

Grafi, cammini minimi in grafi pesati e orientati

Corso di **Algoritmi e strutture dati**

Corso di Laurea in **Informatica**

Docenti: Ugo de'Liguoro, András Horváth

Indice

1. Definizione
2. Algoritmo di Dijkstra
3. Correttezza dell'algoritmo di Dijkstra

Sommario

Obiettivo:

- ▶ capire il concetto “cammino minimo”
- ▶ sviluppare algoritmi per trovare cammini minimi da un singolo sorgente

1. Cammini minimi

- ▶ sia dato un un grafo orientato e pesato
- ▶ distanza di un vertice u da un vertice v : $\delta(v, u)$, il peso di un cammino di peso minimo tra tutti i cammini da v a u
- ▶ $\delta(v, u) = \min\{W(p) | p \text{ è un cammino da } v \text{ a } u\}$ dove $W(p)$ è la somma dei pesi degli archi che formano il cammino
- ▶ $\delta(v, u)$ è ben definito solo se nessun cammino da v ad u contiene un ciclo di peso negativo

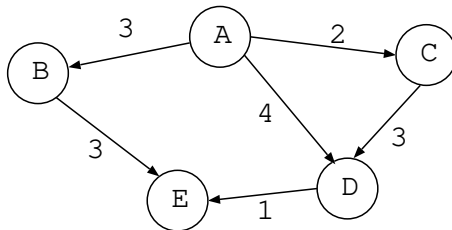
2. Algoritmo per trovare cammini minimi da un dato nodo

- ▶ input:
 - ▶ grafo orientato e pesato
 - ▶ un nodo (sorgente)
- ▶ output: $\forall v \in V(G)$ l'attributo $v.d$ indica la distanza di v dal vertice sorgente
- ▶ l'attributo $v.d$ mantiene una stima (maggiore o uguale) della distanza di v da s

2. L'idea dell'algoritmo

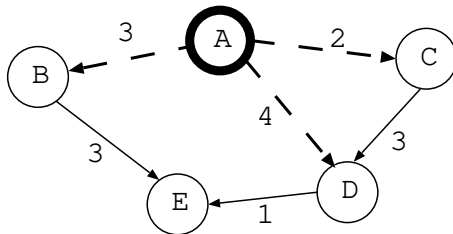
- ▶ inizialmente: $s.d=0, \forall v \in V(G), v \neq s : v.d=\infty$
- ▶ si costruisce un albero, di radice s , in cui viene inserito un vertice per volta
- ▶ l'albero è memorizzato implicitamente come l'insieme dei suoi archi $(v.\pi, v)$
- ▶ quando un vertice u è inserito nell'albero, si aggiornano le stime delle distanze dei vertici v ad esso adiacenti, in quanto potrebbe esistere un cammino da s a v , attraverso il vertice u , meno pesante del cammino da s a v considerato fino a quel momento

2. Un esempio



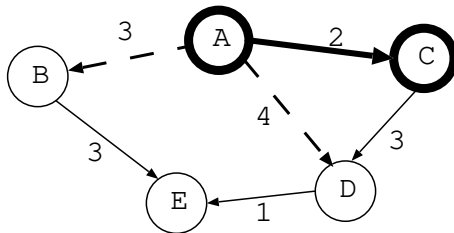
- ▶ nodo di partenza: A
- ▶ distanze: $A.d=0$, $B.d=\infty$, $C.d=\infty$, $D.d=\infty$, $E.d=\infty$
- ▶ da scegliere: A
- ▶ nuove distanze: $A.d=0$, $B.d=3$, $C.d=2$, $D.d=4$, $E.d=\infty$

2. Un esempio



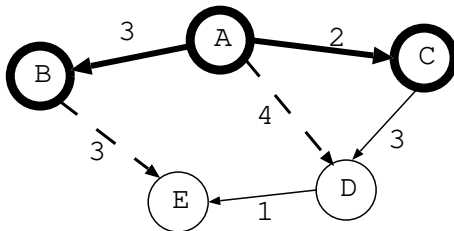
- ▶ distanze: $A.d=0$, $B.d=3$, $C.d=2$, $D.d=4$, $E.d=\infty$
- ▶ da scegliere: C
- ▶ nuove distanze: $A.d=0$, $B.d=3$, $C.d=2$, $D.d=4$, $E.d=\infty$

2. Un esempio



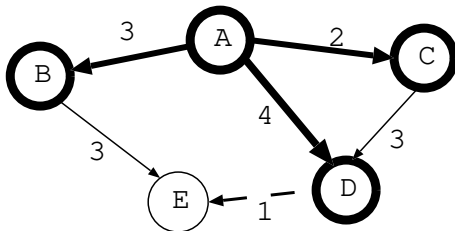
- ▶ distanze: $A.d=0$, $B.d=3$, $C.d=2$, $D.d=4$, $E.d=\infty$
- ▶ da scegliere: B
- ▶ nuove distanze: $A.d=0$, $B.d=3$, $C.d=2$, $D.d=4$, $E.d=6$

2. Un esempio



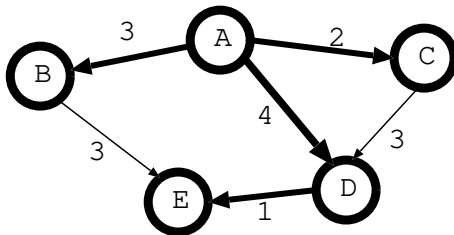
- ▶ distanze: $A.d=0$, $B.d=3$, $C.d=2$, $D.d=4$, $E.d=6$
- ▶ da scegliere: D
- ▶ nuove distanze: $A.d=0$, $B.d=3$, $C.d=2$, $D.d=4$, $E.d=5$

2. Un esempio



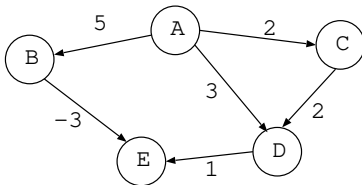
- ▶ distanze: $A.d=0$, $B.d=3$, $C.d=2$, $D.d=4$, $E.d=5$
- ▶ da scegliere: E
- ▶ nuove distanze: $A.d=0$, $B.d=3$, $C.d=2$, $D.d=4$, $E.d=5$

2. Un esempio



► distanze: $A.d=0$, $B.d=3$, $C.d=2$, $D.d=4$, $E.d=5$

2. L'idea non funziona con pesi negativi



- ▶ partendo da A
- ▶ $A.d=0, B.d=\infty, C.d=\infty, D.d=\infty, E.d=\infty$: $\rightarrow A$
- ▶ $A.d=0, B.d=5, C.d=2, D.d=3, E.d=\infty$: $\rightarrow C$
- ▶ $A.d=0, B.d=5, C.d=2, D.d=3, E.d=\infty$: $\rightarrow D$
- ▶ $A.d=0, B.d=5, C.d=2, D.d=3, E.d=4$: $\rightarrow E$
- ▶ $A.d=0, B.d=5, C.d=2, D.d=3, E.d=4$: $\rightarrow B$

2. Algoritmo di Dijkstra

- ▶ applica l'idea vista in precedenza
- ▶ funzione se tutti i pesi sono maggiori o uguali a 0

2. Algoritmo di Dijkstra

Dijkstra(G, s)

$Q \leftarrow V$

for $\forall v \in V$ **do** $v.d \leftarrow \infty, v.\pi \leftarrow nil$

$s.d \leftarrow 0$

$s.\pi \leftarrow nil$

while $Q \neq \emptyset$ **do**

$u \leftarrow$ toglì nodo con d minimo da Q

for $\forall v \in adj[u]$ **do**

if $v \in Q$ e $u.d + W(u, v) < v.d$ **then**

$v.d \leftarrow u.d + W(u, v)$

$v.\pi \leftarrow u$

2. Complessità dell'algoritmo di Dijkstra

- l'algoritmo di Dijkstra è molto simile a quello di Prim

MST_Prim(G, s)

$Q \leftarrow V$

for $\forall v \in V$ **do** $v.d \leftarrow \infty, v.\pi \leftarrow nil$

$s.d \leftarrow 0$

$s.\pi \leftarrow nil$

while $Q \neq \emptyset$ **do**

$u \leftarrow$ toglì nodo con d minimo da Q

for $\forall v \in adj[u]$ **do**

if $v \in Q$ e $\cancel{u.d} + W(u, v) < v.d$ **then**

$v.d \leftarrow \cancel{u.d} + W(u, v)$

$v.\pi \leftarrow u$

- complessità dell'algoritmo di Dijkstra è uguale a quella di Prim

3. Correttezza dell'algoritmo

- ▶ proprietà I: un sottocammino di un cammino minimo è minimo
- ▶ dimostrazione:
 - ▶ siano x e y due vertici qualunque in un cammino minimo p da u a v :
 $p = u \rightsquigarrow_{p_1} x \rightsquigarrow_{p_2} y \rightsquigarrow_{p_3} v = p_1 p_2 p_3$
 - ▶ $W(p) = W(p_1) + W(p_2) + W(p_3)$
 - ▶ se il sottocammino p_2 da x a y non fosse minimo, ne esisterebbe un altro p'_2 di peso inferiore
 - ▶ in tal caso il cammino $p' = p_1 p'_2 p_3$ sarebbe un cammino da u a v con $W(p') < W(p)$
 - ▶ allora p non è un cammino minimo, assurdo

3. Correttezza dell'algoritmo

- ▶ proprietà II: invarianti del ciclo:
 1. $\forall v \in V(G) : v \notin Q \Rightarrow v.d$ non viene modificato
 2. $\forall v \in Q : v.\pi \neq nil \Rightarrow v.\pi \notin Q$
 3. $\forall v \in V(G) - \{s\} : v.d \neq \infty \Leftrightarrow v.\pi \neq nil$
 4. $\forall v \in V(G) - \{s\} : v.d \neq \infty \Rightarrow v.d = v.\pi.d + W(v.\pi, v)$
- ▶ sono ovvii esaminando il ciclo dell'algoritmo

3. Correttezza dell'algoritmo

- ▶ proprietà III: invariante del ciclo: $\forall v \notin Q : v.d \neq \infty \Leftrightarrow$ esiste un cammino da s a v in G
- ▶ dimostrazione:
 - ▶ \Rightarrow :
 - ▶ inizializzazione: $Q = V$, non c'è nessun nodo che non fa parte di Q quindi è vero
 - ▶ mantenimento: supponiamo che l'asserzione sia vera un certo punto dell'esecuzione per ogni nodo che non fa parte di Q
 - ▶ dimostriamo che, se per il vertice u estratto dalla coda $u.d \neq \infty$, allora il cammino esiste
 - ▶ per ipotesi esiste un cammino da s a $u.\pi$
 - ▶ allora il cammino da s a $u.\pi$ più l'arco $(u.\pi, u)$ costituisce un cammino da s a u

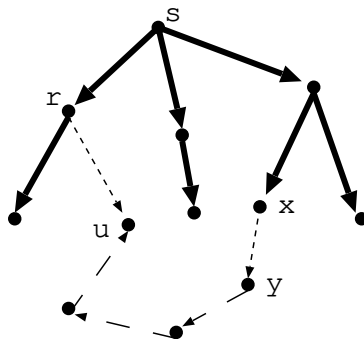
3. Correttezza dell'algoritmo

- ▶ proprietà III: invariante del ciclo: $\forall v \notin Q : v.d \neq \infty \Leftrightarrow$ esiste un cammino da s a v in G
- ▶ dimostrazione:
 - ▶ \Leftarrow :
 - ▶ se u viene estratto da Q con $u.d = \infty$, allora tutti i vertici $t \in Q$ hanno $t.d = \infty$
 - ▶ supponiamo che tra s e u vi sia almeno un cammino:
 $s \rightarrow v_1 \rightarrow v_2 \rightarrow \dots v_{k-1} \rightarrow v_k = u$
 - ▶ allora, tutti i vertici v_i sul cammino devono avere $v_i.d = \infty$
 - ▶ (perché se v_i avesse $v_i.d \neq \infty$, allora $v_i \notin Q$, quindi anche v_{i+1} avrebbe $v_{i+1}.d \neq \infty$)
 - ▶ ma questo è assurdo perché $s.d = 0$

3. Correttezza dell'algoritmo

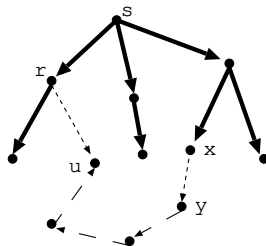
- ▶ invariante principale del ciclo: $\forall t \notin Q : t.d = \delta(s, t)$
- ▶ dimostrazione:
- ▶ il predicato è vero all'inizio poichè $Q = V(G)$
- ▶ supponiamo che sia vero quando l'albero è stato costruito parzialmente
- ▶ dimostriamo che per il nuovo vertice u estratto da Q , il predicato verrà mantenuto

3. Correttezza dell'algoritmo



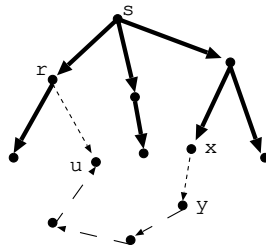
3. Correttezza dell'algoritmo

- ▶ caso I: $u.d \neq \infty$
 - ▶ sia $u.\pi = r$ (proprietà 2.3)
 - ▶ sappiamo allora (proprietà 2.2) che $r \notin Q$
 - ▶ $u.d = r.d + W(r, u)$ (proprietà 2.4)
 - ▶ supponiamo che tra s e u esista un cammino di peso minore di $u.d$
 - ▶ esso deve contenere un arco tra un vertice in $V(G) - Q$ e uno in Q : (x, y)



3. Correttezza dell'algoritmo

- ▶ caso I: $u.d \neq \infty$
 - ▶ cammino tra s e u può allora essere visto come la concatenazione di tre cammini: $s \rightsquigarrow x \rightarrow y \rightsquigarrow u$
 - ▶ se $s \rightsquigarrow x \rightarrow y \rightsquigarrow u$ è minimo, allora anche $s \rightsquigarrow x \rightarrow y$ è minimo (proprietà 1)
 - ▶ quindi $y.d = \delta(s, y)$
 - ▶ $W(s \rightsquigarrow x \rightarrow y \rightsquigarrow u) = W(s \rightsquigarrow x \rightarrow y) + W(y \rightsquigarrow u)$
 - ▶ $W(s \rightsquigarrow x \rightarrow y \rightsquigarrow u) = y.d + W(y \rightsquigarrow u)$
 - ▶ $W(s \rightsquigarrow x \rightarrow y \rightsquigarrow u) \geq y.d$
 - ▶ $W(s \rightsquigarrow x \rightarrow y \rightsquigarrow u) \geq y.d \geq u.d$ (u era astratto da Q)
 - ▶ $\implies u.d = \delta(s, u)$



3. Correttezza dell'algoritmo

- ▶ caso II: $u.d = \infty$
 - ▶ se il vertice u viene estratto quando $u.d = \infty$, allora non esiste nessun cammino tra s e u (proprietà 3)
 - ▶ cioè $u.d = \infty = \delta(s, u)$

