Cammini minimi

Damiano Macedonio
mace@unive.it

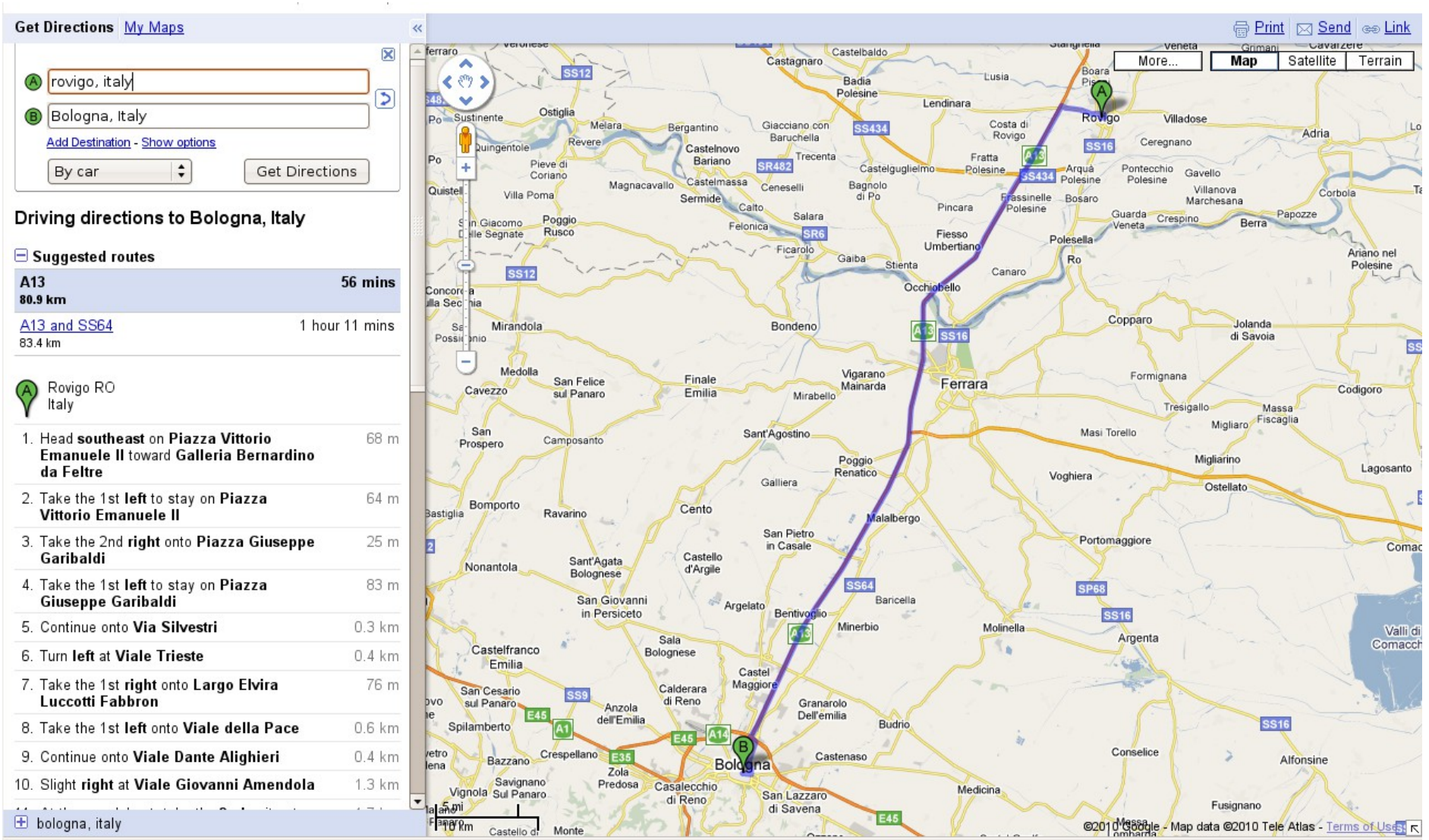
Definizione del problema

- Consideriamo un grafo orientato $G=(V, E)$ in cui ad ogni arco $e \in E$ sia associato un costo $w(e)$
- Il costo di un cammino $\pi=(v_0, v_1, \ldots v_k)$ che connette il vertice v_0 con v_k è definito come

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

- Data una coppia di vertici v_0 e v_k, vogliamo trovare (se esiste) il cammino $\pi^*_{v0\ vk}$ di costo minimo tra tutti i cammini che vanno da v_0 a v_k

Applicazioni dei cammini minimi



Applicazioni dei cammini minimi (oops!)

- Cammino più breve tra Haugesund e Trondheim

Nel 2005

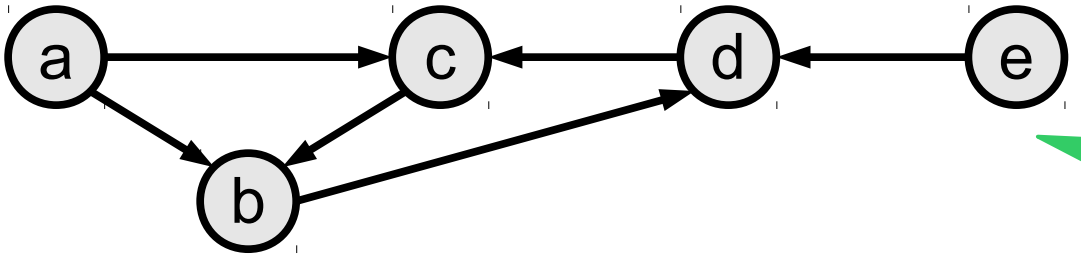
Nel 2010

Diverse formulazioni del problema

- *Single-source shortest path*
  - Determinare i cammini minimi da un vertice sorgente s a tutti i vertici raggiungibili da s
- *All-pairs shortest paths*
  - Determinare i cammini minimi tra ogni coppia di vertici u, v tali che esista un cammino da u a v
- *Cammino minimo fra una singola coppia di vertici u e v*
  - Determinare esclusivamente il cammino minimo π^*_{uv}
  - Nessun algoritmo conosciuto è in grado di risolvere il problema del cammino minimo fra una singola coppia senza risolvere anche il problema dei cammini minimi da singola sorgente

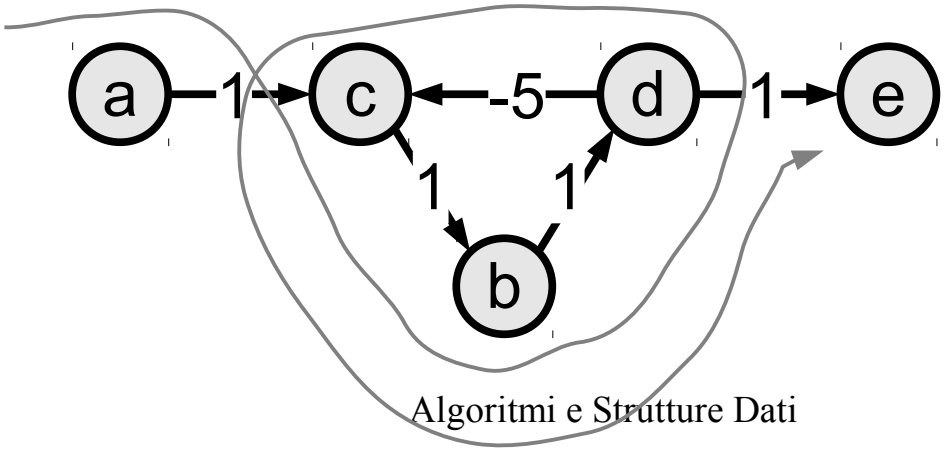
Osservazione

- In quali situazioni non esiste un cammino minimo?
  - Quando i vertici non sono connessi



Non esiste alcun cammino che connette ***a*** con ***e***

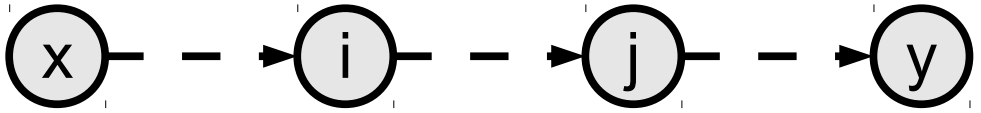
  - Quando ci sono cicli di costo negativo



È sempre possibile trovare un cammino di costo inferiore che connette ***a*** con ***e***

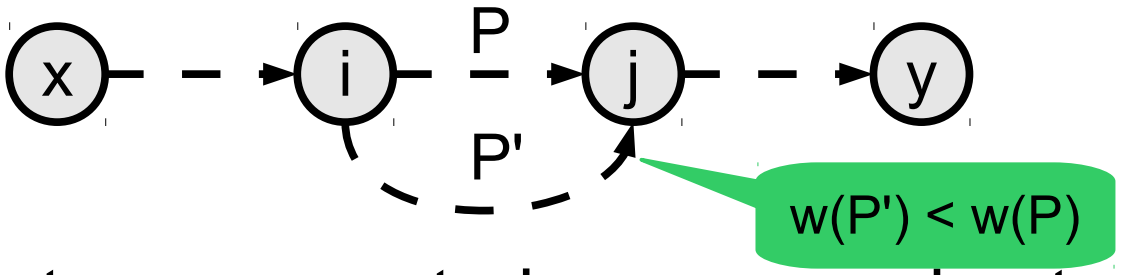
Proprietà (sottostruttura ottima)

- Sia $G=(V, E)$ un grafo orientato con funzione costo w; allora ogni sottocammino di un cammino minimo in G è a sua volta un cammino minimo in G
- Dimostrazione
  - Consderiamo un cammino minimo π^*_{xy} da x a y
  - Siano i e j due nodi intermedi
  - Dimostriamo che il sottocammino di π_{xy}^* che collega i e j è un cammino minimo tra i e j

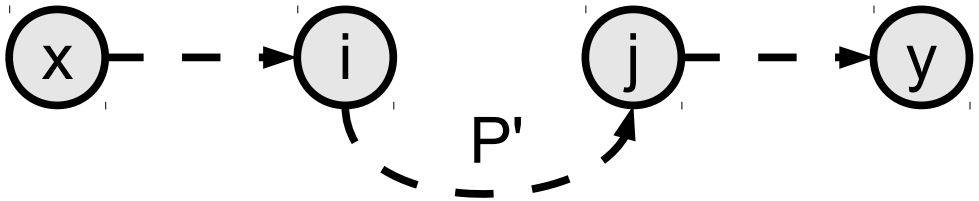


Proprietà (sottostruttura ottima)

- Supponiamo per assurdo che esista un cammino P' tra *i* e *j* di costo strettamente inferiore a P

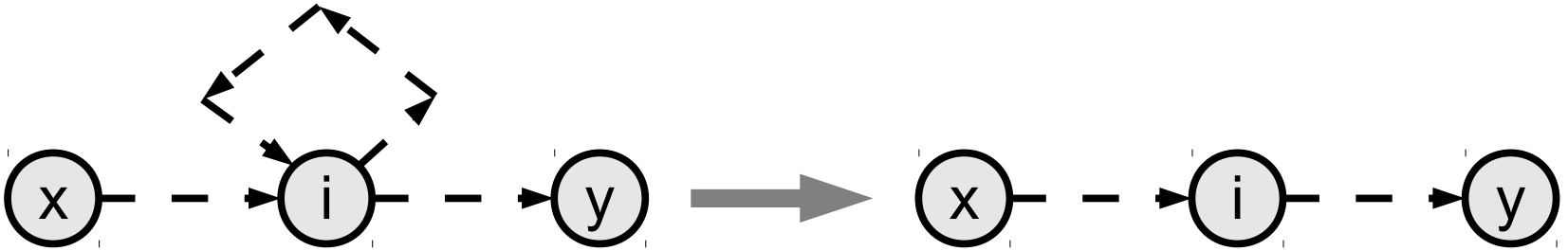


- Ma allora potremmo costruire un cammino tra *x* e *y* di costo inferiore a π_{xy}^*, il che è assurdo perché avevamo fatto l'ipotesi che π_{xy}^* fosse il cammino di costo minimo



Esistenza

- Sia $G=(V, E)$ un grafo orientato con funzione peso w. Se non ci sono cicli negativi, allora fra ogni coppia di vertici connessi in G esiste sempre un cammino minimo
- Dimostrazione
  - Possiamo sempre trasformare un cammino in un cammino semplice (privo di cicli)



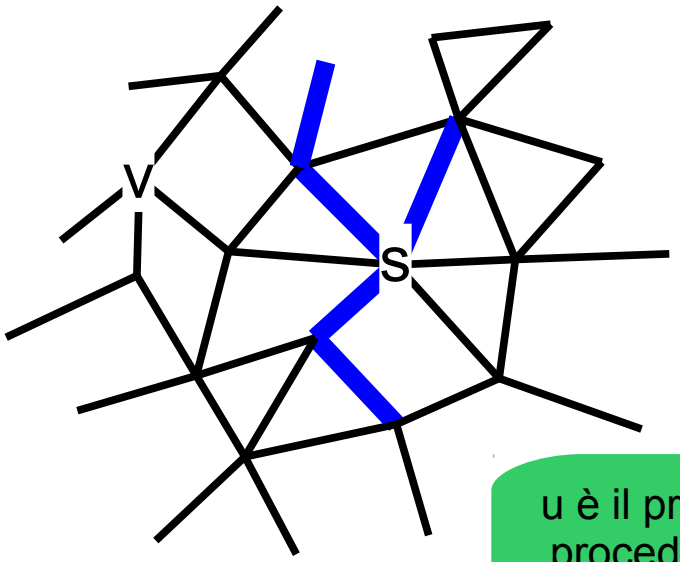
  - Ogni volta che si rimuove un ciclo, il costo diminuisce (o resta uguale), perché non ci sono cicli negativi
  - Per trovare un cammino minimo possiamo restringere la ricerca ai cammini semplici (che sono in numero finito)

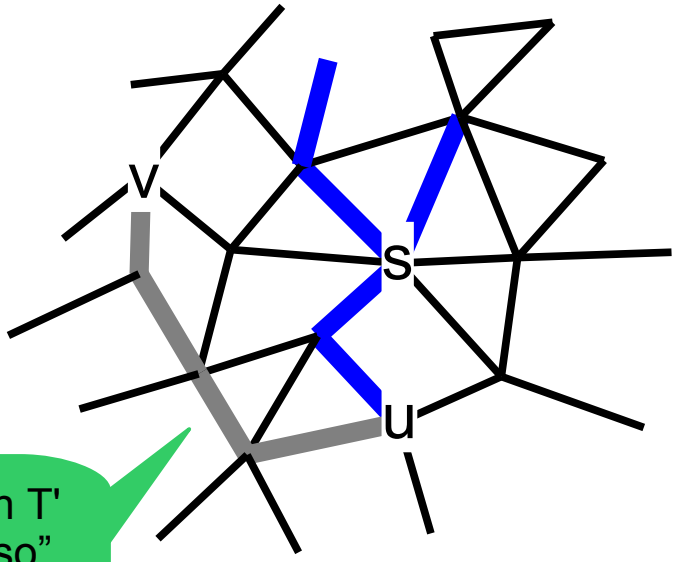
Nota

- D'ora in poi, assumiamo che nel grafo G non esistano cicli di lunghezza negativa

Albero dei cammini minimi

- Sia s un vertice prefissato di un grafo orientato pesato $G=(V, E)$. Allora esiste un albero T che contiene i vertici raggiungibili da s tale che ogni cammino in T sia un cammino minimo
  - Grazie alla proprietà di sottostruttura ottima, è sempre possibile far “crescere” un albero parziale T' fino a includere tutti i vertici raggiungibili





u è il primo vertice in T' procedendo “a ritroso” nel cammino π_{sv}*

Distanza tra vertici in un grafo

- Sia $G=(V, E)$ un grafo orientato con funzione costo w. La distanza d_{xy} tra x e y in G è il costo di un cammino minimo che li connette; $+\infty$ se tale cammino non esiste

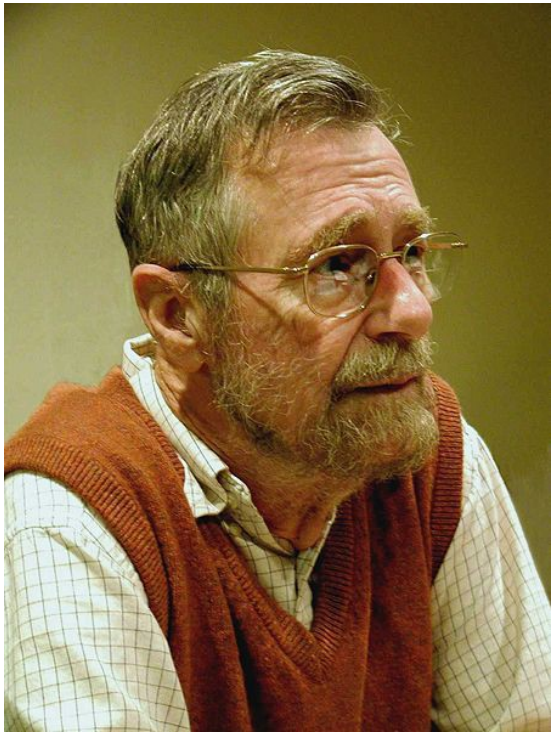
$$d_{xy} = \begin{cases} w(\pi^*_{xy}) & se\ esiste\ un\ cammino\ minimo\ \pi^*_{xy} \\ +\infty & altrimenti \end{cases}$$

- **Nota**: $d_{vv} = 0$ per ogni vertice v
- **Nota**: Vale la disuguaglianza triangolare

$$d_{xz} \leq d_{xy} + d_{yz}$$

Algoritmo di Dijkstra

- Consente di calcolare i cammini minimi da singola sorgente nel caso in cui gli archi abbiano costo ≥ 0
  - Il grafo può contenere cicli, che per l'ipotesi sopra non possono avere lunghezza negativa



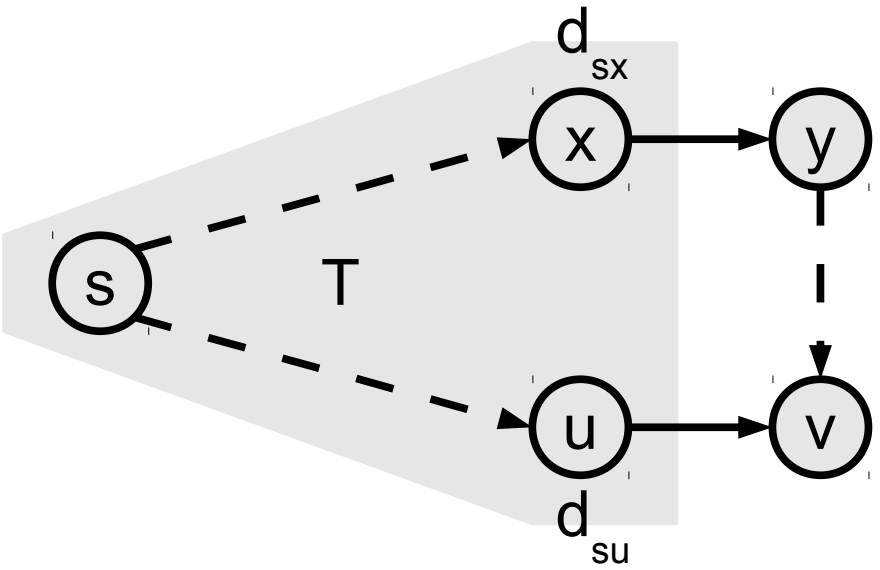
Edsger W. Dijkstra, (1930—2002)
http://en.wikipedia.org/wiki/Edsger_W._Dijkstra

Lemma (Dijkstra)

- Sia $G=(V, E)$ un grafo orientato con funzione costo w; i costi devono essere ≥0. Se T è un albero di cammini minimi radicato in s, ma che non include tutti i vertici raggiungibili da s, allora l'arco (u, v) con $u \in V(T)$ e $v \notin V(T)$ che minimizza la quantità $d_{su} + w(u,v)$ appartiene ad un cammino minimo da s a v

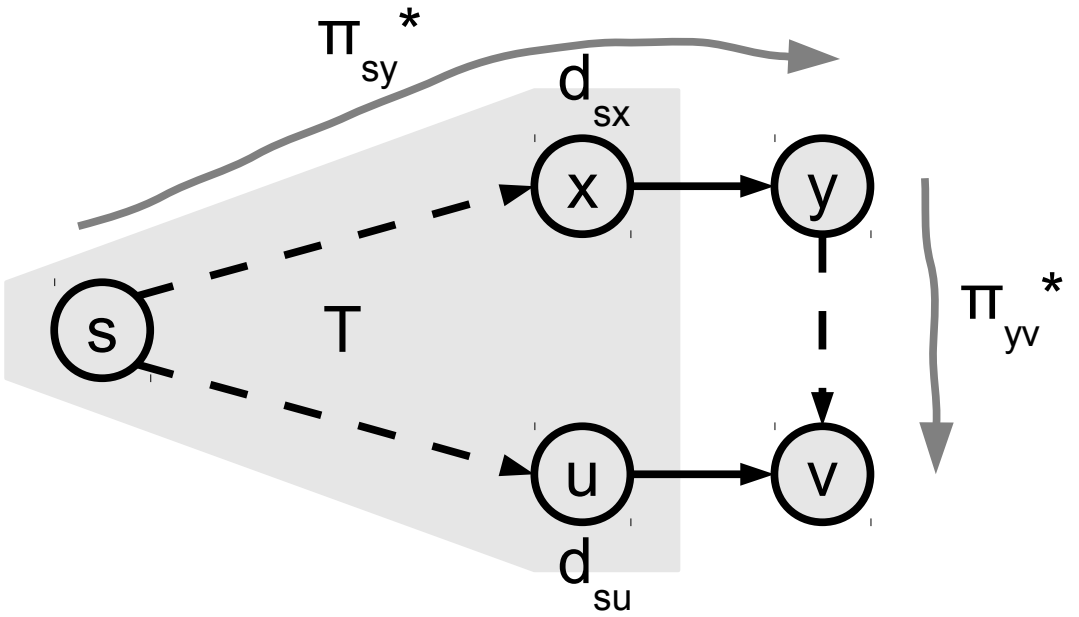
Dimostrazione

- Supponiamo per assurdo che (u,v) *non appartenga* ad un cammino minimo tra s e v
  - quindi $d_{su} + w(u,v) > d_{sv}$ (1)
- Quindi deve esistere π^*_{sv} che porta da s in v senza passare per (u,v) con costo inferiore a $d_{su} + w(u,v)$



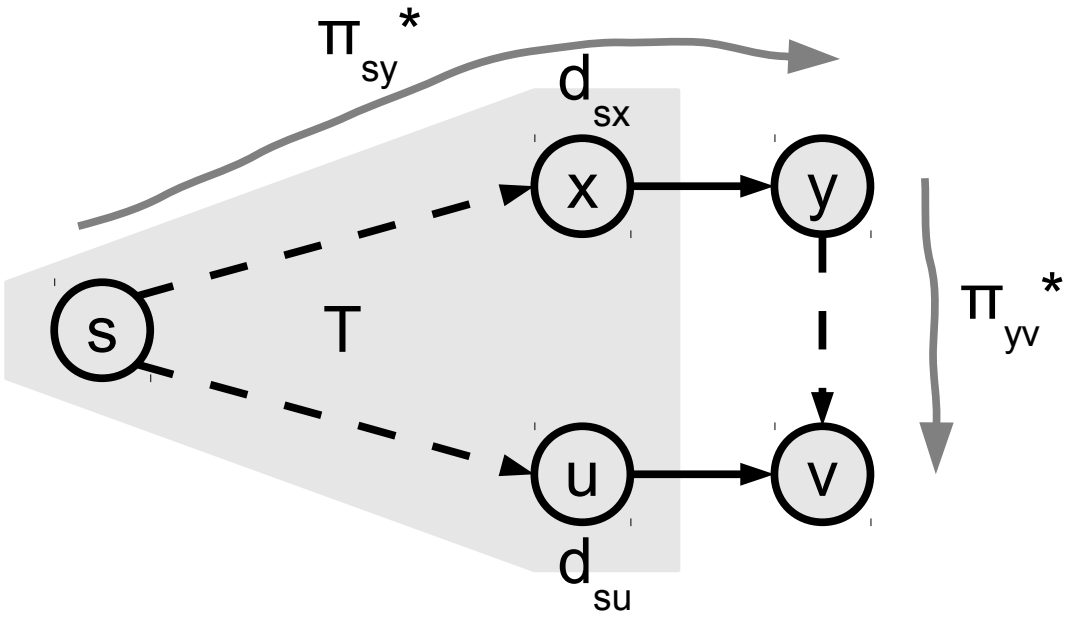
Dimostrazione

- Per il teorema di sottostruttura ottima, il cammino π_{sv}^* si scompone in π_{sy}^* e π_{yv}^*
- Quindi $d_{sv} = d_{sx} + w(x,y) + d_{yv}$ (2)



Dimostrazione

- Per ipotesi (lemma di Dijkstra), l'arco (u,v) è quello che, tra tutti gli archi che collegano un vertice in T con uno non ancora in T, minimizza la somma $d_{su} + w(u,v)$
- In particolare: $d_{su} + w(u,v) \leq d_{sx} + w(x,y)$ (3)



Riassumiamo

- Da (1) abbiamo $d_{su} + w(u,v) > d_{sv}$
- Da (2) abbiamo $d_{sv} = d_{sx} + w(x,y) + d_{yv}$
- Da (3) abbiamo $d_{su} + w(u,v) \leq d_{sx} + w(x,y)$
- Combinando (1) (2) e (3) otteniamo

$$
\begin{aligned}
d_{su} + w(u,v) &> d_{sx} + w(x,y) + d_{yv} && \text{da (1) e (2)} \\
&\geq d_{sx} + w(x,y) \\
&\geq d_{su} + w(u,v) && \text{da (3)}
\end{aligned}
$$

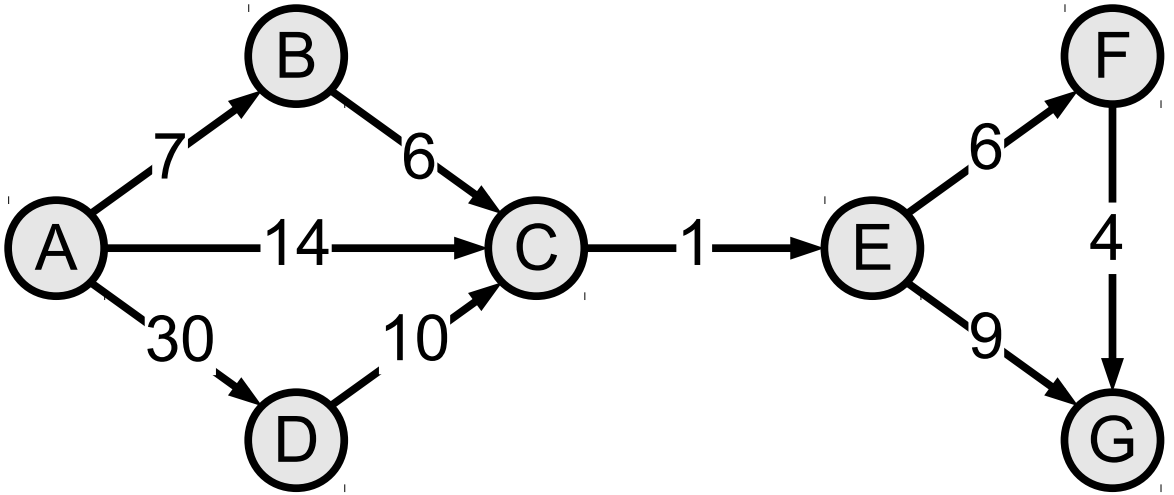
Assurdo!

I pesi sono positivi, quindi d_{yv} non è negativo

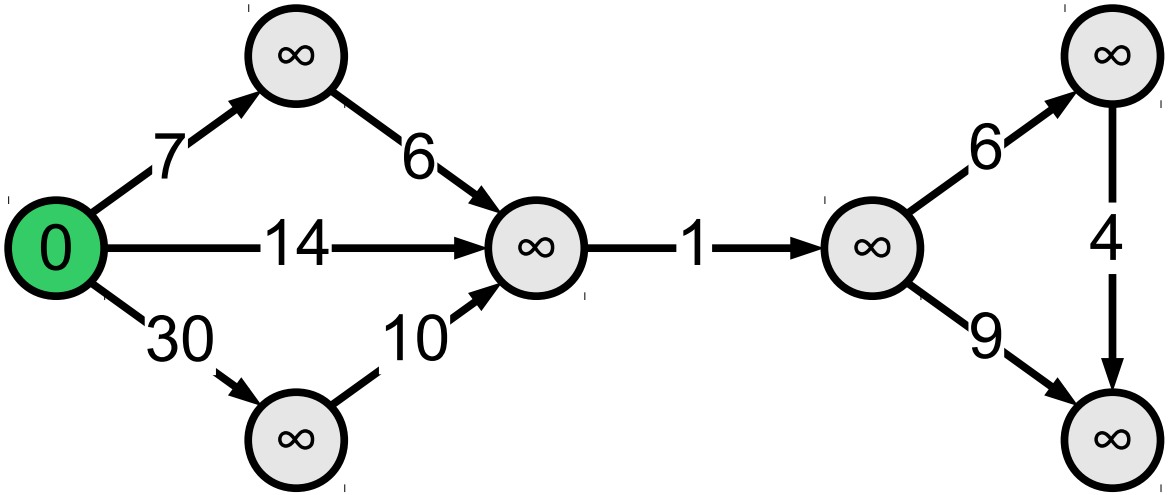
Algoritmo di Dijkstra generico

```
algoritmo DijkstraGenerico(G, s)→albero
    inizializza D tale che
        D_sv := +∞ se v ≠ s
        D_ss := 0
    T := albero formato dal solo vertice s
    while (T ha meno di n nodi) do
        Trova l'arco (u,v) incidente su T con D_su + w(u,v) minimo
        D_sv := D_su + w(u,v)
        rendi u padre di v in T
    endfor
```

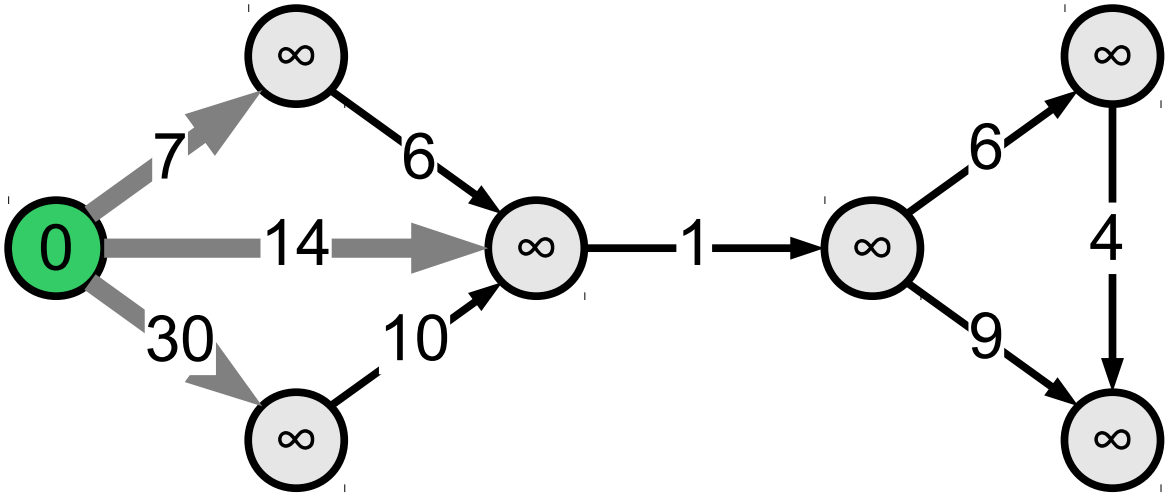
Esempio



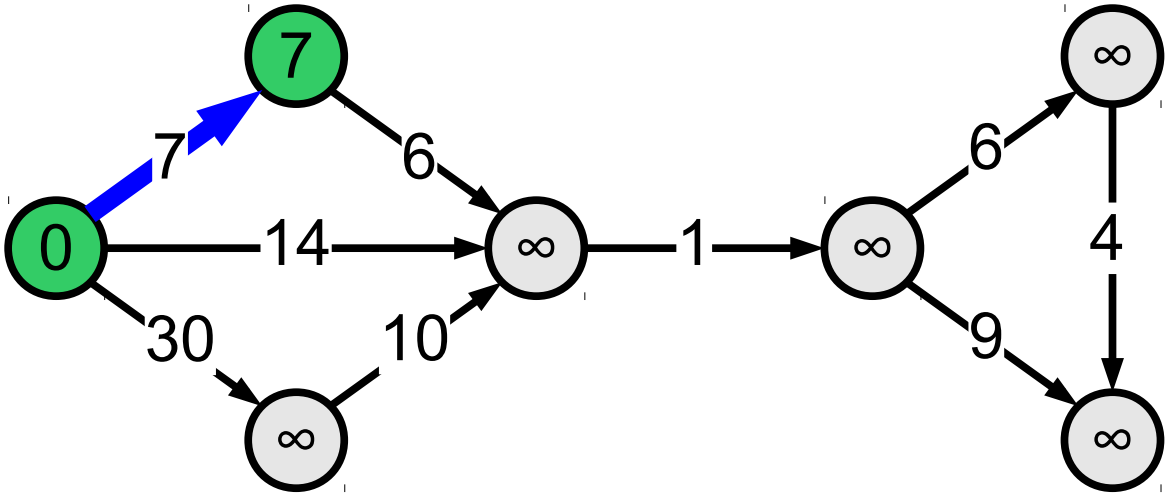
Esempio



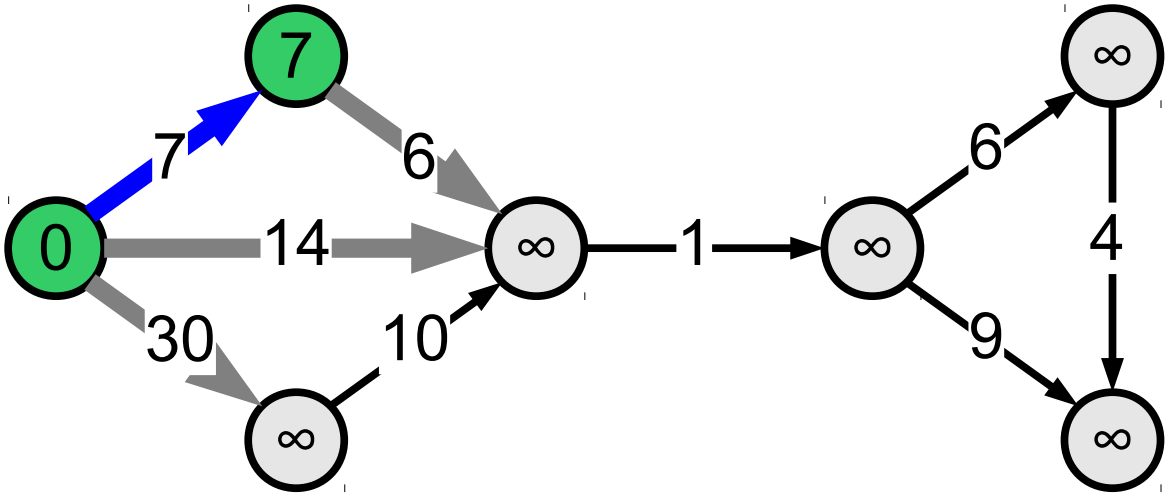
Esempio



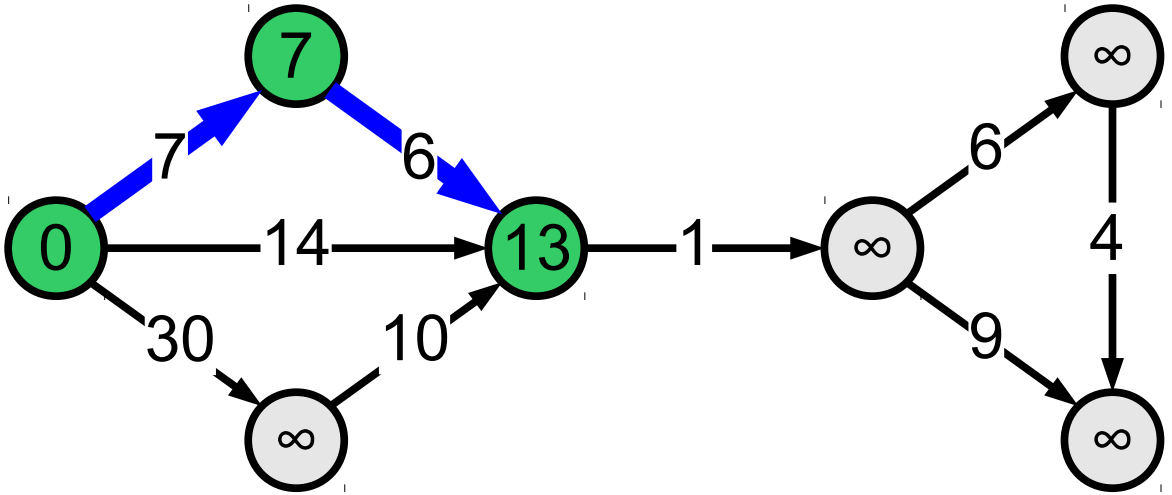
Esempio



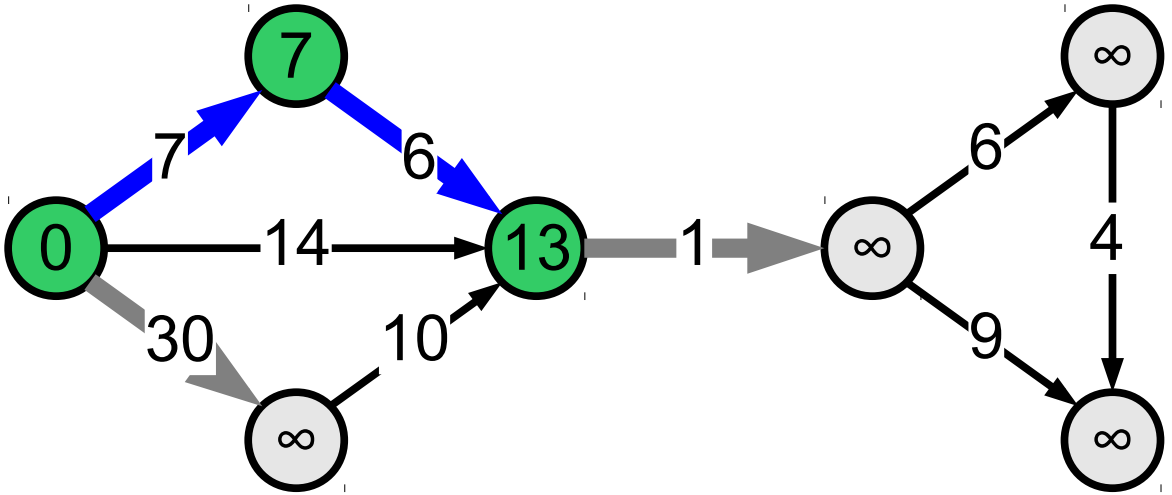
Esempio



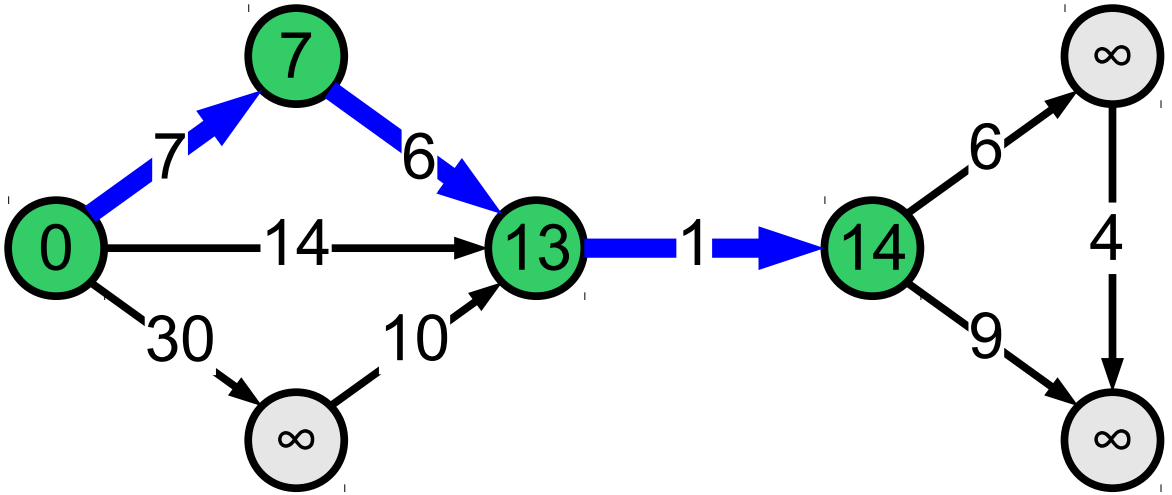
Esempio



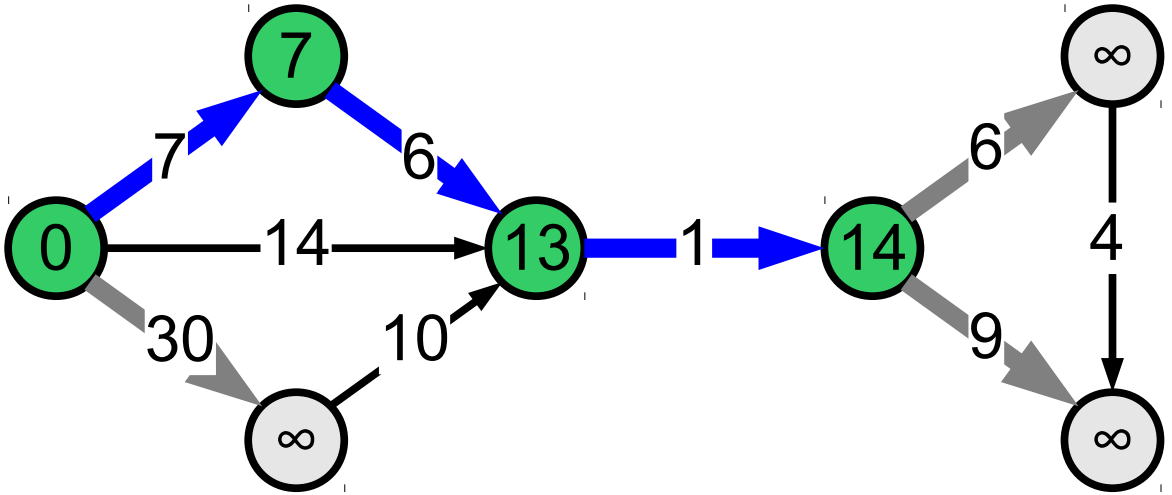
Esempio



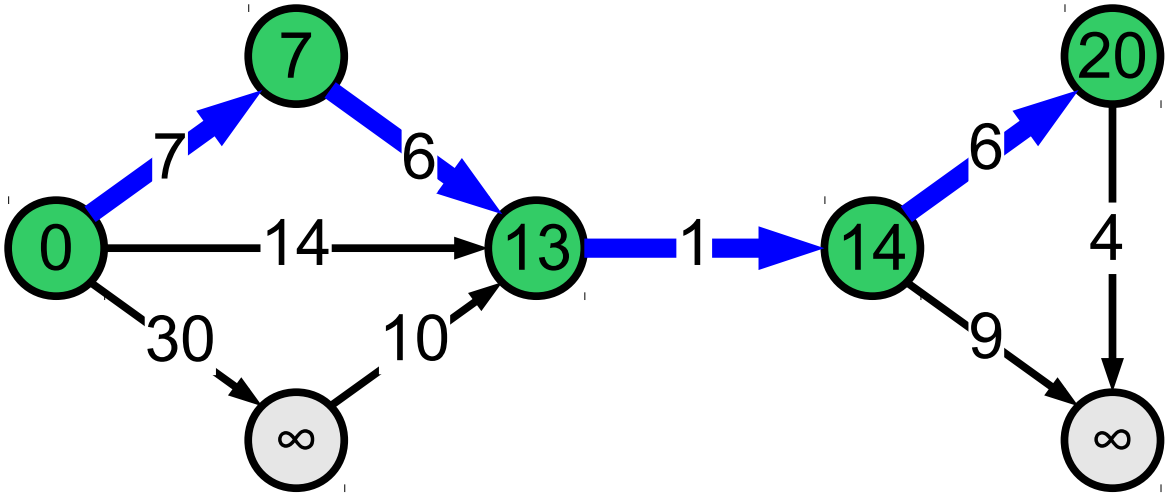
Esempio



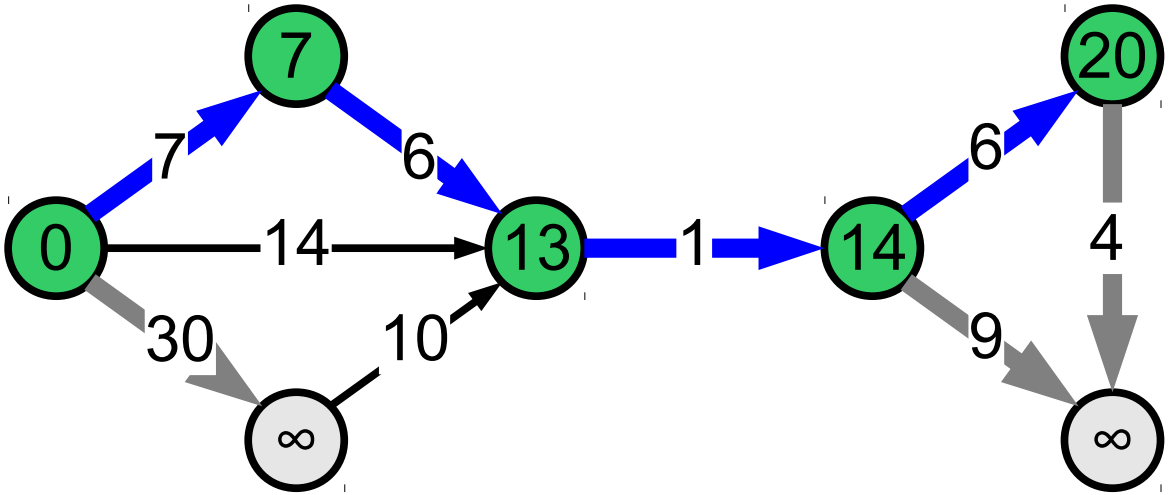
Esempio



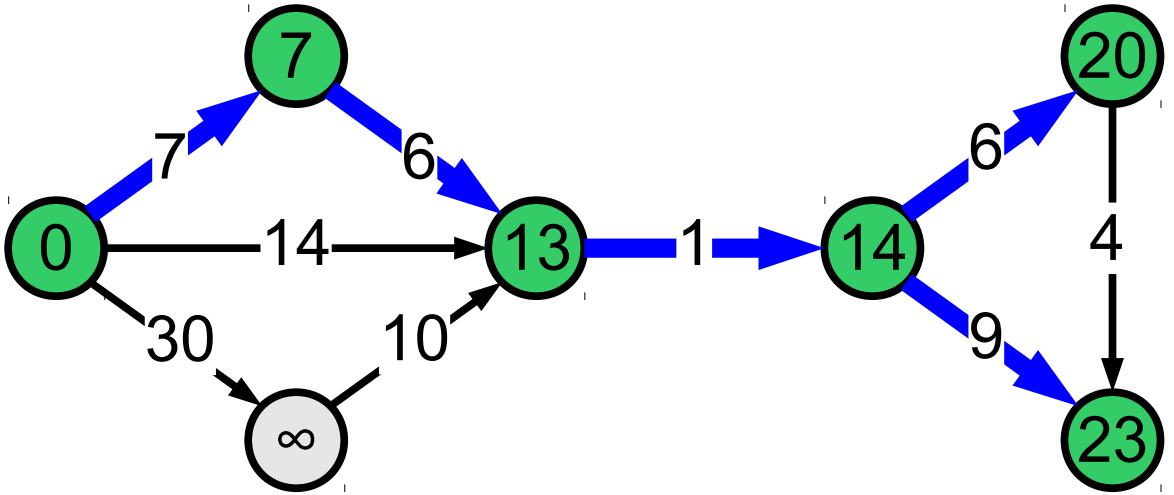
Esempio



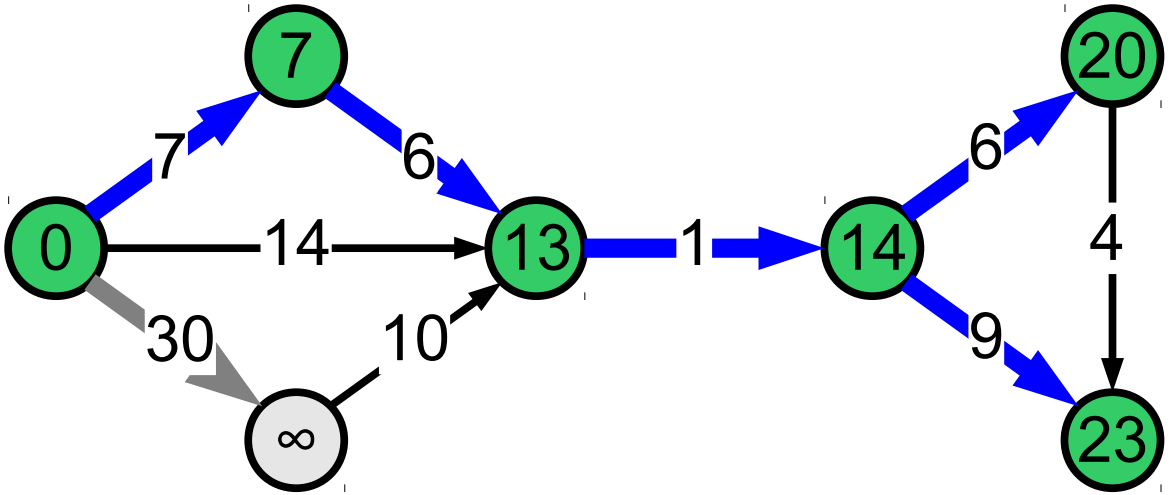
Esempio



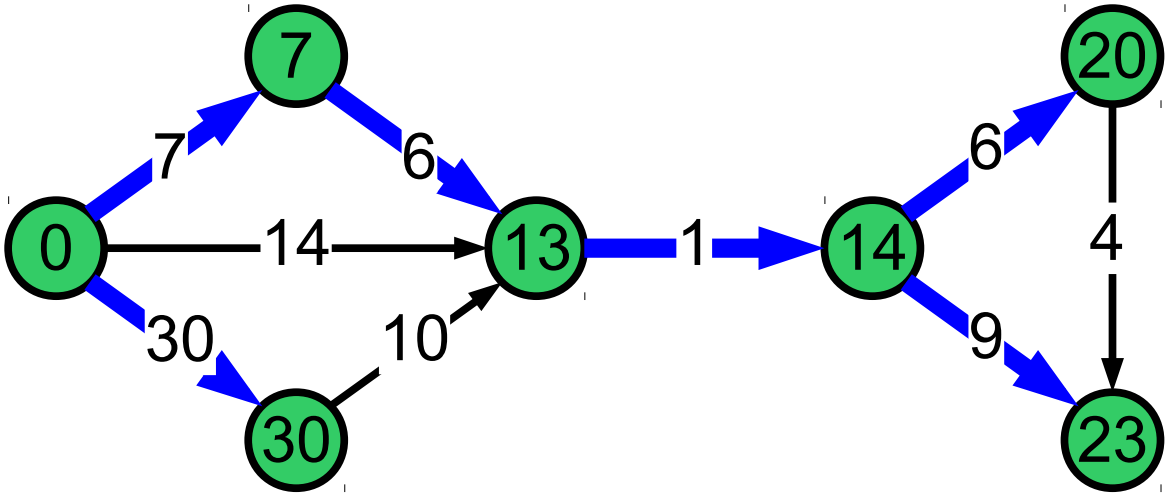
Esempio



Esempio



Esempio



Algoritmo di Dijkstra

```
algoritmo Dijkstra(Grafo G=(V,E), nodo s)→albero
    for each u in V do D_su := ∞ endfor
    T := albero formato dal solo nodo s
    S := CodaPriorita di (nodi, costi);
    D_ss := 0;
    S.insert(s,0);
    while (not S.isEmpty()) do
        u := S.extractMin();
        for each (u,v)∈E do
            if (D_sv = ∞) then
                S.insert(v,D_su + w(u,v));
                D_sv := D_su + w(u,v);
                rendi u padre di v in T;
            else if (D_su + w(u,v) < D_sv)) then
                S.decreaseKey(v,D_su + w(u,v));
                D_sv := D_su + w(u,v);
                rendi u nuovo padre di v in T;
            endif
        endfor
    endwhile
    return T;
```

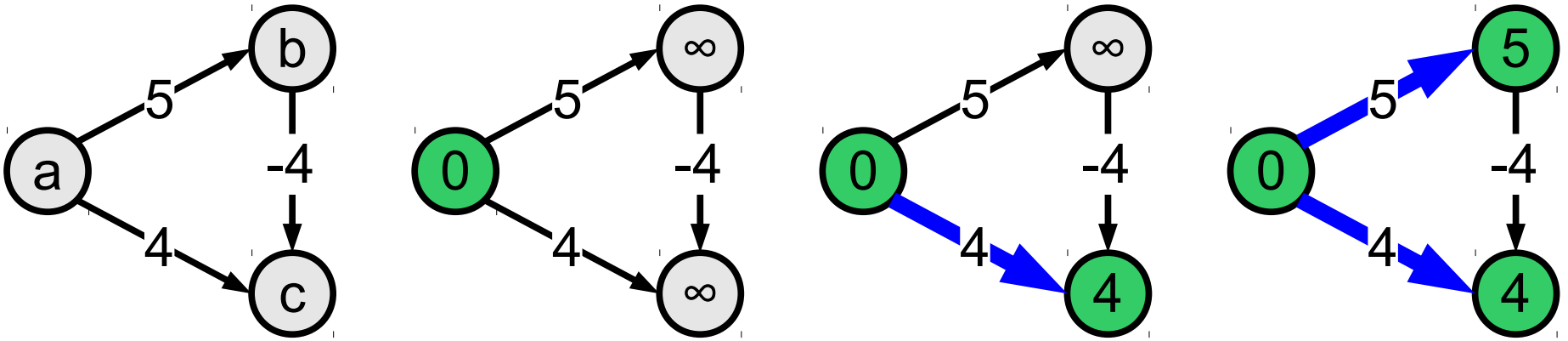
Rendi D_{su}+w(u,v) il nuovo costo di v

Analisi dell'algoritmo di Dijkstra

- $O(m \log n)$ nel caso peggiore
- Osservazione
  - Con un minimo di “astuzia” e una implementazione più efficiente della coda di priorità è possibile implementare l'algoritmo di Dijkstra in tempo $O(m + n \log n)$
  - Vedere libro di testo (sezione 14.5.1) per i dettagli

Osservazione

- Notare che la dimostrazione di correttezza funziona perché si assume che i pesi dei cammini (in particolare, degli archi) non possono mai essere negativi
- Esempio di funzionamento errato



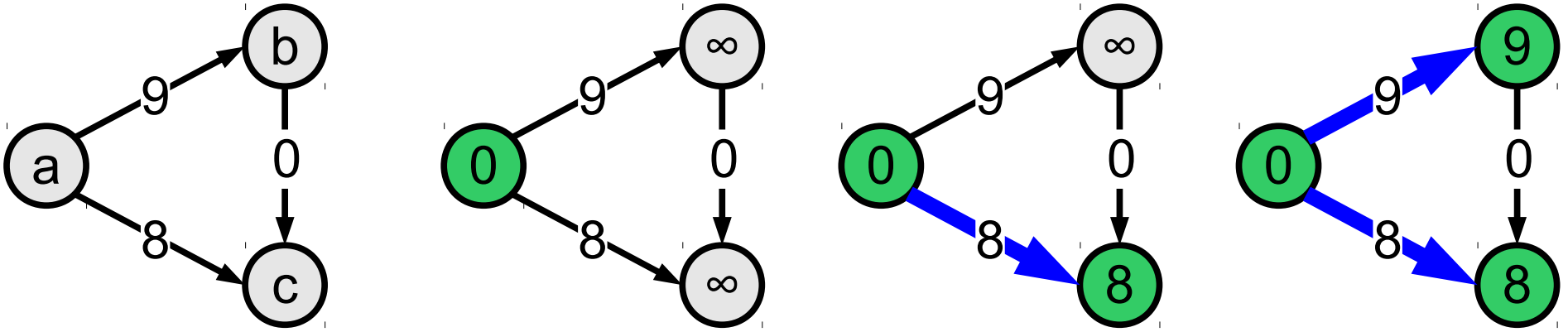
  - Il cammino minimo da a→c non è (a,c) ma (a,b,c) che ha costo complessivo 1!

Domanda

- Sia $G=(V, E)$ un grafo orientato con archi pesati, anche con pesi negativi; supponiamo che in G non esistano cicli negativi. Supponiamo di incrementare i pesi di tutti gli archi di una costante C in modo che tutti i pesi risultanti siano non negativi.
- L'algoritmo di Dijkstra applicato ai nuovi pesi restituisce l'albero dei cammini minimi anche per i pesi originali?

Risposta: NO


a
b
c
5
-4
4
Aggiungo 4 ai
pesi degli archi
a
b
c
9
0
8


a
b
c
9
0
8
0
∞
9
0
8
∞
0
∞
9
0
8
8
0
9
9
0
8
8

Algoritmo di Floyd e Warshall

- Si può applicare a grafi orientati con costi arbitrari (anche negativi), purché non ci siano cicli negativi
  - Basato sulla programmazione dinamica
- Sia $V=\{v_1, v_2, \ldots v_n\}$
- Sia d_{xy}^{k} la minima distanza per andare dal vertice x al vertice y, in cui si usano solo vertici in $\{v_1, v_2, \ldots v_k\}$ (oltre eventualmente x e y)
- La soluzione complessiva è d_{xy}^{n} per ogni coppia di vertici x e y
  - All-pair shortest paths

Algoritmo di Floyd e Warshall

- Per andare da x a y usando solo vertici intermedi in $\{v_1, \ldots v_k\}$ (sempre che ciò sia possibile), ho due alternative
  - Non passo mai per v_k. La distanza in tal caso è d_{xy}^{k-1}
  - Passo per v_k. Per la proprietà di sottostruttura ottima, la distanza in tal caso è $d_{x\,vk}^{k-1} + d_{vk\,y}^{k-1}$
- Quindi

$$d_{xy}^{k} = min\left\{ d_{xy}^{k-1}, d_{xv_k}^{k-1} + d_{v_k y}^{k-1} \right\}$$

Inizializzazione

- Posso calcolare d_{xy}^{0} come

d_{xy}^{0} = minima distanza per andare da x a y senza passare per altri nodi intermedi

$$d_{xy}^{0} = \begin{cases} 0 & \text{se } x = y \\ w(x, y) & \text{se } (x, y) \in E \\ \infty & \text{se } (x, y) \notin E \end{cases}$$

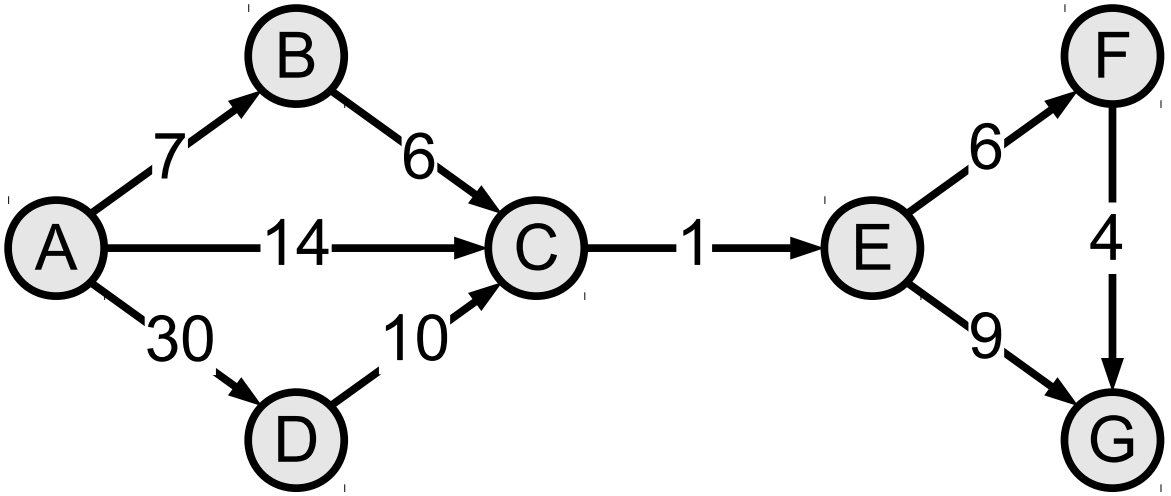
Algoritmo di Floyd e Warshall

```
algoritmo FloydWarshall( G=(V,E) )→distanze
    inizializza D_xy^0
    for k := 1 to n do
        for each x ∈ V do
            for each y ∈ V do
                D_xy^k := D_xy^k-1;
                if (D_x vk^k-1 + D_vk y^k-1 < D_xy^k ) then
                    D_xy^k := D_x vk^k-1 + D_vk y^k-1;
                endif
            enefor
        endfor
    endfor
    return D^n
```

$$d_{xy}^{k} = min\left\{ d_{xy}^{k-1}, d_{xv_k}^{k-1} + d_{v_k y}^{k-1} \right\}$$

- Costo: tempo $O(n^3)$, spazio $O(n^3)$

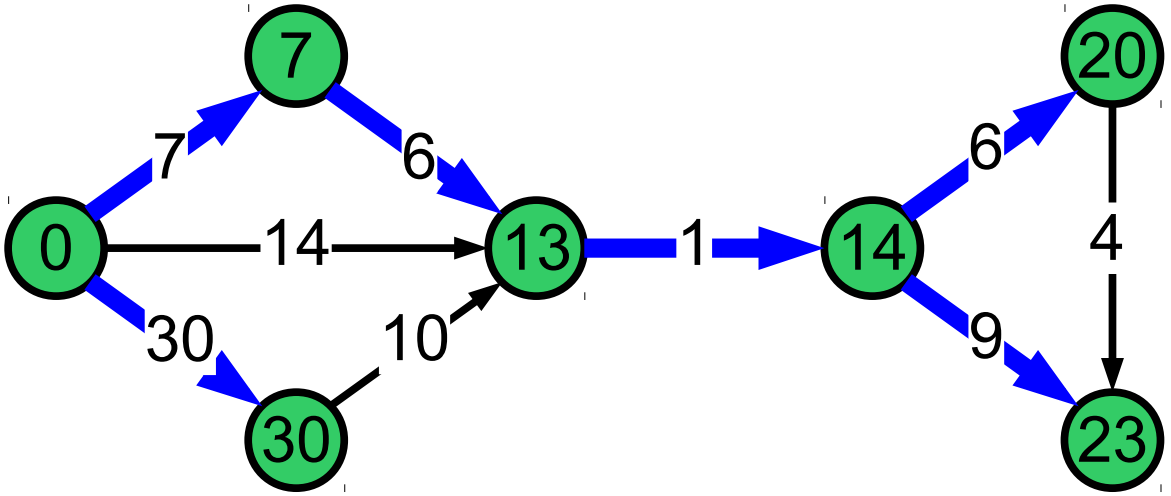
Esempio



D =

	A	***B***	***C***	***D***	***E***	***F***	***G***
A	0	7	14	30	Inf	Inf	Inf
B	Inf	0	6	Inf	Inf	Inf	Inf
C	Inf	Inf	0	Inf	1	Inf	Inf
D	Inf	Inf	10	0	Inf	Inf	Inf
E	Inf	Inf	Inf	Inf	0	6	9
F	Inf	Inf	Inf	Inf	Inf	0	4
G	Inf	Inf	Inf	Inf	Inf	Inf	0

Esempio



D =

	A	B	C	D	E	F	G
A	0	7	13	30	14	20	23
B	Inf	0	6	Inf	7	13	16
C	Inf	Inf	0	Inf	1	7	10
D	Inf	Inf	10	0	11	17	20
E	Inf	Inf	Inf	Inf	0	6	9
F	Inf	Inf	Inf	Inf	Inf	0	4
G	Inf	Inf	Inf	Inf	Inf	Inf	0