Apuntes

Tema ***Grafos – Algoritmo de Dijkstra***

"¿Qué somos las personas sino máquinas muy evolucionadas?"
Marvin Minsky

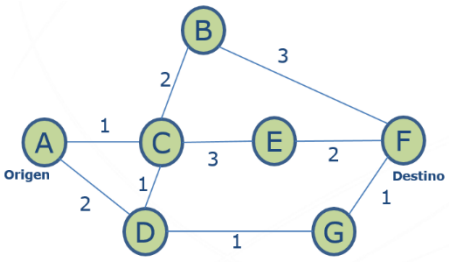
Grafos

Algoritmo de Dijkstra


Edsger Dijkstra

El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos (shortest path first "SPF"), es un algoritmo voraz (greedy) que sirve para la determinación del camino más corto dado un vértice origen al resto de los vértices en un grafo con pesos en cada arista. Su nombre se refiere al holandés Edsger Dijkstra, quien lo describió por primera vez en 1959.

La idea en este algoritmo consiste en ir explorando todos los caminos dándose cuenta de los más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen, al resto de vértices que componen el grafo, el algoritmo se detiene.



El algoritmo es una especialización de la búsqueda de costo uniforme, y como tal, no funciona en grafos con aristas de coste negativo (al elegir siempre el nodo con distancia menor, pueden quedar excluidos de la búsqueda nodos que en próximas iteraciones bajarían el costo general del camino al pasar por una arista con costo negativo).

Complejidad

Orden de complejidad del algoritmo: $\boldsymbol{O}(|\boldsymbol{V}|^{\mathbf{2}} + |\boldsymbol{A}|) = \boldsymbol{O}(|\boldsymbol{V}|^{\mathbf{2}})$ sin utilizar cola de prioridad, $\boldsymbol{O}((|\boldsymbol{A}| + |\boldsymbol{V}|)\boldsymbol{log}|\boldsymbol{V}|) = \boldsymbol{O}(|\boldsymbol{A}|\boldsymbol{log}|\boldsymbol{V}|)$utilizando cola de prioridad *(por ejemplo un montículo)*. Por otro lado, si se utiliza un Montículo de Fibonacci, sería $\boldsymbol{O}(|\boldsymbol{V}|\boldsymbol{log}|\boldsymbol{V}| + |\boldsymbol{A}|)$.
La complejidad computacional del algoritmo de Dijkstra se puede calcular contando las operaciones realizadas:

- El algoritmo consiste en $\boldsymbol{n} - \mathbf{1}$ iteraciones como máximo. En cada iteración se añade un vértice al conjunto distinguido.

- En cada iteración se identifica el vértice con la menor etiqueta entre los que no están en S_k. El número de estas operaciones está acotado por $n - 1$.
- Además se realizan una suma y una comparación para actualizar la etiqueta de cada uno de los vértices que no están en S_k.

Luego, en cada iteración se realizan a lo sumo $2(n-1)$ operaciones. Entonces tenemos:
TEOREMA: El Algoritmo de Dijkstra realiza $O(n^2)$ operaciones *(sumas y comparaciones)* para determinar la longitud del camino más corto entre dos vértices de un grafo ponderado simple, conexo y no dirigido con n vértices.

Aplicaciones

En múltiples aplicaciones donde se aplican los grafos, es necesario conocer el camino de menor costo entre dos vértices dados:

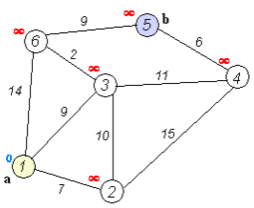
- Distribución de productos a una red de establecimientos comerciales.
- Distribución de correos postales.

Sea G = (V, A) un grafo dirigido ponderado.
El problema del camino más corto de un vértice a otro consiste en determinar el camino de menor costo, desde un vértice u a otro vértice v. El costo de un camino es la suma de los costos (pesos) de los arcos que lo conforman.

Características

- Es un algoritmo voraz (greddy).
- Trabaja por etapas, y toma en cada etapa la mejor solución sin considerar consecuencias futuras.
- El óptimo encontrado en una etapa puede modificarse posteriormente si surge una solución mejor.

Algoritmos



Teniendo un grafo dirigido ponderado de **N** nodos no aislados, sea **x** el nodo inicial, un vector **D** de tamaño **N** guardará al final del algoritmo las distancias desde **x** al resto de los nodos.

1- Inicializar todas las distancias en **D** con un valor infinito relativo ya que son desconocidas al principio, exceptuando la de **x** que se debe colocar en 0 debido a que la distancia de **x** a **x** sería 0.
2- Sea **a = x** (tomamos a como nodo actual).
3- Recorremos todos los nodos adyacentes de **a**, excepto los nodos marcados, llamaremos a estos nodos no marcados **vi**.
4- Para el nodo actual, calculamos la distancia tentativa desde dicho nodo a sus vecinos con la siguiente fórmula: **dt(vi) = Da + d(a,vi)**. Es decir, la distancia tentativa del nodo **"vi"** es la distancia que actualmente tiene el nodo en el vector **D** más la distancia desde dicho el nodo **'a'**

(el actual) al nodo **vi**. Si la distancia tentativa es menor que la distancia almacenada en el vector, actualizamos el vector con esta distancia tentativa. Es decir: Si **dt(vi) < Dvi → Dvi = dt(vi)**
5-Marcamos como completo el nodo **a**.
6- Tomamos como próximo nodo actual el de menor valor en **D** *(puede hacerse almacenando los valores en una cola de prioridad)* y volvemos al paso 3 mientras existan nodos no marcados.
Una vez terminado al algoritmo, **D** estará completamente lleno.

Ejemplo de Pseudo código

```
DIJKSTRA (Grafo G, nodo_fuente s)
 para u ∈ V[G] hacer
   distancia[u] = INFINITO
   padre[u] = NULL
   visto[u] = false
 distancia[s] = 0
 adicionar (cola, (s, distancia[s]))
 mientras que cola no es vacía hacer
   u = extraer_mínimo(cola)
   visto[u] = true
   para todos v ∈ adyacencia[u] hacer
      si no visto[v] y distancia[v] > distancia[u] + peso (u, v) hacer
         distancia[v] = distancia[u] + peso (u, v)
         padre[v] = u
         adicionar(cola,(v, distancia[v]))

función Dijkstra (Grafo G, nodo_salida s)
 //Usaremos un vector para guardar las distancias del nodo salida al resto
 entero distancia[n]
 //Inicializamos el vector con distancias iniciales booleano
 visto[n]
 //vector de boleanos para controlar los vértices de los que ya tenemos la
distancia mínima
 para cada w ∈ V[G] hacer
   Si (no existe arista entre s y w) entonces
      distancia[w] = Infinito //puedes marcar la casilla con un -1 por ejemplo
   Si_no distancia[w] = peso (s, w)
   fin si
 fin para
 distancia[s] = 0
 visto[s] = cierto
 //n es el número de vértices que tiene el Grafo
 mientras que (no_estén_vistos_todos) hacer
   vértice = tomar_el_mínimo_del_vector distancia y que no esté visto;
   visto[vértice] = cierto;
   para cada w ∈ sucesores (G, vértice) hacer
      si distancia[w]>distancia[vértice]+peso (vértice, w) entonces
         distancia[w] = distancia[vértice]+peso (vértice, w)
      fin si
   fin para
 fin mientras
 fin función.
```

Ejemplo de código en C

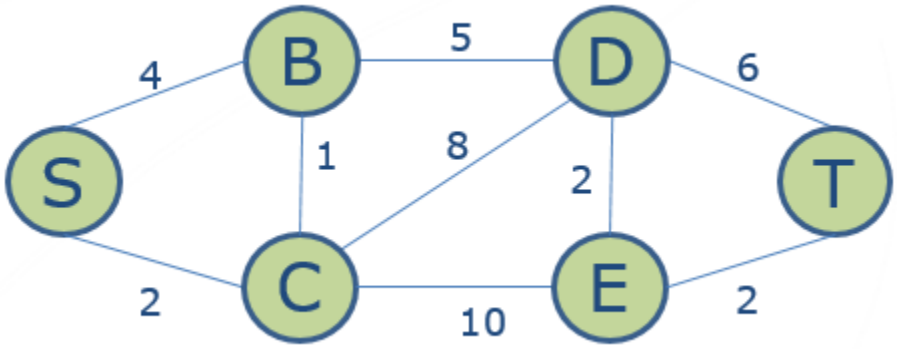
```c
#define MAX_NODES 1024 /* número máximo de nodos */
#define INFINITY 1000000000 /* un número mayor que cualquier ruta máxima */
int n, dist[MAX_NODES][MAX_NODES]; /* dist[i][j] es la distancia de i a j */

void shortest_path(int s, int t, int path[])
{
    struct state { /* la ruta en la que se está trabajando */
        int predecessor; /* nodo previo */
        int length; /* longitud del origen a este nodo */
        enum {permanent, tentative} label; /* estado de la etiqueta */
    } state[MAX_NODES];
    struct state *p;
    int i, k, min;
    for (p = &state[0]; p < &state[n]; p++) { /* estado de inicialización*/
        p->predecessor = -1;
        p->length = INFINITY;
        p->label = tentative;
    }
    state[t].length = 0; state[t].label = permanent;
    k = t; /* k es el nodo de trabajo inicial */
    do{ /* ¿hay una ruta mejor desde k? */
        for (i = 0; i < n; i++) /* este grafo tiene n nodos */
            if (dist[k][i] != 0 && state[i].label == tentative) {
                if (state[k].length + dist[k][i] < state[i].length) {
                    state[i].predecessor = k;
                    state[i].length = state[k].length + dist[k][i];
                }
            }
```

```
        /* Encuentra el nodo etiquetado tentativamente con la etiqueta menor. */
        k = 0; min = INFINITY;
        for (i = 0; i < n; i++)
        if (state[i].label == tentative && state[i].length < min) {
            min = state[i].length;
            k = i;
        }
        state[k].label = permanent;
    } while (k != s);
    /* Copia la ruta en el arreglo de salida. */
    i = 0; k = s;
    do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Ejemplo de resolución

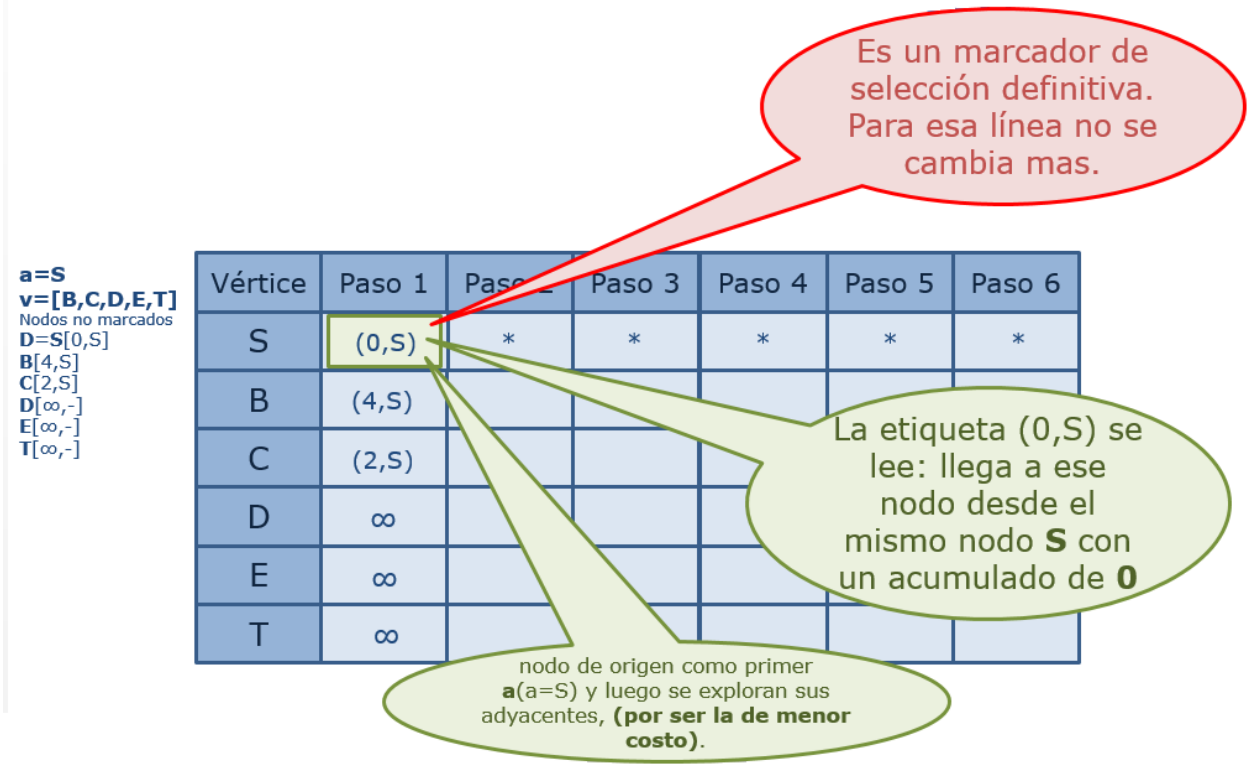
Calcular el camino de menor longitud entre s y t



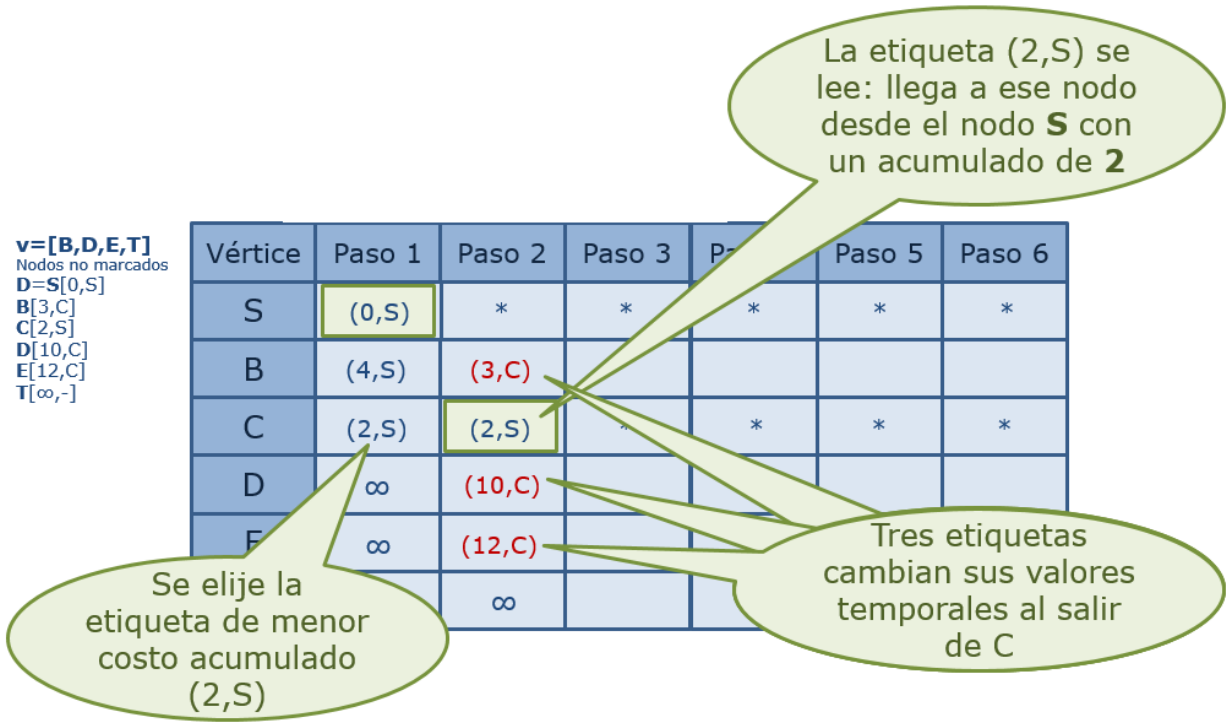
Matriz a completar:

Vértice	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5	Paso 6
S						
B						
C						
D						
E						
T						

Paso 1


Es un marcador de selección definitiva. Para esa línea no se cambia mas.
a=S
v=[B,C,D,E,T]
Nodos no marcados
D=S[0,S]
B[4,S]
C[2,S]
D[∞,-]
E[∞,-]
T[∞,-]
Vértice
Paso 1
Paso 2
Paso 3
Paso 4
Paso 5
Paso 6
S
(0,S)
B
(4,S)
C
(2,S)
D
∞
E
∞
T
∞
La etiqueta (0,S) se lee: llega a ese nodo desde el mismo nodo S con un acumulado de 0
nodo de origen como primer a(a=S) y luego se exploran sus adyacentes, (por ser la de menor costo).

Paso 2


La etiqueta (2,S) se lee: llega a ese nodo desde el nodo S con un acumulado de 2
v=[B,D,E,T]
Nodos no marcados
D=S[0,S]
B[3,C]
C[2,S]
D[10,C]
E[12,C]
T[∞,-]
Vértice
Paso 1
Paso 2
Paso 3
Paso 5
Paso 6
S
(0,S)
B
(4,S)
(3,C)
C
(2,S)
(2,S)
D
∞
(10,C)
∞
(12,C)
∞
Se elije la etiqueta de menor costo acumulado (2,S)
Tres etiquetas cambian sus valores temporales al salir de C

Paso 3

v=[D,E,T]
Nodos no marcados
D=**S**[0,S]
B[3,C]
C[2,S]
D[8,B]
E[12,C]
T[∞,-]

Vértice	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5	Paso 6
S	(0,S)	*		*	*	*
B	(4,S)	(3,C)	(3,C)	*	*	*
C	(2,S)	(2,S)	*	*	*	*
D	∞	(10,C)	(8,B)			
E	∞	(12,C)	(12,C)			
T	∞	∞	∞			

Se elije la etiqueta de menor costo acumulado (3,C) para ser definitiva

La etiqueta en D, cambia de (10,C) por (8,B) por ser de menor costo acumulado

Paso 4

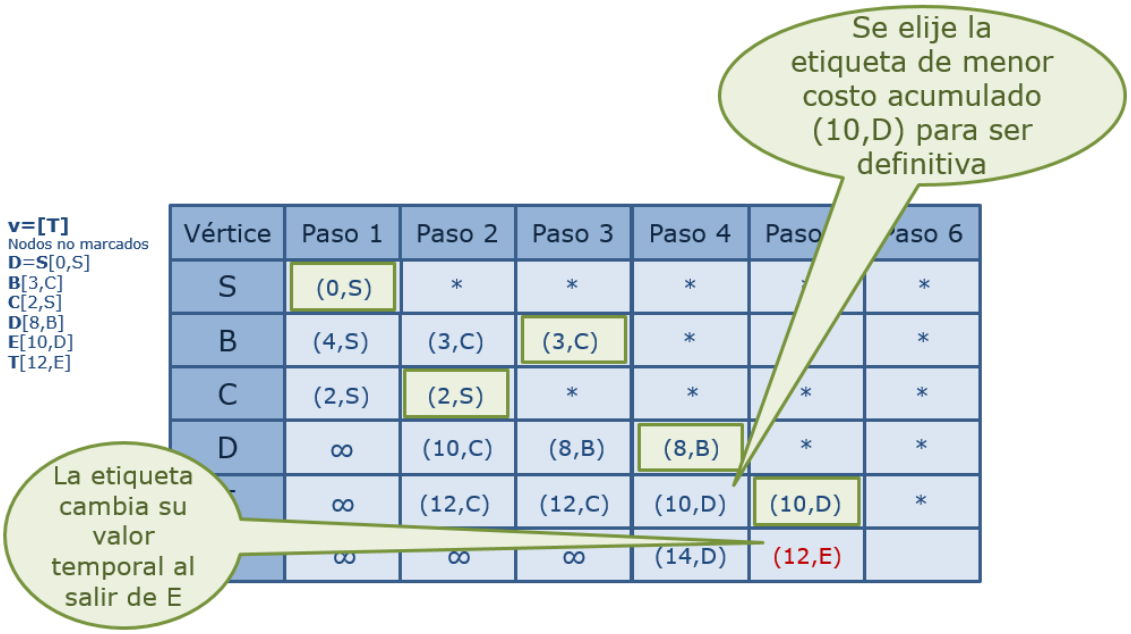
v=[E,T]
Nodos no marcados
D=**S**[0,S]
B[3,C]
C[2,S]
D[8,B]
E[10,D]
T[14,D]

Vértice	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5	Paso 6
S	(0,S)	*	*	*	*	*
B	(4,S)	(3,C)	(3,C)		*	*
C	(2,S)	(2,S)	*	*	*	*
D	∞	(10,C)	(8,B)	(8,B)	*	
E	∞	(12,C)	(12,C)	(10,D)		
T	∞	∞	∞	(14,D)		

Se elije la etiqueta de menor costo acumulado (8,B) para ser definitiva

2 etiquetas cambian sus valores temporales al salir de D

Paso 5



v=[T]
Nodos no marcados
D=**S**[0,S]
B[3,C]
C[2,S]
D[8,B]
E[10,D]
T[12,E]

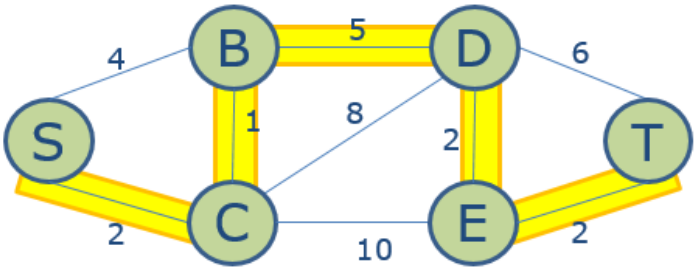
Vértice	Paso 1	Paso 2	Paso 3	Paso 4	Paso	aso 6
S	(0,S)	*	*	*		*
B	(4,S)	(3,C)	(3,C)	*		*
C	(2,S)	(2,S)	*	*	*	*
D	∞	(10,C)	(8,B)	(8,B)	*	*
	∞	(12,C)	(12,C)	(10,D)	(10,D)	*
	∞	∞	∞	(14,D)	(12,E)	

Paso 6

Tabla lista para ser leída (se lee desde T)

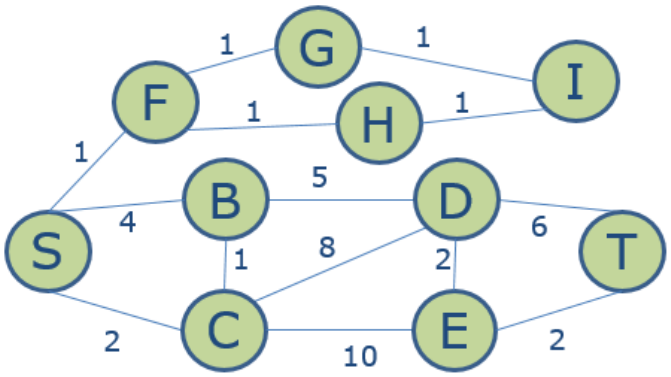
v=[]
Nodos no marcados
D=**S**[0,S]
B[3,C]
C[2,S]
D[8,B]
E[10,D]
T[12,E]

Vértice	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5	Paso 6	D-Final
S	(0,S)	*	*	*	*	*	(0,S)
B	(4,S)	(3,C)	(3,C)	*	*	*	(3,C)
C	(2,S)	(2,S)	*	*	*	*	(2,S)
D	∞	(10,C)	(8,B)	(8,B)	*	*	(8,B)
E	∞	(12,C)	(12,C)	(10,D)	(10,D)	*	(10,D)
T	∞	∞	∞	(14,D)	(12,E)	(12,E)	(12,E)



Ejemplo 2 de resolución

Calcular el camino de menor longitud entre s y t



	S	B	C	D	E	T	F	G	H	I
S		4	2				1			
B	4		1	5						
C	2	1		8	10					
D		5	8		2	6				
E			10	2		2				
T				6	2					
F	1							1	1	
G							1			1
H							1			1
I								1	1	

a=S
v=[B,C,D,E,T,F,G,H,I]
Nodos no marcados
MARK=[S]
Nodos Marcados
D=S[0,S]
B[4,S]
C[2,S]
D[∞,-]
E[∞,-]
T[∞,-]
F[1,S]
G[∞,-]
H[∞,-]
I[∞,-]

Vértice	Paso 1
S	(0,S)
B	(4,S)
C	(2,S)
D	∞
E	∞
T	∞
F	(1,S)
G	∞
H	∞
I	∞

Es un marcador de selección, para no modificar.

nodo de origen como primer **a**(a=S) y luego se exploran sus adyacentes, **empezando por el mas cercano**.

a=S
v=[B,C,D,E,T,F,G,H,I]
Nodos no marcados
MARK=[S]
Nodos Marcados
D=S[0,S]
B[4,S]
C[2,S]
D[∞,-]
E[∞,-]
T[∞,-]
F[1,S]
G[2,F]
H[2,F]
I[∞,-]

Vértice	Paso 1	Paso 2
S	(0,S)	*
B	(4,S)	(4,S)
C	(2,S)	(2,S)
D	∞	∞
E	∞	∞
T	∞	∞
F	(1,S)	(1,S)
G	∞	(2,F)
H	∞	(2,F)
I	∞	∞

Ahora se toma **a=F**
(adyacente mas cercano a S)
y se exploran sus adyacentes.

a=S
v=[B,C,D,E,T,F,G,H,I]
Nodos no marcados
MARK=[S]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[10,C]
E[12,C]
T[∞,-]
F[1,S]
G[2,F]
H[2,F]
I[∞,-]

Vértice	Paso 1	Paso 2	Paso 3
S	(0,S)	*	*
B	(4,S)	(4,S)	(3,C)
C	(2,S)	(2,S)	(2,S)
D	∞	∞	(10,C)
E	∞	∞	(12,C)
T	∞	∞	∞
F	(1,S)	(1,S)	*
G	∞	(2,F)	(2,F)
H	∞	(2,F)	(2,F)
I	∞	∞	∞

Ahora se toma **a=C**
(adyacente mas cercano a S)
y se exploran sus adyacentes.

a=S
v=[B,C,D,E,T,F,G,H,I]
Nodos no marcados
MARK=[S]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[10,C]
E[12,C]
T[∞,-]
F[1,S]
G[2,F]
H[2,F]
I[3,G]

Vértice	Paso 1	Paso 2	Paso 3	Paso 4
S	(0,S)	*	*	*
B	(4,S)	(4,S)	(3,C)	(3,C)
C	(2,S)	(2,S)	(2,S)	*
D	∞	∞	(10,C)	(10,C)
E	∞	∞	(12,C)	(12,C)
T	∞	∞	∞	∞
F	(1,S)	(1,S)	*	*
G	∞	(2,F)	(2,F)	(2,F)
H	∞	(2,F)	(2,F)	(2,F)
I	∞	∞	∞	(3,G)

Ahora se toma **a=G**
(adyacente mas cercano a S)
y se exploran sus adyacentes.

a=S
v=[B,C,D,E,T,F,G,H,I]
Nodos no marcados
MARK=[S]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[10,C]
E[12,C]
T[∞,-]
F[1,S]
G[2,F]
H[2,F]
I[3,G]

Vértice	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5
S	(0,S)	*	*	*	*
B	(4,S)	(4,S)	(3,C)	(3,C)	(3,C)
C	(2,S)	(2,S)	(2,S)	*	*
D	∞	∞	(10,C)	(10,C)	(10,C)
E	∞	∞	(12,C)	(12,C)	(12,C)
T	∞	∞	∞	∞	∞
F	(1,S)	(1,S)	*	*	*
G	∞	(2,F)	(2,F)	(2,F)	*
H	∞	(2,F)	(2,F)	(2,F)	(2,F)
I	∞	∞	∞	(3,G)	(3,G)

Al llegar a I con 3 desde H como con 3 desde G, no se reemplaza

Ahora se toma **a=G** (adyacente mas cercano a S) y se exploran sus adyacentes.

a=S
v=[B,C,D,E,T,F,G,H,I]
Nodos no marcados
MARK=[S]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[8,B]
E[12,C]
T[∞,-]
F[1,S]
G[2,F]
H[2,F]
I[3,G]

Vértice	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5	Paso 6
S	(0,S)	*	*	*	*	*
B	(4,S)	(4,S)	(3,C)	(3,C)	(3,C)	(3,C)
C	(2,S)	(2,S)	(2,S)	*	*	*
D	∞	∞	(10,C)	(10,C)	(10,C)	(8,B)
E	∞	∞	(12,C)	(12,C)	(12,C)	(12,C)
T	∞	∞	∞	∞	∞	∞
F	(1,S)	(1,S)	*	*	*	*
G	∞	(2,F)	(2,F)	(2,F)	*	*
H	∞	(2,F)	(2,F)	(2,F)	(2,F)	*
I	∞	∞	∞	(3,G)	(3,G)	(3,G)

Ahora se toma **a=B** (adyacente mas cercano a S) y se exploran sus adyacentes.

a=S
v=[B,C,D,E,T,F,G,H,I]
Nodos no marcados
MARK=[S]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[8,B]
E[12,C]
T[∞,-]
F[1,S]
G[2,F]
H[2,F]
I[3,G]

Vértice	Paso 6	Paso 7
S	*	*
B	(3,C)	*
C	*	*
D	(8,B)	(8,B)
E	(12,C)	(12,C)
T	∞	∞
F	*	*
G	*	*
H	*	*
I	(3,G)	(3,G)

Ahora se toma **a=I** (adyacente mas cercano a S) y se exploran sus adyacentes.

a=S
v=[B,C,D,E,T,F,G,H,I]
Nodos no marcados
MARK=[S]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[8,B]
E[10,D]
T[14,D]
F[1,S]
G[2,F]
H[2,F]
I[3,G]

Vértice	Paso 6	Paso 7	Paso 8
S	*	*	*
B	(3,C)	*	*
C	*	*	*
D	(8,B)	(8,B)	(8,B)
E	(12,C)	(12,C)	(10,D)
T	∞	∞	(14,D)
F	*	*	*
G	*	*	*
H	*	*	*
I	(3,G)	(3,G)	*

Ahora se toma **a=D**
(adyacente mas cercano a S)
y se exploran sus adyacentes.

a=S
v=[B,C,D,E,T,F,G,H,I]
Nodos no marcados
MARK=[S]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[8,B]
E[10,D]
T[12,E]
F[1,S]
G[2,F]
H[2,F]
I[3,G]

Vértice	Paso 6	Paso 7	Paso 8	Paso 9
S	*	*	*	*
B	(3,C)	*	*	*
C	*	*	*	*
D	(8,B)	(8,B)	(8,B)	*
E	(12,C)	(12,C)	(10,D)	(10,D)
T	∞	∞	(14,D)	(12,E)
F	*	*	*	*
G	*	*	*	*
H	*	*	*	*
I	(3,G)	(3,G)	*	*

Ahora se toma **a=E**
(adyacente mas cercano a S)
y se exploran sus adyacentes.

a=S
v=[B,C,D,E,T,F,G,H,I]
Nodos no marcados
MARK=[S]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[8,B]
E[10,D]
T[12,E]
F[1,S]
G[2,F]
H[2,F]
I[3,G]

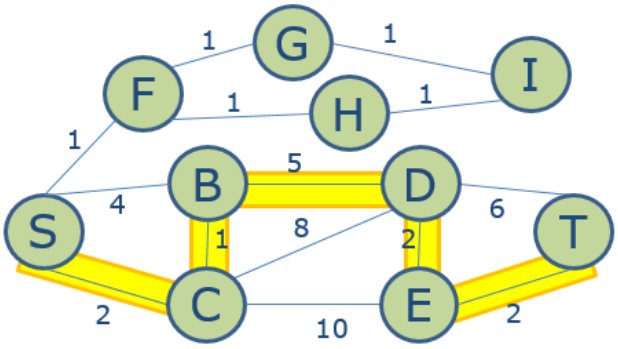
Vértice	Paso 6	Paso 7	Paso 8	Paso 9	Paso 10
S	*	*	*	*	*
B	(3,C)	*	*	*	*
C	*	*	*	*	*
D	(8,B)	(8,B)	(8,B)	*	*
E	(12,C)	(12,C)	(10,D)	(10,D)	*
T	∞	∞	(14,D)	(12,E)	(12,E)
F	*	*	*	*	*
G	*	*	*	*	*
H	*	*	*	*	*
I	(3,G)	(3,G)	*	*	*

Ahora se toma **a=T**
(adyacente mas cercano a S)
y se exploran sus adyacentes.

	Final
S	(0,S)
B	(3,C)
C	(2,S)
D	(8,B)
E	(10,D)
T	(12,E)
F	(1,S)
G	(2,F)
H	(2,F)
I	(3,G)

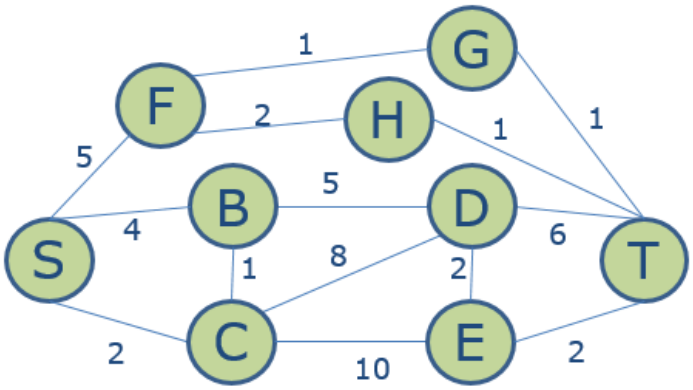
S C B D E T

Tabla lista para ser leída



Ejemplo de resolución 3

Calcular el camino de menor longitud entre s y t



	S	B	C	D	E	T	F	G	H
S		4	2				5		
B	4		1	5					
C	2	1		8	10				
D		5	8		2	6			
E			10	2		2			
T				6	2			1	1
F	5							1	2
G						1	1		
H						1	2		

a=S
v=[B,C,D,E,T,F,G,H]
Nodos no marcados
MARK=[S]
Nodos Marcados
D=S[0,S]
B[4,S]
C[2,S]
D[∞,-]
E[∞,-]
T[∞,-]
F[5,S]
G[∞,-]
H[∞,-]

Vértice	Paso 1
S	(0,S)
B	(4,S)
C	(2,S)
D	∞
E	∞
T	∞
F	(5,S)
G	∞
H	∞

Es un marcador de selección, para no modificar.

La etiqueta (2,S) se lee: llega a ese nodo desde el nodo **S** con un "acumulado" de **2**

Se toma el nodo de origen como primer **a**(a=S) y luego se exploran sus adyacentes, **empezando por el mas cercano**.

a=C
v=[B,D,E,T,F,G,H]
Nodos no marcados
MARK=[S,C]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[10,C]
E[12,C]
T[∞,-]
F[5,S]
G[∞,-]
H[∞,-]

Vértice	Paso 1	Paso 2
S	(0,S)	*
B	(4,S)	(3,C)
C	(2,S)	(2,S)
D	∞	(10,C)
E	∞	(12,C)
T	∞	∞
F	(5,S)	(5,S)
G	∞	∞
H	∞	∞

Ahora se toma **a=C** (adyacente mas cercano a S) y se exploran sus adyacentes.

a=B
v=[D,E,T,F,G,H]
Nodos no marcados
MARK=[S,C,B]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[9,B]
E[12,C]
T[∞,-]
F[5,S]
G[∞,-]
H[∞,-]

Vértice	Paso 1	Paso 2	Paso 3
S	(0,S)	*	*
B	(4,S)	(3,C)	(3,C)
C	(2,S)	(2,S)	*
D	∞	(10,C)	(9,B)
E	∞	(12,C)	(12,C)
T	∞	∞	∞
F	(5,S)	(5,S)	(5,S)
G	∞	∞	∞
H	∞	∞	∞

Se reemplaza (10,C) por (9,B), porque su costo acumulado es menor

Ahora se toma **a=B** y se exploran sus adyacentes.

a=F
v=[D,E,T,G,H]
Nodos no marcados
MARK=[S,C,B, F]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[9,B]
E[12,C]
T[∞,-]
F[5,S]
G[6,F]
H[7,F]

Vértice	Paso 1	Paso 2	Paso 3	Paso 4
S	(0,S)	*	*	*
B	(4,S)	(3,C)	(3,C)	*
C	(2,S)	(2,S)	*	*
D	∞	(10,C)	(9,B)	(9,B)
E	∞	(12,C)	(12,C)	(12,C)
T	∞	∞	∞	∞
F	(5,S)	(5,S)	(5,S)	(5,S)
G	∞	∞	∞	(6,F)
H	∞	∞	∞	(7,F)

Se reemplaza el ∞ en G por (6,G) y el ∞ en H por (7,G)

Ahora se toma **a=F** y se exploran sus adyacentes.

a=G
v=[D,E,T,H]
Nodos no marcados
MARK=[S,C,B, F,G]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[9,B]
E[12,C]
T[7,G]
F[5,S]
G[6,F]
H[7,F]

Vértice	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5
S	(0,S)	*	*	*	*
B	(4,S)	(3,C)	(3,C)	*	*
C	(2,S)	(2,S)	*	*	*
D	∞	(10,C)	(9,B)	(9,B)	(9,B)
E	∞	(12,C)	(12,C)	(12,C)	(12,C)
T	∞	∞	∞	∞	(7,G)
F	(5,S)	(5,S)	(5,S)	(5,S)	*
G	∞	∞	∞	(6,F)	(6,F)
H	∞	∞	∞	(7,F)	(7,F)

Se reemplaza el ∞ en T por (7,T)

Ahora se toma **a=G** y se exploran sus adyacentes.

a=T
v=[D,E,H]
Nodos no marcados
MARK=[S,C,B, F,G,T]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[9,B]
E[12,C]
T[7,G]
F[5,S]
G[6,F]
H[7,F]

Vértice	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5	Paso 6
S	(0,S)	*	*	*	*	*
B	(4,S)	(3,C)	(3,C)	*	*	*
C	(2,S)	(2,S)	*	*	*	*
D	∞	(10,C)	(9,B)	(9,B)	(9,B)	(9,B)
E	∞	(12,C)	(12,C)	(12,C)	(12,C)	(12,C)
T	∞	∞	∞	∞	(7,G)	(7,G)
F	(5,S)	(5,S)	(5,S)	(5,S)	*	*
G	∞	∞	∞	(6,F)	(6,F)	*
H	∞	∞	∞	(7,F)	(7,F)	(7,F)

Ahora se toma **a=T** y se exploran sus adyacentes.

Al haber 2 etiquetas con el mismo costo acumulado se toma simplemente la primera

a=H
v=[D,E]
Nodos no marcados
MARK=[S,C,B, F,G,T,H]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[9,B]
E[12,C]
T[7,G]
F[5,S]
G[6,F]
H[7,F]

Vértice	Paso 6	Paso 7
S	*	*
B	*	*
C	*	*
D	(9,B)	(9,B)
E	(12,C)	(12,C)
T	(7,G)	*
F	*	*
G	*	*
H	(7,F)	(7,F)

Ahora se toma **a=H** y se exploran sus adyacentes.

a=D
v=[E]
Nodos no marcados
MARK=[S,C,B, F,G,T,H,D]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[9,B]
E[11,D]
T[7,G]
F[5,S]
G[6,F]
H[7,F]

Vértice	Paso 6	Paso 7	Paso 8
S	*	*	*
B	*	*	*
C	*	*	*
D	(9,B)	(9,B)	(9,B)
E	(12,C)	(12,C)	(11,D)
T	(7,G)	*	*
F	*	*	*
G	*	*	*
H	(7,F)	(7,F)	*

Se reemplaza el (12,C) en E por (11,D)

Ahora se toma **a=D** y se exploran sus adyacentes.

a=E
v=[]
Nodos no marcados
MARK=[S,C,B, F,G,T,H,D,E]
Nodos Marcados
D=S[0,S]
B[3,C]
C[2,S]
D[9,B]
E[11,D]
T[7,G]
F[5,S]
G[6,F]
H[7,F]

Vértice	Paso 6	Paso 7	Paso 8	Paso 9
S	*	*	*	*
B	*	*	*	*
C	*	*	*	*
D	(9,B)	(9,B)	(9,B)	*
E	(12,C)	(12,C)	(11,D)	(11,D)
T	(7,G)	*	*	*
F	*	*	*	*
G	*	*	*	*
H	(7,F)	(7,F)	*	*

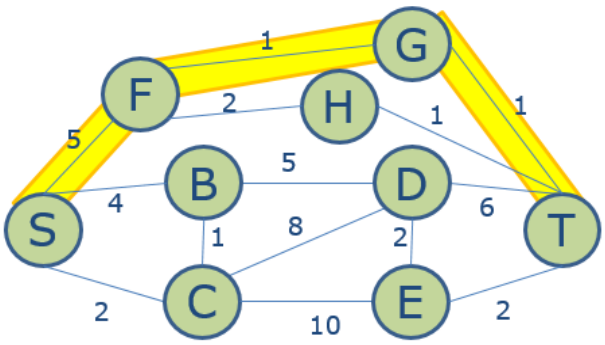
Ahora se toma **a=D** y se exploran sus adyacentes.

Última etiqueta

	Final
S	(0,S)
B	(3,C)
C	(2,S)
D	(9,B)
E	(11,D)
T	(7,G)
F	(5,S)
G	(6,F)
H	(7,F)

S F G T

Tabla lista para ser leída


G
F
H
B
D
S
T
C
E
1
2
1
1
5
5
4
6
1
8
2
2
10
2