DIVISIÓN DE ESTUDIOS DE POSGRADO
FACULTAD DE INGENIERÍA

ANÁLISIS DE ALGORITMOS DE REDES
Y
SU IMPLEMENTACION EN COMPUTADORA

SERGIO ESPINOSA FLORES

TESIS

PRESENTADA A LA DIVISIÓN DE ESTUDIOS DE
POSGRADO DE LA

FACULTAD DE INGENIERÍA

DE LA

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

COMO REQUISITO PARA OBTENER
EL GRADO DE

MAESTRO EN INGENIERÍA
(INVESTIGACIÓN DE OPERACIONES)

CIUDAD UNIVERSITARIA
MARZO, 1994.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

RESUMEN

El presente trabajo de tesis tiene la finalidad de crear un programa de cómputo que resuelva los siguientes problemas de redes: ruta más corta de un nodo a otro y de un nodo a los demas restantes de la red, árbol de expansión mínimo y máximo, flujo máximo y flujo a costo mínimo. Para lo cual se hace un análisis sobre las diversas estructuras de las redes, asi como de los algoritmos utilizados. La tesis consta de los siguientes capítulos:

i)Introducción
ii)Conceptos generales y representacion de redes
iii)Ruta más corta y árbol de expansión
iv)Flujo máximo y flujo a costo mínimo
v)Conclusiones y anexos (manual del programa)

además de un disco de 3.5" en el cual se encuentra el programa ejecutable REDES.EXE.

AGRADECIMIENTO

AGRADEZCO PROFUNDAMENTE
A TODA MI FAMILIA,
PROFESORES
Y AMIGOS,
QUE ME HICIERON LLEGAR
HASTA DONDE HOY.

ABRIL, 1994.

CONTENIDO

INTRODUCCION

Una de las áreas en matemáticas, con bastante aplicación a problemas reales , es la teoría de redes, con ellas sepueden representar infinidad de situaciones, por ejemplo, la representación de un conjunto de ciudades unidas por ciertas carreteras, o representar un conjunto de personas relacionadas (unidas) mediante una proposición, como podría ser "es amigo de". Es decir, una red una red puede ser definida en su forma más simple, como un par de conjuntos N y A, tal que N es un conjunto formado por objetos que llamados nodos y A, un subconjunto de NxN, que se denomina conjunto de arcos. Además, a cada arco se le puede asociar un valor llamado costo, peso, capacidad, etc. por ejemplo, pensemos en la red formada por un par de nodos "n" , "a", que representa a dos ciudades y la carretera que los une por el arco (n,a) entonces, un costo asociado a (n, a) podría ser la distancia que existe entre las dos ciudades , la cantidad de combustible que consume un vehículo al recorrer dicha carretera o quizás el tiempo que se hace al recorrer la distancia de "n" a "a", es decir, dependiendo del problema será el tipo de "costos".

Supongamos que se tiene una red G(N, A) con costos asociados a cada arco, es interesante saber cuando tienen respuesta las siguientes preguntas:

1.- ¿Existirá una trayectoria que nos una un nodo "a" a uno "b"?
2.- ¿De existir será, esta la única trayectoria?
3.- ¿Cuál será el costo por cubrir la trayectoria?
4.- ¿Será esta trayectoria la de costo mínimo o máximo?

Problemas asociados a este tipo de preguntas, a menudo se presentan en la vida diaria, como el el envío de señales digitales a un costo mínimo en una red de telecomunicaciones, que tiene que ver con el problema de flujo a costo mínimo. O el envío

máximo de artículos de un cierto lugar a otro, o envío de una cierta cantidad de artículos a un costo mínimo, que son conocidos como problemas de flujo máximo y flujo a costo mínimo respectivamente. Ejemplos de este tipo existen infinitamente por lo queel propósito del presente trabajo es el de crear un paquete computacional que resuelva la idea general de este tipo de problemas, basándose en algoritmos como son el de Dijkstra y PDM para resolver los problemas de ruta más corta y flujo a costo mínimo y los de Kruskal y Prim para resolver los problemas de árbol de expansión mínimo y máximo respectivamente, finalmente Malhotra para resolver el problema de flujo máximo, es decir implementarlos en un programa que de manera amigable nos permita encontrar la solución a estos problemas así como almacenar y recuperar los datos introducidos. Por otra parte, cabe mencionar que la representación de una red no es única y se deben tomar en cuenta ciertas características de ella, principalmente, como son el número de nodos y arcos pues una red que sea muy densa con respecto al número de nodos pero con solo unos cuantos arcos sería poco conveniente representarla con una matriz de costos o capacidades, la cual es definida mas adelante, sería mas conveniente utilizar una mejor representación para este tipo de red, ayudando esto a economizar tiempo en la ejecución del algoritmo.

DEFINICIONES BASICAS

Demos un repaso y algunas otras definiciones básicas de redes que se utilizarán más adelante.

Una *red* G(N,A) está determinada por dos conjuntos N, A de cardinalidad finita llamados, conjunto de nodos y conjunto de arcos respectivamente donde el conjunto N puede ser representado por $N=\{1,2,3, \ldots , n\}$ con $N \neq \emptyset$ y el conjunto A mediante $A = \{ (a,b) \mid a, b \in N \}$, es decir, $N \subset \mathbb{N}$ (donde $\mathbb{N}$ es el conjunto de los números naturales) mientras que $A \subset N \times N$.

Una red G(N,A,C) se llama red con costos si cada arco $(i,j) \in A$ tiene asignado un número $c_{ij} \in C \subset \mathbb{N}$.

Decimos que los *nodos* i , j son *adyacentes* si existe $(i,j) \in A$ y decimos que dos *arcos* (a,b) y (c,d) son *adyacentes* si tienen un nodo en común.

Una *trayectoria* *T* en una red G(N,A), de un nodo i_1 a uno i_n se define como una sucesión de arcos adyacentes, de la siguiente forma:

$T = \{i_1,(a_1,a_2), i_2,(a_2,a_3), \ldots , i_{n-1},(a_{n-1},a_n),i_n\}$ tal que i_1 nos representa el nodo orígen, e i_n nos representa el nodo destino y (a_j, a_{j+1}) nos representa la forma en que se recorrió el arco.

Una trayectoria *T* se dice positiva, si todos sus arcos se recorren en forma positiva.

Una trayectoria *T* se dice negativa, si todos sus arcos se recorren en forma negativa.

Una trayectoria *T* se dice mixta, si todos sus arcos se recorren sin restricciones de sentido.

Una trayectoria *T* se dice que tiene multiplicidades, si al menos uno de sus arcos se recorre dos o más veces.

Una trayectoria *T* se dice Elemental, si ninguno de sus arcos o nodos se recorre dos o más veces.

Una trayectoria *T* se dice Unitaria, si consiste solamente de dos nodos y un arco.

Una trayectoria *T* se dice Ciclo, si sucede que el nodo inicial es igual al nodo final.

Una trayectoria *T* se dice Ciclo Elemental, si al quitar el nodo inicial o el final, *T* resulta ser una trayectoria elemental.

Una red G(N,A) que tenga restricciones en la forma de como recorrer sus arcos se llama *Red Dirigida*, en caso de no tener restricciones simplemente se llama Red sin dirección.

Si además hablamos de una red G(N,A) con costos, entonces el costo por recorrer una trayectoria será igual a la suma de los costos de cada arco. Una red G(N,A) es llamada *conexa* o *conectada*, si para cualesquier par de nodos i, j, siempre se puede encontrar una trayectoria que los una (no importando en qué sentido se recorran los arcos).

Decimos que los arcos j_1 , j_2 están en paralelo si el nodo inicial de j_1 es igual al nodo inicial de j_2 y el nodo final de j_1 es igual al nodo final de j_2.

Finalmete, una red G(N,A) que sea conexa y que no tenga ciclos, se le conoce como *árbol*.

REPRESENTACION DE UNA RED

Antes de presentar el primer algoritmo, para determinar la trayectoria con costo mínimo de un nodo "o" a uno "d", en una red G(N,A,C) veamos algunas representaciones de ellas y comparemos las ventajas de ellas si fueran implementadas en una computadora. Una primer representación, la más común y que de hecho surge de la realidad, es mediante un dibujo, por ejemplo el siguiente:

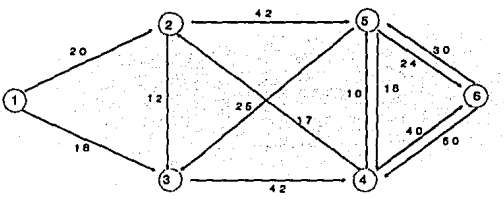


figura 1

Donde los pequeños círculos representan (son) los nodos, cada número dentro de ellos es el número del nodo y las flechas representan (son) los arcos, cada número sobre ellos es el costo asociado a cada uno, aunque no ha sido mencionado, es importante decir, que la punta de la flecha indica la restricción del sentido en el que se puede recorrer el arco.

REPRESENTACION MATRICIAL

La red de la figura anterior puede representarse mediante la matriz **C**, llamada matriz de costos o de pesos, donde la entrada **C[i,j]** representa el costo (peso) por recorrer el arco del nodo i al nodo j ($i \neq j$), en el caso de que $i = j$, se asume que el costo es cero, pues el gasto es nulo por ir a donde ya se esta. Por último si $i \neq j$ y no existe ningún arco que los una la entrada C[i,j] de la matriz de costos podrá ser cualquier número entero a manera que no sean tomados en cuenta durante la ejecución de cualquiera de los algoritmos y por lo tanto los resultados finales no se vean alterados. Por ejemplo puede ser un

número suficientemente grande si lo que nos interesa tomar en cuenta son los arcos con costo mínimo. O puede ser cero si lo que nos interesa es tomar arcos con capacidades positivas, incluso un número negativo si asumimos que la red solo puede contener arcos con costos positivos.

La representación de la red anteriorqueda de la siguiente manera donde * denota la inexistencia de ese arco y toma algún valor de los convenidos previamente:

nodos ↓ / ↘	1	2	3	4	5	6
1	0	20	18	*	*	*
2	*	0	12	*	42	*
3	*	*	0	42	*	*
4	*	17	*	0	10	40
5	*	*	25	18	0	24
6	*	*	*	50	30	0

REPRESENTACION POR TRES VECTORES

Este es un caso particular de la representación matricial sólo que nos olvidamos de las entradas en la diagonal de la matriz y de los arcos que no existen, es decir, en este caso se toma en cuenta el número de arcos y no de nodos como en la representación maricial, resumiendose en tres vectores toda la información de la red, denotados por "*o*" , "*d*" , "*c*", todos de orden m, donde m es el número de arcos de la red (m=13 para el ejemplo anterior) donde "*o*" es el vector de nodos origen, "*d*" es el vector de nodos destino y "*c*" el vector de costos asociados a cada arco, entonces podemos representar la misma red de la siguiente forma:

"*o*" = (1 , 1 , 2 , 2 , 3 , 4 , 4 , 4 , 5 , 5 , 5 , 6 , 6)

"*d*" = (2 , 3 , 3 , 5 , 4 , 2 , 5 , 6 , 3 , 4 , 6 , 4 , 5)

"*c*" = (20, 18, 12, 42, 42, 17, 10, 40, 25, 18, 24, 50, 30)

Donde, si hacemos una biyección de "o" a "d" y de "d" a "c", estaremos formando ternas de la forma (o(j), d(j), c(j)) que querrán decir que existe un arco con nodo inicial o(j), nodo final d(j) y costo asociado c(j), con j = 1,2,3, ... , m. con m = número de arcos. Podemos pensar cada entrada ya sea de la matriz o de los vectores como la "palabra" que se almacenará en memoria,

entonces, es claro ver , que para la matriz se necesitan almacenar n*n nodos palabras (n=número de nodos), mientras que para los vectores, sólo 3*m palabras, en consecuencia, dependiendo de los valores de n y m, nos convendrá usar la representación matricial o la vectorial, es decir, si tenemos una red densa en cuanto a número de arcoslo más adecuado será utilizar la representanción matricial, pues existen pocas entradas iguales a cero, mientras que si la red tiene pocos arcos es preferible utilizar la representación vectorial, además, también se debe fijar uno en el tipo de algoritmo, pues si durante la ejecución se va a necesitar de incertar arcos, entonces es preferible utilizar la forma vectorial en su variante de listas ligadas, que es la que se muestra a continuación.

LISTAS LIGADAS

Esta representación consiste en ubicar los nodos que están conectados con un cierto nodo orígen "o", enlazandose entre ellos, haciendo ésto para cualquier nodo "o" de la red.
La representación para la red de la figura 1.0 es de la siguiente manera:

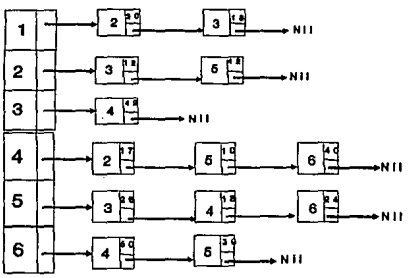


figura 2

Usando este tipo de estructura almacenaríamos primero, n-palabras + 3m-palabras, pues por cada bloque necesitamos tres campos, uno para el nodo destino, otro para el costo del arco asociado y otro para el apuntador hacia el siguiente nodo con orígen común "o".

PILA

Por último, veamos una representación parecida a la anterior, esta estructura se establece mediante el uso de pilas, de la siguiente manera: Sea G una red, entonces en una pila P_1 se almacenan los n nodos de la red G, siendo un registro por cada elemento de la red, es decir, de la pila P_1, cada registro tiene dos campos, uno para almacenar el número de nodo n_1 y otro para guardar la dirección de un elemento en la otra pila P_2, donde se almacena el primer nodo adyacente al n_1 y los elementos que estan por debajo de este elemento y por arriba de algún otro elemento en P_2 direccionado por otro elemento de P_1 son los demás nodos ayacentes a este nodo n_1 , este hecho se repite tantas veces como elementos (nodos) tenga la pila P_1 (red) , la representación en pilas de la red de la fig 1.0 que es la siguiente:

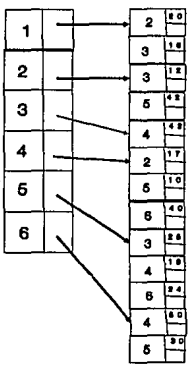


figura 3

RUTA MAS CORTA

Supongamos que de cierta ciudad necesitamos trasladarnos a otra pasando por ciudades intermedias y puesto que tenemos varias opciones para hacerlo nos interesa tomar una ruta en la que el costo sea mínimo, este problema lo podemos pensar en términos de una red donde los nodos son las estaciones (o poblaciones) donde cambiamos de camino, los arcos son las carreteras que unen a cada poblado y los costos asociados a cada arco puede ser pasaje, consumo de gasolina, tiempo, etc. El problema se reduce a encontrar una trayectoria T que nos una las dos ciudades y tenga un costo mínimo respecto a las demas. Un algoritmo que resuelve este tipo de problemas es el que se muestra a continuación y fue implementado por Dijkstra (1959).

ALGORITMO DE DIJKSTRA

El propósito del presente algoritmo es el de encontrar una trayectoria que una un cierto nodo origen "i" a otro nodo destino "j", de tal manera que el costo de la misma sea mínimo. La idea del algoritmo es la siguiente, supongamos que se tiene una red G con n nodos, la cual, como ya vimos, la podemos representar por una matriz de costos W[i, j], y deseamos encontrar el camino más corto de un nodo "i" a uno "j", entonces lo que hace Dijkstra es tomar todos los nodos adyacentes a "i" y comparar los costos de los arcos asociados, tomando el nodo con menor costo el cual se almacena en un cierto vector, llamado vector distancia, al hacer ésto, al nodo escogido se le "etiqueta" permanentemente (e.d. al hacer esto uno esta afirmando que que no existe ninguna trayectoria del nodo origen a este nodoque sea mas corta, por esta razón se pide que la red no tenga costos negativos) tal vector distancia, tiene dimensión igual al número de nodos de la red y es de la forma $(d_1, d_2, \ldots, d_n)$ donde d_i en la k-ésima iteración es la distancia recorrida (el costo acumulado) hasta alcanzar el nodo i a través de la trayectoria de costo mínimo. Se hace uso de un vector que se llamado predecesor, en el cual la entrada j en la i-ésima iteración denota el predecesor al nodo j a través de la trayectoria de costo mínimo, el algoritmo es el siguiente:

ALGORITMO DIJKSTRA

```
BEGIN
 FOR TODA V DESDE 1 HATA N DO
  BEGIN
   DISTANCIA[V] := INFINITO;  (EN ESTE BLOQUE SE INICIALIZAN LOS
   FINAL[V]  :=  FALSO;        VECTORES DISTANCIA, FINAL  Y
   PREDECESOR[V] := -1;        PREDECESOR DE V)
  END;
   DISTANCIA [S] :=0;     (DONDE S ES EL NODO ORIGEN)
   FINAL[S] := TRUE;      (True es la etiqueta que nos permite)
   PATH:=TRUE;    (saber que ya no existe otra trayectoria menor)
   RECENT:=S;           (RECENT ES UNA VARIABLE QUE NOS SIRVE PARA
                      IDETIFICAR QUIEN ES EL ULTIMO NODO ETIQUETADO
                      PERMANENTEMENTE)
  WHILE FIANAL(T) = FALSE DO
   BEGIN
    FOR CADA INMEDIATO SUCESOR V DE RECENT Y FINAL(V) <> FALSE DO
            (MOMENTO EN EL QUE SE COMPARAN LOS COSTOS DE LOS
             ARCOS ADYACENTES A V)
     BEGIN
      NUEVA ETIQUETA := DISTANCIA(RECENT) + W[RECENT,V];
      IF NUEVAETIQUETA < DISTANCIA(V) THEN (COMPARACION DE COSTOS
       BEGIN                                CON DISTANCIA DE V)
        DISTANCIA(V):=NUEVAETIQUETA;
        PREDECESOR(V):=RECENT;
       END
     END
    SEA Y EL NODO CON ETIQUETA TEMPORAL MAS PEQUEÑA
    FINAL(Y):=TRUE;             (ELECCION DEL COSTO MINIMO)
    RECENT:=Y;
   END.
```

Es claro que los datos de entrada, serán la matriz de costos W, el nodo origen y el nodo destino, aunque existe un problema para la matriz W, pues si no existe arco que una un nodo "k" a uno "i", la pregunta sería ¿qué valor hay que dar a la entrada W[k,i]?, si ponemos W[k,i] = 0, es como si existiera el arco (k,i) con costo cero y cuando se calcule Dijkstra sobre el nodo "k", el arco con menor costo sería el nodo "i" lo cual no es cierto pues no existe, así que para no tomar en cuenta este arco (puesto que estamos hablando de costos mínimos), sería deseable asignar el valor de infinito a la entrada W[k,i], y basta sólo con tomar un valor que sea lo suficientemente grande a cualquier arco de la red, dado lo que dijimos, este valor se llamará INF dentro del algoritmo de Dijkstra.

Los datos de salida serán en un principio, el vector DIST que nos da la información de los costos que se cubren del nodo origen "i"

a los demás n-1 nodos (suponiendo la red es de n nodos) además conocemos la trayectoria asociada a cada uno de esos costos mínimos. Por ejemplo, introducir los datos de la red de la fig. 1 del algoritmo de Dijkstra implementado en Pascal se generaron los siguientes resultados.

Datos de entrada:

$$C = \begin{pmatrix} 0 & 20 & 18 & * & * & * \\ * & 0 & 12 & * & * & * \\ * & * & 0 & 42 & * & * \\ * & 17 & * & 0 & 10 & 40 \\ * & * & 25 & 18 & 0 & 24 \\ * & * & * & 50 & 30 & 0 \end{pmatrix}$$

NODO ORIGEN = 1;
NODO DESTINO = 6;
* = 3000;

Datos de salida:

FINAL = (TRUE TRUE FALSE FALSE TRUE TRUE)
DISTANCIA = (0, 20, 18, 60, 62, 86);
PREDECESOR = (-1, 1, 1, 3, 2, 5);

Donde la primer entrada del vector distancia, en donde hay un 0, nos indica que el costo por recorrer del nodo origen al mismo nodo orígen, es de cero, la segunda entrada nos indica que el costo mínimo de ir del nodo origen al nodo dos es de 20,la tercera entrada indica que el costo mínimo de ir del nodo origen al tres es de 18 y así sucesivamente hasta alcanzar el nodo destino (seis) cuyo costo mínimo es de 86 unidades puesto que la entrada correspondiente al nodo seis en el vector final es true, lo cual significa que las demás trayectorias han sido revisadas escogiendose la de menor costo. Recordemos que la manera de como uno va trazando la trayectoria de un nodo origen "o" a otro nodo destino "d" es a partir de analizar todos los nodos adyacentes a él. A todos ellos los etiquetamos temporalmente con sus costos y una vez hecho ésto escogemos el nodo con menor etiqueta temoral para, en ese momento, hacerla permanente, siendo el arco (o, t_1) el primer arco sobre la trayectoria de "o" a "d" con costo mínimo y por ende, de todas las trayectorias que existan de "o" a t_1, el arco (o, t_1) forma la de menor costo, entonces por cada iteración que haga el algoritmo se etiqueta un nodo de manera permanente, supongamos que se tiene el siguiente conjunto:

$E = \{o, t_1.t_2,\ldots\ t_m\}$ de nodos etiquetados permanentemente, entonces lo que se afirma es que el costo de recorrer la trayectoria :

$$T = \{o,(o,t_1),\ t_1,(t_1,t_2),\ \ldots\ ,\ t_{m-1},(t_{m-1},t_m),\ t_m\}$$

tiene un costo mínimo, pues en caso contrario, existiría una trayectoria (t_i,t_{i+1}) la cual su costo no fuera mínimo y su etiqueta fuera permanente, lo cual contradice la manera de como fue escogido por el algoritmo. La trayectoria que describe el vector **PREDECESOR = (-1, 1, 1, 3, 2, 5)** recorrida negativamente es la siguiente T = {6, (5,6); 5, (2,5); 2, (1,2); 1} el -1 que aparece, indica que el nodo 1 no tiene predecesor. Cabe mencionar que las primeras restricciones que se notan en dicho algoritmo son, en primer lugar el gasto de memoria para almacenar la matriz de costos W[i,j], que es directamente proporcional al orden de la matriz, y en segundo lugar, que dicho algoritmo no involucra costos negativos. De existir costos negativos este algoritmo no podría calcular la trayectoria con costo mínimo pues una vez que se ha etiquetado un nodo de modo permanente este ya no puede ser modificado, veamos ésto de una manera más objetiva, es decir, calculemos la trayectoria con costo mínimo del nodo 1 al 6 aplicando Dijkstra a la red de la figura 1.0, sólo que ahora con algunos costos negativos.
La red sería la siguiente:

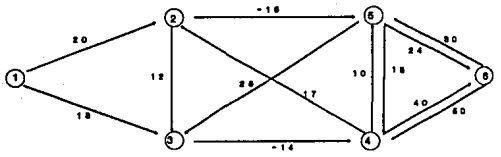


figura 4

Pongamos nuestras tres variables vector y veamos como se modifican a medida que va iterando el algoritmo.
nodo origen s = 1 y nodo destino t = 6.

dist						final						pred						recent					
1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
*	*	*	*	*	*	f	f	f	f	f	f	-1	-1	-1	-1	-1	-1	1					
0	*	*	*	*	*	t	f	f	f	f	f		1	1						3			
0	20	*	*	*	*	t	f	t	f	f	f				3						4		
0	20	18	*	*	*	t	f	t	f	f	f					4	4					5	
0	20	18	4	*	*	t	f	t	t	f	f												
0	20	18	4	14	*	t	f	t	t	t	f								2				
0	20	18	4	14	44	t	t	t	t	t	f												6
0	20	18	4	14	44	t	t	t	t	t	t												6

* = infinito

El algoritmo ha terminado de revisar toda la red hasta alcanzar el nodo destino t = 6, donde el vector dist nos dice que existe una trayectoria de s a t con un costo mínimo de 44 unidades, sin embargo observando la red, podemos ver que ésto no es cierto pues la trayectoria T = {2,(1,2) , 2,(2,5), 3,(5,6) , 6} tiene un costo de 44 unidades existiendo también la trayectoria T = {1,(1,2) , 2,(2,5) , 3,(5,6), 6} con un costo de 34 unidades que es aún menor que la que calculó el algoritmo. El problema fué que los nodos etiquetados permanentemente ya no pueden ser cambiados y es por eso que no se analizaron otras posibilidades. En la segunda mitad de la década de los cincuentas, un algoritmo fué propuesto por Bellman y Moore , que parcialmente resuelve el problema. Consiste en no hacer permanentes las etiquetas de los nodos hasta estar seguro que ya no existe ninguna opción por recorrer, no fue sino hasta los ochentas, cuando D'esopo y Pape implementaron y optimizaron este algoritmo por medio de una rutina para tener más rapidez al accesar los nodos a revisar. En esencia, este algoritmo es igual al de Dijkstra sólo con la opción de etiquetar de manera permanente los nodos hasta el final de la ejecución del algoritmo. Este algoritmo conocido como algoritmo PDM por Pape, D'Esopo y Moore, es el que a continuación mostramos.

ALGORITMO PDM

```
BEGIN
 FOR TODA v ∈ V DO
 BEGIN
  A DISTANCIA DE v ASIGNALE INFINITO;
  A CADA PREDECESOR DE v ASIGNALE -1;
 END;
```

```
  A DISTANCIA DEL NODO ORIGEN s ASIGNALE CERO;
  INICIALIZA Q CON EL NODO ORIGEN s;
  A CABEZA ASIGNALE s;
 WHILE Q SEA DISTINTO DEL VACIO DO
  BEGIN
   BORRA EL NODO CABEZA U DE Q;
   FOR CADA ARCO (u,v) QUE EMPIECE EN u DO
    BEGIN
     NUEVA_ETIQUETA := DISTANCIA DE u + W[u,v];
     IF NUEVA_ETIQUETA < DISTANCIA DE v THEN
      BEGIN
       DISTANCIA DE v := NUEVA_ETIQUETA;
       PREDECESOR DE v := u;
       IF v NUNCA ESTUVO EN Q THEN
        COLOCA v A LA COLA DE Q
       ELSE
        IF v HA ESTADO EN Q PERO YA NO ESTA THEN
         COLOCA v A LA CABEZA DE Q
      END
    END
  END
END.
```

Veamos como trabaja este procedimiento con la red G de la fig 2.0, los siguientes arreglos nos denotan como van cambiando los vectores dist, pred y Q a cada iteración.

Nodo origen s = 1

dist						Q		pred					
1	2	3	4	5	6	cola	cabeza	1	2	3	4	5	6
*	*	*	*	*	*			-1	-1	-1	-1	-1	-1
0	*	*	*	*	*		1	-1	1	-1	-1	-1	-1
0	20	*	*	*	*	2		-1	1	1	-1	-1	-1
0	20	18	*	*	*	3 2		-1	1	1	-1	2	-1
0	20	18	*	4	*	5 3		-1	1	1	3	2	-1
0	20	18	4	4	*	4 5		-1	1	1	3	2	5
0	20	18	4	4	34	6 4		-1	1	1	3	2	5
0	20	18	4	4	34	6		-1	1	1	3	2	5

* = infinito

El vector final dist = (0, 20, 18, 4, 4, 34) nos indica los costos mínimos de ir del nodo orígen 1 a todos los demás nodos y el vector pred = (-1, 1, 1, 3, 2, 5) nos indica la trayectoria que se siguio (de la misma manera que con Dijkstra) , por ejemplo en particular nos interesa saber la trayectoria (los nodos que se siguieron) con costo mínimo 34 del nodo orígen 1 al nodo destino 6. Entonces nos fijamos en la entrada seis del vector pred, puesto que seis es el nodo destino y pred(6) es su predecesor a lo largo de la trayectoria, puesto que pred(6) = 5 nos fijamos

ahora en el predecesor del nodo cinco, que es pred(5) = 2 y a su vez pred(2) = 1 y como uno es el nodo origen, nos detenemos ahí. Es decir la trayectoria que se siguió fue

$$T = \{ J_1(1,2),\ J_2(2,5),\ J_3(5,6) \}$$

es decir, del nodo uno caminamos al nodo 2, del nodo dos al cinco y del nodo cinco al seis, pues :

{ 6 , pred(6), pred(pred(6)), pred(pred(pred(6))) }
{ 6 , 5 , pred(5), pred(pred(5)) }
{ 6 , 5 , 2 , pred(2) }
{ 6 , 5 , 2 , 1 }

Sin embargo este algoritmo tiene un detalle, el cual es, que si una red tiene un ciclo con costo negativo, el algoritmo PDM caería en un loop, es decir, se seguiría optimizando indefinidamente.

En la siguiente red G de tres nodos, supongamos que queremos ir del nodo uno al dos, entonces si tomamos el arco (1,2) su costo sería de tres unidades, pero no sería el mínimo, pues si completamos un ciclo a partir del nodo dos, el costo de ir de uno a tres sería de menos diez unidades y si completamos otro ciclo, su costo disminuiría a menos veinte unidades, en general, si completamos t ciclos el costo por ir de uno a dos sería de t*(-10) y como no hay restricción bajo t, nunca acabaríamos.

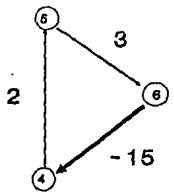


figura 5

ARBOL DE EXPANSION

Recordemos que un árbol es una gráfica conexa que no acepta ciclos, bien dada un a red G(N,A) podemos asociarle a esta distintos árboles, por ejemplo, para la siguiente red

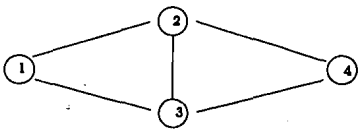


figura 6

podemos asociarle los siguientes árboles

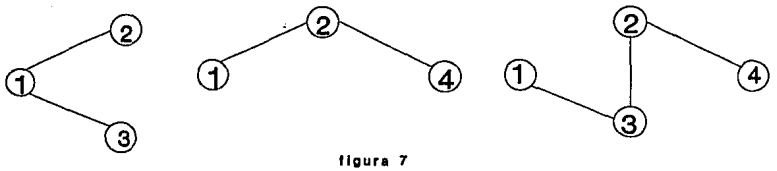


figura 7

es decir dada una red G(N,A) podemos tomar un subconjunto N_1 de N y un subconjunto A_1 de A de manera que al formar la gráfica $G_1(N_1,A_1)$ esta resulte ser un árbol, si además pedimos que $N_1 = N$, entonces diremos que $G_1(N_1,A_1)$ es un árbol de expansión, definamos ésto más formalmente.

Sea G(N,A) una red conexa sin dirección, un árbol de expansión $T(N_1,A_1)$ asociado a la red G(N,A) es un árbol en el que $N_1 = N$ y $A \subseteq A_1$.

Algunos árboles de expansión asociados a la red de la figura anterior son:

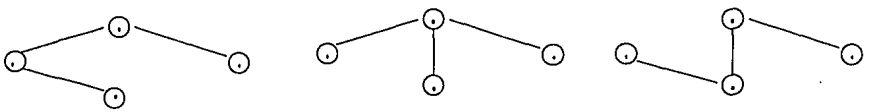


figura 8

Una vez definido lo que es un árbol de expansión vale la pena mostrar la siguiente observación.
Sea G(N,A) una red conexa y sea $T(N_1,A_2)$ un árbol de expansión asociado a G, entonces se cumple:
i) $\forall\ n,m \in N_1$ existe una única trayectoria de n a m.
ii) si $(n,m) \in A - A_1$ entonces la gráfica $G(N , A_1 \cup \{(n,m)\})$ tiene un ciclo único.

Dada una red G(N,A,C) con costos asociados a cada arco, sería interesante ahora no preguntarse por la trayectoria de costo mínimo, sino por el árbol de expansión asociado a G(N,A,C) que tenga costo mínimo, es decir, de todos los árboles de expansión asociados a G(N,A,C) escoger el más "barato", un árbol de expansión que reuna esta característica se le conoce como árbol de expansión mínimo y al problema de encontrarlo, dada una red, se le conoce como el problema del árbol de expansión mínimo.

Definición Sea G(N,A,C) una red sin dirección y con costos asociados a sus arcos, el árbol de expansión mínimo asociado a la red G(N,A,C) es un árbol de expansión tal que tiene costo mínimo con respecto a los demás árboles de expansión asociados a la red G(N,A,C).
El problema del árbol de expansión mínimo surge cuando se quieren resolver problemas como el conectar n puntos (ciudades) utilizando un mínimo de material, por ejemplo al tender cable de energía eléctrica, no hay que confundirse con el algoritmo PDM que calcula el costo mínimo de ir de un cierto punto a los demás n-1 nodos de la red. J.B. Kruskal [1956] propuso un algoritmo bastante sencillo que permite calcular el mínimo árbol de expansión de una red G(N,A,C) sin dirección. El algoritmo trabaja de la siguiente manera, sea G(N,A,C) como arriba, primero clasificamos los arcos de menor a mayor con respecto a sus costos, entonces lo que se va haciendo es ir tomando de uno en uno a partir del que tiene menor costo y así sucesivamente, no importando que no sean adyacentes, lo único que se debe tener cuidado es de no formar ciclos. Veamos ésto con el siguiente ejemplo.

Pensemos en la red G(N,A,C)

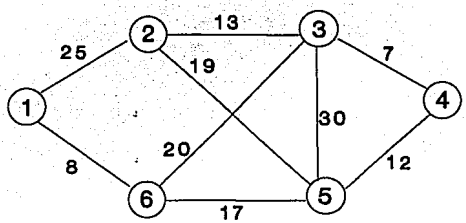


figura 9

busquemos el mínimo árbol de expansión asociado a G(N,A,C) y formemos seis conjuntos de nodos-arcos de la siguiente manera.

{1}, {2}, {3}, {4}, {5}, {6}

Iteración 1) Tomemos el arco con costo más pequeño J_1= (3, 4) entonces nuestros conjuntos se transforman en:

{1}, {2}, {3,(3,4)}, {5}, {6}

Iteración 2) El siguiente arco con costo más pequeño es J_2= (1,6) entonces los conjuntos se transforman en:

{1, (1,6)}, {2}, {3,(3,4)}, {5}

Iteración 3) El siguiente arco con costo más pequeño es J_3= (4,5) entonces los conjuntos se transforman en:

{1, (1,6)}, {2}, {3,(3,4), (4,5)}

Iteración 4) El siguiente arco con costo más pequeño es J_4= (2,3) entonces los conjuntos se transforman en:

{1, (1,6)}, {2,(2,3), (3,4), (4,5)}

Iteración 5) El siguiente arco con costo más pequeño es J_5= (2,5) notemos que los nodos terminales de este arco pertenecen a un mismo conjunto, entonces lo desechamos pues nos forma un ciclo.

El siguiente arco que le sigue en valor es el arco J_6= (5,6) entonces nuestros conjuntos se transforman en:

{1, (1,6), (2,3), (3,4), (4,5), (5,6)}

y puesto que nuestra red consta de seis nodos necesitamos solamente cinco arcos para formar nuestro mínimo árbol de expansión que se muestra a continuación.

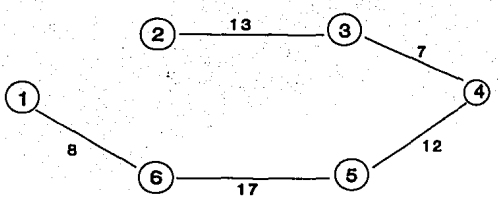


figura 10

Notemos que un mínimo árbol de expansión no es único, es decir, una red G(N,A,C) puede admitir más de uno, pues pueden existir arcos distintos con costos iguales y entonces se puede ser indiferente en escoger cualquiera de ellos, siempre y cuando no se este formando un ciclo.

ALGORITMO DE KRUSKAL
PARA ENCONTRAR EL MINIMO ARBOL DE EXPASION

INICIALIZACION

begin

Sea el arreglo formado por conjuntos unitarios cuyos elementos son los nodos de la red.

Inicial := {{n_1}, {n_2}, {n_3}, {n_4}, {n_5}, {n_6}}

Sea Pila un arreglo formado por los m arcos de la red en forma decreciente con respecto a sus costos. {de hecho ésto es un árbol}

cuentarc := 0; {*variable que cuenta cuantos arcos han sido examinados*}

cuentarbol := 0; {* variable que cuenta cuantos arcos han entrado al árbol de expansión*}

árbol := 0;

while cuentarbol < (n-1) **and** cuentarc < m **do**

begin

e := arco(u,v) más chico de Pila;

```
cuentarc := cuentarc + 1;
borra e de la pila;
actualiza Pila;
r1 := Encuentra(u);      {* Encuentra es un procedimiento que
r2 := Encuentra(v);        encuentra a que conjunto de Inicial
 if r1 ≠ r2 then         pertenece *}
  begin                    {* r1 ≠ r2 condición para que no se
   T := T ∪ {e}           formen ciclos *}
   cuentarbol := cuentarbol + 1;
   Union(r1, r2)
  end
 end;
if cuentarbol < (n-1) then
 writeln('red disconexa');
end;
```

ARBOL DE EXPANSION MAXIMO

Otro algoritmo que se presenta aquí es el algoritmo conocido como algoritmo Prim (debido a R.C. Prim [1957]), este algoritmo trabaja de la siguiente manera, se toma un nodo arbitrario n_1 y se fija uno en todos los arcos incidentes a n_1 eligiendo el de mayor costo, entonces se tienen ahora dos nodos n_1 y n_2 ,se repite lo anterior sólo que para los dos nodos y así sucesivamente, teniendo cuidado de no formar ciclos. Veamos como trabaja Prim con la siguiente red:

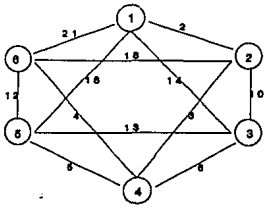


figura 11

Inicialización) Nodo selecionado $n_1 = 1$

Iteración 1) Nodos posibles a escoger 3 y 2, se escoge el arco (1,6) por tener mayor costo, entonces el árbol de expansión máximo se inicializa con el arco (1,6) , e.d. T = {(1,6)}

Iteración 2) nodos posibles a escoger 2, 3, 4, 5, escogiendo el arco (1,5) por tener mayor costo a los demás, entonces T = {(1,6), (1,5)} así sucesivamente hasta que se obtienen los cinco arcos que forman el árbol de expansión máximo, quedando T de la siguiente manera T = {(1,6), (1,5), (2,6), (1,3), (3,4)}

Tanto Kruskal como Prim nos permiten encontrar un árbol de expansión mínimo o máximo , la unica diferencia es que la selección de los arcos se hace en base al de menor o mayor costo o peso según se desée maximizar o minimizar, veamos ésto con el algoritmo Prim.

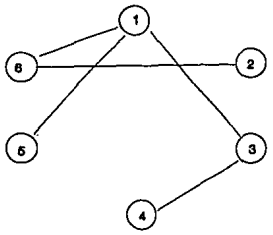


figura 12
Máximo árbol de expansión
de la red anterior.

Algoritmo Prim

Sea s un nodo arbitrario

```
near(s):=0;
for cada nodo i ≠ j do
 begin
  near(i)=s;
  dist(i):=  W[i,j];
 end
```

V_T := {S}; {En V_T se van almacenando los nodos del árbol de expansión máximo}

E_T := ø {En E_T se van almacenando los arcos del árbol de expansión máximo}

```
while  |V_T| < n  do        {Dentro de este While, se hacen las
                            iteraciones necesarias para la selección
begin                           de los arcos del árbol}
 v:=w;          {Donde w ∈ V - V_T  y w es el nodo que forma el arco
                máxima capacidad incidente en s}
```

```
 if dist(u) ≥ inf then
  writeln('La gráfica es disconexa');
  E_T := E_T ∪ ({u,near(u)});
  V_T := V_T  ∪ {u};
  for x ∈ (V - V_T) do
   if W[u,x] > dist(x) then          {Condición que checa si el
    begin                          arco elegido cumple con tomar
     dist(x):= W[u,x];             la mínima trayectoria}
     near(x):=u;
    end
end;    {While}
```

En el programa ejecutable de este programa se dejará al algoritmo Kruskal para encontrar el mínimo árbol de expansión, mientras que el algoritmo Prim, para encontrar el máximo árbol de expansión.

JUSTIFICACION DEL ALGORITMO DEL ARBOL DE EXPANSION MINIMO

Teorema.- Sea G(N,A,C) una n-red conexa y sea S un árbol de expansión mínimo asociado a G, si T es el árbol de expansión generado por el algoritmo Prim, entonces S = T

Demostración

Sea $T = \{e_1, e_2, e_3, \ldots, e_{n-1}\}$ el árbol generado por Prim, entonces es claro que $e_1 \leq e_2 \leq e_3 \leq \ldots \leq e_{n-1}$

sea $e_i = (x,y)$ el primer arco de T que no está en S, entonces debe existir una trayectoria T única en S que una x con y por lo que $T \cup \{e_i\}$ tiene un ciclo, pero T no contiene ciclos, entonces debe existir al menos un arco e que no éste en T, pero sí éste en S entonces $S' = \{S \setminus e\} \cup \{e_i\}$ es un árbol de expansión por lo que $C(e_i) \geq C(e)$.

Como e_i es el primer arco de T que no está en S, entonces tenemos que si e fue examinado antes que e_i, entonces e formaba un ciclo y entonces e no estaría en T ; pero e_i es el pimer $e_i \in T$ y $e_i \notin S$ entonces $(T \setminus e_i) \subset S$ por lo que $e \in S$ y $e \in T$, lo cual es una contradicción .

por lo tanto $C(e) \geq C(e_i)$

y $C(e) = C(e_i)$ y entonces $C(S') \geq C(S)$ pero $S' = T$

$\therefore S = T$ ∎

Una demostración análoga se puede hacer para justificar el algoritmo Kruskal para calcular el árbol de expansión máximo.

FLUJO MAXIMO

Pensemos en las preguntas que nos hicimos en la introducción y también en el problema, de enviar la mayor cantidad de agua de un cierto punto A a otro punto B de una cuidad, asumiendo que existen diversos caminos para poder hacerlo, entonces, una nueva pregunta sería: ¿Qué trayectorias se deben tomar a manera de enviar la mayor cantidad de agua de A a B?

Existen infinidad de problemas como el anterior, pero todos guardan la misma pregunta:

¿Cuáles son las trayectorias que maximizan el flujo de un punto A a uno B en una red G(N,A,C)?

Este tipo de problemas de maximización de flujo es conocido como el "Problema de flujo máximo" y fue planteado y resuelto por primera vez en 1956 por Ford-Fulkerson, sin embargo el algoritmo utilizado no era muy eficiente en la manera de encontrar la solución, pues podía hacer un número innecesario de iteraciones antes de encontrar la solución, siendo mejorado por Edmonds-Karp en 1969. En 1970 Dinic propuso otro algoritmo aparentemente mejor que el de Ford-Fulkerson y más adelante mejorado por Malhotra (1978) que es el que se presenta.

Pensemos en una red G(N,A,C) conexa, sólo que ahora a C le cambiaremos de nombre, en lugar de costos le llamaremos capacidades, es decir a C le conoceremos como conjunto de capacidades asociados a cada arco de G. Si nos interesara enviar objetos de un cierto nodo i a otro j a través del arco (i,j) y la capacidad asociada a este arco es $c(i,j)$ entonces se podrá enviar cualquier número de objetos siempre y cuando este no exceda a $c(i,j)$. A la cantidad de objetos que se pueden enviar de un nodo i a otro j a través del arco (i,j) (si existe) o más general a través de una trayectoria T, se le denomina *flujo* y a los elementos de este flujo, unidades de flujo. Es claro que el número de unidades de flujo puede variar dependiendo de la trayectoria que se escoja o de las trayectorias que se escojan, por lo que si nos preguntamos por cual sería la mayor cantidad de flujo que podemos llevar del nodo i a uno j, sería un buen problema, obviamente sin violar las capacidades de los arcos. El problema del flujo máximo consiste en maximizar el flujo de un

nodo s a uno t en G, sobre el conjunto de todos los flujos que se conservan a través de los arcos que unen s a t y sean factibles con respecto a sus capacidades. Aunque sólo hemos hablado de capacidades como un número máximo, en realidad es un intervalo, es decir, a cada arco (i,j) podemos asociarle un intervalo de capacidad $(c^-(i,j), c^+(i,j))$ en donde $c^-(i,j)$ es la mínima capacidad y $c^+(i,j)$ es la máxima capacidad de flujo que podemos llevar a través del arco (i,j) en particular y de manera natural puede permitirse a $c^-(i,j)$ como cero. Existen al menos dos restricciones que se deben tomar en cuenta, si es que se quiere resolver un problema de flujo máximo, la primera es que la red debe ser una red dirigida ya que arcos sin dirección no restringe la forma en como corre el flujo a través del arco. Para ver la forma en como corregir este tipo de arcos pensemos en el arco de la figura siguiente.

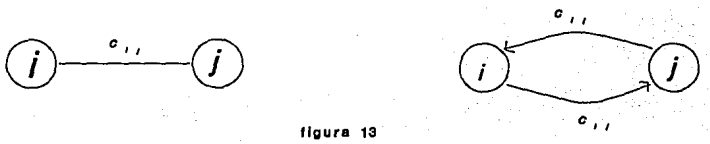


figura 13

Sustituyendo este arco sin dirección por otros dos, pero con direcciones opuestas y capacidades iguales a la del arco sin dirección. Una segunda restricción es que se asume que los nodos no tienen capacidad, cosa que en la realidad sí es importante, por ejemplo, una represa puede ser tomada como un nodo y está físicamente si tiene una capacidad limitada, en este caso el problema se resolvería de la siguiente manera, pues de nuevo el nodo es sustituido por dos arcos con sentidos opuestos con capacidades iguales a la capacidad del nodo, teniendo así en esta nueva gráfica nodos de capacidad ilimitada.

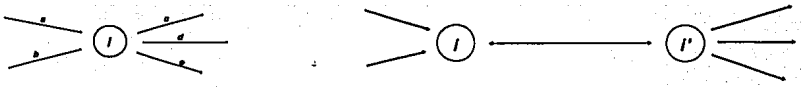


figura 14

La definición formal de flujo es la siguiente:
Sea G(N,A,C,) una red dirigida donde C es el conjunto de capacidades asociadas a cada arco (i,j) en A, un flujo de *s* a *t* (con s, t $\in$ N) es una asignación de valores reales no negativos f_{ij} asociados a cada arco (i,j) respectivamente tal que las cuatro condiciones siguientes se cumplen:
i) Para todo (i,j) $\in$ A, $0 \leq f_{ij} \leq c^{+}(i,j)$.

ii) $\sum_{i} f_{si} - \sum_{i} f_{is} = \mathcal{F}$

iii) $\sum_{i} f_{ti} - \sum_{i} f_{it} = -\mathcal{F}$

iii) $\sum_{i} f_{ji} - \sum_{i} f_{ij} = 0$

$\mathcal{F}$ es llamada la cantidad de flujo de "s" a "t" y lo que se trata es de maximizar $\mathcal{F}$ sobre todo el conjunto de flujos de "s" a "t". Cabe mencionar que la condición i nos asegura que el flujo sea factible, las condiciones ii) y iii) que el flujo se conserva y la condición iv) nos dice que todo el flujo que entra en un nodo es igual a todo el que sale del mismo. Antes de mostrar el algoritmo para encontrar el flujo máximo en una red G(N, A, C) (obviamente conexa) veamos como puede ser este encontrado en un tipo especial de red conocida como red en capas. Posteriormente dada una red G(N,A,C) dirigida veremos como calcular el flujo máximo en ella en base a la descomposición de G(N,A,C) en redes en capas.

Definición.- Decimos que una red G(N,A,C) es una *Red en Capas* si el conjunto de nodos de G puede particionarse en N_k, subconjuntos de G tal que se cumplen las siguientes condiciones:

i) $N_1 = \{s\}$, donde s es el nodo origen a partir del cual queremos que se origine el flujo.

ii) N_2 es el conjunto de nodos que son sucesores de s, es decir, son los nodos que están a distancia 1 del origen.

iii) N_ℓ es el conjunto de nodos sucesores a los nodos de $N_{\ell-1}$ y que estan a distancia ℓ-1 de N_1 (a distancia 1 de $N_{\ell-1}$) , con $1 < \ell < k$.

iv) $N_k = \{t\}$ donde t es el nodo destino.

v) Dados n, m en N_ℓ , para toda $\ell \in \{1,2,\ldots,k\}$ no existe arco $(a,b) \in A$ tal que $a = n$ y $b = m$ o $a = m$ y $b = n$.

Para fijar ideas de lo que es una red en capas veamos la siguiente figura.

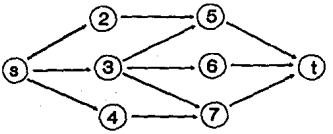


figura 16

Donde $N_1 = \{s\}$, $N_2 = \{2,3,4\}$, $N_3 = \{5,6,7\}$, $N_4 = \{t\}$.
notemos que no existen arcos adyacentes entre los nodos de N_ℓ para $\ell \in \{1,2,3,4\}$.

Es claro que no cualquier red G(N,A,C) es una red en capas, sin embargo de ésta puede extraerse casi siempre una red en capas, Antes de hacer ésto veamos como resolver el problema de flujo maximo en este tipo de redes, para posteriormente hacerlo en cualquier tipo de red.

Sea G(N,A,C) una red conexa y definamos el potencial de un nodo v $\in$ N como el máximo flujo que puede atravesar a v, entonces, el potencial de v denotado por pot(v) se define como el min {flujo absorbente, flujo brotante} es decir, el potencial de un nodo *v* es la mínima cantidad de, entre lo que puede entrar y lo que puede salir de el. Llamaremos nodo de referencia al nodo *v* de G(N,A,C) que tenga mínimo potencial, es claro que puede haber más de un nodo de referencia, pues nada restringe el hecho de que existan dos o más arcos bajo las mismas condiciones de absorción y de brote de flujo.

Bien, la manera de encontrar un flujo saturador en una red G(N,A,C) es la siguiente, primero encontramos el nodo de referencia (o los nodos de referencia) tomando uno de ellos, digamos v, entonces colocamos un flujo que pase a través de v igual a pot(v), dicho flujo debe provenir desde el nodo orígen s, del cual queremos encontrar el flujo saturador. Como segundo paso de la primer iteración, borramos el nodo v que ya ha quedado saturado y desde luego los arcos que quedaron saturados al saturar v, debe existir al menos un arco que salga o que entre ya que el potencial de v de ahí se calculó. Se puede notar que al colocar este flujo inicial igual a pot(v) se modificaron las capacidades de los arcos, así que en una segunda iteración se deben volver a recalcular los potenciales para todos los nodos que no se borraron así como las capacidades de los arcos, aún cuando pudieron haber quedado nodos sin modificar y repetir el paso anterior, finalizando las iteraciones hasta que ya no exista una trayectoria de s a t en la red resultante en la última iteración. Calculemos un flujo saturador a la siguiente red G(N,A,C) direccional.

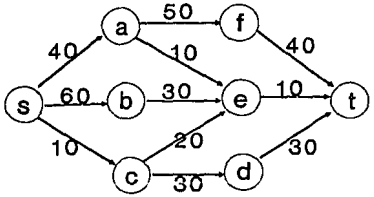


figura 16

Primero calculemos los potenciales de cada nodo para así poder encontrar el nodo de referencia.

	s	a	b	c	d	e	f	t
pot(v)	110	40	30	10	30	10	40	80

por lo que el nodo de referencia puede ser *c* o *e*.
Tomemos arbitrariamente a e y coloquemos un flujo de diez unidades a través de la trayectoria T = {(s,a), (a,e), (e,t)}. puesto que hemos saturado a los arcos (a,e) y (e,t) los borramos junto con el nodo e, además de los arcos (s,b), (b,e) y (c,e) pues ya no podemos colocar flujo a través de ellos, quedando la red:

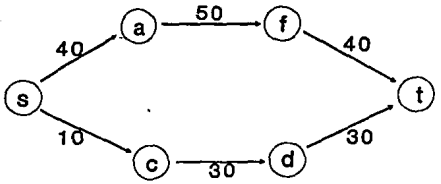


figura 17

Haciendo cálculos encontramos que los potenciales de los nodos restantes son:

	s	a	c	d	f	t
pot(*)	50	30	10	30	40	70

entonces el nodo de referencia es c. Y colocando un flujo de pot(c) unidades saturamos el arco (s,c) así que es borrado y los arcos (c,d) y (d,t) también, pues ya no existe trayectoria a través de ellos, transformandose de la manera siguiente:

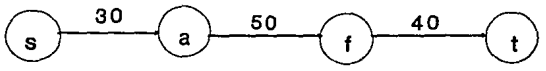


figura 18

Haciendo de nuevo la iteración tenemos que:

	s	a	f	t
pot(*)	30	30	40	40

y entonces el nodo de referencia es *s* o *a*, así que escogemos al nodo *a* quedando:

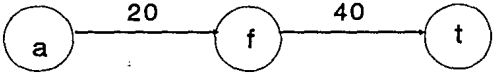


figura 19

Deteniendo ahí las iteraciones, pues ya no existe trayectoria que una el nodo *s* con el *t*.

El flujo saturador en la red G original quedaría:

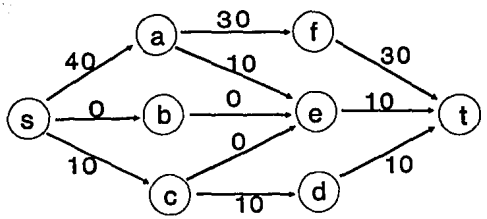


figura 20

Notemos que no existe una trayectoria del nodo s al t que no esté saturada.

ALGORITMO DE FLUJO MAXIMO

Dada una red $G(N,A,C,\mathcal{F})$ a esta red se le asigna un flujo $\mathcal{F}$ de cero unidades, es decir, inicializamos la red con un flujo cero. Una vez hecho ésto extraemos una red en capas G_L y de ésta encontramos un flujo saturador $\mathcal{F}_L$ asociado a G_L, este flujo saturador lo asignamos al flujo $\mathcal{F}$ de G (e.d. $\mathcal{F}:=\mathcal{F}_L$) construyendo así una red $G'(N,A,C,\mathcal{F}=\mathcal{F}_L)$; ahora a G' le extraemos una nueva red direccional G'_L y tambien un flujo saturador $\mathcal{F}'_L$, asignando a $\mathcal{F}$ el valor de $\mathcal{F}'_L$ ($\mathcal{F}:=\mathcal{F}_L$) notemos que para este paso $\mathcal{F}$, es distinto de cero. Repitiendo estas iteraciones tantas veces como redes direccionales se le puedan extraer a G. El algoritmo es el siguiente:

Sea $G(N,A,C)$ y se $C[i,j]$ la matriz de capacidades de la red.

```
begin
 for toda (u,v) ∈ A do
  Obtén una red direccional asociada a G;
 while stpath = true do        {stpath es una variabble que nos
 begin                          identifica si en la red direccional
  Satura la red direccional     existe una trayectoria de s a t}
  for toda (i,j) ∈ A do
   f_ij := f_ij + ff_ij        {donde ff_ij es el flujo saturador de la
                                red direccional}
```

Encuentra una nueva red direccional para G con el

nuevo flujo asignado

end; {While}

Es importante mencionar que el problema de flujo máximo puede ser visto como un problema de programación lineal, el cual puede ser descrito de la siguiente manera. Aunque no es la idea del presente trabajo verlo de esta forma.

$$\text{Max}(\sum_{i} f_{si} = \mathcal{F})$$

$$\text{s.a.} \quad \text{i)} \ \sum_{i} f_{ia} - \sum_{k} f_{ak} = 0$$

$$\text{ii)} \ f_{ij} \leq c_{ij} \quad \text{Para todo arco } (i,j) \text{ en } A$$

$$\text{iii)} \ f_{ij} \geq 0 \quad \text{Para todo arco } (i,j) \text{ en } A$$

donde de nuevo i) es la condición de conservación del flujo, ii) es la factibilidad del flujo y iii) la no negatividad del mismo. Varios autores como Fulkerson llaman al "problema de flujo máximo" como "problema de flujo máximo-corte mínimo", donde un corte es un conjunto Q de arcos tal que al eliminarlos de la red G(N,A,C) se tiene que G(N, A-Q, C) es disconexa y $N = N^+ \cup N^-$, donde N^+ y N^- son los conjuntos de nodos de cada parte conexa. Para mayor facilidad, puede representarse el corte Q como $C(N^+, N^-, Q)$. Es evidente que cualquier flujo máximo que vaya del nodo origen al nodo destino debe pasar necesariamente por un conjunto Q de arcos saturados, este conjunto de arcos es llamado corte mínimo.

Observación.- Sea G(N,A,C) una red conexa y sea $\mathcal{F}$ un flujo máximo, definamos para este flujo máximo a N^+ como $N^+ = \{i, j \in N \mid$ si $(i,j) \in A$ ent. $f(i,j) < C(i,j)$ y $f(i,j) > 0\}$ y a N^- como $N^- = N - N^+$ entonces $C(N^+, N^-, Q)$ es un corte donde si $(i,j) \in A$ con $i \in N^+$ y $j \in N^-$ entonces (i,j) esta saturado.

La observación anterior es cierta, pues de no ser un corte, debe existir un arco (i,j) no saturado en A con $i \in N^+$ y $j \in N^-$ entonces se debe cumplir una de las tres condiciones siguientes

$f(i,j) < C(i,j)$ o $f(i,j) = C(i,j)$ o $f(i,j) > C(i,j)$

$f(i,j) < C(i,j)$ no puede ser, pues estamos suponiendo que $j \in N^-$

$f(i,j) > C(i,j)$ no puede ser, pues $\mathcal{F}$ es máximo y ésto implica que sea factible, entonces $f(i,j) = C(i,j)$ lo cual contradice que (i,j) no era saturado. por lo tanto $(N^+ \cup N^-)$ es un corte.

Teorema del Flujo Máximo - Corte Mínimo.

Sea $\mathcal{F}$ el valor del flujo en una red G(N,A,C)

entonces $\max_{\mathcal{F}} \mathcal{F} = \min_{(N^+N^-)} C(N^+ \cup N^-)$

Por lo anterior se deduce que si $\mathcal{F}$ es un flujo (no necesariamente maximo) asociado a una red G(N,A,C)

entonces $\mathcal{F} \leq \min_{N^+N^-} C(N^+ \cup N^-)$

En la siguiente Red G(N,A), es calculado el flujo máximo del nodo uno al seis. Cada número sobre los arcos de la red denotan sus capacidades.

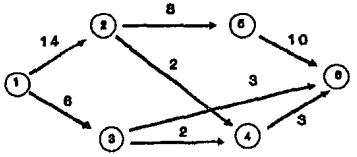


figura 21

Red en capas G' extraida de G.

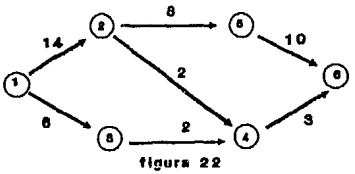


figura 22

Flujo saturador obtenido en G'.

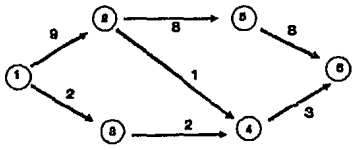


figura 23

Actualización de las capacidades de G.

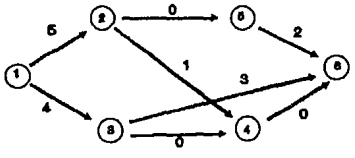


figura 24

Nueva Red G" en capas, eliminando los nodos y arcos que no pertenecen a ninguna trayectoria, del nodo uno al seis, al eliminar los arcos de capacidad cero.

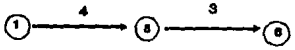


figura 25-a

Flujo saturador obtenido en G".

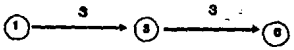


figura 25-b

Nueva actualización de G con respecto a sus capacidades

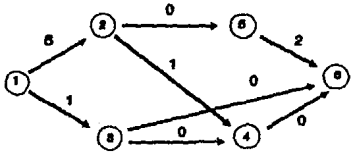


figura 26

Al eliminar el conjunto de arcos {(2,5), (3,4), (3,6), (4,6)} ,de capacidad cero, la red resultante es disconexa, lo cual quiere decir que el flujo asignado a G es un flujo máximo por la forma en como se encontro, quedando la asignación de flujos de la siguiente manera:

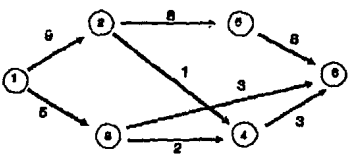


figura 27

EL PROBLEMA DEL FLUJO FIJO A COSTO MINIMO

Supongase que ahora en cierta red, se tienen identificados dos nodos "s" y "t", llamados origen y destino respectivamente, sólo que ahora no nos interesa mandar la mayor capacidad de flujo de s a t, lo que se quiere decir es que el nodo destino no desea abastecerse del mayor flujo posible sino que solo se desea una cantidad fija de flujo, entonces es claro que puede existir más de una trayectoria que cumpla con ésto, el problema ahora se transforma, en encontrar una trayectoria (o conjunto de trayectorias) por la que se pueda enviar una cantidad fija de flujo a un costo mínimo.

La diferencia entre el problema de flujo máximo y el problema de flujo a costo mínimo, es que en el flujo máximo no involucramos costos mientra que en el de costo mínimo si, además de maximizar flujo en el primer problema y minimizar costos en el segundo.

Al igual que en flujo máximo se deben cumplir las siguientes cuatro propiedades

i) $\sum_i \ell_{si} - \sum_i \ell_{is} = \mathcal{F}$ donde s es el nodo origen.

ii) $\sum_i \ell_{ti} - \sum_i \ell_{it} = -\mathcal{F}$ donde t es el nodo destino.

iii) $\ell_{ij} \leq c_{ij}$ para toda $(i,j) \in A$

iv) $\sum_i \ell_{ji} - \sum_i \ell_{ij} = 0$ flujo balanceado

Dado ésto podemos pensar a nuestro problema de costo mínimo, como un problema de programación lineal de la siguiente manera.

$$\text{Min} \sum_{ij} d_{ij}\ell_{ij}$$

s.a. i) $\sum_i \ell_{ji} - \sum_i \ell_{ij} = 0; \quad j \in N \setminus \{s,t\}$

ii) $\ell_{ij} \leq c_{ij}$

iii) $\ell_{ij} \geq 0;$

Veamos un caso particular de este problema. Pensemos en la siguiente red G(N,A,C,¢) (Nodos, Arcos, Capacidades, ¢ostos)

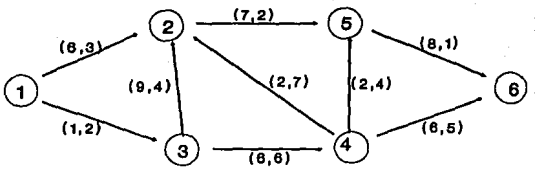


figura 28

Donde cada arco (i,j) tiene asociado un vector de la forma (c_{ij}, $¢_{ij}$) , donde c_{ij}, indica la capacidad máxima de flujo asociado al arco (i,j) y $¢_{ij}$, indica el costo asociado al arco (i,j) por mandar una unidad de flujo a través del mismo arco. En este caso particular la capacidad asociada a cualesquier arco (i,j) es de una unidad de flujo y el costo, si varía dependiendo del arco.

Supongase que la cantidad de flujo que se desea enviar es de una unidad, entonces no nos preocupa la elección de arcos en cuanto a su capacidad, pues es claro que cualquier arco puede ser utilizado para enviar α unidades de flujo (α=1), más bien la elección depende de los costos de envío, por lo que, si nos olvidamos de las capacidades y del valor α de flujo y pensamos los costos como "distancias", nuestro problema se reduce a encontrar la ruta más corta de "s" a " t", pudiendo utilizar el algoritmo PDM o el de Dijkstra (vistos al inicio de este trabajo).

Es claro que si $\alpha \leq ¢_{ij}$ para toda $i,j \in \{1,2,3,\ldots,n\}$ y por supuesto con $\alpha > 0$, PDM o Dijkstra pueden ser utilizados inclusive, pues encontrando la ruta más corta de "s" a "t" bastaría con checar que el arco con menor capacidad $¢_{ij}$ en la trayectoria de la ruta más corta cumpliera que $\alpha \leq ¢_{ij}$.

También puede inferirse que si $\mathcal{F} < \alpha$ (con $\mathcal{F}$ el valor del flujo máximo de "s" a "t") el problema no tiene solución. Faltaría ahora preguntarnos que pasaría si $\alpha > ¢_{ij}$, para toda i,j en el siguiente conjunto $\{1,2,3,\ldots,n\}$ y con $\alpha \leq \mathcal{F}$. Bien la manera de proceder es la siguiente. Supongamos que deseamos mandar $\alpha = 7$

unidades de s a t, calculando la ruta más corta de s a t tomando a los costos como distancias y enviando a través de dicha ruta (trayectoria) tantas unidades de flujo como sean posibles. Bien, aplicando Dijkstra a la red anterior tenemos que:

dist = (0, 3, 2, 8, 5, 6) y **pred** = (-1, 1, 1, 3, 2, 5)

es decir, la trayectoria T = { (1,2), (2,5), (5,6)},

es la ruta más "corta" (costo mínimo) para alcanzar a t desde s, ahora nos fijamos que las capacidades de los arcos (1,2), (2,5) y (5,6) son 6, 7, 8 respectivamente, entonces a través de esta trayectoria podemos mandar un máximo de seis unidades de flujo a costo mínimo, pero necesitamos mandar siete unidades, así que actualicemos la red G.

Red actualizada con el flujo inicial $\alpha' = 6$.

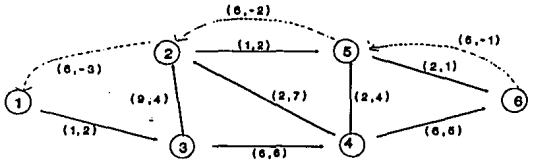


figura 29

Ahora tenemos una red con costos negativos y positivos, entonces calculamos de nuevo la ruta más corta de s a t, pero ahora utilizando PDM pues Dijkstra no trabaja con costos negativos, obtenemos que,

dist = (0, 6, 2, 8, 8, 9) y **pred** = (-1, 3, 1, 3, 2, 5)

por lo que ahora la trayectoria más corta de s a t :

T = {(1,3),(3,2),(2,5),(5,6)} donde sus capacidades respectivas son 2, 4, 2, 1 respectivamente, así que podemos mandar a lo más una unidad de flujo a través de esta ruta, terminando aquí el proceso pues se alcanzó el flujo deseado. Quedando la red con los siguientes flujos.

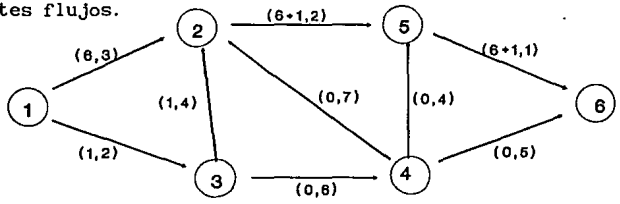


figura 30

pagando un costo mínimo de 6*3 + 7*2 + 7*1 + 1*2 + 1*4 = 45 unidades monetarias.

JUSTIFICACION DEL ALGORITMO DE FLUJO A COSTO MINIMO

Notemos que al actualizar la red original G(N,A,C) se colocaron arcos artificiales en sentido opuesto a los arcos originales que pertenecen a la trayectoria y además, con una capacidad igual al flujo colocado en el arco original a través de dicha trayectoria, a esta red resultante le llamaremos Red Actualizada o Red Modificada. Antes de mostrar el teorema que nos justifique el algoritmo, definamos bien lo que llamamos red actualizada o modificada.

Definicion. Sea G(N,A,C,¢) y sea f un flujo factible en G(N,A,C,¢) una red G'(N,A',C',¢') decimos que es una red actualizada o modificada a partir de f, si para todo $(i,j) \in A$ con capacidad C_{ij} y $¢_{ij}$ se coloca en este el flujo f_{ij}, este se convierte en el arco (i,j) con capacidad $C_{ij} - f_{ij}$ y costo $¢_{ij}$ y además se anexa un arco (j,i) con capacidad $C_{ji} = f_{ij}$ y costo $¢_{ji} = -¢_{ij}$

Entonces A' = A ∪ {conjunto de arcos que se anexan}

C' = C - f

¢' = ¢ ∪ -¢.

La idea de crear una red G' (red modificada) a partir de colocar un flujo en f en una red G es la siguiente, recordemos que el algoritmo PDM no puede ser utilizado si existen circuitos negativos, pues para cada ruta que recorra un circuito negativo, sería "más corta" si se recorriera dos veces el mismo circuito, (ver ejem. pag.16) entonces, al poner arcos auxiliares con costos negativos a la red original, posiblemente encontraremos circuitos negativos lo que querría decir que podemos mejorar el costo del flujo, además, el concepto físico de estos arcos es que dado un arco (i,j) en este podemos aumentar o decrementar flujo. Lo anterior se resume en el siguiente teorema.

Teorema.- Sea G(N,A,C,¢) una n-red conexa y sea f un flujo f, factible de s a t entonces f es de costo mínimo si y sólo si en la red modificada G'(N,A',C',¢') no existen circuitos negativos.

Para una demostración puede consultarse la referencia 7.

Denotemos por G'(f) a la red modificada, a partir de colocar un flujo f en una red G.

Teorema.- Sea G(N,A,C,¢) una n-red conexa y sea f un flujo factible de valor *v* y de costo mínimo de cierto nodo origen s a otro nodo destino t, sea T la trayectoria más corta que une s a t en la red modificada G'(f), entonces el flujo f' = f+1 colocado a través de T, es un flujo de valor v + 1 y de costo mínimo.

Demostración.

Sea G''(f') la red modificada a partir de colocar un flujo f'=f+1 en la red modificada G'(f) entonces lo que hay que probar es que G''(f') no tiene circuitos negativos.

G'(f) y G''(f') tienen los mismos arcos excepto quizá en los arcos de T, e.d. si al colocar el flujo f'=f+1 en T no se satura ningún arco entonces G'(f) = G''(f'), en otras palabras, si $f''_{ij} = f_{ij} + 1 < C_{ij}$ para todo arco (i,j) recorrido positivamente y $f''_{ij} = f_{ij} - 1 > 0$ para todo arco (i,j) recorrido negativamenteentonces G'(f) = G''(f'), pero G'(f) no tiene circuitos negativos por lo que G''(f') tampoco y por lo tanto f' = f + 1 es de costo mínimo.

Supongamos ahora que existe (i,j) en T tal que $f''_{ij} = f_{ij} + 1 = C_{ij}$ entonces el arco (i,j) ∈ T ∈ G'(f) e (i,j) ∉ g''(f') pero (j,i) ∈ G''(f').

Sea *C* un circuito en G''(f') con $C = \{i, n_1, n_2, \ldots, n_k, j, i\}$ para algunos $n_i \in N$, entonces el costo asociado a este circuito es $¢'_{in_1} + ¢'_{n_1n_2} + \ldots + ¢'_{n_kj} + ¢'_{ji}$ pero el costo de ir de i a j es $¢'_{ij}$ pues T es la ruta más corta, por lo que

$$¢'_{in_1} + ¢'_{n_1n_2} + \ldots + ¢'_{n_kj} + ¢'_{ji} > ¢'_{ij} \geq 0$$

$$¢'_{in_1} + ¢'_{n_1n_2} + \ldots + ¢'_{n_kj} + ¢'_{ji} - ¢'_{ij} \geq 0.$$

Un razonamiento análogo puede pensarse para el caso de que f'=f-1 = 0,

por lo que G''(f') no tiene circuitos negativos y por lo tanto f' es de costo mínimo y de valor v+1.

■

Otra manera de ver el teorema anterior es la siguiente:
Sea $\mathcal{F}_1$ un flujo a costo mínimo con α_1 unidades en una red G, y sea la red G_1^* la red actualizada con respecto a $\mathcal{F}_1$. Sea T una trayectoria a costo mínimo de s a t en G_1^*, entonces el flujo $\mathcal{F}_2$ obtenido, al enviar unidades adicionales de s a t a lo largo de T, también es un flujo a costo mínimo.
En el siguiente ejemplo, es calculado el costo mínimo por enviar siete unidades de flujo a través de la red G, partiendo del nodo uno para llegar al cuatro. Cada pareja sobre los arcos de la red representan la capacidad y el costo respectivamente de ese arco.

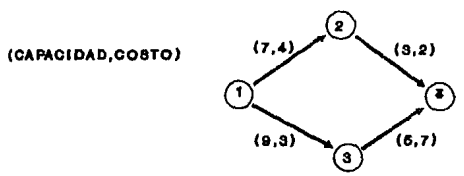
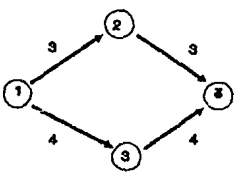


Figura 32

El flujo máximo a través de la red uno a cuatro, es de ocho unidades lo cual quiere decir que si existe solución al problema de flujo a costo mínimo. La asignación de de flujos en G queda de la siguiente manera:



Siendo el costo mínimo de $3(4+2) + 4(3+7) = 58$ unidades monetarias.

CONCLUSIONES

En el presente trabajo se creó un programa de cómputo, que resuelve cuatro problemas clásicos de optimización en redes; este programa puede ser ejecutado en computadoras personales, compatibles con IBM, con procesador 286, 512 K-bytes en memoria RAM y con disponibilidad de 750 K-bytes de espacio en disco duro o flexible. Las características de este software es que se le pueden anexar mas opciones al menú a manera de resolver otro tipo de problemas como coloración, etc. es decir la estructura del programa consiste de una lista de procedimentos para resolver todos y cada uno de los los problemas, un menú dinámico y dos unidades (UNIT) donde se realiza la presentación y validación del programa. La razón por la que los procedimientos principales no se implementaron en una UNIT es para que el usuario que en algún momento desée analizar los algoritmos pueda hacerlo.

Se noto que las características de los problemas de redes pueden modelarse de distintas formas dependiendo de su estructura en cuanto a número de arcos y nodos, tomando el modelo que mejor optimice la manera de resolver este tipo de problemas, a manera de evitar gastos innecesarios de almacenamiento y tiempo al aplicarles los procedimentos de algún algoritmo eficiente. Sin embargo si los problemas a resolver no son de gran magnitud, no es muy importante preocuparse de estos hechos ya que los tiempos de ejecución y gasto de memoria pueden considerarse despreciables. Y si se fuera a trabajar en problemas de mayor magnitud, sería bueno entonces implementar un programa especifico para cada tipo de red o pensar en trabajar con un equipo de cómputo más poderoso, optando quizás por la segunda opción, ya que generalmente en los problemas grandes es dificil saber la relación nodos arcos.

ANEXO I
MANUAL DEL PROGRAMA

INSTALACION

Este programa puede ejecutarse desde disco flexible o disco duro según se desée. Si se requiere trabajar de disco duro solo se deberán seguir los siguientes pasos:

1.- Elabore un subdirectorio en disco duro llamado REDES, siendo este nombre opcional y que puede usted ponerle uno de su preferencia de manera que le sea facil recordarlo. El proceso es el siguiente: Una vez que ha encendido su computadora y ha aparecido el prompt C:> teclée md redes ↵ (donde ↵ indica la tecla de enter)

C:>md redes ↵

2.- A continuación teclée cd redes

C:>cd redes

apareciendo el siguiente prompt

C:\REDES>

3.- Introduzca el disco en la lectora de la computadora y teclée a:↵

C:\REDES>a:↵

cambiando al promt

A:>

Escriba copy *.* c:↵

A:>copy *.* c:↵

espere hasta que se apague la luz de la lectora y aparezca el mensaje:

2 archivo(s) copiado(s)

4.- Regrese al disco duro escribiendo c:↵

apareciendo de nuevo el prompt C:\REDES> , para este momento se han copiado el programa ejecutable y un archivo de datos. Si se desea correr el programa desde la lectora de disco, solo realice el paso tres.

C:>a:↵

EJECUCION DEL PROGRAMA

Para ejecutar el programa teclée redes↵

C:\REDES>redes↵ si se esta en disco duro

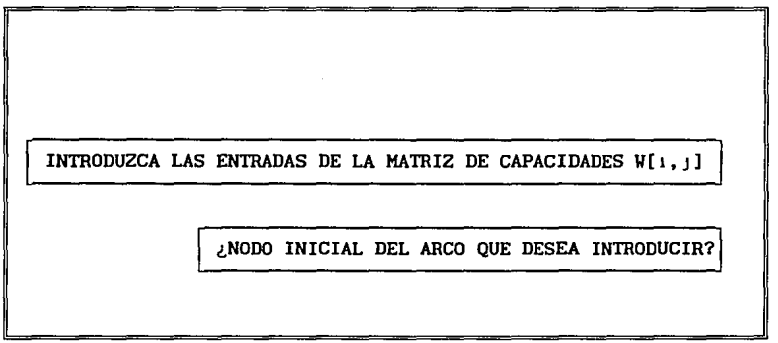
A:>redes↵ si se esta en disco flexible

Apareciendo en la pantalla un juego de ventanas, en donde se pueden leer las palabras **DEPFI-UNAM** y **REDES**. Para entrar al menú principal, el cual es parecido a la figura de abajo, presione cualquier tecla espere algunos segundos.

La selección de cualquier opción puede hacerse moviendo el cursor mediante el uso de las flechas. Para ejecutar cualquiera de las seis primeras opciones es necesario haber cargado en memoria una red mediante la opción **LEER RED** o **INTRODUCIR RED.**

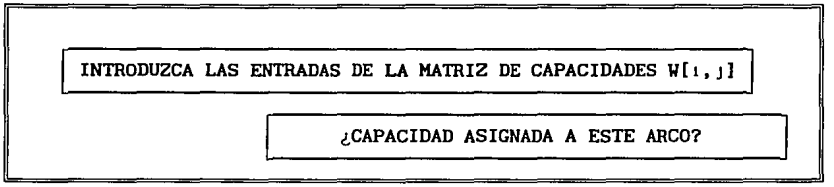
INTRODUCIR RED

Si es primera vez que va a trabajar en este sistema seleccione la opción de introducir red. En la pantalla aparecerá una ventana preguntando cuántos nodos tiene la red, introduzca el dato dependiendo del tamaño de la red que se desea introducir, este debe ser un numero entero menor o igual a 23, que es el maximo de nodos que se pueden introducir, aunque este dato puede ser modificado al cambiar el valor de la constante NODOS en el programa fuente y de nuevo compilando a disco. La manera de introducir una red es mediante una matriz de capacidades, por lo que aparecen dos ventanas con los siguientes mensajes:

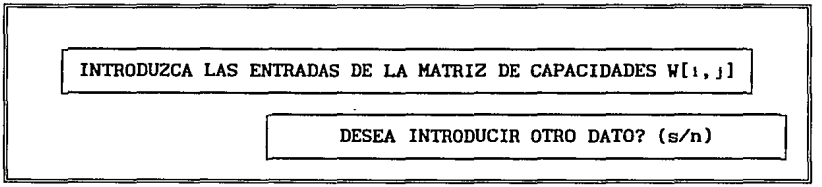


Solo son introducidos los arcos con capacidades positivas en la red, razón por la cual se pregunta el número asignado al nodo inicial del arco a introducir. La manera de introducirlo es tecleando el número y pulsando la

tecla de enter (↵). La siguiente ventana es parecida a la anterior solo que ahora se pregunta por el número j del nodo final. Una vez hecho esto la segunda ventana cambia a:

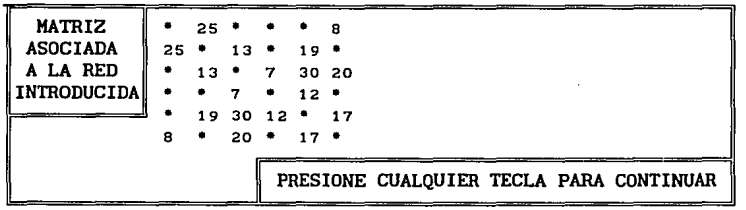


Despues de introducir una capacidad se pregunta si se desea introducir otra.



Basta con presionar la tecla de enter, si es que se quiere introducir otro valor o la letra **N** si no.

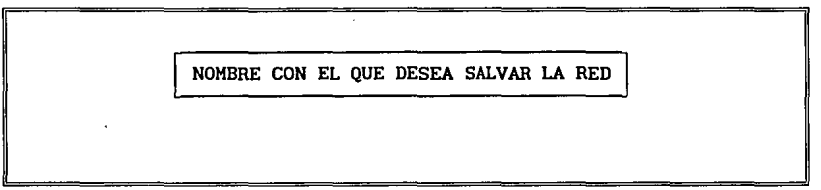
Al seleccionar la tecla **N**, aparecera desplegada en pantalla una matriz con los valores que se acaban de introducir, la cual representa la red con la que se quiere trabajar. El siguiente ejemplo ilustra la matriz asociada a una red de seis nodos llamada **EJEMPLO1.RED** :



los asteriscos representan la inexistencia de los arcos con esas coordenadas. Si existe algún error, este puede ser corregido salvando la red y leyendo este mismo archivo mediante la opción **LEER**. Para regresar al menú principal basta con presionar cualquier tecla.

SALVAR

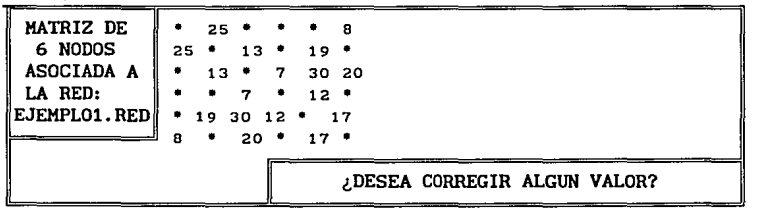
Al seleccionar esta opción aparece una ventana como la que se muestra abajo y la red capturada, previamente, puede ser almacenada en disco.



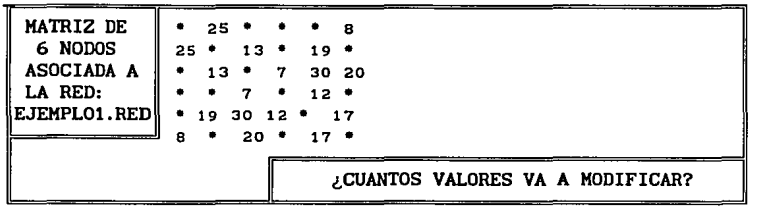
La longitud del nombre no debe de exceder de ocho caracteres, en caso de exceder este número solo serán tomados en cuenta los ocho primeros. La extención del archivo que aparecerá en disco es **.RED** , por ejemplo:
NETWORK.RED , **GRAFICA.RED** , **TELE-RED.RED** , etc.

LEER RED

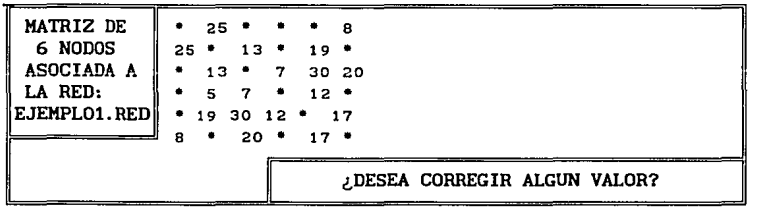
Si se selecciona **LEER RED**, será desplegado el directorio de archivos con extención **.RED**. Puede ser seleccionado cualquiera de ellos ubicando su posición con las flechas del cursor y pulsando la tecla de enter. Desplegandose la matriz, nombre de la red y el número de nodos.

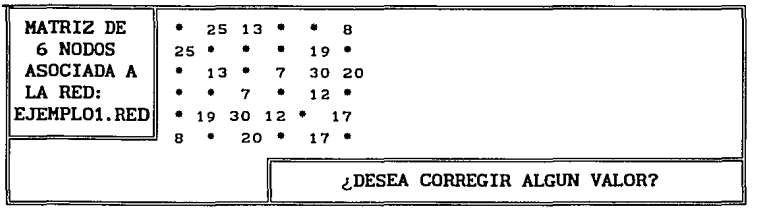


Es en este paso cuando se pueden corregir datos de la red, la manera de hacerlo es la siguiente: Se debe presionar la tecla **S** para contestar a la pregunta de la ventana inferior derecha. Cambiando el mensaje de esta ventana a:



Se debe tener cuidado al contestar la pregunta ¿CUANTOS VALORES VA A MODIFICAR? para fijar ideas, supongase que la entrada (4,2) de la matriz EJEMPLO1.RED es un arco con costo positivo 5 pero no se dio de alta al introducir los datos, por lo que aparece un asterisco. En este caso, el número de valores a modificar es de uno. Otro caso sería cuando por un error de dedo, el valor de la entrada (1,3) fue colocado en la entrada (2,3), en este caso el número de valores a modificar es dos, pues es el valor que se dará de alta y el valor que se dará de baja. *La manera de dar de baja un valor asociado a un arco inexistente es* asignando *el valor de -1 a este arco.* Una vez que se ha indicado el número de valores a modificar, estos se introducen de la manera que se cargo la red al seleccionar la opcion INTRODUCIR RED, los resultados obtenidos en cada caso anterior son de la siguiente manera:

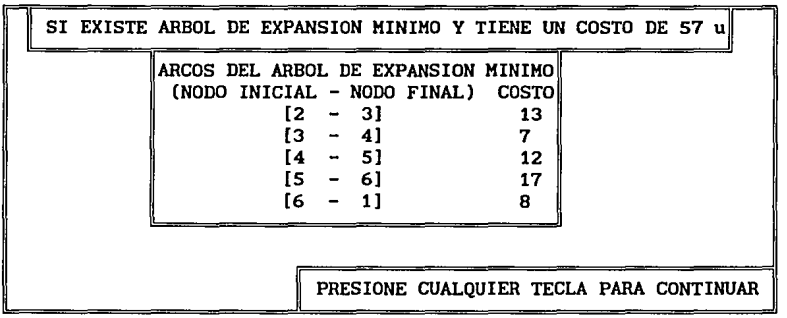




Las redes deben ser salvadas para no perder las modificaciones hechas.

ARBOL DE EXPANSION MINIMO

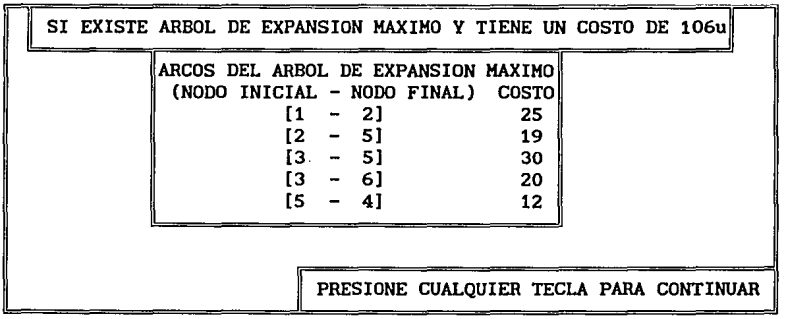
Al seleccionar esta opción automáticamente es calculado el árbol de expansión mínimo de la red almacenada en la memoria de la computadora. La manera de desplegar el árbol de expansión mínimo es listando los arcos que lo forman, así como sus respectivos costos y el costo total del árbol. Es importante mencionar que los arcos de la red son tomados sin dirección aún cuando en esta si lo sean.



Para regresar al menú principal basta con presionar cualquier tecla.

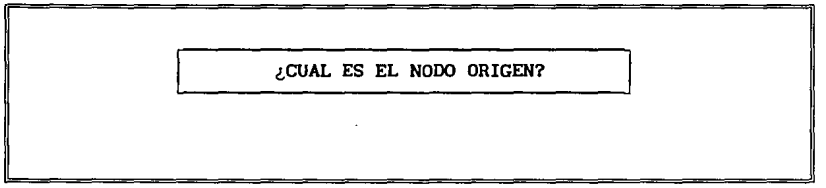
ARBOL DE EXPANSION MAXIMO

Los datos son desplegados de igual manera que como en el árbol de expansión mínimo.

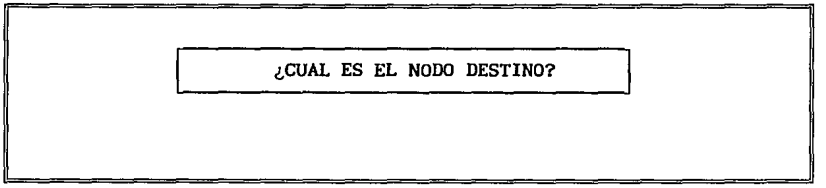


RUTA MAS CORTA DEL NODO s AL t.

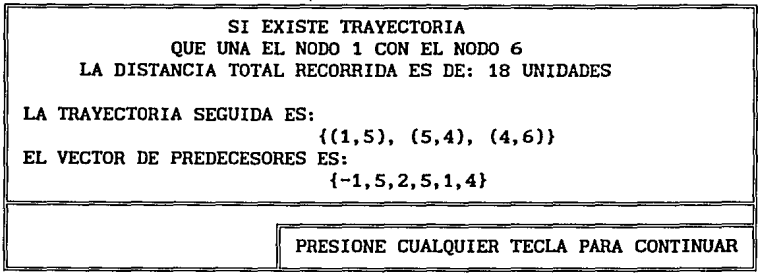
En este procedimiento es calculada una trayectoria *T*, que une un cierto nodo origen llamado **s** a otro nodo destino llamado **t**, a un costo que debe ser mínimo con respecto a las demás trayectorias. Al seleccionar esta opción aparece una ventana que nos pregunta cuál es el nodo origen **s**, es decir el nodo desde el cual se desea empezar a construir la trayectoria *T*. Se debe introducir el número de nodo en la red que se quiere sea el nodo origen.



A continuación se debe introducir el número del nodo que se quiere sea **t**, es decir el nodo hasta el cual se desea llegue la trayectoria *T*.



Una vez introducido este último dato el algoritmo de ruta mas corta encuentra esta y despliega los datos de la siguiente manera:



En esta pantalla se indica si existe trayectoria o no, los nodos origen y destino, el costo de la misma, así como los arcos que la forman. También se muestra un vector llamado vector de predecesores que es otra forma de representar la trayectoria, solo que nodo a nodo. Por ejemplo en la pantalla anterior el predecesor del nodo destino seis (entrada seis de este vector) es cuatro, el predecesor del nodo cuatro (entrada cuatro) es cinco, finalmente el predecesor del nodo cinco (entrada cinco) es uno, que es el nodo origen. así de esta manera tambien se traza la trayectoria a costo minimo del nodo s al t. Siempre en la entrada correspondiente al nodo origen aparecerá un -1 lo cual indica que no tiene predecesor ya que es el inicio de la trayectoria. Basta presionar cualquier tecla para regresar al menu principal.

RUTA MAS CORTA DEL NODO s A TODOS

En este procedimiento, dada una red, se selecciona un nodo origen s y son calculadas las rutas mas cortas para alcanzar los demás nodos de la red.

La manera de introducir el nodo origen es igual al procedimiento anterior y los resultados obtenidos son como se muestran a continuación:

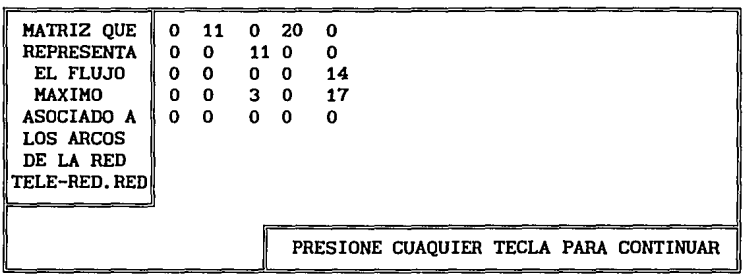
```
        RUTAS MAS CORTAS DEL NODO 1 A LOS NODOS RESTANTES
                    EN LA RED TELE-RED.RED
DEL NODO 1 AL NODO 6
{(1,5),(5,4),(4,6)}  A UN COSTO MINIMO DE 18 UNIDADES
DEL NODO 1 AL NODO 5
{(1,5)}  A UN COSTO DE MINIMO DE 9 UNIDADES
DEL NODO 1 AL NODO 4
{(1,5),(5,4)}  A UN COSTO MINIMO DE 11 UNIDADES
DEL NODO 1 AL NODO 3
{(1,5),(5,4),(4,6),(6,3)}  A UN COSTO MINIMO DE 23 UNIDADES
DEL NODO 1 AL NODO 2
{(1,5),(5,2)}  A UN COSTO MINIMO DE 13 UNIDADES

                        PRESIONE CUALQUIER TECLA PARA CONTINUAR
```

En esta ventana son mostradas las trayectorias, a costo mínimo, que unen el nodo origen 1 con los demás nodos de la red así como también los costos respectivos por recorrerlas. Basta con presionar cualquier tecla para regresar al menú principal.

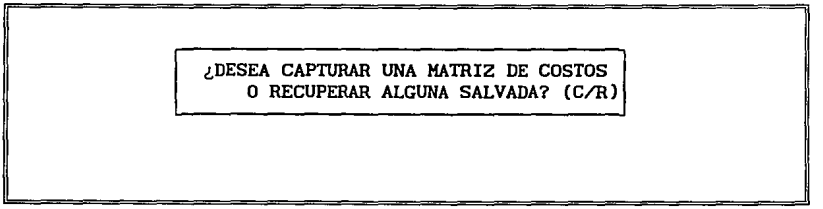
FLUJO MAXIMO

Al igual que en ruta mas corta, aparecen dos ventanas preguntando por el nodo fuente, de donde se quiere mandar el flujo, y por el nodo destino o sumidero, a donde se desea llegue el flujo. A la red existente en memoria le es calculado, mediante el algoritmo de flujo máximo, el mayor número de unidades de flujo que pueden ser enviadas a través de la red. La manera de visualizar los flujos asignados a cada arco de la red es mediante una matriz, como aparece en la figura de abajo, donde cada entrada (i, j) de esta matriz representa las unidades de flujo asignadas al arco (i, j) de la red.



FLUJO A COSTO MINIMO

En este tipo de problema se hace uso de dos representaciones de la red, una que contenga las capacidades de los arcos de la red y otra que contenga los costos asociados por hacer uso de estos arcos al mandar unidades de flujo a través de ellos. Se asume que la matriz de capacidades es la que tiene extensión .RED y se ha cargado en memoria cuando se hizo uso de la opción LEER RED. La representación de la matriz de costos es introducida o recuperada solo en la ejecución de esta opción y es un archivo con extensión .FLW. Es por eso que al seleccionar esta opción aparence una ventana como la siguiente:



Si se presiona la tecla C, se indica que se quiere introducir una nueva red de costos, el procedimiento es parecido a la opción INTRODUCIR RED del menú principal, preguntando solo por los arcos que pueden tener un costo asignado y dando la opción a salvar esta red. Si se presiona la tecla R, el directorio con los archivos de extensión .COS es desplegado, pudiendo seleccionar alguno de ellos de igual manera que cuando se utilizo la opción LEER RED del menú principal. Será marcado un error si la red de costos que se leyó no es compatible con la red de de capacidades.

La siguiente ventana pregunta por el nodo fuente de donde se quiere mandar cierta cantidad de flujo y por el nodo destino o sumidero, a donde se desea llegue el flujo. También se pregunta por el número de unidades de flujo que se quieren mandar a través de la red. El algoritmo calcula el valor del flujo máximo a través de la red de capacidades y si este valor es menor al del flujo que se desea enviar entonces sonará un beep, saliendo de la rutina y dejandolo a uno en el menú principal, pues esto significa que el problema no tiene solución. Una vez calculado el flujo máximo y habiendo checado si existe solución, son calculados los flujos asignados a los arcos de la red para mandar el flujo deseado a un costo mínimo. Los datos de salida son desplegados de la siguiente manera:

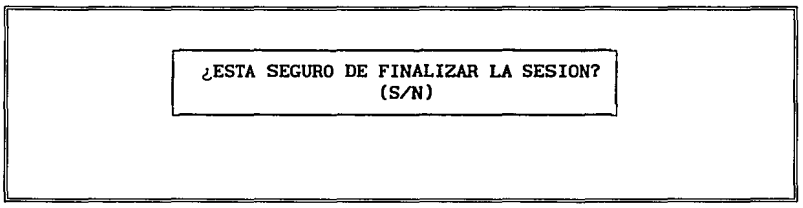
```
FLUJOS ASIG   0  8   0  17  0
NADOS A LA:   0  0   8  0   0
  RED:        0  0   0  0   8
EJEMPLOS.RED  0  0   0      17
A UN COSTO    0  0   0  0   0
MINIMO DE:
    225
UNIDADES ($)

                  PRESIONE CUAQUIER TECLA PARA CONTINUAR
```

Donde al igual que en flujo máximo, la entrada (i,j) de la matriz desplegada corresponde al flujo asignado al arco (i,j) de la red. En la ventana suoerior derecha aparece el costo que se debe pagar por enviar las unidades de flujo preestablecidas.

FINALIZAR SESION

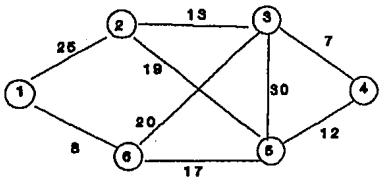
FIANLIZAR SESION es la ultima opción del menú principal, esta opción se utiliza cuando se ha desea terminar, no sin antes preguntar si uno esta seguro de finalizar, en caso negativo nos regresa al menú principal. Es recomendable usarla para evitar cualquier daño al programa ejecutable.



ANEXO II

REDES ALMACENADAS EN EL DISCO FLEXIBLE

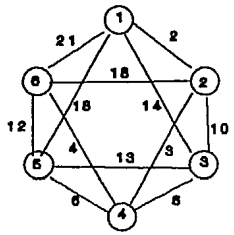
EJEMPLO1.RED



*	25	*	*	*	8
25	*	19	*	19	*
*	19	*	7	30	20
*	*	7	*	12	*
*	19	30	12	*	17
8	*	20	*	17	*

Donde se encontró el árbol de exp. mínimo.

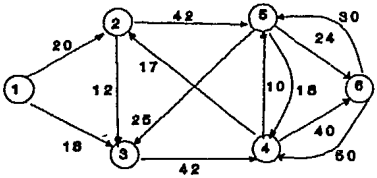
EJEMPLO2.RED



*	2	14	*	18	21
2	*	10	3	*	18
14	10	*	8	13	*
*	3	8	*	6	4
18	*	13	6	*	12
21	18	*	4	12	*

Donde se encontró el árbol de exp. máximo.

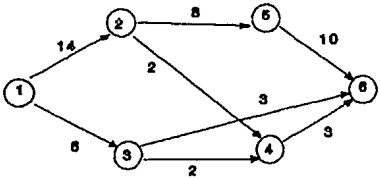
EJEMPLO3.RED



*	20	18	*	*	*
*	*	12	*	42	*
*	*	*	42	*	*
*	17	*	*	10	40
*	*	25	18	*	24
*	*	*	50	30	*

Donde se encontró la ruta mas corta de uno a seis.

EJEMPLO4.RED

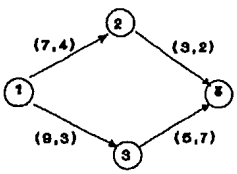


*	14	6	*	*	*
*	*	*	2	8	*
*	*	*	2	*	9
*	*	*	*	*	9
*	*	*	*	*	10
*	*	*	*	*	*

Donde se encontró el flujo máx. posible de uno a seis

EJEMPLO5.RED

(CAPACIDAD,COSTO)



*	7	9	*
*	*	*	9
*	*	*	5
*	*	*	*

EJEMPLO5.FLW

*	4	9	*
*	*	*	2
*	*	*	7
*	*	*	*

Donde se encontro el costo mínimo para enviar un flujo fijo del nodo uno al seis.

ANEXO III

LISTADO DEL PROGRAMA

```
PROGRAM REDES;
USES DOS, CRT,LIB_LEC,VENTANAS,TURBO3;{EN LIB_LEC Y VENTANAS SE ENCUENTRAN}
CONST                                  {LIBRERIAS PARA DESPLEGAR VENTANAS, }
 NODOS=23;  {*NUMERO DE NODOS*}        {MENUS, PAUSAS, ETC}
 ARCOS=400; {numero de arcos}
 INF=32000;
 TYPE
 ARRNN=ARRAY[1..NODOS,1..NODOS] OF INTEGER;
 ARRN =ARRAY[1..NODOS] OF INTEGER;
 ARRMAX=ARRAY[1..1200] OF INTEGER;
 ARRM =ARRAY[1..ARCOS] OF INTEGER;
 ARRN1 =ARRAY[1..NODOS+1] OF INTEGER;
 ARRNM1 =ARRAY[1..NODOS-1] OF INTEGER;
 BOOLARRN =ARRAY[1..NODOS] OF BOOLEAN;
VAR
 ANTE,TWEIGHT, OPCION, M,CONTADOR,NOD,Z,B   : INTEGER;
 C,D,Y,K,I,J,S,T,N,NUMERO,NUMERO1,NUMERO2   : INTEGER;
 MATGEN, W,COST, CAPA, FLOW                 : ARRNN;
 YAND, CONNECT, REPETIR, PATH, CONTINUA     : BOOLEAN;
 FINAL                                      : BOOLARRN;
 DIST, PRED                                 : ARRN;
 TEDGE1,TEDGE2                              : ARRNM1;
 ENDV,ENDV1,ENDV2,WEIGHT,WT                 : ARRM;
 POINTER                                    : ARRN1;
 TECLASO                                    : CHAR;
 CADENA, NOMBRE, NOM                        : STRING;
 ARCH, ARCHL                                : FILE OF ARRNN;
PROCEDURE ADIOS; FORWARD;
PROCEDURE MINIMOARBOL(VAR MATGEN : ARRNN;N:INTEGER);{PROCEDIMIENTO  PARA
                                  ENCONTRAR EL ARBOL DE EXPANSIÓN MINIMO}
 PROCEDURE TRANSFO2(VAR MATGEN : ARRNN; VAR ENDV1, ENDV2, WEIGHT : ARRM;
                    VAR N, M : INTEGER);
 VAR
 CONTADOR, I, J     : INTEGER;
 APUNTA             : ARRN1;
 INICIAL,FIN, PESO  : ARRM;
 BEGIN
  CLRSCR;
  M:=0; i:=1;j:=1;
  CONTADOR:=0;
  FOR i:=1 TO N DO
   FOR j:=1 TO N DO
   BEGIN
    IF MATGEN[i,j]<0 THEN
     MATGEN[i,j]:=INF
    ELSE
     CONTADOR:=CONTADOR+1;
   END;
   M:=CONTADOR DIV 2;
   i:=1;
   FOR i:=1 TO 400 DO
    ENDV1[i]:=0;
```

```
   ENDV2[i]:=0;
   WEIGHT[i]:=0;
   i:=1;
   j:=1;
   CONTADOR:=0;
   FOR i:=1 TO N DO
    BEGIN
     FOR j:=1 TO N DO
      IF j>i THEN
       BEGIN
        IF (MATGEN[i,j]<>0) AND (MATGEN[i,j]<>INF) THEN
         begin
          CONTADOR:=CONTADOR+1;
          ENDV1[contador]:=i;
          ENDV2[contador]:=j;
          WEIGHT[contador]:=MATGEN[i,j];
         end
       END;
    END;
END;
PROCEDURE KRUSKAL(N,M:INTEGER; VAR ENDV1, ENDV2, WEIGHT :ARRM;
                  VAR CONNECT : BOOLEAN; VAR TEDGE1, TEDGE2 : ARRNM1;
                  VAR TWEIGHT : INTEGER);
VAR
I,LAST,U,V,R1,R2,ECOUNT,TCOUNT  : INTEGER;
INICIAL, FINAL ,COSTO           : INTEGER;
FATHER                          : ARRN;
 PROCEDURE  HEAP(FIRST, LAST : INTEGER);
 VAR J,K,TEMP1,TEMP2,TEMP3 : INTEGER;
 BEGIN
  J:=FIRST;
  WHILE J<= TRUNC(LAST/2) DO
  BEGIN
   IF (2*J < LAST) AND ( WEIGHT[2*J+1] < WEIGHT[2*J] ) THEN
    K:=2*J + 1
   ELSE
    K:=2*J;
   IF WEIGHT[K]<WEIGHT[J] THEN
   BEGIN
    TEMP1:=ENDV1[J];
    TEMP2:=ENDV2[J];
    TEMP3:=WEIGHT[J];
    ENDV1[J]:=ENDV1[K];
    ENDV2[J]:=ENDV2[K];
    WEIGHT[J]:=WEIGHT[K];
    ENDV1[K]:=TEMP1;
    ENDV2[K]:=TEMP2;
    WEIGHT[K]:=TEMP3;
    J:=K;
   END
   ELSE
    J:=LAST;
  END{WHILE}
 END;{HEAP}
```

```
FUNCTION FIND(I:INTEGER):INTEGER;
VAR PTR : INTEGER;
BEGIN
 PTR:=I;
 WHILE FATHER[PTR] > 0 DO
  PTR:=FATHER[PTR];
  FIND:=PTR
END;
PROCEDURE UNION(I,J:INTEGER);
VAR
X : INTEGER;
BEGIN
 X:=FATHER[I]+FATHER[J];
 IF FATHER[I]>FATHER[J] THEN
 BEGIN
  FATHER[I]:=J;
  FATHER[J]:=X
 END
 ELSE
 BEGIN
  FATHER[J]:=I;
  FATHER[I]:=X;
 END
END;{UNION}
BEGIN
 FOR I:=1 TO N DO                    {INICIALIZA EL ARREGLO PADRE}
  FATHER[I]:=-1;
  FOR I:=TRUNC(M/2) DOWNTO 1 DO      {HEAP INICIAL}
   HEAP(I,M);
   LAST:=M;
   ECOUNT:=0;
   TCOUNT:=0;
   TWEIGHT:=0;
   CONNECT:=TRUE;
   WHILE ((TCOUNT < N-1) AND (ECOUNT<M)) DO
    BEGIN          {QUITAR UN ARCO PARA EXAMINARLO}
     ECOUNT:=ECOUNT+1;
     U:=ENDV1[1];
     V:=ENDV2[1];      {SE CHECA SI U y V PERTENECEN AL MISMO ARBOL}
     R1:=FIND(U);
     R2:=FIND(V);
     IF R1<>R2 THEN    {INCLUIR EL ARCO (U,V) AL ARBOL DE EXP}
     BEGIN
      TCOUNT:=TCOUNT+1;
      UNION(R1,R2);
      TEDGE1[TCOUNT]:=U;
      TEDGE2[TCOUNT]:=V;
      TWEIGHT:=TWEIGHT+WEIGHT[1]
     END;
     ENDV1[1]:=ENDV1[LAST];
     ENDV2[1]:=ENDV2[LAST];
     WEIGHT[1]:=WEIGHT[LAST];
     LAST:=LAST-1;
     HEAP(1,LAST)
```

```
      END;
      IF TCOUNT<>N-1 THEN
       CONNECT:= FALSE;
  END;  {FIN PRINCIPAL}
 BEGIN {MINIMOARBOL}
  TRANSFO2(MATGEN,ENDV1,ENDV2,WEIGHT,N,M);
  KRUSKAL(N,M,ENDV1,ENDV2,WEIGHT,CONNECT,TEDGE1,TEDGE2,TWEIGHT);
  BEGIN
  IF CONNECT=TRUE THEN
   BEGIN
    VENTANA(1,1,79,24,1,0,3);
    VENTANA(8,2,73,4,3,0,0);
    WRITE(' SI EXISTE ARBOL DE EXPANSION ');
    WRITE('Y TIENE UN COSTO MINIMO DE ',TWEIGHT,' u.');
    VENTANA(24,4,61,7+N,3,0,0);
    WRITELN('ARCOS DEL ARBOL DE EXPANSION MINIMO');
    WRITELN(' (NODO INICIAL - NODO FINAL)  COSTO');
    FOR I:=1 TO N-1 DO
    BEGIN
    WRITELN('[',TEDGE1[i],' - ',TEDGE2[i],'] ',MATGEN[TEDGE1[i],TEDGE2[j]]);
    END;
  END
  ELSE
   WRITELN('LA GRAFICA ES DISCONEXA');
  END;
   DETENTE
 END;

PROCEDURE MAXIARBOL(MATGEN : ARRNN);{PROCEDIMIENTO PARA ENCONTRAR EL ARBOL
DE EXPANSIÓN MÁXIMO}
PROCEDURE MATRIZ(MATGEN : ARRNN; VAR W : ARRNN; VAR N :
INTEGER);  VAR
   i,j : INTEGER;
 BEGIN
   N:=MATGEN[NODOS,NODOS];
   I:=1;
    J:=1;
    FOR I:=1 TO N DO
     BEGIN
      FOR J:=1 TO N DO
       BEGIN
        IF MATGEN[i,j]=-1 THEN
        W[i,j]:=0
        ELSE
        W[I,J]:=MATGEN[i,j];
       END;
     END;
 END;
 PROCEDURE PRIM(N : INTEGER;
                VAR W                : ARRNN;
                VAR CONNECT          : BOOLEAN;
                VAR TEDGE1, TEDGE2 : ARRNM1;
                VAR TWEIGHT          : INTEGER);
VAR
```

```
 U,K,MIN,TCOUNT,I : INTEGER;
 NEAR,DIST        : ARRN;
BEGIN
 NEAR[1]:=0;
 FOR i:=2 TO N DO
 BEGIN
  NEAR[i]:=1;
  DIST[i]:=W[1,i]
 END;
 TCOUNT:=0;
 TWEIGHT:=0;
 CONNECT:=TRUE;
 WHILE (TCOUNT < N-1) AND (CONNECT = TRUE) DO
 BEGIN
  MIN:=-1;
  FOR K:=2 TO N DO
   IF NEAR[K] <> 0 THEN
    IF DIST[K] > MIN THEN
    BEGIN
     U:=K;
     MIN:=DIST[K]
    END;
  IF  DIST[U] <= -1 THEN
   CONNECT:=FALSE
  ELSE
  BEGIN
   TCOUNT:=TCOUNT+1;
   TEDGE1[TCOUNT]:=NEAR[U];
   TEDGE2[TCOUNT]:=U;
   TWEIGHT:=TWEIGHT+DIST[U];
   NEAR[U]:=0;
   FOR K:=2 TO N DO
    IF NEAR[K] <> 0 THEN
     IF W[K,NEAR[K]] < W[K,U] THEN
     BEGIN
      DIST[K]:=W[K,U];
      NEAR[K]:=U;
     END
  END
 END;
END;
BEGIN
 N:=MATGEN[NODOS,NODOS];
 MATRIZ(MATGEN,W,N);
 PRIM(N,W,CONNECT,TEDGE1,TEDGE2,TWEIGHT);
 VENTANA(1,1,79,24,1,0,3);
 VENTANA(4,2,75,4,3,0,0);
    WRITE(' SI EXISTE ARBOL DE EXPANSION MAXIMO');
    WRITE(' Y TIENE UN COSTO TOTAL DE ',TWEIGHT,' u.');
 VENTANA(24,4,61,7+N,3,0,0);
    WRITELN('ARCOS DEL ARBOL DE EXPANSION MAXIMO');
    WRITELN('  NODO INICIAL-NODO FINAL   COSTO');
    FOR i:=1 TO n-1 DO
     BEGIN
```

```
      WRITELN('( ',TEDGE1[i],' - ',TEDGE2[i],')  ',W[TEDGE1[i],TEDGE2[i]);
     END;
     VENTANA(38,22,79,24,3,0,0);
     DETENTE;
END;

PROCEDURE RUTACORTATODOS(MATGEN:ARRNN);{PROCEDIEMTO PARA ENCONTRAR LA RUTA
      MAS CORTA DE UN NODO ORIGEN A TODOS LOS DEMAS NODOS EN UN RED CONEXA}
VAR
 RASTREA : ARRN;
 f,b : INTEGER;
 PROCEDURE MATRIZ(MATGEN:ARRNN; VAR POINTER:ARRN1;
                  VAR ENDV,WT:ARRM;VAR N,M:INTEGER);
 VAR  i,j : INTEGER;
 BEGIN
  FOR i:=1 TO NODOS+1 DO
   POINTER[i]:=0;
  FOR i:=1 TO ARCOS DO
  BEGIN
   ENDV[i]:=0;
   WT[i]:=0;
  END;
   i:=1;j:=1;
   CONTADOR:=1;
   POINTER[i]:=CONTADOR;
    FOR i:=1 TO N DO
     BEGIN
      FOR j:=1 TO N DO
       BEGIN
        IF (MATGEN[i,j]<>-1) THEN
         BEGIN
          CONTADOR:=CONTADOR+1;
          ENDV[contador-1]:=j;
          WT[contador-1]:=MATGEN[i,j];
         END
       END;
       POINTER[i+1]:=CONTADOR;
      END;
      M:=CONTADOR-1;
    END;

PROCEDURE PDM(N, M, S, INF    : INTEGER;
              VAR POINTER     : ARRN1;
              VAR ENDV, WT    : ARRM;
              VAR DIST, PRED : ARRN);
VAR U, V, HEAD, NEXT, FIRST, LAST, J, NEWLABEL, TEMP : INTEGER;
    QUEUE                                            : ARRN;
BEGIN
 FOR I:=1 TO N DO
  BEGIN
   DIST[I]:=0;
   PRED[I]:=0;
  END;
  FOR V:=1 TO N DO
```

```
   BEGIN
    DIST[V]:= INF;
    PRED[V]:=-1;
    QUEUE[V]:=-1;
   END;
   DIST[S]:=0; HEAD:=S; U:=S;
   QUEUE[HEAD]:=INF;
   WHILE U<> INF DO
   BEGIN
    NEXT:=QUEUE[U];
    QUEUE[U]:=0;
    FIRST:=POINTER[U];
    LAST:=POINTER[U+1]-1;
     FOR J:=FIRST TO LAST DO
      BEGIN
       V:=ENDV[J];
       NEWLABEL:=DIST[U] + WT[J];
       IF DIST[V] > NEWLABEL THEN
        BEGIN
         PRED[V]:=U;
         DIST[V]:=NEWLABEL;
         TEMP:=NEXT;
          IF QUEUE[V] < 0 THEN
           BEGIN
            QUEUE[HEAD]:=V;
            HEAD:=V;
            QUEUE[HEAD]:=INF;
             IF TEMP = INF THEN
              TEMP:=V; NEXT:=TEMP;
           END
          ELSE
           IF QUEUE[V] = 0 THEN
            BEGIN
             QUEUE[V]:=TEMP; NEXT:=V;
            END
           ELSE NEXT:=TEMP;
        END
     END;
     U:=NEXT;
  END;
END;

BEGIN {RUTA CORTA TODOS}
 MATRIZ(MATGEN,POINTER,ENDV,WT,N,M);
 VENTANA(1,1,79,24,1,0,3);
 VENTANA(24,12,50,15,3,0,0);
 WRITELN('QUIEN ES EL NODO ORIGEN');
 WRITE('              ');
 READLN(CADENA);
 VALIDAR(CADENA,NUMERO);
 S:=NUMERO;
 PDM(N,M,S,INF,POINTER,ENDV,WT,DIST,PRED);
 CLRSCR;
VENTANA(1,1,79,24,1,0,3);
VENTANA(2,2,76,22,3,0,0);
```

```
WRITELN('          RUTAS MAS CORTAS DEL NODO ',S,' A LOS NODOS RESTANTES');
WRITELN('                                EN LA RED ',NOM);;
FOR i:=N DOWNTO 1 DO
 IF i<>S THEN
 BEGIN
  b:=1;
  ANTE:=i;
  RASTREA[b]:=i;
  REPEAT
   ANTE:=PRED[ANTE];
   b:=b+1;
   RASTREA[b]:=ANTE;
  UNTIL
   ANTE=S;
   WRITELN('DEL NODO ',S,' AL NODO ',i);
   WRITE('{');
  FOR f:=b DOWNTO 2 DO
   WRITE('(',RASTREA[f],',',RASTREA[f-1],' ),');
   WRITE('}');
   WRITELN(' A UN COSTO MINIMO DE ',DIST[i],' UNIDADES');
 END;
 DETENTE;
END;

PROCEDURE RUTAMASCORTA(MATGEN : ARRNN); {PROCEDIMIENTO PARA ENCONTRAR LA
        RUTA MAS CORTA DE UN NODO ORIGEN A OTRO NODO ESPECIFICO DESTINO}
VAR
 f       : INTEGER;
 RASTREA : ARRN;
PROCEDURE MATRIZ( MATGEN : ARRNN; VAR W : ARRNN;VAR N : INTEGER);
VAR
 i,j,b : INTEGER;

 BEGIN
   N:=MATGEN[NODOS,NODOS];
   I:=1;
    J:=1;
    FOR I:=1 TO NODOS DO
     BEGIN
      FOR J:=1 TO NODOS DO

       BEGIN
        IF i=j THEN
        W[i,j]:=0
        ELSE
        W[I,J]:=INF;
       END;
     END;
     i:=1;j:=1;
    FOR i:=1 TO NODOS DO
     FOR j:=1 TO NODOS DO
     IF MATGEN[i,j]<>-1 THEN
      W[i,j]:=MATGEN[i,j];
 END;
PROCEDURE DIJKSTRA(N,INF,S,T:INTEGER;VAR W : ARRNN; VAR PATH:BOOLEAN;
```

```
                    VAR FINAL:BOOLARRN; VAR DIST,PRED:ARRN);

VAR U,V,Y,RECENT,NEWLABEL,TEMP : INTEGER;
BEGIN
  V:=1;
  FOR V:= 1 TO N DO
   BEGIN
    DIST[V] := INF;
    FINAL[V]:=FALSE;
    PRED[V]:=-1;
   END;
   DIST[S]:=0;
   FINAL[S]:=TRUE;
   PATH:=TRUE;
   RECENT:=S;
   WHILE NOT FINAL[T] DO
    BEGIN
    FOR V:= 1 TO N DO
    IF (W[RECENT,V] < INF) AND (NOT FINAL[V]) THEN
    BEGIN
     NEWLABEL:=DIST[RECENT] + W[RECENT,V];
     IF NEWLABEL < DIST[V] THEN
      BEGIN
       DIST[V]:= NEWLABEL;
       PRED[V]:=RECENT;
      END
    END;
   TEMP:=INF;
   U:=1;
   FOR U:=1 TO N DO
     IF (NOT FINAL[U]) AND (DIST[U] <TEMP) THEN
      BEGIN
       Y:=0;
       Y:=U;
       TEMP:=DIST[U]
      END;
   IF TEMP < INF THEN
    BEGIN
     FINAL[Y]:=TRUE;
     RECENT := Y
    END
   ELSE
    BEGIN
     PATH:=FALSE;
     FINAL[T]:=TRUE
    END
   END;
END;
BEGIN
 VENTANA(1,1,79,24,1,0,3);
 VENTANA(26,10,55,13,3,0,0);
 WRITELN('  QUIEN ES EL NODO ORIGEN');
 WRITE('                ');
 READLN(CADENA);
 VALIDAR(CADENA,NUMERO);
```

```
 S:=NUMERO;
 WRITELN('  QUIEN ES EL NODO DESTINO');
 WRITE('                ');
 READLN(CADENA);
 VALIDAR(CADENA,NUMERO);
 T:=NUMERO;
 MATRIZ(MATGEN,W,N);
 DIJKSTRA(N,INF,S,T, W, PATH, FINAL, DIST,PRED);
    VENTANA(1,1,79,24,1,0,3);
   IF PATH=TRUE THEN
   VENTANA(3,3,76,17,3,0,0);
   GOTOXY(26,1);
   WRITELN('SI EXISTE TRAYECTORIA');
   GOTOXY(21,2);
   WRITELN('QUE UNA EL NODO  ',S,' CON EL NODO ',T);
   GOTOXY(14,3);
   WRITELN('LA DISTANCIA TOTAL RECORRIDA ES DE: ',DISTIT],' UNIDADES');
   GOTOXY(24,5);
   WRITELN('LA TRAYECTORIA SEGUIDA ES: ');
    BEGIN
    b:=1;
    ANTE:=t;
    RASTREA[b]:=t;
    REPEAT
     ANTE:=PRED[ANTE];
     b:=b+1;
     RASTREA[b]:=ANTE;
    UNTIL
     ANTE=S;
    WRITE('{');
    FOR f:=b DOWNTO 2 DO
     WRITE('(',RASTREA[f],',',RASTREA[F-1],'),');
   END;
   WRITELN('}');
   GOTOXY(24,7);
   WRITELN('Y EL VECTOR DE PREDECESORES ES: ');
   WRITE('{');
   FOR i:=1 TO N DO
   BEGIN
    WRITE(PRED[i],',');
   END;
   WRITE('}');
   DETENTE;
END;
PROCEDURE MAXFLOW(N,s,t,max : integer;
                  var capa, flow : arrnn);{PROCEDIENTO PAR ENCONTRAR EL
               VALOR DEL FLUJO MAXIMO DE UN NODO ORIGEN A OTRO DESTINO}
var
 lcapa, lflow, el          : arrnn;
 labels, layers, pred, succ : arrn;

 data, link                : arrmax;
 stpath                    : boolean;
 av, i, j, l, dif          : integer;
```

```
procedure ADD(var a: arrn;
              i, x : integer);
var
 p : integer;
 found : boolean;
 begin
  found:=false;
  p:=A[I];
  while (not found) and (p<>0) do
   if data[p] = x then
    found:= true
   else
    p:=link[p];
   if not found then
    begin
     p:=av;
     av:=link[av];
     data[p]:=x;
     link[p]:=a[i];
     a[i]:=p
    end
 end;

procedure DELETE(var a : arrn;
                 i, x : integer);
var
 p, q : integer;

 begin
  p:=a[i];
  if data[p]=x then
   a[i]:=link[p]
  else
  begin
   while data[p]<>x do
    begin
     q:=p;
     p:=link[p]
    end;
    link[q]:=link[p]
  end
 end;  {*Delete*}

 procedure LAYER(var l : integer;
                 var lcapa, el : arrnn;
                 var stpath : boolean);
  var
  i, j, p, q, x, y, v, w : integer;
 begin
  stpath:= true;
  for v:=1 to n do
   begin
    labels[v]:=-1;
    succ[v]:=0;
    pred[v]:=0;
```

```
end;
p:=av;
av:=link[av]; {*obtener un nodo*}
data[p]:=s;
link[p]:=0;
layers[1]:=p;
labels[s]:=1;
for x:=1 to n do
 for y:=1 to n do
  lcapa[x,y]:=capa[x,y];
   for x:=1 to n do
    for y:=1 to n do
    begin
     el[x,y]:=0;
     if flow[x,y] > 0 then
      begin
       lcapa[x,y]:=capa[x,y] - flow[x,y];
       lcapa[y,x]:=flow[x,y] + lcapa[y,x];
      end;
    end;
    i:=1;
    while (layers[i] <> 0) and (labels[t]=-1) do
    begin
     layers[i+1]:=0;
     p:=layers[i];
     while p<>0 do
     begin
      x:=data[p];
      for y:=1 to n do
       if ((labels[y]=-1) or (labels[y]= i+1)) and (lcapa[x,y] > 0) then
       begin
        add(layers, i+1, y);
        labels[y]:=i+1;
        add(succ,x,y);
        add(pred,y,x);
        el[x,y]:=1
       end
       else lcapa[x,y]:=0;
       p:=link[p]
     end;
     i:=i+1
    end;
    l:=i;
    if labels[t] = -1 then
    stpath:=false {*no existe trayectoria util*}
    else

    begin
     j:=i;
     while j <> 1 do
     begin
      p:=layers[j];
      while p <> 0 do
      begin
       w:=data[p];
       if (succ[w] = 0) and (w <> t) then
```

```
          begin
           q:=pred[w];
           while  q<>0 do
           begin
            x:=data[q];
            el[x,w]:=0;
            delete(succ,x,w);
            lcapa[x,w]:=0;
            q:=link[q];
           end;
           delete(layers,j,w);
           pred[w]:=0;
           labels[w]:=-1;
          end;
          p:=link[p]
         end;
         j:=j-1;
        end
       end
end;

procedure SATURATE(var l  : integer;
                   var lflow, el : arrnn);
var
 inpot, outpot, poten,inflow, outflow : arrn;
 v,x,y,i,j,k,p,r,rlayer                :integer;
 flag                                  : boolean;

 procedure REFNODE(var l,r,rlayer : integer);
 var
  v : integer;

  begin
   poten[s]:=outpot[s];
   poten[t]:=inpot[t];
   if poten[s] < poten[t] then
   begin
    r:=s;
    rlayer:=l
   end
   else
   begin
    r:=t;
    rlayer:=l;
   end;
   for v:=1 to n do
    if (labels[v]<> -1) and (v<>s) and (v<>t) then
    begin
     if inpot[v] < outpot[v] then
     poten[v]:=inpot[v]
     else poten[v]:=outpot[v];
     if poten[v] < poten[r] then
     begin
      r:=v;
      rlayer:=labels[v]
```

```
    end
   end
 end;
procedure PUSH(var i: integer);
var
 p,q,u,v,avacap : integer;
begin
 p:=layers[i];
 while p <> 0 do
 begin
  u:=data[p];
  if outflow[u] > 0 then
  begin
   q:=succ[u];
   while (outflow[u] > 0) and (q<>0) do
   begin
    v:=data[q];
    if labels[v] <> -1 then
    begin
     avacap:=lcapa[u,v] - Lflow[u,v];
     if avacap > 0 then
     begin {*envia el minimo entre avacap y outflow[u] atravezando (u,v)*}
      if avacap > outflow[u] then
      avacap:=outflow[u];
      lflow[u,v]:=lflow[u,v] + avacap;
      outflow[u]:= outflow[u] - avacap;
      outflow[v]:=outflow[v] + avacap;
      outpot[u] := outpot[u] - avacap;
      inpot[v]:=inpot[v] - avacap;
     end
    end;
    q:=link[q]
   end
  end;
  p:=link[p];
 end;
end;
procedure PULL(var j : integer);
var
 p,q,u,v,avacap : integer;
 begin
  p:=layers[j];
  while p<>0 do
  begin
   u:=data[p];
   if inflow[u] > 0 then
   begin
    q:=pred[u];
    while (inflow[u] > 0) and (q<>0)do
    begin
     v:=data[q];
     if labels[v] <> -1 then
     begin
      avacap:=lcapa[v,u] - lflow[v,u];
      if avacap > 0 then
```

```
      begin {*envia el min(avacap, inflow[u]) a traves de (v,u)*}
       if avacap > inflow[u] then
       avacap:=inflow[u];
       lflow[v,u]:=lflow[v,u] + avacap;
       inflow[u]:=inflow[u] - avacap;
       inflow[v]:=inflow[v] + avacap;
       outpot[v]:=outpot[v] - avacap;
       inpot[u]:=inpot[u]-avacap
      end
     end;
     q:=link[q]
    end
   end;
   p:=link[p]
  end
 end; {* fin de Pull*}
begin
for v:=1 to n do
 begin
  inpot[v]:=0;
  outpot[v]:=0;
 end;
 for i:=1 to n do
  for j:=1 to n do
  if el[i,j] = 1 then
  begin
   inpot[j]:=inpot[j] + lcapa[i,j];
   outpot[i]:=outpot[i] + lcapa[i,j];
  end;
 for i:=1 to n do
  for j:=1 to n do
   lflow[i,j]:=0;
   flag:=true;  {*GL esta sin saturar*}
   While flag do
   begin
    refnode(l,r,rlayer);
    if poten[r]<>0 then
    begin
     inflow[r]:=poten[r];
     outflow[r]:=poten[r];
     for v:=1 to n do
      if (labels[v]<>-1) and (v<>r) then
      begin
       inflow[v]:=0;
       outflow[v]:=0;
      end;
      for k:=rlayer to l-1 do
       push(k);
       for j:=rlayer downto 2 do
        pull(j)
    end;
    if (poten[r]<>0) or ((r<>s) and (r<>t)) then
    begin
     labels[r]:=-1;
     p:=succ[r];
```

```
      while p<>0 do
      begin
       y:=data[p];
       inpot[y]:=inpot[y]-(lcapa[r,y]-lflow[r,y]);
       el[r,y]:=0;
       delete(succ,r,y);
       delete(pred,y,r);
       p:=link[p]
      end;
      p:=pred[r];
      while p<>0 do
      begin
       x:=data[p];
       outpot[x]:=outpot[x] - (lcapa[x,r] - lflow[x,r]);
       el[x,r]:=0;
       delete(succ,x,r);
       delete(pred,r,x);
       p:=link[p];
      end;
     end
     else flag:=false;  {*gl esta saturada*}
    end;            {*fin del while*}
end; {*fin de saturate*}
procedure INITIALIZE(max : integer);
var i : integer;
begin
 for i:=1 to max do
  link[i]:=i + 1;
  av:=1
end;
BEGIN {*FLUJO MAXIMO*}
 for i:=1 to n do
  for j:=1 to n do
   flow[i,j]:=0;
   initialize(max);  {*produce una lista ligada*}
   layer(1,lcapa,el,stpath);
   while stpath do
   begin
    saturate(1,lflow,el);
    for i:=1 to n do
    for j:=1 to n do
     if lflow[i,j] > 0 then
     begin
      dif:=lflow[i,j]-flow[j,i];
      if dif > 0 then
      begin
       flow[i,j]:=flow[i,j]+dif;
       flow[j,i]:=0;
      end
      else flow[j,i]:=-dif;
     end;
     initialize(max);
     layer(1,lcapa,el,stpath)
   end;
END; {*MAXFLOW*}
```

```
PROCEDURE FLUJOMAX(MATGEN:ARRNN);
VAR      TECLA : CHAR;
    TOTAL, MAX : INTEGER;

 PROCEDURE MATRIZ(MATGEN:ARRNN;VAR CAPA:ARRNN;VAR N,MAX:INTEGER);
 VAR CONTADOR,i,j:INTEGER;
  BEGIN
   i:=1;j:=1;
   CONTADOR:=0;
   N:=MATGEN[NODOS,NODOS];
   FOR i:=1 TO N DO
    FOR j:=1 TO N DO
      CAPA[i,j]:=0;
       i:=1;j:=1;
    FOR i:=1 TO N DO
     FOR j:=1 TO N DO
     BEGIN
      IF (MATGEN[i,j]<>-1) THEN
       CAPA[i,j]:=MATGEN[i,j];
       CONTADOR:=CONTADOR+1;
     END;
   BEGIN
    MAX:=3*CONTADOR;
   END
END;
{PROCEDURE MAXFLOW(N,s,t,max : integer;
                   var capa, flow : arrnn);}
 BEGIN {PROGRAMA PRINCIPAL DE FLUJOMAX}
  MATRIZ(MATGEN,CAPA,N,MAX);
  REPEAT
   VENTANA(1,1,79,24,1,0,3);
   VENTANA(24,5,60,9,3,0,0);
   CLRSCR;
   WRITELN('    QUIEN ES EL NODO ORIGEN?');
   WRITE('                    ');
   READLN(CADENA);
   VALIDAR(CADENA,NUMERO);
   S:=NUMERO;
   IF S>N THEN
    BEGIN
     SOUND(220);
     DELAY(200);
     NOSOUND;
     WRITE('SU RED TIENE SOLO ',N,' NODOS');
     DETENTE;
     CADENA:='';
    END;
  UNTIL S<=N;
  REPEAT
   VENTANA(1,1,79,24,1,0,3);
   VENTANA(24,5,60,9,3,0,0);
   CLRSCR;
   WRITELN('     QUIEN ES EL NODO DESTINO?');
   WRITE('                      ');
   READLN(CADENA);
```

```
   VALIDAR(CADENA,NUMERO);
   T:=NUMERO;
   IF T>N THEN
    BEGIN
     SOUND(220);
     DELAY(200);
     NOSOUND;
     WRITE('SU RED TIENE SOLO ',N,' NODOS');
     DETENTE;
    END;
   UNTIL T<=N;
  MAXFLOW(N,S,T,MAX,CAPA,FLOW);
  VENTANA(1,1,79,24,1,0,3);
  TOTAL:=0;
    FOR i:=1 TO N DO {COLUMNAS}
     FOR j:=1 TO N DO  {RENGLONES}
     BEGIN
      GOTOXY(3*j+12,i);
       WRITE(FLOW[I,J]);
       TOTAL:=TOTAL+FLOW[i,j];
     END;
  VENTANA(1,1,14,12,3,0,0);
    WRITELN(' MATRIZ');
    WRITELN('   QUE');
    WRITELN('REPRESENTA');
    WRITELN(' EL FLUJO');
    WRITELN(' MAXIMO');
    WRITELN('ASOCIADO A');
    WRITELN(' LOS ARCOS');
    WRITELN(' DE LA RED');
    WRITELN(NOM);
    WRITE(TOTAL);
    DETENTE;
  CLRSCR;
END; {PROGRAMA PRINCIPAL DE FLUJO MAXIMO}
{FLUJO A COSTO MINIMO}
PROCEDURE FLUJOACOSTOMINIMO(MATGEN:ARRNN);{PROCEDIENTO PARA ENCONTRAR EL
COSTO MINIMO PARA ENVIAR UNA CIERTA CANTIDAD DE FLUJO DE UN NODO ORIGEN A
OTRO DESTINO}
VAR MAXIMO,
TOTALCOST, TARGETFLOW : INTEGER;    TECLA                          : CHAR;
 PROCEDURE MATRIZ(MATGEN:ARRNN;VAR CAPA:ARRNN); {TRANSFORMA LA MATRIZ}
 VAR CONTADOR,i,j:INTEGER;                {MATGEN A LA MATRIZ CAPACIDADES}
  BEGIN
   N:=MATGEN[NODOS,NODOS];
   CONTADOR:=0;
   FOR i:=1 TO N DO
    FOR j:=1 TO N DO
     CAPA[i,j]:=0;
   FOR i:=1 TO N DO
    FOR j:=1 TO N DO
    BEGIN
     IF (MATGEN[i,j]<>-1) THEN
      BEGIN
       CAPA[i,j]:=MATGEN[i,j];
```

```
      CONTADOR:=CONTADOR+1;
     END;
   END;
 END;

PROCEDURE CAPTURA(VAR COST:ARRNN); {CAPTURA UNA MATRIZ DE COSTOS}
                                   {Y DA LA OPCION PARA SALVARLA}
VAR
CAPACIDAD,I,J : INTEGER;
SALIR         : BOOLEAN;
TECLA         : CHAR;

 PROCEDURE SALVA(CAPA:ARRNN);
  VAR
   SN  : STRING;        {  NOMBRE DEL ARCHIVO     }
   VV  : BOOLEAN;
  BEGIN
   GOTOXY(17,12);
   VENTANA(1,1,79,24,1,0,3);
   VENTANA(25,6,50,10,3,0,0);
   CLRSCR;
   NOM:='';
   WRITELN('  NOMBRE CON EL QUE SE');
   WRITE  ('      DESEA SALVAR: ');
   LEE_S(15,3,8,NOM);
   IF NOM='' THEN
    BEGIN
     SOUND(220);
     DELAY(200);
     NOSOUND;
     EXIT
    END
   ELSE
    BEGIN
     NOM:=NOM+'.FLW';
     ASSIGN(ARCH,NOM);
     REWRITE(ARCH);           { ABRE EL ARCHIVO PARA ESCRITURA            }
     WRITE (ARCH,COST);       { ESCRIBE EN EL PRIMER REGISTRO DEL ARCHIVO  }
    END;
  END;
BEGIN
VENTANA(1,1,79,24,1,0,3);
I:=1; J:=1;
FOR I:=1 TO NODOS DO
 BEGIN
  FOR J:=1 TO NODOS DO
   BEGIN
    COST[I,J]:=INF;
   END;
 END;
N:=MATGEN[NODOS,NODOS];     {AQUI SE GUARDA EL NUMERO DE NODOS}
FOR i:=1 TO N DO
 FOR j:=1 TO N DO
  BEGIN
   IF MATGEN[i,j]<>-1 THEN
```

```
      BEGIN
       CADENA:='';
       VENTANA(32,14,65,17,3,0,0);
       WRITELN(' COSTO ASIGNADO AL ARCO (',i,', ',j,')?');
       WRITE('                    ');READLN(CADENA);
       VALIDAR(CADENA,NUMERO);
       COST[i,j]:=NUMERO;
      END;
    END;
   VENTANA(1,1,79,24,1,0,3);
     FOR i:=1 TO N DO {COLUMNAS}
      FOR j:=1 TO N DO  {RENGLONES}
      BEGIN
       GOTOXY(3*j+12,i);
       IF COST[i,j]<>INF THEN
        WRITE(COST[I,J])
       ELSE
        WRITE('*');
      END;
     VENTANA(1,1,15,7,3,0,0);
     WRITELN(' MATRIZ');
     WRITELN('DE COSTOS');
     WRITELN('ASSOCIADA');
     WRITELN('A LA RED ');
     WRITE(NOM);
     TECLA:='+';
     VENTANA(38,22,79,24,3,0,0);
     WRITE('       DESEAS SALVARLA? S/N');
     TECLA:=READKEY;
     IF (TECLA='s') OR (TECLA='S') THEN
     SALVA(CAPA);
END;
 PROCEDURE LEE2(VAR COST:ARRNN);{PROCEDIENTO PARA LEER UNA MATRIZ DE COSTOS}
 VAR
  CUENTA : INTEGER;
  ARCHL  : FILE OF ARRNN;  {  DECLARACION DE BUFFER  }
  NOM    : STRING;         {  NOMBRE DEL ARCHIVO     }
 BEGIN
 VENTANA(1,1,79,24,1,0,3);
  NOM:=DIRECTORIO('FLW',1);{ DESPLIEGA EL DIRECTORIO PARA ESCOGER UN ARCHIVO }
  IF NOM<>'@@@' THEN
   BEGIN
    ASSIGN(ARCHL,NOM);          { ASIGNA UN BUFFER A ESTE NOMBRE DE ARCHIVO  }
    RESET(ARCHL);               { ABRE EL ARCHIVO PARA LECTURA }
    READ (ARCHL,COST);          { LEE EL PRIMER REGISTRO DEL ARCHIVO  }
  END;
  N:=MATGEN[NODOS,NODOS];
  VENTANA(1,1,79,24,1,0,3);
  FOR i:=1 TO N DO {COLUMNAS}
   FOR j:=1 TO N DO  {RENGLONES}
    BEGIN
     GOTOXY(3*j+12,i);
     IF COST[i,j]<>INF THEN
      WRITE(COST[I,J])
     ELSE
```

```
      WRITELN('*');
    END;
  VENTANA(1,1,14,7,3,0,0);
  WRITELN('   MATRIZ');
  WRITELN('DE COSTOS');
  WRITELN('  ASOCIADA');
  WRITELN('A LA RED:');
  WRITE(NOM);
  VENTANA(42,22,79,24,3,0,0);
  TECLASO:=' ';
  WRITE('DESEA CORREGIR ALGUN VALOR? (S/N)'); TECLASO:=READKEY;
  IF (TECLASO='s') OR (TECLASO='S') THEN
   BEGIN
    CLRSCR;
    k:=1;i:=0;j:=0;CUENTA:=0;
    WRITE('CUANTOS VALORES VA A MODIFICAR? ');
    READLN(CADENA);
    VALIDAR(CADENA,NUMERO);
    CUENTA:=NUMERO;
    FOR k:=1 TO CUENTA DO
     BEGIN
      CADENA:='';
      WRITE('NODO INICIAL A CORREGIR? ');
      READLN(CADENA);
      VALIDAR(CADENA,NUMERO);
      i:=NUMERO;
      WRITE('NODO FINAL A CORREGIR? ');
      READLN(CADENA);
      VALIDAR(CADENA,NUMERO);
      j:=NUMERO;
      WRITE('COSTO ASIGNADO AL ARCO (',i,', ',j,')');
      READLN(CADENA);
      VALIDAR(CADENA,NUMERO);
      COST[i,j]:=NUMERO;
     END;
    END
    ELSE
    EXIT;
END;

PROCEDURE BUSACKER(N,S,T,INF,TARGETFLOW : INTEGER;
                   VAR COST,CAPA,FLOW : ARRNN;
                        VAR TOTALCOST : INTEGER);
VAR
 MCOST : ARRNN;
 DIST,PRED : ARRN;
 STPATH : BOOLEAN;
 I,J,CURRENTFLOW,DELTA,D:INTEGER;

 PROCEDURE PDM(N,S,T,INF : INTEGER;
                   VAR A : ARRNN;
              VAR STPATH : BOOLEAN;
          VAR DIST, PRED : ARRN);

 VAR
```

```
 QUEUE                          : ARRN;
 U,V,HEAD,NEXT,TEMP,NEWLABEL : INTEGER;
BEGIN
 FOR V:=1 TO N DO
  BEGIN
   DIST[V]:=INF;
   PRED[V]:=-1;
   QUEUE[V]:=-1;
  END;
 DIST[S]:=0;
 U:=S;
 HEAD:=S; QUEUE[HEAD]:=INF;
 WHILE U<> INF DO
 BEGIN
  NEXT:=QUEUE[U];
  QUEUE[U]:=0;
  FOR V:=1 TO N DO
   IF A[U,V]<>INF THEN
   BEGIN
    NEWLABEL:=DIST[U]+A[U,V];
    IF DIST[V]>NEWLABEL THEN
    BEGIN
     DIST[V]:=NEWLABEL;
     PRED[V]:=U;
     TEMP:=NEXT;
     IF QUEUE[V]<0 THEN
     BEGIN
      QUEUE[HEAD]:=V;
      HEAD:=V;QUEUE[HEAD]:=INF;
      IF TEMP=INF THEN
      TEMP:=V;
      NEXT:=TEMP;
     END
     ELSE
     IF QUEUE[V]=0 THEN
     BEGIN
      QUEUE[V]:=TEMP;
      NEXT:=V;
     END
      ELSE NEXT:=TEMP;
     END
    END;
   U:=NEXT
  END;
  IF PRED[T]=-1 THEN STPATH:=FALSE
END;

BEGIN  {*PRINCIPAL*}
 FOR I:=1 TO N DO
  FOR J:=1 TO N DO
  BEGIN
   FLOW[I,J]:=0;
   MCOST[I,J]:=COST[I,J];
  END;
  CURRENTFLOW:=0;
```

```
    STPATH:=TRUE;
    WHILE (CURRENTFLOW < TARGETFLOW) AND (STPATH) DO
    BEGIN
     PDM(N, S, T, INF, MCOST, STPATH, DIST, PRED);
     IF STPATH THEN
     BEGIN
      DELTA:=INF;
      I:=T;
      REPEAT
       J:=I;
       I:=PRED[J];
       IF CAPA[I,J]>0 THEN
       BEGIN
        D:=CAPA[I,J]-FLOW[I,J];
        IF D<DELTA THEN DELTA:=D
       END
       ELSE
       IF FLOW[J,I]<DELTA THEN
       DELTA:=FLOW[J,I]
      UNTIL I=S;
      IF CURRENTFLOW + DELTA > TARGETFLOW THEN
       DELTA:=TARGETFLOW - CURRENTFLOW;
      I:=T;
      REPEAT
       J:=I;
       I:=PRED[J];
       IF CAPA[I,J]>0 THEN
       BEGIN
        FLOW[I,J]:=FLOW[I,J] + DELTA;
        IF FLOW[I,J]=CAPA[I,J] THEN
        MCOST[I,J]:=INF;
        MCOST[J,I]:=-COST[I,J]
       END
       ELSE
       BEGIN
        FLOW[J,I]:=FLOW[J,I]-DELTA;
        MCOST[J,I]:=COST[J,I];
        IF FLOW[J,I]=0 THEN
        MCOST[I,J]:=COST[I,J]
       END
      UNTIL I=S;
      CURRENTFLOW:=CURRENTFLOW + DELTA
     END
    END;
    TOTALCOST:=0;
    FOR I:=1 TO N DO
     FOR J:=1 TO N DO
      IF FLOW[I,J]>0 THEN
       TOTALCOST:=TOTALCOST+COST[I,J]*FLOW[I,J];
END; {FIN DE BUSAKER}
BEGIN {FLUJO A COSTO MINIMO}
 MATRIZ(MATGEN,CAPA);
 VENTANA(1,1,79,24,1,0,3);
 VENTANA(24,12,61,15,3,0,0);
 WRITELN('DESEA CAPTURAR UNA MATRIZ DE COSTOS');
```

```
WRITE('  O RECUPERAR ALGUNA SALVADA? C/R ');
 READLN(TECLA);
 IF (TECLA='C') OR (TECLA='c') THEN
  CAPTURA(COST)
 ELSE
  LEE2(COST);
 BEGIN
  FOR i:=1 TO N DO
   FOR j:=1 TO N DO
   BEGIN
    IF ((COST[i,j]=INF) AND (CAPA[i,j]<>0)) THEN
     BEGIN
      SOUND(220);
      DELAY(200);
      NOSOUND;
      VENTANA(1,1,79,24,1,0,3);
      VENTANA(18,12,61,15,3,0,0);
      WRITELN('   LO SIENTO SU MATRIZ DE COSTOS NO ES');
      WRITE(' COMPATIBLE CON SU MATRIZ DE CAPACIDADES');
      DETENTE;
      EXIT
     END
   END;
 END;
VENTANA(1,1,79,24,1,0,3);
VENTANA(24,15,54,18,3,0,0);
REPEAT
 WRITELN('   CUAL ES EL NODO FUENTE?  ');
 WRITE  ('                ');
 READLN(CADENA);
 VALIDAR(CADENA,NUMERO);
 IF NUMERO>N THEN
  BEGIN
   SOUND(220);
   DELAY(200);
   NOSOUND;
  END
 ELSE
  S:=NUMERO;
UNTIL
 NUMERO<N+1;
CLRSCR;
REPEAT
 WRITELN('   CUAL ES EL NODO DESTINO? ');
 WRITE  ('                ');
 READLN(CADENA);
 VALIDAR(CADENA,NUMERO);
 IF NUMERO>N THEN
  BEGIN
   SOUND(220);
   DELAY(200);
   NOSOUND;
  END
 ELSE
  T:=NUMERO;
```

```
 UNTIL
  NUMERO<N+1;
 CLRSCR;
 WRITELN(' UNIDADES DE FLUJO A ENVIAR?');
 WRITE  ('               ');
 READLN(CADENA);
 VALIDAR(CADENA,NUMERO);
 TARGETFLOW:=NUMERO;
 BUSACKER(N,S,T,INF,TARGETFLOW,COST,CAPA,FLOW,TOTALCOST);
     BEGIN
       VENTANA(1,1,79,24,1,0,3);
            FOR i:=1 TO N DO {COLUMNAS}
             FOR j:=1 TO N DO  {RENGLONES}
             BEGIN
              GOTOXY(3*j+12,i);
              WRITE(FLOW[I,J])
             END;
            VENTANA(1,1,14,11,3,0,0);
            WRITELN('FLUJOS ASIG');
            WRITELN('NADOS  A LA');
            WRITELN('    RED:    ');
            WRITELN(NOM);
            WRITELN('A  UN COSTO');
            WRITELN('MINIMO  DE:');
            WRITELN('    ',TOTALCOST);
            WRITELN(' UNIDADES ');
            WRITE  ('MONETARIAS');
            DETENTE;
         END;
 END;   {FLUJO A COSTO MINIMO}
{*FLUJO A COSTO MINIMO*}
PROCEDURE INTRODUCIR(VAR MATGEN : ARRNN);{PROCEDIENTO PARA COLOCAR  EN  MEMORI
UNA RED EN FORMA MATRICIAL}
VAR
CAPACIDAD,I,J : INTEGER;
SALIR, SALIR1 : BOOLEAN;
TECLA : CHAR;
BEGIN
 CADENA:=' ';
 I:=1;
 J:=1;
 SALIR:=TRUE;
 FOR I:=1 TO NODOS DO
 BEGIN
         FOR J:=1 TO NODOS DO
         BEGIN
                 MATGEN[I,J]:=-1;
         END;
 END;
 CONTINUA:=TRUE;
 WHILE CONTINUA DO
 BEGIN
  CLRSCR;
  WRITELN;
  VENTANA(1,1,79,24,1,0,3);
```

```
  VENTANA(23,6,55,10,3,0,0);
  WRITELN('  CUANTOS NODOS TIENE LA RED');
  WRITE('                 ');
  READLN(CADENA);
  VALIDAR(CADENA,NUMERO);
  IF NUMERO>NODOS-1 THEN
  BEGIN
   VENTANA(23,6,62,10,15,0,4+BLINK);
   SOUND(220);
   DELAY(200);
   NOSOUND;
   CLRSCR;
   WRITELN(' LO SIENTO, ESTE SISTEMA SOLO TRABAJA');
   WRITELN('    CON REDES DE A LO MAS ',NODOS,' NODOS');
   WRITE('     CONTINUA CON OTRO DATO? S/N ');
   TECLASO:=READKEY;
   IF (TECLASO='N') OR (TECLASO='n') THEN
   BEGIN
    CLRSCR;
    ADIOS;

    EXIT;
   END
   ELSE
   CONTINUA:=TRUE;
  END
  ELSE
  CONTINUA:=FALSE;
  SALIR1:=TRUE;
 END;
IF (SALIR1=TRUE) AND (CONTINUA=FALSE) THEN
BEGIN
 N:=NUMERO;
 MATGEN[NODOS,NODOS]:=N;    {AQUI SE GUARDA EL NUMERO DE NODOS}
 VENTANA(1,1,79,24,1,0,3);
 VENTANA(15,5,50,10,3,0,0);
    CLRSCR;
    WRITELN;
    WRITELN('   INTRODUZCA LAS ENTRADAS DE');
    WRITELN(' LA MATRIZ DE CAPACIDADES W[i,j]');
    SALIR:=TRUE;
    VENTANA(32,14,61,19,3,0,0);
    REPEAT
     REPEAT
      CLRSCR;
      WRITELN;
      WRITELN(' (NODO INICIAL DEL ARCO QUE');
      WRITELN('     DESEA INTRODUCIR?');
      WRITE('              ');
      READLN(CADENA);
      VALIDAR(CADENA,NUMERO);
      I:=NUMERO;
      IF (N<I) THEN
      BEGIN
       CLRSCR;
       WRITELN;
```

```
    WRITELN('  SU RED TIENE A LO MAS ',N);
    WRITELN('  NODOS, RECTIFIQUE SU DATO');
    WRITE('                ');
    SOUND(220);
    DELAY(200);
    NOSOUND;
    READLN;
   END
  UNTIL
  i<=N;
  VENTANA(32,14,61,19,3,0,0);
  REPEAT
   CLRSCR;
   WRITELN;
   WRITELN('  (NODO FINAL DEL ARCO QUE');
   WRITELN('       DESEA INTRODUCIR?');
   WRITE('                ');
   READLN(CADENA);
   VALIDAR(CADENA,NUMERO);
   J:=NUMERO;
   IF j>N THEN
   BEGIN
    CLRSCR;
    SOUND(220);
    DELAY(200);
    NOSOUND;
    WRITELN;
    WRITELN('  SU RED TIENE A LO MAS ',N);
    WRITELN('  NODOS, RECTIFIQUE SU DATO');
    WRITE('                ');
    READLN;
   END
  UNTIL
   j<=N;
   VENTANA(32,14,61,19,3,0,0);
   CLRSCR;
   WRITELN;
   WRITELN('    (CAPACIDAD ASIGNADA A');
   WRITELN('          ESTE ARCO?');
   WRITE('              ');
   READLN(CADENA);
   VALIDAR(CADENA,NUMERO);
   CAPACIDAD:=NUMERO;
 MATGEN[i,j]:=CAPACIDAD;
 VENTANA(32,14,61,19,3,0,0);
 CLRSCR;
 WRITELN;
 WRITELN('(DESEA INTRODUCIR OTRO DATO?');
 WRITELN('           (S/N)');
 WRITE('              ');
 TECLA:=READKEY;
 IF (TECLA='N') OR (TECLA='n') THEN
 SALIR:=FALSE;
UNTIL
 SALIR=FALSE;
```

```
   VENTANA(1,1,79,24,1,0,3);
    FOR i:=1 TO N DO {COLUMNAS}
     FOR j:=1 TO N DO  {RENGLONES}
     BEGIN
      GOTOXY(3*j+12,i);
      IF MATGEN[i,j]<>-1 THEN
       WRITE(MATGEN[I,J])
      ELSE WRITE('*');
     END;
    VENTANA(1,1,13,6,3,0,0);
    CLRSCR;
    WRITELN;
    WRITELN('  MATRIZ');
    WRITELN(' ASOCIADA');
    WRITELN(' A LA RED');
    WRITELN('  CARGADA');
    WRITE('      ');
    DETENTE;
 END
 ELSE
END;
{***DIRECTORIO***}
PROCEDURE LEE(VAR MATGEN :ARRNN; VAR N:INTEGER; VAR NOM :  STRING);
{PROCEDIEMTO PARA LEER UNA MATRIZ DE CAPACIDADES PREVIAMENTE ALMACENADA EN
DISCO}
 VAR
  ARCHL: FILE OF ARRNN; {  DECLARACION DE BUFFER  }
  {NOM  : STRING;          NOMBRE DEL ARCHIVO     }
  K, CUENTA    : INTEGER;
   BEGIN
     NOM:=DIRECTORIO('RED',1);{ DESPLIEGA EL DIRECTORIO P/ ESCOGER ARCHIVO}
     IF NOM<>'@@@' THEN
      BEGIN
        ASSIGN(ARCHL,NOM);   { ASIGNA UN BUFFER A ESTE NOMBRE DE ARCHIVO}
        RESET(ARCHL);        { ABRE EL ARCHIVO PARA LECTURA }
        READ (ARCHL,MATGEN); { LEE EL PRIMER REGISTRO DEL ARCHIVO  }
      END;
     N:=MATGEN[NODOS,NODOS];
       VENTANA(1,1,79,24,1,0,3);
            FOR i:=1 TO N DO {COLUMNAS}
             FOR j:=1 TO N DO  {RENGLONES}
             BEGIN
              GOTOXY(3*j+12,i);
              IF MATGEN[i,j]<>-1 THEN
              WRITE(MATGEN[I,J])
              ELSE
              WRITE('*');
             END;
            VENTANA(1,1,14,9,3,0,0);
            WRITELN('   MATRIZ');
            WRITELN('    DE ',N);
            WRITELN('   NODOS');
            WRITELN('  ASOCIADA');
            WRITELN('  A LA RED');
            WRITELN('   LEIDA:');
```

```
            WRITE(NOM);
            VENTANA(42,22,79,24,3,0,0);
            TECLASO:=' ';
            WRITE('DESEA CORREGIR ALGUN VALOR? (S/N)');TECLASO:=READKEY;
            IF (TECLASO='S') OR (TECLASO='s') THEN
             BEGIN
              CLRSCR;
              K:=1;i:=0;j:=0;CUENTA:=0;
              WRITE('CUANTOS VALORES VA A MODIFICAR? ');
              READLN(CADENA);
              VALIDAR(CADENA,NUMERO);
              CUENTA:=NUMERO;
              FOR K:=1 TO CUENTA DO
              BEGIN
               CADENA:='';
               WRITE('NODO INICIAL A CORREGIR? ');
               READLN(CADENA);
               VALIDAR(CADENA,NUMERO);i:=NUMERO;
               WRITE('NODO FINAL A CORREGIR? ');
               READLN(CADENA);
               VALIDAR(CADENA,NUMERO);
               j:=NUMERO;
               WRITE('CAPACIDAD ASOCIADA AL ARCO (',i,', ',j,')');
               READLN(CADENA);
               VALIDAR(CADENA,NUMERO);
               MATGEN[i,j]:=NUMERO;
              END;
             END
            ELSE
            EXIT;
    END;
{***DIRECTORIO***}
{****SALVAR****}
  PROCEDURE GUARDA(MATGEN:ARRNN);{PROCEDIENTO PARA GUARDAR EN DISCO UNA RED
                                  EN FORMA DE MATRIZ}
   BEGIN
     GOTOXY(17,12);
     VENTANA(1,1,79,24,1,0,3);
     VENTANA(25,6,50,11,3,0,0);
     CLRSCR;
     NOM:='';
     WRITELN;
     WRITELN(' NOMBRE CON EL QUE SE');
     WRITE('DESEA SALVAR: ');
     LEE_S(15,3,8,NOM);
     IF NOM='' THEN
     BEGIN
     SOUND(220);
     DELAY(200);
     NOSOUND;
     EXIT;
     END
     ELSE
     BEGIN
     NOM:=NOM+'.RED';
```

```
     ASSIGN(ARCH,NOM);  { ASIGNA UN BUFFER A ESTE NOMBRE DE ARCHIVO  }
     REWRITE(ARCH);           { ABRE EL ARCHIVO PARA ESCRITURA            }
     WRITE(ARCH,MATGEN);       { ESCRIBE EN EL PRIMER REGISTRO DEL ARCHIVO  }
     CLOSE(ARCH);
    END;
   END;
{****SALVAR****}
PROCEDURE ADIOS;{PROCEDIENTO PARA FINALIZAR SESION}
 VAR
  SI_O_NO : BOOLEAN;
  TECLASO : CHAR;
   BEGIN
    SI_O_NO:=TRUE;
    VENTANA(1,1,79,24,1,0,3);
    VENTANA(15,8,65,15,3,0,0);
    REPEAT
    CLRSCR;
     WRITELN('       (Quiere finalizar su sesi"n? S / N');
     WRITE('                          ');
     TECLASO:=READKEY;
     IF ((TECLASO<>'S') OR (TECLASO<>'s'))
        XOR ((TECLASO<>'N') OR (TECLASO<>'n'))
     ELSE
     BEGIN
      IF (TECLASO='S') OR (TECLASO='s') THEN
       SI_O_NO:=FALSE
      ELSE
       IF (TECLASO='N') OR (TECLASO='n') THEN
       BEGIN
        SI_O_NO:=FALSE;
        OPCION:=0;
       END;
     END;
    UNTIL
    SI_O_NO=FALSE;
    IF (TECLASO='s') OR (TECLASO='S') THEN
    BYE;
   END;
 {*PROGRAMA PRINCIPAL*}
BEGIN
PORTADA;
 REPEAT
 VENTANA(1,1,79,24,0,15,0);
 gotoxy(27,5);
 letmenu[1]:='          MENU PRINCIPAL          ';
 letmenu[2]:=' ARBOL DE EXPANSION MINIMO        ';
 letmenu[3]:=' ARBOL DE EXPANSION MAXIMO        ';
 letmenu[4]:=' RUTA MAS CORTA DEL NODO s AL t   ';
 letmenu[5]:=' RUTA MAS CORTA DEL NODO s A TODOS';
 letmenu[6]:=' FLUJO MAXIMO                     ';
 letmenu[7]:=' FLUJO A COSTO MINIMO             ';
 LETMENU[8]:=' INTRODUCIR RED                   ';
 letmenu[9]:=' LEER RED                         ';
 LETMENU[10]:=' SALVAR RED                      ';
 LETMENU[11]:=' FINALIZAR SESION                ';
```

```
 OPCION:=menu(22,8,35,11);
 CASE OPCION OF
      1: MINIMOARBOL(MATGEN,N);
      2: MAXIARBOL(MATGEN);
      3: RUTAMASCORTA(MATGEN);
      4: RUTACORTATODOS(MATGEN);
      5: FLUJOMAX(MATGEN);
      6: FLUJOACOSTOMINIMO(MATGEN);
      7: INTRODUCIR(MATGEN);
      8: LEE(MATGEN,N,NOM);
      9: GUARDA(MATGEN);
     10: ADIOS;
 END;
 UNTIL opcion=10;
 WRITELN('                          ADIOS');
END.
```

REFERENCIAS

(1) ALFRED V. AHO & JOHN E. HOPCROFT & JEFFREY D. ULLMAN
THE DESIGN AND ANALYSIS OF COMPUTER ALGORITHMS
EDITORIAL : ADDISON WESLEY
1974

(2) MACIEJ M. SYSLO & NARSINGH DEO & JANUSZ S. KOWALIK.
DISCRETE OPTIMIZATION ALGORITHMS
EDITORIAL : PRENTICE - HALL, INC.
1983

(3) FORD & FULKERSON
FLOWS IN NETWORKS
EDITORIAL : PRINCENTON UNIVERSITY
1962

(4) T B BOFFEY
GRAPH THEORY IN OPERATIONS RESEARCH
EDITORIAL : THE MACMILLAN PRESS LTD
1984

(5) EDWARD MINIEKA
OPTIMIZATION ALGORITHMS FOR NETWORKS AND GRAPHS
EDITORIAL : MARCEL DEKKER, INC.
1978

(6) POTTS, R.B. & R.M. OLIVER
FLOWS IN TRANSPORTATION NETWORKS
EDITORIAL : NEW YORK ACADEMIC
1972

(7) TAHA
INVESTIGACION DE OPERACIONES
EDITORIAL : ALFAOMEGA
1991

(8)MA. DEL CARMEN HERNANDEZ AYUSO
VINCULOS MATEMATICOS "ANALISIS DE REDES"
FACULTAD DE CIENCIAS, UNAM.
#161, 1992.

(9) STEVE WOOD
TURBO PASCAL
EDITORIAL : MC. GRAW HILL
1988

(10)BY O. BERMAN & D. EINAV & G. HANDLER
OPERATION RESEARCH
"THE CONSTRAINED BOTTLENECK PROBLEM IN NETWORKS"
VOLUME 38, NUMBER 1
JUANARY - FEBRUARY 1990.

(11) EDWARD ALLBURN
GRAPH DECOMPOSITION
DR. DOBB'S JOURNAL
JANUARY 1991

(12) **NOTAS DE CLASE "TEORIA DE REDES"**