Caminos más cortos a todos los pares de nodos

- Este problema se puede resolver invocando n veces para pesos no negativos a Dijkstra
  - Cola por prioridad basada en Vector $T(n)=O(N^3)$,
  - Montículo binario $T(n)= O(N A \lg N)$,
  - Montículo Fibonacci $T(n)= O(N^2 \lg N + N A))$
- si hay pesos negativos a Bellman-Ford $T(n)= O(N^2 A)$ y para digrafos densos $T(n)= O(N^4)$
- Para mejorar esos T(n) se tratará el problema desde otro punto de vista
  - Usando multiplicación de matrices

Caminos más cortos a todos los pares de nodos

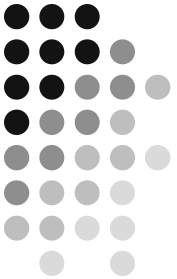
- G se implanta con matriz de adyacencia, en vez de listas de adyacencia.
- La matriz de pesos W (N x N) representa los pesos de cada arco de G.

$$w_{ij} = \begin{cases} 0 & \text{si } i = j \\ \text{peso de } (i, j) & \text{si } i \neq j \text{ y } (i, j) \in A \\ \infty & \text{si } i \neq j \text{ y } (i, j) \notin A \end{cases}$$

- Se permiten pesos negativos, pero **no** ciclos de peso negativo.

UNIVERSIDAD DE LOS ANDES

Caminos mínimos a todos los pares de nodos



- La salida es una matriz D (N x N) donde cada dij contiene el peso del C+C desde **i** hasta **j**, esto es dij = δ(i, j) al finalizar el algoritmo.
- Se necesita calcular la matriz predecesora o matriz con los nodos padres Π = P = padre(i, j), donde
  - padre(i, j) = Nulo si no hay camino de **i** a **j** ó si i = j,
  - de lo contrario es el predecesor de **j** en el C+C desde **i**.

Caminos mínimos a todos los pares de nodos

- Para cada nodo $i \in N$ se define el subgrafo predecesor de G para i como $G_{\pi i} = (N_{\pi i}, A_{\pi i})$, donde

$N_{\pi i} = \{j \in N : \text{padre}(i, j) \neq \text{Nulo}\} \cup \{ i \}$ y

$A_{\pi i} = \{ (\text{padre}(i, j), j) : j \in N_{\pi i}$ y $\text{padre}(i, j) \neq \text{Nulo}\}$.

Caminos mínimos a todos los pares de nodos

- **Convenciones**:
  - Las matrices se nombran con letras mayúsculas,
  - sus elementos con minúsculas,
  - la iteración de una matriz con superíndices entre paréntesis ($D^{(m)} = \left(d_{ij}^{(m)}\right)$) y
  - que el número de filas de una matriz está en A.filas().
- El procedimiento que despliega el C+C desde **i** hasta **j** es:

Caminos mínimos a todos los pares de nodos

26/11/98

despliegueC+CPares(Arreglo(n x n)De [Real]: P, Entero: i, j)

{pre: $n > 0 \wedge 1 \leq i, j \leq n$ } {pos: $n > 0$ }

```
1   Si ( i = j ) entonces
      despliegue   j
    sino  Si (P(i, j) = 0) entonces
            despliegue "No hay camino de ", i, " a ", j
          sino
             despliegueC+CPares(P, i, P(i, j))
             despliegue  j
          fsi
    fsi
2   regrese
```

-i, j: Entero. Indican nodos del grafo.

-p: Arreglo(n x n)De [Real]. Matriz predecesora de G.

Solución recursiva de abajo hacia arriba

- Matriz de adyacencia con sus pesos $W = (w_{i\,j})$
- Suponga que el C+C de **i** a **j** llamado **p** contiene al menos **m** arcos.
- Si no hay ciclos con peso negativo, entonces **m** es finito.
- Si $i = j$, entonces **p** tiene peso = 0 y no tiene arcos
- Si $i \neq j$, entonces **p** se puede descomponer en el camino **p'** de (i **p'** k) $\rightarrow$ j, donde **p'** contiene a lo sumo m - 1 arcos y el arco de **k** a **j**.

Solución recursiva de abajo hacia arriba

- Sea $d^{(m)}_{ij}$ el peso mínimo de cualquier camino de **i** a **j** que contiene **m** arcos.
- Cuando m = 0 se tiene $d_{ij}^{(0)} = \begin{cases} 0 & \text{si } i = j \\ \infty & \text{si } i \neq j \end{cases}$ para $m > 0$

$$d_{ij}^{(m)} = \min_{1 \le k \le n} \{ d_{ik}^{(m-1)} + w_{kj} \}$$

- Los pesos del C+C actual están dados por

$$\delta(i, j) = d_{ij}^{(n-1)} = d_{ij}^{(n)} = d_{ij}^{(n+1)} = \cdots$$

Caminos mínimos a todos los pares de nodos

- Teniendo como entrada la matriz W se calculan las matrices $D^{(1)}$, $D^{(2)}$, ..., $D^{(n-1)}$, donde m = 1, 2, ..., (n-1), la matriz $D^{(n-1)}$ contiene los pesos del C+C actual y $D^{(1)}$ = W.

26/11/98 extenderC+CPares(Arreglo(n x n)De [Real]: D, W): Arreglo(n x n)De [Real] {pre: n > 0 } {pos: n > 0 }		
1 2 3	n = D.filas() [[D'(i, j) = ∞ [D'(i, j) = min(D'(i, j), D(i, k) + W(k, j))] k = 1, n] j = 1, n] i = 1, n regrese D'	**-i, j, k:** Entero. Indican nodos del grafo y son los subíndices de los lazos. **-D':** Arreglo(n x n)De [Real]. Matriz con los pesos de los C+C actuales.

$T(n) = \Theta(n^3)$

Caminos mínimos a todos los pares de nodos

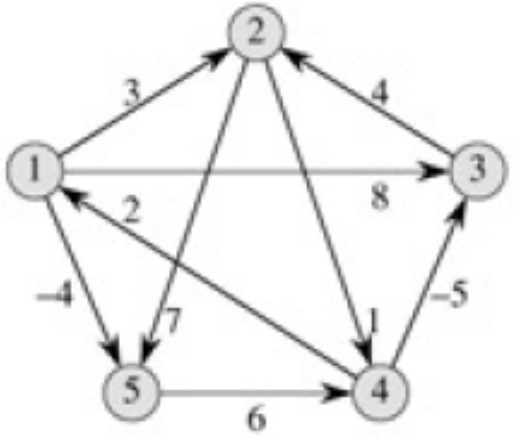
- El cálculo de todos los C+C entre los pares de nodos de G se realiza con

26/11/98 lentoC+CPares(Arreglo(n x n)De [Real]: W): Arreglo(n x n)De [Real] {pre: n > 0 } {pos: n > 0 }		
1 2 3 4	n = W.filas() $D^{(1)}$ = W [$D^{(m)}$ = extenderC+CPares($D^{(m-1)}$, W)] m = 2, n - 1 regrese $D^{(n-1)}$	**-n, m:** Entero. Número de nodos del grafo y subíndice del lazo, respectivamente. **-$D^{(m)}$:** Arreglo(n x n)De [Real]. Matriz con los pesos de los C+C actuales.

$$T(n) = \Theta(n^4)$$

UNIVERSIDAD DE LOS ANDES

Ejemplo



lentoC+Cpares
(Cormen, 2001)

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

$$L^{(3)} = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad L^{(4)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Caminos mínimos a todos los pares de nodos

- **Mejor tiempo**: Solo interesa tener la matriz final y **no** las intermedias, por lo cual se aplica la técnica de los cuadrados sucesivos.

$D(1) = W$

$D(2) = W^2 = W \,.\, W$

$D(4) = W^4 = W^2 \,.\, W^2$

$D(8) = W^8 = W^4 \,.\, W^4$

$$D^{(2^{\lceil \lg(n-1) \rceil})} = W^{2^{\lceil \lg)n-1) \rceil}} = W^{2^{\lceil \lg(n-1) \rceil - 1}} . W^{2^{\lceil \lg(n-1) \rceil - 1}}$$

Caminos mínimos a todos los pares de nodos

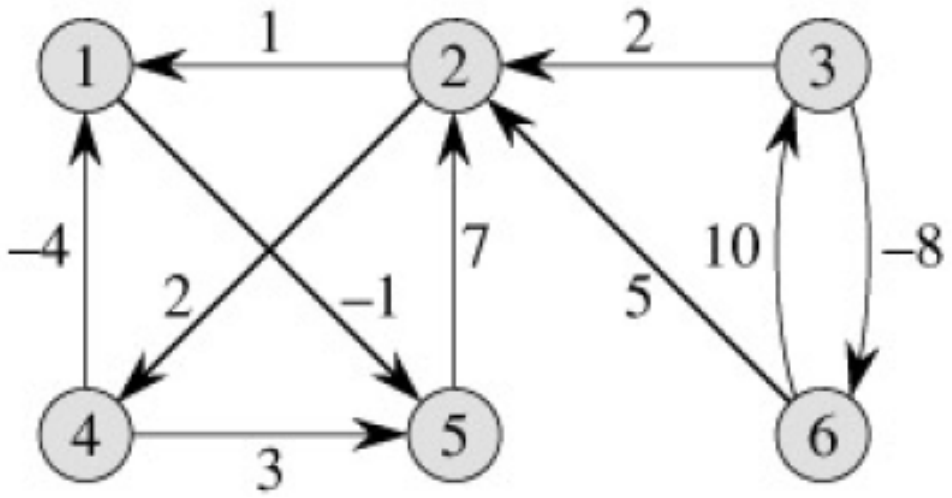
26/11/98 rapidoC+CPares(Arreglo(n x n)De [Real]: W): Arreglo(n x n)De [Real] {pre: n > 0 } {pos: n > 0 }		
1	n = W.filas()	**-n, m:** Entero. Número de nodos del grafo y subíndice del lazo, respectivamente.
2	$D^{(1)} = W$	
3	(n - 1 > m) [$D^{(2m)}$ = extenderC+CPares($D^{(m)}$, $D^{(m)}$) m = 2n]	**-$D^{(m)}$:** Arreglo(n x n)De [Real]. Matriz con los pesos de los C+C actuales.
4	regrese $D^{(m)}$	

$$T(n) = \Theta(n^3 \lg n)$$

UNIVERSIDAD DE LOS ANDES

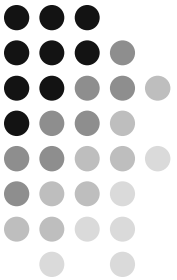
Ejercicio

- Pruebe el algoritmo rapidoC+Cpares para el grafo dado a continuación



UNIVERSIDAD DE LOS ANDES

Algoritmo de Floyd-Warshall



- Encuentra la solución al problema con otra formulación basada en programación dinámica
- Un nodo intermedio de un camino simple $\mathbf{p} = \langle n_1, n_2, ..., n_h \rangle$ es cualquier nodo de **p** diferente de n_1 y n_h
- Si **k** es un nodo intermedio de **p**, entonces **p** puede dividirse en p' y p'', donde
  - p' es el C+C de **i** a **k** y
  - p'' el C+C de **k** a **j**.

  $\mathbf{i} \rightarrow p' \rightarrow \mathbf{k} \rightarrow p'' \rightarrow \mathbf{j}$

Algoritmo de Floyd-Warshall

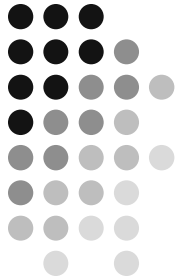
- Sea $d^{(k)}_{ij}$ el peso del C+C de **i** a **j** con todos los nodos intermedios en {1, 2, ..., k}.
- Si k = 0, no hay nodos intermedios por lo que $d^{(0)}_{ij} = w(i, j)$.
- Se define la recursión

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

- La matriz $D^{(n)}$ contiene los pesos de los C+C entre todos los pares de nodos de G, pero **no** contiene los C+C.

UNIVERSIDAD DE LOS ANDES

Algoritmo de Floyd-Warshall



26/11/98

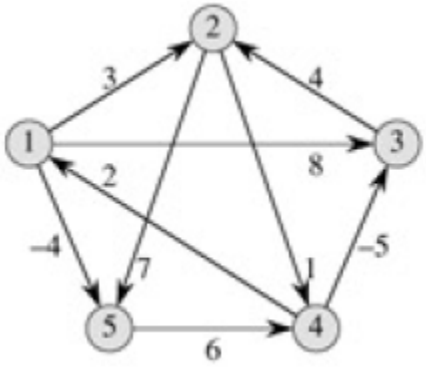
FloydWarshall(Arreglo(n x n)De [Real]: W): Arreglo(n x n)De [Real]

{pre: n > 0 } {pos: n > 0 }

1	n = W.filas()	**-i, j, k:** Entero. Indican nodos del grafo y son los subíndices de los lazos.
2	$D^{(0)} = W$	**-$D^{(n)}$:** Arreglo(n x n)De [Real]. Matriz con los pesos de los C+C actuales.
3	[[[$D^{(k)}(i, j) = \min(D^{(k-1)}(i, j), D^{(k-1)}(i, k) + D^{(k-1)}(k, j))$] j = 1, n] i = 1, n] k = 1, n	
4	regrese $D^{(n)}$	

$$T(n) = \Theta(n^3)$$

Ejemplo del algoritmo de Floyd-Warshall



(Cormen, 2001)

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

UNIVERSIDAD DE LOS ANDES

Continuación

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

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

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

(Cormen, 2001)

Aplicación del algoritmo de Floyd-Warshall

- Cierre transitivo
  - El cierre transitivo de G se define con un grafo $G^* = (N, A^*)$ donde $A^* = \{(i, j) : \text{hay un camino de } i \text{ a } j \text{ en } G\}$.

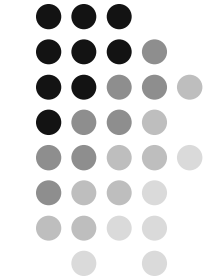
1. Una forma de calcularlo es utilizar el algoritmo de Floyd-Warshall colocando todos los pesos en 1.

Si hay un camino de i a j, entonces $d_{ij} < n$, sino $d_{ij} = \infty$.

2. Otra forma es utilizar el mismo algoritmo, pero sustituyendo **min** y **+** por $\vee$ y $\wedge$**,** y tratar los 1 y 0 de la matriz como valores lógicos.

La matriz resultante es la clausura transitiva de G.

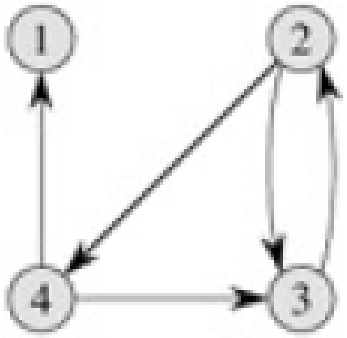
Algoritmo para calcular el cierre transitivo



26/11/98		
	cierreTransitivo( ): Arreglo(n x n)De [Real]	
{pre: n > 0 }		{pos: n > 0 }
1	[[Si (i = j $\vee$ (i, j) $\in$ A) entonces $T^{(0)}(i, j) = 1$ sino $T^{(0)}(i, j) = 0$ fsi] j = 1, n] i = 1, n [[[$T^{(k)}(i, j) = T^{(k-1)}(i, j) \vee (T^{(k-1)}(i, k) \wedge T^{(k-1)}(k, j))$] j = 1, n] i = 1, n] k = 1, n	**-i, j, k:** Entero. Indican nodos del grafo y son los subíndices de los lazos. **-$D^{(n)}$:** Arreglo(n x n)De [Real]. Matriz con los pesos de los C+C actuales.
2	regrese $T^{(n)}$	

$$T(n) = \Theta(n^3)$$

Ejemplo del algoritmo de cierre transitivo



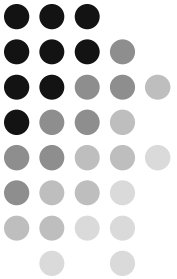
$$T^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad T^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad T^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$T^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad T^{(4)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

(Cormen, 2001)

UNIVERSIDAD DE LOS ANDES

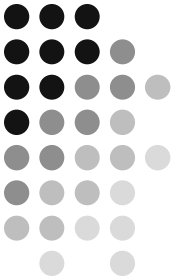
Algoritmo de Johnson para grafos esparcidos



- Es mejor asintóticamente que hacer potencias de matrices o que el algoritmo de Floyd-Warshall para grafos esparcidos.
- Utiliza como subprogramas el algoritmo de Dijkstra y el de Bellman-Ford.
- Utiliza la técnica de reasignar pesos, que consiste en:
  - Si todos los pesos de los arcos de G son positivos, entonces se usa Dijkstra con montículos de Fibonacci y se corre para cada nodo de G
  - Si G tiene pesos negativos, se calcula un nuevo conjunto de pesos no negativos w' que permitan usar el mismo método. Este preprocesamiento de G para calcular w' se puede hacer en O(N A)

UNIVERSIDAD DE LOS ANDES

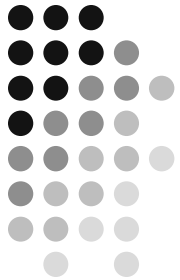
Algoritmo de Johnson para grafos esparcidos



- Los nuevos pesos positivos calculados deben cumplir:
  1. $\forall$ par de nodos u, v $\in$ N, el C+C de **u** a **v** usando **w** es también el C+C de **u** a **v** usando w'
  2. $\forall$ arcos (u, v), el nuevo peso w'(u, v) es no negativo
- *Lema 18: La reasignación de pesos no cambia los C+C*
  - Dado un digrafo etiquetado G con w : A$\rightarrow\Re$ y sea h : N$\rightarrow\Re$, para cada arco (u, v) $\in$ A se define w'(u, v) = w (u, v) + h(u) - h(v).
  - Sea p = $\langle n_0, n_1, ..., n_k\rangle$ el camino de n_0 a n_k, entonces w(p) = $\delta(n_0, n_k)$ si y solo si w'(p) = δ' (n_0, n_k).
  - G tiene ciclos con peso negativos usando **w** si y solo si G los tiene también usando w'.

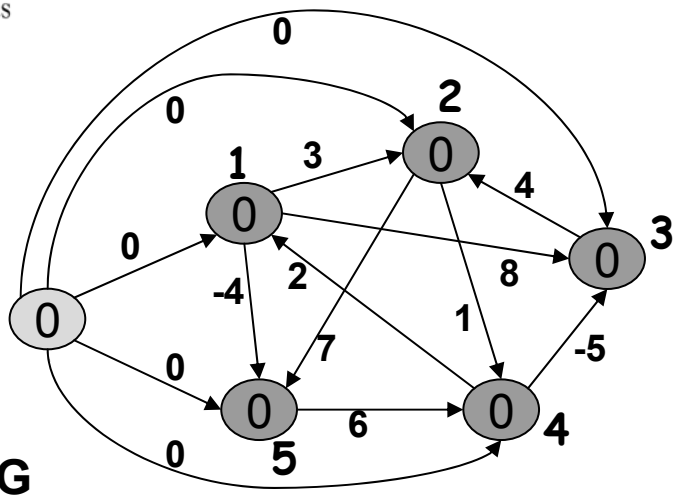
UNIVERSIDAD DE LOS ANDES

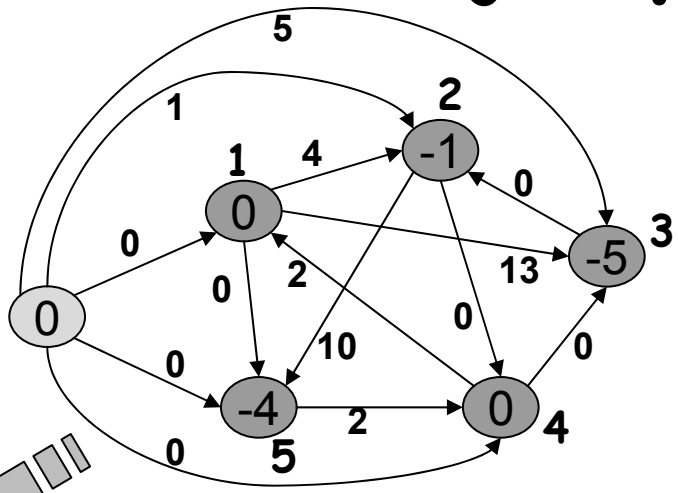
Algoritmo de Johnson



- Se calcula G' con los pesos reasignados de G según $w'(u, v) = w(u, v) + h(u) - h(v) \geq 0$.
- Se asume que las listas de adyacencia de G guardan los pesos por cada arco.
- El algoritmo de Johnson regresa
  - la matriz (n x n) D con los pesos de los C+C o
  - despliega la imposibilidad de su cálculo por tener ciclos de peso negativo
- Se asumen los nodos numerados de 1 a n
- Para crear G', se crea un nodo ficticio 0 y un arco desde él hasta el resto de los nodos de G, con peso inicial 0.

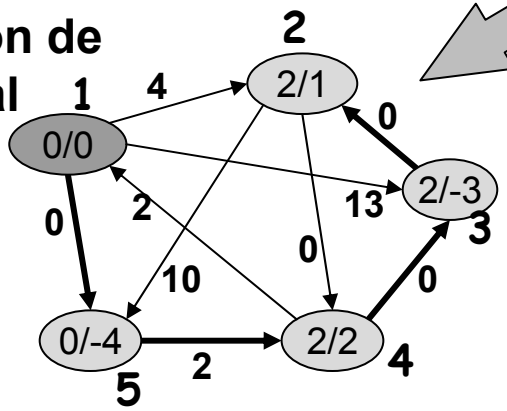
Ejemplo



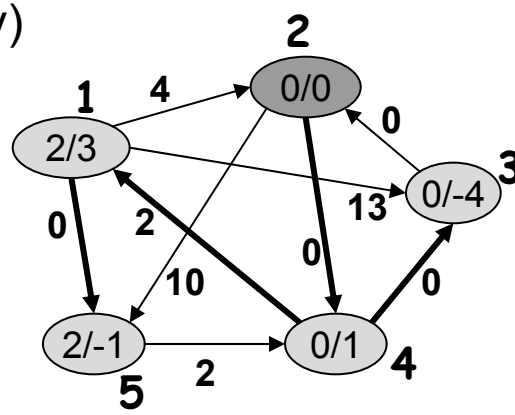


G' luego de aplicar BellmanFord que calcula los $\delta(s,v)$, de asignar h(v) y de calcular w'

Aplicación de Dijkstra al nodo 1



$\delta'(u,v)/\delta(u,v)$



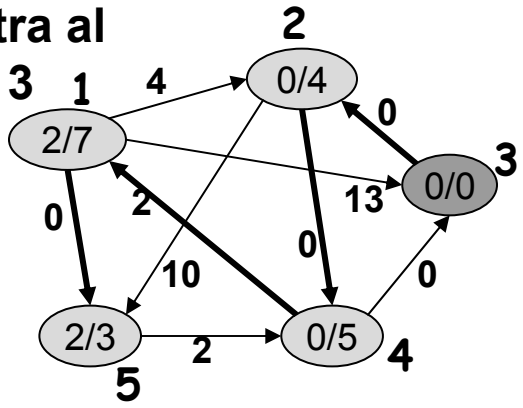
Aplicación de Dijkstra al nodo 2

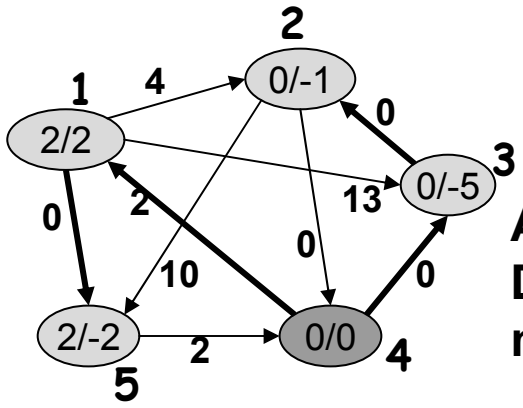
$\delta'(5)=\delta'(1)+w'(1,5)=0+0=0$

$\delta(5)=\delta(1)+w(1,5)=0+(-4)=-4$

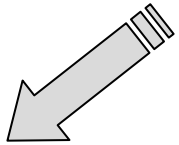
Continuación ejemplo

Aplicación de Dijkstra al nodo 3

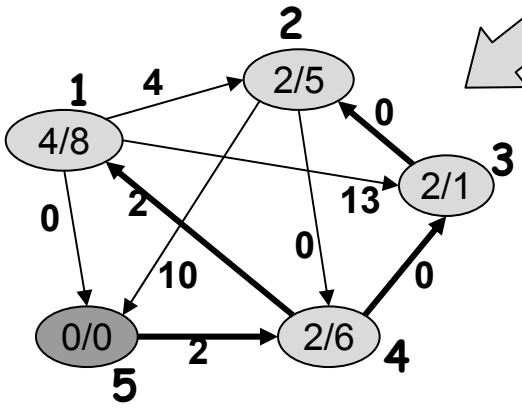




Aplicación de Dijkstra al nodo 4



Aplicación de Dijkstra al nodo 5



UNIVERSIDAD DE LOS ANDES

Algoritmo de Johnson

26/11/98

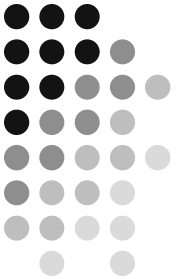
C+CJohnson():Arreglo(n x n)De [Real]

{pre: n > 0 } {pos: n > 0 }

1	Calcular G' con N' = N ⋃ { s } y A' = A ⋃ {(s, v): v∈N}	**-u, v:** Entero. Indican nodos del grafo y son los subíndices de los lazos.
2	Si (¬G'.C+CBellmanFord(s)) entonces despliegue "G contiene ciclos de peso negativo" sino [h(v) = δ(s, v) calculado por Bellman-Ford] v ∈ N [w'(u,v) = w(u,v)+h(u)-h(v)] (u, v) ∈ A [C+CDijkstra(u) para calcular δ'(u,v) [D(u, v) = δ'(u, v) + h(v) - h(u)] v ∈ N] u ∈ N fsi	**-D:** Arreglo(n x n)De [Real]. Matriz con los pesos de los C+C actuales. **-G', N', A':** Grafo auxiliar con un nodo adicional (s). **-h:** Arreglo(n)De [Real]. Contiene los valores para reasignar los pesos.
3	regrese D	

UNIVERSIDAD DE LOS ANDES

Análisis del algoritmo de Johnson



- $T(n) = O(N^2 \lg N + N A)$ si el algoritmo de Dijkstra está implantado con montículos de Fibonacci.
- $T(n) = O(N A \lg N)$ si está implantado con montículos binarios.
- Aún así es asintóticamente más rápido que el algoritmo de Floyd-Warshall.