Algoritmo de Floyd-Warshall

Clase 24

IIC 2133 - Sección 2

Prof. Mario Droguett

Sumario

Rutas más cortas

Ya sabemos atacar el problema de rutas más cortas (baratas) desde una fuente

- Si los costos son no negativos: Dijkstra
- Si hay costos negativos: Bellman-Ford cuando no hay ciclos negativos

¿Qué hacemos si nos interesan las distancias más cortas entre **todos** los pares de vértices?

Rutas más cortas entre todos los nodos

Distancia en Km.	Antofagasta	Arica	Concepcion	Copiapó	Iquique	La Serena	Osorno	Puerto Montt	Santiago
Angol	1945	2630	190	1380	2420	1045	405	505	575
Antofagasta	0	700	1880	565	490	900	2315	2415	1370
Arica	700	0	2565	1245	300	1580	2995	3,100	2050
Calama	210	405	2075	755	385	1090	2505	2610	1560
Cartagena	1430	2115	585	865	1905	530	1020	1120	110
Cauquenes	1745	2445	155	1200	2240	865	660	765	395
Concepción	1880	2565	0	1320	2355	985	560	660	515
Constitucion	1735	2415	325	1170	2205	835	755	845	365
Copiapó	565	1245	1320	0	1040	335	1750	1850	805
Coquimbo	910	1590	975	345	1385	10	1405	1510	460
Curicó	1560	2245	320	995	2035	665	755	855	195
Chillán	1770	2455	110	1205	2245	875	545	645	405

Rutas más cortas entre todos los nodos

Si el grafo no tiene costos negativos

- Llamamos a Dijkstra desde todos los nodos
- Esto tiene complejidad $\mathcal{O}(VE \log(V))$

Si el grafo tiene costos negativos

- Llamamos a Bellman-Ford desde todos los nodos
- Esto tiene complejidad $\mathcal{O}(V^2 E)$
- Además, si el grafo es denso, $|E| \in \mathcal{O}(V^2)$ y el algoritmo sería $\mathcal{O}(V^4)$

¿Podemos hacerlo mejor?

Cambio en la forma de representación de G

Hasta ahora hemos usado adyacencias en la forma $\alpha[u]$

- Listas de adyacencia para cada vértices
- Incluyen los costos

Utilizaremos una representación de aristas mediante **matriz de adyacencia**

- Matriz cuadrada de $n \times n$, para $n = |V|$
- Incluye los pesos

La gracia de esta representación es que podemos aprovechar operaciones matriciales

Cambio en la forma de representación de G

Dado G, supondremos que los nodos están etiquetados como $V = \{1, \ldots, n\}$

Definimos la matriz de adyacencia según $W = (w_{ij})_{n \times n}$ con entradas

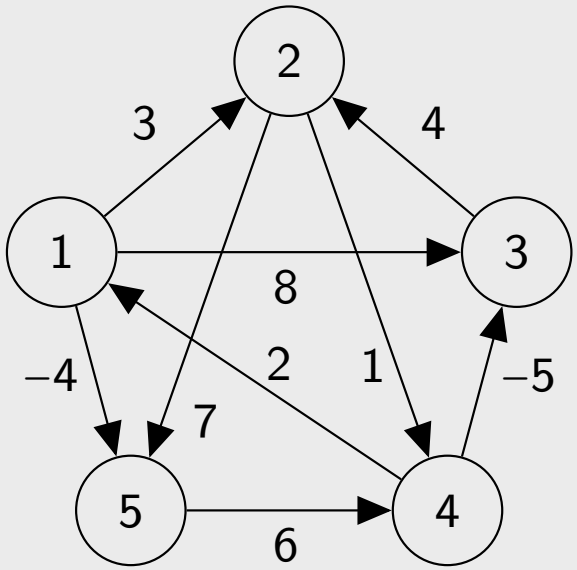
$$w_{ij} = \begin{cases} 0 & \text{si } i = j, \\ cost(i,j) & \text{si } i \neq j \wedge (i,j) \in E, \\ \infty & \text{si } i \neq j \wedge (i,j) \notin E, \end{cases}$$

para cada $1 \leq i, j \leq n$

La gracia de esta representación es que podemos aprovechar el acceso a las celdas de la matriz

Representación matricial de grafos con costos

Ejemplo



$$W = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Desde ahora, supondremos que G no tiene ciclos de costo negativo

Sumario

Nodos intermedios

Definición

Dado un camino dirigido simple p tal que $p = v_0, v_1, \ldots, v_k$, diremos que v es **intermedio** en p si $v \neq v_0$ y $v \neq v_k$. Es decir, $v \in \{v_1, \ldots, v_{k-1}\}$.

Recordemos que asumimos que $V = \{1, \ldots, n\}$

- Consideremos el conjunto $\{1, \ldots, k\} \subseteq V$, para algún k fijo
- Para $i, j \in V$, sea $\mathcal{P}$ tal que

$$\mathcal{P} := \{c \mid c \text{ es camino simple dirigido desde } i \text{ hasta } j \text{ y todo nodo intermedio está en } \{1, \ldots, k\}\}$$

- Sea p el camino más barato de $\mathcal{P}$

¿k está en p?

Nodos intermedios

Si k no es un nodo de p

- Todos los nodos intermedios de p están en $\{1 \ldots, k-1\}$
- Luego, una ruta más corta de i a j con intermedios en $\{1, \ldots, k-1\}$ también es ruta más corta de i a j con intermedios en $\{1, \ldots, k\}$

Si k es un nodo de p

- Podemos dividir p en dos subcaminos
- Sean p_1, p_2 tales que $i \rightsquigarrow_{p_1} k$ y $k \rightsquigarrow_{p_2} j$
- Como p es de costo mínimo, p_1 es la ruta más corta de i a k con intermedios en $\{1, \ldots, k-1\}$
- Como p es de costo mínimo, p_2 es la ruta más corta de k a j con intermedios en $\{1, \ldots, k-1\}$

Usaremos programación dinámica

Recurrencia y nodos intermedios

Definimos

$$d_{ij}^{(k)} := \text{costo de ruta más corta de } i \text{ hasta } j \text{ tal que todos los nodos intermedios están en } \{1, \ldots, k\}$$

Observemos que

- Para $k = 0$, no hay nodos intermedios posibles
- Esto obliga a considerar rutas sin intermedios: i.e. la ruta directa (i, j)
- Con esto, $d_{ij}^{(0)} = w_{ij}$
- Notemos que si no hay arista directa, $d_{ij}^{(0)} = \infty$ por construcción de W

¿Cómo definimos $d_{ij}^{(k)}$ recursivamente?

Recurrencia y nodos intermedios

Con las observaciones respecto a si k pertenece o no a la ruta óptima,

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{si } k = 0 \\ \min\left(d_{ij}^{(k-1)}, \quad d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{si } k \geq 1 \end{cases}$$

Por conveniencia, llamamos $D^{(k)} = (d_{ij}^{(k)})$ a la matriz con las distancias óptimas para cada $0 \leq k \leq n$

Recordemos

- k especifica el conjunto de nodos intermedios permitidos
- si $k = n$, permitimos cualquier nodo intermedio

En la matriz $D^{(n)}$ tenemos los valores óptimos $d_{ij}^{(n)} = \delta(i, j)$

Algoritmo de Floyd-Warshall

Implementando esta estrategia de programación dinámica, obtenemos el siguiente algoritmo de **todas las rutas más cortas**

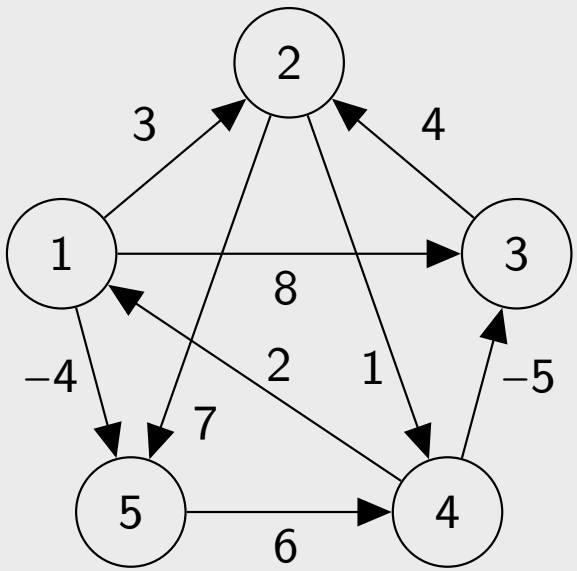
```
FloydWarshall(W):
1    D^(0) ← W
2    for k = 1...n :
3        D^(k) ← (d_ij^(k)) una nueva matriz
4        for i = 1...n :
5            for j = 1...n :
6                d_ij^(k) ← min( d_ij^(k-1),  d_ik^(k-1) + d_kj^(k-1) )
7    return D^(n)
```

Este es un ejemplo de algoritmo **bottom-up**:
se construye de manera incremental hasta llegar a la solución esperada

Algoritmo de Floyd-Warshall

Ejercicio

Obtenga las rutas más cortas entre todos los pares de vértices del siguiente grafo con costos negativos

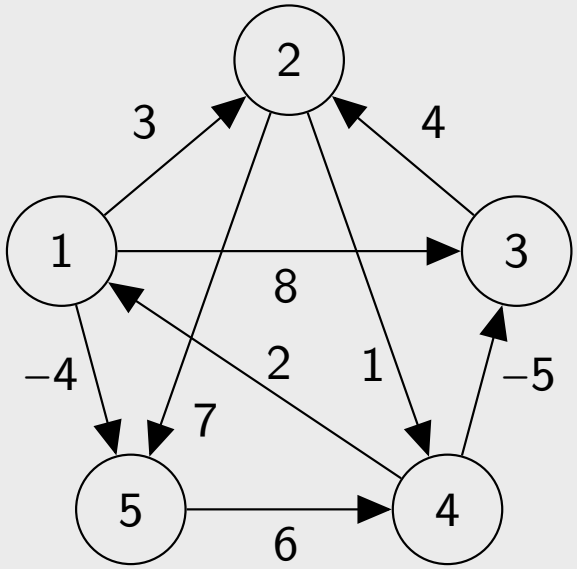


$$W = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Algoritmo de Floyd-Warshall

Ejercicio

Primero mantenemos los costos cuando los nodos intermedios están en $\varnothing$



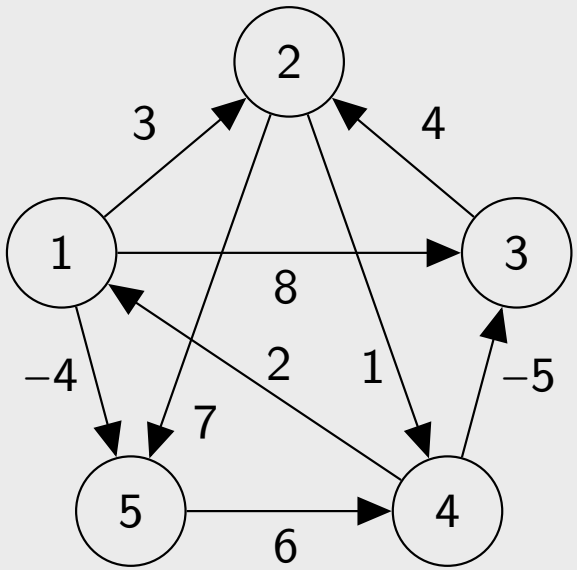
$$D^{(0)} = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

La primera matriz corresponde a los costos de ir directamente (sin nodos intermedios)

Algoritmo de Floyd-Warshall

Ejercicio

Ahora, los intermedios pueden estar en $\{1\}$



$$D^{(0)} = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

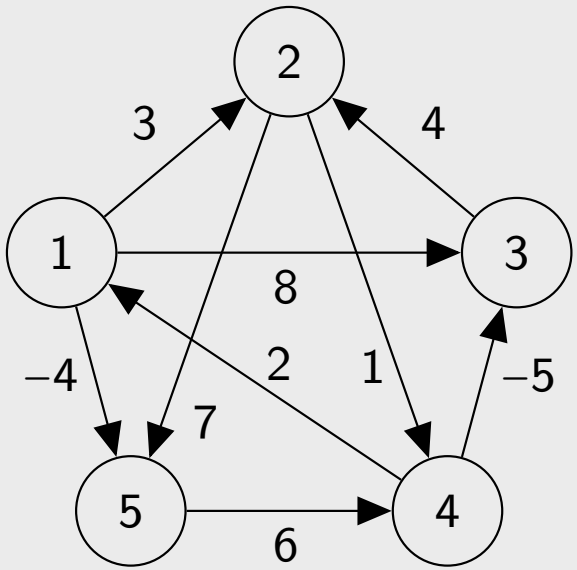
$$D^{(1)} = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Notemos que se actualizan solo aquellos trayectos
que mejoran incluyendo al 1 como intermedio

Algoritmo de Floyd-Warshall

Ejercicio

Ahora, los intermedios pueden estar en $\{1, 2\}$



$$D^{(1)} = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

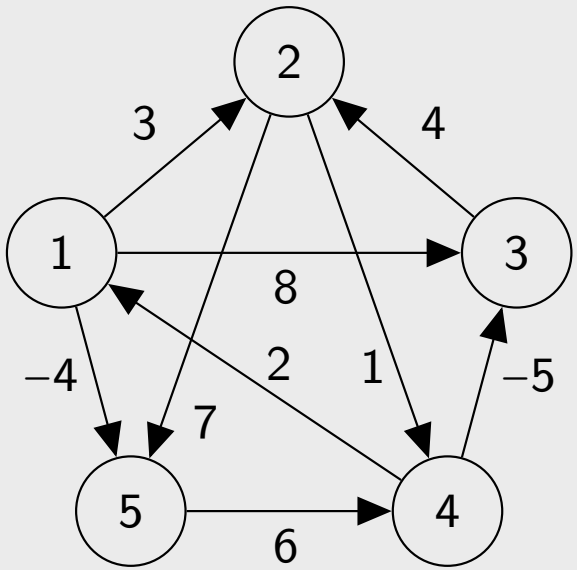
$$D^{(2)} = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Ahora **también** permitimos usar el 2 como nodo intermedio

Algoritmo de Floyd-Warshall

Ejercicio

Ahora, los intermedios pueden estar en $\{1, 2, 3\}$



$$D^{(2)} = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

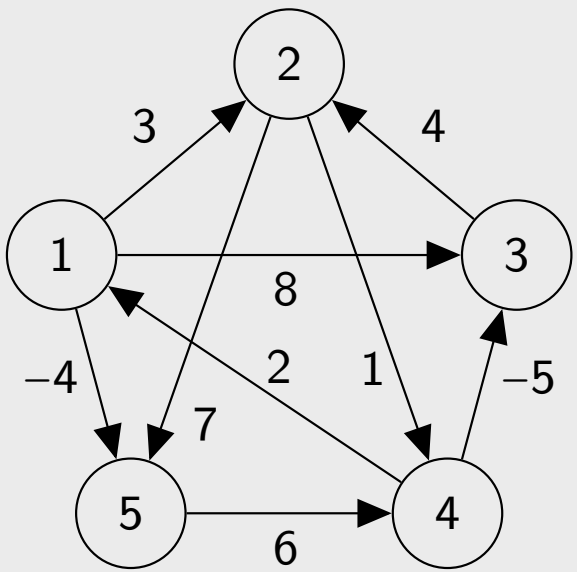
$$D^{(3)} = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Notemos que al permitir el 3 como intermedio,
mejoramos el camino existente $4 \to 1 \to 2$ con $4 \to 3 \to 2$

Algoritmo de Floyd-Warshall

Ejercicio

Ahora, los intermedios pueden estar en $\{1, 2, 3, 4\}$



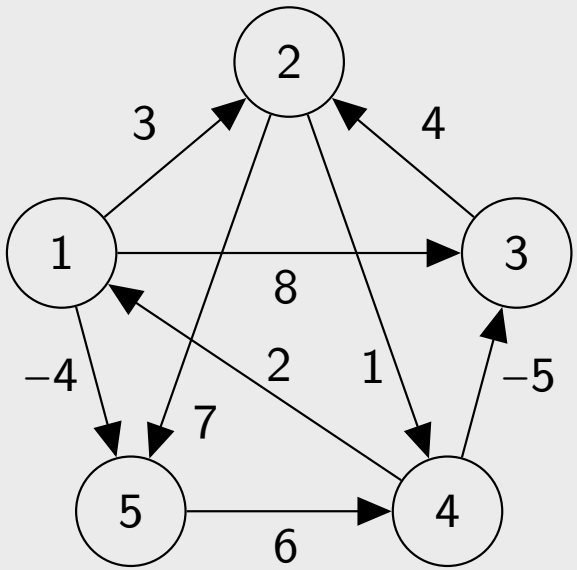
$$D^{(3)} = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

$$D^{(4)} = \begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

Algoritmo de Floyd-Warshall

Ejercicio

Ahora, los intermedios pueden estar en $\{1, 2, 3, 4, 5\}$, i.e. se permite cualquier nodo

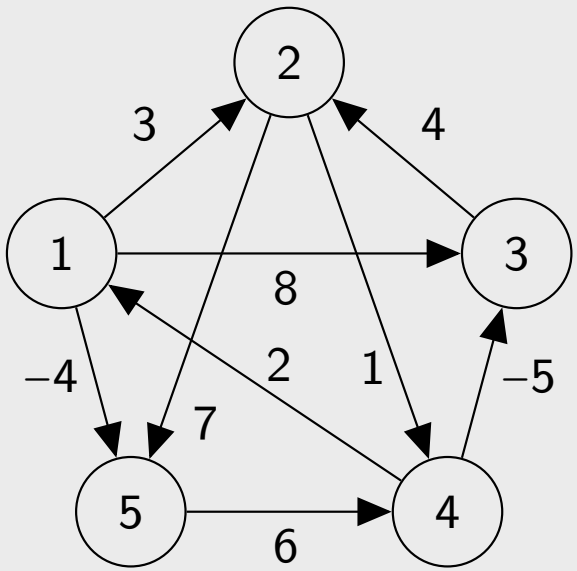


$$D^{(4)} = \begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

$$D^{(5)} = \begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

Algoritmo de Floyd-Warshall

Ejercicio



$$D^{(5)} = \begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

En $D^{(5)}$ tenemos los costos óptimos, sin importar el largo de esos caminos.

Por ejemplo

$$\delta(4,5) = d_{45}^{(5)} = -2$$

¿Qué complejidad tiene este algoritmo?

Complejidad

`FloydWarshall`(W):

1 $D^{(0)} \leftarrow W$

2 **for** $k = 1 \ldots n$ **:**

3 $D^{(k)} \leftarrow (d_{ij}^{(k)})$ vacía

4 **for** $i = 1 \ldots n$ **:**

5 **for** $j = 1 \ldots n$ **:**

6 $d_{ij}^{(k)} \leftarrow \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$

7 **return** $D^{(n)}$

Calculo incremental

- $\mathcal{O}(V)$ iteraciones para los k posibles
- Combinación de pares i, j posibles en $\mathcal{O}(V^2)$
- Acceso $\mathcal{O}(1)$ a la matriz y comparación $\mathcal{O}(1)$

El algoritmo de Floyd-Warshall es $\mathcal{O}(V^3)$

Obtención de los caminos

Además de los costos, nos interesa determinar los caminos óptimos

- Podemos usar los costos calculados y deducir *hacia atrás*
- O podemos modificar el algoritmo base para ir almacenando los ancestros

Para esta última estrategia, para cada k usaremos una matriz $\Pi^{(k)} = (\pi_{ij}^{(k)})$ tal que

$$\pi_{ij}^{(k)} := \quad \text{ancestro inmediato de } j \text{ en una ruta más corta desde } i \text{ hasta } j \text{ con intermedios en } \{1, \ldots, k\}$$

¿Cómo se define el ancestro recursivamente?

Obtención de los caminos

Para $k = 0$ tenemos caminos sin intermedios, por lo que el ancestro de j es

$$\pi_{ij}^{(0)} = \begin{cases} \varnothing & \text{si } i = j \vee (i,j) \notin E \\ i & \text{si } i \neq j \wedge (i,j) \in E \end{cases}$$

Para $k \geq 1$, nos preguntamos nuevamente si k está en un camino más corto

- Si está, $i \rightsquigarrow_{p_1} k \rightsquigarrow_{p_2} j$ y por lo tanto el ancestro de j en p es el mismo que en p_2
- Si no está, entonces el ancestro de j con intermedios en $\{1, \ldots, k\}$ es el mismo que con intermedios en $\{1, \ldots, k-1\}$

Es decir,

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{si } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ \pi_{kj}^{(k-1)} & \text{si } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

Con esto podemos extender el algoritmo y almacenar los antecesores

Obtención de los caminos

```
   FloydWarshallPaths(W):
1      D^(0) ← W;   Π^(0) ← (π_ij^(0)) vacía
2      for i = 1...n :
3          for j = 1...n :
4              if i ≠ j ∧ d_ij^(0) < ∞ :
5                  π_ij^(0) ← i
6      for k = 1...n :
7          D^(k) ← (d_ij^(k)) vacía;   Π^(k) ← (π_ij^(k)) vacía
8          for i = 1...n :
9              for j = 1...n :
10                 if d_ij^(k-1) ≤ d_ik^(k-1) + d_kj^(k-1) :
11                     d_ij^(k) ← d_ij^(k-1);   π_ij^(k) ← π_ij^(k-1)
12                 else:
13                     d_ij^(k) ← d_ik^(k-1) + d_kj^(k-1);   π_ij^(k) ← π_kj^(k-1)
14     return D^(n), Π^(n)
```

Obtención de los caminos

El algoritmo resultante mantiene la complejidad

- Es $\mathcal{O}(V^3)$
- Además entrega la matriz $\Pi^{(n)}$ que permite deducir los caminos explícitos

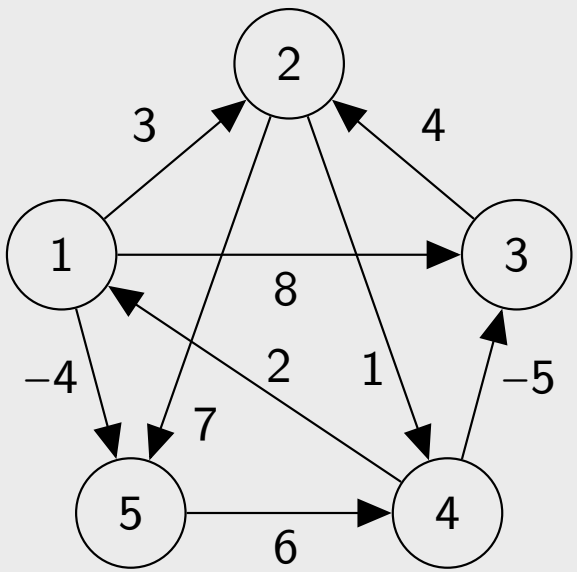
Para nodos i, j,

- Si $\pi_{ij}^{(n)} = t$, t es ancestro de j en la ruta más corta desde i
- Luego, el ancestro de t en ese camino es $\pi_{it}^{(n)}$
- ...

Obtención de los caminos

Ejemplo

La ejecución de Floyd-Warshall extendido para el ejemplo entrega



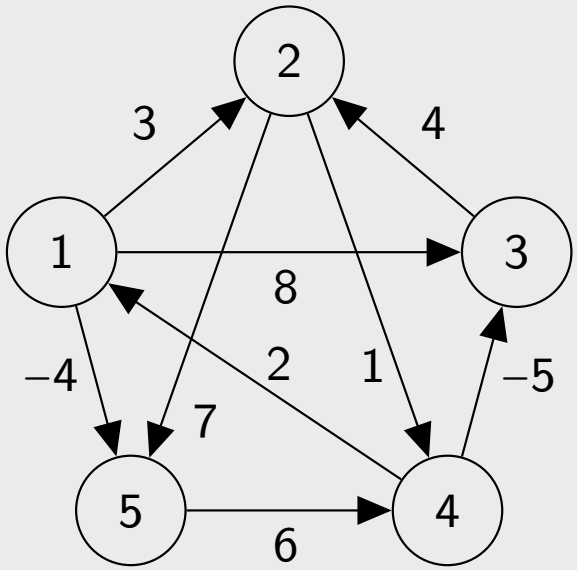
$$D^{(5)} = \begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

$$\Pi^{(5)} = \begin{bmatrix} \varnothing & 3 & 4 & 5 & 1 \\ 4 & \varnothing & 4 & 2 & 1 \\ 4 & 3 & \varnothing & 2 & 1 \\ 4 & 3 & 4 & \varnothing & 1 \\ 4 & 3 & 4 & 5 & \varnothing \end{bmatrix}$$

Obtención de los caminos

Ejemplo

La ejecución de Floyd-Warshall extendido para el ejemplo entrega



$$\Pi^{(5)} = \begin{bmatrix} \varnothing & 3 & 4 & 5 & 1 \\ 4 & \varnothing & 4 & 2 & 1 \\ 4 & 3 & \varnothing & 2 & 1 \\ 4 & 3 & 4 & \varnothing & 1 \\ 4 & 3 & 4 & 5 & \varnothing \end{bmatrix}$$

El camino más corto desde 2 hasta 5 está dado por

$$\pi_{25}^{(5)} = 1 \quad \pi_{21}^{(5)} = 4 \quad \pi_{24}^{(5)} = 2$$

Es decir, $2 \to 4 \to 1 \to 5$