Matemática Discreta

Año: 2019

Trabajo integrador

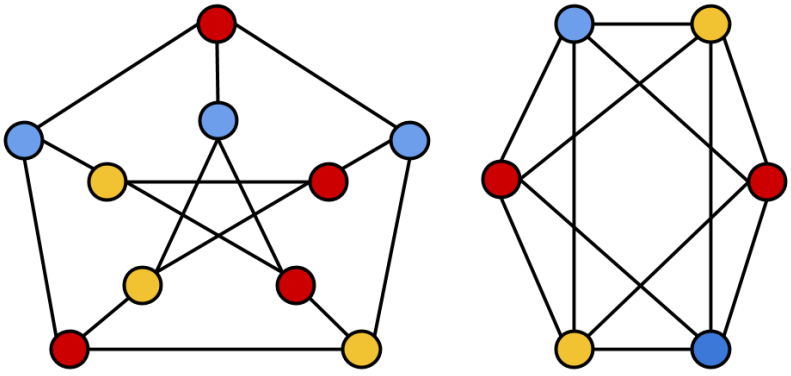
Algoritmos de búsqueda en grafos

Godoy Moreno, Thomas; La Madrid, Leonel Federico; Salim Taleb, Nasim Anibal.

Docentes: Insfrán, Jordán; Taborda, Liliana.

Carrera: Lic. en Bioinformática.

Fecha de entrega: 06/11/2019



Actividades

1.- Algoritmo Floyd-Warshall: El algoritmo de Floyd-Warshall tiene como objetivo encontrar el camino más corto entre los vértices, en grafos que sean dirigidos y tengan un peso asociado a cada arco.

- Algoritmo de Dijkstra: también conocido como algoritmo de los caminos mínimos, es un algoritmo para la determinación del camino más corto, dado un vértice origen, hacia alguno de los restantes vértices del grafo.

- Algoritmo de Warshall: Sirve para encontrar la cerradura transitiva de una relación binaria en el conjunto A. La clausura transitiva de una relación binaria es la relación binaria más pequeña que siendo transitiva contenga el conjunto de pares de la relación binaria original. No tiene en cuenta la distancia entre los vértices.

- Algoritmo de Prim: se emplea para encontrar el MST es decir los arboles generadores mínimos, un árbol es un grafo en el que cualquier par nodos están conectados por exactamente un camino.

- Algoritmo de Hierholzer: también conocido como ciclo euleriano, busca encontrar un camino el cual pase por todos los arcos del grafo una única vez.

2. a- Usaríamos el algoritmo Dijkstra, debido a que se nos solicita una ruta entre dos puntos específicos (origen, fin) minimizando el costo.

b- Emplearíamos el algoritmo de Floyd-Warshall, porque al obtener la matriz de relaciones entre todos los nodos por comparación podríamos determinar el punto más óptimo para la sede.

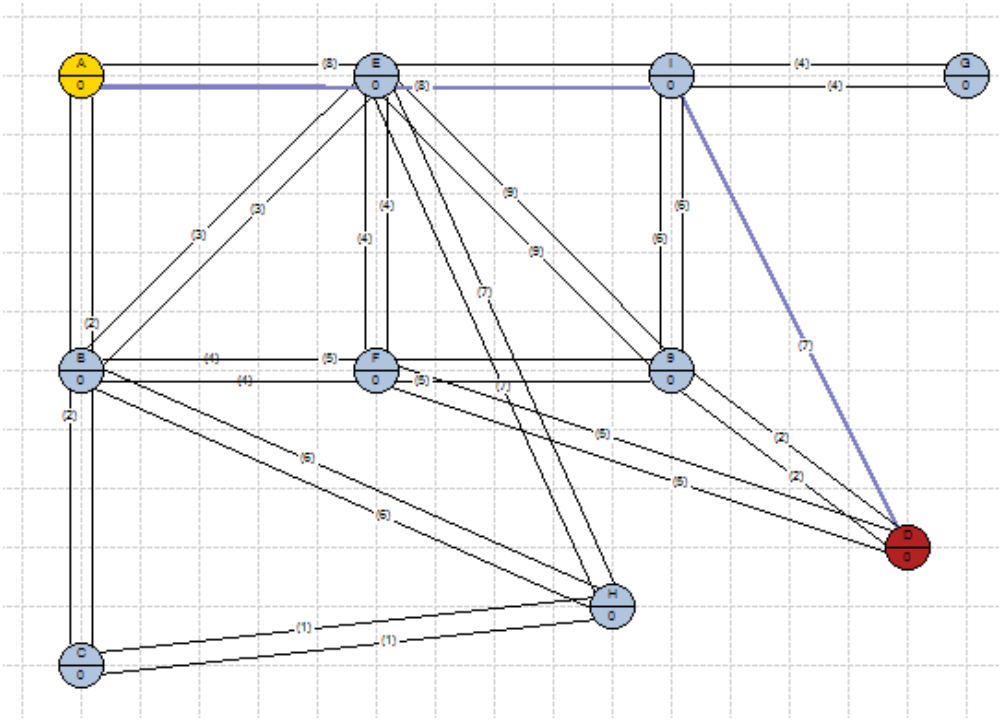
c- Emplearíamos el algoritmo de Warshall ya que este verificaría la propiedad de cerradura transitiva y nos permitiría saber si todos los nodos de la red se encuentran conectados entre sí.

d- Usaríamos el algoritmo de Hierholzer o ciclo euleriano ya que nos permite encontrar un camino el cual pase por todos los arcos del grafo una única vez.
Consideramos que las calles son los nodos, ya que esto nos asegura que cada nodo tendrá grado par y nos asegura el uso del ciclo euleriano.

3. a- Circuito Euleriano: Un camino euleriano es una trayectoria cerrada por la cual pueden recorrerse todos los arcos de un grafo una única vez. Tiene como requerimiento que existan como máximo 2 nodos de grado impar (con número impar de arcos unidos al nodo).

b. Circuito Hamiltoniano: Un circuito hamiltoniano es un camino por el cual se recorren todo los nodos o vértices de un grafo una sola vez.

4. a-

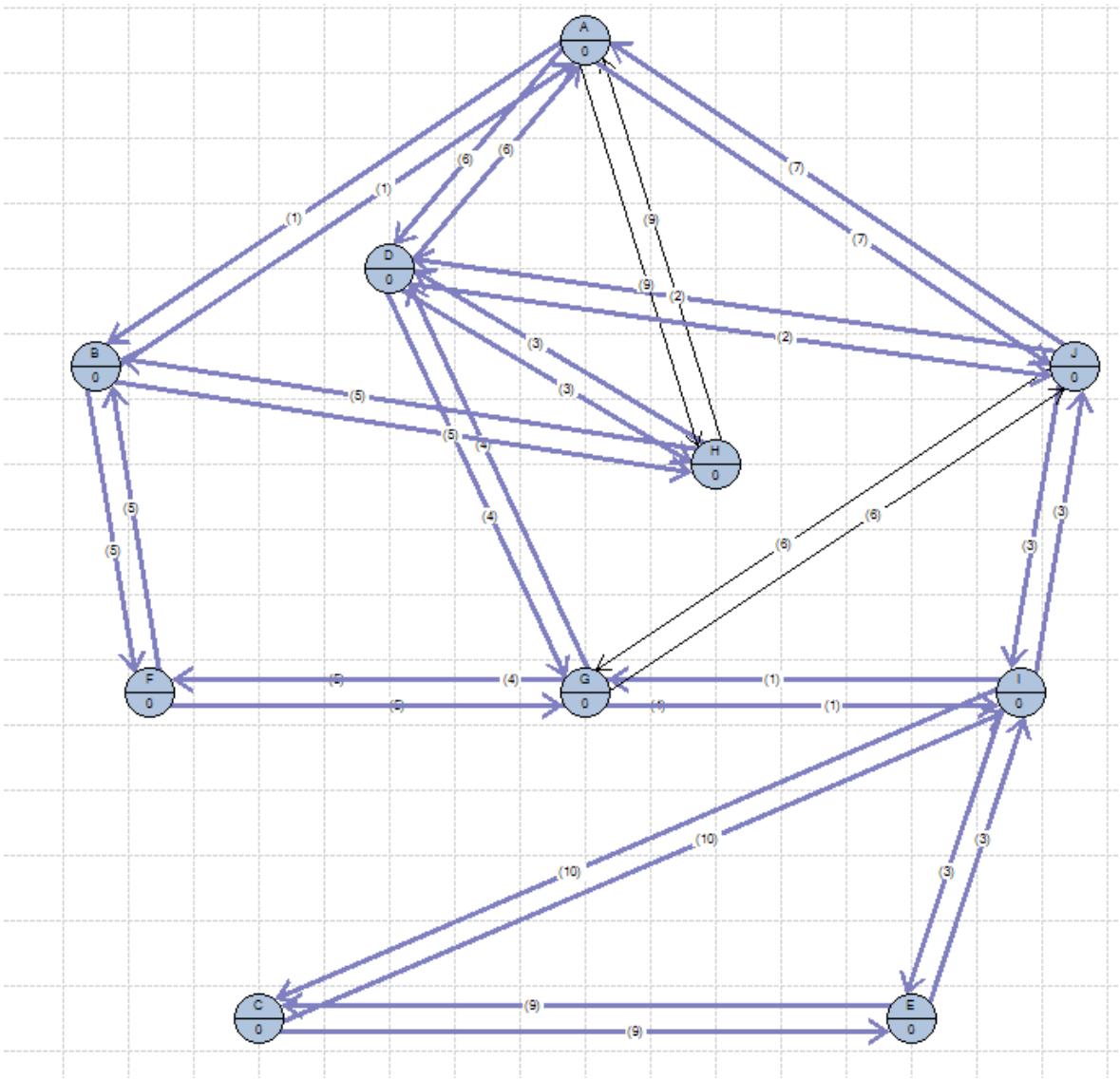


Arcos calculados desde el nodo origen (A) hasta el nodo destino (D):

* A ---- (8) ---> I
* I ---- (7) ---> D

Coste total = 15

c-

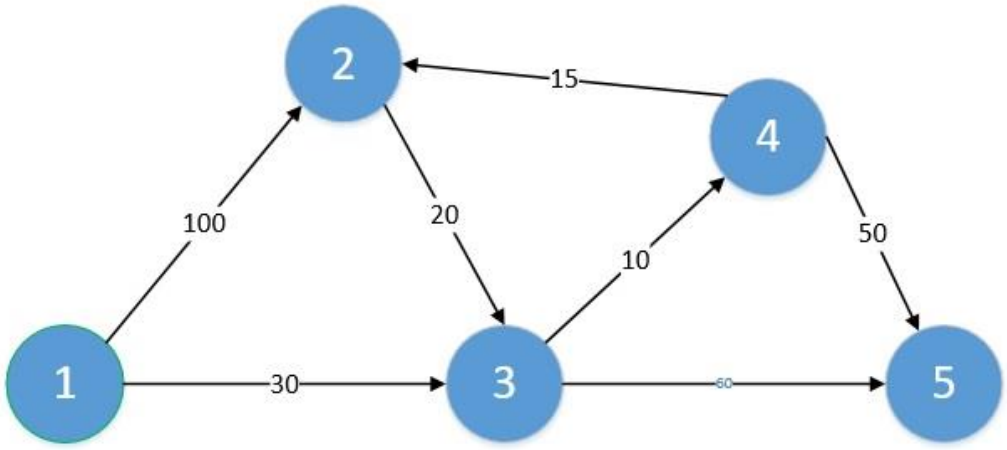


Matriz de distancias minimas:

N1\N2	A	B	C	D	E	F	G	H	I	J	Totales
A	0	1	20	6	13	6	10	6	10	7	79
B	1	0	19	7	12	5	10	5	9	8	76
C	20	19	0	15	9	14	11	18	10	13	129
D	6	7	15	0	8	9	4	3	5	2	59
E	13	12	9	8	0	7	4	11	3	6	73
F	6	5	14	9	7	0	5	10	4	7	67
G	10	10	11	4	4	5	0	7	1	4	56
H	6	5	18	3	11	10	7	0	8	5	73
I	10	9	10	5	3	4	1	8	0	3	53
J	7	8	13	2	6	7	4	5	3	0	55

La sede se ubicara en I, al tener la menor distancia hacia todas las localidades.

5) a. Algoritmo de Dijkstra: Su metodología se basa en iteraciones, de manera tal que en la práctica, su desarrollo se dificulta a medida que el tamaño de la red aumenta, dejándolo en clara desventaja, frente a métodos de optimización basados en programación matemática. El algoritmo de Dijkstra hace uso y define etiquetas a partir del nodo origen y para cada uno de los nodos subsiguientes. Estas etiquetas contienen información relacionada con un valor acumulado del tamaño de los arcos y con la procedencia más próxima de la ruta.
Las etiquetas corresponden a los nodos, no a los arcos. En el algoritmo de Dijkstra, estas etiquetas son temporales y permanentes. Las etiquetas temporales son aquellas que son susceptibles de modificarse mientras exista la posibilidad de hallar para sí, una ruta más corta; de lo contrario, dicha etiqueta pasa a ser permanente.
Teniendo en cuenta la siguiente red, que tiene como origen el nodo 1:



Procedemos con las iteraciones, partimos de la iteración 0, es decir, asignar la etiqueta para el nodo origen:

Iteración 0: En este paso, asignamos una etiqueta permanente para el nodo origen. Recordemos que la etiqueta se compone por un valor acumulado, que en este caso debe ser 0, ya que es el punto base y que no existe procedencia:

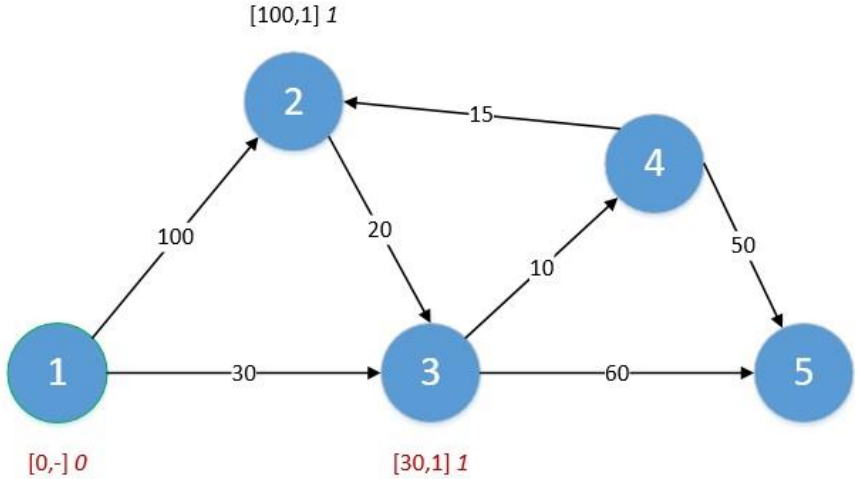
Iteración 1: En este paso, evaluamos a cuáles nodos se puede llegar desde el nodo 1. En este caso se puede llegar a los nodos 2 y 3. De manera que debemos asignar las etiquetas para cada nodo:

Etiqueta nodo 2 = [valor acumulado, procedencia] iteración
Etiqueta nodo 2 = [0 + 100, 1] 1
0 es el valor acumulado en la etiqueta del nodo de procedencia, en este caso el valor acumulado para el nodo 1. 100 es el valor del arco que une el nodo procedencia (1) y el nodo destino (2). 1 es el número de la iteración.

Etiqueta nodo 3 = [valor acumulado, procedencia] iteración

Etiqueta nodo 3 = [0 + 30, 1] 1

En este caso, podemos observar que la etiqueta del nodo 3, ya contiene el menor valor acumulado posible para llegar a este. Ya que 30 es la mínima distancia posible, toda vez que para llegar al nodo 3 por medio del nodo 2, tendrá que recorrer como mínimo el valor absoluto del arco que une el origen con el nodo 2 = 100. Así entonces, la etiqueta del nodo 3 pasa a ser permanente.



Iteración 2: En este paso, evaluamos las posibles salidas desde el nodo 3, es decir los nodos 4 y 5. De manera que debemos asignar las etiquetas para cada nodo:

Etiqueta nodo 4 = [valor acumulado, procedencia] iteración

Etiqueta nodo 4 = [30 + 10, 3] 2

Etiqueta nodo 4 = [40, 3] 2

30 es el valor acumulado en la etiqueta del nodo de procedencia, en este caso el valor acumulado para el nodo 3. 10 es el valor del arco que une el nodo procedencia (3) y el nodo destino (4). 2 es el número de la iteración.

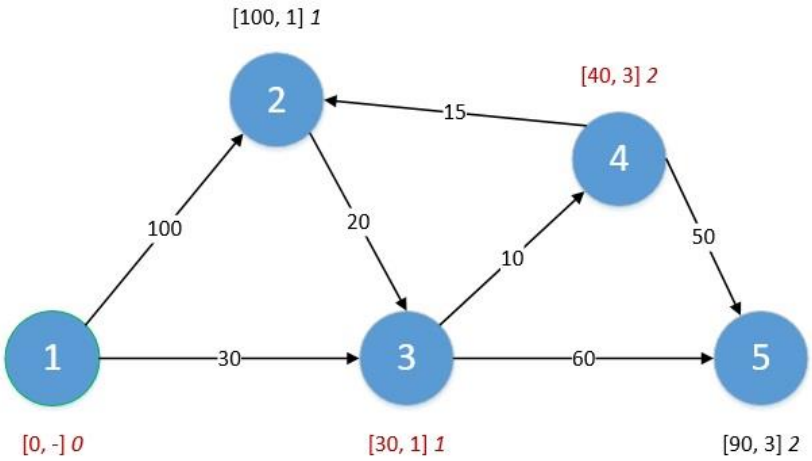
Etiqueta nodo 5 = [valor acumulado, procedencia] iteración

Etiqueta nodo 5 = [30 + 60, 3] 2

Etiqueta nodo 5 = [90, 3] 2

30 es el valor acumulado en la etiqueta del nodo de procedencia, en este caso el valor acumulado para el nodo 3. 60 es el valor del arco que une el nodo procedencia (3) y el nodo destino (5). 2 es el número de la iteración.

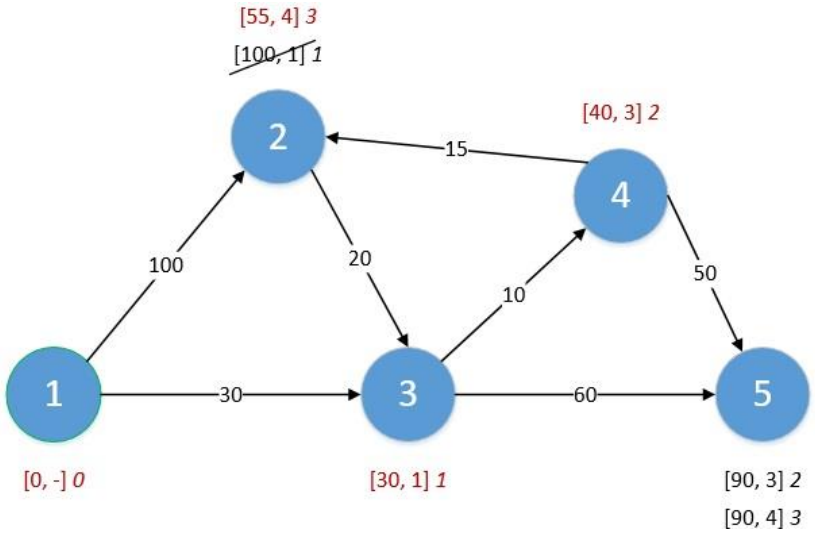
En este caso, podemos observar que la etiqueta del nodo 4, contiene el menor valor acumulado posible para llegar a este. Así entonces, la etiqueta del nodo 4 pasa a ser permanente.



Iteración 3: En este paso, evaluamos las posibles salidas desde el nodo 4, es decir los nodos 2 y 5. De manera que debemos asignar las etiquetas para cada nodo:

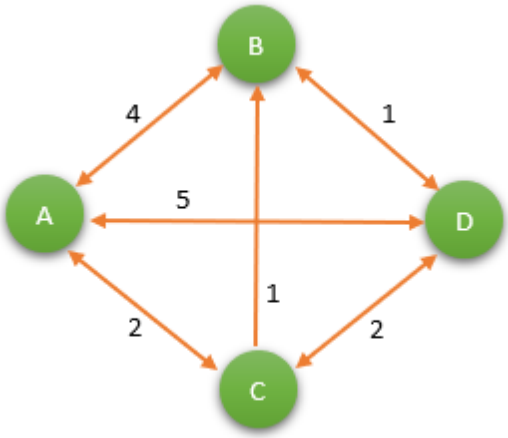
Etiqueta nodo 2 = [valor acumulado, procedencia] iteración
Etiqueta nodo 2 = [40 + 15, 4] 3
Etiqueta nodo 2 = [55, 4] 3
40 es el valor acumulado en la etiqueta del nodo de procedencia, en este caso el valor acumulado para el nodo 4. 15 es el valor del arco que une el nodo procedencia (4) y el nodo destino (2). 3 es el número de la iteración.

Etiqueta nodo 5 = [valor acumulado, procedencia] iteración
Etiqueta nodo 5 = [40 + 50, 4] 3
Etiqueta nodo 5 = [90, 4] 3
40 es el valor acumulado en la etiqueta del nodo de procedencia, en este caso el valor acumulado para el nodo 3. 50 es el valor del arco que une el nodo procedencia (4) y el nodo destino (5). 3 es el número de la iteración.



Iteración 4: En este paso, evaluamos las posibles salidas desde el nodo 2 y el nodo 5. Sin embargo, el nodo 2 solo tiene un posible destino, el nodo 3, el cual ya tiene una etiqueta permanente, de manera que no puede ser reetiquetado. Ahora, evaluamos el nodo 5 y es un nodo que no tiene destinos. Así entonces, su etiqueta temporal pasa a ser permanente, en este caso cuenta con 2 etiquetas que tienen el mismo valor, es decir, alternativas óptimas. De esta manera concluye el algoritmo de Dijkstra.

b. Para este ejemplo, supongamos un grafo dirigido, como el que se muestra a continuación:



Matriz de distancias				
	A	B	C	D
A	-	4	2	5
B	4	-	∞	1
C	2	5	-	2
D	5	1	2	-

Matriz de recorridos				
	A	B	C	D
A	-	B	C	D
B	A	-	C	D
C	A	B	-	D
D	A	B	C	-

El algoritmo de Warshall es un método iterativo, por lo que nuestro grafo al ser de 4 nodos y por consiguiente nuestras matrices de distancia y recorrido de tamaño 4 x 4, realizaremos para este ejemplo, 4 iteraciones. (k = 1, 2, 3, 4).

Comenzamos con nuestra iteración k = 1, dónde en nuestra matriz de distancias “cancelaremos” nuestra fila 1 y nuestra columna 1.

A continuación, analizaremos los valores que no fueron “cancelados”, y la idea es mejorar cada uno de esos valores, con excepción de las celdas marcadas con - . Por lo que nos plantearemos: ¿la suma de la celda (2,1) 4 + la celda (1,3) 2 es menor a la celda (2,3) ∞? Si, por lo tanto ∞ será reemplazado por la suma, que da 6.

De tal manera que habrás que hacer este análisis con cada una de las celdas no “canceladas” o en este caso marcadas en amarillo.

Matriz de distancias				
	A	B	C	D
A	-	4	2	5
B	4	-	∞	1
C	2	5	-	2
D	5	1	2	-

Quedando de esta manera:

Matriz de distancias				
	A	B	C	D
A	-	4	2	5
B	4	-	6	1
C	2	5	-	2
D	5	1	2	-

Y para nuestra matriz de recorridos, en la celda (2,3) que fue donde realizamos el cambio de ∞ a 6, cambiaremos la letra **C** por la letra **A**, que será nuestro nodo antecesor. De tal manera que en nuestra matriz de recorridos, realizaremos modificaciones en las mismas celdas en las que realizamos cambios en la matriz de distancias.

Matriz de recorridos				
	A	B	C	D
A	-	B	C	D
B	A	-	A	D
C	A	B	-	D
D	A	B	C	-

Se realiza el mismo procedimiento que en el paso anterior, hasta que k sea igual a n, es decir, en que en nuestro ejemplo repetiremos el procedimiento anterior hasta que k = 4.

Para **k = 2** ahora tachamos nuestra fila y columna **B** en nuestra matriz de distancias.

Matriz de distancias				
	A	B	C	D
A	-	4	2	5
B	4	-	6	1
C	2	5	-	2
D	5	1	2	-

Matriz de recorridos				
	A	B	C	D
A	-	B	C	D
B	A	-	C	D
C	A	B	-	D
D	A	B	C	-

Realizando el procedimiento nos damos cuenta de que no hay ninguna mejora (cambio) en la matriz de distancias, por lo que la matriz anterior y la de recorridos permanecen igual.

Para **k = 3** ahora tachamos la fila y columna **C**.

Matriz de distancias				
	A	B	C	D
A	-	4	2	5
B	4	-	6	1
C	2	5	-	2
D	5	1	2	-

Matriz de distancias				
	A	B	C	D
A	-	4	2	4
B	4	-	6	1
C	2	5	-	2
D	4	1	2	-

La matriz de recorridos será:

Matriz de recorridos				
	A	B	C	D
A	-	B	C	C
B	A	-	C	D
C	A	B	-	D
D	C	B	C	-

Para **k = 4** (nuestra última iteración) tachamos la columna y fila **D**.

Matriz de distancias				
	A	B	C	D
A	-	4	2	5
B	4	-	6	1
C	2	5	-	2
D	5	1	2	-

Matriz de distancias				
	A	B	C	D
A	-	4	2	4
B	4	-	3	1
C	2	3	-	2
D	4	1	2	-

Matriz de recorridos				
	A	B	C	D
A	-	B	C	C
B	A	-	D	D
C	A	D	-	D
D	C	B	C	-

Para interpretar nuestros resultados, basándonos en nuestra tabla de recorridos podemos concluir que:

• Para ir del nodo A al nodo B, llegaremos directo al B.

• Para ir del nodo A al nodo D, tendremos como nodo intermedio el nodo C.

• Para ir del nodo B al nodo C, tendremos que pasar por el nodo D.

• Para ir del nodo D al nodo C, llegaremos directamente al C.

Y así, podremos indicar cuales son las rutas de un nodo a otro, siendo estas rutas las más cortas entre ambos nodos.

Bibliografía

(10 de abril de 2016), Estructuras de Datos II, extraído el 2 de noviembre de 2019 https://estructurasite.wordpress.com/

(29 de octubre de 2015), Algoritmo de Warshall, extraído el 2 de noviembre de 2019 http://estructdatos2incca.blogspot.com/2015/10/algoritmo-de-warshall.html

(2016) Algoritmo de Dijkstra, extraído el 5 de noviembre de 2019.
https://www.ingenieriaindustrialonline.com/herramientas-para-el-ingeniero-industrial/investigacion-de-operaciones/algoritmo-de-dijkstra/

Algoritmo de Floyd-Warshall, extraído el 5 de noviembre de 2019
https://www.academia.edu/38918580/Algoritmo_de_Floyd_Floyd-Warshall