Dijkstra, Floyd

Pedro O. Pérez M., PhD.

Análisis y diseño de algoritmos avanzados
Tecnológico de Monterrey

pperezm@tec.mx

12-2022

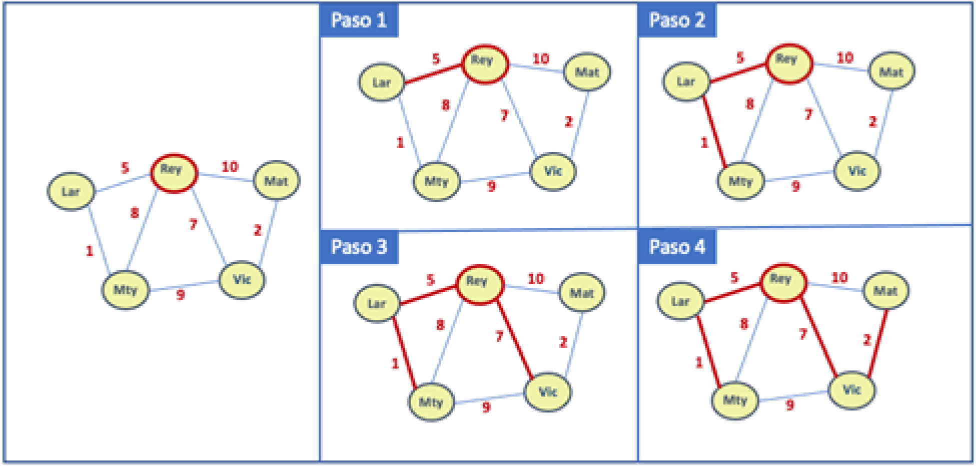
Contenido

Un grafo no direccionado $G = (V, E)$ consiste de una colección de vértices, V y una colección de arcos, E. Representamos cada arco, $e \in E$, como un subcojunto de dos elementos $e = \{u, v\}$ siendo $u, v \in V$, llamando a u y v los puntos terminales de e.

Un grafo direccionado $G = (V, E)$ consiste de una colección de vértices, V y una colección de arcos, E. Representamos cada arco, $e \in E$, como un par ordenado de dos elementos $e = (u, v)$ siendo $u, v \in V$. Llamamos a u el punto inicial y a v el punto final de e.

- Un grafo no direccionado está conectado si, para cada par de nodos u y v, existe un camino de u a v.
- Un grafo direccionado está fuertemente conectado si, para cada par de vértices u y v, existe un camino de u a v y de v a u.
- En un grafo no direccionado $G = (V, E)$ una secuencia de nodos P $= [v_1$, v_2, ..., v_{k-1}, $v_k]$ con la propiedad de que cada par consecutivo v_i, v_{i+1} está conectado por un arco en G, es llamado un camino (path) de v_1 a v_k.
- Un ciclo es un camino P $= [v_1$, v_2, ..., v_{k-1}, $v_k]$ en el cual, para cualquier $k > 2$, los primeros k˘1 vértices son distintos y $v_1 = v_k$.

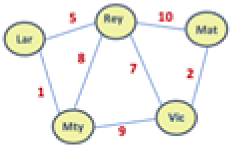
En muchas ocasiones requerimos saber la ruta mas corta de un punto a El algoritmo de Dijkstra también conocido como el algoritmo de caminos mínimos, es un algoritmo desarrollado por el científico en computación Edsger Dijkstra para determinar el camino más corto para un grafo ponderado dado un vértices inicial al resto de los vértices, utilizando la técnica de Algoritmos Avaros $(O(|E| + |V|log|V|))$.


Paso 1
Paso 2
Paso 3
Paso 4
Lar
Rey
Mat
Mty
Vic
5
10
8
7
2
1
9

Procedure 1 DIJKSTRA

```
Input: u : Vertex, G : Graph
  Set reached
  PiorityQueue pending
  pending.enqueue(PAIR(u, 0))
  while pending is not empty do
    e ← pending.dequeue()
    if e.first is not in reached then
      Add to Reached
      for each (e, v) incident to e do
        Q.enqueue(PAIR(v, COST(e, v) + e.second))
      end for
    end if
  end while
  return reached
```

El algoritmo Floyd – Warshall permite encontrar las rutas más cortas en un grfo ponderado con costos positivos o negativos (pero sin ciclos negativos). Una sola ejecución del algoritmo encontrará los costos de las rutas más cortas entre **todos los pares de vértices**. Aunque no devuelve detalles de las rutas en sí, es posible reconstruir las rutas con simples modificaciones al algoritmo. Las versiones del algoritmo también se pueden usar para encontrar el cierre transitivo de una relación R.



1 = Rey
2 = Lar
3 = Mty
4 = Vic
5 = Mat

K = 0

	1	2	3	4	5
1	0	5	8	7	10
2	5	0	1	INF	INF
3	8	1	0	9	INF
4	7	INF	9	0	2
5	10	INF	INF	2	0

K = 1

	1	2	3	4	5
1	0	5	8	7	10
2	5	0	1	12	15
3	8	1	0	9	18
4	7	12	9	0	2
5	10	15	18	2	0

K = 2

	1	2	3	4	5
1	0	5	6	7	10
2	5	0	1	12	15
3	6	1	0	9	16
4	7	12	9	0	2
5	10	15	16	2	0

K = 3

	1	2	3	4	5
1	0	5	6	7	10
2	5	0	1	10	15
3	6	1	0	9	16
4	7	10	9	0	2
5	10	15	16	2	0

K = 4

	1	2	3	4	5
1	0	5	6	7	9
2	5	0	1	10	12
3	6	1	0	9	11
4	7	10	9	0	2
5	9	12	11	2	0

K = 5

	1	2	3	4	5
1	0	5	6	7	9
2	5	0	1	10	12
3	6	1	0	9	11
4	7	10	9	0	2
5	9	12	11	2	0

Procedure 2 VERSION1

Input: $M : Adjacent\_Matrix$

```
for k ← 1 to M.length do
  for i ← 1 to M.length do
    for j ← 1 to M.length do
      M[i][j] ← M[i][k] and M[k][j]
    end for
  end for
end for
```

Procedure 3 VERSION2

Input: $M : Adjacent\_Matrix$

```
for k ← 1 to M.length do
    for i ← 1 to M.length do
        for j ← 1 to M.length do
            M[i][j] ← MIN(M[i][j], M[i][k] + M[k][j])
        end for
    end for
end for
```