Apuntes

Tema ***Grafos – Algoritmo de Floyd o Floyd-Warshall***

"Una computadora puede ser llamada inteligente si logra engañar a una persona haciéndole creer que es un humano."
Alan Mathison Turing.

Grafos

Algoritmo de Floyd o Floyd-Warshall

En informática, el algoritmo de Floyd-Warshall, descrito en 1959 por **Bernard Roy**, es un algoritmo de análisis sobre grafos para encontrar el camino mínimo en grafos dirigidos ponderados. El algoritmo encuentra el camino entre todos los pares de vértices en una única ejecución. El algoritmo de Floyd-Warshall es un ejemplo de programación dinámica.


Bernard Roy

El algoritmo de Floyd es más general que el de Dijkstra, ya que determina la ruta más corta entre dos nodos cualesquiera de la red (lo que significa que calcula la ruta más corta entre todos los pares de nodos).

El algoritmo representa una red de **n** nodos como una matriz cuadrada de orden n, la llamaremos matriz **C**. De esta forma, el valor **Cij** representa el coste de ir desde el nodo **i** al nodo **j**, inicialmente en caso de no existir un arco entre ambos, el valor **Cij** será ***infinito(∞)***.

Definiremos otra matriz **D**, también cuadrada de orden **n**, cuyos elementos van a ser los nodos predecesores en el camino hacia el nodo origen, es decir, el valor **Dij** representará el nodo predecesor a **j** en el camino mínimo desde **i** hasta **j**. Inicialmente se comienza con caminos de longitud **1**, por lo que **Dij = i**.

Las diagonales de ambas matrices representan el coste y el nodo predecesor para ir de un nodo a si mismo, por lo que no sirven para nada, ***estarán bloqueadas***.

Los pasos a dar en la aplicación del algoritmo de Floyd son los siguientes:

- Formar las matrices iniciales C y D; Se toma k=1.
- Se selecciona la fila y la columna k de la matriz C y entonces, para i y j, con i≠k, j≠k e i≠j, hacemos:

  Si (Cik + Ckj) < Cij → Dij = Dkj y Cij = Cik + Ckj

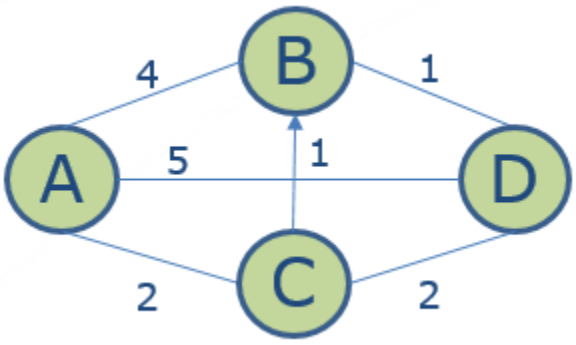
  En caso contrario, dejamos las matrices como están.
- Si k ≤ n, aumentamos k en una unidad y repetimos el paso anterior, en caso contrario paramos las iteraciones.

La matriz final C contiene los costes óptimos para ir de un vértice a otro, mientras que la matriz D contiene los penúltimos vértices de los caminos óptimos que unen dos vértices, lo cual permite reconstruir cualquier camino óptimo para ir de un vértice a otro.

Matriz C, de distancias

Matriz D, de recorridos

Ejemplo:

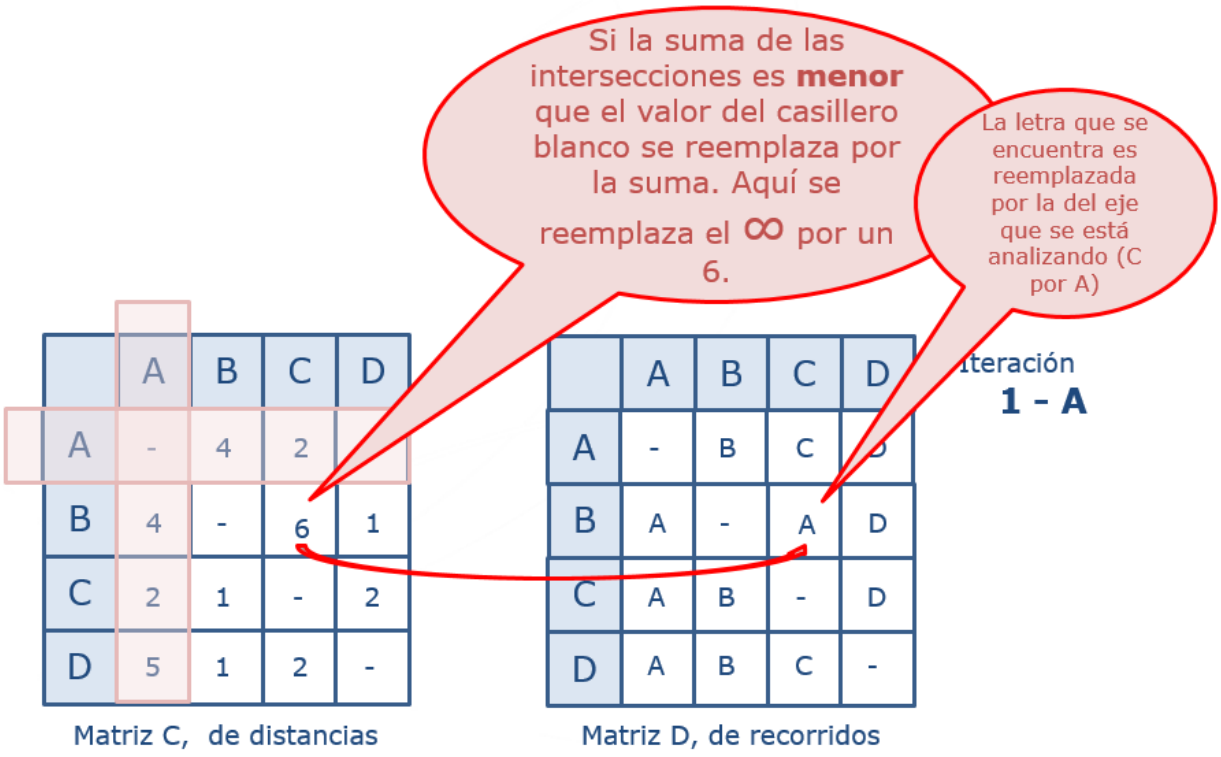


	A	B	C	D
A	-	4	2	5
B	4	-	∞	1
C	2	1	-	2
D	5	1	2	-

Matriz C, de distancias

	A	B	C	D
A	-	B	C	D
B	A	-	C	D
C	A	B	-	D
D	A	B	C	-

Matriz D, de recorridos



	A	B	C	D
A	-	4	2	
B	4	-	6	1
C	2	1	-	2
D	5	1	2	-

Matriz C, de distancias

	A	B	C	D
A	-	B	C	D
B	A	-	A	D
C	A	B	-	D
D	A	B	C	-

Matriz D, de recorridos

No hay reemplazos

	A	B	C	D
A	-	4	2	5
B	4	-	6	1
C	2	1	-	2
D	5	1	2	-

Matriz C, de distancias

	A	B	C	D
A	-	B	C	D
B	A	-	A	D
C	A	B	-	D
D	A	B	C	-

Matriz D, de recorridos

Iteración
1 - A
2 - B

	A	B	C	D
A	-	3	2	4
B	4	-	6	1
C	2	1	-	2
D	4	1	2	-

Matriz C, de distancias

	A	B	C	D
A	-	C	C	C
B	A	-	A	D
C	A	B	-	D
D	C	B	C	-

Matriz D, de recorridos

Iteración
1 - A
2 - B
3 - C

	A	B	C	D
A	-	3	2	4
B	4	-	3	1
C	2	1	-	2
D	4	1	2	-

Matriz C, de distancias

	A	B	C	D
A	-	C	C	C
B	A	-	D	D
C	A	B	-	D
D	C	B	C	-

Matriz D, de recorridos

Iteración
1 - A
2 - B
3 - C
4 - D

	A	B	C	D
A	-	3	2	4
B	4	-	3	1
C	2	1	-	2
D	4	1	2	-

Matriz C, de distancias

	A	B	C	D
A	-	C	C	C
B	A	-	D	D
C	A	B	-	D
D	C	B	C	-

Matriz D, de recorridos

Iteración
1 - A
2 - B
3 - C
4 - D