Problema del camino mínimo

Definición

Un **grafo ponderado** es un par (G, w) donde $G = (V, E)$ es un grafo y w es una función $w : E \to \mathbb{R}$ que asigna pesos a las aristas del grafo.

Definición

Un **grafo ponderado** es un par (G,w) donde $G=(V,E)$ es un grafo y w es una función $w: E \to \mathbb{R}$ que asigna pesos a las aristas del grafo.

Definición

Dado un grafo ponderado (G,w) y un camino $C: v_0, v_1, \ldots, v_k$ se define el **peso del camino** C como

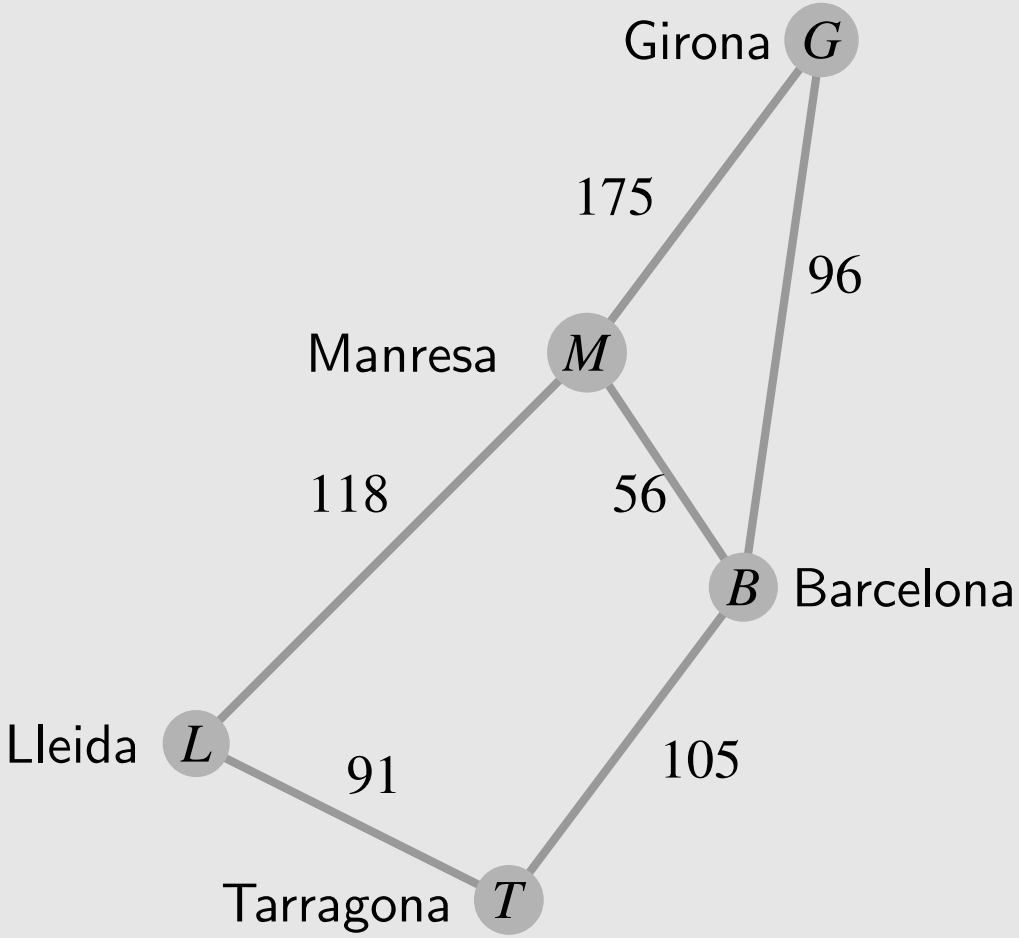
$$w(C) = \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

y la **distancia** entre dos vértices $u, v \in G$ como

$$d_G(u,v) = \text{mín}\{w(C) : \ C \text{ es un } u-v \text{ camino } \}.$$

Ejemplo

Determina la distancia entre cada par de ciudades.



Algoritmo de Dijkstra

Algunas consideraciones sobre el algoritmo de Dijkstra

- El algoritmo recibe como entrada un grafo G y un vértice inicial o de partida s.
- El algoritmo devuelve para cada vértice u del grafo G un par (d, v), donde d es la distancia desde s hasta u, mientras que v es el vértice anterior a u en el camino mínimo $s - u$.
- El algoritmo es codicioso (greedy), siempre elige en cada paso el vértice con distancia menor a s. Por esa razón, no funciona en grafos con aristas de coste negativo (ver algoritmo de Bellman-Ford en caso de que el peso de alguna de las aristas puede ser negativo). En la Figura 1 se puede ver el resultado erróneo que daría Dijkstra para un grafo con al menos arista con coste negativo. En la Figura 2 se muestra que tampoco sería correcto sumar un mismo número positivo al peso de cada arista de modo que todos los pesos quedaran no negativos para luego aplicar Dijkstra.

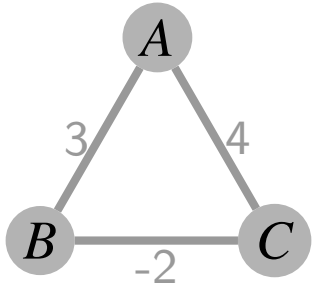
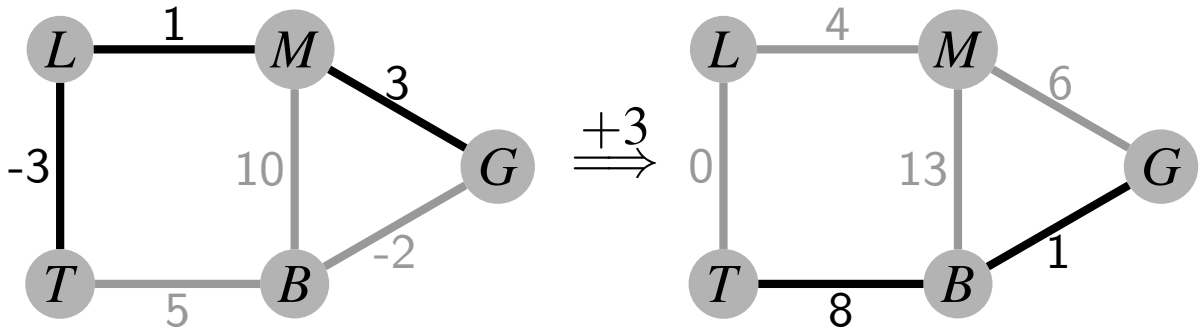


Figura: 1. Grafo ponderado con al menos una arista con coste negativo. Dijkstra, partiendo del vértice A daría $d(A,B) = 3$ (camino $A - B$) cuando lo correcto sería que $d(A,B) = 2$ (camino $A - C - B$).



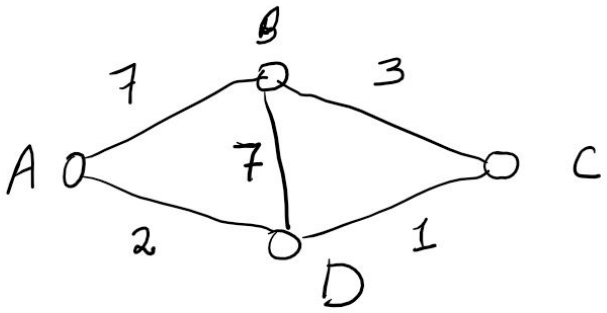
$d(T,G) = 1$ $\qquad$ $d(T,G) = 9$

Figura: 2. Grafo ponderado original a la izquierda y grafo ponderado transformado a la derecha con pesos no negativos (sumando 3 al peso de cada arista). Camino mínimo de T a G en negrita en ambos grafos ponderados.

Algoritmo Dijkstra (G, s)

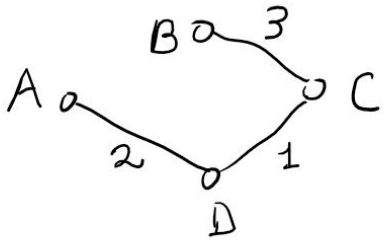
$U \leftarrow \emptyset$ (U es la lista de vértices visitados)
para $v \in V \backslash \{s\}$
    $dist(v) \leftarrow \infty$
    Se etiqueta v con $(dist(v), s)$
finpara
$dist(s) \leftarrow 0$
Se etiqueta s con $(dist(s), s)$
para $i \leftarrow 0$ hasta $\leftarrow n-1$
    u_i **vértice que alcanza** $\min_{v \in V-U} \{dist(v)\}$
    $U \leftarrow U \cup \{u_i\}$
    para $v \in V - U$ adyacente a u_i
        si $dist(u_i) + w(u_i, v) < dist(v)$
            entonces $dist(v) \leftarrow dist(u_i) + w(u_i, v)$
            Se etiqueta v con $(dist(v), u_i)$
        finsi
    finpara
finpara
retorno ($dist$)

Ejemplo



A	B	C	D
(0,A)	(∞,A)	(∞,A)	(∞,A)
(0,A)*	(7,A)	(∞,A)	(2,A)
	(7,A)	(3,D)	(2,A)*
	(6,C)	(3,D)*	
	(6,C)*		

Árbol de distancias



d(A,B)=6
d(A,C)=3
d(A,D)=2

Ejercicio

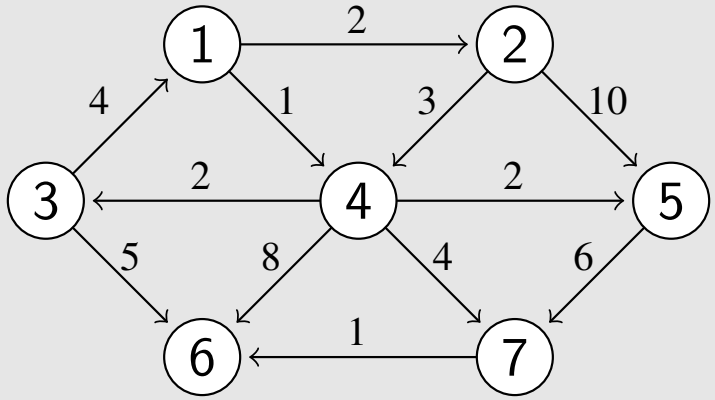
Determina la distancia de A a cada una de los vértices de la red de carreteras mostrada en la tabla.

	A	B	C	D	E	F	G
A	0	5	3	2	-	-	-
B	5	0	2	-	3	-	1
C	3	2	0	7	7	-	-
D	2	-	7	0	2	6	-
E	-	3	7	2	0	1	1
F	-	-	-	6	1	0	-
G	-	1	-	-	1	-	0

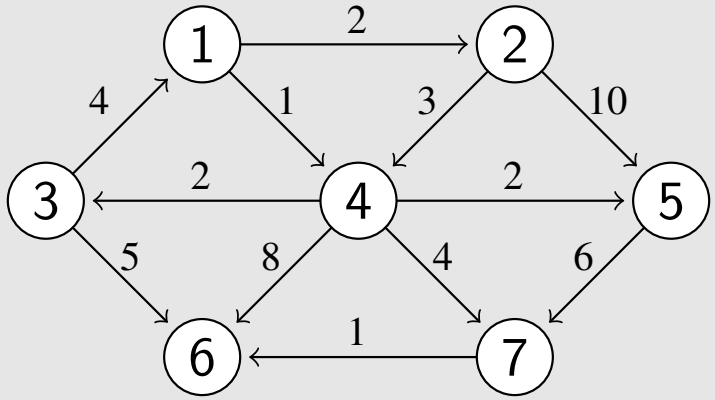
Solución

A	B	C	D	E	F	G
(0,A)	(∞,A)	(∞,A)	(∞,A)	(∞,A)	(∞,A)	(∞,A)
(0,A)*	(5,A)	(3,A)	(2,A)	(∞,A)	(∞,A)	(∞,A)
(0,A)	(5,A)	(3,A)	(2,A)*	(4,D)	(8,D)	(∞,A)
(0,A)	(5,A)	(3,A)*	(2,A)	(4,D)	(8,D)	(∞,A)
(0,A)	(5,A)	(3,A)	(2,A)	(4,D)*	(5,E)	(5,E)
(0,A)	(5,A)*	(3,A)	(2,A)	(4,D)	(5,E)	(5,E)
(0,A)	(5,A)	(3,A)	(2,A)	(4,D)	(5,E)*	(5,E)
(0,A)	(5,A)	(3,A)	(2,A)	(4,D)	(5,E)	(5,E)*

Ejemplo



Ejemplo



Solución

1	2	3	4	5	6	7
(0,1)	(∞,1)	(∞,1)	(∞,1)	(∞,1)	(∞,1)	(∞,1)
(0,1)*	(2,1)	(∞,1)	(1,1)	(∞,1)	(∞,1)	(∞,1)
(0,1)	(2,1)	(3,4)	(1,1)*	(3,4)	(9,4)	(5,4)
(0,1)	(2,1)*	(3,4)	(1,1)	(3,4)	(9,4)	(5,4)
(0,1)	(2,1)	(3,4)*	(1,1)	(3,4)	(8,3)	(5,4)
(0,1)	(2,1)	(3,4)	(1,1)	(3,4)*	(8,3)	(5,4)
(0,1)	(2,1)	(3,4)	(1,1)	(3,4)	(6,7)	(5,4)*

Ejercicio

La siguiente matriz es la matriz de adyacencia de un grafo ponderado de vértices A, B, C, D, E y F.

$$\begin{pmatrix} 0 & 8 & 0 & 0 & 5 & 0 \\ 8 & 0 & 7 & 2 & 2 & 0 \\ 0 & 7 & 0 & 8 & 0 & 3 \\ 0 & 2 & 8 & 0 & 0 & 4 \\ 5 & 2 & 0 & 0 & 0 & 9 \\ 0 & 0 & 3 & 4 & 9 & 0 \end{pmatrix}$$

Aplica el algoritmo de Dijkstra partiendo del vértice C

Solución

A	B	C	D	E	F
(∞,C)	(∞,C)	(0,C)	(∞,C)	(∞,C)	(∞,C)
(∞,C)	(7,C)	(0,C)*	(8,C)	(∞,C)	(3,C)
(∞,C)	(7,C)	(0,C)	(7,F)	(12,F)	(3,C)*
(15,B)	(7,C)*	(0,C)	(7,F)	(9,B)	(3,C)
(15,B)	(7,C)	(0,C)	(7,F)*	(9,B)	(3,C)
(14,E)	(7,C)	(0,C)	(7,F)	(9,B)*	(3,C)
(14,E)*	(7,C)	(0,C)	(7,F)	(9,B)	(3,C)

Algoritmo de Floyd

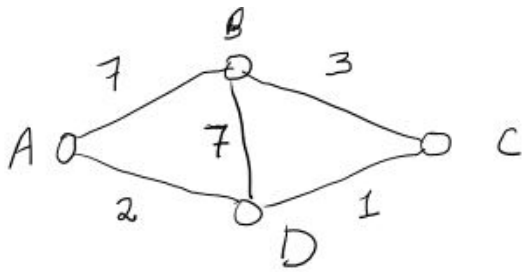
Algunas consideraciones sobre el algoritmo de Floyd

- El algoritmo también se conoce como Floyd-Warshall.
- El algoritmo, dado un grafo G, devuelve una matriz de distancia entre todos los vértices (All-Pairs-Shortest-Path).
- El algoritmo es un ejemplo de programación dinámica (método para reducir el tiempo de ejecución de un algoritmo mediante la utilización de subestructuras óptimas y subproblemas superpuestos).
- El algoritmo permite aristas de coste negativo y da una respuesta correcta siempre y cuando no haya ciclos de coste negativo (entre cualquier par de vértices que forme parte de un ciclo negativo, el camino mínimo dado por el algoritmo no está bien definido porque sería infinitamente pequeño). No obstante, si hay tal ciclo de peso negativo, el algoritmo detecta su existencia, indicando que no existe solución.

Algoritmo *Floyd*(G)

para $i \leftarrow 1$ hasta n
    para $j \leftarrow 1$ hasta n
        si $i = j$ entonces $d_{ij}^0 \leftarrow 0$ finsi
        si $(i,j) \in E$ entonces $d_{ij}^0 \leftarrow w(i,j)$ finsi
        si $(i,j) \notin E$ entonces $d_{ij}^0 \leftarrow \infty$ finsi
    finpara
finpara
para $k \leftarrow 1$ hasta n
    para $i \leftarrow 1$ hasta n
        para $j \leftarrow 1$ hasta n
            $d_{ij}^k \leftarrow \text{mín}(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
        finpara
    finpara
finpara
retorno (d_{ij}^n)

Ejemplo:



d^0=

0	7	∞	2
7	0	3	7
∞	3	0	1
2	7	1	0

d^1=

0	7	∞	2
7	0	3	7
∞	3	0	1
2	7	1	0

d^2=

0	7	10	2
7	0	3	7
10	3	0	1
2	7	1	0

d^3=

0	7	10	2
7	0	3	4
10	3	0	1
2	4	1	0

d^4=

0	6	3	2
6	0	3	4
3	3	0	1
2	4	1	0

Excentricidades:
A--> 6
B-->6
C-->3
D-->4

Radio de la red =3
Diámetro de la red= 6

Ejercicio

La siguiente matriz es la matriz de adyacencia de un grafo ponderado de vértices A, B, C, D, E y F.

$$\begin{pmatrix} 0 & 8 & 0 & 0 & 5 & 0 \\ 8 & 0 & 7 & 2 & 2 & 0 \\ 0 & 7 & 0 & 8 & 0 & 3 \\ 0 & 2 & 8 & 0 & 0 & 4 \\ 5 & 2 & 0 & 0 & 0 & 9 \\ 0 & 0 & 3 & 4 & 9 & 0 \end{pmatrix}$$

Aplica el algoritmo de Floyd y determina las excentricidades de los vértices y el diámetro de la red.

$$d^0 = \begin{pmatrix} 0 & 8 & \infty & \infty & 5 & \infty \\ 8 & 0 & 7 & 2 & 2 & \infty \\ \infty & 7 & 0 & 8 & \infty & 3 \\ \infty & 2 & 8 & 0 & \infty & 4 \\ 5 & 2 & \infty & \infty & 0 & 9 \\ \infty & \infty & 3 & 4 & 9 & 0 \end{pmatrix} \qquad d^1 = \begin{pmatrix} 0 & 8 & \infty & \infty & 5 & \infty \\ 8 & 0 & 7 & 2 & 2 & \infty \\ \infty & 7 & 0 & 8 & \infty & 3 \\ \infty & 2 & 8 & 0 & \infty & 4 \\ 5 & 2 & \infty & \infty & 0 & 9 \\ \infty & \infty & 3 & 4 & 9 & 0 \end{pmatrix}$$

$$d^2 = \begin{pmatrix} 0 & 8 & 15 & 10 & 5 & \infty \\ 8 & 0 & 7 & 2 & 2 & \infty \\ 15 & 7 & 0 & 8 & 9 & 3 \\ 10 & 2 & 8 & 0 & 4 & 4 \\ 5 & 2 & 9 & 4 & 0 & 9 \\ \infty & \infty & 3 & 4 & 9 & 0 \end{pmatrix} \qquad d^3 = \begin{pmatrix} 0 & 8 & 15 & 10 & 5 & 18 \\ 8 & 0 & 7 & 2 & 2 & 10 \\ 15 & 7 & 0 & 8 & 9 & 3 \\ 10 & 2 & 8 & 0 & 4 & 4 \\ 5 & 2 & 9 & 4 & 0 & 9 \\ 18 & 10 & 3 & 4 & 9 & 0 \end{pmatrix}$$

$$d^4 = \begin{pmatrix} 0 & 8 & 15 & 10 & 5 & 14 \\ 8 & 0 & 7 & 2 & 2 & 6 \\ 15 & 7 & 0 & 8 & 9 & 3 \\ 10 & 2 & 8 & 0 & 4 & 4 \\ 5 & 2 & 9 & 4 & 0 & 8 \\ 14 & 6 & 3 & 4 & 8 & 0 \end{pmatrix} \qquad d^5 = \begin{pmatrix} 0 & 7 & 14 & 9 & 5 & 13 \\ 7 & 0 & 7 & 2 & 2 & 6 \\ 14 & 7 & 0 & 8 & 9 & 3 \\ 9 & 2 & 8 & 0 & 4 & 4 \\ 5 & 2 & 9 & 4 & 0 & 8 \\ 13 & 6 & 3 & 4 & 8 & 0 \end{pmatrix}$$

$$d^6 = \begin{pmatrix} 0 & 7 & 14 & 9 & 5 & 13 \\ 7 & 0 & 7 & 2 & 2 & 6 \\ 14 & 7 & 0 & 7 & 9 & 3 \\ 9 & 2 & 7 & 0 & 4 & 4 \\ 5 & 2 & 9 & 4 & 0 & 8 \\ 13 & 6 & 3 & 4 & 8 & 0 \end{pmatrix}$$

Problem (homework)

Suppose that in a communication network the weights (w) of the edges reflect the quality of the information transmitted or the trust between nodes. For instance:

- The probability that a bit transmitted by node i is erroneously received by node j is a simple way to quantify the quality of transmission between node i and node j. This probability can be viewed as the weight $w(i,j)$ of edge (i,j).
- The level of trust between node i and node j in a social network can be quantified as a weight $w(i,j) \in [0,1]$ proportional to how much node i trusts node j. Extreme cases are no trust $(w(i,j) = 0)$ and maximum trust $(w(i,j) = 1)$.

In general, we are interested in the cases where the weight of a path can be computed as the product of the weights of the edges belonging to it. The weight of the path is called its *reliability*. More formally, let $G = (V,E,w)$ be a weighted digraph with vertex set V and edge set E, where $w : E \mapsto (0;1]$ is the weight function of G. For a path $P : u = v_i, v_{i+1}, \ldots, v_k = v$ in G we define the *reliability* of P as

$$w(P) = \prod_{l=1}^{k-1} w(v_l, v_{l+1}).$$

Problem (homework)

Let us assume that several routes exist from one node to another. The maximum reliability between two nodes (in terms of transmission quality, trust, etc.) is reached using the path with maximum reliability among those connecting both nodes.
For two vertices $u, v \in V$, we denote by $P_{\overrightarrow{uv}}$ the set of all directed paths from u to v. We denote by $F_{\overrightarrow{uv}}$ the weight of the most reliable path from u to v:

$$F_{\overrightarrow{uv}} = \max_{P \in P_{\overrightarrow{uv}}} \{w(P)\}. \tag{1}$$

(a) Find a method to compute the weights of the most reliable path from u to v for every pair of nodes of $u, v \in V$.

(b) Let $G = (V, E, w)$ be the weighted digraph whose adjacency matrix is given in a Moodle file. Compute the values of $F_{\overrightarrow{uv}}$ for every $u, v \in V$.

Profundiza lo suficiente en cualquier cosa y encontrarás las matemáticas.

Dean Schlicter.