

Algoritmi e Strutture Dati

Lezione 26

28 novembre 2022

PROBLEMI "CAMMINI MINIMI"

$$G = (V, E) \quad \omega: E \rightarrow \mathbb{R}$$

- Cammino minimo tra due vertici
- Cammini minimi da un vertice a tutti gli altri
- Cammini minimi tra ogni coppia di vertici

PROBLEMI "CAMMINI MINIMI"

$$G = (V, E) \quad w: E \rightarrow \mathbb{R}$$

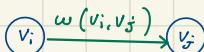
- Cammino minimo tra due vertici
- Cammini minimi da un vertice a tutti gli altri
- Cammini minimi tra ogni coppia di vertici

Algoritmo di Floyd & Marshall

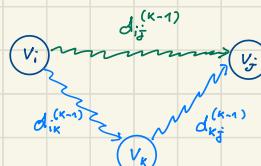
$$V = \{v_1, \dots, v_n\}$$

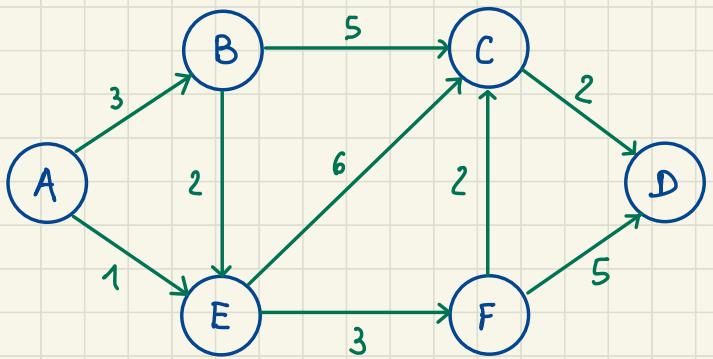
$d_{ij}^{(k)}$ = peso cammino minimo da v_i a v_j che passa
solo per vertici di indice $\leq k$

$$d_{ij}^{(0)} = \begin{cases} w(v_i, v_j) & \text{se } (v_i, v_j) \in E \text{ e } v_i \neq v_j \\ 0 & \text{se } v_i = v_j \\ \infty & \text{altrimenti} \end{cases}$$

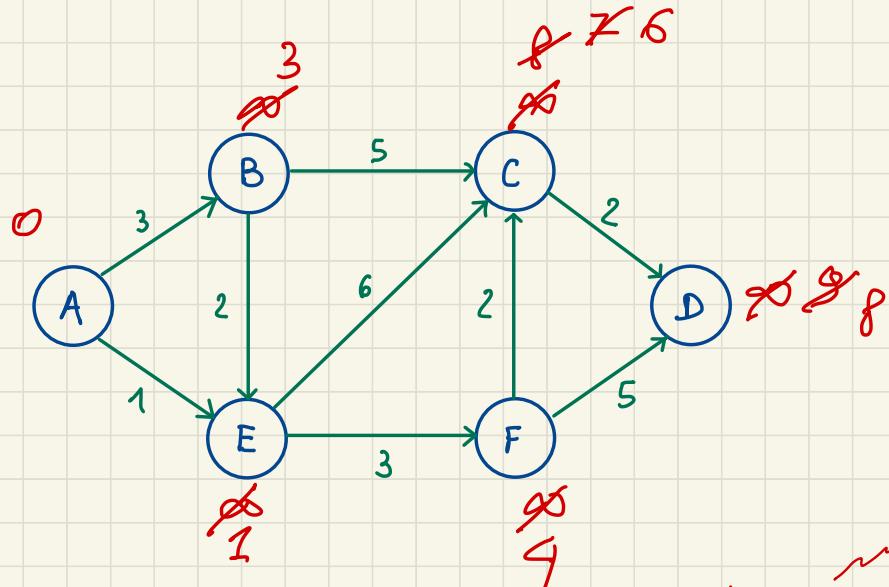


$$d_{ij}^{(k)} = \min \{ d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \}$$





L'ALGORITMO DI Bellman & Ford



$s = A$

ad ogni passo si seguono TUTTI i possibili archi

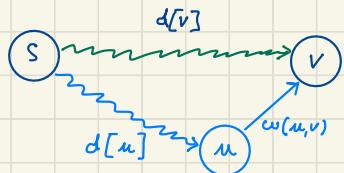
INPUT $G = (V, E)$ $w: E \rightarrow \mathbb{R}$
 $s \in V$ vertice di partenza

$d[v]$ = peso del cammino minimo da s a v sinora trovato

Inizialmente:

$$d[v] = \begin{cases} 0 & \text{se } v=s \\ \infty & \text{altrimenti} \end{cases}$$

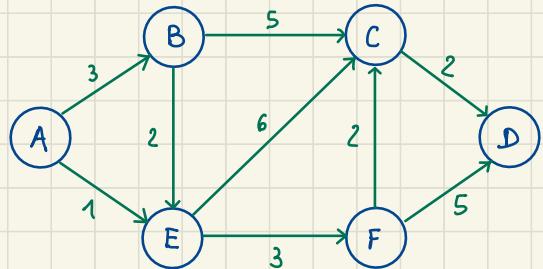
Aggiornamento:



$$d[u] + w(u,v) < d[v] ?$$

IF $d[u] + w(u,v) < d[v]$ THEN
 $| \quad d[v] \leftarrow d[u] + w(u,v)$

L'ALGORITMO DI Bellman & Ford



$s = A$

INPUT $G = (V, E)$ $w: E \rightarrow \mathbb{R}$
 $s \in V$ vertice di partenza

Ipotesi G privo di cicli negativi



Tra ogni coppia di vertici esiste
un cammino minimo semplice



E' sufficiente considerare cammini
composti da al più n vertici,
e dunque $n-1$ archi



In al più $n-1$ passi si
raggiungono tutti i nodi
ottenendo i pesi dei
cammini minimi

ALGORITMO Bellman & Ford (Grafo G , vertice s) \rightarrow Vettore

Sia $d[V]$ un vettore con indici in V

$$d[s] \leftarrow 0$$

FOR EACH $v \in V \setminus \{s\}$ DO $d[v] \leftarrow \infty$

FOR $K \leftarrow 1$ TO $n-1$ DO

FOR EACH $(u, v) \in E$ DO

IF $d[u] + w(u, v) < d[v]$ THEN

$$d[v] \leftarrow d[u] + w(u, v)$$

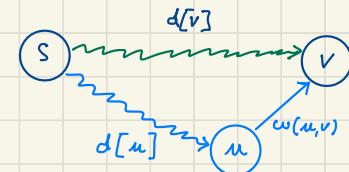
RETURN d

$d[v] = \text{peso del cammino minimo da } s \text{ a } v \text{ sinora trovato}$

Inizialmente:

$$d[v] = \begin{cases} 0 & \text{se } v=s \\ \infty & \text{altrimenti} \end{cases}$$

Aggiornamento:



$$d[u] + w(u, v) < d[v]?$$

ALGORITMO DI Bellman & Ford: Tempo

$$n = \# V \quad m = \# E$$

rappresentazione:

Lista d'archi:

tempo
 $O(n \cdot m)$

$n-1$ iterazioni

tempo
 $O(n \cdot m)$

ALGORITMO Bellman & Ford (Grafo G , vertice s) \rightarrow Vettore

tempo
 $O(n)$

Sia $d[V]$ un vettore con indici in V
 $d[s] \leftarrow 0$

FOR EACH $v \in V \setminus \{s\}$ DO $d[v] \leftarrow \infty$ n-1 iterazioni:

FOR $k \leftarrow 1$ TO $n-1$ DO

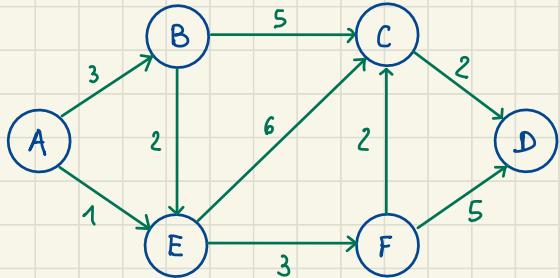
FOR EACH $(u, v) \in E$ DO

IF $d[u] + w(u, v) < d[v]$ THEN
 $d[v] \leftarrow d[u] + w(u, v)$

RETURN d

m iterazioni ogni volta

L'ALGORITMO DI Dijkstra



ad ogni passo si considerano gli archi che escono da un vertice u scelto con strategia GREEDY

distanze "provisorie"

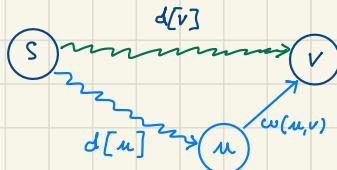
INPUT $G = (V, E)$ $w: E \rightarrow \mathbb{R}$
 $s \in V$ vertice di partenza

$d[v] =$ peso del cammino minimo da s a v sinora trovato

Inizialmente:

$$d[v] = \begin{cases} 0 & \text{se } v=s \\ \infty & \text{altrimenti} \end{cases}$$

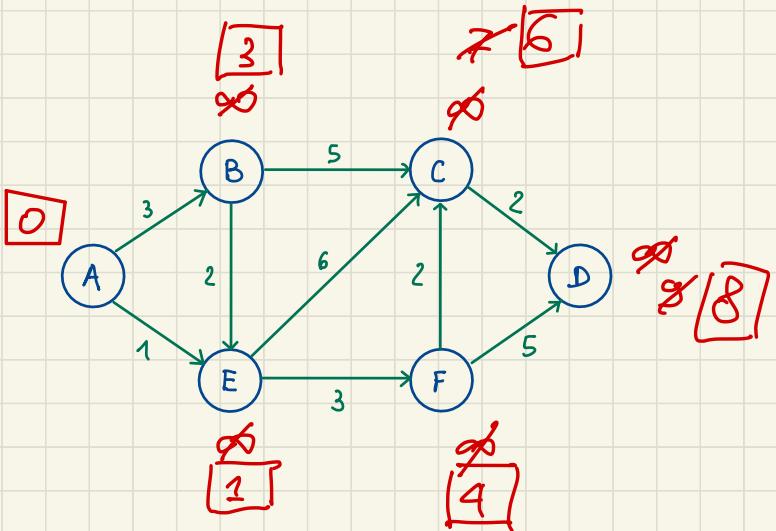
Aggiornamento:



$$d[u] + w(u,v) < d[v] ?$$

IF $d[u] + w(u,v) < d[v]$ THEN
 $| \quad d[v] \leftarrow d[u] + w(u,v)$

L'ALGORITMO DI Dijkstra



$s = A$

INPUT $G = (V, E)$ $w: E \rightarrow \mathbb{R}$
 $s \in V$ vertice di partenza

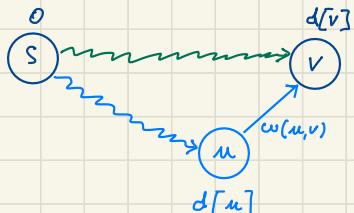
- Distanza provvisoria vettore $d[V]$

$$\text{initialmente } d[v] = \begin{cases} 0 & \text{se } v=s \\ \infty & \text{se } v \neq s \end{cases}$$

- $C \subseteq V$ insieme vertici candidati

initialmente $C = V$

- strategia "greedy"
 preleva da C il vertice u con $d[u]$ minima
 $d[u]$ diventa definitiva
 aggiorna $d[v]$ per ogni v adiacente a u



$$d[u] + w(u,v) < d[v]?$$

ALGORITMO Dijkstra (Grafo G , vertice s) \rightarrow Vettore

Sia $d[V]$ un vettore con indici in V

$$d[s] \leftarrow 0$$

FOR EACH $v \in V - \{s\}$ DO $d[v] \leftarrow \infty$

$$C \leftarrow V$$

WHILE $C \neq \emptyset$ DO

$u \leftarrow$ elemento di C con $d[u]$ minima

scelta
greedy

$$C \leftarrow C - \{u\}$$

u fissato

FOR EACH $(u, v) \in E$ DO

IF $d[u] + w(u, v) < d[v]$ THEN

$$d[v] \leftarrow d[u] + w(u, v)$$

RETURN d

- Distanza provvisoria vettore $d[V]$

$$\text{initialmente } d[v] = \begin{cases} 0 & \text{se } v=s \\ \infty & \text{se } v \neq s \end{cases}$$

- $C \subseteq V$ insieme vertici candidati

initialmente $C = V$

- strategia "greedy"

preleva da C il vertice u con $d[u]$ minima

$d[u]$ diventa definitiva

aggiorna $d[v]$ per ogni v adiacente a u



$$d[u] + w(u, v) < d[v]?$$

ALGORITMO di DIJKSTRA: correttezza

→ Trova senza la soluzione corretta

Si può dimostrare (induzione sul numero di iterazioni) che ogni volta che viene valutata la condizione del ciclo WHILE si ha:

(a) $\forall v \in C$:

$d[v]$ = lunghezza del cammino più corto da s a v che, prima di raggiungere v , visita solo vertici non in C

(b) $\forall v \notin C$:

$d[v]$ = lunghezza del cammino più corto da s a v

ALGORITMO Dijkstra (Grafo G , vertice s) → Vettore

Sia $d[V]$ un vettore con indici in V

$d[s] \leftarrow 0$

FOR EACH $v \in V \setminus \{s\}$ DO $d[v] \leftarrow \infty$

$C \leftarrow V$

WHILE $C \neq \emptyset$ DO

$u \leftarrow$ elemento di C con $d[u]$ minima

$C \leftarrow C \setminus \{u\}$

FOR EACH $(u, v) \in E$ DO

IF $d[u] + w(u, v) < d[v]$ THEN

$d[v] \leftarrow d[u] + w(u, v)$

RETURN d

ALGORITMO Dijkstra (Grafo G, vertice s) \rightarrow Vettore

Sia $d[V]$ un vettore con indici in V

$$d[s] \leftarrow 0$$

FOR EACH $v \in V \setminus \{s\}$ DO $d[v] \leftarrow \infty$

Sia C una coda con priorità wofar

FOR EACH $v \in V$ DO $C.insert(v, d[v])$

WHILE $C \neq \emptyset$ DO

$m \leftarrow C.deleteMin()$

FOR EACH $(u, v) \in E$ DO

IF $d[u] + w(u, v) < d[v]$ THEN

$d[v] \leftarrow d[u] + w(u, v)$

$C.changeKey(v, d[v])$

RETURN d

IMPLEMENTAZIONE CON CODE CON PRIORITA'

ALGORITMO Dijkstra (Grafo G, vertice s) \rightarrow Vettore

Sia $d[V]$ un vettore con indici in V

$$d[s] \leftarrow 0$$

FOR EACH $v \in V \setminus \{s\}$ DO $d[v] \leftarrow \infty$

$C \leftarrow V$

WHILE $C \neq \emptyset$ DO

$m \leftarrow \text{elemento di } C \text{ con } d[m] \text{ minima}$

$C \leftarrow C \setminus \{m\}$

FOR EACH $(u, v) \in E$ DO

IF $d[u] + w(u, v) < d[v]$ THEN

$d[v] \leftarrow d[u] + w(u, v)$

RETURN d



ALGORITMO di DIJKSTRA: tempo di calcolo

1 inizializzazione

2

3 riempimento vettore pesi distanze

2 ciclo

4 definizione n iterazioni
 $O(n \log n)$ ciascuna

5 in Orafo m iterazioni

m pesi per trovare tutti gli archi

6 m volte in cui

al più m changeKey

$O(\log n)$ ciascuna

tempo

$O(n)$

$O(n)$

$O(n \log n)$

n iterazioni

$O(m \log n)$

$$\frac{O(n \log n) + O(m \log n)}{O(n \log n) + O(m \log n)} = O(m \log n)$$

grafo \leftrightarrow lista di adiacenze,
 coda con priorità \rightarrow Min-Heap
 vettore pesi distanze

ALGORITMO Dijkstra (Grafo G, vertice s) \rightarrow Vettore

Sia d[V] un vettore con indici in V

$d[s] \leftarrow 0$

FOR EACH $v \in V - \{s\}$ DO $d[v] \leftarrow \infty$

Sia C una coda con priorità vuota

FOR EACH $v \in V$ DO C.insert(v, d[v])

WHILE $C \neq \emptyset$ DO

$u \leftarrow C.\text{deletemin}()$

FOR EACH $(u, v) \in E$ DO

IF $d[u] + w(u, v) < d[v]$ THEN

$d[v] \leftarrow d[u] + w(u, v)$

C.changeKey(v, d[v])

RETURN d

grafo连通 $n-1 \leq m \leq n^2$

Tempo $O(m \log n)$:

- grafo: lista di adiacenze
 - coda con priorità: MinHeap - vertice prioritario
 - coda con priorità: Heap di Fibonacci
- Tempo $O(m + n \log n)$

ALGORITMO di DIJKSTRA: come ricavare i cammini minimi?

vettore dei predecessori: $\text{pred}[v]$

ALGORITMO Dijkstra (Grafo G , vertice s) \rightarrow Vettore

Sia $d[v]$ un vettore con indici in V

Sia $\text{pred}[v]$ un vettore con indici in V

$d[s] \leftarrow 0$

FOR EACH $v \in V - \{s\}$ DO $d[v] \leftarrow \infty$

Sia C una coda con priorità wofar

FOR EACH $v \in V$ DO $C.\text{insert}(v, d[v])$

WHILE $C \neq \emptyset$ DO

$u \leftarrow C.\text{deleteMin}()$

FOR EACH $(u, v) \in E$ DO

IF $d[u] + w(u, v) < d[v]$ THEN

$d[v] \leftarrow d[u] + w(u, v)$

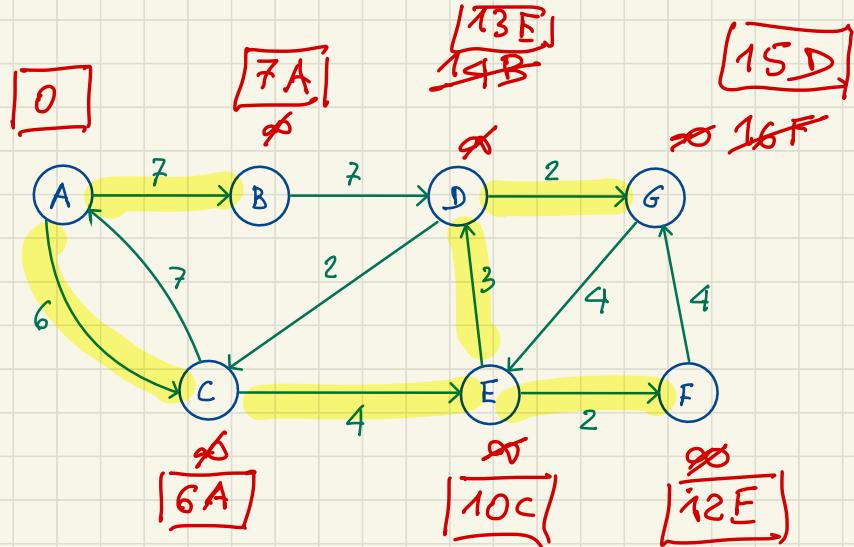
$C.\text{changeKey}(v, d[v])$

$\text{pred}[v] \leftarrow u$

RETURN d

ALGORITMO di DIJKSTRA: come ricavare i cammini minimi?

vettore dei predecessori: pred [v]



Sí A

Algores (con radice A)

dei Cammini minimi

ALGORITMO Dijkstra (Grafo G , vertice s) \rightarrow Vettore

Sia $d[V]$ un vettore con indici in V

Sia $\text{pred}[V]$ un vettore con indici in V

$d[s] \leftarrow 0$

FOR EACH $v \in V - \{s\}$ DO $d[v] \leftarrow \infty$

Sia C una coda con priorità wfa

FOR EACH $v \in V$ DO $C.insert(v, d[v])$

WHILE $C \neq \emptyset$ DO

$n \leftarrow C.\text{deleteMin}()$

FOR EACH $(u, v) \in E$ DO

IF $d[u] + \omega(u, v) < d[v]$ THEN

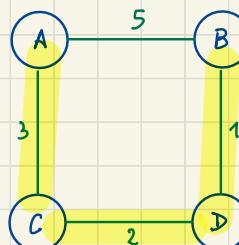
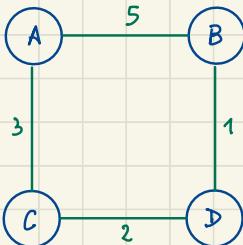
$$d[v] \leftarrow d[u] + \omega(u, v)$$

C.changeKey(v, d[v])

$\text{pred}[v] \leftarrow u$

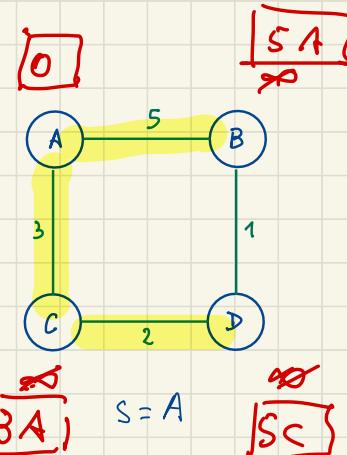
RETURN d

GRAFI NON ORIENTATI: albero ricoprente minimo vs
albero dei cammini minimi



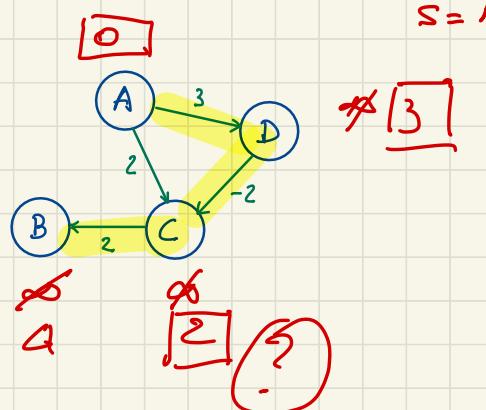
Albero ricoprente
minimo

peso 6



Albero dei
cammini minimi da A
peso 10

PESI NEGATIVI



$$s = A$$

~~3~~

cammino minimo 3

D. jkrrr A → C → B 4

sol pesi non negativi!

PROBLEMI "CAMMINI MINIMI"

- Cammino minimo tra due vertici

NO algoritmo diretto

- Cammini minimi da un vertice a tutti gli altri

Dijkstra

no pesi negativi

Tempo $O(m + n \lg n)$ o $O(m \lg n)$

- Cammini minimi tra ogni coppia di vertici

Floyd & Warshall

Tempo $O(n^3)$

Bellman Ford

$O(mn)$

anche pesi negativi

no cicli

negativi

anche pesi negativi

no cicli negativi