Cammini minimi con sorgente singola

1

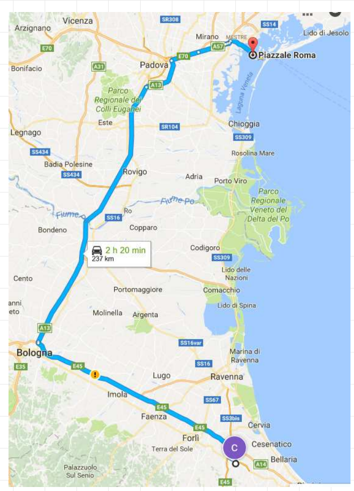
Cammini minimi con sorgente singola

Dato:
- un grafo (orientato o non orientato) $G=(V,E,W)$ con funzione di peso $w{:}E\rightarrow\mathbf{R}$
- un particolare vertice $s\in V$,

Trovare: per ogni vertice $v\in V$ il cammino di peso minimo da s a v.

Altri casi:
- trovare il cammino minimo fra una coppia di vertici u e v.
- trovare i cammini minimi fra tutte le coppie di vertici.

Ipotesi: nel grafo non esistono *cicli* di peso negativo.



2

Cammini minimi

Sia $\boldsymbol{G} = (\boldsymbol{V},\boldsymbol{E})$ un grafo orientato ai cui archi è associato un costo $\boldsymbol{W(u,v)}$.

Il *costo di un cammino* $\boldsymbol{p} = (\boldsymbol{v_0},\boldsymbol{v_1},\ldots,\boldsymbol{v_k})$ è la somma dei costi degli archi che lo costituiscono.

$$W(p) = \sum_{i=1}^{k} W(v_{i-1}, v_i)$$

Il *costo di un cammino minimo* da un vertice $\boldsymbol{u}$ ad un vertice $\boldsymbol{v}$ è definito come:

$$\delta(u,v) = \begin{cases} \min\{W(p)\} & \textit{se esistono cammini p da u a v} \\ \infty & \textit{altrimenti} \end{cases}$$

Un *cammino minimo* da $\boldsymbol{u}$ a $\boldsymbol{v}$ è un cammino $\boldsymbol{p}$ da $\boldsymbol{u}$ a $\boldsymbol{v}$ di costo $W(p) = \delta(u,v)$.

Rappresentazione

I cammini vengono rappresentati analogamente agli alberi BFS.

Per ogni vertice $v \in V$ si mantiene un predecessore $\pi(v)$.

I valori di π inducono un sottografo dei predecessori $G_\pi=(V_\pi,E_\pi)$.

G_π risulterà essere un albero di cammini minimi, cioè:

- V_π è l'insieme dei vertici raggiungibili da s in G.
- G_π forma un albero con radice in S
- per ogni $v \in V_\pi$, l'unico cammino semplice da s a v in G_π è un cammino minimo da s a v in G.

Archi negativi

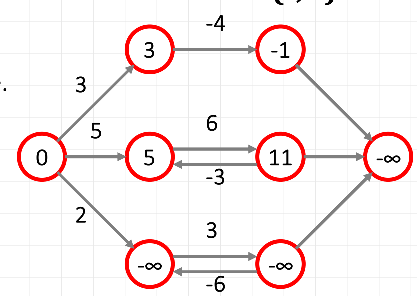
Il costo degli archi può essere negativo.

Questo può non creare problemi nella ricerca dei cammini minimi da una sorgente $\boldsymbol{s}$ a meno che vi siano cicli di costo negativo raggiungibili da $\boldsymbol{s}$.

Se $\boldsymbol{u}$ è un vertice raggiungibile da $\boldsymbol{s}$ con un cammino $\boldsymbol{p}$ passante per un vertice $\boldsymbol{v}$ di un ciclo negativo allora esistono cammini da $\boldsymbol{s}$ a $\boldsymbol{u}$ di costi sempre minori e il costo di cammino minimo $\boldsymbol{\delta(s,u)}$ non è definito.

In questo caso poniamo $\boldsymbol{\delta(s,u) = -\infty}$.

Per l'algoritmo di Dijkstra invece non si accettano archi di costo negativo.



Sottostruttura ottima

Sottostruttura ottima dei cammini minimi. Se il cammino $\boldsymbol{p = (v_0, v_1, \ldots, v_k)}$ è minimo allora sono minimi anche tutti i sottocammini $\boldsymbol{p_{ij} = (v_i, \ldots, v_j)}$.

Dimostrazione. Se esistesse un cammino $\boldsymbol{q}$ da $\boldsymbol{v_i}$ a $\boldsymbol{v_j}$ di costo minore di $\boldsymbol{p_{ij}}$ allora sostituendo nel cammino $\boldsymbol{p}$ il sottocammino $\boldsymbol{p_{ij}}$ con il cammino $\boldsymbol{q}$ si otterrebbe un cammino da $\boldsymbol{v_0}$ a $\boldsymbol{v_k}$ di costo minore di $\boldsymbol{p}$. Impossibile se $\boldsymbol{p}$ è minimo.

Quindi se $\boldsymbol{p}$ è un cammino minimo da $\boldsymbol{s}$ ad un vertice $\boldsymbol{v}$ diverso da $\boldsymbol{s}$ ed $\boldsymbol{u}$ è il vertice che precede $\boldsymbol{v}$ nel cammino allora

$$\delta(s,v) = \delta(s,u) + W(u,v)$$

Dijkstra, InitializeSingleSource

Algoritmo basato su un *rilassamento*: in $\mathrm{d[v]}$ si tiene un limite superiore al costo del cammino minimo sa s a v *(stima di cammino minimo).*

$p[\mathrm{v}]$: predecessore del vertice v.

Inizializzazione delle strutture:

```
Initialize-Single-Source(G,s)
foreach v ∈ V[G] do
  d[v] = ∞
  p[v] = 0
d[s] = 0
```


Algoritmo di Dijkstra, Relax

Rilassamento di un arco (u,v): verifica se è possibile migliorare il cammino minimo per v passante per u trovato fino a quel momento.
Se sì, si aggiornano $\mathrm{d[v]}$ e $\mathrm{p[v]}$.

```
Relax(u, v, w)
if   d[v] > d[u] + w(u,v)
then d[v] = d[u] + w(u,v)
     p[v] = u
```

Algoritmo di Dijkstra, Relax

Effetto del rilassamento: dopo aver eseguito *Relax*(*G*,*u*,*v*) vale la diseguaglianza $\boldsymbol{d[v] \leq d[u] + W(u,v)}$

Dimostrazione.

Se $\boldsymbol{d[v] > d[u] + W(u,v)}$ prima del rilassamento viene posto $\boldsymbol{d[v] = d[u] + W(u,v)}$.

Se $\boldsymbol{d[v] \leq d[u] + W(u,v)}$ prima del rilassamento non viene fatto nulla e quindi è vero anche dopo.

Algoritmo di Dijkstra

Mantiene un insieme S che contiene i vertici v il cui peso del cammino minimo da s, $\delta(s,v)$, è già stato determinato.

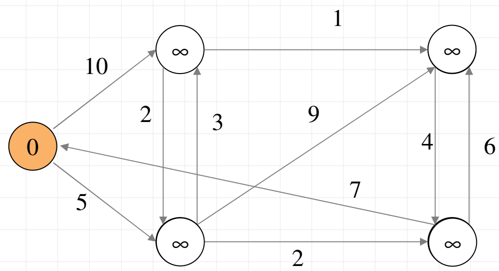
```
Dijkstra(G, w, s)
Initialize-Single-Source(G, s)
S = ∅
Q = V[G]
while Q ≠ 0 do
  u = Extract-Min(Q)
  S = S ∪ {u}
  for v ∈ Adj[u] do
    Relax(u,v,w)
```

Algoritmo di Dijkstra

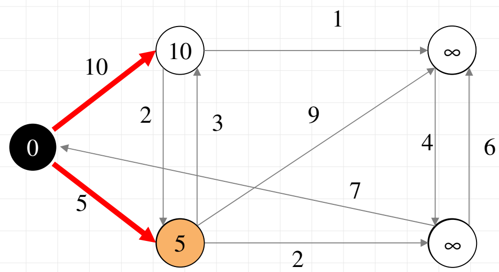
In un'unica funzione:

```
Dijkstra(G, w, s)
foreach v ∈ V[G] do
   d[v] = ∞
   p[v] = 0
d[s] = 0
S = ∅
Q = V[G]
while Q != 0 do
   u = Extract-Min(Q)
   S = S ∪ {u}
   for v ∈ Adj[u] do
      if d[v] > d[u] + w(u,v) then
         d[v] = d[u] + w(u,v)
         p[v] = u
      endif
```

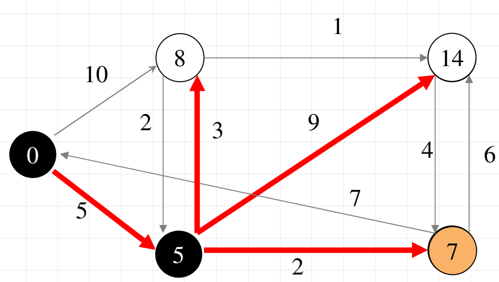

Dijkstra, esempio



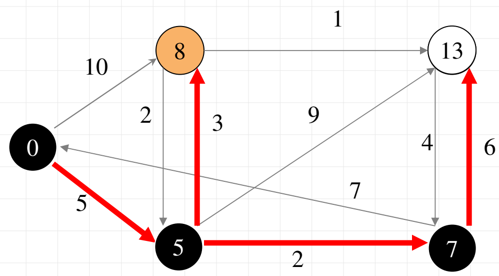
Dijkstra, esempio



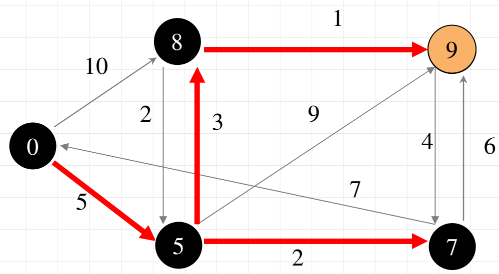
Dijkstra, esempio



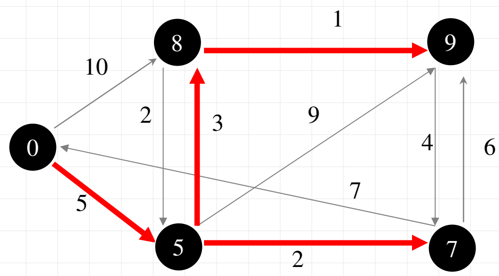
Dijkstra, esempio



Dijkstra, esempio



Dijkstra, esempio



Correttezza

Teorema:
Se si esegue l'algoritmo di Dijkstra su di un grafo G=(V,E,W) con funzione di peso non negativa W e sorgente s, allora al termine si ha $d[v]=\delta(s,v)$ per ogni vertice $v \in V_\pi$.

Dimostrazione.

È semplice verificare che ogni nodo raggiungibile da s verrà considerato nella ricerca (v. graph traversal).

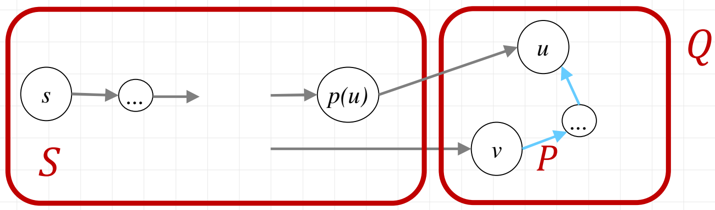
Si vuole provare che quando un nodo u viene scoperto, $d(u) = \delta(s,u)$.

Correttezza

Per induzione:
Base: per la sorgente s si ha $d(s) = 0 = \delta(s,s)$
Passo induttivo: all'iterazione k si estrae da Q il vertice u tale per cui $d(u) = \min_{v \in Q} d(v)$ e $d(p(u)) = \delta(s, p(u))$.

Gli archi del grafo hanno tutti lunghezza $w(i,j) \geq 0$, per cui ogni cammino $P(v,u)$ con $v,u \in Q$, ha lunghezza $d_P \geq 0$.

Quindi $d(u) \leq d(v) \leq d(v) + d_P$ per ogni $v \in Q$ e quindi $d(u)$ è la distanza minima da s a u, $\delta(s,u)$.



Complessitá

InitializeSingleSource $T_I(V,E) = O(V)$

Relax $T_R(V,E) = O(1)$

Dijkstra (coda Q=V-S come array)

$T(V,E) = T_I(V,E) + O(V^2) + E\ T_R(V,E) =$

$= O(V) + O(V^2) + E\ O(1) = O(V^2+E) = O(V^2)$

Inizializzazione

Extract-min

Relax

PQ con array?

Ma vedi complessità di Prim! (perchè?)

Algoritmo di Bellman-Ford

È un algoritmo di programmazione dinamica che risolve lo stesso problema che Dijkstra risolveva in modo greedy.

Basato sull'equazione ricorsiva

$$d^k[v] = \begin{cases} (v = s\ ?\ 0 : \infty) & k = 0 \\ \min\limits_{u \in V}\{d^{k-1}[u] + w(u,v)\} & 1 \le k \le n \end{cases}$$

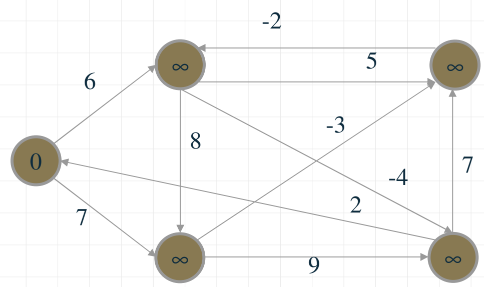
dove k indica il numero di archi intermedi che il cammino può utilizzare.

Algoritmo di Bellman-Ford

Accetta anche archi con peso negativo.
Restituisce un booleano che dice se esiste un ciclo di peso negativo (nessuna soluzione) oppure produce l'albero dei cammini minimi.

```
BellmanFord(G, w, s)
Initialize-Single-Source(G, s)
for i = 1 to |V[G] – 1| do
   for (u,v) ∈  E[G]  do
      Relax(u,v,w)
for (u,v) ∈  E[G]  do
   if d[v] > d[u] + w(u,v)
   then return false
return true
```

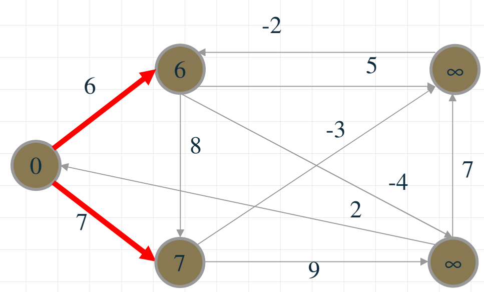
Bellman-Ford, esempio



Vittorio Maniezzo - Universita di Bologna 23

23

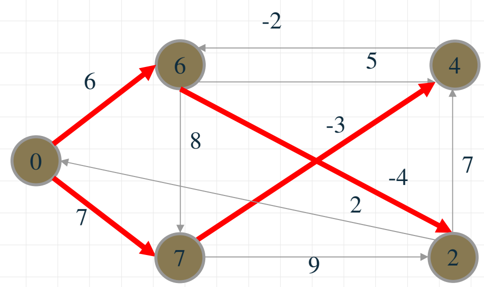
Bellman-Ford, esempio



Vittorio Maniezzo - Universita di Bologna 24

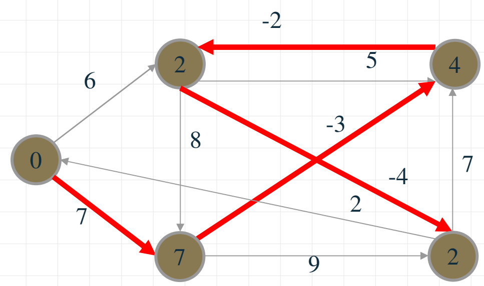
24

Bellman-Ford, esempio



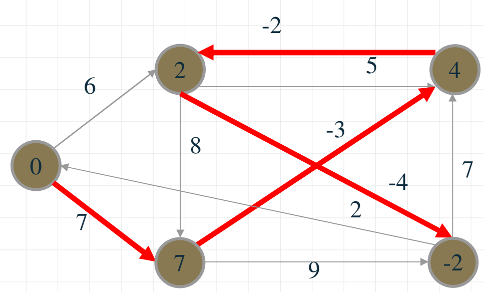
25

Bellman-Ford, esempio



26

Bellman-Ford, esempio



TRUE

Bellman-Ford, correttezza

Teorema

Si esegua Bellman-Ford su un grafo orientato e pesato $G=(V,E)$ con sorgente s e funzione di peso $w: E \rightarrow \mathbf{R}$.

- Se G non contiene cicli di peso negativo l'algoritmo restituisce TRUE e si ha che $d[v]=\delta(s,v)$ per tutti i vertici $v \in V$ e che il sottografo dei predecessori è un albero di cammini minimi radicato in s.
- Se G ha un ciclo di peso negativo, l'algoritmo restituisce FALSE.

Bellman-Ford, complessitá

InitializeSingleSource $T_I(V,E) = O(V)$

Relax $T_R(V,E) = O(1)$

BellmanFord

$T(V,E) = T_I(V,E) + V\ E\ T_R(V,E) + E =$
$= O(V) + V\ E\ O(1) + E = O(V\ E)$

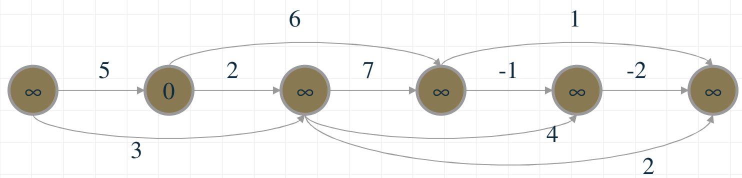
Inizializzazione | Propagazione | Controllo

Cammini minimi in DAG

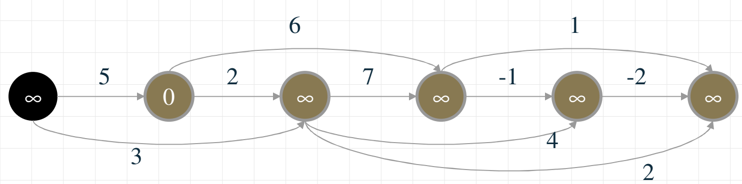
Rilassando gli archi di un grafo pesato G=(V,E) secondo un ordine topologico, si possono calcolare i cammini minimi da una singola sorgente in tempo O(V+E).

```
DAG-Shortest-Path(G, w, s)
ordina topologicamente i vertici di G
Initialize-Single-Source(G, s)
foreach vertice u nell’ordine topologico do
  foreach vertice v ∈ Adj[u] do
    Relax(u,v,w)
```

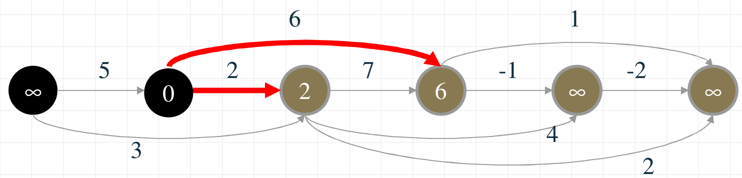
Cammini minimi in DAG, esempio



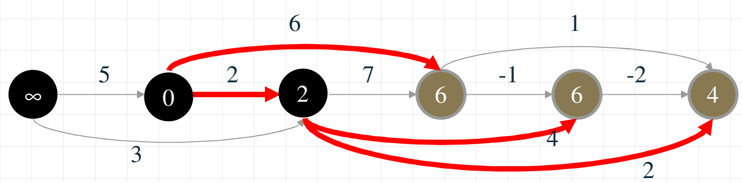
Cammini minimi in DAG, esempio



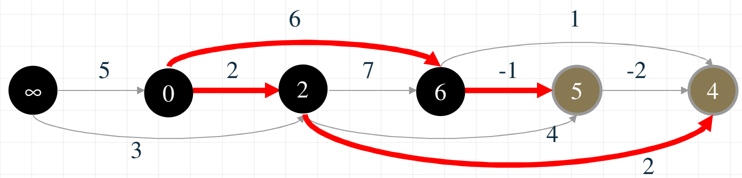
Cammini minimi in DAG: esempio



Cammini minimi in DAG: esempio



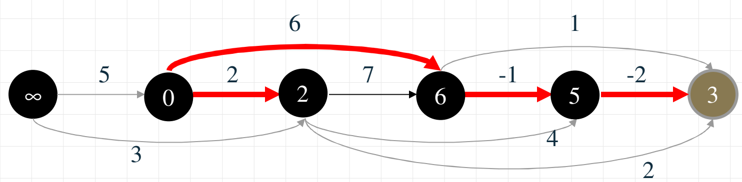
Cammini minimi in DAG: esempio



Vittorio Maniezzo - Universita di Bologna 35

35

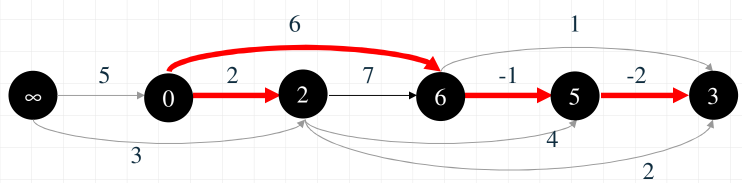
Cammini minimi in DAG: esempio



Vittorio Maniezzo - Universita di Bologna 36

36

Cammini minimi in DAG: esempio



Cammini minimi in DAG, complessitá

$$T(V,E) = \Theta(V + E) + \Theta(V) + E\,\Theta(1) = \Theta(V + E)$$

Ordinamento

Inizializzazione

Propagazione