Optimización III (525551)

Módulo III: Problema del Camino Más Corto

Pr. Julio Aracena.

Departamento de Ingeniería Matemática
Facultad de Ciencias Físicas y Matemáticas
Universidad de Concepción

Primer semestre, 2021

Problema del Camino Más Corto (PCC)

Sea $G = (V, E)$ un digrafo con función de peso en los arcos $w : E \to \mathbb{R}$. Un camino en G es una secuencia de vértices (**no necesariamente distintos**) $p : v_1, \ldots, v_r$ tal que $\forall i = 1, \ldots, r$, $(v_i, v_{i+1}) \in E$. El peso del camino p se define por:

$$w(p) := \sum_{i=1}^{r-1} w(v_i, v_{i+1}).$$

Problema del Camino Más Corto (PCC)

Sea $G = (V, E)$ un digrafo con función de peso en los arcos $w : E \to \mathbb{R}$. Un camino en G es una secuencia de vértices (**no necesariamente distintos**) $p : v_1, \ldots, v_r$ tal que $\forall i = 1, \ldots, r$, $(v_i, v_{i+1}) \in E$. El peso del camino p se define por:

$$w(p) := \sum_{i=1}^{r-1} w(v_i, v_{i+1}).$$

Para todo $u, v \in V$ se define el peso mínimo de los caminos de u a v por:

$$\delta(u, v) := \begin{cases} \text{ínf}\{w(p) : p \text{ es un camino de } u \text{ a } v\} & \text{si existe camino de } u \text{ a } v \text{ en } G, \\ +\infty & \text{en otro caso.} \end{cases}$$

Un camino p de u a v en G se dice de **peso mínimo** si $w(p) = \delta(u, v)$.

Problema del Camino Más Corto (PCC)

Sea $G = (V, E)$ un digrafo con función de peso en los arcos $w : E \to \mathbb{R}$. Un camino en G es una secuencia de vértices (**no necesariamente distintos**) $p : v_1, \ldots, v_r$ tal que $\forall i = 1, \ldots, r$, $(v_i, v_{i+1}) \in E$. El peso del camino p se define por:

$$w(p) := \sum_{i=1}^{r-1} w(v_i, v_{i+1}).$$

Para todo $u, v \in V$ se define el peso mínimo de los caminos de u a v por:

$$\delta(u, v) := \begin{cases} \text{ínf}\{w(p) : p \text{ es un camino de } u \text{ a } v\} & \text{si existe camino de } u \text{ a } v \text{ en } G, \\ +\infty & \text{en otro caso.} \end{cases}$$

Un camino p de u a v en G se dice de **peso mínimo** si $w(p) = \delta(u, v)$.

Observación: : $\delta(u, v) \in \mathbb{R} \cup \{-\infty, +\infty\}$. Luego, puede que exista camino de u a v en G pero no necesariamente un camino de peso mínimo.

Problema del Camino Más Corto (PCC)

Sea $G = (V, E)$ un digrafo con función de peso en los arcos $w : E \to \mathbb{R}$. Un camino en G es una secuencia de vértices (**no necesariamente distintos**) $p : v_1, \ldots, v_r$ tal que $\forall i = 1, \ldots, r$, $(v_i, v_{i+1}) \in E$. El peso del camino p se define por:

$$w(p) := \sum_{i=1}^{r-1} w(v_i, v_{i+1}).$$

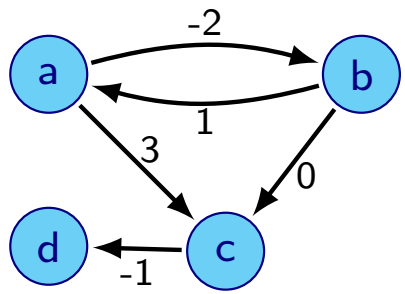
Para todo $u, v \in V$ se define el peso mínimo de los caminos de u a v por:

$$\delta(u, v) := \begin{cases} \text{ínf}\{w(p) : p \text{ es un camino de } u \text{ a } v\} & \text{si existe camino de } u \text{ a } v \text{ en } G, \\ +\infty & \text{en otro caso.} \end{cases}$$

Un camino p de u a v en G se dice de **peso mínimo** si $w(p) = \delta(u, v)$.

Observación: : $\delta(u, v) \in \mathbb{R} \cup \{-\infty, +\infty\}$. Luego, puede que exista camino de u a v en G pero no necesariamente un camino de peso mínimo.

Ejemplo:



Problema del Camino Más Corto (PCC)

Sea $G = (V, E)$ un digrafo con función de peso en los arcos $w : E \to \mathbb{R}$. Un camino en G es una secuencia de vértices (**no necesariamente distintos**) $p : v_1, \ldots, v_r$ tal que $\forall i = 1, \ldots, r$, $(v_i, v_{i+1}) \in E$. El peso del camino p se define por:

$$w(p) := \sum_{i=1}^{r-1} w(v_i, v_{i+1}).$$

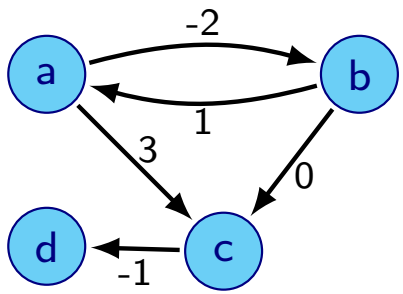
Para todo $u, v \in V$ se define el peso mínimo de los caminos de u a v por:

$$\delta(u, v) := \begin{cases} \text{ínf}\{w(p) : p \text{ es un camino de } u \text{ a } v\} & \text{si existe camino de } u \text{ a } v \text{ en } G, \\ +\infty & \text{en otro caso.} \end{cases}$$

Un camino p de u a v en G se dice de **peso mínimo** si $w(p) = \delta(u, v)$.

Observación: : $\delta(u, v) \in \mathbb{R} \cup \{-\infty, +\infty\}$. Luego, puede que exista camino de u a v en G pero no necesariamente un camino de peso mínimo.

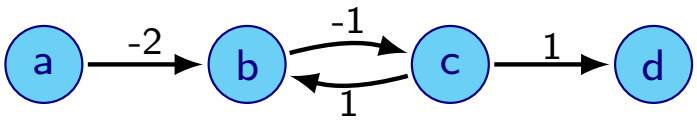
Ejemplo:



$$\delta(a, b) = -\infty,\ \delta(c, d) = -1, \delta(c, a) = +\infty.$$

Problema del Camino Más Corto (PCC)

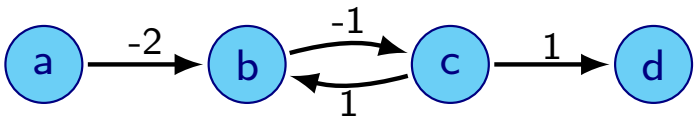
Ejemplo:



[1] PCC también puede ser definido en grafos no dirigidos. Todos los resultados y algoritmos relacionados con PCC pueden ser también aplicados en este caso.

Problema del Camino Más Corto (PCC)

Ejemplo:

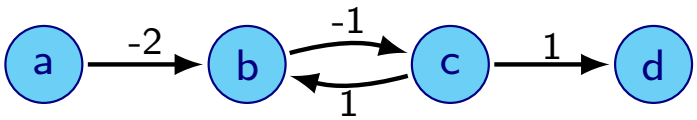


$\delta(a, d) = -2$, $p : a, b, c, d$ y
$p' : a, b, c, b, c, d$
son caminos de peso mínimo.

[1] PCC también puede ser definido en grafos no dirigidos. Todos los resultados y algoritmos relacionados con PCC pueden ser también aplicados en este caso.

Problema del Camino Más Corto (PCC)

Ejemplo:



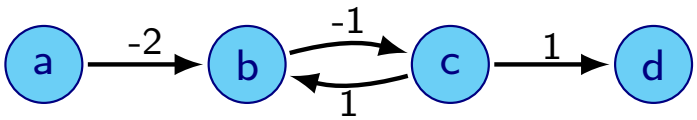
$\delta(a, d) = -2$, $p : a, b, c, d$ y
$p' : a, b, c, b, c, d$
son caminos de peso mínimo.

Definición: Dado $G = (V, E)$ un digrafo, $w : E \to \mathbb{R}$ una función peso y $s \in V$. Se define el **Problema del Camino Más Corto (PCC)** como aquel que consiste en encontrar un camino de peso mínimo (si existe) de s a u para todo $u \in V$. [1]

[1] PCC también puede ser definido en grafos no dirigidos. Todos los resultados y algoritmos relacionados con PCC pueden ser también aplicados en este caso.

Problema del Camino Más Corto (PCC)

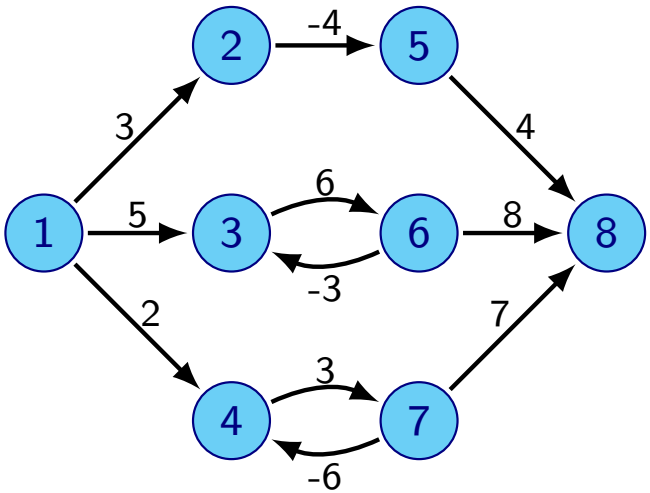
Ejemplo:



$\delta(a, d) = -2$, $p : a, b, c, d$ y
$p' : a, b, c, b, c, d$
son caminos de peso mínimo.

Definición: Dado $G = (V, E)$ un digrafo, $w : E \to \mathbb{R}$ una función peso y $s \in V$. Se define el **Problema del Camino Más Corto (PCC)** como aquel que consiste en encontrar un camino de peso mínimo (si existe) de s a u para todo $u \in V$. [1]

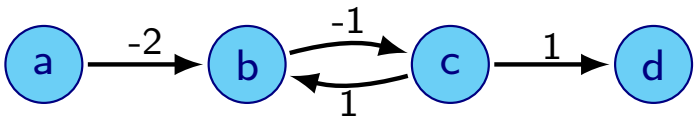
Ejemplo:



[1] PCC también puede ser definido en grafos no dirigidos. Todos los resultados y algoritmos relacionados con PCC pueden ser también aplicados en este caso.

Problema del Camino Más Corto (PCC)

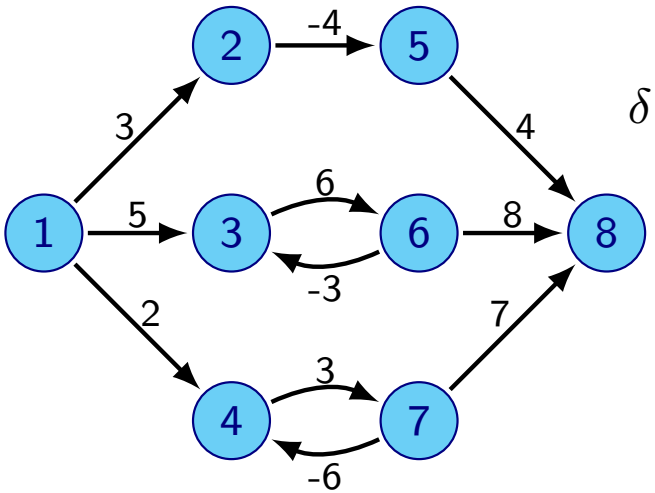
Ejemplo:



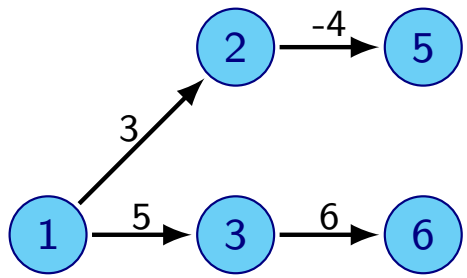
$\delta(a,d) = -2$, $p : a, b, c, d$ y
$p' : a, b, c, b, c, d$
son caminos de peso mínimo.

Definición: Dado $G = (V, E)$ un digrafo, $w : E \to \mathbb{R}$ una función peso y $s \in V$. Se define el **Problema del Camino Más Corto (PCC)** como aquel que consiste en encontrar un camino de peso mínimo (si existe) de s a u para todo $u \in V$. [1]

Ejemplo:



$\delta(1,2) = 3, \delta(1,3) = 5, \delta(1,4) = -\infty.$



[1] PCC también puede ser definido en grafos no dirigidos. Todos los resultados y algoritmos relacionados con PCC pueden ser también aplicados en este caso.

Observaciones del PCC

Proposición: *Dado $G = (V, E)$ un digrafo y $w : E \to \mathbb{R}$ una función de peso. Si $p : v_1, \ldots, v_r$ es un camino de peso mínimo, entonces*

$$\forall 1 \leq i < j \leq r, \quad p_{ij} : v_i, v_{i+1}, \ldots, v_j \text{ es camino de peso mínimo.}$$

Observaciones del PCC

Proposición: *Dado $G = (V, E)$ un digrafo y $w : E \to \mathbb{R}$ una función de peso. Si $p : v_1, \ldots, v_r$ es un camino de peso mínimo, entonces*

$$\forall 1 \leq i < j \leq r, \quad p_{ij} : v_i, v_{i+1}, \ldots, v_j \text{ es camino de peso mínimo.}$$

Demo: Si $\exists 1 \leq i < j \leq r$ tal que $p_{ij} : v_i, v_{i+1}, \ldots, v_j$ no es un camino de peso mínimo, entonces existe otro camino p'_{ij} de v_i a v_j en G tal que $w(p'_{ij}) < w(p_{ij})$.
Luego, $p' : v_1 \ldots, v_{i-1}, p'_{ij}, v_{j+1}, \ldots, v_r$ es tal que $w(p') < w(p)$ $(\to\leftarrow)$ ■.

Observaciones del PCC

Proposición: *Dado $G = (V, E)$ un digrafo y $w : E \to \mathbb{R}$ una función de peso. Si $p : v_1, \ldots, v_r$ es un camino de peso mínimo, entonces*

$$\forall 1 \leq i < j \leq r, \quad p_{ij} : v_i, v_{i+1}, \ldots, v_j \text{ es camino de peso mínimo.}$$

Demo: Si $\exists 1 \leq i < j \leq r$ tal que $p_{ij} : v_i, v_{i+1}, \ldots, v_j$ no es un camino de peso mínimo, entonces existe otro camino p'_{ij} de v_i a v_j en G tal que $w(p'_{ij}) < w(p_{ij})$.
Luego, $p' : v_1 \ldots, v_{i-1}, p'_{ij}, v_{j+1}, \ldots, v_r$ es tal que $w(p') < w(p)$ $(\rightarrow\leftarrow)$ ■.

Corolario: *Dado $G = (V, E)$ un digrafo y $w : E \to \mathbb{R}$ una función de peso. Si $p : s, \ldots, u, v$ es un camino de peso mínimo de s a v, entonces*

$$\delta(s, v) = \delta(s, u) + w(u, v).$$

Observaciones del PCC

Proposición: *Dado $G = (V, E)$ un digrafo y $w : E \to \mathbb{R}$ una función de peso. Si $p : v_1, \ldots, v_r$ es un camino de peso mínimo, entonces*

$$\forall 1 \leq i < j \leq r, \quad p_{ij} : v_i, v_{i+1}, \ldots, v_j \text{ es camino de peso mínimo.}$$

Demo: Si $\exists 1 \leq i < j \leq r$ tal que $p_{ij} : v_i, v_{i+1}, \ldots, v_j$ no es un camino de peso mínimo, entonces existe otro camino p'_{ij} de v_i a v_j en G tal que $w(p'_{ij}) < w(p_{ij})$.
Luego, $p' : v_1 \ldots, v_{i-1}, p'_{ij}, v_{j+1}, \ldots, v_r$ es tal que $w(p') < w(p)$ $(\to\leftarrow)$ ■.

Corolario: *Dado $G = (V, E)$ un digrafo y $w : E \to \mathbb{R}$ una función de peso. Si $p : s, \ldots, u, v$ es un camino de peso mínimo de s a v, entonces*

$$\delta(s, v) = \delta(s, u) + w(u, v).$$

Corolario: *Dado $G = (V, E)$ un digrafo, $w : E \to \mathbb{R}$ una función de peso y $s \in V$. Entonces, $\forall (u, v) \in E$:*

$$\delta(s, v) \leq \delta(s, u) + w(u, v).$$

Algoritmo de Dijkstra

El algoritmo de Dijkstra (1959) resuelve PCC con pesos no negativos.

Algorithm Dijkstra(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y
$w : E \to \mathbb{R}^+ \cup \{0\}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0,\ Q \leftarrow V,\ S \leftarrow \emptyset$
5: **while** $Q \neq \emptyset$ **do**
6: $\quad$ Extraer $u \in Q$ tal que
$\quad d[u] = \text{mín}\{d[a] : a \in Q\}$
7: $\quad S \leftarrow S \cup \{u\}$
8: $\quad$ **for all** $v \in V$ sucesor de u **do**
9: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
10: $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v),\ \pi[v] \leftarrow u$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: **end while**
14: **return** (d, π)

Obs: La acción de la línea 9 se conoce como **relajación del arco** (u, v). Después de la relajación de (u, v) se tiene: $d[v] \leq d[u] + w(u, v)$.

Algoritmo de Dijkstra

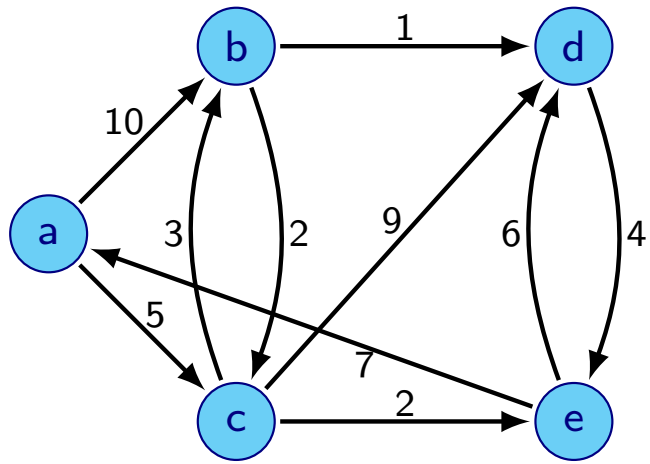
El algoritmo de Dijkstra (1959) resuelve PCC con pesos no negativos.

Algorithm Dijkstra(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y
$w : E \to \mathbb{R}^+ \cup \{0\}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0,\ Q \leftarrow V,\ S \leftarrow \emptyset$
5: **while** $Q \neq \emptyset$ **do**
6: Extraer $u \in Q$ tal que
$d[u] = \text{mín}\{d[a] : a \in Q\}$
7: $S \leftarrow S \cup \{u\}$
8: **for all** $v \in V$ sucesor de u **do**
9: **if** $d[v] > d[u] + w(u, v)$ **then**
10: $d[v] \leftarrow d[u] + w(u, v),\ \pi[v] \leftarrow u$
11: **end if**
12: **end for**
13: **end while**
14: **return** (d, π)

Obs: La acción de la línea 9 se conoce como **relajación del arco** (u, v). Después de la relajación de (u, v) se tiene: $d[v] \leq d[u] + w(u, v)$.



Algoritmo de Dijkstra

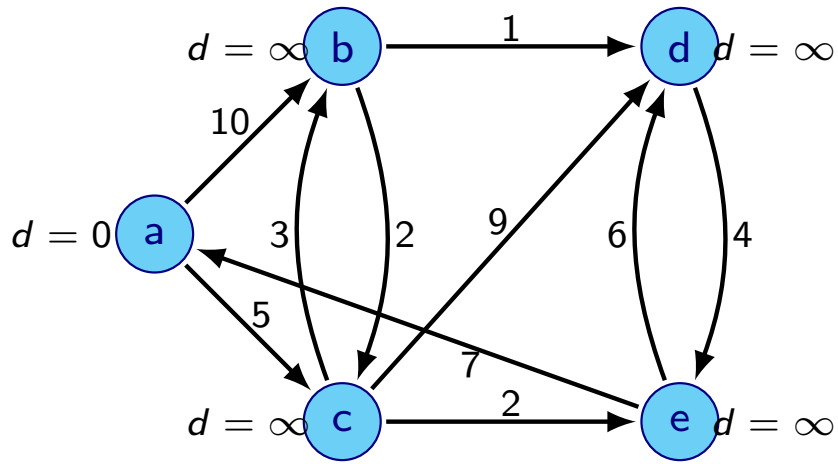
El algoritmo de Dijkstra (1959) resuelve PCC con pesos no negativos.

Algorithm Dijkstra(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y
$w : E \to \mathbb{R}^+ \cup \{0\}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0,\ Q \leftarrow V,\ S \leftarrow \emptyset$
5: **while** $Q \neq \emptyset$ **do**
6: $\quad$ Extraer $u \in Q$ tal que
$\quad d[u] = \text{mín}\{d[a] : a \in Q\}$
7: $\quad S \leftarrow S \cup \{u\}$
8: $\quad$ **for all** $v \in V$ sucesor de u **do**
9: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
10: $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v),\ \pi[v] \leftarrow u$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: **end while**
14: **return** (d, π)

Obs: La acción de la línea 9 se conoce como **relajación del arco** (u, v). Después de la relajación de (u, v) se tiene: $d[v] \leq d[u] + w(u, v)$.



- Inicio: $s = a$;
$Q = \{a, b, c, d, e\}$; $S = \emptyset$;
$\pi[s] = Null$.

Algoritmo de Dijkstra

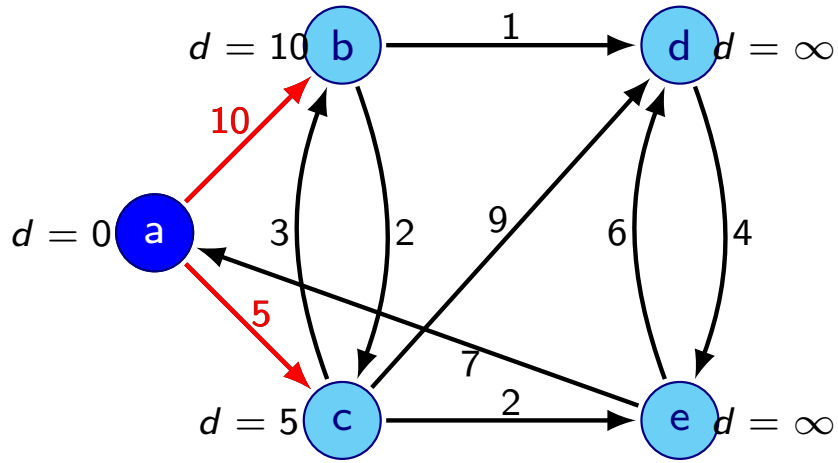
El algoritmo de Dijkstra (1959) resuelve PCC con pesos no negativos.

Algorithm Dijkstra(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y
$w : E \to \mathbb{R}^+ \cup \{0\}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0,\ Q \leftarrow V,\ S \leftarrow \emptyset$
5: **while** $Q \neq \emptyset$ **do**
6: $\quad$ Extraer $u \in Q$ tal que
$\quad d[u] = \text{mín}\{d[a] : a \in Q\}$
7: $\quad S \leftarrow S \cup \{u\}$
8: $\quad$ **for all** $v \in V$ sucesor de u **do**
9: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
10: $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v),\ \pi[v] \leftarrow u$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: **end while**
14: **return** (d, π)

Obs: La acción de la línea 9 se conoce como **relajación del arco** (u, v). Después de la relajación de (u, v) se tiene: $d[v] \leq d[u] + w(u, v)$.



- $t = 1$: $Q = \{b, c, d, e\}$;
$S = \{a\}$;
$d[b] = 10, d[c] = 5$;
$\pi[b] = \pi[c] = a$.

Algoritmo de Dijkstra

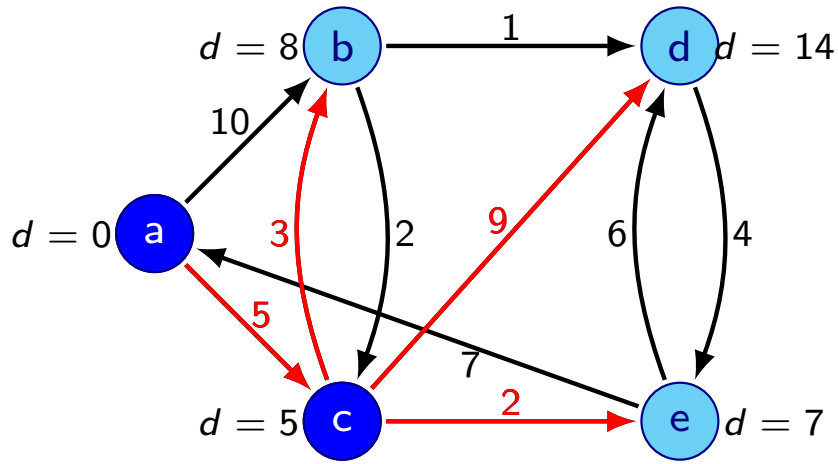
El algoritmo de Dijkstra (1959) resuelve PCC con pesos no negativos.

Algorithm Dijkstra(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y
$w : E \to \mathbb{R}^+ \cup \{0\}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0,\ Q \leftarrow V,\ S \leftarrow \emptyset$
5: **while** $Q \neq \emptyset$ **do**
6: Extraer $u \in Q$ tal que
$d[u] = \text{mín}\{d[a] : a \in Q\}$
7: $S \leftarrow S \cup \{u\}$
8: **for all** $v \in V$ sucesor de u **do**
9: **if** $d[v] > d[u] + w(u, v)$ **then**
10: $d[v] \leftarrow d[u] + w(u, v),\ \pi[v] \leftarrow u$
11: **end if**
12: **end for**
13: **end while**
14: **return** (d, π)

Obs: La acción de la línea 9 se conoce como **relajación del arco** (u, v). Después de la relajación de (u, v) se tiene: $d[v] \leq d[u] + w(u, v)$.



- $t = 2$: $Q = \{b, d, e\}$;
$S = \{a, c\}$;
$d[b] = 8, d[d] = 14$,
d[e]=7;
$\pi[b] = \pi[d] = \pi[e] = c$.

Algoritmo de Dijkstra

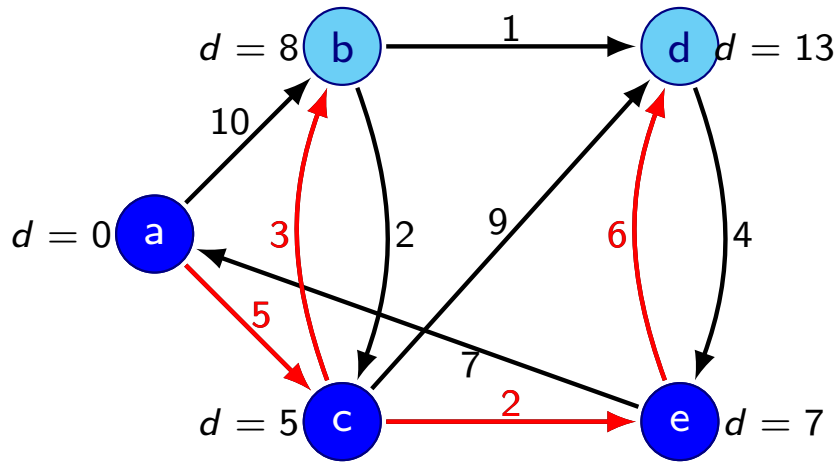
El algoritmo de Dijkstra (1959) resuelve PCC con pesos no negativos.

Algorithm Dijkstra(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y
$w : E \to \mathbb{R}^+ \cup \{0\}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0,\ Q \leftarrow V,\ S \leftarrow \emptyset$
5: **while** $Q \neq \emptyset$ **do**
6: $\quad$ Extraer $u \in Q$ tal que
$\quad d[u] = \text{mín}\{d[a] : a \in Q\}$
7: $\quad S \leftarrow S \cup \{u\}$
8: $\quad$ **for all** $v \in V$ sucesor de u **do**
9: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
10: $\quad\quad\quad d[v] \leftarrow d[u] + w(u, v),\ \pi[v] \leftarrow u$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: **end while**
14: **return** (d, π)

Obs: La acción de la línea 9 se conoce como **relajación del arco** (u, v). Después de la relajación de (u, v) se tiene: $d[v] \leq d[u] + w(u, v)$.



- $t = 3$: $Q = \{b, d\}$;
$S = \{a, c, e\}$; $d[d] = 13$;
$\pi[d] = e$.

Algoritmo de Dijkstra

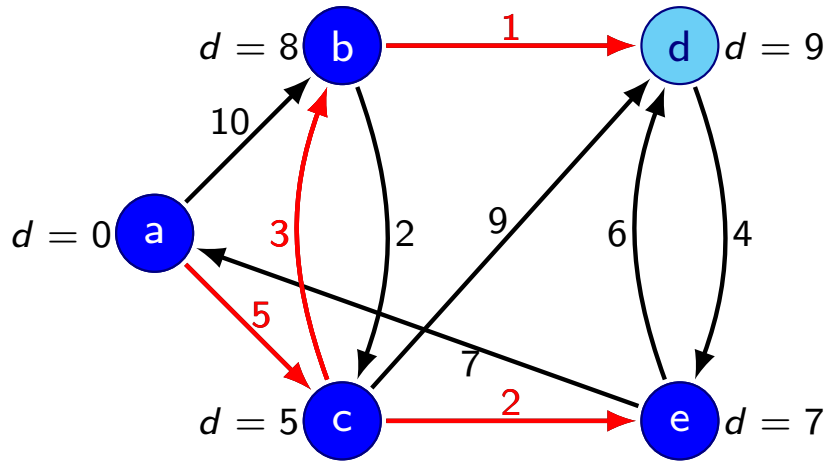
El algoritmo de Dijkstra (1959) resuelve PCC con pesos no negativos.

Algorithm Dijkstra(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y
$w : E \to \mathbb{R}^+ \cup \{0\}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0,\ Q \leftarrow V,\ S \leftarrow \emptyset$
5: **while** $Q \neq \emptyset$ **do**
6: Extraer $u \in Q$ tal que
$d[u] = \text{mín}\{d[a] : a \in Q\}$
7: $S \leftarrow S \cup \{u\}$
8: **for all** $v \in V$ sucesor de u **do**
9: **if** $d[v] > d[u] + w(u, v)$ **then**
10: $d[v] \leftarrow d[u] + w(u, v),\ \pi[v] \leftarrow u$
11: **end if**
12: **end for**
13: **end while**
14: **return** (d, π)

Obs: La acción de la línea 9 se conoce como **relajación del arco** (u, v). Después de la relajación de (u, v) se tiene: $d[v] \leq d[u] + w(u, v)$.



- $t = 4$: $Q = \{d\}$;
$S = \{a, c, e, b\}$.

Algoritmo de Dijkstra

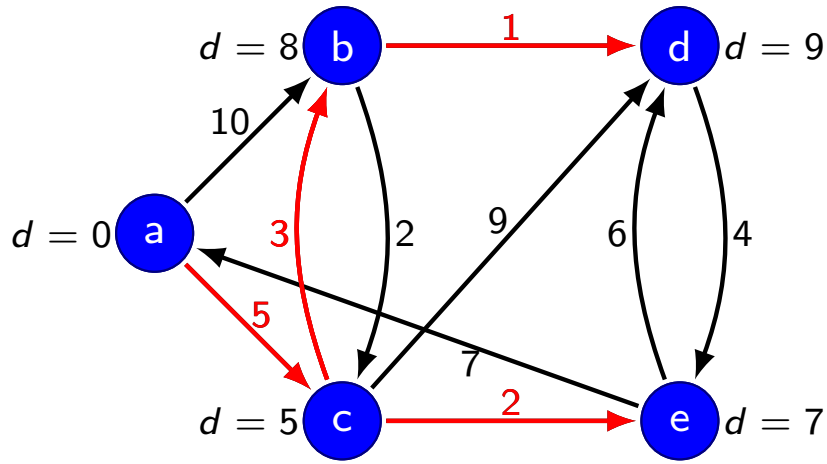
El algoritmo de Dijkstra (1959) resuelve PCC con pesos no negativos.

Algorithm Dijkstra(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y
$w : E \to \mathbb{R}^+ \cup \{0\}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0,\ Q \leftarrow V,\ S \leftarrow \emptyset$
5: **while** $Q \neq \emptyset$ **do**
6: Extraer $u \in Q$ tal que
$d[u] = \text{mín}\{d[a] : a \in Q\}$
7: $S \leftarrow S \cup \{u\}$
8: **for all** $v \in V$ sucesor de u **do**
9: **if** $d[v] > d[u] + w(u, v)$ **then**
10: $d[v] \leftarrow d[u] + w(u, v),\ \pi[v] \leftarrow u$
11: **end if**
12: **end for**
13: **end while**
14: **return** (d, π)

Obs: La acción de la línea 9 se conoce como **relajación del arco** (u, v). Después de la relajación de (u, v) se tiene: $d[v] \leq d[u] + w(u, v)$.



- $t = 5$: $Q = \emptyset$;
$S = \{a, c, e, d, b\}$.

Algoritmo de Dijkstra

A continuación algunos resultados que permiten probar que algoritmo de Dijkstra resuelve el problema PCC con pesos no negativos.

Algoritmo de Dijkstra

A continuación algunos resultados que permiten probar que algoritmo de Dijkstra resuelve el problema PCC con pesos no negativos.

Lema: *Sea* $G = (V, E)$ *un digrafo y* $w : E \to \mathbb{R}^+ \cup \{0\}$ *una función de peso con valores no negativos. Entonces,* $\forall u \in V,\ d[u] \geq \delta(s, u)$ *durante toda la ejecución del algoritmo de Dijkstra. Además, si* $d[u] = \delta(s, u)$*, entonces esta igualdad se mantiene en el resto de la ejecución del algoritmo.*

Algoritmo de Dijkstra

A continuación algunos resultados que permiten probar que algoritmo de Dijkstra resuelve el problema PCC con pesos no negativos.

Lema: *Sea* $G = (V, E)$ *un digrafo y* $w : E \to \mathbb{R}^+ \cup \{0\}$ *una función de peso con valores no negativos. Entonces,* $\forall u \in V,\ d[u] \geq \delta(s, u)$ *durante toda la ejecución del algoritmo de Dijkstra. Además, si* $d[u] = \delta(s, u)$*, entonces esta igualdad se mantiene en el resto de la ejecución del algoritmo.*

Demo: Notar que al inicio del algoritmo: $\forall v \in V, v \neq s,\ d[v] = \infty \geq \delta(s, v)$ y $d[s] = 0 = \delta(s, s)$. Luego, en cualquier caso: $d[v] \geq \delta(s, v)$.

Algoritmo de Dijkstra

A continuación algunos resultados que permiten probar que algoritmo de Dijkstra resuelve el problema PCC con pesos no negativos.

Lema: *Sea* $G = (V, E)$ *un digrafo y* $w : E \to \mathbb{R}^+ \cup \{0\}$ *una función de peso con valores no negativos. Entonces,* $\forall u \in V,\ d[u] \geq \delta(s, u)$ *durante toda la ejecución del algoritmo de Dijkstra. Además, si* $d[u] = \delta(s, u)$*, entonces esta igualdad se mantiene en el resto de la ejecución del algoritmo.*

Demo: Notar que al inicio del algoritmo: $\forall v \in V, v \neq s,\ d[v] = \infty \geq \delta(s, v)$ y $d[s] = 0 = \delta(s, s)$. Luego, en cualquier caso: $d[v] \geq \delta(s, v)$.
Supongamos que $\exists v \in V$ tal que $d[v] < \delta(s, v)$ y podemos suponer que es el primer vértice en verificar esta condición. Como al comienzo, $d[v] \geq \delta(s, v)$, $\exists (u, v) \in E$ tal que antes de la relajación de (u, v), $d[v] \geq \delta(s, v)$ y justo después $d[v] < \delta(s, v)$. Luego,

$$d[u] + w(u, v) = d[v] < \delta(s, v) \leq \delta(s, u) + w(u, v) \Longrightarrow d[u] < \delta(s, u) (\rightarrow\leftarrow).$$

Algoritmo de Dijkstra

A continuación algunos resultados que permiten probar que algoritmo de Dijkstra resuelve el problema PCC con pesos no negativos.

Lema: *Sea $G = (V, E)$ un digrafo y $w : E \to \mathbb{R}^+ \cup \{0\}$ una función de peso con valores no negativos. Entonces, $\forall u \in V,\ d[u] \geq \delta(s, u)$ durante toda la ejecución del algoritmo de Dijkstra. Además, si $d[u] = \delta(s, u)$, entonces esta igualdad se mantiene en el resto de la ejecución del algoritmo.*

Demo: Notar que al inicio del algoritmo: $\forall v \in V, v \neq s,\ d[v] = \infty \geq \delta(s, v)$ y $d[s] = 0 = \delta(s, s)$. Luego, en cualquier caso: $d[v] \geq \delta(s, v)$.
Supongamos que $\exists v \in V$ tal que $d[v] < \delta(s, v)$ y podemos suponer que es el primer vértice en verificar esta condición. Como al comienzo, $d[v] \geq \delta(s, v)$, $\exists (u, v) \in E$ tal que antes de la relajación de (u, v), $d[v] \geq \delta(s, v)$ y justo después $d[v] < \delta(s, v)$. Luego,

$$d[u] + w(u, v) = d[v] < \delta(s, v) \leq \delta(s, u) + w(u, v) \Longrightarrow d[u] < \delta(s, u) (\rightarrow\leftarrow).$$

Por otro lado, si $d[v] = \delta(s, v)$ y supongamos que después de relajar el arco (u, v) se tiene que $d[v] > \delta(s, v)$, entonces

$$d[v] = \delta(s, v) > d[u] + w(u, v) \geq \delta(s, u) + w(u, v) (\rightarrow\leftarrow). \blacksquare$$

Algoritmo de Dijkstra

Teorema: *Al término del algoritmo Dijkstra con entrada* (G, w, s)*, donde* $G = (V, E)$ *es un digrafo,* $w : E \to \mathbb{R}^+ \cup \{0\}$ *es función de pesos no negativos y* $s \in V$*, se tiene que:*

$$\forall v \in V,\ d[v] = \delta(s, v).$$

Algoritmo de Dijkstra

Teorema: *Al término del algoritmo Dijkstra con entrada* (G, w, s)*, donde* $G = (V, E)$ *es un digrafo,* $w : E \to \mathbb{R}^+ \cup \{0\}$ *es función de pesos no negativos y* $s \in V$*, se tiene que:*

$$\forall v \in V,\ d[v] = \delta(s, v).$$

Demo: Probemos por contradicción que al momento de $v \in V$ ser marcado (i.e. v ingresa a S), entonces $d[v] = \delta(s, v)$.
Supongamos que $\exists v \in V$ tal que cuando es marcado u se tiene que $d[v] \neq \delta(s, v)$, i.e. por Lema anterior que $d[v] > \delta(s, v)$. Supondremos que v es el primer nodo marcado que verifica lo anterior. Notar que $v \neq s$, pues $d[s] = 0 = \delta(s, s)$ durante todo el algoritmo y si no hay camino de s a v, entonces $d[v] = \infty = \delta(s, v)$. Luego, supongamos que existe $p : s = v_1, \ldots, v_k, v$ camino de s a v en G. Como los pesos son todos no negativos, podemos suponer también que p es un camino de peso mínimo. Además, podemos suponer, sin pérdida de generalidad, que no hay vértices repetidos en p.

Algoritmo de Dijkstra

Demo (continuación): Consideremos el momento en que v es marcado. Antes de eso, si $\forall i = 1, \ldots, k,\ v_i \in S$ (i.e. estaban marcados), entonces por hipótesis $\forall i = 1, \ldots, k,\ d[v_i] = \delta(s, v_i)$. Luego, al relajar el arco (v_k, v) se tiene que:

$$d[v] \leq d[v_k] + w(v_k, v) = \delta(s, v_k) + w(v_k, v) = \delta(s, v),$$

y por Lema anterior $\delta(s, v) \leq d[v]$, entonces $d[v] = \delta(s, v)$ lo que es una contradicción con la hipótesis.

Algoritmo de Dijkstra

Demo (continuación): Consideremos el momento en que v es marcado. Antes de eso, si $\forall i = 1, \ldots, k,\ v_i \in S$ (i.e. estaban marcados), entonces por hipótesis $\forall i = 1, \ldots, k,\ d[v_i] = \delta(s, v_i)$. Luego, al relajar el arco (v_k, v) se tiene que:

$$d[v] \leq d[v_k] + w(v_k, v) = \delta(s, v_k) + w(v_k, v) = \delta(s, v),$$

y por Lema anterior $\delta(s, v) \leq d[v]$, entonces $d[v] = \delta(s, v)$ lo que es una contradicción con la hipótesis.
Supongamos entonces que existe $u \neq v$ en p tal que u no estaba marcado al momento de marcar v y sea el primer vértice que verifica esta condición.
Entonces, por el mismo argumento que antes se tiene que $d[u] = \delta(s, u)$. Pero $d[u] = \delta(s, u) \leq \delta(s, v) < d[v]$, pues p es camino de peso mínimo. Sin embargo, esto es contradictorio con elegir a v ser marcado antes que u. ■

Algoritmo de Dijkstra

Corolario: *Al término del algoritmo de Dijkstra con entrada* (G, w, s) *obtenemos el digrafo predecesor* $G_\pi = (V_\pi, E_\pi)$ *definido por:*

- $V_\pi = \{v \in V : \ \pi[v] \neq Null\} \cup \{s\}$,
- $E_\pi = \{(\pi[v], v) : \ v \in V_\pi \setminus \{s\}\}$.

Además, G_π *contiene una solución al problema PCC, i. e. un camino de peso mínimo desde* s *a* v *para todo* $v \in V_\pi$.

Algoritmo de Dijkstra

Corolario: *Al término del algoritmo de Dijkstra con entrada* (G, w, s) *obtenemos el digrafo predecesor* $G_\pi = (V_\pi, E_\pi)$ *definido por:*

- $V_\pi = \{v \in V : \ \pi[v] \neq Null\} \cup \{s\}$,
- $E_\pi = \{(\pi[v], v) : \ v \in V_\pi \setminus \{s\}\}$.

Además, G_π *contiene una solución al problema PCC, i. e. un camino de peso mínimo desde* s *a* v *para todo* $v \in V_\pi$.

Demo: Notar que si $\pi[v] \neq Null$, entonces $d[v] = \delta(s, v) = \infty$ y por consiguiente existe camino de s a v en G. Sea $v \in V_\pi$ con $\pi[v] = u$. Luego, al relajar el arco $(u, v) \in E_\pi$ se tiene que $d[v] = d[u] + w(u, v)$. Por teorema anterior, al ser marcado v, y por ende u estaba marcado, se tiene que:

$$\delta(s, v) = d[v] = d[u] + w(u, v) = \delta(s, u) + w(u, v).$$

Algoritmo de Dijkstra

Corolario: *Al término del algoritmo de Dijkstra con entrada* (G, w, s) *obtenemos el digrafo predecesor* $G_\pi = (V_\pi, E_\pi)$ *definido por:*

- $V_\pi = \{v \in V : \ \pi[v] \neq Null\} \cup \{s\}$,
- $E_\pi = \{(\pi[v], v) : \ v \in V_\pi \setminus \{s\}\}$.

Además, G_π *contiene una solución al problema PCC, i. e. un camino de peso mínimo desde* s *a* v *para todo* $v \in V_\pi$.

Demo: Notar que si $\pi[v] \neq Null$, entonces $d[v] = \delta(s, v) = \infty$ y por consiguiente existe camino de s a v en G. Sea $v \in V_\pi$ con $\pi[v] = u$. Luego, al relajar el arco $(u, v) \in E_\pi$ se tiene que $d[v] = d[u] + w(u, v)$. Por teorema anterior, al ser marcado v, y por ende u estaba marcado, se tiene que:

$$\delta(s, v) = d[v] = d[u] + w(u, v) = \delta(s, u) + w(u, v).$$

Así, si $p : s = v_1, \ldots, v_k$ es un camino en G_π, entonces $\forall i = 1, \ldots, k$:

$$\sum_{i=1}^{k-1}(\delta(s, v_{i+1}) - \delta(s, v_i)) = \delta(s, v_k) - \delta(s, s) = \delta(s, v_k) = \sum_{i=1}^{k-1} w(v_{i+1}, v_i) = w(p).$$

Por lo tanto, p es un camino de peso mínimo en G. ■

Algoritmo de Dijkstra

Observar que el algoritmo BFS es un caso particular del algoritmo Dijkstra, considerando un grafo sin pesos como un grafo con peso igual a 1 en cada arco.

Corolario: *Al término del algoritmo de BFS con entrada* (G, s) *el digrafo predecesor obtenido contiene caminos de largo mínimo de* s *a* v *para todo* $v \in V_\pi$ *y* $d[v]$ *es la distancia de* s *a* v *para todo* $v \in V$.

Tiempo de ejecución de Dijkstra

El número de operaciones elementales ejecutadas por el algoritmo de Dijkstra con entrada $(G = (V, E), w, s)$ está dado por:

- Inicialización: $O(|V|)$ operaciones de asignación.
- Ciclo while: $O(|V|)$ iteraciones. En cada iteración del while determinar mín$\{d[a] : \ a \in Q\}$ requiere $Q - 1 = O(|V|)$ comparaciones.
- En total se realizan $O(|E|)$ relajaciones de arcos (una sola vez a lo más cada arco), que realizan $O(1)$ operaciones de comparación y/o asignación.
- Total:
  $O(|V|) + O(|V|) \cdot O(|V|) + O(|E|) = O(|V|^2 + |E|)(= O(|V|^2))$
  operaciones elementales. Es decir, el algoritmo Dijkstra es un algoritmo polinomial.

Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [2]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

```
1: for all u ∈ V do
2:     π[u] ← Null,  d[u] ← ∞
3: end for
4: d[s] ← 0
5: for all i = 1 to |V| − 1 do
6:     for all (u, v) ∈ E do
7:         if d[v] > d[u] + w(u, v) then
8:             d[v] ← d[u] + w(v, u), π[v] ← u
9:         end if
10:    end for
11: end for
12: for all (u, v) ∈ E do
13:    if d[v] > d[u] + w(u, v) then
14:        return Hay ciclo de peso negativo
15:    end if
16: end for
17: return (d, π)
```

[2] El problema PCC con pesos negativos es NP-completo

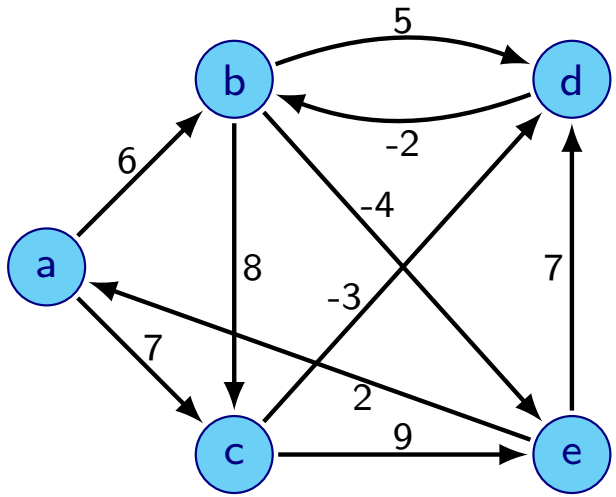
Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [2]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0$
5: **for all** $i = 1$ to $|V| - 1$ **do**
6: $\quad$ **for all** $(u, v) \in E$ **do**
7: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
8: $\quad\quad\quad d[v] \leftarrow d[u] + w(v, u),\ \pi[v] \leftarrow u$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: **for all** $(u, v) \in E$ **do**
13: $\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
14: $\quad\quad$ **return** Hay ciclo de peso negativo
15: $\quad$ **end if**
16: **end for**
17: **return** (d, π)



[2] El problema PCC con pesos negativos es NP-completo

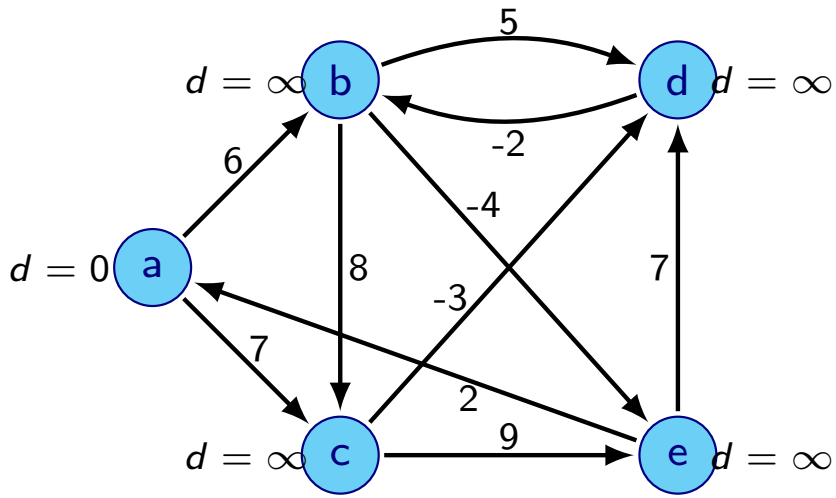
Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [2]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0$
5: **for all** $i = 1$ to $|V| - 1$ **do**
6: $\quad$ **for all** $(u, v) \in E$ **do**
7: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
8: $\quad\quad\quad d[v] \leftarrow d[u] + w(v, u),\ \pi[v] \leftarrow u$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: **for all** $(u, v) \in E$ **do**
13: $\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
14: $\quad\quad$ **return** Hay ciclo de peso negativo
15: $\quad$ **end if**
16: **end for**
17: **return** (d, π)



- Inicio: $s = a$; $\pi[a] = Null$.

[2] El problema PCC con pesos negativos es NP-completo

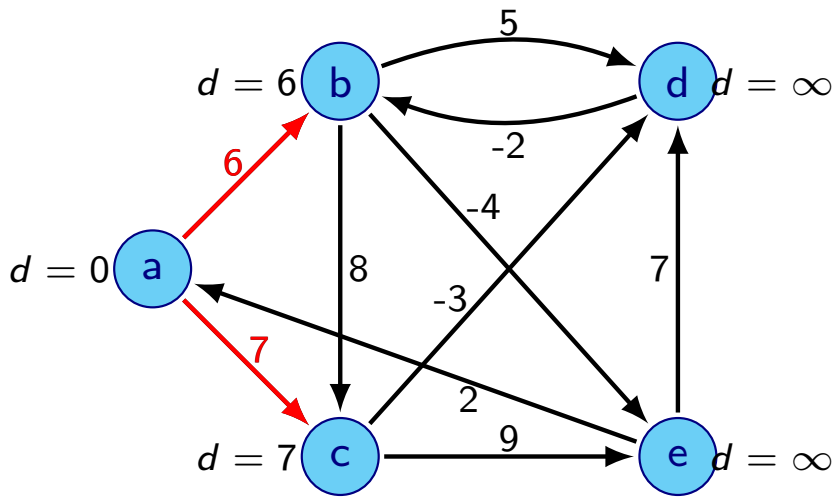
Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [2]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0$
5: **for all** $i = 1$ to $|V| - 1$ **do**
6: $\quad$ **for all** $(u, v) \in E$ **do**
7: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
8: $\quad\quad\quad d[v] \leftarrow d[u] + w(v, u),\ \pi[v] \leftarrow u$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: **for all** $(u, v) \in E$ **do**
13: $\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
14: $\quad\quad$ **return** Hay ciclo de peso negativo
15: $\quad$ **end if**
16: **end for**
17: **return** (d, π)



- $t = 1$: (a, b):
  $d[b] = 6,\ \pi[b] = a.$
  (a, c):
  $d[c] = 7,\ \pi[c] = a.$

[2] El problema PCC con pesos negativos es NP-completo

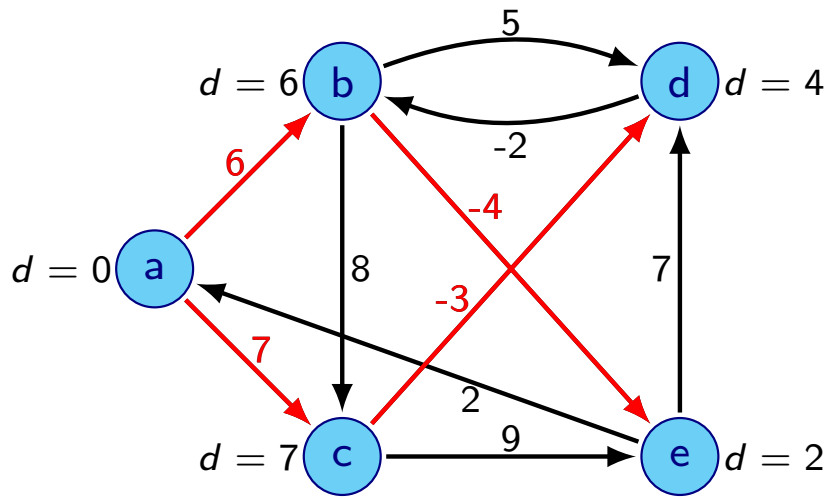
Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [2]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0$
5: **for all** $i = 1$ to $|V| - 1$ **do**
6: $\quad$ **for all** $(u, v) \in E$ **do**
7: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
8: $\quad\quad\quad d[v] \leftarrow d[u] + w(v, u),\ \pi[v] \leftarrow u$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: **for all** $(u, v) \in E$ **do**
13: $\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
14: $\quad\quad$ **return** Hay ciclo de peso negativo
15: $\quad$ **end if**
16: **end for**
17: **return** (d, π)



- $t = 2$: (b, e):
  $d[e] = 2,\ \pi[e] = b.$
  (c, d):
  $d[d] = 4,\ \pi[d] = c.$

[2] El problema PCC con pesos negativos es NP-completo

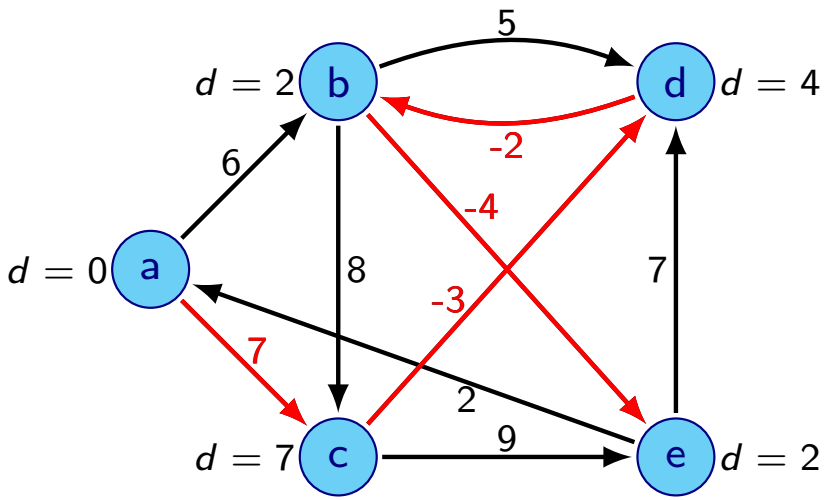
Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [2]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0$
5: **for all** $i = 1$ to $|V| - 1$ **do**
6: $\quad$ **for all** $(u, v) \in E$ **do**
7: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
8: $\quad\quad\quad d[v] \leftarrow d[u] + w(v, u),\ \pi[v] \leftarrow u$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: **for all** $(u, v) \in E$ **do**
13: $\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
14: $\quad\quad$ **return** Hay ciclo de peso negativo
15: $\quad$ **end if**
16: **end for**
17: **return** (d, π)



- $t = 3$: (d, b):
  $d[b] = 2,\ \pi[b] = d.$

[2] El problema PCC con pesos negativos es NP-completo

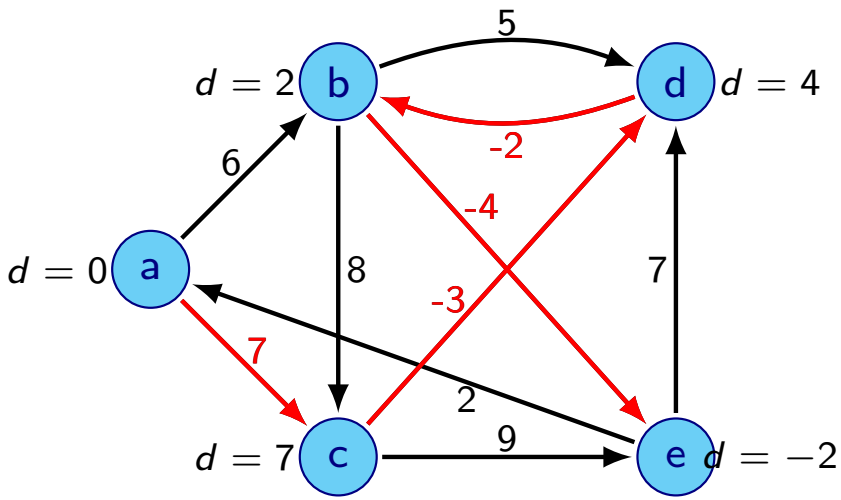
Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [2]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0$
5: **for all** $i = 1$ to $|V| - 1$ **do**
6: $\quad$ **for all** $(u, v) \in E$ **do**
7: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
8: $\quad\quad\quad d[v] \leftarrow d[u] + w(v, u),\ \pi[v] \leftarrow u$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: **for all** $(u, v) \in E$ **do**
13: $\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
14: $\quad\quad$ **return** Hay ciclo de peso negativo
15: $\quad$ **end if**
16: **end for**
17: **return** (d, π)



- $t = 4$: (b, e):
  $d[e] = -2,\ \pi[e] = b.$

[2] El problema PCC con pesos negativos es NP-completo

Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [3]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0$
5: **for all** $i = 1$ to $|V| - 1$ **do**
6: $\quad$ **for all** $(u, v) \in E$ **do**
7: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
8: $\quad\quad\quad d[v] \leftarrow d[u] + w(v, u),\ \pi[v] \leftarrow u$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: **for all** $(u, v) \in E$ **do**
13: $\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
14: $\quad\quad$ **return** Hay ciclo de peso negativo
15: $\quad$ **end if**
16: **end for**
17: **return** (d, π)

[3] El problema PCC con pesos negativos es NP-completo

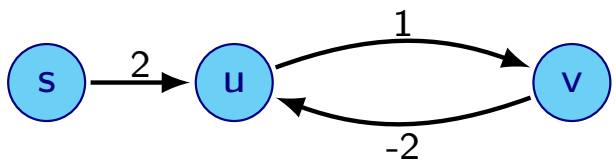
Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [3]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0$
5: **for all** $i = 1$ to $|V| - 1$ **do**
6: $\quad$ **for all** $(u, v) \in E$ **do**
7: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
8: $\quad\quad\quad d[v] \leftarrow d[u] + w(v, u),\ \pi[v] \leftarrow u$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: **for all** $(u, v) \in E$ **do**
13: $\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
14: $\quad\quad$ **return** Hay ciclo de peso negativo
15: $\quad$ **end if**
16: **end for**
17: **return** (d, π)



[3] El problema PCC con pesos negativos es NP-completo

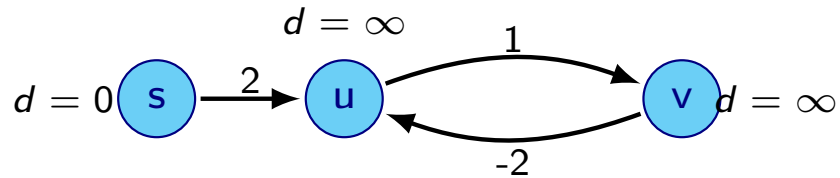
Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [3]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0$
5: **for all** $i = 1$ to $|V| - 1$ **do**
6: $\quad$ **for all** $(u, v) \in E$ **do**
7: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
8: $\quad\quad\quad d[v] \leftarrow d[u] + w(v, u),\ \pi[v] \leftarrow u$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: **for all** $(u, v) \in E$ **do**
13: $\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
14: $\quad\quad$ **return** Hay ciclo de peso negativo
15: $\quad$ **end if**
16: **end for**
17: **return** (d, π)



- Inicio: $\pi[s] = Null$; $d[s] = 0$; $d[u] = d[v] = \infty$.

[3] El problema PCC con pesos negativos es NP-completo

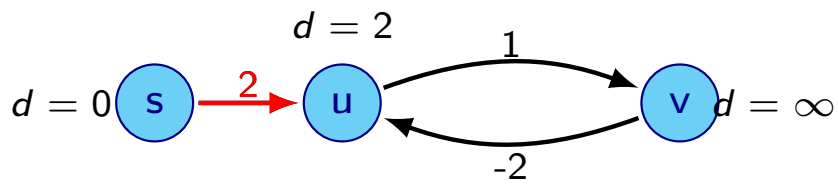
Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [3]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0$
5: **for all** $i = 1$ to $|V| - 1$ **do**
6: $\quad$ **for all** $(u, v) \in E$ **do**
7: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
8: $\quad\quad\quad d[v] \leftarrow d[u] + w(v, u),\ \pi[v] \leftarrow u$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: **for all** $(u, v) \in E$ **do**
13: $\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
14: $\quad\quad$ **return** Hay ciclo de peso negativo
15: $\quad$ **end if**
16: **end for**
17: **return** (d, π)



- $t = 1$: (s, u):
  $d[u] = 2,\ \pi[u] = s.$

[3] El problema PCC con pesos negativos es NP-completo

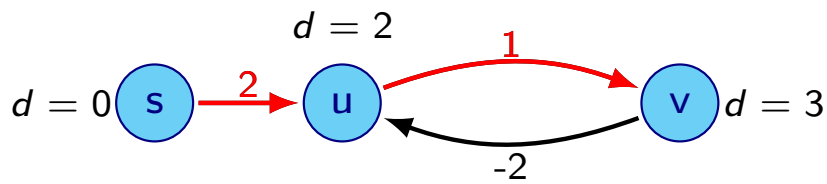
Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [3]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

```
 1: for all u ∈ V do
 2:     π[u] ← Null,  d[u] ← ∞
 3: end for
 4: d[s] ← 0
 5: for all i = 1 to |V| − 1 do
 6:     for all (u, v) ∈ E do
 7:         if d[v] > d[u] + w(u, v) then
 8:             d[v] ← d[u] + w(v, u), π[v] ← u
 9:         end if
10:     end for
11: end for
12: for all (u, v) ∈ E do
13:     if d[v] > d[u] + w(u, v) then
14:         return  Hay ciclo de peso negativo
15:     end if
16: end for
17: return  (d, π)
```



- $t = 2$: (u, v):
  $d[v] = 3,\ \pi[v] = u.$

[3] El problema PCC con pesos negativos es NP-completo

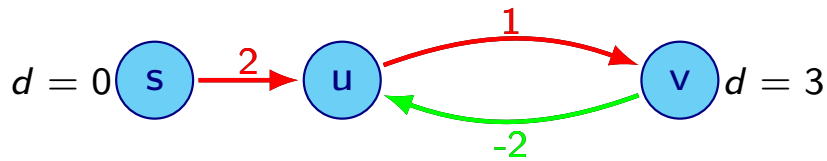
Algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford (1956) resuelve PCC con pesos reales siempre que no haya ciclos de peso negativo. Si hay un ciclo de peso negativo el algoritmo se detiene. [3]

Algorithm Bellman-Ford(G,w,s)

Input: $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ función de pesos.

1: **for all** $u \in V$ **do**
2: $\quad \pi[u] \leftarrow Null,\ d[u] \leftarrow \infty$
3: **end for**
4: $d[s] \leftarrow 0$
5: **for all** $i = 1$ to $|V| - 1$ **do**
6: $\quad$ **for all** $(u, v) \in E$ **do**
7: $\quad\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
8: $\quad\quad\quad d[v] \leftarrow d[u] + w(v, u),\ \pi[v] \leftarrow u$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: **for all** $(u, v) \in E$ **do**
13: $\quad$ **if** $d[v] > d[u] + w(u, v)$ **then**
14: $\quad\quad$ **return** Hay ciclo de peso negativo
15: $\quad$ **end if**
16: **end for**
17: **return** (d, π)



Test: (v,u):
$2 = d[u] > d[v] + w(v, u) = 1$
retornar: Hay ciclo de peso negativo.

[3] El problema PCC con pesos negativos es NP-completo

Tiempo de ejecución de Bellman-Ford

El número de operaciones elementales ejecutadas por el algoritmo de Dijkstra con entrada $(G = (V, E), w, s)$ está dado por:

- Inicialización: $O(|V|)$ operaciones de asignación.
- Ciclo for: $O(|V|)$ iteraciones. En cada iteración se realizan $O(|E|)$ relajaciones de arcos. Cada una de ellas hace $O(1)$ operaciones fundamentales.
- En el ultimo ciclo for (Test) se realizan $O(|E|)$ operaciones fundamentales.
- Total:
  $O(|V|) + O(|V|) \cdot O(|E|) \cdot O(1) + O(|E|) = O(|V| \cdot |E|)(= O(|V|^3)$ operaciones elementales. Es decir, el algoritmo Bellman-Ford es un algoritmo polinomial.

Algoritmo de Bellman-Ford

A continuación algunos resultados que permiten probar que el algoritmo de Bellman-Ford (G, w, s) resuelve el problema PCC con instancia (G, w, s) sin ciclos negativos alcanzables desde s.

Algoritmo de Bellman-Ford

A continuación algunos resultados que permiten probar que el algoritmo de Bellman-Ford (G, w, s) resuelve el problema PCC con instancia (G, w, s) sin ciclos negativos alcanzables desde s.

Lema: *Sea $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ una función de peso tal que G no tiene ciclo de peso negativo alcanzable desde s. Entonces, al término del algoritmo de Bellman-Ford con entrada (G, w, s) se tiene que $\forall u \in V : \ d[u] = \delta(s, u)$.*

Algoritmo de Bellman-Ford

A continuación algunos resultados que permiten probar que el algoritmo de Bellman-Ford (G, w, s) resuelve el problema PCC con instancia (G, w, s) sin ciclos negativos alcanzables desde s.

Lema: *Sea $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ una función de peso tal que G no tiene ciclo de peso negativo alcanzable desde s. Entonces, al término del algoritmo de Bellman-Ford con entrada (G, w, s) se tiene que $\forall u \in V : \ d[u] = \delta(s, u)$.*

Demo: De manera análoga a lo hecho para el algoritmo Dijkstra se puede probar que $\forall u \in V : \ d[u] \geq \delta(s, u)$ durante todo el algoritmo y que si en algún momento $d[u] = \delta(s, u)$, entonces esta igualdad permanece hasta al final del algoritmo.

Algoritmo de Bellman-Ford

A continuación algunos resultados que permiten probar que el algoritmo de Bellman-Ford (G, w, s) resuelve el problema PCC con instancia (G, w, s) sin ciclos negativos alcanzables desde s.

Lema: *Sea $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ una función de peso tal que G no tiene ciclo de peso negativo alcanzable desde s. Entonces, al término del algoritmo de Bellman-Ford con entrada (G, w, s) se tiene que $\forall u \in V : \ d[u] = \delta(s, u)$.*

Demo: De manera análoga a lo hecho para el algoritmo Dijkstra se puede probar que $\forall u \in V : \ d[u] \geq \delta(s, u)$ durante todo el algoritmo y que si en algún momento $d[u] = \delta(s, u)$, entonces esta igualdad permanece hasta al final del algoritmo.
Sea $p : s = v_1, \ldots, v_k$ un camino de s a v_k que podemos suponer de peso mínimo, pues de no existir entonces habría un camino de s a v_k que contiene un ciclo de peso negativo en su interior, i.e. un ciclo de peso negativo alcanzable desde s, lo que es contradictorio con la hipótesis. También podemos suponer, sin pérdida de generalidad, que p no tiene vértices repetidos.

Algoritmo de Bellman-Ford

Demo (continuación): Probemos por inducción que al final del algoritmo: $\forall i = 1, \ldots, k,\ d[v_i] = \delta(s, v_i)$.
Para $i = 1 : d[v_1 = s] = 0 = \delta(s, s)$ (al inicio del algoritmo $d[s] = 0 = \delta(s, s)$ y por resultado anterior esta igualdad se mantiene hasta el final).
Supongamos que $\forall j = 1, \ldots, i-1, i \leq k,\ d[v_j] = \delta(s, v_j)$. Luego, sabemos que durante todo el algoritmo: $d[v_i] \geq \delta(s, v_i)$ y como p es de peso mínimo, $\delta(s, v_i) = \delta(s, v_{i-1}) + w(v_{i-1}, v_i)$. Así, $d[v_i] \geq \delta(s, v_{i-1}) + w(v_{i-1}, v_i)$. Además, al relajar el arco (v_{i-1}, v_i) se tiene que
$d[v_i] \leq d[v_{i-1}] + w(v_{i-1}, v_i) = \delta(s, v_{i-1}) + w(v_{i-1}, v_i)$. De aquí,
$d[v_i] = \delta(s, v_{i-1}) + w(v_{i-1}, v_i) = \delta(s, v_i)$. ■

Teorema: *Sea $G = (V, E)$ un digrafo, $s \in V$ y $w : E \to \mathbb{R}$ una función de peso. Si G no tiene ciclo de peso negativo alcanzable desde s, entonces al término del algoritmo de Bellman-Ford con entrada (G, w, s) se tiene que $\forall u \in V, d[u] = \delta(s, u)$. Además, el digrafo predecesor G_π, definido de la misma manera que para Dijkstra, contiene una solución a PCC con instancia (G, w, s). Si G tiene un ciclo de peso negativo alcanzable desde s, el algoritmo retorna "Hay un ciclo de peso negativo".*

Algoritmo de Bellman-Ford

Demo: Si G no tiene ciclo de peso negativo alcanzable desde s, por lema anterior se tiene que: $\forall u \in V, d[u] = \delta(s, u)$. Además, en forma análoga a lo hecho en el caso del algoritmo de Dijkstra se prueba que G_π contiene una solución a PCC con instancia (G, w, s). Supongamos entonces que G tiene un ciclo $c : v_1, \ldots, v_k, v_1$ de peso negativo alcanzable desde s. Como C es alcanzable desde s entonces $\forall i = 1, \ldots, k,\ 0 \leq d[v_i] < \infty$. Supongamos que justo antes del Test se tiene que: $\forall i = 1, \ldots, k,\ d[v_{i+1}] \leq d[v_i] + w(v_i, v_{i+1})$, con $v_{k+1} \equiv v_1$, entonces se tiene que:

$$\sum_{i=1}^{k} d[v_{i+1}] \leq \sum_{i=1}^{k} d[v_i] + \sum_{i=1}^{k} w(v_i, v_{i+1}) = w(c) \Longrightarrow w(c) \leq 0 (\rightarrow\leftarrow).$$

Por lo tanto, $\exists (v_i, v_{i+1}) \in E(c),\ d[v_{i+1}] > d[v_i] + w(v_i, v_{i+1})$. ■

Algoritmo de Floyd-Warshall

El algoritmo de Floyd-Warshall (1962) permite resolver en forma simultánea el PCC desde cualquier nodo inicial, con pesos reales y sin ciclo de peso negativo.

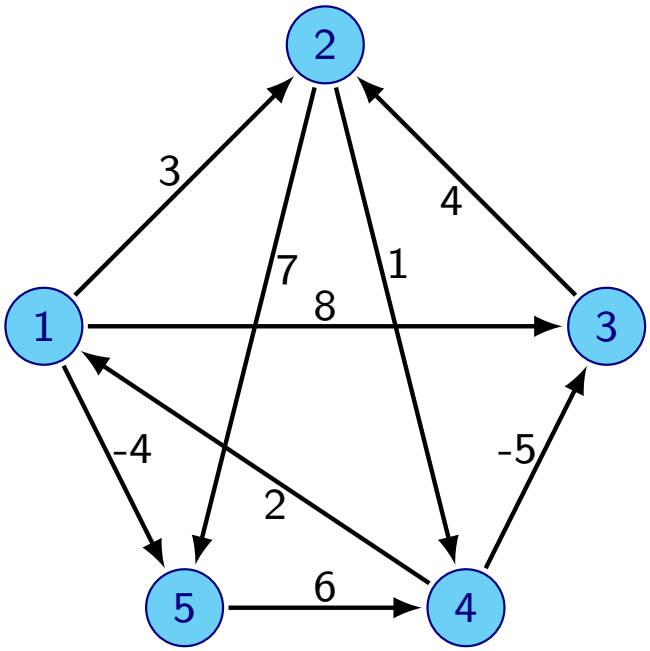
Definición: Sea $G = (V, E)$ un digrafo, que supondremos sin bucles, con $V = \{1, 2, \ldots, n\}$ y $w : E \to \mathbb{R}$ función de peso. Se define la **matriz de peso** asociada $W = (w_{ij}) \in M_n(\mathbb{R} \cup \{+\infty\})$ por:

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

Por otro lado, $\forall i \neq j \in V$, $\forall k = 0, \ldots, n$, denotaremos por p_{ij}^k (cuando exista) un camino de i a j en G con vértices intermedios en el conjunto $\{1, \ldots, k\}$ si $k \geq 1$ y sin vértices intermedios si $k = 0$. Además, p_{ij}^k es camino de peso mínimo entre todos los caminos con las características anteriores. Sin pérdida de generalidad, supondremos que p_{ij}^k no tiene vértices repetidos. Por otro lado, se define d_{ij}^k como el peso de un camino p_{ij}^k cuando existe, i.e. $d_{ij}^k := w(p_{ij}^k)$, y en caso contrario $d_{ij}^k := \infty$.

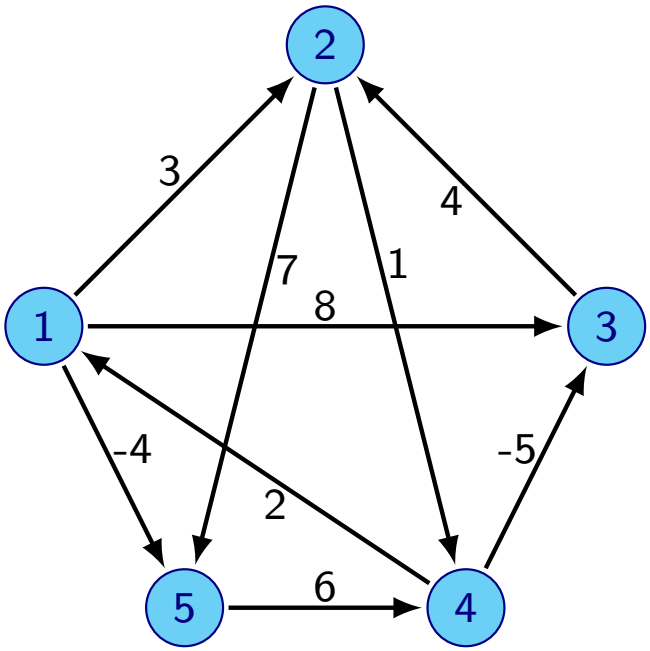
Algoritmo de Floyd-Warshall

Ejemplo:



Algoritmo de Floyd-Warshall

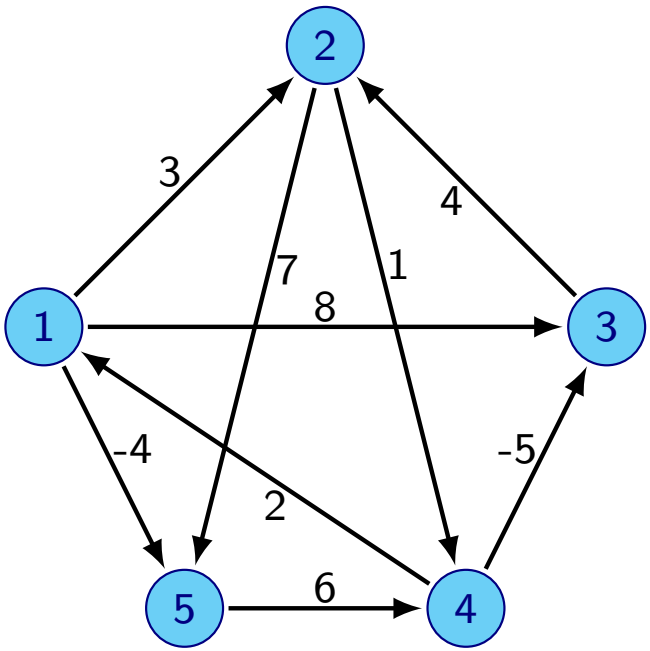
Ejemplo:



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

Algoritmo de Floyd-Warshall

Ejemplo:

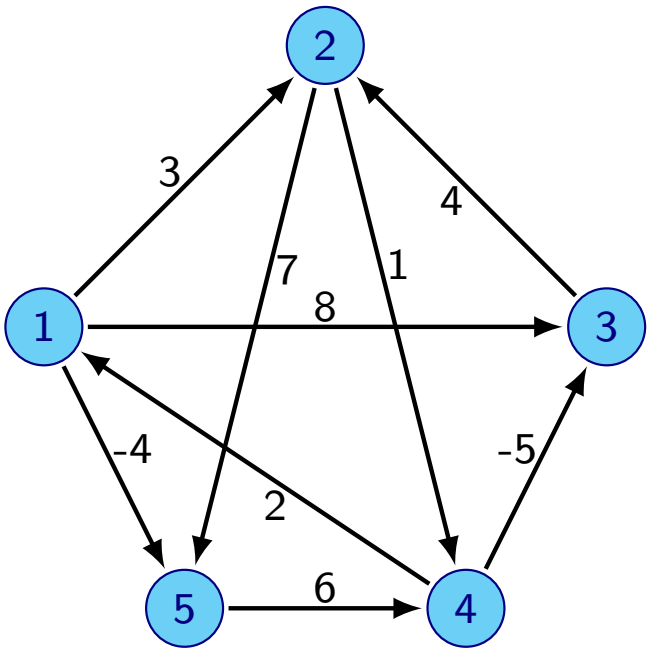


$$W = (w_{ij}) = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

$p_{25}^0 : 2, 5 = p_{25}^1 = p_{25}^2 = p_{25}^3$,
$p_{25}^4 : 2, 4, 1, 5 = p_{25}^5$. Luego,
$d_{25}^0 = d_{25}^1 = d_{25}^2 = d_{25}^3 = 7$,
$d_{25}^4 = d_{25}^5 = -1 = \delta(2, 5)$.

Algoritmo de Floyd-Warshall

Ejemplo:



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

No existen: p^0_{23}, p^1_{23}, p^2_{23}, p^3_{23}.
$p^4_{23} : 2, 4, 3 = p^5_{23}$. Luego,
$d^0_{23} = d^1_{23} = d^2_{23} = d^3_{23} = \infty$,
$d^4_{23} = d^5_{23} = -4 = \delta(2, 3)$.

Algoritmo de Floyd-Warshall

Observación: :

- Si k es un vértice interior de p_{ij}^k, i.e. $p_{ij}^k : i, \ldots, k, \ldots, j$, entonces p_{ik}^{k-1} y p_{kj}^{k-1} son respectivamente caminos de i a k y de k a j con vértices interiores en $\{1, \ldots, k-1\}$ y de peso mínimo entre todos los caminos de estas características. Luego, se tiene que:

$$d_{ij}^k = d_{ik}^{k-1} + d_{kj}^{k-1}.$$

- Si k no es un vértice interior de p_{ij}^k, entonces $p_{ij}^k = p_{ij}^{k-1}$. Luego,

$$d_{ij}^k = d_{ij}^{k-1}.$$

- En resumen, se tiene que $\forall i, j, k \in V$:

$$d_{ij}^k = \text{mín}\{d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\}.$$

- Además, se define $\forall i, j \in V$, d_{ij}^0 el peso de un camino mínimo de i a j sin vértices intermedios, i.e. $\forall i, j \in V$:

$$d_{ij}^0 = \begin{cases} 0 & \text{si } i = j, \\ w(i,j) & \text{si } (i,j) \in E, \\ \infty & \text{si } (i,j) \notin E. \end{cases} \implies d_{ij}^0 = w_{ij}.$$

Algoritmo de Floyd-Warshall

Observación:

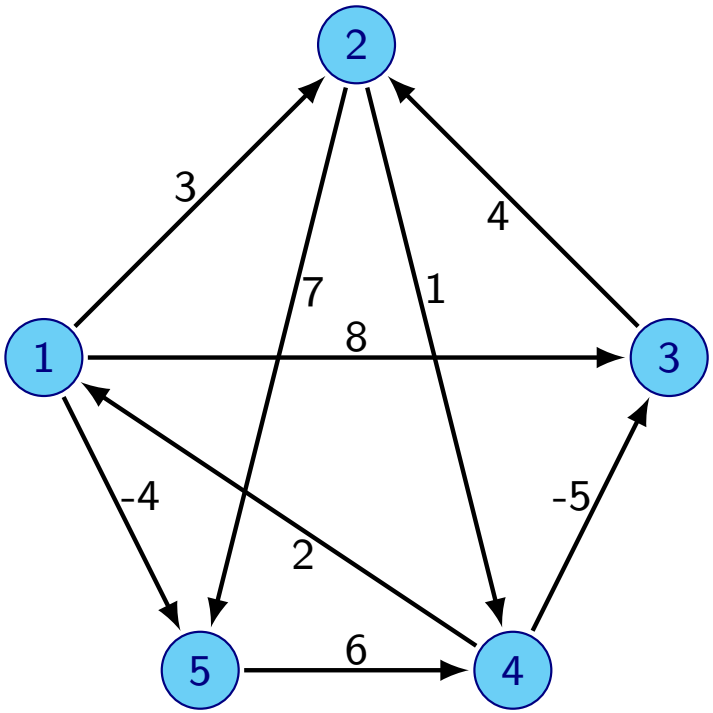
- Por lo tanto, se tiene $\forall i, j \in V, \forall k = 0, 1, \ldots, n$:

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

Así, $\delta(i, j) = d_{ij}^{n}$.

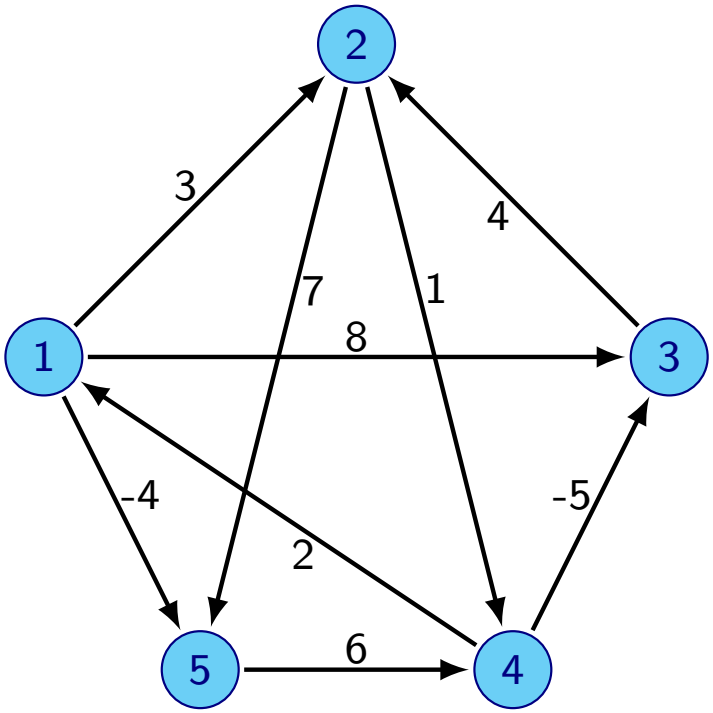
Algoritmo de Floyd-Warshall

Ejemplo:



Algoritmo de Floyd-Warshall

Ejemplo:

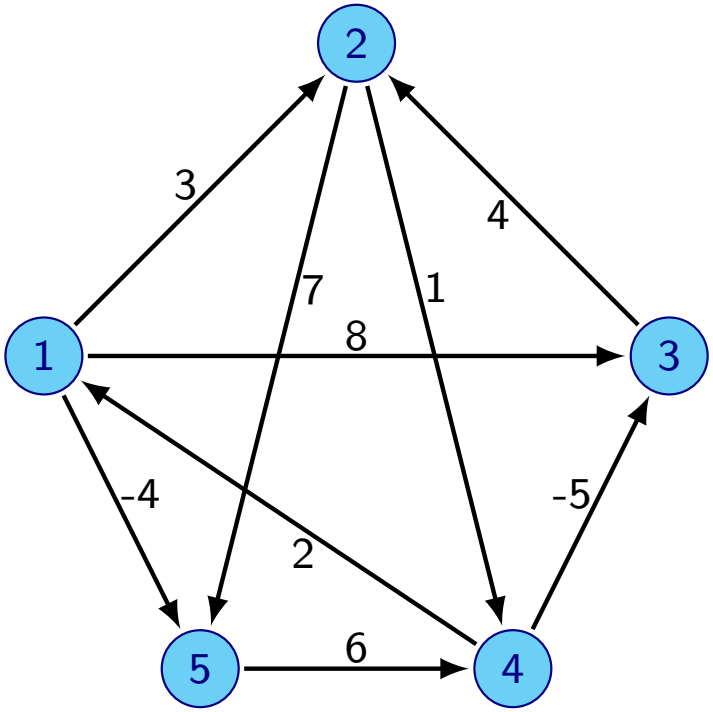


$$D^0 = W = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

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

Algoritmo de Floyd-Warshall

Ejemplo:

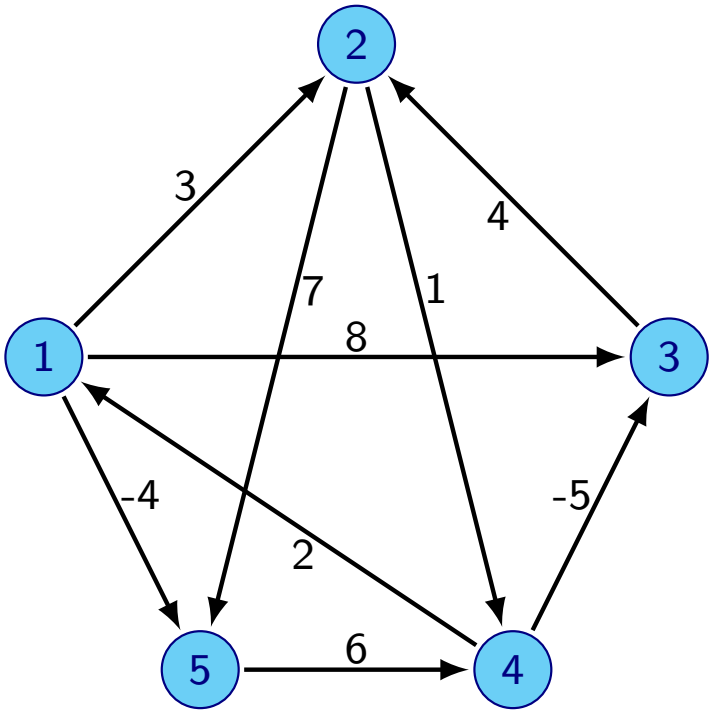


$$D^1 = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

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

Algoritmo de Floyd-Warshall

Ejemplo:

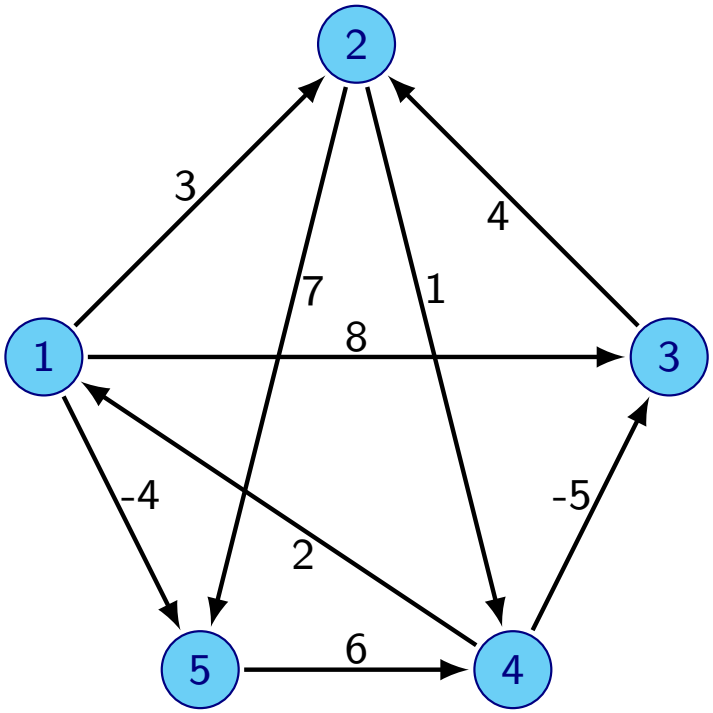


$$D^2 = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

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

Algoritmo de Floyd-Warshall

Ejemplo:

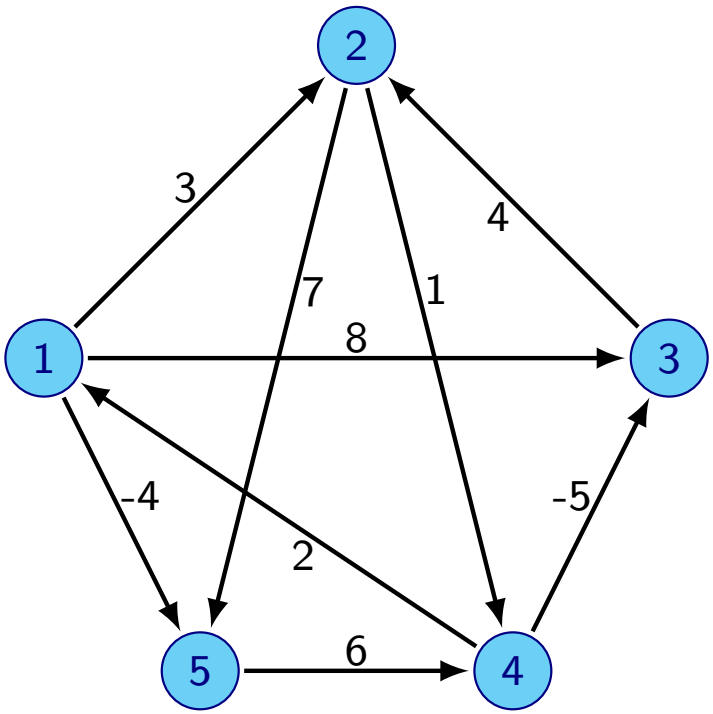


$$D^3 = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

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

Algoritmo de Floyd-Warshall

Ejemplo:

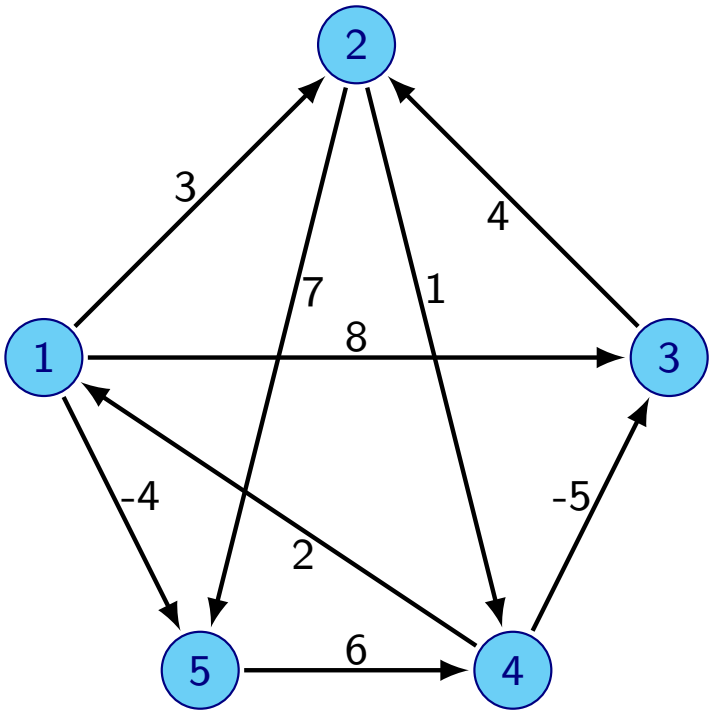


$$D^4 = \begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

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

Algoritmo de Floyd-Warshall

Ejemplo:



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

Algoritmo de Floyd-Warshall: Matriz de predecesores

Para la construcción de los caminos más cortos entre cualquier par de vértices se define las matrices de predecesores como sigue:
Definición: Sea $\Pi^k = (\pi_{ij}^k) \in M_m(\{1, \ldots, n\})$ la matriz donde π_{ij}^k es el predecesor del nodo j en el camino p_{ij}^k, es decir

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

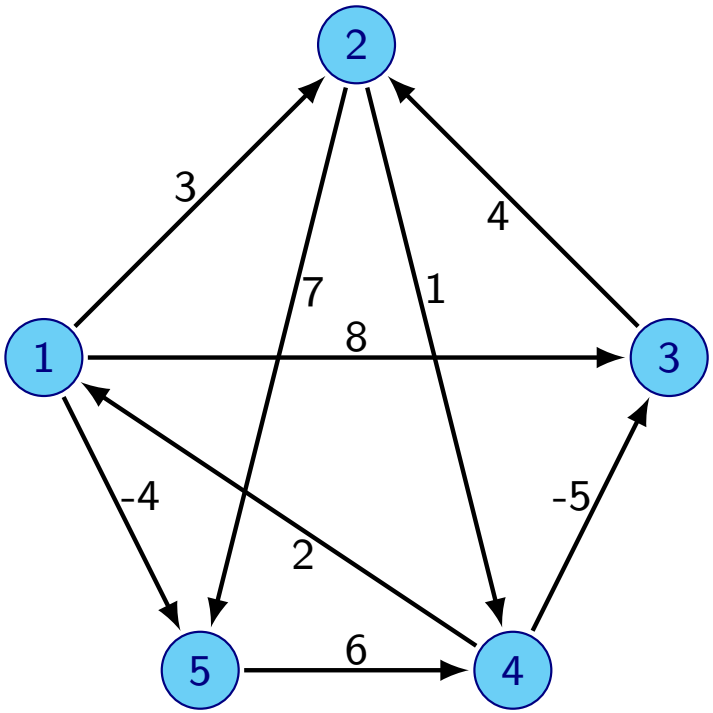
y $\forall k \in \{1, \ldots, n\}$:

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

De esta forma, π_{ij}^n es el predecesor del nodo j en un camino de peso mínimo desde i a j.

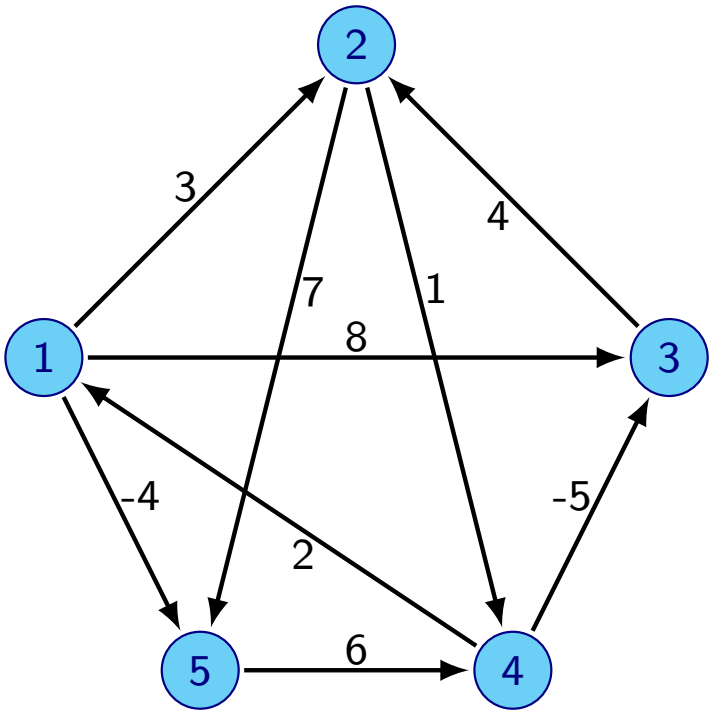
Algoritmo de Floyd-Warshall

Ejemplo:



Algoritmo de Floyd-Warshall

Ejemplo:

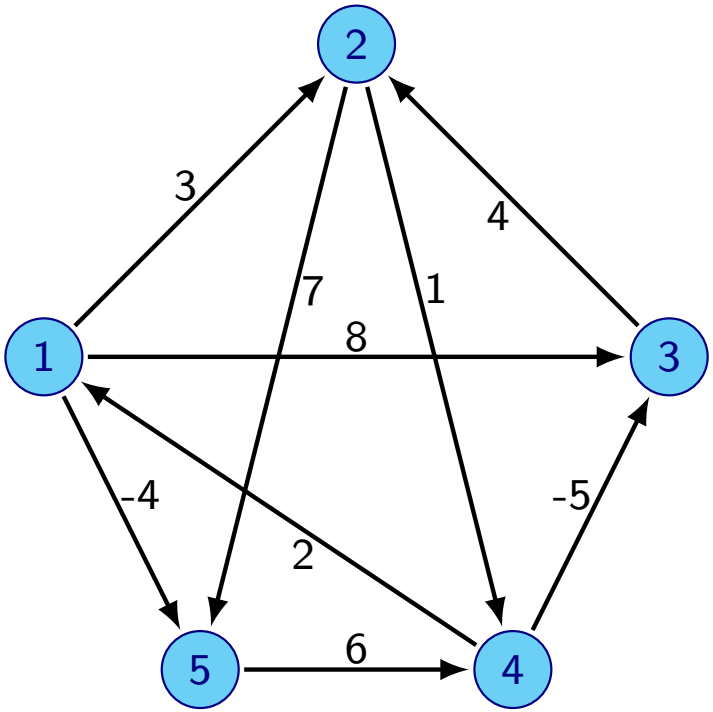


$$D^0 = W = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

$$\Pi^0 = \begin{bmatrix} N & 1 & 1 & N & 1 \\ N & N & N & 2 & 2 \\ N & 3 & N & N & N \\ 4 & N & 4 & N & N \\ N & N & N & 5 & N \end{bmatrix}$$

Algoritmo de Floyd-Warshall

Ejemplo:



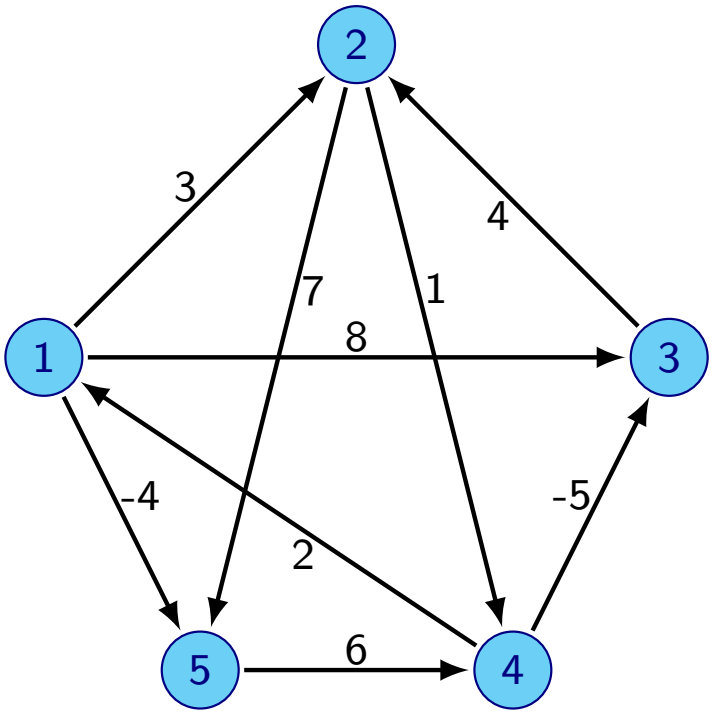
$$D^0 = W = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

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

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

Algoritmo de Floyd-Warshall

Ejemplo:



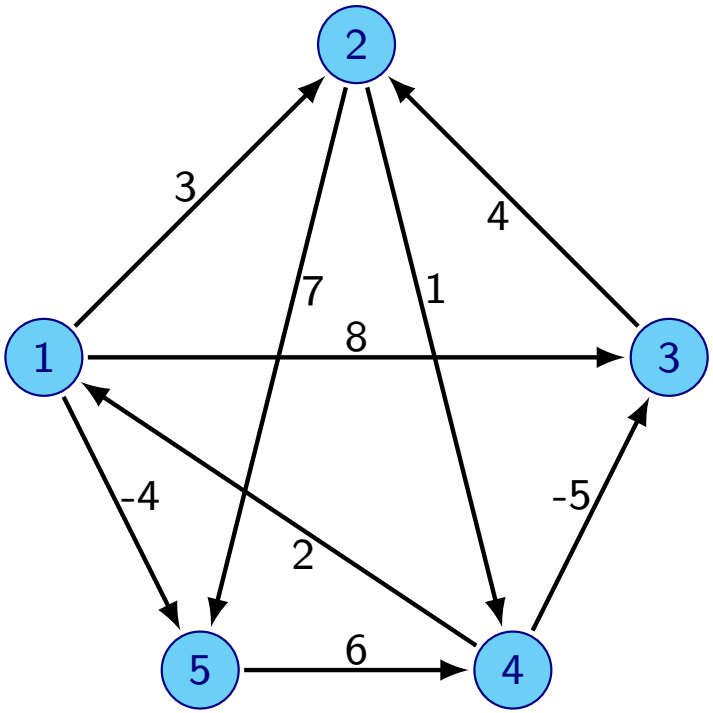
$$D^1 = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

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

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

Algoritmo de Floyd-Warshall

Ejemplo:



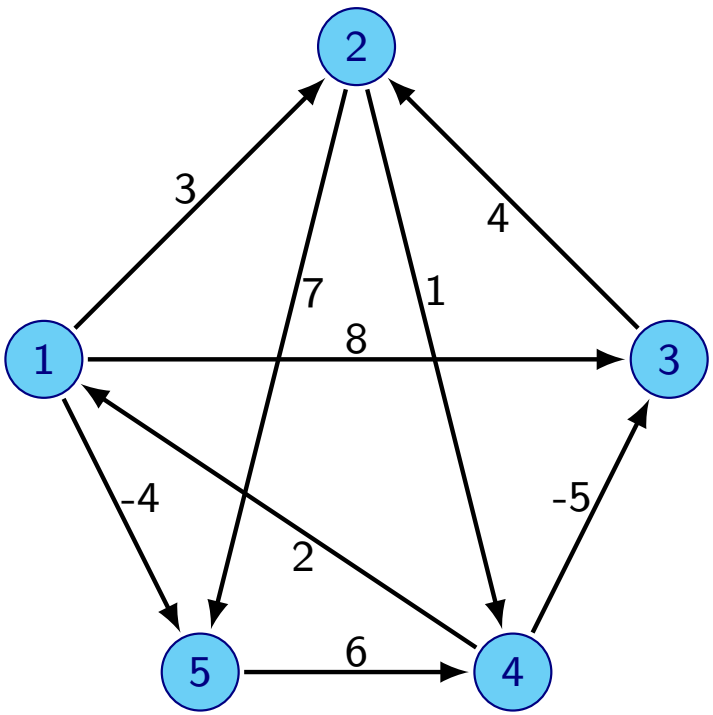
$$D^2 = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

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

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

Algoritmo de Floyd-Warshall

Ejemplo:



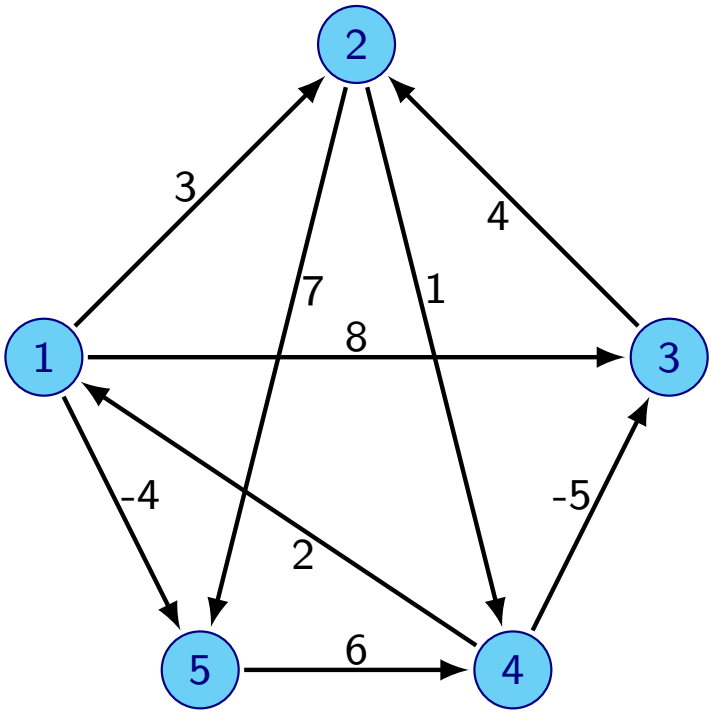
$$D^3 = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

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

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

Algoritmo de Floyd-Warshall

Ejemplo:



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

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

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

Algoritmo de Floyd-Warshall

A partir de la matriz Π^n podemos definir los digrafos predecesores $G_{\pi,i} = (V_{\pi,i}, E_{\pi,i})$ que contienen un camino de peso mínimo de i a todos los vértices alcanzables desde i en G como sigue:

$$\forall i \in V, \quad V_{\pi,i} = \{j \in V : \pi_{ij}^n \neq N\} \cup \{i\} \quad \wedge \quad E_{\pi,i} = \{(\pi_{ij}^n, j) : j \in V_{\pi,i} \backslash \{i\}\}.$$

Ejemplo: Para el ejemplo anterior se tiene:

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

Algoritmo de Floyd-Warshall

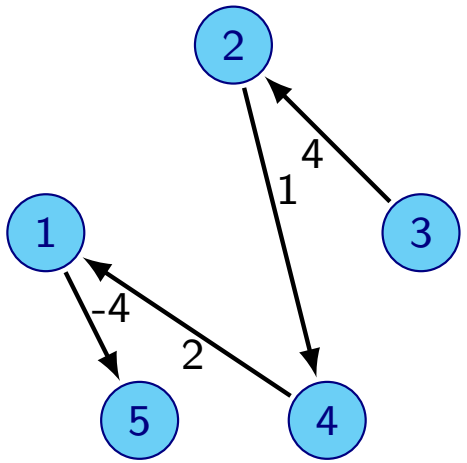
A partir de la matriz Π^n podemos definir los digrafos predecesores $G_{\pi,i} = (V_{\pi,i}, E_{\pi,i})$ que contienen un camino de peso mínimo de i a todos los vértices alcanzables desde i en G como sigue:

$$\forall i \in V, \quad V_{\pi,i} = \{j \in V : \pi_{ij}^n \neq N\} \cup \{i\} \quad \wedge \quad E_{\pi,i} = \{(\pi_{ij}^n, j) : j \in V_{\pi,i} \backslash \{i\}\}.$$

Ejemplo: Para el ejemplo anterior se tiene:

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

$G_{\pi,3}$:



Algoritmo de Floyd-Warshall

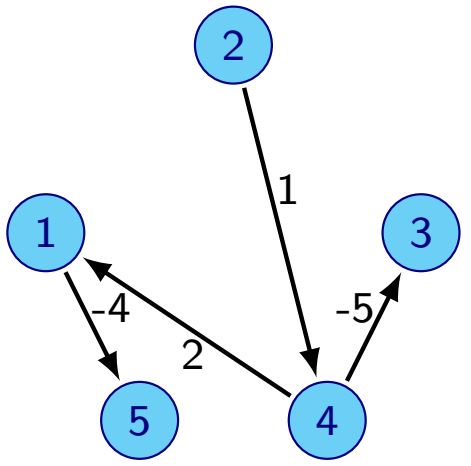
A partir de la matriz Π^n podemos definir los digrafos predecesores $G_{\pi,i} = (V_{\pi,i}, E_{\pi,i})$ que contienen un camino de peso mínimo de i a todos los vértices alcanzables desde i en G como sigue:

$$\forall i \in V, \quad V_{\pi,i} = \{j \in V : \pi_{ij}^n \neq N\} \cup \{i\} \quad \wedge \quad E_{\pi,i} = \{(\pi_{ij}^n, j) : j \in V_{\pi,i} \backslash \{i\}\}.$$

Ejemplo: Para el ejemplo anterior se tiene:

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

$G_{\pi,2}$:



Algoritmo de Floyd-Warshall

Algorithm Floyd-Warshall(G,w)

Input: $G = (V, E)$ un digrafo y $w : E \to \mathbb{R}$ función de pesos tal que no hay ciclos de peso negativo.

1: $D^0 \leftarrow W$
2: **for** $i, j =$ to $|V|$ **do**
3: **if** $i \neq j \wedge w_{ij} < \infty$ **then**
4: $\pi^0_{ij} \leftarrow i$
5: **else**
6: $\pi^0_{ij} \leftarrow N$ (Null)
7: **end if**
8: **end for**
9: **for** $i, j, k = 1$ to $|V|$ **do**
10: **if** $d^{k-1}_{ik} + d^{k-1}_{kj} < d^{k-1}_{ij}$ **then**
11: $d^k_{ij} \leftarrow d^{k-1}_{ik} + d^{k-1}_{kj}$
12: $\pi^k_{ij} \leftarrow \pi^{k-1}_{kj}$
13: **else**
14: $d^k_{ij} \leftarrow d^{k-1}_{ij}$
15: $\pi^k_{ij} \leftarrow \pi^{k-1}_{ij}$
16: **end if**
17: **end for**
18: **return** (D^n, Π^n)

Algoritmo de Floyd-Warshall

Teorema: *El algoritmo de Floyd-Warshall con entrada* (G, w)*, donde* $G = (V, E)$ *es un digrafo con* $|V| = n$ *y* $w : E \to \mathbb{R}$ *es función de pesos tal que* G *no tiene ciclo de peso negativo, retorna* (D^n, Π^n)*, después de* $O(|V|^3)$ *operaciones elementales, donde* $\forall i, j \in V : D^n_{ij} = \delta(i, j)$ *y cuyos digrafos predecesores* $G_{\pi,i}$ *formados a partir de* Π^n *contiene una solución al PCC desde todo vértice* $i \in V$*.*