 8-1(d) es el mismo grafo que la figura 8-1(a), excepto que ahora el grafo tiene peso Observe que $P_1$=(B, C, D) y $P_2$=(B, A, E, D) son ambos caminos del nodo B al nodo D. Sin embargo, aunque $P_2$ contiene mas aristas que $P_1$ el peso w($P_2$)=9 es menor que el peso *w*($P_1$)=10.




## Grafos dirigidos

Un grafo dirigido G, también llamado digrafo o grafo, es lo mismo que un multigrafo, solo que cada arista e de G tiene una dirección asignada o , en otras palabras ,cada arista *e* está identificada por un par ordenado *(u,v)* de nodos G en vez del par desordenado *[u.v]*.
Suponga que G es un grafo dirigido con una arista dirigida *e=(u,v)*.Entonces *e* también se llama arco . Más aún se usa la siguiente terminología:
(1) *e* empieza en *u* y termina en *v*
(2) *u* es el origen o punto inicial de *e* ,y *v* es el destino o punto terminal de *e* .
(3) *u* es un predecesor de *v* y *v* es un sucesor o vecino de *u*
(4) *u* es adyacente hacia *v* y *v* es adyacente a *u*

El grado de salida de un nodo *u* de G, escrito gradsal(*u*), es el número de aristas que empiezan en *u* similarmente , el grado de entrada *u*, escrito gradent(*u*), es el número de aristas que terminan en *u*. Un nodo *u* se llama fuente si tiene un grado de salida positivo y un grado de entrada nulo . Similarmente *u* se le llama sumidero si tiene un grado de salida nulo y un grado de entrada positivo.

Las nociones de camino , camino simple y ciclo se aplican en los grafos dirigidos igual que en los grafos no dirigidos excepto que la dirección de cada arista de un camino (ciclo) debe coincidir con la dirección del camino (ciclo) . Se dice que un nodo *v* es alcanzable desde un nodo *u* si existe un camino (dirigido) de *u* a *v* .

Un grafo dirigido G se dice que es conexo, o fuertemente conexo, si para cada par *u,v* de nodos de G existe un camino de *u* a *v* y también un camino de *v* a *u* . Por otro lado, G se dice que es unilateralmente conexo para cada par *u,v* de nodos de G hay un camino de *u* a *v* o un camino de *v* a *u*.

EJEMPLO 8.2

La figura 8.2 muestra un grafo dirigido G con cuatro nodos y siete aristas (dirigidas).Las aristas $e_2$ y $e_3$ se dice que son paralelas , ya que las dos empiezan en B y terminan en A .La arista *e*-es un bucle , ya que empieza y termina en el mismo nodo B. La secuencia $P_1$=(D, C, B, A), no es un camino ya que (C, B) no es una arista o sea , la dirección de la arista $e_5$=(B, C) no concuerda con la dirección del camino $P_1$. Pos otro lado $P_2$=(D, B, A) es un camino de D a A , ya que (D, B) y (B, A) son aristas .Así A es alcanzable desde D . No existe camino entre C y el resto de los nodos, así que G no es fuertemente conexo.

Sin embargo G es unilateralmente conexo. Observe que gradent(D)=1 y gradsal(D)=2. El nodo C es un sumidero, ya que gradent(C)=2 y gradsal(C)=0. Ningún nodo G es una fuente.

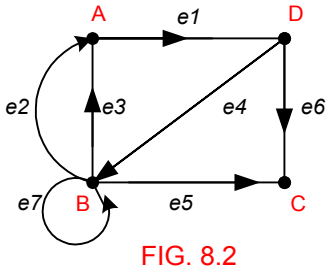


FIG. 8.2

Sea T un grafo árbol no vacío. Suponga que elegimos un nodo R de T. Así T con ese nodo R designado se llama árbol con raíz y R se llama su raíz. Recuerde que existe un único camino simple desde la raíz R hasta cualquier otro nodo de T. Esto define unja dirección para las aristas de T, así que el árbol con raíz puede ser visto como un grafo dirigido. Mas aún suponga que también ordenamos los sucesores de cada nodo *v* de T. Entonces T se llama árbol con raíz ordenado. Los árboles con raíz ordenados no son otra cosa que los árboles generales.

Un grafo dirigido G se dice que es simple si no tiene aristas paralelas. Un grafo simple G puede tener bucles, pero no puede tener mas de un bucle en un nodo dado. Un grafo G no dirigido puede verse como un grafo dirigido simple asumiendo que cada arista *[u,v]* de G representa dos aristas dirigidas *(u,v)* y *(v.u)*. (Observe que usamos la notación *(u,v)* para denotar un par ordenado).

Advertencia: El asunto principal de este capítulo son los grafos dirigidos simples. Por tanto, a menos que se diga o se deduzca lo contrario, el término grafo significará grafo dirigido simple y el término arista significará arista dirigida.

## 8.3 .-REPRESENTACION SECUENCIAL DE GRAFOS; MATRIZ DE ADYACENCIA MATRIZ DE CAMINOS

Existen dos formas estándar de mantener un grafo G en la memoria de una computadora. Una forma llamada presentación secuencial de G, se basa en la matriz de adyacencia de A. La otra forma, llamada representación enlazada de G, se basa en las listas enlazadas de vecinos. Esta sección cubre la primera representación y muestra como se puede usar la matriz de adyacencia A de G para responder fácilmente ciertas cuestiones sobre conectividad de G. La representación enlazada de G se vera en la sección 8.5.
Independientemente de la forma que se mantenga el grafo G en la memoria de la computadora, el grafo G normalmente se introduce en la computadora por su definición formal: un conjunto de nodos y un conjunto de aristas.

### MATRIZ DE ADYACENCIA

Suponga que G es un grafo dirigido simple de *m* nodos y suponga que los nodos de G han sido ordenados y llamados $v_1, v_2 \ldots v_m$ .Así la matriz de adyacencia A=($a_{i,j}$) del grafo G es la matriz de *m x m* elementos definida como sigue:

$$a_{i,j} = \begin{cases} 1 & \text{si } v_i \text{ es adyacente a } v_j, \text{ o sea, si hay una arista}(v_i, v_j) \\ 0 & \text{en caso contrario} \end{cases}$$

Una matriz A así, que contiene entradas de 0y1 , se llama matriz de bits o matriz booleana. La matriz de adyacencia de A del grafo G depende de la ordenación de los nodos de G; esto es, diferentes ordenaciones de los nodos pueden resultar en diferentes matrices de adyacencia. Sin embargo las matrices obtenidas por dos ordenaciones diferentes están fuertemente relacionadas en cuanto que a una puede ser obtenida de la otra simplemente cambiando filas y columnas. A menos que se indique lo contrario asumiremos que los nodos de nuestro grafo G tienen un orden fijo.

Suponga que G es un grafo no dirigido. Entonces la matriz de adyacencia de G, A será una matriz simétrica , o sea , con $a(i,j)= a(j,i)$ para cada $i$ y $j$ . Esto viene del hecho de que cada arista no dirigida $[u,v]$ corresponde a las dos aristas dirigidas $(u,v),(v,u)$.
La anterior representación matricial de un grafo se puede extender a multigrafos. Específicamente, si G es un multigrafo, entonces la matriz de adyacencia de G es la matriz A de m x m= a(i,j) definida haciendo $a(i,j)$ igual al numero de aristas desde $v_i$ hasta $v_j$.

### Ejemplo 8.3

Considere el grafo G de la figura 8.3. Suponga que los nodos se guardan en memoria en un array lineal Datos tal como sigue;

DATOS: X, Y, Z, W

Asumimos que el orden de los nodos de G es el siguiente $v_1$=X, $v_2$=Y, $v_3$=Z y $v_a$=W. La matriz de adyacencia de A de G es la siguiente:

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

Observe que el número 1 en A es igual al número de aristas en G .

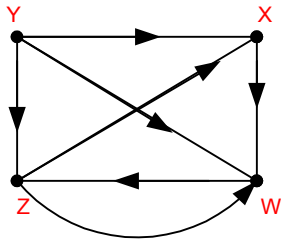



FIG. 8.3

Considere las potencias A,$A_{(1)}$,$A_{(2)}$, $A_{(3)}$,… de la matriz de adyacencia de A de un grafo G. Sea $a_1(i,j)$=la entrada $i\ j$ de la matriz $A_{(M)}$ potencia.

Observe que $a_1(i,j)=a(i,j)$ da el numero de caminos de longitud 1 desde el nodo $v_1$ hasta el nodo $v_3$ . Se puede demostrar que $a_2(i,j)$ da el numero de caminos de longitud 2 desde $v_1$,hasta $v_3$. De hecho, en el problema 8.19 probaremos el siguiente resultado general.

Proposición 8.2 Sea A la matriz de adyacencia de un grafo G de la figura 8.3. Entonces $a_k(i,j)$, la entrada $ij$ en la matriz , $a_k$ da el número de caminos de longitud $k$ desde $v_1$ hasta $v_3$.

Considere el nuevo grafo G de la figura 8.3 cuya matriz de adyacencia A se da en el ejemplo 8.3. Las potencias $A_{(2)}$,$A_{(3)}$, $A_{(4)}$,… de la matriz A son las siguientes.

$$A^2=\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \quad A^3=\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 2 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad A^4=\begin{pmatrix} 0 & 0 & 1 & 1 \\ 2 & 0 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

Así en particular, hay un camino de longitud 2 de $v_4$ a $v_1$, hay dos caminos de longitud 3 de $v_2$ a $v_3$ y hay 3 caminos de longitud 4 de $v_2$ a $v_4$ (Aquí ,$v_1$=X, $v_2$=Y, $v_3$=Z y $v_4$=W) Suponga que ahora definimos la matriz B como sigue:

$$B_r = A+A^2+A^3+....+A^r$$

Entonces la entrada $ij$ de la matriz B, da el número de caminos de longitud $r$ o menor, del nodo $vi$ al $vj$.

## MATRIZ DE CAMINOS

Sea G un grafo dirigido simple con $m$ nodos $v_1, v_2 \ldots v_m$. La matriz de caminos o matriz de alcance de G es la matriz $m$-cuadrada P= $v(i,j)$ definida como sigue:

$$P = \begin{cases} 1 \text{ Si hay un camino desde } \boldsymbol{vi} \text{ hasta } \boldsymbol{vj} \\ 0 \text{ en otro caso} \end{cases}$$

Suponga que hay un camino desde $v$i hasta $v$j. Entonces tiene que haber un camino simple desde $vi$ hasta $vj$ cuando $vi=vj$, o un ciclo de $vi$ a $vj$ cuando $vi=vj$. Como G solo tiene $m$ nodos $n$ un camino simple así ha de tener longitud m-1 o menor, o un ciclo así ha de tener longitud $m$ o menor . Esto significa que existe una entrada $ij$ no nula de la matriz B$m$ definida de la anterior subsección . De acuerdo con esto, tenemos la siguiente relación entre la matriz de caminos P y la matriz de adyacencia A.

Proposición 8.3. Sea A la matriz de adyacencia y P= p$i,j$, la matriz de caminos de un grafo G , Entonces p$i,j$ =1 si y solo si hay un número positivo en la entrada $ij$ de la matriz

$$B_m = A + A^2 + A^3 + \ldots + A^m$$

Considere el grafo G con $m$=4 nodos de la figura 8.3 .Sumando las matrices con las potencias A,A(2),A(3) y A(4) obtenemos la siguiente matriz $B_4$ , reemplazando las entradas positivas por 1, obtenemos la matriz de caminos P del grafo G .

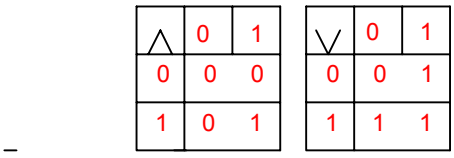
$$B_4 = \begin{pmatrix} 1 & 0 & 2 & 3 \\ 5 & 0 & 6 & 8 \\ 3 & 0 & 3 & 5 \\ 2 & 0 & 3 & 3 \end{pmatrix} \quad Y \quad P = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

Examinando la matriz P, vemos que el nodo $v_2$ no es alcanzable por ninguno de los otros nodos. Recuerde que se dice que un dirigido G es fuertemente conexo si, para cada par de nodos $u$ y $v$ de G, existe tanto un camino de $u$ a $v$ como de $v$ a $u$. Por tanto, G es fuertemente conexo si y solo si la matriz de caminos P de G no tiene entradas nulas. Así el grafo G de la figura 8.3 no es fuertemente conexo.

El cierre transitivo de un grafo G se define como el grafo G' tal que G' tiene los mismos nodos que G y existe una arista ($vi,vj$) en G siempre que existe un camino de $vi$ a $vj$ en G. Así la matriz de caminos P del grafo G es precisamente la matriz de adyacencia de su cierre transitivo G`. Mas aún un grafo G es fuertemente conexo si y solo si su cierre transitivo es un grafo completo.

Nota: La matriz de adyacencia A y la matriz de caminos P de un grafo G se pueden ver como matrices lógicas (booleanas), donde el 0 representa Falso y el 1 representa cierto. Así como las operaciones lógicas de $\wedge$(Y) y O(V) se pueden aplicar a las entradas de A y

P los valores de Y y O aparecen en la figura 8.4. Las operaciones se usarán en la siguiente sección .




(a).Y (b).O

FIG 8.4

## 8.4 ALGORITMO DE WARSHALL ; CAMINOS MÍNIMOS

Sea G un grafo dirigido con m nodos, $v_1$ ,$v_2$…..$v_m$. Suponga que queremos encontrar la matriz de caminos P para el grafo G. Warshall dio un algoritmo para este propósito que es mucho mas eficiente que calcular las potencias de la matriz de adyacencia A y aplicar la proposición 8.3. Este algoritmo se describe en esta sección y un algoritmo similar se usa para calcular el camino mínimo de G cuando G tiene peso.
Primero definimos las matrices m-cuadradas booleanas $P_0$, $P_1$,….$P_m$ como sigue. Sea $P_k[i, j]$ la entrada $i,j$ de la matriz $P_k$. Así definimos

$$P_k[i,j] = \begin{cases} 1 & \text{Si existe un camino simple de } v_i \text{ a } v_j \\ & \text{que no usa otros nodos aparte de posiblemente} \\ & v_1, v_2, \ldots v_k \\ 0 & \text{en otro caso} \end{cases}$$

En otras palabras,

$P_0[i,j]$ =1 si hay una arista de $v_i$ a $v_j$
$P_1[i, j]$ = 1 si hay un camino simple de $v_i$ a $v_j$ que no usa otros nodos excepto posiblemente $v_i$.
$P_2[i, j]$ = 1 si hay un camino simple de $v_i$ a $v_j$ que no usa otros nodos excepto posiblemente $v_i$ y $v_j$.

En primer lugar observe que $P_0$ =A , la matriz de adyacencia de G. Mas aún, como G solo tiene $m$ nodos, la última matriz $P_m$ =P, la matriz de caminos de G .
Warshall observó que $P_k[i. j]$ = 1 solo puede ocurrir si se da uno de los dos siguientes casos:

(1).- Existe un camino simple de $v_i$ a $v_j$ que no usa otros nodos excepto posiblemente $v_1$, $v_2$,..$v_{k-1}$ ; por tanto,

$$P_{k-1}[i, j] = 1$$

(2).- Existe un camino simple de $v_i$ a $v_k$ y otro camino simple de $v_k$ a $v_j$ que no usa otros nodos excepto posiblemente $v_1$,$v_2$,..$v_{k-1}$ ; por tanto,

$$P_{k-1}[i, k] = 1 \quad \text{y} \quad P_{k-1}[k, j] = 1$$

Estos dos casos se representan, respectivamente, en las figuras 8.5(a) y (b), donde


Fig 8.5

Así, los elementos de la matriz $P_k$ se pueden obtener por :

$$P_k[i, j] = P_{k-1}[i, j] \quad V \quad (P_{k-1}[i, k] \wedge P_{k-1}[k, j])$$

Donde usamos las operaciones lógicas $\wedge$ (Y) y V (O). En otras palabras podemos obtener cada entrada de la matriz $P_k$ mirando solo tres entradas de la matriz $P_{k-1}$. El algoritmo de Warshall va a continuación.

Algoritmo 8.1 (Algoritmo de Warshall ). Un grafo dirigido G con M nodos esta en memoria por su raíz adyacente A. Este algoritmo encuentra la matriz de caminos (booleana) P del grafo G.

1.-[Inicializar P]. Repetir para I, J = 1, 2,…,M :
Si A[I, J] = 0 , entonces : Hacer P[I, J] := 0;
Si no : Hacer P[I, J] : = 1.
[Fin del Bucle]
2.-[Actualizar P] .Repetir pasos 3 y 4 para K = 1, 2,…,M:
3.- Repetir paso 4 para I = 1, 2,…,M:
4.- Repetir para J = 1, 2,…,M:
Hacer P [I, J] : = P [I, J] V (P[I, K] $\wedge$P[K, J].
[Fin del Bucle].
[Fin del Bucle del paso 3]
[Fin del Bucle del paso 2].
5.- Salir.

## ALGORITMO DEL CAMINO MÍNIMO

Sea G un grafo dirigido con $m$ nodos $v_1, v_2,\ldots v_m$. Suponga que G tiene peso; o sea, suponga que cada arista $e$ de G tiene asignado un número no negativo w($e$) llamado peso o longitud de la arista $e$. Entonces G se pude mantener en memoria por su raíz de pesos W= (w $i,j$ ), definida como sigue;

$$W\,i, j \begin{cases} w(e) & \text{Si hay uan arista e de } v\text{i a } v\text{j} \\ 0 & \text{Si no hay arista de } v\text{i a } v\text{j} \end{cases}$$

La matriz de caminos P nos dice si hay o no caminos entre los nodos. Ahora queremos encontrar una matriz Q = (q $i,j$) que nos diga las longitudes de los caminos entre los nodos o, mas exactamente una matriz Q($i,j$) donde:

$$q\,i,j = \text{A la longitud del camino mínimo de } v\text{i a } v\text{j}$$

Ahora describiremos una modificación del algoritmo de Warshall que encuentra la matriz Q. Definimos una secuencia de matrices $Q_0, Q_1,\ldots Q_n$ (análogas a las anteriores matrices $P_0, P_1,\ldots,P_m$ ) cuyas entradas vienen dadas por:

$Q_k[i, j]$ = La menor de las longitudes de los anteriores caminos de $v_i$ a $v_j$ o la
Suma de las longitudes de los anteriores caminos de $v_i$ a $v_k$ y de $v_k$ a $v_j$.

Más exactamente:

$$Q_0[i, j] = \text{MIN}\,(Q_{k-1}\,[i, j]\,,\; Q_{k-1}\,[i, k] + Q_{k-1}\,[k, j])$$

La matriz inicial $Q_0$ es la misma de la matriz de pesos W excepto que cada 0 de W se reemplaza por $\infty$ (o un número muy, muy grande). La matriz final $Q_m$ será la matriz Q deseada.

Ejemplo 8.4

Considera el grafo con peso G de la figura 8.6 . Asumimos que $v_i$ = R, $v_2$=S, $v_3$=T, $v_4$=U. Así la matriz de pesos W de G es la siguiente:

$$W=\begin{pmatrix} 7 & 5 & 0 & 0 \\ 7 & 0 & 0 & 2 \\ 0 & 3 & 0 & 0 \\ 4 & 0 & 1 & 0 \end{pmatrix}$$

Aplicando el algoritmo modificado de Warshall, obtenemos las siguientes matrices: $Q_0$, $Q_1$, $Q_2$, $Q_3$, $Q_4 = Q$. A la derecha de cada matriz $Q_k$ se muestra la matriz de caminos que corresponde a las longitudes de la matriz $Q_k$.

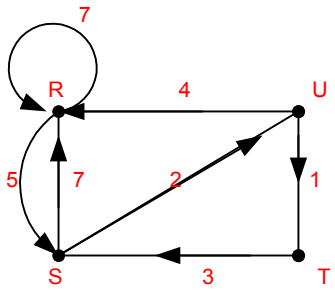



FIG. 8.6

$$Q_0 = \begin{pmatrix} 7 & 5 & \infty & \infty \\ 7 & \infty & \infty & 2 \\ \infty & 3 & \infty & \infty \\ 4 & \infty & 1 & \infty \end{pmatrix} \qquad \begin{pmatrix} RR & RS & --- & --- \\ SR & --- & --- & SU \\ --- & TS & --- & --- \\ UR & --- & UT & --- \end{pmatrix}$$

$$Q_1 = \begin{pmatrix} 7 & 5 & \infty & \infty \\ 7 & 12 & \infty & 2 \\ \infty & 3 & \infty & \infty \\ 4 & 9 & 1 & \infty \end{pmatrix} \qquad \begin{pmatrix} RR & RS & --- & --- \\ SR & SRS & --- & SU \\ --- & TS & --- & --- \\ UR & URS & UT & --- \end{pmatrix}$$

$$Q_2 = \begin{pmatrix} 7 & 5 & \infty & \infty \\ 7 & 12 & \infty & 2 \\ 10 & 3 & \infty & 5 \\ 4 & 9 & 1 & 11 \end{pmatrix} \qquad \begin{pmatrix} RR & RS & --- & RSU \\ SR & SRS & --- & SU \\ TSR & TS & --- & TSU \\ UR & URS & UT & URS \end{pmatrix}$$

$$Q_3 = \begin{pmatrix} 7 & 5 & \infty & 7 \\ 7 & 12 & \infty & 2 \\ 10 & 3 & \infty & 5 \\ 4 & 4 & 1 & 6 \end{pmatrix} \qquad \begin{pmatrix} RR & RS & --- & RSU \\ SR & SRS & --- & SU \\ TSR & TS & --- & TSU \\ UR & URS & UT & URS \end{pmatrix}$$

$$Q_4 = \begin{pmatrix} 7 & 5 & 8 & 7 \\ 7 & 11 & 3 & 2 \\ 9 & 3 & 6 & 5 \\ 4 & 4 & 1 & 6 \end{pmatrix} \qquad \begin{pmatrix} RR & RS & RSUT & RSU \\ SR & SRS & SUT & SU \\ TSR & TS & TSUT & TSU \\ UR & URS & UT & URS \end{pmatrix}$$

Indicamos como se obtienen las entradas rodeadas por círculos:

$Q_1[4, 2] = MIN (Q_0[4, 2], Q_0[4, 1] + Q_0 [1, 2] ) = MIN (\infty, 4+5) = 9$

$Q_2[1, 3] = MIN (Q_1[1, 3], Q_1[1, 2] + Q_1 [2, 3] ) = MIN (\infty, 5+\infty) = \infty$

$Q_3[4, 2] = MIN (Q_2[4, 2], Q_2[4, 3] + Q_2 [3, 2] ) = MIN (9, 3+1) = 4$

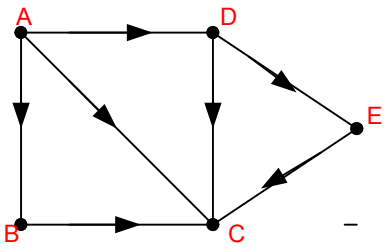
$Q_4[3, 1] = MIN (Q_3[3, 1], Q_3[3, 4] + Q_3 [4, 1] ) = MIN (10, 5+4) = 9$

A continuación se da la definición formal del algoritmo.

Algoritmo 8.2 (Algoritmo del camino mínimo). Un grafo con peso G de M nodos esta en memoria mediante su matriz de pesos W. Este algoritmo encuentra la matriz Q tal que [I, J] es la longitud del camino mínimo del nodo *v*i al nodo *v*j INFINITO es un numero muy grande y MIN es la función del valor mínimo.

1.- [Inicializar Q] .Repetir para I, J = 1, 2,…,M:
Si W [I, J] = 0, entonces: Hacer Q[I, J] : = INFINITO;
Si no: Hacer Q[I, J[: = W[I, J].
[Fin del Bucle].
2.-[Actualizar Q]. Repetir paso 3 y 4 para K = 1, 2,..,M:
3.- Repetir paso 4 para I = 1, 2,…,M:
4.- Repetir para J = 1, 2,…,M:
Hacer Q[I, J] : = MIN (Q[I, J], Q[I, k] + Q[k, J].
[Fin del Bucle].
[Fin del Bucle del paso 3].
[Fin del Bucle del paso 2].

5.-Salir.




| Nodo | Lista de adyacencia |
|---|---|
| A | B, C, D |
| B | C |
| C | |
| D | C, E |
| E | C |

(A) Grafo G (B) Listas de adyacencia

Fig 8.7

Observe el parecido entre los algoritmos 8.1 y 8.2.

El algoritmo 8.2 también se puede usar para un grafo G sin pesos, simplemente asignando el peso w(*e*) = 1 a cada arista *e* de G .

## 8.5 REPRESENTACIÓN ENLAZADA DE UN GRAFO

Sea G un grafo dirigido con m nodos. La representación secuencial de G en memoria--- o sea, la representación de G por su matriz de adyacencia A – tiene unas cuantas desventajas importantes. En primer lugar, puede ser difícil insertar y eliminar nodos de G.

Esto es porque el tamaño de A debería de ser cambiado y los nodos deberían de ser reordenados, así que habría muchos cambios en la matriz A. Más aún si el número de aristas es O(*m)* o O(*m* log *i m*), entonces la matriz A estará desperdiciada (contendrá muchos ceros) por tanto , se desperdiciara una gran cantidad de espacio. Por tanto G normalmente se representa en memoria por una representación enlazada, también llamada estructuras de adyacencia, la cual se describe en esta sección.

Considere el grafo G de la figura 8.7 (a) La tabla de la figura 8.7 (b) muestra cada nodo de G seguido por su lista de nodos adyacentes, también llamados sucesores o vecinos. La figura 8.8 muestra el diagrama esquemático de la representación enlazada de G en memoria. Específicamente la representación enlazada contendrá dos listas (o archivos) una lista de nodos NODO y una lista de aristas ARISTA, tal como sigue;

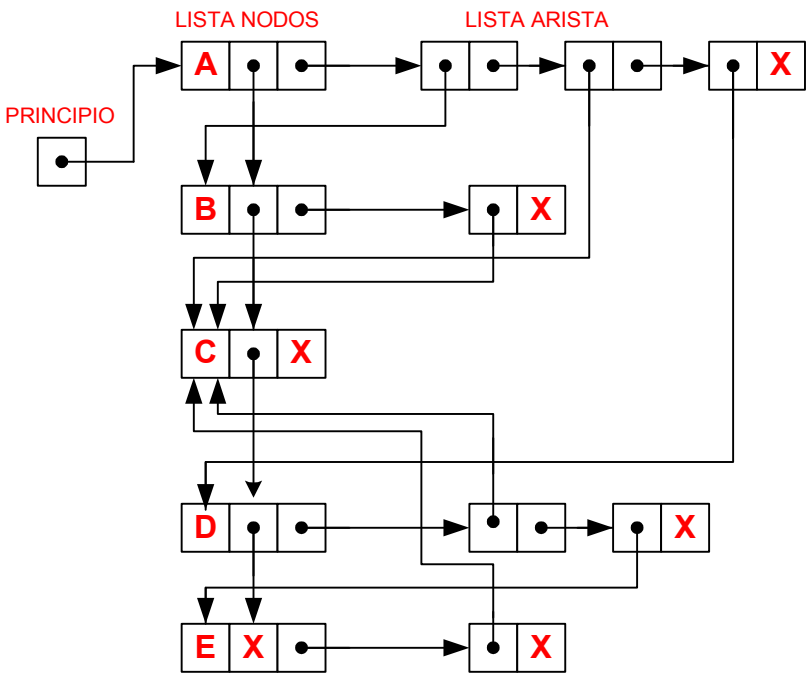



FIG 8.8

(a) Lista de nodos. Cada elemento de la lista de NODO corresponderá a un nodo de G y será un registro de la forma:

| NODO | SIG | ADY | |
|---|---|---|---|

Aquí NODO será el nombre o valor clave del nodo, SIG será un puntero al siguiente NODO de la lista NODO y ADY será un puntero al primer elemento de la lista de adyacencia del nodo que se mantiene en la lista ARISTA. El área sombreada indica que puede haber otra información en el registro, tal como el grado de entrada GRADENT del nodo, el grado de salida GRADSAL del nodo, el ESTADO del nodo durante la ejecución de un algoritmo, etc. (Alternativamente se puede asumir que NODO es un array de registros conteniendo campos como NOMBRE, GRADENT, GRADSAL, ESTADO). Los nodos mismos, como se ve en la figura 8.7 estarán organizados como una lista enlazada y por tanto habrá una variable puntero PRINCIPIO para el comienzo de la lista y una variable puntero NDISP para la lista de espacio disponible. A veces, dependiendo de la aplicación, los nodos pueden estar organizados como un array ordenado o un árbol binario de búsqueda en vez de una lista enlazada.

(b) Listas de aristas Cada elemento de la lista ARISTA corresponderá a una arista de G y será un registro de la forma;

| DEST | ENL | |
|---|---|---|

El campo DEST apuntara a la posición de la lista NODO del nodo destino o terminal de la arista. El campo ENL enlazara juntas las aristas con el mismo nodo inicial, o sea, los nodos de la misma lista de adyacencia. El área sombreada indica que puede haber otra información en el registro correspondiente a la arista, tal como un campo ARIS conteniendo los datos etiquetados de la arista cuando G es un grafo con etiquetas, un campo PESO conteniendo el peso de la arista cuando G es un grafo con peso, etc.

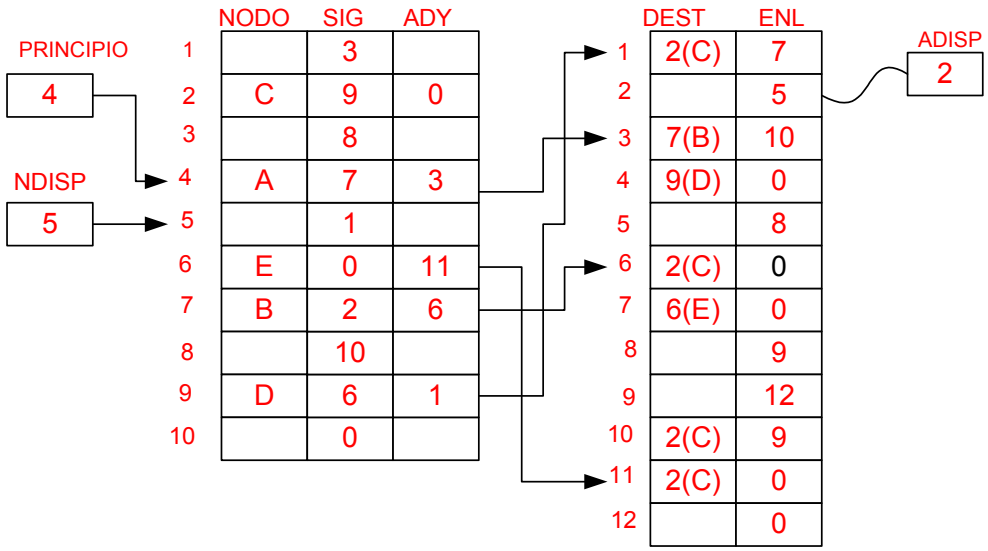



FIG 8.9

También necesitamos una variable puntero ADISP para la lista de espacio disponible en la ARISTA.
La figura 8.9 muestra como el grafo G de la figura 8.7(a) aparecería en memoria. La elección de 10 posiciones para la lista NODO y de 12 posiciones para la lista ARISTA es arbitraria. La representación enlazada de un grafo G que hemos estado viendo se puede denotar por:

GRAFO (NODO, SIG, ADY, PRINCIPIO, NDISP, ENL, ADISP)

La representación puede incluir un array PESO cuando G tiene peso o puede incluir un array ARIS cuando G es un grafo con etiquetas.

Ejemplo 8.5

Suponga que las líneas aéreas amistosas tienen nueve vuelos diarios:

| | | |
|---|---|---|
| 103 Atlanta Houston | 203 Boston a Denver | 305 Chicago a Miami |
| 106 Houston a Atlanta | 204 Denver a Boston | 308 Miami a Boston |
| 201 Boston a Chicago | 301 Denver a Reno | 402 Reno a Chicago |

Claramente, los datos pueden guardarse eficientemente en un archivo en el que cada registro contenga tres campos:

Número de vuelo Ciudad de origen Ciudad destino

Sin embargo, tal representación no puede responder fácilmente a preguntas elementales:

(a).-¿Hay un vuelo directo entre la ciudad X y la ciudad Y?
(b).-¿Se puede volar; con posibles escalas, de la ciudad X a la ciudad Y?
(c).-¿Cuál es la ruta mas directa , o sea, con el menor número de escalas, desde la ciudad X a la ciudad Y?

Para hacer que las repuestas a esas preguntas sean más fáciles de encontrar, puede ser muy útil que los datos sean organizados también como un grafo G con las ciudades como nodos y los vuelos como aristas. La figura 8.10 es un dibujo del grafo G.

La figura 8.11 muestra como el grafo G aparece en memoria usando la representación enlazada. Note que G es un grafo con etiquetas, no con peso, ya que el número de vuelo está simplemente como identificación. Incluso aunque los datos están organizados como un grafo, aun se requerirá algún tipo de algoritmo para contestar a las preguntas (b) y (c). Tales algoritmos se discuten mas adelante en este capítulo.

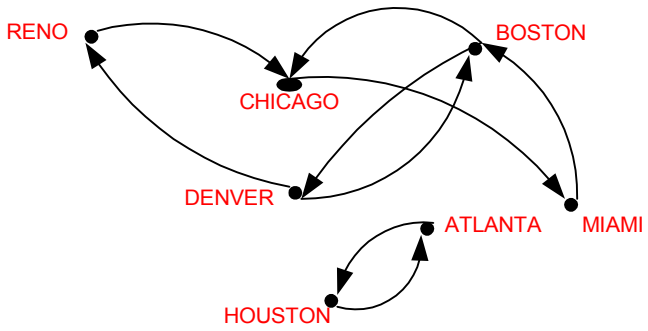



FIG 8.10

LISTA NODO

| | CIUDAD | SIG | ADY |
|---|---|---|---|
| 1 | | 0 | |
| 2 | ATALNTA | 12 | 1 |
| 3 | CHICAGO | 11 | 7 |
| 4 | HOUSTON | 7 | 2 |
| 5 | | 6 | |
| 6 | | 8 | |
| 7 | MIAMI | 10 | 8 |
| 8 | | 9 | |
| 9 | | 1 | |
| 10 | RENO | 0 | 9 |
| 11 | DENVER | 4 | 5 |
| 12 | BOSTON | 3 | 3 |

PRINCIPIO = 2, NDISP = 5

LISTA ARISTA

| | NUMERO | ORIG | DEST | ENL |
|---|---|---|---|---|
| 1 | 103 | 2 | 4 | 0 |
| 2 | 106 | 4 | 2 | 0 |
| 3 | 201 | 12 | 3 | 4 |
| 4 | 203 | 12 | 11 | 0 |
| 5 | 204 | 11 | 12 | 6 |
| 6 | 301 | 11 | 10 | 0 |
| 7 | 305 | 3 | 7 | 0 |
| 8 | 308 | 7 | 12 | 0 |
| 9 | 402 | 10 | 3 | 0 |
| 10 | | | | 11 |
| 11 | | | | 12 |
| 12 | | | | 0 |

ADISP = 10

FIG 8.11

## 8.6 OPERACIONES SOBRE GRAFOS

Suponga que un grafo G se mantiene en memoria mediante la representación enlazada

GRAFO (NODO, SIG, ADY, PRINCIPIO, NDISP, ENL, ADISP)

Discutida en la sección anterior. Esta sección discute las operaciones, de búsqueda inserción y eliminación de nodos y aristas en el grafo G. La operación de recorrido se trata en la siguiente sección.

Procedimiento 8.3 Busca(INFO, ENL, PRINCIPIO, ITEM, POS) [Algoritmo 5.2]
Busca la posición POS del primer nodo que contiene a ITEM, o hace POS : = NULO.

1.-Hacer PTR : =PRINCIPIO.
2.-Repetir mientras PTR sea diferente de NULO:
Si ITEM = INFO[PTR], entonces; Hacer POS := PTR y Volver.
Si no hacer PTR : = ENL [PTR].
[Fin del Bucle].
3.-Hacer POS : = NULO y volver.

Diferente que en el capítulo 5. Por supuesto, si se usan listas enlazadas circulares a árboles binarios de búsqueda en vez de listas enlazadas, se deben de usar los procedimientos análogos.

El procedimiento 8.3 (originalmente procedimiento 5.2) busca loa posición POS de un ITEM en una lista enlazada .

El procedimiento 8.4 (originalmente procedimiento 5.9 y algoritmo 5.10) elimina un ITEM dado de una lista enlazada. Se usa una variable lógica INDIC para decir si ITEM aparece originalmente en la lista enlazada o no.

## BÚSQUEDA EN UN GRAFO

Suponga que queremos encontrar la posición POS de un nodo N de un grafo C . Esto se puede llevar a cabo mediante el Procedimiento 8.3, tal como sigue:

Llamar BUSCA(NODO, SIG, PRINCIPIO, N, POS)

O sea, que esta sentencia de llamada busca en la lista NODO el nodo N.

Procedimiento 8.4 ELIMINAR(INFO, ENL, PRINCIPIO, DISP, ITEM, INDIC)
[Algoritmo 5.10]
Elimina el primer nodo de la lista que contenga a ITEM, o hace INDIC:=FALSO si ITEM no aparece en la lista.
1.-[¿Lista vacia?] Si PRINCIPIO = NULO, entonces: Hacer
INDIC : = FALSO y volver .
2.-[¿ITEM en primer nodo?] Si INFO [PRINCIPIO] = ITEM,
Entonces: Hacer PTR : = PRINCIPIO,
PRINCIPIO: =ENL[PRINCIPIO],
ENL[PTR]: = DISP, DISP : = PTR,
INDIC : = VERDADERO y volver.
[Fin del condicional].
3.-Hacer PTR : = ENL[PRINCIPIO] y SALVA : = PRINCIPIO.
[Inicializar punteros].

4.-Repetir pasos 5 y 6 mientras PTR sea diferente a NULO:
5.- Si INFO [PTR] = ITEM entonces:
Hacer ENL [SALVA]: = ENL[PTR],
ENL[PTR]: = DISP, DISP: = PTR,
INDIC: =VERDADERO y volver.
[Fin de la condicional].
6.- Hacer SALVA := PTR y PTR := ENL [PTR].
[Actualizar punteros].
[Fin Bucle del paso 4].
7.- Hacer INDIC : = FALSO y Volver.

Por otro lado, suponga que queremos encontrar la posición POS de una arista (A B) del grafo G. Primero debemos encontrar la posición POSA de A y la posición POSB de B en la lista NODO. Luego debemos de buscar en la lista de sucesores de A, que tiene el puntero de lista ADY[POSA], la posición POS de POSB. Esto se implementa con el Procedimiento 8.5, que también comprueba que A y B sean nodos de G. Observe que POS de la posición de POSB en la lista ARISTA.

Procedimiento 8.5 BUSCARISTA(NODO, SIG, ADY, PRINCIPIO, DEST, ENL, A, B, POS)
Este procedimiento busca la posición POS de una arista (A, B) del grafo G, o pone POS : = NULO
1.-Llamar BUSCA(NODO, SIG, PRINCIPIO, A, POSA).
2.-Llamar BUSCA(NODO, SIG, PRINCIPIO, B, POSB).
3.-Si POSA: = NULO o POSB : = NULO, entonces: Hacer
POS : = NULO.
Si no: Llamar BUSCA (DEST, ENL, ADY[POSA], POSB, POS).
4.-Volver.

## INSERCIÓN EN UN GRAFO

Suponga que se va a insertar un nodo N en el grafo G. Observe que N será asignado a NODO [NDISP], el primer nodo disponible. Mas aún, como N será un nodo aislado, se debe hacer ADY[NDISP]: = NULO. El procedimiento 8.6 hace esto mediante una variable lógica INDIC que indica si hay desbordamiento.

Por supuesto, el Procedimiento 8.6 debe ser modificado si la lista NODO se mantiene como una lista ordenada o como un árbol binario de búsqueda.

Procedimiento 8.6 : INSNODO(NODO, SIG, ADY, PRINCIPIO, NDISP, N, INDIC)
Este procedimiento inserta el nodo N en le grafo G.

1.-[¿DESBORDAMIENTO?] Si NDISP= NULO, entonces: Hacer
INDIC := FALSO y volver.
2.- Hacer ADY[NDISP] : = NULO.
3.-[Quitar el nodo de la lista NDISP].
Hacer NUEVO: =NDISP y NDISP :=SIG[NDISP].
4.-[Insertar nodo N en la lista NODO].
Hacer NODO[NUEVO] : = N, SIG[NUEVO] :=PRINCIPIO y

PRINCIPIO : = NUEVO.
5.-Hacer INDIC := VERDADERO y Volver.

Suponga que se va a insertar una arista (A, B) en el grafo G (El procedimiento asumirá que tanto A como B son nodos de G). El procedimiento busca la posición POSA de A y la posición POSB de B en la lista NODO. Luego se inserta(A, B) como una arista de G insertando POSB en la lista de sucesores de A, que tiene el puntero de la lista ADY[POSA]. De nuevo se usa una variable lógica INDIC para indicar el desbordamiento.

El procedimiento es el siguiente:

Procedimiento 8.7.- INSARISTA(NODO, SIG, ADY, PRINCIPIO, DEST, ENL, ADISP, A, B, INDIC)

Este procedimiento inserta la arista (A, B) en el grafo G

1.-Llamar BUSCA(NODO, SIG, PRINCIPIO, A, POSA)
2.- Llamar BUSCA(NODO, SIG, PRINCIPIO, B, POSB)
3.- [¿DESBORDAMIENTO?] Si ADISP = NULO, entonces: Hacer INDIC := FALSO y Volver.
4.- [Eliminar nodo de la lista ADISP]. Hacer NUEVO: = ADISP y ADISP ; = ENL[ADISP]
5.-[Insertar POSB en la lista de sucesores de A ].
Hacer DEST[NUEVO]: = POSB,
ENL[NUEVO]: = ADY[POSA] y ADY[POSA]: = NUEVO.
6.-Hacer INDIC : = VERDADERO y Volver.

El procedimiento debe de ser modificado usando el procedimiento 8.6 si A o B no son nodos del grafo G.

ELIMINACIÓN DE UN ARCO EN UN GRAFO

Suponga que se va a eliminar una arista (A, B) del grafo G. (Nuestro procedimiento asumirá que A y B son nodos de G). De nuevo debemos encontrar primero la posición POSA de A y la posición POSB de B en la lista de nodos. Entonces simplemente eliminamos POSB de la lista de sucesores de A, que tiene el puntero de la lista ADY[POSA].Se usa una variable INDIC lógica para indicar si no existe tal arista en el grafo G. El procedimiento es el siguiente:

Procedimiento 8.8 ELIMARISTA(NODO, SIG, ADY, PRINCIPIO, DEST, ENL, ADISP, A, B, INDIC).

Este procedimiento elimina la arista (A, B) del grafo G .

1.-Llamar BUSCA(NODO, SIG, PRINCIPIO, A, POSA)[Localizar Nodo A].
2.- Llamar BUSCA(NODO, SIG, PRINCIPIO, B, POSB) [Localizar Nodo B].
3.-Llamar ELIMINAR (DEST, ENL, ADY[POSA], ADISP, POSB,

INDIC).[Usar el procedimiento 8.4].
4.-Volver.

Suponga que se va a eliminar un nodo N del grafo G. Esta operación es más complicada que las operaciones de búsqueda e inserción y que la de eliminar una arista, ya que debemos eliminar todas las aristas que contengan a N. Note que esas aristas son de dos tipos; aquellas que empiezan en N y aquellas que terminan en N. Así nuestro procedimiento consistirá fundamentalmente en los siguientes cuatro pasos:

(1).- Encontrar la posición POS del nodo N en G.

(2).- Eliminar todas las aristas que terminen en N; o sea, eliminar POS de la lista de sucesores de cada nodo M de G (este paso requiere recorrer la lista de nodos de G).

(3).- Eliminar todas las aristas que empiecen en N. Esto se hace encontrando la posición COM del primer sucesor y la posición FIN del ultimo sucesor de N, y luego añadiendo la lista de sucesores de N a la lista ADISP.

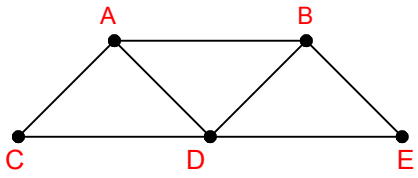
(4).- Eliminar el nodo N mismo de la lista de NODO.

El procedimiento es el siguiente :

Procedimiento 8.9 ELIMNODO(NODO, SIG, ADY, PRINCIPIO, NDISP, DEST, ENL, ADISP, N, INDIC).
Este procedimiento elimina el nodo N del grafo G.
1.- Llamar BUSCA (NODO, SIG, PRINCIPIO, N, POS).[Localizar el nodo N ].
2.- Si POS = NULO, entonces : Hacer INDIC ) : = FALSO y volver.
3.- [Eliminar aristas que terminan en N].
(a).- Hacer PTR : =PRINCIPIO.
(b).- Repetir mientras PTR se a diferente de NULO :
(i).- Llamar ELIMINAR (DEST, ENL, ADY,[PTR], ADISP, POS, INDIC).
(ii).- Hacer PTR := SIG[PTR].
[Fin del Bucle].
4.- [¿Lista de sucesores vacía?] Si ADY[POS]= NULO, entonces : Ir al paso 7.
5.-[Encontrar primer y ultimo sucesor de N].
6.-[Añadir lista de sucesores de N a la lista ADISP].
Hacer ENL[FIN] :=ADISP y ADISP := COM.
7.-[Eliminar N usando el procedimiento 8.4].
Llamar ELIMINAR(NODO, SIG, PRINCIPIO, NDISP, N, INDIC).
8.- Volver.

Ejemplo 8.6

Considere el grafo (no dirigido) G de la figura 8.12(a), cuyas listas de adyacencia aparecen en la figura 8.12(b) Observe que G tiene 14 aristas dirigidas, ya que existe 7 aristas no dirigidas.




| Listas de adyacencia |
| --- |
| A:B, C, D |
| B: A, D, E |
| C: A, D |
| D: A, B, C, E |
| E: B, D |

(a) (b)

Fig. 8.12

| | NODO | SIG | ADY |
|---|---|---|---|
| 1 | A | 2 | 1 |
| 2 | B | 3 | 4 |
| 3 | C | 4 | 7 |
| 4 | D | 5 | 9 |
| 5 | E | 0 | 13 |
| 6 | | 7 | |
| 7 | | 8 | |
| 8 | | 0 | |

PRINCIPIO - 1
NDISP --6

| | NODO | SIG | ADY |
|---|---|---|---|
| 1 | A | 3 | 2 |
| 2 | | 6 | |
| 3 | C | 4 | 7 |
| 4 | D | 5 | 9 |
| 5 | E | 0 | 14 |
| 6 | | 7 | |
| 7 | | 8 | |
| 8 | | 0 | |

PRINCIPIO - 1
NDISP --2

**FIG. 8.13**

| | DEST | ENL |
|---|---|---|
| 1 | 2 | 2 |
| 2 | 3 | 3 |
| 3 | 4 | 0 |
| 4 | 1 | 5 |
| 5 | 4 | 6 |
| 6 | 5 | 0 |
| 7 | 1 | 8 |
| 8 | 4 | 0 |
| 9 | 1 | 10 |
| 10 | 2 | 11 |
| 11 | 3 | 12 |
| 12 | 5 | 0 |
| 13 | 2 | 14 |
| 14 | 4 | 0 |
| 15 | | 16 |
| 16 | | 0 |

ADISP -16

a .- Antes de eliminar

| | DEST | ENL |
|---|---|---|
| 1 | | 15 |
| 2 | 3 | 3 |
| 3 | 4 | 0 |
| 4 | | 5 |
| 5 | | 6 |
| 6 | | 13 |
| 7 | 1 | 8 |
| 8 | 4 | 0 |
| 9 | 1 | 11 |
| 10 | | 1 |
| 11 | 3 | 12 |
| 12 | 5 | 0 |
| 13 | | 10 |
| 14 | 4 | 0 |
| 15 | | 16 |
| 16 | | 0 |

ADISP -4

b .-Tras eliminar B

Suponga que G se mantiene en memoria como en la figura 8.13(a). Más aún suponga que el nodo B se elimina de G mediante el Procedimiento 8.9. Obtenemos los siguientes pasos:
Paso 1.- Encuentra POS = 2, la posición de B en la lista de nodos.
Paso 3.- Elimina POS = 2 de la lista de aristas, o sea, de cada lista de sucesores.
Paso 5.- Encuentra COM = 4 y FIN = 6, el primer sucesor de B y el último.
Paso 6.- Elimina la lista de sucesores de la lista de aristas.
Paso 7.- Elimina el nodo B de la lista de nodos.
Paso 8.- Vuelve.

Los elementos eliminados esta rodeados con un círculo en la figura 8.13(a). La figura 8.13(b) muestra G en memoria tras eliminar el nodo B (y sus aristas).

## 8.7 RECORRIDO DE UN GRAFO

Muchos algoritmos de grafos requieren que se examinen sistemáticamente los nodos y las aristas de un grafo G. Esto se puede hacer de dos formas estándar. Una forma se llama búsqueda en anchura y la otra búsqueda en profundidad. La búsqueda en anchura usará una cola como estructura auxiliar para mantener los nodos que se vayan a procesar posteriormente, y análogamente, la búsqueda en profundidad usará una pila.

Durante la ejecución de nuestros algoritmos, cada nodo N de G estará en uno de tres estados, lo que se llama *estado* de N, tal como sigue:

ESTADO = 1: (Estado preparado). El estado inicial del nodo N.
ESTADO = 2: (Estado de espera). El nodo N está en la cola o en la pila, esperando a ser procesado.
ESTADO = 3: (Estado de procesado). El nodo N ha sido procesado.

Ahora discutiremos las dos búsquedas por separado.

### Búsqueda en anchura

La idea general de la búsqueda en anchura comenzando en un nodo de partida A es la siguiente. Primero examinamos el nodo de la partida A. Luego examinamos todos los vecinos de A. Luego examinamos todos los vecinos de los vecinos de A. Y así sucesivamente. Naturalmente, necesitamos seguir la pista de los vecinos de un nodo y tenemos que garantizar que ningún nodo se procesa más de una vez. Esto se hace usando una cola para mantener los nodos que estén esperando para ser procesados y usando un campo ESTADO que nos diga el estado actual de los nodos. El algoritmo es el siguiente:

Algoritmo A : Este algoritmo realiza una búsqueda en anchura en un grafo G comenzando en un nodo de partida A.

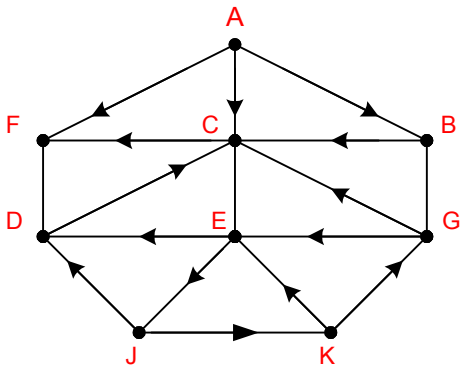
1.- Inicializar todos los nodos al estado de preparados (ESTADO = 1)
2.- Poner al nodo de partida A en COLA y cambiar su estado a espera (ESTADO = 2).
3.-Repetir pasos 4 y 5 hasta que COLA este vacía;
4.- Quitar el nodo del principio de la cola, N. Procesar N y cambiar su estado a procesado (ESTADO = 3).
5.- Añadir a COLA todos los vecinos de N que estén en estado de preparados (ESTADO = 1) y cambiar su estado al de espera (ESTADO = 2).
[Fin del bucle del paso 3].
6.- Salir.

El anterior algoritmo procesará solo aquellos nodos que sean alcanzables desde el nodo de partida A. Suponga que se quiere examinar todos los nodos del grafo G. Entonces el

algoritmo debe de ser modificado para que comience de nuevo en otro nodo (que llamaremos B) que este todavía en el estado de preparado. Este nodo B se pude recorrer obteniendo la lista de nodos.

Ejemplo 8.7

Considere el grafo G de la figura 8.14(a). [Las listas de adyacencia de los nodos aparecen en la figura 8.14(b)]. Suponga que G representa los vuelos diarios entre ciudades de alguna compañía aérea, y suponga que deseamos volar de la ciudad A a la ciudad J con el mínimo número de paradas. En otras palabras, queremos saber el camino mínimo P desde A hasta J (donde cada arista tiene longitud 1).




(a)

| LISTAS DE ADYACENCIA |
|---|
| A: F,C, B |
| B:G, C |
| C: F |
| D: C |
| E: D, C, J |
| F: D |
| G: C, E |
| J: D, K |
| K: E, G |

(b)

FIG. 8.14

El camino mínimo de P se puede obtener usando el algoritmo de búsqueda en anchura comenzando en la ciudad A y terminando cuando se encuentre la ciudad J. Durante la búsqueda, también seguiremos la pista del origen de cada arista usando un array ORIG junto con el array COLA. Los pasos para nuestra búsqueda son:

(a).- Inicialmente, añadir A a COLA y añadir NULO a ORIG como sigue:

FRENTE = 1 COLA: A
FINAL = 1 ORIG: 0

(b).- Quitar el elemento A al frente de COLA haciendo FRENTE : = FRENTE+1 y añadir a COLA los vecinos de A como sigue:

FRENTE = 2 COLA: A, F, C, B
FINAL = 4 ORIG : 0, A, A, A

Note que el origen A de cada una de las tres aristas se añade a ORIG.

(c).- Quitar el elemento F del frente de COLA haciendo FRENTE := FRENTE +1 y añadir a COLA los vecinos de F como sigue:

FRENTE = 3 COLA : A, F, C, B, D
FINAL = 5 ORIG : 0, A, A, A, F

(d).- Quitar el elemento C al frente de COLA y añadir a COLA los vecinos de C (que estén en el estado de preparados) como sigue:

FRENTE = 4 COLA : A, F, C, B, D
FINAL = 5 ORIG : 0, A, A, A, F

Note que el vecino F de G no se añade a COLA, ya que F no esta en estado de preparado (porque ya ha sido añadido a COLA).

(e).- Quitar el elemento B al frente de COLA y añadir a COLA los vecinos de B (con estado de preparados) como sigue:

FRENTE = 5 COLA : A, F, C, B, D, G
FINAL = 6 ORIG: 0, A, A, A, F, B

(f).- Quitar el elemento D al frente de COLA y añadir a COLA los vecinos de D (con estado de preparados) como sigue:

FRENTE = 6 COLA : A, F, C, B, D, G
FINAL = 6 ORIG: : 0, A, A, A, F, B

(g).- Quitar el elemento G al frente de COLA y añadir a COLA los vecinos de G (con estado de preparados) como sigue:

FRENTE = 7 COLA : A, F, C, B, D, G, E
FINAL = 7 ORIG : 0, A, A, A, F, B, G

(h).- Quitar el elemento E al frente de COLA y añadir a COLA los vecinos de E (con estado de preparados) como sigue:

FRENTE = 8 COLA : A, F, C, B, D, G, E, J
FINAL = 8 ORIG : 0, A, A, A, F, B, G, E

Paramos tan pronto como J se añade a COLA, ya que J es nuestro destino final. Ahora volvemos hacia atrás desde J usando el array ORIG para encontrar el camino P. Así

J ← E ← G ← B ← A

Es el camino P requerido.

## Búsqueda en profundidad

La idea general de la búsqueda en profundidad comenzando en un nodo A es la siguiente, primero examinamos el nodo inicial A, luego examinamos cada nodo N de un camino P que comience en A; o sea, procesamos un vecino de A. Luego un vecino de un vecino de A y así sucesivamente. Tras llegar a un punto muerto, o sea, al final del camino P, volvemos atrás por P hasta que podamos continuar por otro camino P'. Y así sucesivamente (Este algoritmo es similar al recorrido en orden de un árbol binario, y también a la forma en que se debe pasar a través de un laberinto). El algoritmo es muy similar al de búsqueda en anchura excepto que usamos una pila en lugar de una cola. De nuevo se usa un campo ESTADO para indicar el estado actual de un nodo.
El algoritmo es el siguiente:

Algoritmo B Esta algoritmo realiza una búsqueda en profundidad en el grafo G comenzando en un nodo A.

1.- Inicializar todos los nodos al estado preparado (ESTADO =1)
2.- Meter en el nodo inicial A en l apila y cambiar su estado a estado de espera (ESTADO = 2) .
3.- Repetir los paso 4 y 5 hasta que la pila esta vacía.
4.- Sacar el nodo N en la cima de la pila. Procesar el nodo N y cambiar su estado a procesado (ESTADO = 3) .
5.- Meter en la pila todos los vecinos de N que estén en estado de preparados (ESATDO = 1) y cambiar su estado al de espera (ESTADO = 2).
[Fin de l Bucle del paso 3] .
6.- Salir.

De nuevo este algoritmo procesa solo los nodos que sean alcanzables desde el nodo de partida A. Suponga que queremos examinar todos los nodos de G. Entonces el algoritmo debe de ser modificado para que comience de nuevo con otro nodo al que llamaremos B que este en su estado de preparado. Este nodo se puede obtener recorriendo la lista de nodos.

## Ejemplo 8.8

Considere el grafo G de la figura 8.14 (a). Suponga que queremos encontrar e imprimir todos los nodos alcanzables desde el nodo J (incluyendo J). Una forma de hacerlo es usando el algoritmo de búsqueda en profundidad para G comenzando en el nodo J. Los pasos de la búsqueda son:

(a).- Inicialmente, meter J en la pila como sigue:

PILA: J

(b).- Sacar e imprimir el elemento en la cima y luego meter en la pila todos los vecinos de J (con estado de preparado) como sigue:

Imprimir J PILA: D, K

(c).- Sacar e imprimir el elemento en la cima K y luego meter en la pila todos los vecinos de K (en estado preparado) como sigue:

Imprimir K  PILA: D, E, G

(d).- Sacar e imprimir G y luego meter en la pila todos los vecinos de G (en estado preparado)

Imprimir G  PILA D, E, G

Note que solo se mete C en la pila, ya que el otro vecino, E no esta en estado preparado (porque E ya ha sido metido en la pila)

(e).- Sacar e imprimir C y luego meter en la pila todos los vecinos de C (en estado preparado) como sigue:

Imprimir C  PILA: D, E, F

(f).- Sacar e imprimir el elemento F y luego meter en la pila todos los elementos de F (en estado preparado) como sigue:

Imprimir F  PILA: D, E

Note que el único vecino de F, D, no se mete en la pila, ya que D esta en estado preparado (porque ya ha sido metido en la pila)

(g).- Sacar e imprimir E y meter en la pila todos los vecinos de E (en estado preparado) como sigue:

Imprimir E  PILA: D

(Note que ninguno de los tres vecinos de E esta en estado preparado)

(h).- Sacar e imprimir el elemento D y meter en la pila todos los vecinos de D (en estado preparado) como sigue:

Imprimir D  PILA:

Ahora la pila esta vacía, así que la búsqueda en profundidad de G comenzando en J esta completa. De acuerdo con ello, los nodos que se imprimieron.

J, K, G, C, F, E, D

Son precisamente los nodos que son alcanzables desde J.

## 8.8 CONJUNTO PO : ORDENACION TOPOLÓGICA

Suponga que S es un grafo tal que cada nodo $v_1$ de S representa una tarea y cada arista *(u, v)* significa que la finalización de la tarea *u* es un prerrequisito para que comience la tarea *v*. Suponga que tal grafo S contiene un ciclo tal que:

P= (u, v, w, u)

Esto significa que no podemos empezar $v$ hasta completar $u$, no podemos empezar $w$ hasta terminar $v$ y no podemos empezar $u$ hasta terminar $w$. Así no podemos completar ninguna tarea del ciclo. Por tanto, un grafo S así, que representa tareas y relaciones de precedencia, no puede tener ciclos.

Suponga que S es un grafo sin ciclos. Considere la relación < sobre S definida como sigue;

$$u < v \quad \text{si existe un camino de } u \text{ a } v$$

(1).-Para cada elemento de $u$ de S, tenemos que $u \not< u$ (Irreflexividad)
(2).-S $u < v$, entonces $v \not< u$ (Asimetría)
(3).-Si $u < v$ y $v < w$, entonces u < w (Transitividad)

Tal relación sobre S se le llama ordenación parcial de S, y S con tal ordenación se llama conjunto parcialmente ordenado o conjunto po. Así un grafo S sin ciclos se puede considerar un conjunto parcialmente ordenado.

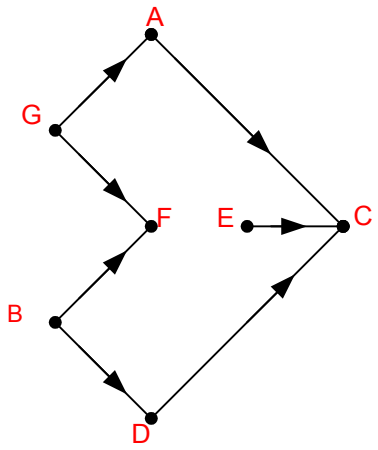
Por otro lado, suponga que S es un conjunto parcialmente ordenado con la ordenación parcial denotado por <. Entonces S se puede considerar un grafo cuyos nodos son los elementos de S y cuyas aristas están definidas como sigue:

$$(u, v) \quad \text{es una arista en S si } u < v$$

Más aún, se pude demostrar que un conjunto S parcialmente ordenado, considerado como un grafo, no tiene ciclos.

Ejemplo 8.9

Sea S el grafo de la figura 8.15. Observe que S no tiene ciclos. Así S se puede considerar un conjunto parcialmente ordenado. Note que G < C, ya que existe un camino desde G hasta C. Similarmente, F < B y B <C. Por otro lado, B $\not<$ A, ya que no existe camino de B a A. También A$\not<$B.




(a)

| Listas de adyacencia |
|---|
| A: C |
| B: D, F |
| C: |
| D: C |
| E: C |
| F: |
| G: A, F |

(b)

Fig: 8.15

## Ordenación topológica

Sea S un grafo dirigido sin ciclos (o conjunto parcialmente ordenado). Una ordenación topológica T de S es una ordenación lineal de los nodos de S que preserva la ordenación parcial de S original. O sea, que si $u < v$ en S (si hay un camino de $u$ a $v$ en S ) entonces $u$ va delante de $v$ en la ordenación lineal T. La figura 8.16 muestra dos ordenaciones topológicas del grafo S de la figura 8.15. Hemos concluido las aristas en la figura 8.16 para indicar que concuerdan en la dirección de la ordenación lineal.

El siguiente es el resultado principal de esta sección:

Proposición 8.4 Sea S un grafo dirigido finito sin ciclos o un conjunto parcialmente ordenado. Entonces existe una ordenación topológica T del conjunto S.

Note que la proposición solo establece que la ordenación topológica existe. Ahora damos un algoritmo que encuentra la ordenación topológica.

La idea principal de nuestro algoritmo de búsqueda de la ordenación topológica T de un grafo S sin ciclos es cualquier nodo N con grado de entrada cero, o sea, sin predecesores, debe ser :

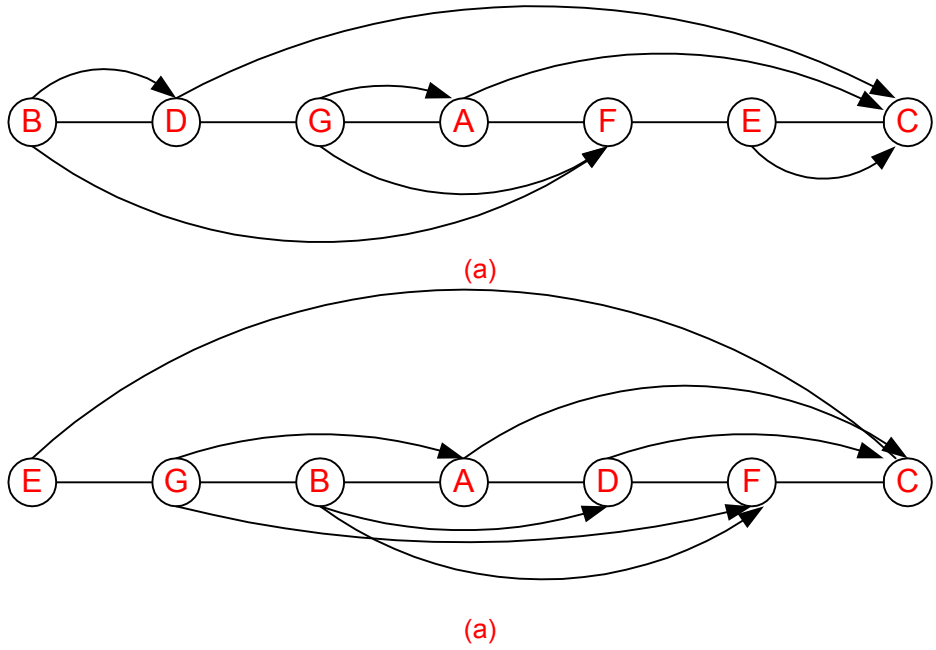



Fig. 8.16 ORDENACIONES TOPOLOGICAS

escogido como primer elemento en la ordenación T. De acuerdo con esto, nuestro algoritmo repetirá los siguientes pasos hasta que el grafo S este vacío:

(1).- Buscar un nodo N con grado de entrada nulo.
(2).- Eliminar N y sus aristas del grafo S.

El orden en que los nodos son eliminados del grafo S usara un array auxiliar COLA que contendrá temporalmente todos los nodos con grado de entrada nulo. El algoritmo también usa un campo GRADENT tal que GRADENT(N) contendrá el grafo de entrada actual del nodo N. El algoritmo es el siguiente:

Algoritmo C Este algoritmo encuentra una ordenación topológica T de un grafo S sin ciclos.

1.- Encontrar el grado de entrada GRADENT(N) de cada nodo N de S.(se puede hacer haciendo el recorriendo cada lista de adyacencia como el problema 8.15)
2.- Poner en una cola todos los nodos con grado de entrada nulo.
3.- Repetir pasos 4 y 5 hasta que la cola este vacía:
4.- Quitar el nodo N al frente de la cola (haciendo FRENTE := FRENTE +1).
5.- Repetir lo siguiente para cada vecino M de N:
(a).- Hacer GRADENT(M): = GRADENT(M)-1.
[esto elimina la arista de N a M]
(b).- Si GRADENT(M)= 0, entonces añadir M al final de la cola.
[Fin de Bucle].
[Fin de Bucle paso 3].
6.- Salir.

Ejemplo 8.10

Considere el grafo S de la figura 8.15(a). Aplicamos nuestro algoritmo C para buscar una ordenación topológica T del grafo S. Los pasos del algoritmo van a continuación:

1.- Buscar el grado de entrada GRADENT(N) de cada nodo N del grafo S. Esto da:

GRADENT(A) = 1 GRADENT(B) = 0 GRADENT(C) = 3 GRADENT(M)= 1
GRADENT(E) = 0 GRADENT(F) = 2 GRADENT(G) = 0

[Esto se pude hacer como en el problema 8.15.]

2.- Inicialmente añadir ala cola cada nodo con grado de entrada cero como sigue:

FRENTE : = 1, FINAL: = 3, COLA: B, E, G

3a.- Quitar el elemento B al frente de la cola haciendo FRENTE:= FRENTE +1, tal como sigue:

FRENTE : = 2, FINAL: = 3, COLA: B, E, G

3b.- Decrementar en 1 el grado de entrada de cada vecino de B, tal como sigue:

GRADENT(D) = 1-1 = 0 y GRADENT(F) = 2-1 = 1

[La lista de adyacencia de B en la figura 8.15 (b) se usa para encontrar a los vecinos de B, D y F.] El vecino D se añade al final de la cola, ya que su grado de

entrada ahora es cero:

FRENTE : = 2,  FINAL: = 4,  COLA: B, E, G, D

[El grafo S ahora esta como en la figura 8.15 (a), donde el nodo B y sus aristas han sido eliminados, como se indica por las líneas punteadas.]

4a.- Quitar el elemento E de la cola haciendo FRENTE: = FRENTE+1 , tal como sigue:

FRENTE : = 3,  FINAL: = 4,  COLA: B, E, G, D

4b.- Decrementar en 1el grado de entrada de cada vecino de E, tal como sigue:

GRADENT(C) = 3-1 = 2

[Como el grado de entrada no es cero, la COLA no cambia. El grafo S ahora esta como en la figura 8.17(b), donde el nodo E y sus aristas han sido eliminados.]

5a.- Quitar el elemento G de la cola haciendo FRENTE:= FRENTE +1, tal como sigue:

FRENTE : = 4,  FINAL: = 4,  COLA: B, E, G, D

5b.- Decrementar en 1 el grado de entrada de cada vecino de G:

GRADENT (A) = 1-1 = 0  y  GRADENT(F) = 1-1 = 0

Ambos A y F se añaden al final de la cola tal como sigue:

FRENTE : = 4,  FINAL: = 6,  COLA: B, E, G, D, A, F

[El grafo S ahora esta como en la figura 8.17(c), donde G y sus dos aristas han sido eliminados.]

6a.- Quitar D de la cola haciendo FRENTE:= FRENTE+1, como sigue:

FRENTE : = 5,  FINAL: = 6,  COLA: B, E, G, D, A, F

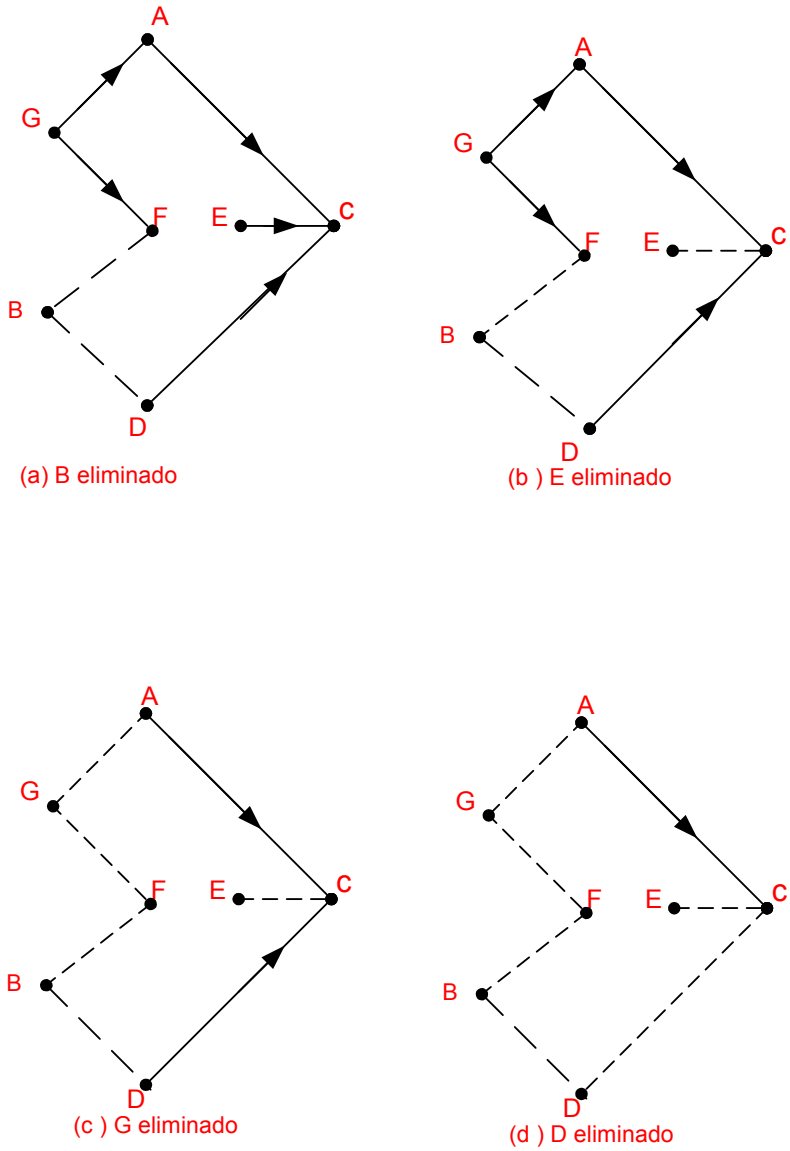



Fig. 8.17

6b.- Decrementar en 1 el grado de entrada de cada vecino de D:

GRADENT(C) = 2-1 = 1

[Como el grado de entrada no es cero, la COLA no cambia. El grafo S ahora esta como en la figura 8.17(d), donde D y su arista han sido eliminados.]

7a.- Quitar el elemento A de la cola haciendo FRENTE : = FRENTE+1, tal como sigue:

FRENTE : = 6, FINAL: = 6, COLA: B, E, G, D, A, F

7b.- Decrementar en 1 el grado de entrada de cada vecino de A :

GRADENT(C) = 1-1 = 0

Añadir C al final de la cola, ya que ahora su grado de entrada es cero:

FRENTE : = 6,  FINAL: = 7,  COLA: B, E, G, D, A, F, C

8a.- Quitar de la cola el elemento F haciendo FRENTE: = FRENTE+1, tal como sigue:

FRENTE : = 7,  FINAL: = 7,  COLA: B, E, G, D, A, F, C

8b.- El nodo F no tiene vecinos, así que no hay cambios.

9a.- Quitar el elemento C de la cola haciendo FRENTE: = FRENTE+1, quedando:

FRENTE : = 8,  FINAL: = 7,  COLA: B, E, G, D, A, F, C

9b.- El nodo C no tiene vecinos, así que no cambia nada.

Ahora la cola no tiene elemento al frente, así que el algoritmo ha terminado. Los elementos en el array COLA dan la ordenación topológica T de S requerida, que es:

T: B, E, G, D, A, F, C

El algoritmo podría haber parado en el paso 7b, donde FINAL es igual al número de nodos del grafo S.

## PROBLEMAS RESUELTOS

## TERMINOLOGÍA DE GRAFOS

8.1 Considere el grafo (no dirigido) G de la figura 8.18. (a) Describir G formalmente en términos de su conjunto V de nodos y de su conjunto E de aristas. (b) Encontrar el grado de cada nodo.

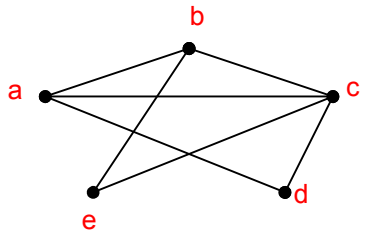


Fig. 8.18

(a).- Existen 5 nodos, a, b, c, d y e; por tanto, V = { a, b, c, d, e}. Hay siete pares [x, y] de nodos tales que el nodo x esta conectado con el nodo y; por tanto:

E = {[ a, b], [a, c,], [a, d],[b, c,], [b, e], [c, d ],[ c, e]}

(b).- El grado de un nodo es igual al numero de aristas a las que pertenece; por ejemplo, grad(a) = 3, ya que a pertenece a tres aristas, [ a, b], [a, c,] y [a, d]. De forma similar, grad(b) = 3, grad(c) = 4, grad(d) = 2 y grad(e) = 2.

8.2 Considere los multigrafos de la figura 8.19. ¿Cuáles de ellos son (a) conexos; (b) libres de bucles, (c) grafos?

(a).- Solo los multigrafos 1 y 3 son conexos.
(b).- Solo el multigrafo 4 tiene un bucle(o sea, un arista con iguales extremos).
(c).- Solo los multigrafos 1 y 2 son grafos. El multigrafo 3 tiene aristas múltiples y el multigrafo 4 tiene aristas múltiples y un bucle.

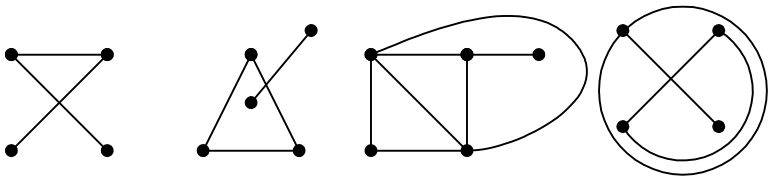



Fig. 8.19

Considere el grafo conexo G de la figura 8.20. (a). Encontrar todos los caminos simples del nodo A al nodo F. (b) Encontrar la distancia entre A y F. (c) Encontrar el diámetro de G. (El diámetro de G es la máxima distancia existente entre dos de sus nodos.)

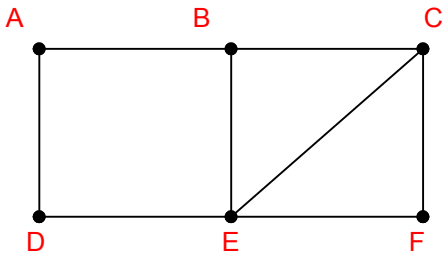



Fig. 8.20

(a).- Un camino simple entre A y F es un camino tal que no se repiten nodos o aristas. Existen 7 caminos simples así:

(A, B, C, F) (A, B, E, F) (A, D, E, F) (A, D, E, C, F)

(A, B, C, E, F) (A, B, E, C, F) (A, D, E, B, C, F)

(b).- La distancia entre A y F es igual a 3, ya que existe un camino simple (A, B, C, F), desde A hasta F de longitud 3 y no hay otro camino mas corto entre A y F.

(c).- La distancia entre A y F es igual a 3, y la distancia entre cualesquiera otros dos nodos no excede de 3; por tanto, el diámetro del grafo G es igual a 3.

8.4 Considere el grafo (dirigido) G de la figura 8.21 (a) Encontrar todos los caminos simples de X a Z. (b). Encontrar todos los camino simples de Y a Z. (c). Encontrar el gradent(Y) y el gradsal(Y). ¿Existen fuentes o sumideros?

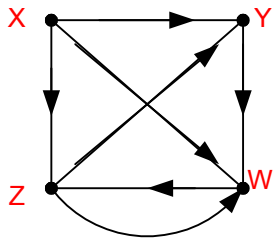


Fig. 8.21

(a).- Hay tres caminos simples de X a Z: (X, Z), (X, W, Z) y (X, Y, W, Z).
(b).- Solo hay un camino simple de W a Z: (Y, W, Z).
(c).- Como dos aristas entran en Y (o sea, terminan en Y), tenemos que gradent(Y) = 2. Como solo una arista sale de Y (o sea empieza en Y), el gradsal(Y) = 1.
(d).- X es una fuente, ya que ninguna arista entra en X [o sea, gradent(X) = 0], pero algunas aristas salen de X [o sea, gradsal (X) > 0]. No hay sumideros, ya que todos los nodos tienen un grado de salida no nulo(o sea, cada nodo es punto inicial de alguna arista).

8.5 Dibujar todos los árboles (no similares) con exactamente seis nodos. [Un grafo G es similar a un grafo G' si existe una correspondencia de uno a uno entre el conjunto V de nodos de G y el conjunto V' de nodos de G' tal que *(u, v)* es una arista en G si y solo si el correspondiente par *(u', v')* de nodos es una arista de G']

Existen seis árboles así, que se muestran en la figura 8.22. El primer árbol tiene diámetro 5, los dos siguientes tienen diámetro 4, los dos siguientes diámetro 3 y el último diámetro 2. Cualquier otro árbol de seis nodos será similar a alguno de estos árboles.

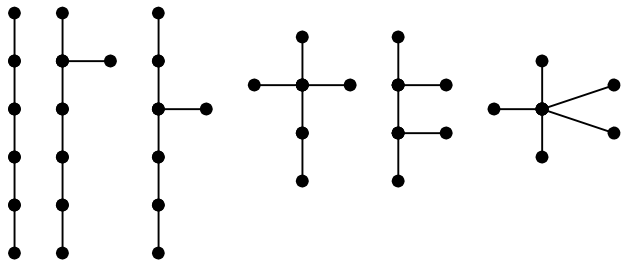


Fig. 8.22

8.6.- Encontrar todos los árboles extendiendo el grafo G de la figura 8.23 (a). (Un árbol T se llama árbol extendido de un grafo conexo G si T tiene los mismo s nodos que G y todas las aristas de T están contenidas entre las aristas de G.)

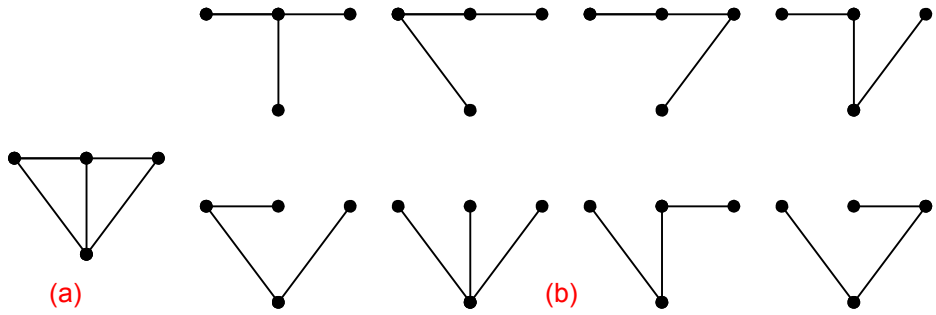



Fig. 8.23

Existen 8 árboles extendidos así, como se muestra en la figura 8.23 (b). Como G tiene 4 nodos, cada árbol extendido T debe tener 4-1 = 3 nodos. Así, cada árbol extendido se puede obtener eliminando dos de las 5 aristas de G. Esto se puede hacer de 10 formas, pero dos de ellas producen grafos disconexos. Por tanto, los ocho árboles extendidos mostrados son los árboles extendidos del grafo G.

## REPRESENTACIÓN SECUENCIAL DE GRAFOS

8.7.- Considere el grafo G de la figura 8.21. Suponga que los nodos se mantienen en memoria en un array DATOS tal como sigue:

DATOS: X, Y, Z, W

(a).- Encontrar la matriz de adyacencia A del grafo G.
(b).- Encontrar la matriz de caminos P de G mediante las potencias de la matriz de adyacencia A.
(c).- ¿Es G fuertemente conexo?

(a).- Los nodos están normalmente ordenados de acuerdo con la forma en que aparecen en memoria; o sea, asumimos que $v_1$= X, $v_2$= Y, $v_3$= Z, $v_4$= W. La matriz de adyacencia A de G es la siguiente:

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

Aquí $a_{i,j} = 1$ si hay una arista de $vi$ a $vj$; si no, $a_{i,j} = 0$

(b).- Como G tiene cuatro nodos , se calcula las potencias de $A_{(2)}$, $A_{(3)}$, $A_{(4)}$ y $B_4 = A + A_{(2)} + A_{(3)} + A_{(4)}$.

$$A^2=\begin{pmatrix}0&1&1&2\\0&0&1&1\\0&0&1&1\\0&1&0&1\end{pmatrix} \quad A^3=\begin{pmatrix}0&1&2&2\\0&1&0&1\\0&1&1&1\\0&0&1&1\end{pmatrix} \quad A^4=\begin{pmatrix}0&2&2&3\\0&0&1&1\\1&0&1&1\\0&1&1&1\end{pmatrix} \quad B_4=\begin{pmatrix}0&5&6&8\\0&1&2&3\\0&3&3&5\\0&2&3&5\end{pmatrix}$$

La matriz de caminos P se obtiene ahora haciendo $p_{ij} = 1$ siempre que haya una entrada positiva en la matriz $B_4$ . Así.

$$P=\begin{pmatrix}0&1&1&1\\0&1&1&1\\0&1&1&1\\0&1&1&1\end{pmatrix}$$

(c).- La matriz de caminos muestra de $v_2$ a $v_1$. De hecho, no hay camino de ningún nodo a $v_1$ . Por tanto G no es fuertemente conexo.

8.8 Considere el grafo G de la figura 8.21 y su matriz de adyacencia A obtenida en el problema 8.7. Encontrar la matriz de caminos P de G mediante el algoritmo de Warshall en vez de por las potencias de A.

Calculamos las potencias $P_0$, $P_1$, $P_2$, $P_3$ y $P_4$, donde inicialmente $P_0 = A$ y

$$P_{k-1}[i, j] = P_{k-1}[i, j] \quad V \quad (P_{k-1}[i, k] \wedge P_{k-1}[k, j])$$

O sea,

$$P_k[i, j] = 1 \quad \text{si} \quad P_{k-1}[i, j] = 1 \quad \text{o tanto} \quad P_{k-1}[i, k] = 1 \quad \text{como} \quad P_{k-1}[k, j] = 1$$

Entonces;

$$P_1=\begin{pmatrix}0&1&1&1\\0&0&0&1\\0&1&0&1\\0&0&1&0\end{pmatrix} \quad P_2=\begin{pmatrix}0&1&1&1\\0&0&0&1\\0&1&0&1\\0&0&1&0\end{pmatrix}$$

$$P_3=\begin{pmatrix}0&1&1&1\\0&0&0&1\\0&1&0&1\\0&1&1&1\end{pmatrix} \quad P_4=\begin{pmatrix}0&1&1&1\\0&1&1&1\\0&1&1&1\\0&1&1&1\end{pmatrix}$$

Observe que $P_0 = P_1 = P_2 = A$ . Los cambios en $P_3$ se dan por las siguientes razones:

$P_3(4, 2) = 1$ ya que $P_2(4, 3) = 1$ y $P_2(3, 2) = 1$
$P_3(4, 4) = 1$ ya que $P_2(4, 3) = 1$ y $P_2(3, 4) = 1$

Los cambios en $P_4$ se dan similarmente. La ultima matriz, $P_4$, es la matriz de caminos P requerida para el grafo G.

8.9 .-Considere el grafo (no dirigido) con peso de G de la figura 8.24. Suponga que los nodos se guardan en memoria en un array DATOS como sigue:

DATOS: A, B, C, X, Y

Encontrar la matriz de pesos W=($w\ ij$) del grafo G.

Asumiendo que $v_1$= A, $v_2$ = B, $v_3$=C, $v_4$=X y $v_5$=Y, llegamos a la siguiente matriz de pesos W de G:

$$W = \begin{pmatrix} 0 & 6 & 0 & 4 & 1 \\ 6 & 0 & 5 & 0 & 8 \\ 0 & 5 & 0 & 0 & 2 \\ 4 & 0 & 0 & 0 & 3 \\ 1 & 8 & 2 & 3 & 0 \end{pmatrix}$$

Aquí w $ij$ denota el peso de la arista de $v$i a $v$j. Como G es no dirigido, $w$ es una matriz simétrica, o sea, $w\ ij = w\ ji$ .

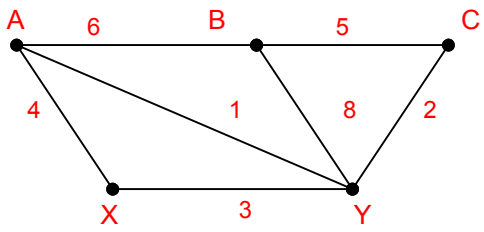


Fig. 8.24

8.10 Suponga que G es un grafo (no dirigido) libre de bucles. Sea P= ($p\ ij$) la matriz de caminos de G

(a).- ¿Cuándo se pude añadir una arista [$vi\ vj$] a G de forma que G siga siendo libre de ciclos?
(b).- ¿Cómo cambia la matriz de caminos P cuando se añade una arista [$vi\ vj$] a G?
(a).- La arista [$vi\ vj$] formara un ciclo al añadirse a G si y solo si ya existe un camino entre $vi$ y $vj$ . Por tanto, la arista se puede añadir a G si $pij$ = 0.
(b).- Primero hacemos $p\ ij$ = 1, ya que la arista es un camino de $v$i a $v$j . También hacemos $p\ si$= 1 si $psi$ = 1 y $p\ jt$ = 1. En otras palabras, si existe tanto un camino $P_1$ de $vs$ a vi como un camino $P_2$ de $vj$ a $vi$, entonces $P_1$, [$vi$ vj] y $P_2$ formaran un camino de $vs$ a $vt$ e

8.11 Un árbol extendido mínimo T para un grafo G con peso es el árbol extendido de G (véase prob. 8.6) que tiene el mínimo peso de entre todos los árboles extendidos de G.
(a).- Describir un algoritmo que encuentre el árbol extendido mínimo T de un grafo G.
(b).- Encontrar un árbol extendido mínimo T para el grafo de la figura 8.24.

(a).- Algoritmo P8.11 Este algoritmo encuentra el árbol extendido mínimo T de un

grafo con peso G.

1.- Ordenar todas las aristas de G en orden creciente de pesos.

2.- Inicializar T como un grafo consistente en los mismos nodos que G sin Aristas.

3.- Repetir lo siguiente M-1 veces, donde M es el numero de nodos de G :

Añadir a T una arista E de G con peso mínimo de forma que E no forme un ciclo en T.

[Fin del Bucle].

4.- Salir.

El paso 3 se puede implementar usando los resultados del problema 8.10. El problema 8.10 (a) nos dice que arista *e* de G se pude añadir a T de forma que no forme un ciclo, o sea, para que T este libre de ciclos, y el problema 8.10(b) nos dice como seguir la pista a la matriz de caminos P de T a medida que se añade cada arista *e* a T.

(b).- Se aplica el algoritmo P8.11 para obtener el árbol expandido T de la figura 8.25. Aunque [A, X] tiene menor peso que [B, C], no podemos añadir [A, X] a T, ya que formaría un ciclo con [A, Y] y [Y, Z].

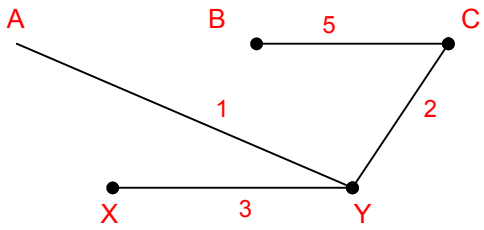


Fig. 8.25

8.12 Suponga que un grafo con peso G se mantiene en memoria mediante un array de nodos DATOS y una matriz de pesos W como sigue:

DATOS X, Y, S, T

$$W = \begin{pmatrix} 0 & 0 & 3 & 0 \\ 5 & 0 & 1 & 7 \\ 2 & 0 & 0 & 4 \\ 0 & 6 & 8 & 0 \end{pmatrix}$$

Dibujar el diagrama de G.

El dibujo aparece en la figura 8.26. los nodos están etiquetados como las entradas de DATOS. También, si $w\ ij = 0$, entonces existe una arista de $vi$ a $vj$ con peso $w\ ip$ (asumimos que $v_1 = X$, $v_2 = Y$, $v_3 = S$, $v_4 = T$), el orden en que aparecen los nodos en el array DATOS.)

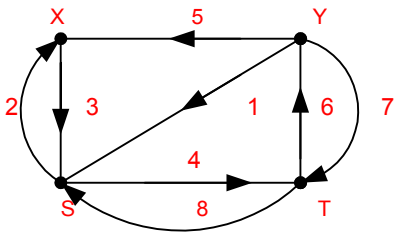



FIG. 8.26

## REPRESENTACIÓN ENLAZADA DE GRAFOS

8.13 Un grafo G se guarda en memoria como sigue:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| NODO | A | B | | E | | D | C | |
| SIG | 7 | 4 | 0 | 6 | 8 | 0 | 2 | 3 |
| ADY | 1 | 2 | | 5 | | 7 | 9 | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

PRINCIPIO -1, NDISP - 5

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| DEST | 2 | 6 | 4 | | 6 | 7 | 4 | | 4 | 6 |
| ENL | 10 | 3 | 6 | 0 | 0 | 0 | 0 | 4 | 0 | 0 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

ADISP- 8

Dibujar el grafo G.
Primero buscamos todos los vecinos de cada nodo NODO(K) recorriendo su lista de adyacencia que tiene el puntero ADY[J]. Esto da:

A: 2(B) y 6(D)
B: 6(D), 4(E) y 7(C)
C: 4(E)
D: 4(E)
E: 6(D)

Entonces se dibuja el diagrama como en la figura 8.27.

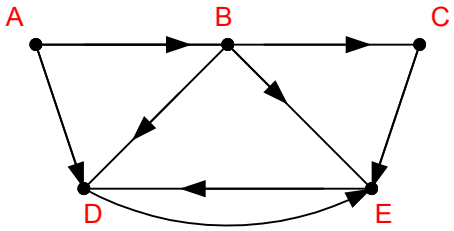



Fig:8.27

8.14 Encontrar los cambios en la representación enlazada del grafo G del problema 8.13 si se dan las siguientes operaciones: (a) Se añade el nodo F a G. (b) Se elimina la arista (B, E) de G, (c) Se añade la arista (A, F) a G. Dibujar el grafo G resultante.

(a).- La lista de nodos no esta ordenada, así que insertamos F al principio de la lista, usando el primer nodo libre disponible tal como sigue:

PRINCIPIO 5

NDISP 8

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| NODO | A | B | | E | F | D | C | |
| SIG | 7 | 4 | 0 | 6 | 1 | 0 | 2 | 3 |
| ADY | 1 | 2 | | 5 | 0 | 7 | 9 | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Observe que la lista de aristas no cambia.

(b).- Eliminar POS = 4 del nodo E de la lista de adyacencia del nodo B tal como sigue:

ADISP 3

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| DEST | 2 | 6 | 4 | | 6 | 7 | 4 | | 4 | 6 |
| ENL | 10 | 6 | 8 | 0 | 0 | 0 | 0 | 4 | 0 | 0 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

ADISP 8

Observe que la lista de nodos no cambia.

(c).- La posición POS = 5 del nodo F se inserta al principio de la lista de adyacencia de l nodo A, usando la primera arista libre disponible. Los cambios son los siguientes:

ADY[1]= 3
ADISP=8

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| DEST | 2 | 6 | 5 | | 6 | 7 | 4 | | 4 | 6 |
| ENL | 10 | 6 | 1 | 0 | 0 | 0 | 0 | 4 | 0 | 0 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

El único cambio de la lista de nodos ADY[1] = 3 (Observe que el sombreado indica los cambios en las listas) El grafo G actualizado aparece en la figura 8.28.

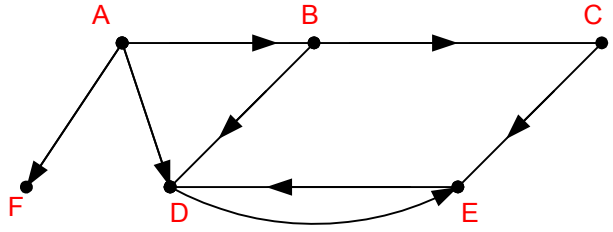



Fig:8.28

8.15 Suponga que un grafo G se mantiene en memoria en la forma:

GRAFO(NODO, SIG, ADY, PRINCIPIO, DEST, ENL)

Escribir un procedimiento que encuentre los grados de entrada GRADENT y los grados de salida GRADSAL de cada nodo de G.

Primero recorremos la lista de nodos, mediante el puntero PTR, para inicializar los arrays GRADENT y GRADSAL a cero. Entonces recorremos la lista de nodos, mediante el puntero PTRA, y para cada valor de PTRA, recorremos la lista de vecinos de NODO[PTRA], usando el puntero PTRB. Cada vez que se encuentra una arista, PTRA da la posición de su nodo inicial y DEST[PTRB] da la posición de su nodo terminal. De acuerdo con esto, cada arista actualizada los arrays GRADENT y GRADSAL tal como sigue:

GRADSAL[PTRA] : = GRADSAL[PTRA]+1

Y

GRADENT[DEST[PTRB]]: = GRADENT[DEST[PTRB]]+1

El procedimiento formalmente escrito es el siguiente:

Procedimiento P8.15 GRADO(NODO, SIG, ADY, PRINCIPIO, DEST, ENL, GRADENT, GRADASAL).

Este procedimiento encuentra el grado de entrada GRADENT y el de salida GRADSAL de cada nodo del grafo G en memoria.

1.-[Inicializa los arays GRADENT y GRADSAL].
   (a).- Hacer PTR:= PRINCIPIO.
   (b).- Repetir mientras PTR sea diferente de NULO: [Recorrer la lista de nodos].
      (i).- Hacer GRADENT[PTR] : = 0 y GRADSAL[PTR]:= 0.
      (ii).- Hacer PTR: = SIG[PTR].
   [Fin de Bucle].
2.- Hacer PTRA: = PRINCIPIO.
3.- Repetir pasos 4 a 6 mientras que PTRA se diferente a NULO:[recorrer lista de nodos]
4.- Hacer PTRB: = ADY[PTRA].
5.- Repetir mientras que PTRB sea diferente de NULO:[recorrer lista de vecinos] .
   (a).-Hacer GRADSAL[PTRA]:= GRADSAL[PTRA]+1 y GRADENT [DEST[PTRB]]+1.
   (b).- Hacer PTRB:= ENL [PTRB].
   [Fin del Bucle anterior que usa PTRB]
6.- Hacer PTRA:=SIG[PTRA].
[Fin del Bucle exterior del paso 3 que usa el puntero PTRA].
7.- Volver.

8.16 Suponga que G es un grafo finito no dirigido. Entonces G consiste en un número finito de componentes conexas disjuntas. Describir un algoritmo que encuentre el número NCOMP de componentes conexas de G. Mas aun, el algoritmo debe asignar un número de componente COMP(N) a cada nodo N de la misma componente conexa de G de tal forma que el rango de números de componentes vaya de 1 a NCOMP.

La idea general del algoritmo es la utilización de una búsqueda en anchura o en profundidad para encontrar todos los nodos N alcanzables desde un nodo inicial A y asignarles el mismo número de componente. El algoritmo es el siguiente:

Algoritmo P8.16 Encuentra las componentes conexas de un grafo no dirigido G.

1.- Inicialmente hacer COMP(N): = 0 para cada nodo N de G, e inicialmente hacer L:= 0
2.- Encontrar un nodo A tal que COMP(A) = 0, Si no existe tal nodo A, entonces hacer: NCOMP: = L y Salir.
3.- Encontrar todos los nodos N de G que son alcanzables desde A (mediante una búsqueda en anchura o en profundidad) y hacer COMP(N) = L para cada nodo de esos, N.
4.- Volver al paso 2.

## PROBLEMAS VARIADOS

8.17 Suponga que G es un grafo no dirigido con m nodos $v_1$, $v_2$,…$v$m y $n$ aristas $e_1$, $e_2$,…en. La matriz de incidencia de G es la matriz de m x n M=(m $ij$), donde

$$m\ ij = \begin{cases} 1 \text{ si el nodo } vi \text{ pertenece a la arista } ej \\ 0 \text{ de otro modo} \end{cases}$$

Encontrar la matriz de incidencia M del grafo G de la figura 8.29.

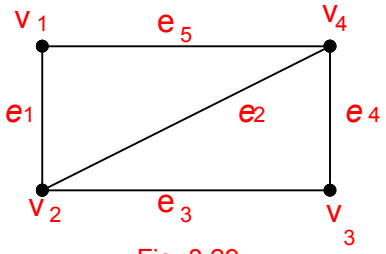



Fig :8.29

Como G tiene cuatro nodos y cinco aristas, M es una matriz de 4 x 5. se hace $m\ ij = 1$ si $vi$ pertenece a $ej$ Esto da la siguiente matriz:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

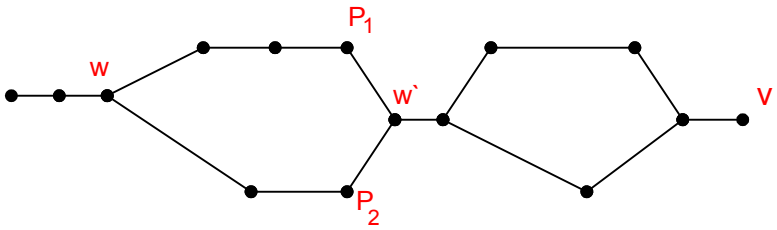



Fig:8.30

8.18 Suponga que u y v son dos nodos distinto s de un grafo no dirigido G. Probar que:

(a).-Si existe un camino P de $u$ a $v$, entonces existe un camino simple Q de $u$ a $v$.

(b).- Si existen dos caminos distintos P1 y P2 de $u$ a $v$, entonces G contiene un ciclo.

(a).- Suponga que P = ($v_0$, $v_1$,….$v_n$), donde $u$ = v0 y $v$= $v$n . Si $v$i = $v$j, entonces:

$$P'=(v_0, vi, vj+1,\ldots,vn)$$

Es un camino de $u$ a $v$ que es mas corto que P. Repitiendo este proceso, obtenemos un camino Q de $u$ a $v$ cuyos nodos son distintos. Así Q es un camino simple de $u$ a $v$.

(b).- Sea $w$ un nodo de P1 y P2 tal que los siguientes nodos en P1 y P2 son distintos. Sea $w$' el primer nodo siguiente a $w$ que esta tanto en P1 como en P2 (véase figura 8.30). Entonces los subcaminos de P1 y P2 entre $w$ y $w$' no tienen nodos en común excepto $w$ y $w$'; por tanto, estos dos subcaminos forman un ciclo.

8.19 Probar la proposición 8.2: Sea A la matriz de adyacencia de un grafo G. entonces $a_k$ ($i$ $j$), la entrada $ij$ de la matriz $A_k$, da el numero de caminos de longitud K de $v$i a $v$j

Se prueba por inducción en K. Note primero que un cambio de longitud 1 de $v$i a $v$j es precisamente la arista ($v$i, $v$j) . Por definición de la matriz de adyacencia A, $a_1$(i,j) igual a $a$ ij da el numero de aristas de $v$i a $v$j. Por tanto la proposición es cierta para K igual a 1

Suponga que K es mayor a 1. (Asuma que G tiene m nodos.) como a las potencias de A(k) igual a A(k-1) A,

$$a_k(i, j) = \sum_{s=1}^{m} a_{k-1}(i, s)\, a_1(s,j)$$

Por inducción, $a_{k-1}(i,s)$ da el numero de caminos de longitud K-1 de $v$i a $v$s, y $a_1$ ($s, j$) da el numero de caminos de longitud 1 de $v$s a $v$j. Asi, $a_{k-1}$ ($i, s$) $a_1$($s, j$) da el numero de caminos de longitud K de $v$i a $v$j, donde $v$s es el penúltimo nodo. Así todos los caminos de longitud K de $v$i a $v$j se pueden obtener sumando los productos $a_{k-1}$ ($i, s$) $a_1$($s, j$) para todos los s. Por tanto, $a_k$ ($i, j$) es el numero de caminos de longitud K de $v$i a $v$j . Así la proposición esta probada.

8.20 .- Suponga que G es un grafo finito no dirigido sin ciclos. Probar cada una de las siguientes proposiciones:

(a).- Si G tiene al menos una arista, entonces G tiene un nodo $v$ con grado 1.

(b).- Si G es conexo—de forma que G es un árbol— y si G tiene $m$ nodos, entonces G tiene $m$-1 aristas.

(c).- Si G tiene m nodos y m-1 aristas, entonces G es un árbol.

(a).- Sea P=($v_0$, $v_1$,…$v$n) un camino simple de longitud máxima. Suponga que grad($v_0$) es diferente a 1 y asuma que [$u$, $v_0$] es una arista y u diferente que $v_1$. Si $u$ = $v$i para i >1, entonces C= ($v$i ,..$v$0,..$v$i) es un ciclo. Si $u$ es diferente a $v$i, entonces P'=($u$, $v_0$,… $v$n) es un camino simple de longitud mayor que P. Cada caso lleva a una contradicción, por lo que grad($v_0$)= 1.

(b).- Se prueba por inducción en $m$. Suponga que $m$= 1. Entonces G consiste en un nodo aislado y G tiene $m$-1 =0 aristas. Por tanto el resultado es cierto para $m$=1. Suponga que $m$>1. Entonces G tiene un nodo $v$ tal que grad($v$)=1. Eliminando $v$ y su única arista [$v$, $v$'] del grado G se obtiene el grafo G'. Entonces G sigue siendo conexo y G es un árbol con $m$-1 nodos. Por inducción, G' tiene $m$-2 aristas. Por tanto, G tiene $m$-1 aristas. Por tanto el resultado es cierto.

(c).- Sean $T_1$, $T_2$,..$T_3$, las componentes conexas de G. Entonces cada T$i$ es un árbol. Por tanto, cada T$i$ tiene un nodo más que aristas. Pero G solo tiene un nodo más que aristas. Por tanto, s=1 y G es un árbol.

## PROBLEMAS SUPLEMENTARIOS.
## TERMINOLOGÍA DE GRAFOS

8.21 Considere el grafo no dirigido G de la figura 8.31. Encontrar : (a) todos los caminos simples de l nodo A al nodo H,(b) el diámetro de G, (c) el grado de cada nodo.

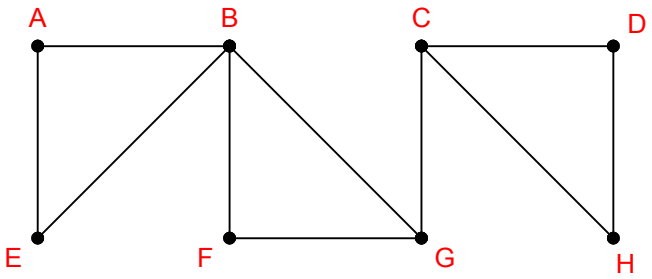



Fig. 8.31

8.22 ¿Cuales de los multigrafos de la figura 8.32 son (a) conexos, (b) libre de bucles (c) grafos?

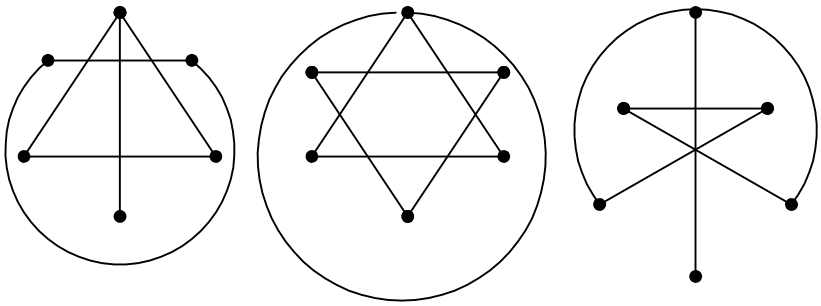



Fig. 8.32

8.23 Considere el grafo dirigido de la figura 8.33 (a) Encontrar el grado de entrada y de salida de cada nodo. (b) encontrar el número de caminos simples de $v_1$ a $v_4$. (c) Existen fuentes o sumideros?

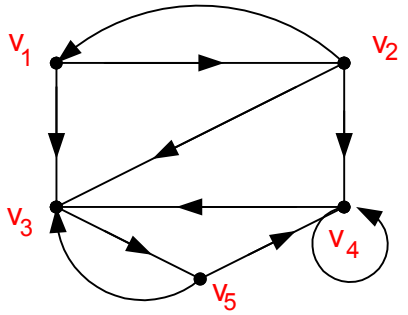



Fig.8.33

8.24 Dibujar todos los árboles (no similares) de cinco o menos nodos. (Hay ocho árboles así.)

8.25 Encontrar el número de árboles extendidos del grafo G de la figura 8.34.

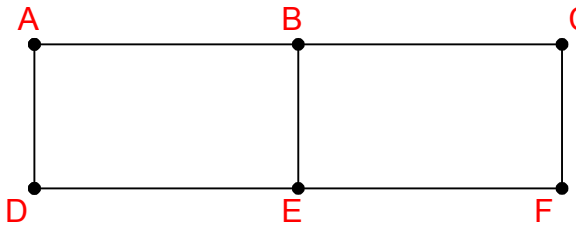



Fig. 8.34

## REPRESENTACIÓN SECUENCIAL DE GRAFOS ; GRAFOS CON PESO

8.26 Considere el grafo G de la figura 8.35 Suponga que los nodos se mantienen en memoria en un array DATOS de la forma siguiente:

DATOS : X, Y, Z, S, T

(a) Encontrar la matriz de adyacencia A de G. (b) Encontrar la matriz de caminos P de G . (c) ¿Es fuertemente conexo?

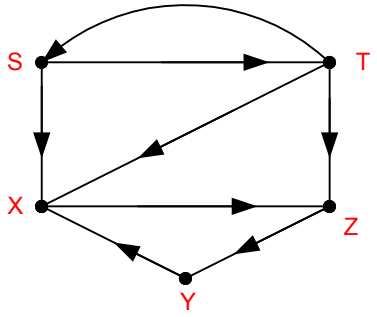



Fig.8.35

8.27 Considere el grafo con peso G de la figura 8.36. Suponga que los nodos se guardan en un array DATOS como sigue:

DATOS: X, Y, S, T

(a).- Encontrar la matriz de pesos W de G . (b) Encontrar la matriz de caminos mínimos mediante el algoritmo 8.2, algoritmo de Warshall.

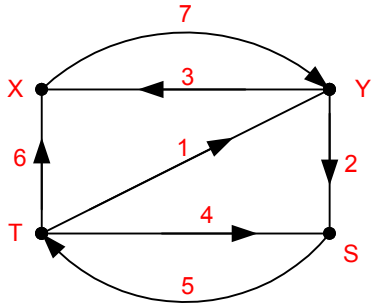



Fig.8.36

8.28 Encontrar un árbol extendido mínimo del grafo G de la figura 8.37

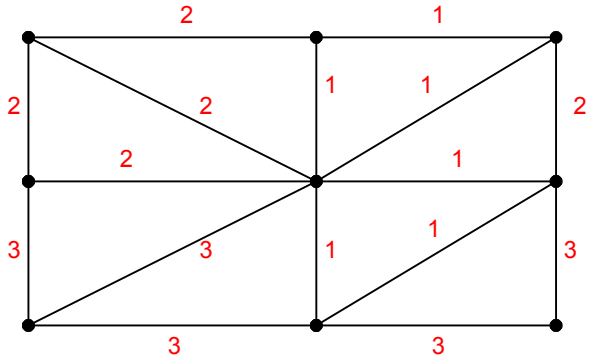



Fig.8.37

8.29.- La siguiente es la matriz de incidencia M de un grafo G no dirigido:

$$M = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

(Note que G tiene cinco nodos y ocho aristas). Dibujar G y encontrar su matriz de adyacencia A.

8.30 La siguiente es la matriz de adyacencia A de un grafo no dirigido G.

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

(Note que G tiene cinco nodos). Dibujar G y encontrar su matriz de incidencia M.

## REPRESENTACIÓN ENLAZADA DE GRAFOS

8.31 Suponga que un grafo G se guarda en memoria como sigue:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| NODO | A | | C | E | | D | | B |
| SIG | 4 | 0 | 8 | 0 | 7 | 3 | 2 | 1 |
| ADY | 6 | | 1 | 10 | | 2 | | 9 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

PRINCIPIO= 6, NDISP= 5

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| DEST | 8 | 8 | | 1 | 4 | 3 | 3 | | 6 | 3 |
| ENL | 5 | 7 | 8 | 0 | 0 | 0 | 0 | 0 | 4 | 0 |

ADISP=3

Dibujar el grafo G

8.32 Encontrar los cambios que ocurren en la representación enlazada del grafo G del problema 8.31 si se elimina la arista (C, E) y se inserta la arista (D,E).

8.33 Encontrar los cambios en la representación enlazada del grafo G del problema 8.31 si se insertan el nodo F y las aristas (E, F) y (F, D) en G.

8.34 Encontrar los cambios en la representación enlazada del grafo G del problema 8.31 si se elimina el nodo B de G,

Los problemas 8.35 a 8.38 se refieren a un grafo G que se mantiene en memoria mediante la siguiente representación enlazada:

GRAFO(NODO, SIG, ADY, PRINCIPIO, NDISP, DEST, ENL, ADISP)

8.35 Escribir un procedimiento para:

(a).- Imprimir la lista de sucesores de un nodo ND.
(b).- Imprimir la lista de predecesores de un nodo dado ND.

8.36 Escribir un procedimiento que determine si G es o no un grafo no dirigido.

8.37 Escribir un procedimiento que encuentre el numero de nodos M de G y luego la matriz de M x M de adyacencia A de G. (Los nodos están ordenados de acuerdo con su posición en la lista de nodos.)

8.38 Escribir un procedimiento que determine si hay fuentes o sumideros en G.

Los problemas 8.39 y 8.40 se refieren a un grafo con peso G que se mantiene en memoria con la siguiente representación enlazada:

GRAFO(NODO, SIG, ADY, PRINCIPIO, NDISP, PESO, DEST, ENL, ADISP)

8.39 Escribir un procedimiento que encuentre el camino mínimo desde un nodo dado NA hasta un nodo dado NB.

8.40 Escribir un procedimiento que encuentre el camino mas largo desde un nodo dado NA hasta un nodo dado NB.

## PROBLEMAS DE PROGRAMACIÓN

8.41 Suponga que un grafo G se determina por medio de un entero M, representando los nodos 1, 2, …, M y una lista de N pares ordenados, representando las aristas de G. escribir u procedimiento para:

(a).- Encontrar la matriz M x M de adyacencia A del grafo G.
(b).- Encontrar la matriz de caminos P del grafo G mediante la matriz de adyacencia A y el algoritmo de Warshall.

Probar cada uno de ellos con los siguientes datos:

(i).- M = 5; N= 8;(3, 4), (5, 3). (2, 4). (1, 5). (3, 2). (4, 2). (3, 1). (5, 1).
(ii).- M = 6; N= 10;(1, 6), (2, 1). (2, 3). (3, 5). (4, 5). (4, 2). (2, 6). (5, 3), (4, 3), (6, 4).

8.42 Suponga que un grafo con peso G se determina por medio de un entero M, representando los nodos1, 2,..M, y una lista de N tripletas ordenadas (ai, bi, wi) de enteros tales que el par (ai, bi) es una arista de G y wi es su peso. Escribir un procedimiento para:

(a).- Encontrar la matriz M x M de pesos W del grafo G.
(b).- Encontrar la matriz Q de caminos minimos entre los nodos, usando la matriz de pesos W y el algoritmote Warshall, algoritmo 8.2 .

Probarlos con los siguientes datos:

(i).- M =4; N=7; (1, 2, 5), (2, 4, 2), (3, 2, 3), (1, 1, 7), (4, 1, 4), (4, 3, 1),(Compararlo con el ejemplo 8.4)
(ii).-M=5; N=8; (3 5 3), (4, 1, 2), (5, 2, 2), (1, 5, 5), (1, 3, 1), (2, 4, 1), (3, 4, 4), (5, 4, 4),

8.43 Suponga que un grafo vacio G se guarda en memoria mediante la representación ´ enlazada

GRAFO(NODO, SIG, ADY, PRINCIPIO, NDISP, DEST, ENL, ADISP)

Asuma que NODO tiene sitio para ocho nodos y DEST para doce aristas. Escribir un programa que realice las siguientes operaciones sobre G:

(a).- Introducir los nodos A, B , C y D.
(b).- Introducir las aristas (A, B), (A, C), (C, B), (D, A), (B, D) Y (C, D).
(c).- Introducir los nodos E y F.
(d).- Introducir las aristas (B, E), (F, E), (D, F) y (F, B).
(e).- Eliminar las aristas (D, A) y (B, D).
(f).- Eliminar el nodo A.

Los problemas 8.44 a 8.48 se refieren a los datos de la figura 8.38

8.44 Escribir un procedimiento con entradas CIUDADA y CIUDADB que encuentre el número de vuelo y coste del vuelo de la ciudad A a la CIUDADB, si es que existe. Probar el procedimiento con (a) CIUDADA = Chicago, CIUDADB= Boston; (b) CIUDADA = Washington, CIUDADB= Denver, y (c) CIUDADA= Nueva Cork, CIUDAD B = Filadelfia.

8.45 Escribir un procedimiento con entradas CIUDADA y CIUDADB que encuentre un vuelo entre la ciudad A y la ciudad B con el menor numero de paradas y ademas encuentre su coste. Probar el procedimiento con (a) CIUDADA= Boston, CIUDADB =Houston; (b) CIUDADA= Denver, CIUDADB= Washington, y (c) CIUDADA = Nueva York, CIUDADB = Atlanta.

| | CIUDAD | IZQ | DER | ADY |
|---|---|---|---|---|
| 1 | Atlanta | 0 | 2 | 12 |
| 2 | Boston | 0 | 0 | 1 |
| 3 | Houston | 0 | 0 | 14 |
| 4 | Nueva York | 3 | 8 | 4 |
| 5 | | 6 | | |
| 6 | | 0 | | |
| 7 | Washington | 0 | 0 | 10 |
| 8 | Filadelfia | 0 | 7 | 6 |
| 9 | Denver | 10 | 4 | 8 |
| 10 | Chicago | 1 | 0 | 2 |

PRINCIPIO= 9 NDISP= 5

| | NUMERO | PRECIO | ORIG | DEST | ENL |
|---|---|---|---|---|---|
| 1 | 201 | 80 | 2 | 10 | 3 |
| 2 | 202 | 80 | 10 | 2 | 0 |
| 3 | 301 | 50 | 2 | 4 | 0 |
| 4 | 302 | 50 | 4 | 2 | 5 |
| 5 | 303 | 40 | 4 | 8 | 7 |
| 6 | 304 | 40 | 8 | 4 | 9 |
| 7 | 305 | 120 | 4 | 9 | 0 |
| 8 | 306 | 120 | 9 | 4 | 13 |
| 9 | 401 | 40 | 8 | 7 | 0 |
| 10 | 402 | 40 | 7 | 8 | 11 |
| 11 | 403 | 80 | 7 | 1 | 0 |
| 12 | 404 | 80 | 1 | 7 | 16 |
| 13 | 501 | 80 | 9 | 3 | 15 |
| 14 | 502 | 80 | 3 | 9 | 0 |
| 15 | 503 | 140 | 9 | 1 | 0 |
| 16 | 504 | 140 | 1 | 9 | 0 |
| 17 | | | | | 18 |
| 18 | | | | | 19 |
| 19 | | | | | 20 |
| 20 | | | | | 0 |

NUM=16 ADISP=17

Fig. 8.38

8.46 Escribir un procedimiento con entradas CIUDADA y CIUDADB que encuentre la forma mas barata de volar desde la ciudad A hasta la ciudad B y además encuentre el coste. Probar el procedimiento con los datos del problema 8.45(Compare los resultados.)

8.47 Escribir un procedimiento que elimine un registro del archivo, dado un número de vuelo NUMB, Pruebe el programa con (a) NUMB=503 y NUMB=504 y(b) NUMB=303 y NUMB=304.

8.48 Escribir un procedimiento que introduzca un registro de la forma:

(NUEVONUMB, NUEVOPRECIO, NUEVOORIGEN, NUEVODEST)

Pruebe el procedimiento con los siguientes datos:

(a).- NUEVONUMB=505, NUEVOPRECIO= 80, NUEVOORIGEN= Denver, NUEVODEST= Chicago
(b).- NUEVONUMB=601, NUEVOPRECIO=70, NUEVOORIGEN= Atlanta, NUEVODEST= Miami, NUEVONUMB=602, NUEVOPRECIO=70,NUEVOORIG= Miami, NUEVODEST= Atlanta).

(Observe que se ha de insertar una nueva ciudad en el árbol binario de búsqueda de ciudades).

8.49 Traduzca el algoritmo de ordenación topológica en un programa que ordene un grafo G. Asuma que G se introduce por su conjunto V de nodos y por su conjunto E de aristas. Pruebe el programa con los nodos A, B, C, D, X, Y, Z, S y las aristas:

(a) (A, Z), (S, Z), (X, D), (B, T), (C, B), (Y, X), (Z, X), (S, C), (Z, B).
(b) (A, Z), (D, Y), (A, X), (Y, B), (S, Y), (C, T), (X, S), (B, A), (C, S) y (X, T)
(c) (A, C), (B, Z), (Y, A), (Z, X), (D, Z), (A, S), (B, T), (Z, Y), (T, Y) y (X, A).

8.50 Escribir un programa que encuentre el número de componentes conexas de un grafo G desordenado y además asigne un número de componente a cada uno de sus nodos. . Asuma que G se introduce por su conjunto V de nodos y por su conjunto E de aristas (no dirigidas). Pruebe el programa con los nodos A, B, C, D, X, Y, Z, S y T y las aristas:

(a) [A, X], [B, T], [Y, C], [S, Z], [D, T], [A, S], [Z, A], [D, B] y [X, Z].
(b) [Z, C], [D, B], [A, X], [S, C], [D, T], [X, S], [Y, B], [T, B] y [S, Z].

## BIBLIOGRAFÍA


- Lipschutz, Seymour. "Estructura de Datos". Serie Schaum en Computación. McGraw-Hill.