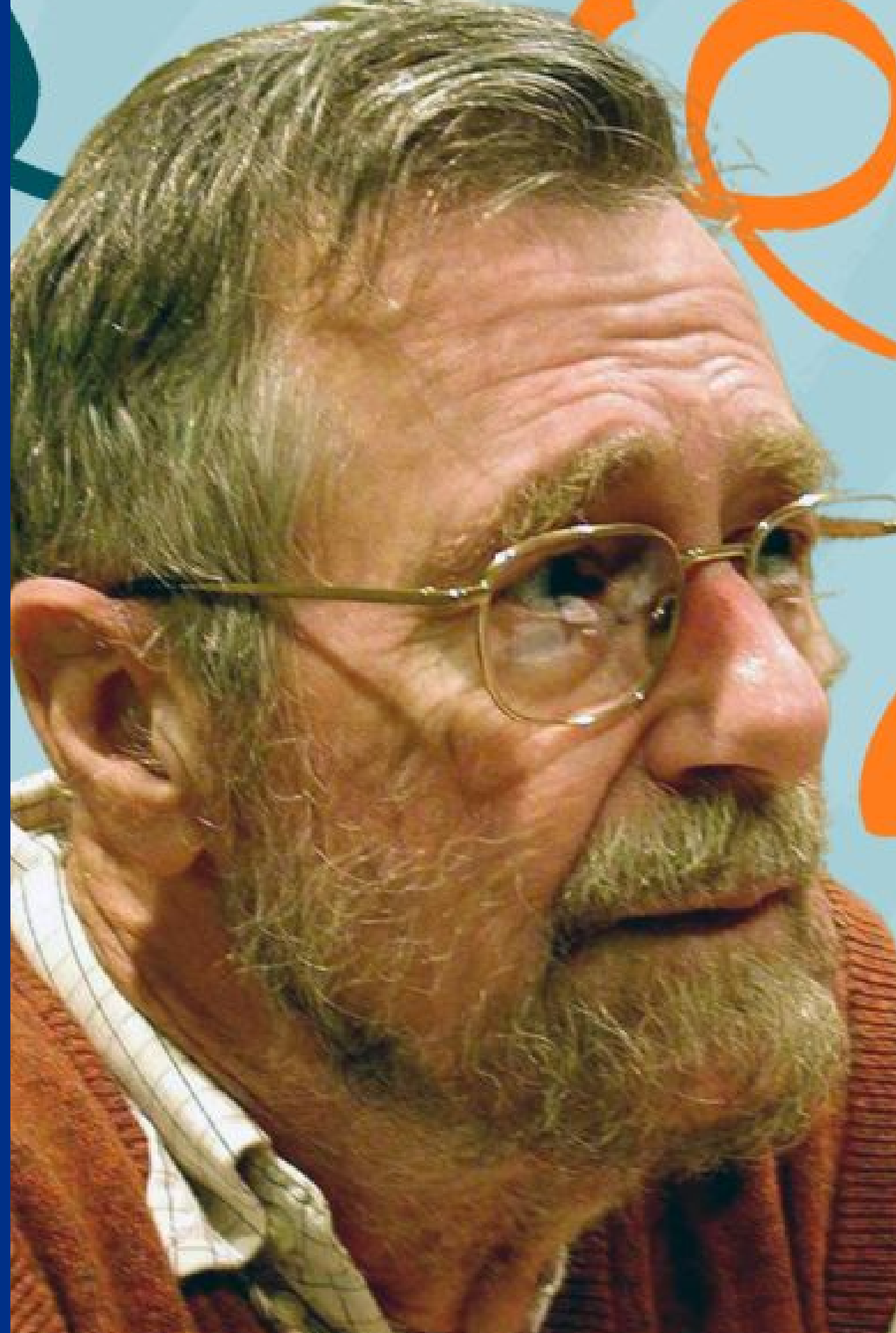
Caminos Cortos

UNIVERSIDAD DE CHILE

Arturo Merino



Plan

Recuerdo y plan

- **Recuerdo:**

Recuerdo y plan

- **Recuerdo:**
  - BFS + DFS

Recuerdo y plan

- **Recuerdo:**
  - BFS + DFS
  - Prim + Kruskal

Recuerdo y plan

- **Recuerdo:**
  - BFS + DFS
  - Prim + Kruskal

- **Plan:**

Recuerdo y plan

- **Recuerdo:**
  - BFS + DFS
  - Prim + Kruskal

- **Plan:**
  - caminos más cortos

Recuerdo y plan

- **Recuerdo:**
  - BFS + DFS
  - Prim + Kruskal

- **Plan:**
  - caminos más cortos
  - Dijkstra

Caminos más cortos

Camino más corto

- Común en ciencia de la computación y aplicaciones

Camino más corto

- Común en ciencia de la computación y aplicaciones

Def.

Un *digrafo con pesos positivos* es una terna $G = (V, E, w)$, donde (V, E) es un digrafo y $w \colon E \to \mathbb{R}_{>0}$ es un peso en cada arco.

Camino más corto

- Común en ciencia de la computación y aplicaciones

Def.

Un *digrafo con pesos positivos* es una terna $G = (V, E, w)$, donde (V, E) es un digrafo y $w\colon E \to \mathbb{R}_{>0}$ es un peso en cada arco.

Si $P = v_0, \ldots, v_K$ es un dicamino, su *largo* o *costo* es

$$w(P) = \sum_{i=1}^{k} w(v_{i-1}v_i)$$

Ejemplo:

Camino más corto

- Común en ciencia de la computación y aplicaciones

Def.

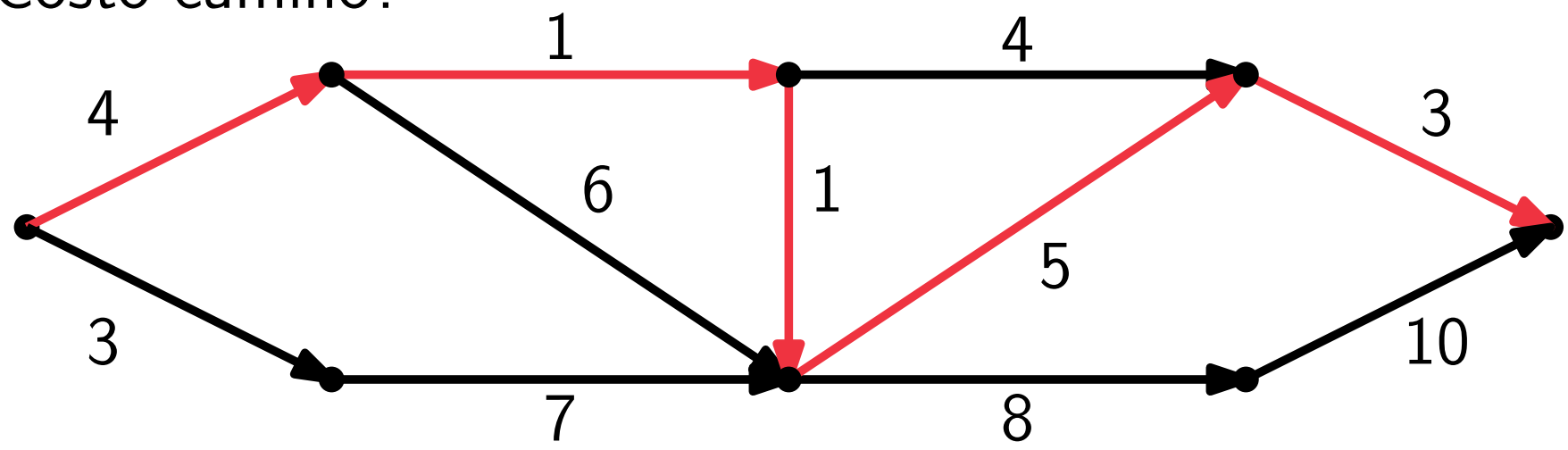
Un *digrafo con pesos positivos* es una terna $G = (V, E, w)$, donde (V, E) es un digrafo y $w\colon E \to \mathbb{R}_{>0}$ es un peso en cada arco.

Si $P = v_0, \ldots, v_K$ es un dicamino, su *largo* o *costo* es

$$w(P) = \sum_{i=1}^{k} w(v_{i-1}v_i)$$

Ejemplo:

- ¿Costo camino?



Distancia

- Distancia $\leftarrow$ camino de costo minimo

Distancia

- Distancia $\leftarrow$ camino de costo minimo

Def.

Sea $G = (V, E, w)$ un digrafo con pesos positivos y sean $u, v \in V$.

Distancia

- Distancia $\leftarrow$ camino de costo minimo

Def.

Sea $G = (V, E, w)$ un digrafo con pesos positivos y sean $u, v \in V$.

La *distancia* de u a v, denotada $d(u, v)$, es

$$d(u,v) = \begin{cases} \infty & \text{si no existe} \\ \min\{w(P) : P \text{ es un dicamino de } u \text{ a } v\} & \text{en otro caso.} \end{cases}$$

Ejemplo:

Distancia

- Distancia $\leftarrow$ camino de costo minimo

Def.

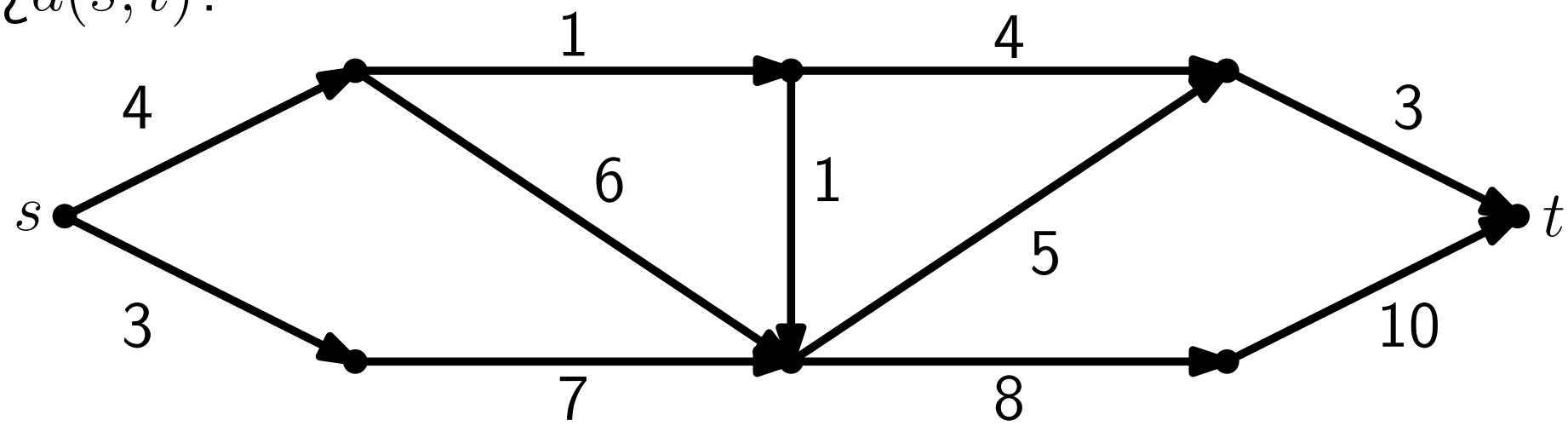
Sea $G = (V, E, w)$ un digrafo con pesos positivos y sean $u, v \in V$.

La *distancia* de u a v, denotada $d(u, v)$, es

$$d(u,v) = \begin{cases} \infty & \text{si no existe} \\ \min\{w(P) : P \text{ es un dicamino de } u \text{ a } v\} & \text{en otro caso.} \end{cases}$$

Ejemplo:

- ¿$d(s, t)$?



SSSP (Single-source shortest path)

- **SSSP:** encontrar todas las distancias a un vértice s.

SSSP (Single-source shortest path)

- **SSSP:** encontrar todas las distancias a un vértice s.
- Costo positivo $\rightarrow$ no repetición.

SSSP (Single-source shortest path)

- **SSSP:** encontrar todas las distancias a un vértice s.
- Costo positivo $\rightarrow$ no repetición.

Prop.

Sea $G = (V, E, w)$ un digrafo con pesos positivos.

SSSP (Single-source shortest path)

- **SSSP:** encontrar todas las distancias a un vértice s.
- Costo positivo $\rightarrow$ no repetición.

Prop.

Sea $G = (V, E, w)$ un digrafo con pesos positivos.

Si existe un dicamino de costo mínimo desde s hasta v, entonces existe uno que no repite vértices.

Idea.

SSSP (Single-source shortest path)

- **SSSP:** encontrar todas las distancias a un vértice s.
- Costo positivo $\rightarrow$ no repetición.

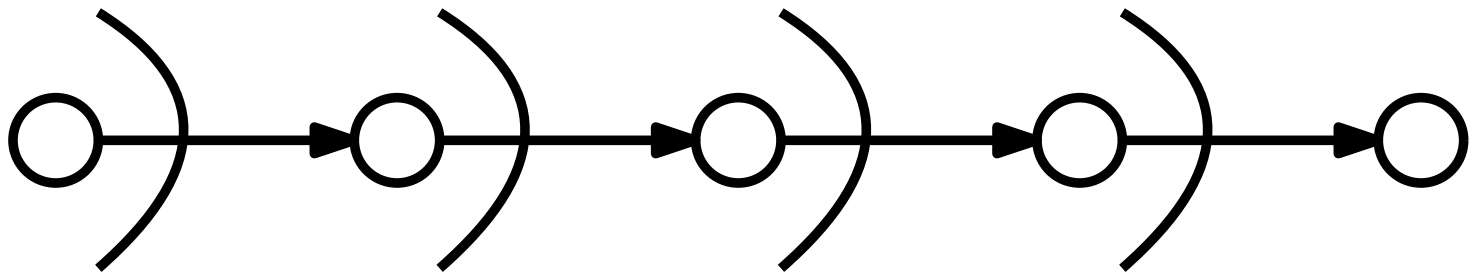
Prop.

Sea $G = (V, E, w)$ un digrafo con pesos positivos.

Si existe un dicamino de costo mínimo desde s hasta v, entonces existe uno que no repite vértices.

Idea.

Construir *iterativamente* los caminos más cortos



Vecinos de salida

- Agregar un vecino de salida a la vez

Vecinos de salida

- Agregar un vecino de salida a la vez

Def.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$.

Vecinos de salida

- Agregar un vecino de salida a la vez

Def.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$.

Los *vecinos de salida* de U son

$$N^+(U) = \{v \in V \setminus U : \text{existe } u \in U \text{ tal que } (u, v) \in E\}.$$

Vecinos de salida

- Agregar un vecino de salida a la vez

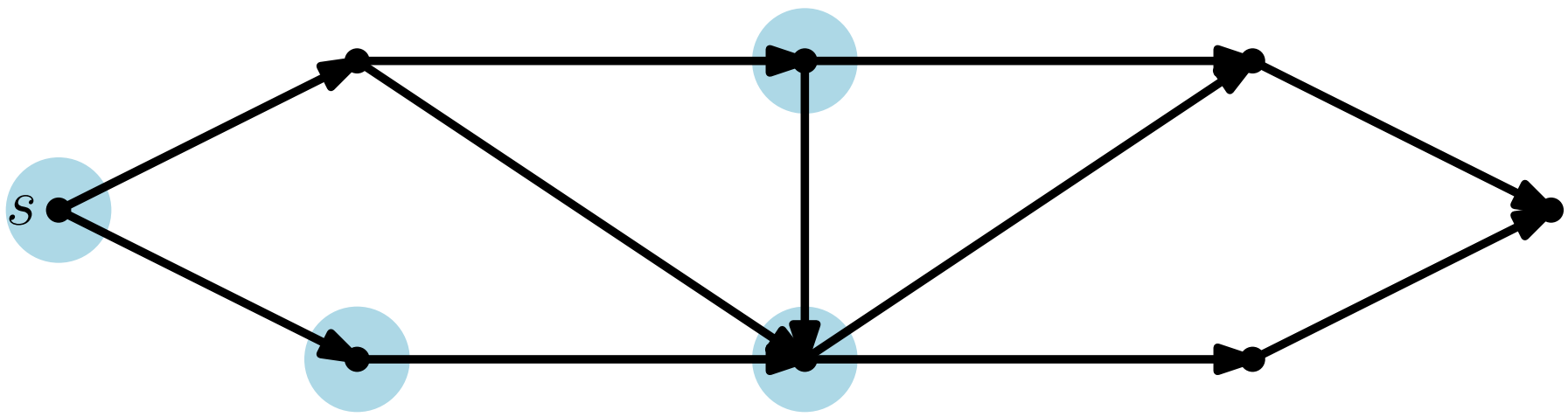
Def.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$.

Los *vecinos de salida* de U son

$$N^+(U) = \{v \in V \setminus U : \text{existe } u \in U \text{ tal que } (u, v) \in E\}.$$

- ¿$N^+(U)$?



Vecinos de salida

- Agregar un vecino de salida a la vez

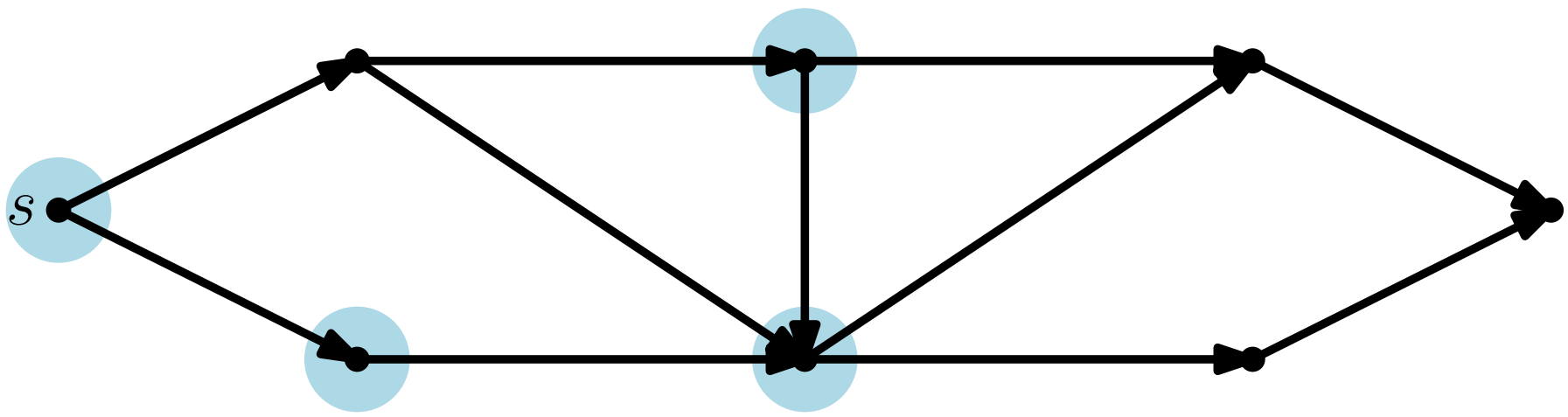
Def.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$.

Los *vecinos de salida* de U son

$$N^+(U) = \{v \in V \setminus U : \text{existe } u \in U \text{ tal que } (u, v) \in E\}.$$

- ¿$N^+(U)$?



- ¿Cómo elegir?

Elección avara

- Elijamos el vecino más cercano a s

Elección avara

- Elijamos el vecino más cercano a s

Def.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$.

Elección avara

- Elijamos el vecino más cercano a s

Def.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$.

Para $v \in N^+(U)$ definimos su *costo candidato desde U* como

$$c_U(v) = \min\{d(s,u) + w(u,v) : u \in U \text{ y } (u,v) \in E\}.$$

Elección avara

- Elijamos el vecino más cercano a s

Def.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$.

Para $v \in N^+(U)$ definimos su *costo candidato desde U* como

$$c_U(v) = \min\{d(s,u) + w(u,v) : u \in U \text{ y } (u,v) \in E\}.$$

Lem.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$ tal que $s \in U$

Elección avara

- Elijamos el vecino más cercano a s

Def.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$.

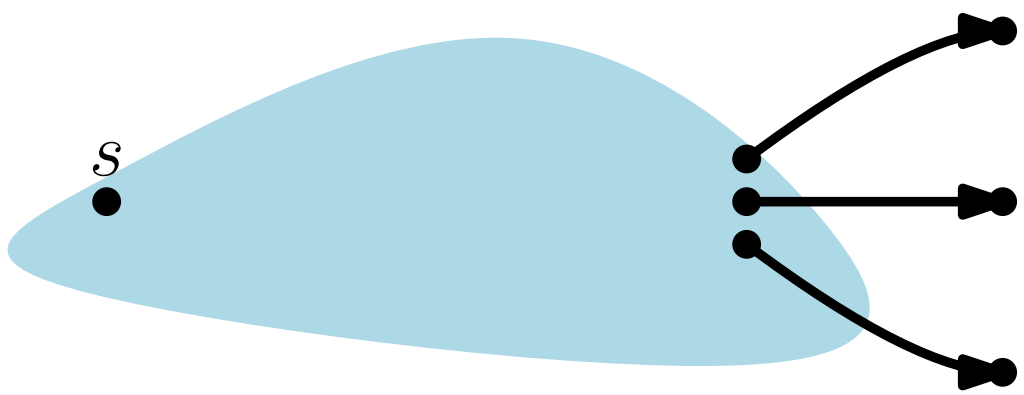
Para $v \in N^+(U)$ definimos su *costo candidato desde U* como

$$c_U(v) = \min\{d(s,u) + w(u,v) : u \in U \text{ y } (u,v) \in E\}.$$

Lem.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$ tal que $s \in U$

Si $v^\star \in N^+(U)$ minimiza $c_U(v)$ entre todos los vértices de $N^+(U)$, entonces $d(s, v^\star) = c_U(v^\star)$.

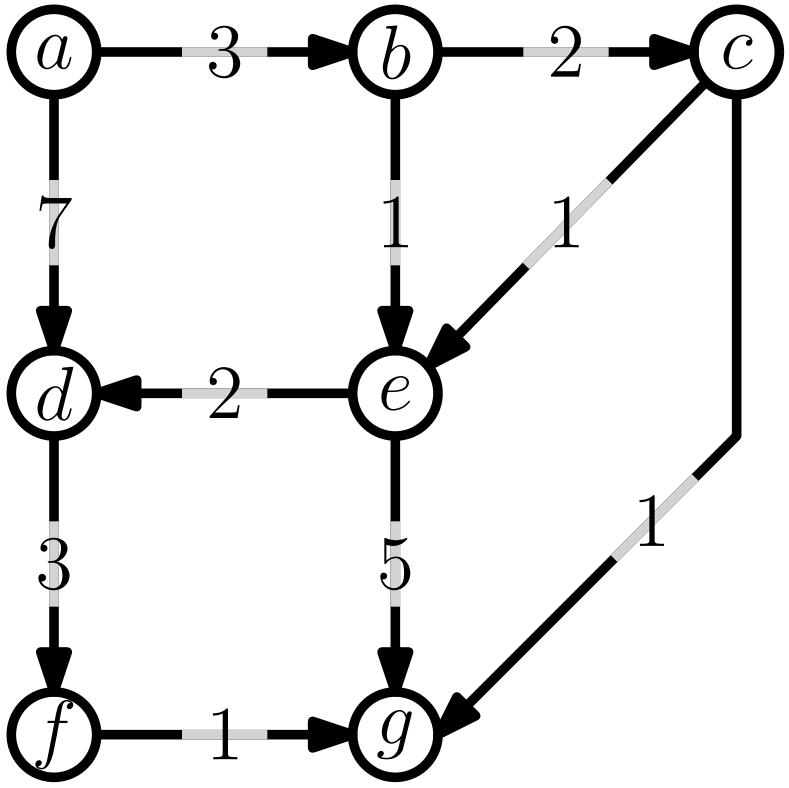


Algoritmo de Dijkstra

Lem.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$ tal que $s \in U$

Si $v^\star \in N^+(U)$ minimiza $c_U(v)$ entre todos los vértices de $N^+(U)$, entonces $d(s, v^\star) = c_U(v^\star)$.

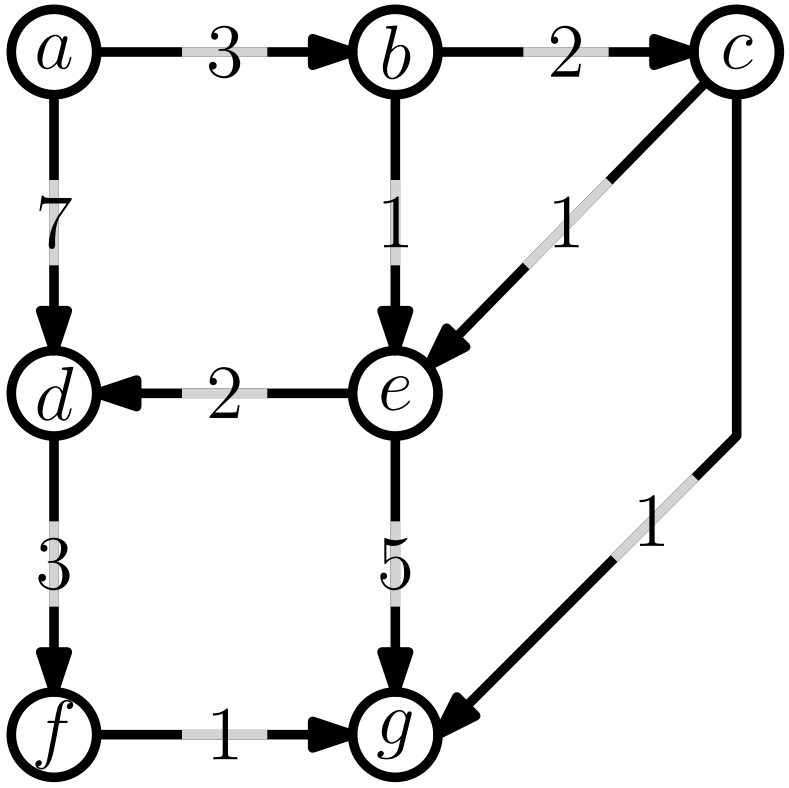


Algoritmo de Dijkstra

Lem.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$ tal que $s \in U$

Si $v^\star \in N^+(U)$ minimiza $c_U(v)$ entre todos los vértices de $N^+(U)$, entonces $d(s, v^\star) = c_U(v^\star)$.



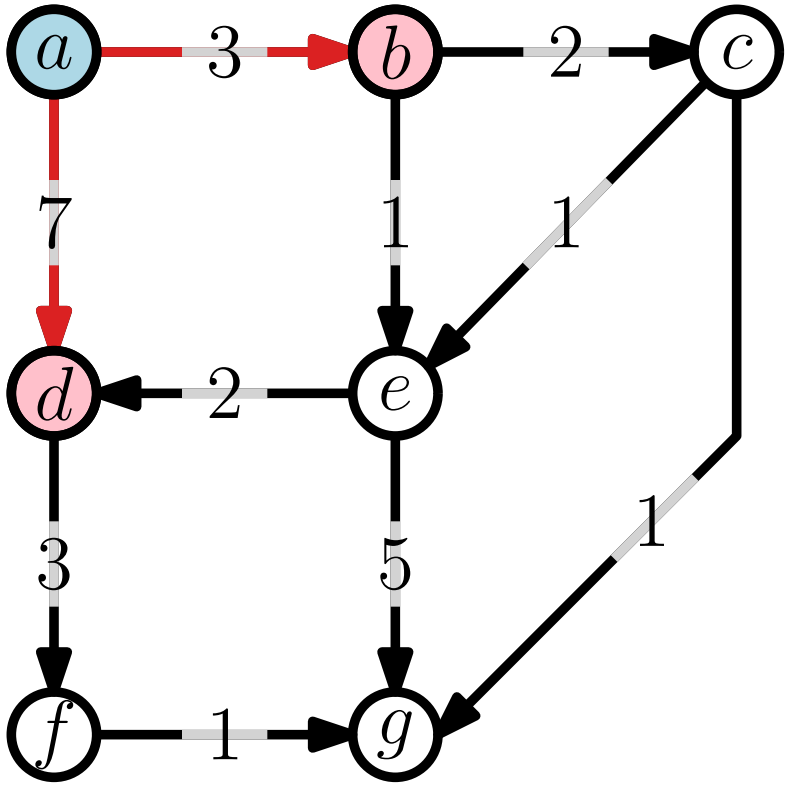
Entra a U	a	b	c	d	e	f	g

Algoritmo de Dijkstra

Lem.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$ tal que $s \in U$

Si $v^\star \in N^+(U)$ minimiza $c_U(v)$ entre todos los vértices de $N^+(U)$, entonces $d(s, v^\star) = c_U(v^\star)$.



Entra a U

a

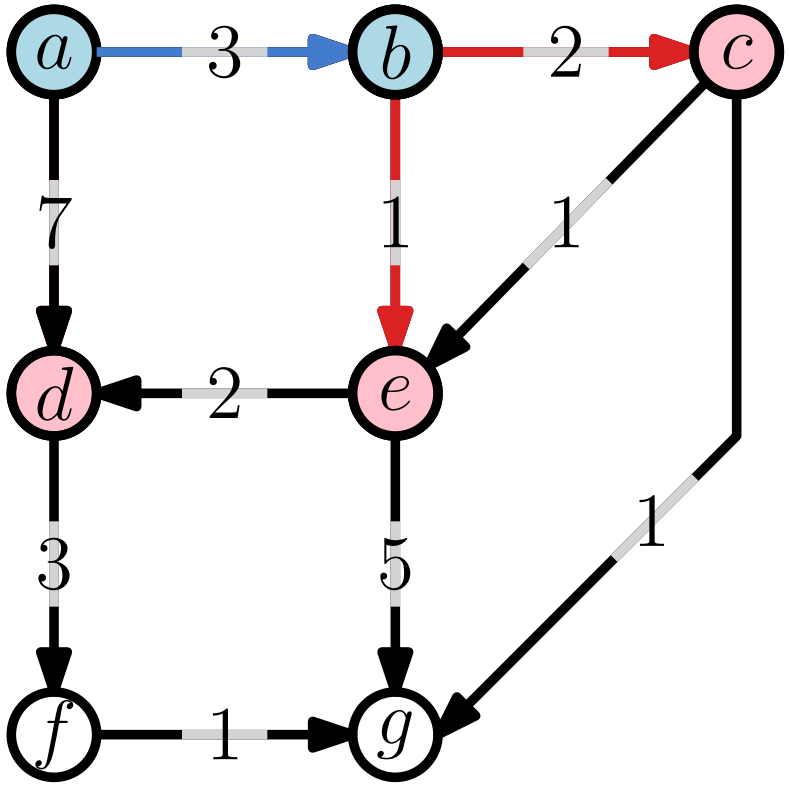
a	b	c	d	e	f	g
0	3	∞	7	∞	∞	∞

Algoritmo de Dijkstra

Lem.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$ tal que $s \in U$

Si $v^\star \in N^+(U)$ minimiza $c_U(v)$ entre todos los vértices de $N^+(U)$, entonces $d(s, v^\star) = c_U(v^\star)$.



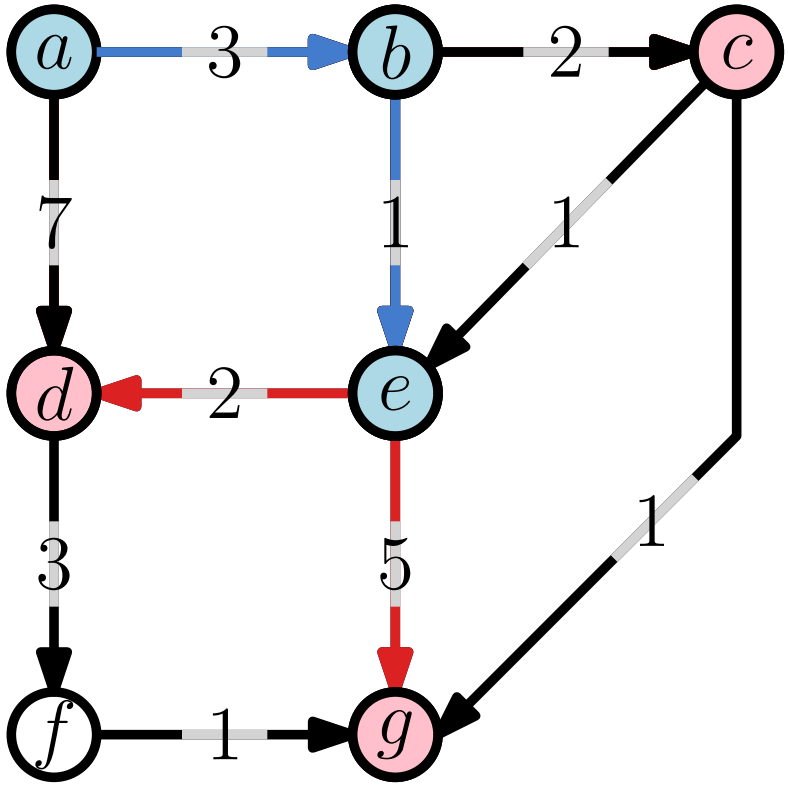
Entra a U	a	b	c	d	e	f	g
	0	3	∞	7	∞	∞	∞
a	0	3	5	7	4	∞	∞
b							

Algoritmo de Dijkstra

Lem.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$ tal que $s \in U$

Si $v^\star \in N^+(U)$ minimiza $c_U(v)$ entre todos los vértices de $N^+(U)$, entonces $d(s, v^\star) = c_U(v^\star)$.



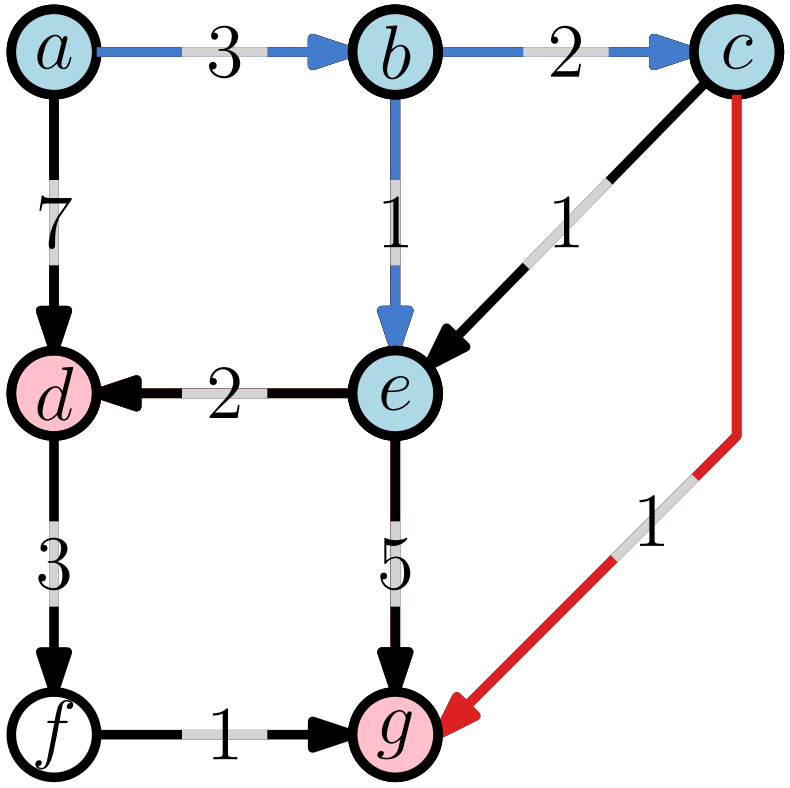
Entra a U	a	b	c	d	e	f	g
	0	3	∞	7	∞	∞	∞
a	0	3	5	7	4	∞	∞
b	0	3	5	6	4	∞	9
e							

Algoritmo de Dijkstra

Lem.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$ tal que $s \in U$

Si $v^\star \in N^+(U)$ minimiza $c_U(v)$ entre todos los vértices de $N^+(U)$, entonces $d(s, v^\star) = c_U(v^\star)$.



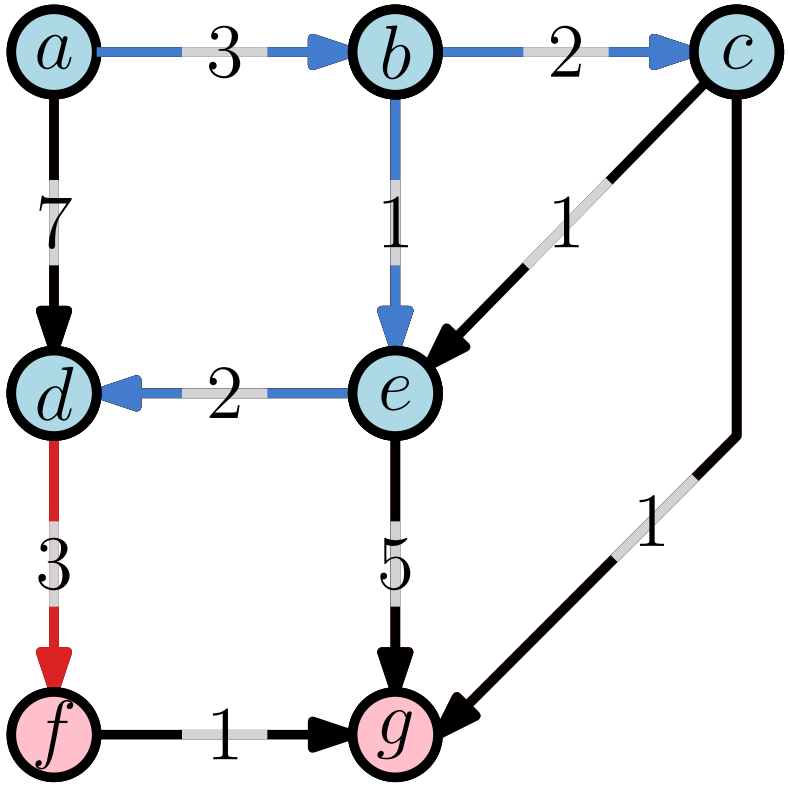
Entra a U	a	b	c	d	e	f	g
a	0	3	∞	7	∞	∞	∞
b	0	3	5	7	4	∞	∞
e	0	3	5	6	4	∞	9
c	0	3	5	6	4	∞	6

Algoritmo de Dijkstra

Lem.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$ tal que $s \in U$

Si $v^\star \in N^+(U)$ minimiza $c_U(v)$ entre todos los vértices de $N^+(U)$, entonces $d(s, v^\star) = c_U(v^\star)$.



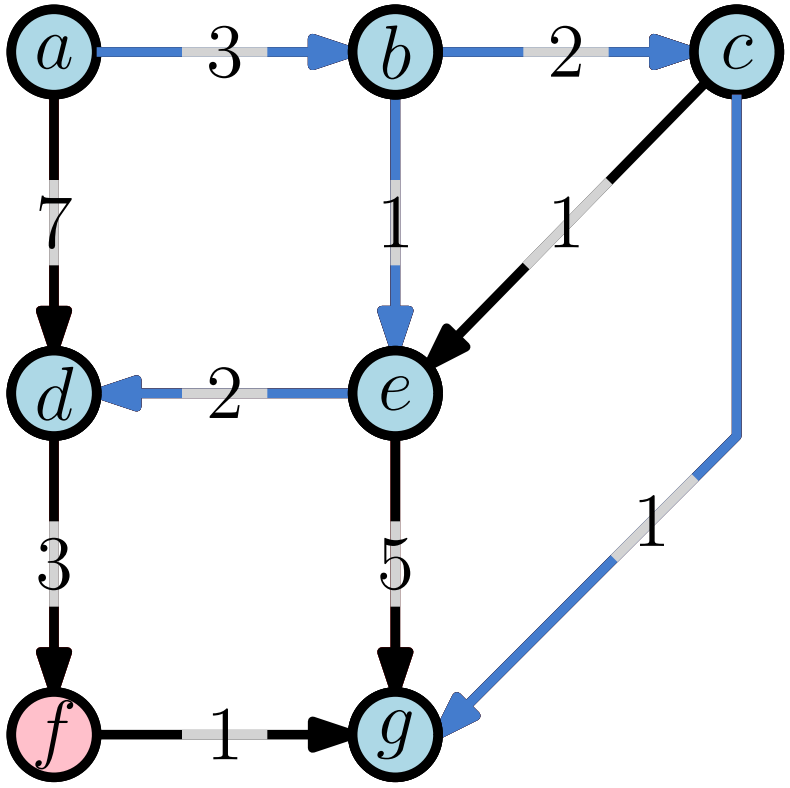
Entra a U	a	b	c	d	e	f	g
a	0	3	∞	7	∞	∞	∞
b	0	3	5	7	4	∞	∞
e	0	3	5	6	4	∞	9
c	0	3	5	6	4	∞	6
d	0	3	5	6	4	9	6

Algoritmo de Dijkstra

Lem.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$ tal que $s \in U$

Si $v^\star \in N^+(U)$ minimiza $c_U(v)$ entre todos los vértices de $N^+(U)$, entonces $d(s, v^\star) = c_U(v^\star)$.



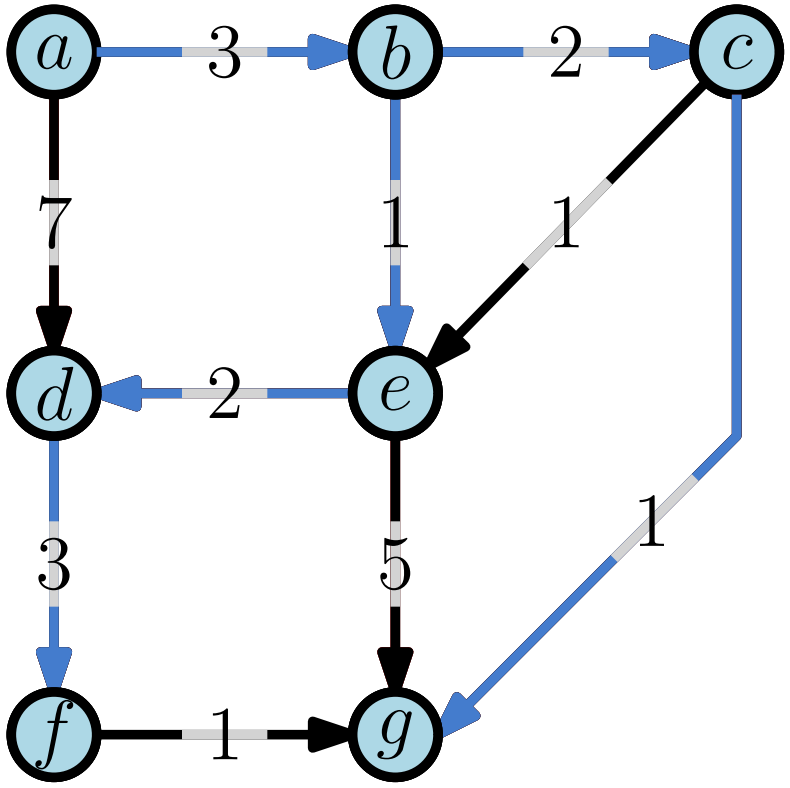
Entra a U	a	b	c	d	e	f	g
a	0	3	∞	7	∞	∞	∞
b	0	3	5	7	4	∞	∞
e	0	3	5	6	4	∞	9
c	0	3	5	6	4	∞	6
d	0	3	5	6	4	9	6
g	0	3	5	6	4	9	6

Algoritmo de Dijkstra

Lem.

Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$ tal que $s \in U$

Si $v^\star \in N^+(U)$ minimiza $c_U(v)$ entre todos los vértices de $N^+(U)$, entonces $d(s, v^\star) = c_U(v^\star)$.



Entra a U	a	b	c	d	e	f	g
a	0	3	∞	7	∞	∞	∞
b	0	3	5	7	4	∞	∞
e	0	3	5	6	4	∞	9
c	0	3	5	6	4	∞	6
d	0	3	5	6	4	9	6
g	0	3	5	6	4	9	6
f	0	3	5	6	4	9	6

Algoritmo de Dijkstra

- Para encontrar los caminos más cortos desde $s \in V$

Algoritmo de Dijkstra

- Para encontrar los caminos más cortos desde $s \in V$

Alg.

Partir con $U \leftarrow \{s\}$ y $\delta(s) = 0$.

Algoritmo de Dijkstra

- Para encontrar los caminos más cortos desde $s \in V$

Alg.

Partir con $U \leftarrow \{s\}$ y $\delta(s) = 0$.

while $N^+(U) \neq \emptyset$:

Algoritmo de Dijkstra

- Para encontrar los caminos más cortos desde $s \in V$

Alg.

Partir con $U \leftarrow \{s\}$ y $\delta(s) = 0$.

while $N^+(U) \neq \emptyset$:

Actualizar los costos de $v \in N^+(U)$ de la forma $\delta(v) \leftarrow c_U(v)$.

Algoritmo de Dijkstra

- Para encontrar los caminos más cortos desde $s \in V$

Alg.

Partir con $U \leftarrow \{s\}$ y $\delta(s) = 0$.

while $N^+(U) \neq \emptyset$:

Actualizar los costos de $v \in N^+(U)$ de la forma $\delta(v) \leftarrow c_U(v)$.

Agregar a U el vértice de $N^+(U)$ de menor costo.

Algoritmo de Dijkstra

- Para encontrar los caminos más cortos desde $s \in V$

Alg.

Partir con $U \leftarrow \{s\}$ y $\delta(s) = 0$.

while $N^+(U) \neq \emptyset$:

Actualizar los costos de $v \in N^+(U)$ de la forma $\delta(v) \leftarrow c_U(v)$.

Agregar a U el vértice de $N^+(U)$ de menor costo.

- Correcto por el lema anterior

Algoritmo de Dijkstra

- Para encontrar los caminos más cortos desde $s \in V$

Alg.

Partir con $U \leftarrow \{s\}$ y $\delta(s) = 0$.

while $N^+(U) \neq \emptyset$:

Actualizar los costos de $v \in N^+(U)$ de la forma $\delta(v) \leftarrow c_U(v)$.

Agregar a U el vértice de $N^+(U)$ de menor costo.

- Correcto por el lema anterior

Teo.

Sea $G = (V, E, w)$ un digrafo con pesos positivos y sea $s \in V$.

Algoritmo de Dijkstra

- Para encontrar los caminos más cortos desde $s \in V$

Alg.

Partir con $U \leftarrow \{s\}$ y $\delta(s) = 0$.

while $N^+(U) \neq \emptyset$:

Actualizar los costos de $v \in N^+(U)$ de la forma $\delta(v) \leftarrow c_U(v)$.

Agregar a U el vértice de $N^+(U)$ de menor costo.

- Correcto por el lema anterior

Teo.

Sea $G = (V, E, w)$ un digrafo con pesos positivos y sea $s \in V$.

El algoritmo de Dijkstra calcula correctamente $d(s, v)$ para todo vértice $v \in V$.

Algoritmo de Dijkstra

- Para encontrar los caminos más cortos desde $s \in V$

Alg.

Partir con $U \leftarrow \{s\}$ y $\delta(s) = 0$.

while $N^+(U) \neq \emptyset$:

Actualizar los costos de $v \in N^+(U)$ de la forma $\delta(v) \leftarrow c_U(v)$.

Agregar a U el vértice de $N^+(U)$ de menor costo.

- Correcto por el lema anterior

Teo.

Sea $G = (V, E, w)$ un digrafo con pesos positivos y sea $s \in V$.

El algoritmo de Dijkstra calcula correctamente $d(s, v)$ para todo vértice $v \in V$.

- ¿No alcanzables? ¿Loops? ¿Multigrafos? ¿Grafos no dirigidos?

Dudas/Preguntas