CURSO DE PROGRAMACIÓN COMPETITIVA
URJC - 2018

Sesión 5 (7ª Semana)

David Morán (ddavidmorang@gmail.com)
Juan Quintana (juandavid.quintana@urjc.es)
Sergio Pérez (sergioperezp1995@gmail.com)

Contenidos

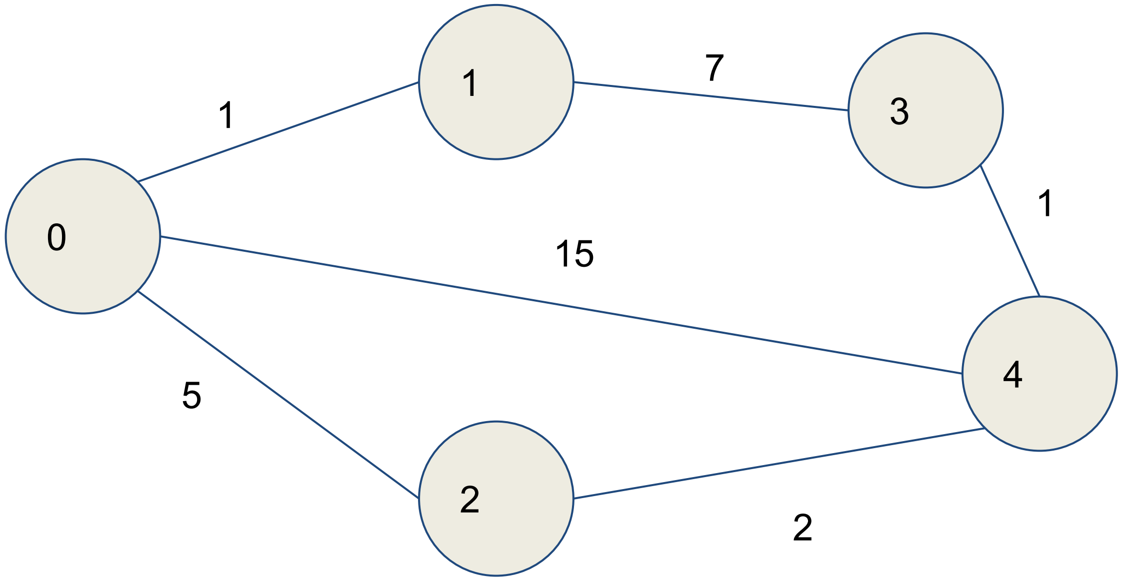
- Grafos (Ponderados)
    - Algoritmos de caminos más cortos
    - Floyd Warshall
    - Dijkstra

Grafos Ponderados

- En la vida real, el camino más corto entre dos puntos no es siempre el número de aristas por las que se pasa de un vértice i a un j cualquiera
- Los grafos muchas veces son ponderados

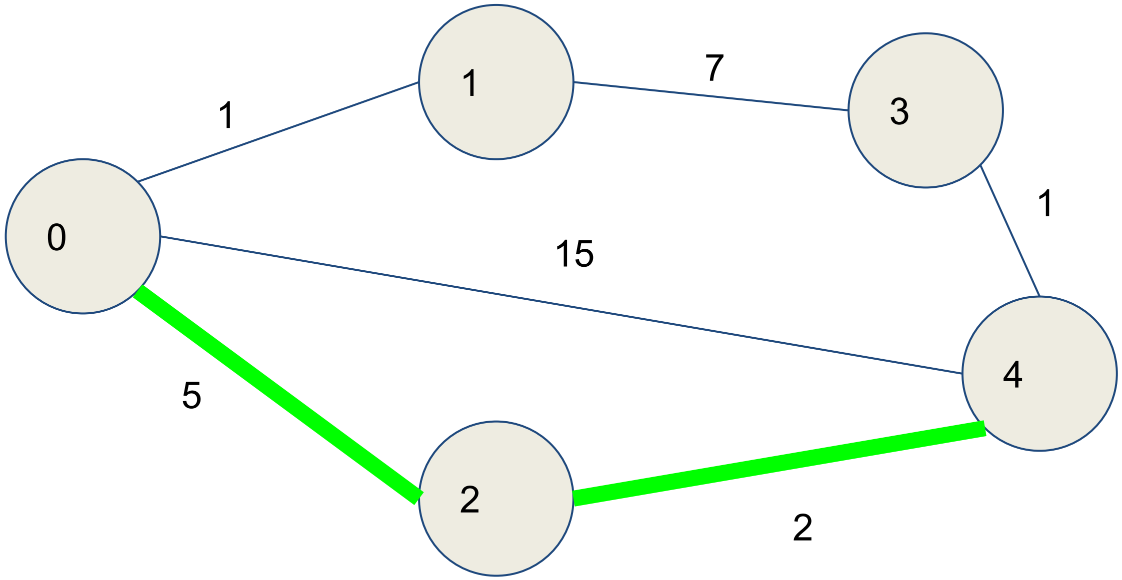
Grafos Ponderados

- ¿Cuál es el camino más corto del siguiente grafo para ir de 0 a 4?



Grafos Ponderados

- ¿Cuál es el camino más corto del siguiente grafo para ir de 0 a 4?



Grafos Ponderados

- Algoritmo de Floyd Warshall
- Compara todos los puntos para ir de cualquier nodo i a cualquier nodo j considerando un posible tercer nodo k
- El mínimo (*) entre todos estos objetivos siempre dará como resultado el camino mínimo entre cualquier par de nodos i,j dentro del grafo

Grafos Ponderados

- Algoritmo de Floyd Warshall
- (Contra) Es N^3 por lo que su utilización está bastante limitada
- (Pro) Funciona sobre cualquier par de vértices
- (Pro) es muy fácil de escribir
- (Pro) suele actuar sobre matrices de adyacencia, más fácil de implementar

Grafos Ponderados

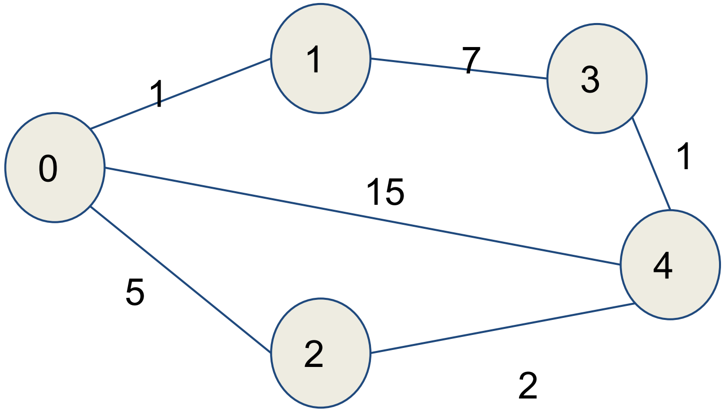
- Algoritmo de Floyd Warshall

```
for k from 0..N
   for i from 0..N
      for j from 0..N
         graph[i][j] = min(
graph[i][j], graph[i][k] +
graph[k][j])
```

Grafos Ponderados

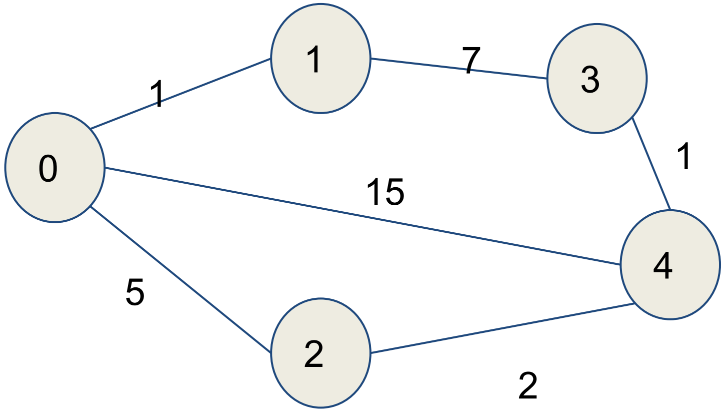
- Algoritmo de Floyd Warshall
- Por cada par de vértices en la matriz de adyacencia i,j veremos si es mejor el camino actual que tenemos ó hacer un camino nuevo pasando por cualquier otro nodo k entre medias
- Si no existe un camino entre i,j; graph[i][j] = INF

Grafos Ponderados



0	1	5	INF	15
1	0	INF	7	INF
5	INF	0	INF	2
INF	7	INF	0	1
15	INF	2	1	0

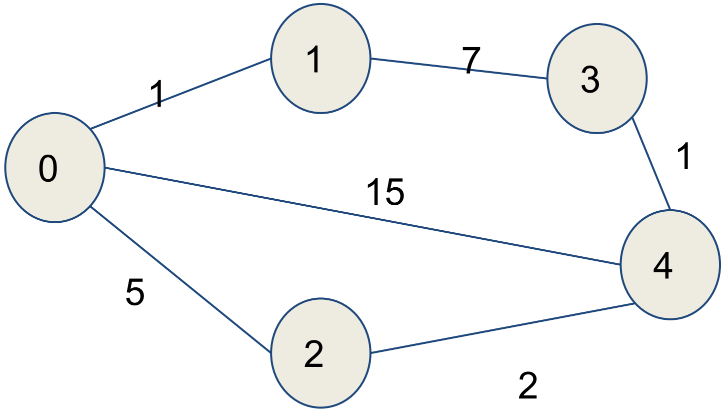
Grafos Ponderados



MIN({0,4},
{0,2}+{2,4})
MIN(15, 5+2) = 7

0	1	5	INF	15 (7)
1	0	INF	7	INF
5	INF	0	INF	2
INF	7	INF	0	1
15 (7)	INF	2	1	0

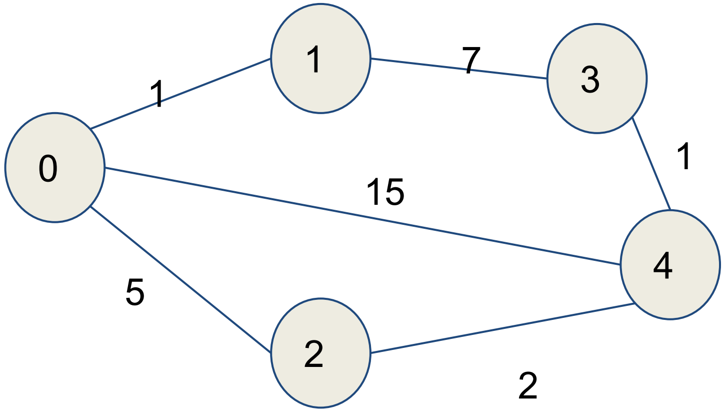
Grafos Ponderados



MIN({0,3},
{0,1}+{1,3})
MIN(INF, 1+7) = 8

0	1	5	INF (8)	15 (7)
1	0	INF	7	INF
5	INF	0	INF	2
INF (8)	7	INF	0	1
15 (7)	INF	2	1	0

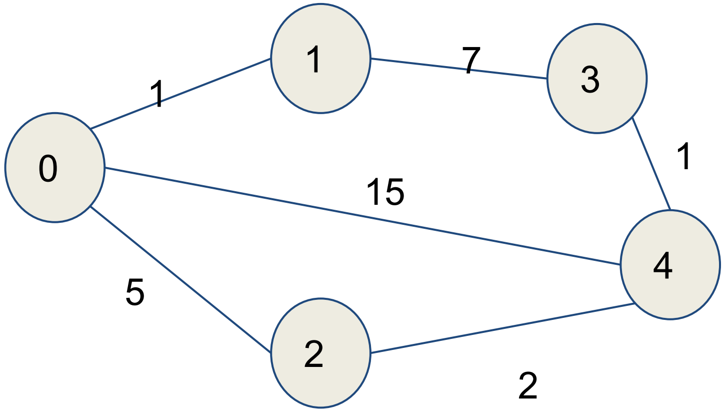
Grafos Ponderados



MIN({0,4},
{0,3}+{3,4})
MIN(7, 8+1) = 7

0	1	5	INF (8)	15 (**7**)
1	0	INF	7	INF
5	INF	0	INF	2
INF (8)	7	INF	0	1
15 (**7**)	INF	2	1	0

Grafos Ponderados



MUCHAS ITERACIONES DESPUÉS...

0	1	5	8	7
1	0	6	7	8
5	6	0	3	2
8	7	3	0	1
7	8	2	1	0

Grafos Ponderados

- Algoritmo de Dijkstra
- Dado un nodo inicio y un nodo fin, computa en O(VLg(E)) la distancia más corta entre esos puntos
- Algoritmo voraz
- No computa la distancia de todos los pares de nodo
- Computa la distancia de i hacia todos

Grafos Ponderados

- Algoritmo de Dijkstra
- Guarda en un array de distancias todas las distancias desde i hacia el resto
- La implementación es muy parecida a un BFS
- Es casi igual al algoritmo de Prim

Grafos Ponderados

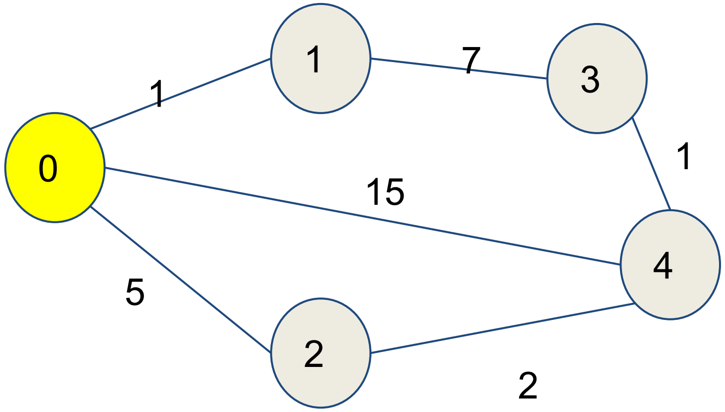
- Algoritmo de Dijkstra

```
function dijkstra(start, end)
  array_dist(start) = 0
  prio_queue.encolar({start, 0})
  mientras !prio_queue.vacio()
    n = prio_queue.desencolar()
    si array_dist[n.node] >= n.dist
      para cada arista D en n.node
        peso = n.dist + D.weight
        si array_dist[D.to] > peso
          prio_queue.encolar({ D.to, peso })
          array_dist[D.to] = peso
  retornar array_dist[end]
```

Grafos Ponderados

- Algoritmo de Dijkstra
- Encolamos el primer nodo con distancia 0
- Tenemos una cola de prioridad ordenando de menor a mayor según la sumatoria del peso de los nodos
- Si la distancia hasta ese nodo es menor o igual, descartamos, si no, encolamos
- Retornamos al final la distancia hasta end ó, si en algún momento de la cola nos topamos con end podemos retornar directamente (es un voraz, no conseguiremos nada mejor)

Grafos Ponderados

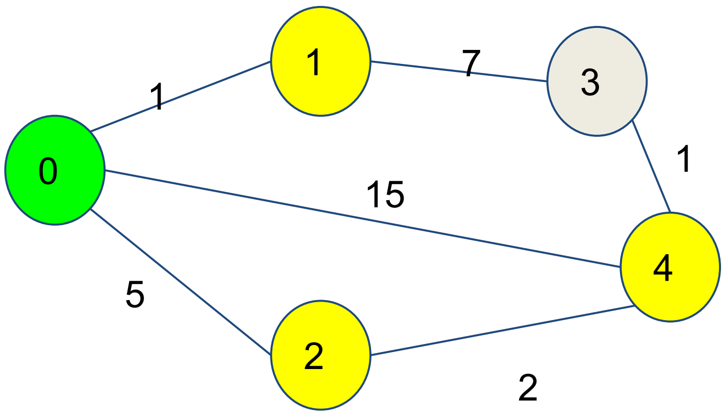


PQ = { {0, 0} }

Dist = { 0, INF, INF, INF, INF}

0	1	5	INF	15
1	0	INF	7	INF
5	INF	0	INF	2
INF	7	INF	0	1
15	INF	2	1	0

Grafos Ponderados

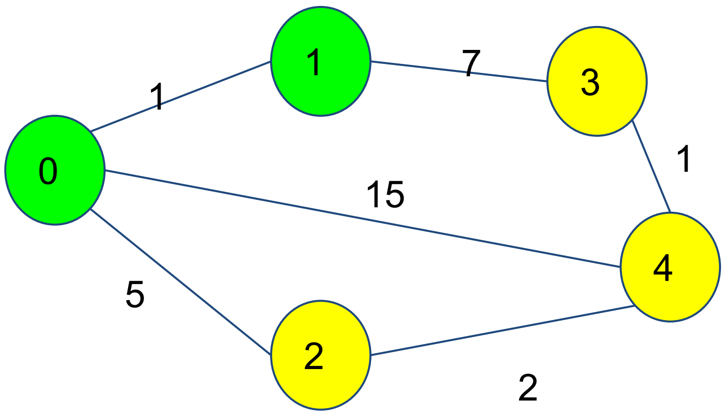


PQ = { {1, 1}, {2, 5}, {4, 15} }

Dist = { 0, 1, 5, INF, 15}

0	1	5	INF	15
1	0	INF	7	INF
5	INF	0	INF	2
INF	7	INF	0	1
15	INF	2	1	0

Grafos Ponderados

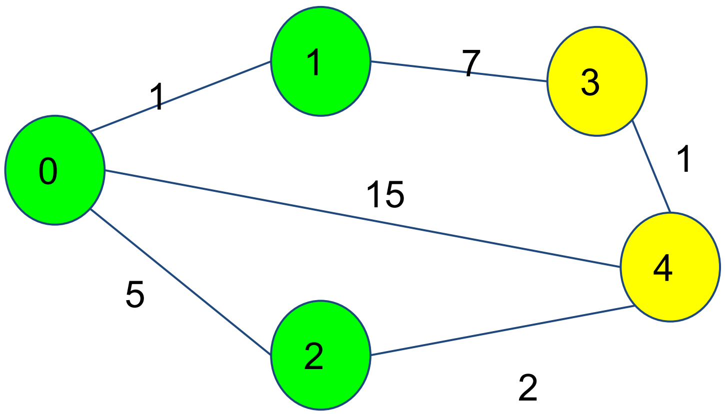


PQ = { {2, 5}, {3, 8}, {4, 15} }

Dist = { 0, 1, 5, 8, 15}

0	1	5	INF	15
1	0	INF	7	INF
5	INF	0	INF	2
INF	7	INF	0	1
15	INF	2	1	0

Grafos Ponderados

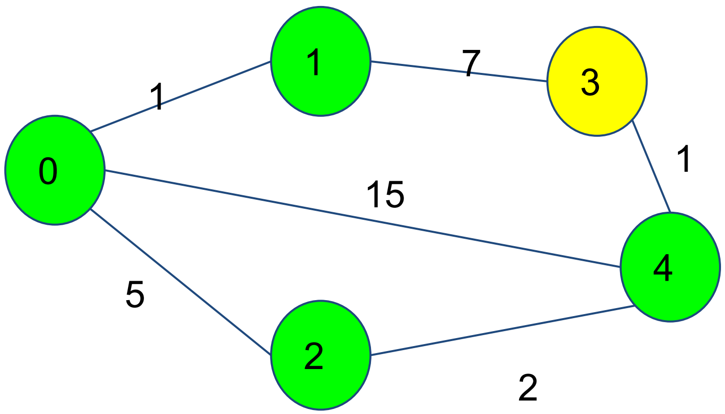


PQ = { {4, 7}, {3, 8}, {4, 15} }

Dist = { 0, 1, 5, 8, 7}

0	1	5	INF	15
1	0	INF	7	INF
5	INF	0	INF	2
INF	7	INF	0	1
15	INF	2	1	0

Grafos Ponderados

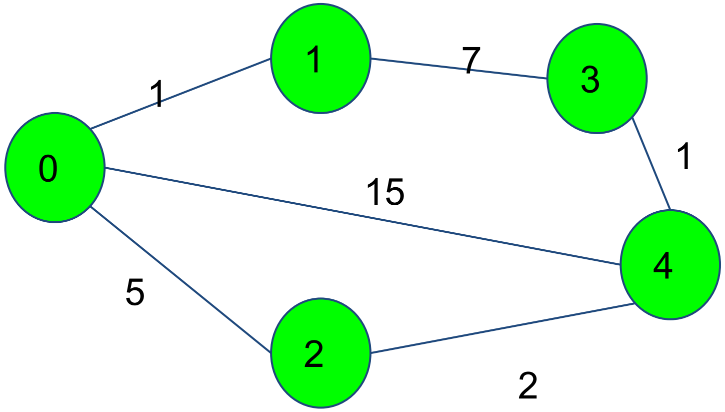


PQ = { {3, 8}, {4, 15} }

Dist = { 0, 1, 5, 8, 7}

0	1	5	INF	15
1	0	INF	7	INF
5	INF	0	INF	2
INF	7	INF	0	1
15	INF	2	1	0

Grafos Ponderados



PQ = { {4, 15} }

Dist = { 0, 1, 5, 8, 7}

0	1	5	INF	15
1	0	INF	7	INF
5	INF	0	INF	2
INF	7	INF	0	1
15	INF	2	1	0

Semana que viene...

- Semana que viene:
  - Concurso de programación práctico
  - LAB III, Aula 106 y 108
  - Entrarán como posibles problemas todo lo que hemos visto hasta ahora
  - Clasificatorio Ada-Byron para después de semana santa

¡Hasta la próxima semana!

Ante cualquier duda sobre el curso o sobre los problemas podéis escribirnos (preferiblemente copia a los tres).
También podéis utilizar el slack del curso:

https://urjc-cp.slack.com

David Morán (ddavidmorang@gmail.com)
Juan Quintana (juandavid.quintana@urjc.es)
Sergio Pérez (sergioperezp1995@gmail.com)