

Cammini minimi: sequenza di nodi

Cammini minimi

- ▷ Sia $G = (V, E)$ un grafo orientato e pesato
- ▷ $w : E \rightarrow \mathbb{R} \longrightarrow \text{ANCHE NEGATIVI}$
- ▷ Un cammino $s \rightsquigarrow t$ da $s \in V$ a $t \in V$ è una sequenza di k nodi $p = \langle v_1, v_2, \dots, v_k \rangle$ tale che
 1. $v_i \in V$ per $1 \leq i \leq k$
 2. $(v_i, v_{i+1}) \in E$ per $1 \leq i \leq k - 1$
 3. $s = v_1$ e $t = v_k$

Cammini minimi: sequenza di archi

- ▷ Sia $G = (V, E)$ un grafo orientato e pesato
- ▷ $w : E \rightarrow \mathbb{R}$
- ▷ Un cammino $s \rightsquigarrow t$ da $s \in V$ a $t \in V$ è una sequenza di k archi $p = \langle e_1, e_2, \dots, e_k \rangle$ tale che
 1. $e_i \in E$ per $1 \leq i \leq k$
 2. $e_1 = (s, \cdot)$ e $e_k = (\cdot, t)$
 3. $e_i = (\cdot, v)$ e $e_{i+1} = (v, \cdot)$ per $1 \leq i \leq k - 1$

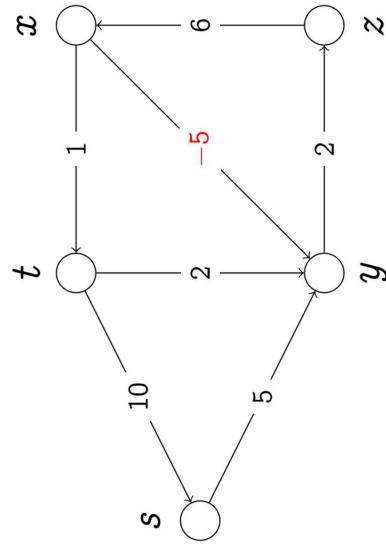
Archi e cicli di peso negativo

- ▷ Il costo $c(p)$ di un cammino $p = \langle e_1, e_2, \dots, e_k \rangle$ è pari a $c(p) = \sum_{i=1}^k w(e_i)$
- ▷ Indicheremo con $s \rightsquigarrow t$ un cammino da s a t e con $s \rightsquigarrow t$ un cammino di costo minimo da s a t

Archi negativi

Cammini minimi: sorgente singola (single source shortest paths SSSP)

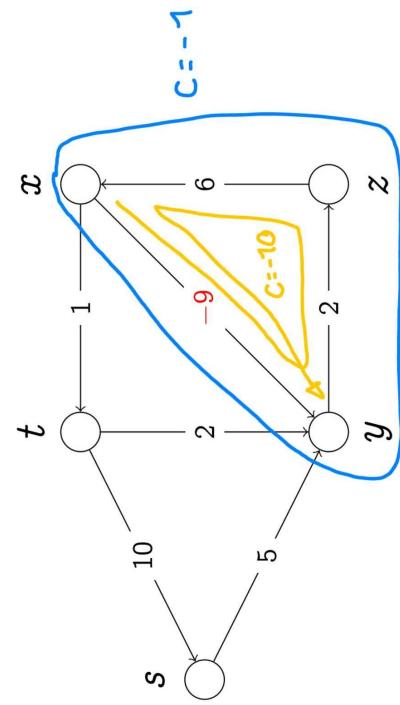
$$s \rightsquigarrow t = \langle s, y, z, x, t \rangle$$



- ▷ Input: un grafo $G = (V, E)$ orientato e pesato e un nodo sorgente $s \in V$
- ▷ Output: insieme dei cammini minimi $P = \{s \rightsquigarrow v : v \in V\}$

Cicli negativi

$$c(\langle s, \overbrace{y, z, x, y, z, x, \dots, y, z, x}, t \rangle) \rightarrow -\infty$$



Cammini minimi: destinazione singola (single destination shortest paths SDSP)

- ▷ Input: un grafo $G = (V, E)$ orientato e pesato e un nodo destinazione $t \in V$
- ▷ Output: insieme dei cammini minimi $P = \{v \rightsquigarrow t : v \in V\}$

Cammini minimi: sorgente singola e destinazione singola (single pair shortest path SPSP)

Cammini minimi: proprietà della sottostruutura ottima

Sottostruutura ottima

Sia

$$p = \langle v_1, \dots, \overbrace{v_i, \dots, v_j}, \dots, v_k \rangle^{p'}$$

- ▷ Input: un grafo $G = (V, E)$ orientato e pesato e una coppia di nodi $s, t \in V$
 - ▷ Output: un cammino minimo $s \rightsquigarrow t$
- $p' = \langle v_i, \dots, v_j \rangle$ è un cammino minimo da v_i a v_j

Cammini minimi: da tutti i nodi a tutti i nodi (all pairs shortest paths APSP)

Cammini minimi: proprietà della sottostruutura ottima

- Se per assurdo esistesse un cammino $p'' = v_i \rightsquigarrow v_j$ tale che $c(p'') < c(p')$ allora avremmo
- $$c(\langle v_1, \dots, v_{i-1}, p'', v_{j+1}, \dots, v_k \rangle) < c(p)$$
- che è un assurdo visto che p è per ipotesi un cammino minimo

- ▷ Input: un grafo $G = (V, E)$ orientato e pesato
- ▷ Output: insieme dei cammini minimi $P = \{s \rightsquigarrow t : s, t \in V\}$

Cammini minimi: cicli

Cammini minimi: cicli

Un cammino p contiene un ciclo se e solo se

$$p = \langle \dots, \overbrace{\widehat{v, \dots, v}}^{p'}, \dots \rangle$$

Sia $p = \langle \dots, \overbrace{\widehat{v, \dots, v}}^{p'}, \dots \rangle$ un cammino minimo.

Caso 1: $c(p') > 0$. Rimuoviamo p' da p e otteniamo un cammino più corto. Quindi p non è minimo.

Caso 2: $c(p') < 0$. Aggiungiamo copie consecutive di p' a p e otteniamo cammini sempre più corti. In questo caso non esiste un cammino minimo.

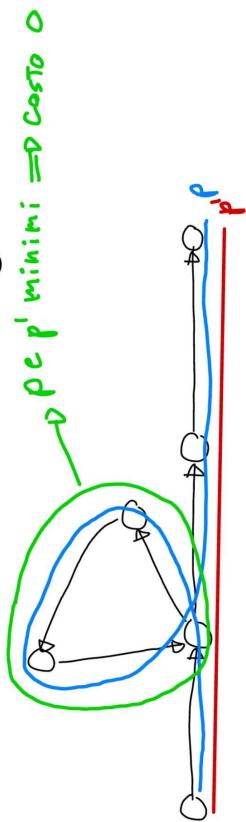
Caso 3: $c(p') = 0$. Rimuoviamo p' da p e otteniamo un cammino della stessa lunghezza. Reiterando il procedimento otteniamo un cammino senza cicli.

Cammini minimi: cicli

Cammini minimi: rappresentazione

Cicli dentro i cammini minimi

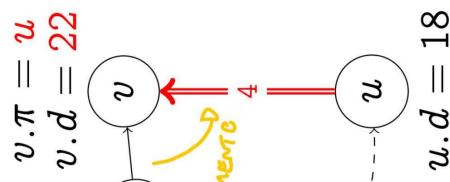
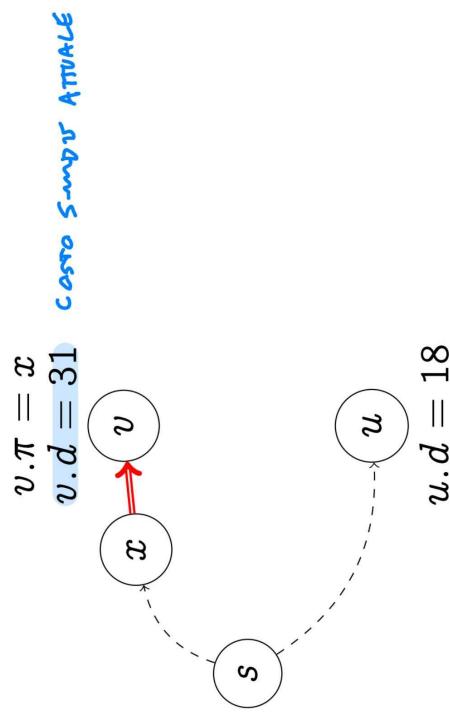
Se esiste un cammino minimo p da s a t allora esiste un cammino minimo p' senza cicli da s a t . Se un cammino minimo da s a t contiene un ciclo allora il costo del ciclo è uguale a 0



Aggiungiamo un campo puntatore π ad ogni nodo del grafo che punterà al nodo padre nell'albero dei cammini minimi

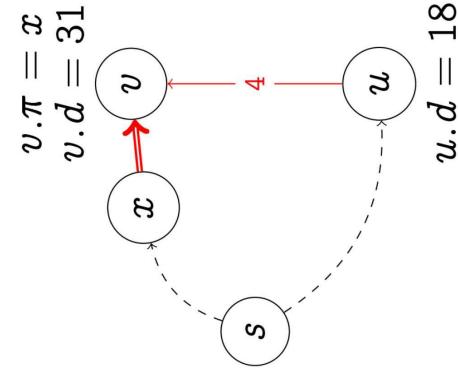
Rilassamento di un arco: esempio

Rilassamento di un arco: esempio



Rilassamento di un arco: esempio

Rilassamento (Relax)



RELAX(u, v, w) *non* **peso**

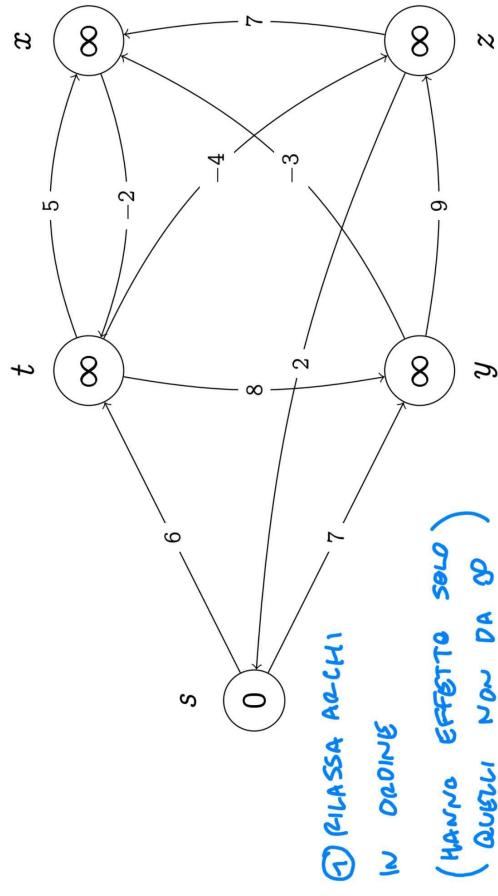
- 1 if $v.d > u.d + w(u, v)$
- 2 $v.d = u.d + w(u, v)$
- 3 $v.\pi = u$

Inizializzazione

Algoritmo di Bellman-Ford: esempio

$t \rightarrow x, t \rightarrow y, t \rightarrow z, x \rightarrow t, y \rightarrow x$
 $y \rightarrow z, z \rightarrow x, z \rightarrow s, s \rightarrow t, s \rightarrow y$

```
INITIALIZE-SINGLE-SOURCE( $G, s$ )
1   for each vertex  $v \in G.V$ 
2      $v.d = \infty$ 
3      $v.\pi = \text{NIL}$ 
4    $s.d = 0$ 
```

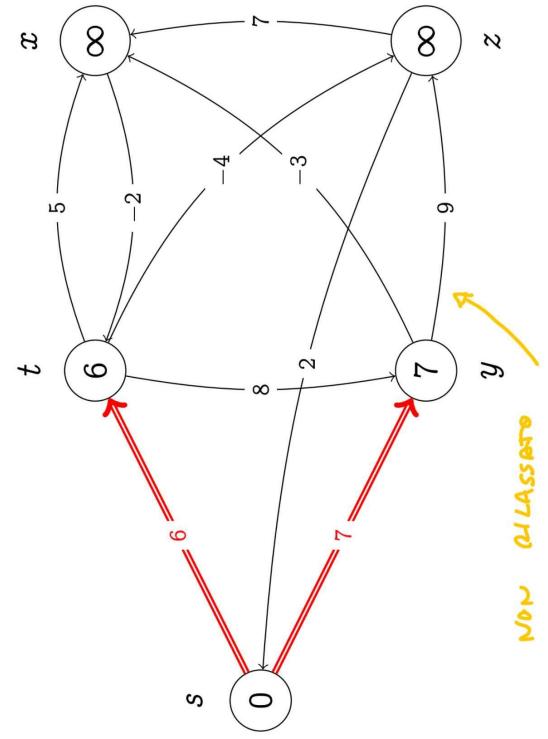


Inizializzazione: costo computazionale

Algoritmo di Bellman-Ford: esempio

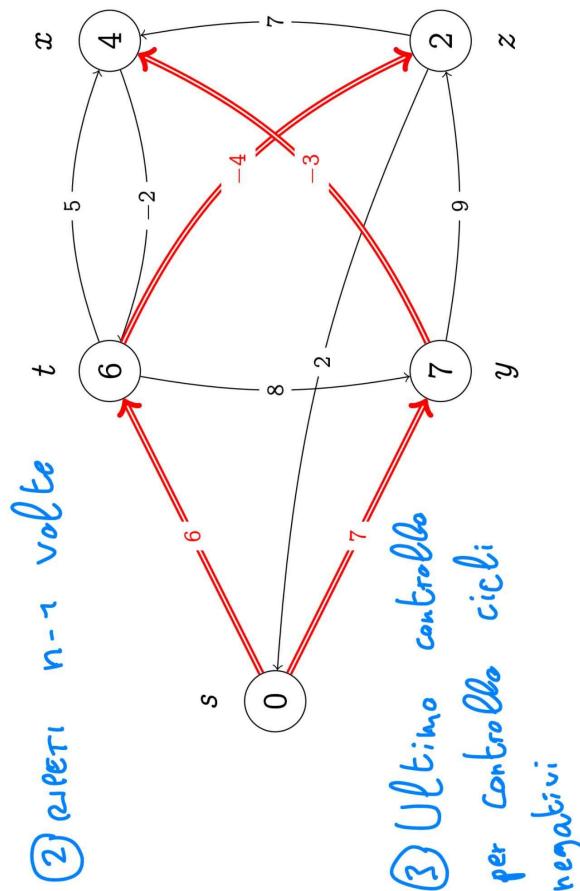
$t \rightarrow x, t \rightarrow y, t \rightarrow z, x \rightarrow t, y \rightarrow x$
 $y \rightarrow z, z \rightarrow x, z \rightarrow s, s \rightarrow t, s \rightarrow y$

Initialize-Single-Source costa $\Theta(n)$



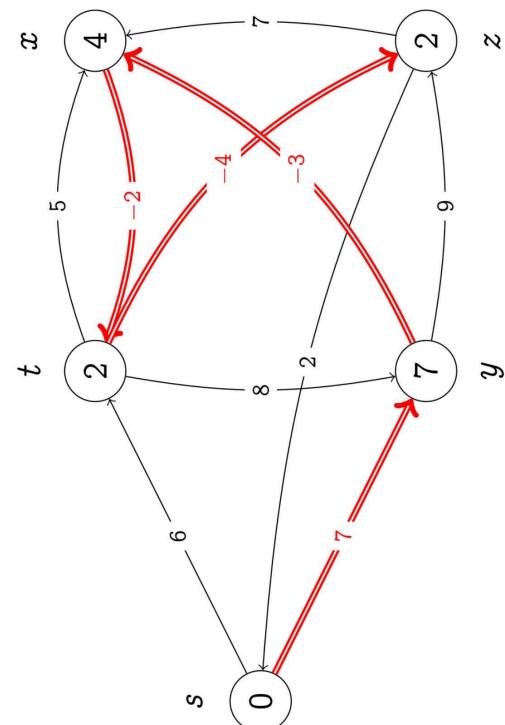
Algoritmo di Bellman-Ford: esempio

$t \rightarrow x, t \rightarrow y, t \rightarrow z, x \rightarrow t, y \rightarrow x$
 $y \rightarrow z, z \rightarrow x, z \rightarrow s, s \rightarrow t, s \rightarrow y$



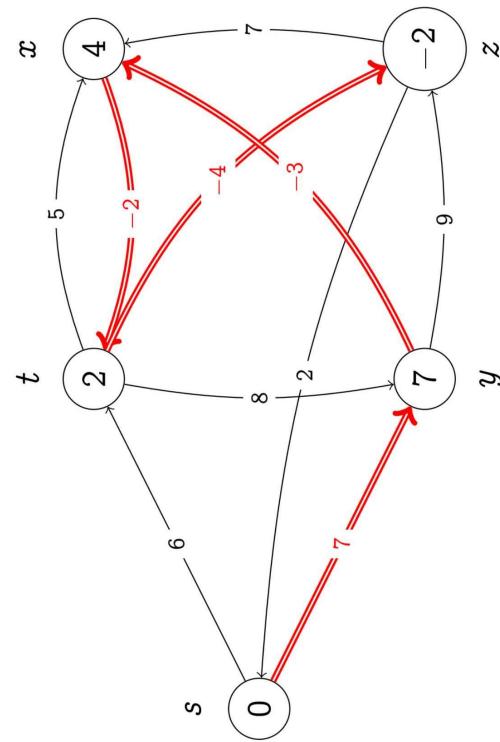
Algoritmo di Bellman-Ford: esempio

$t \rightarrow x, t \rightarrow y, t \rightarrow z, x \rightarrow t, y \rightarrow x$
 $y \rightarrow z, z \rightarrow x, z \rightarrow s, s \rightarrow t, s \rightarrow y$



Algoritmo di Bellman-Ford: esempio

$t \rightarrow x, t \rightarrow y, t \rightarrow z, x \rightarrow t, y \rightarrow x$
 $y \rightarrow z, z \rightarrow x, z \rightarrow s, s \rightarrow t, s \rightarrow y$



Prendiamo il cammino minimo



Caso peggiore ha $m = n - 1$ archi

Riforssando ogni volta tutti gli archi anche uno della soluzione ottima verrà rilassata !!!

In particolare, in ordine da s in quanto unico 0

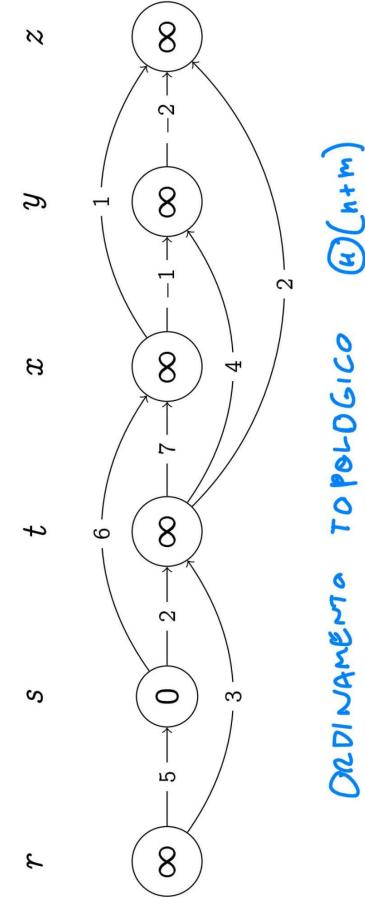
Controllando con un ultimo ciclo se vengono modificati i valori, si trova se è presente un ciclo

SSSP: algoritmo di Bellman-Ford (pesi negativi)

Algoritmo su DAG: esempio

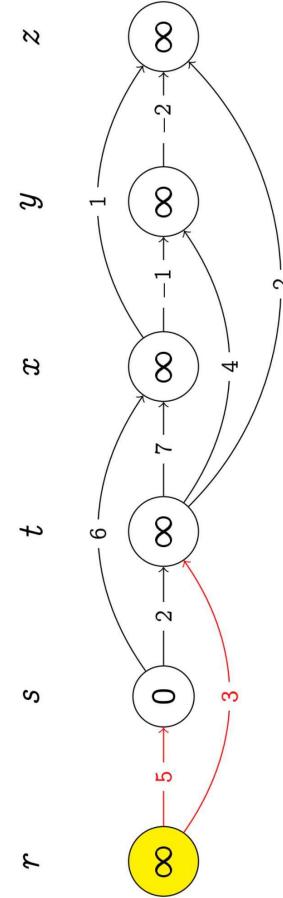
Directed Acyclic Graph

```
BELLMAN-FORD( $G, w, s$ )
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$ 
3    for each edge  $(u, v) \in G.E$ 
4      RELAX( $u, v, w$ )
5  for each edge  $(u, v) \in G.E$ 
6    if  $v.d > u.d + w(u, v)$ 
7      return FALSE
8  return TRUE
```



Algoritmo di Bellman-Ford: costo computazionale

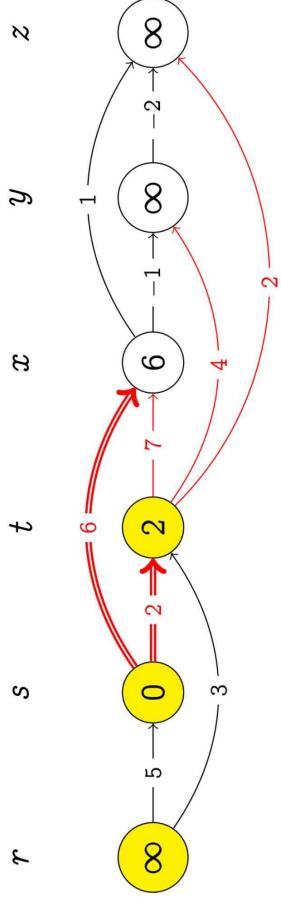
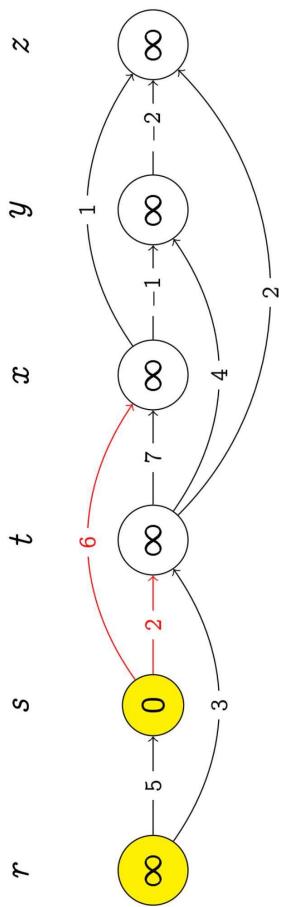
Algoritmo su DAG: esempio



- ▷ G implementato con liste di adiacenza
- ▷ Siano $n = |V|$ e $m = |E|$
- ▷ Costo di Relax = $\Theta(1)$
- ▷ Passo 1 = $\Theta(n)$
- ▷ Passo 2 - 7 = $\Theta(n.m)$
- ▷ Costo totale = $\Theta(n.m)$

Algoritmo su DAG: esempio

Algoritmo su DAG: esempio

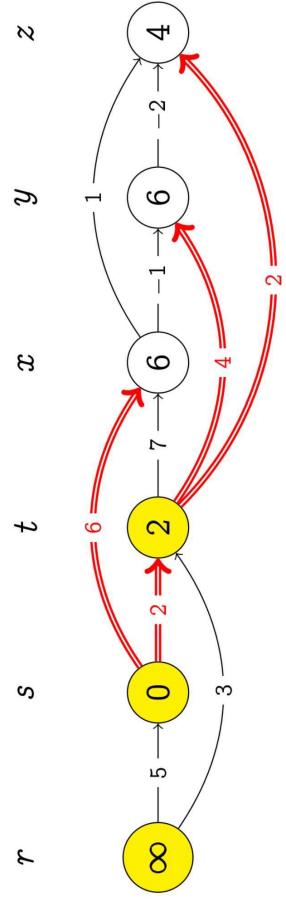
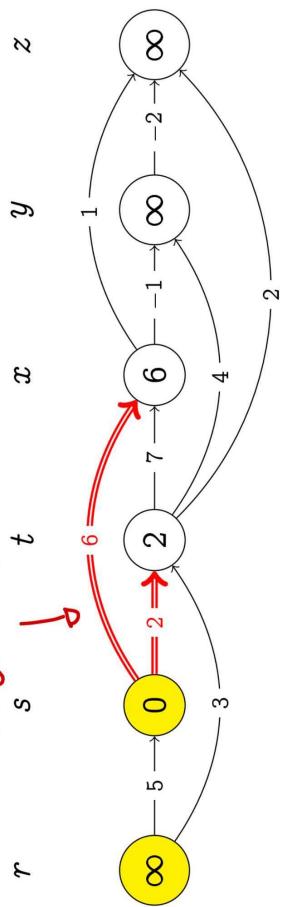


Algoritmo su DAG: esempio

Algoritmo su DAG: esempio

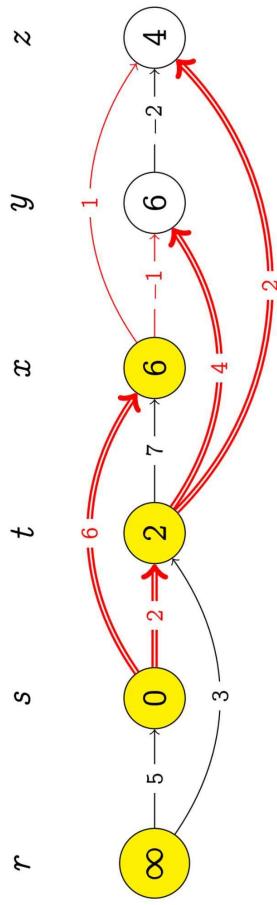
Riassunto in ordine

\Rightarrow CAMMINO MINIMO

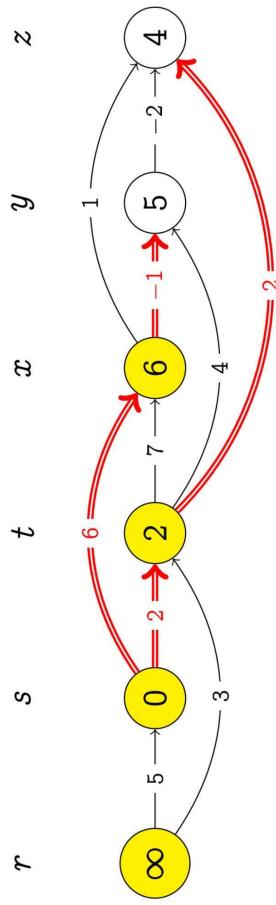


Algoritmo su DAG: esempio

Algoritmo su DAG: esempio



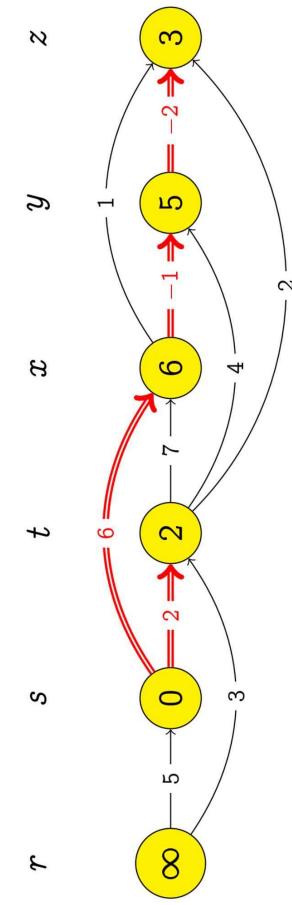
Algoritmo su DAG: esempio



Algoritmo su DAG: esempio

Algoritmo su DAG: esempio

SSSP: grafi orientati aciclici (pesi negativi)



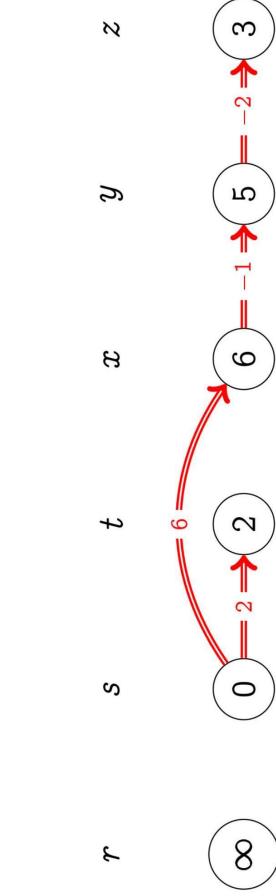
Algoritmo su DAG: esempio

Algoritmo DAG-Shortest-Paths: costo computazionale

```

DAG-SHORTEST-PATHS( $G, w, s$ )
1   topologically sort the vertices of  $G$ 
2   INITIALIZE-SINGLE-SOURCE( $G, s$ )
3   for each vertex  $u$  taken in topologically sorted order
4       for each vertex  $v \in G.Adj[u]$ 
5           RELAX( $u, v, w$ )

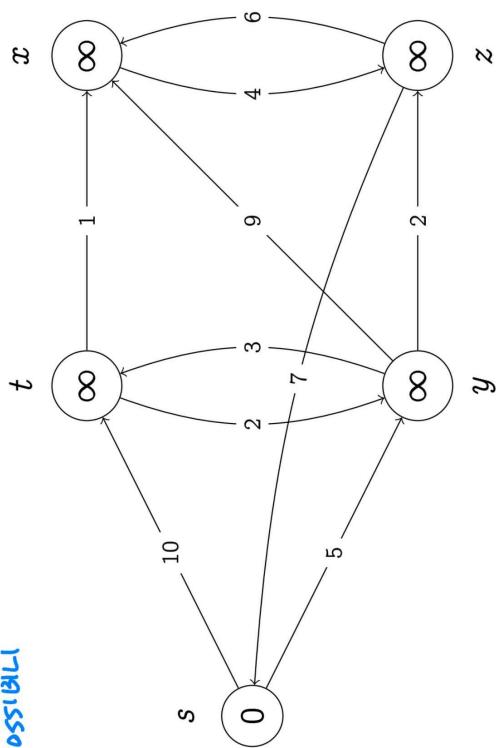
```



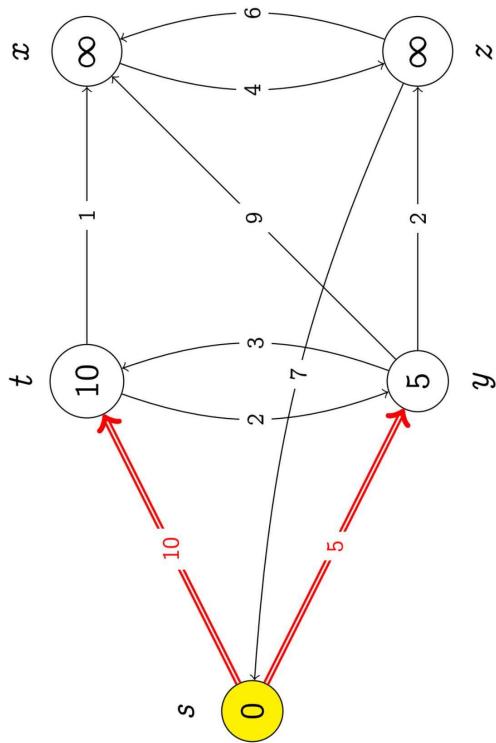
- ▷ G implementato con liste di adiacenza
- ▷ Siano $n = |V|$ e $m = |E|$
- ▷ Costo della singola Relax = $\Theta(1)$
- ▷ Passo 1 = $\Theta(n + m)$
- ▷ Passo 2 = $\Theta(n)$
- ▷ Passo 3 – 5 = $\Theta(m)$
- ▷ Costo totale = $\Theta(m + n)$

Algoritmo di Dijkstra: esempio

Pesi positivi
Cicli possibili

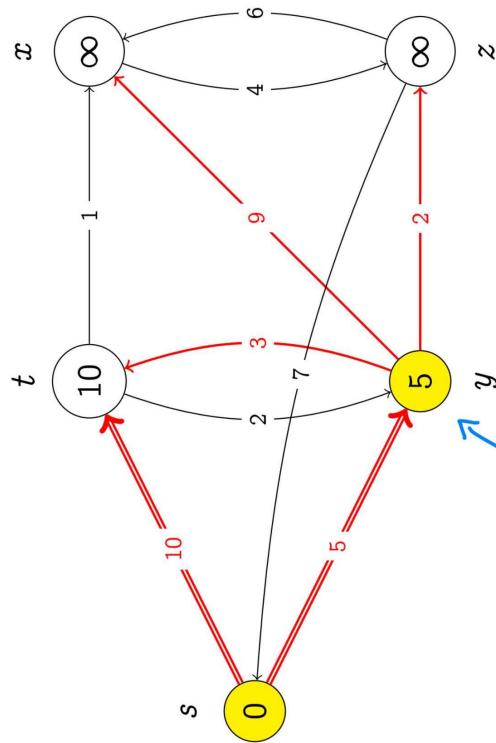
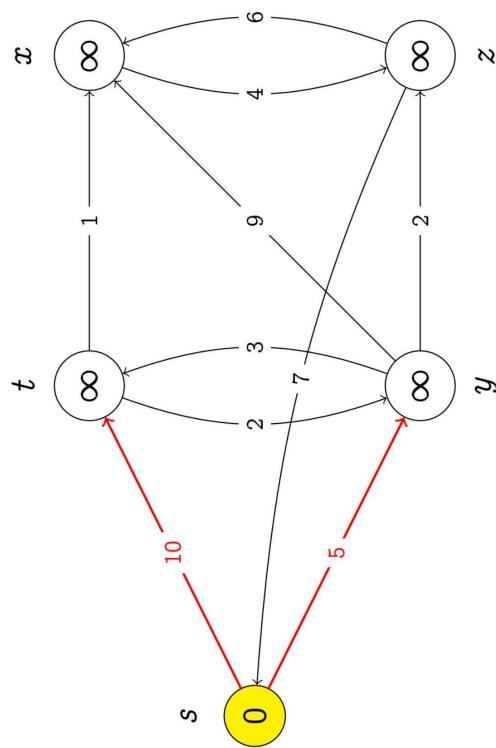


Algoritmo di Dijkstra: esempio



Algoritmo di Dijkstra: esempio

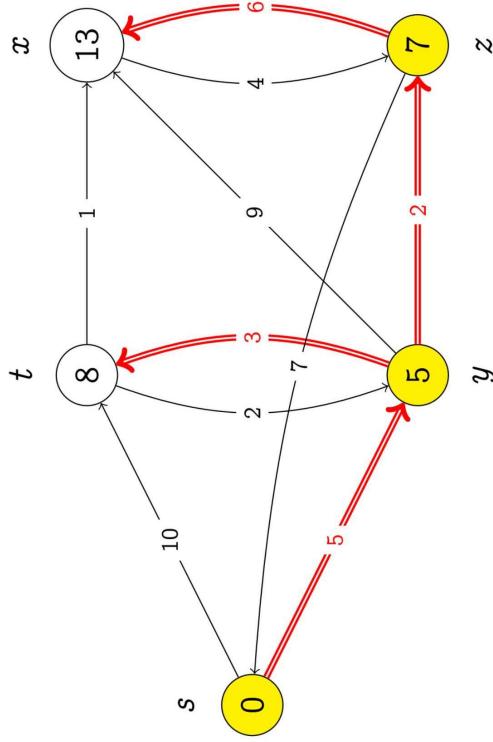
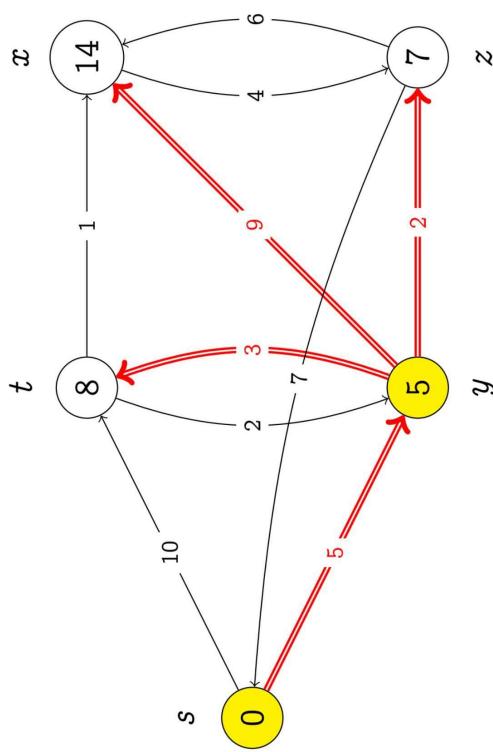
Algoritmo di Dijkstra: esempio



MIN TEA BIANCHI

Algoritmo di Dijkstra: esempio

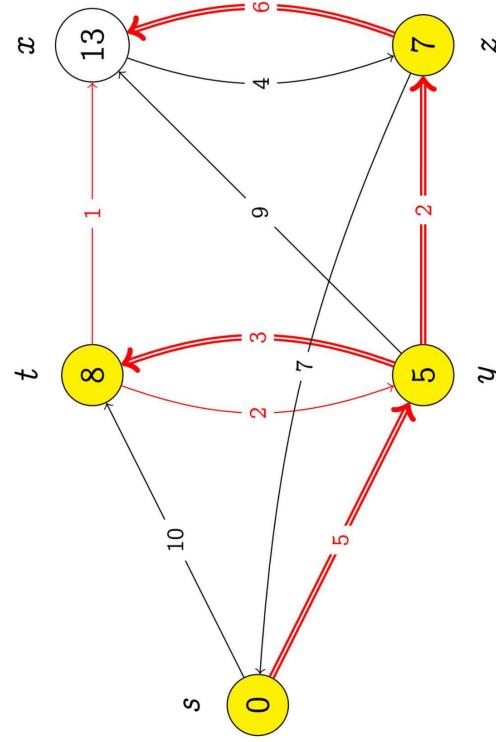
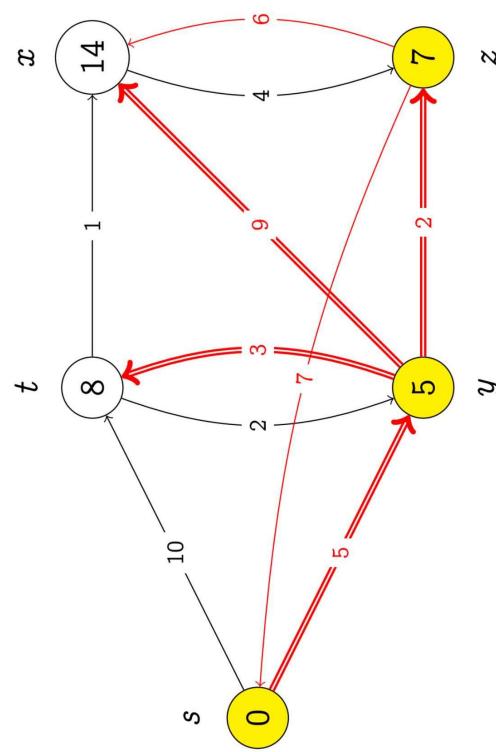
Algoritmo di Dijkstra: esempio



Algoritmo di Dijkstra: esempio

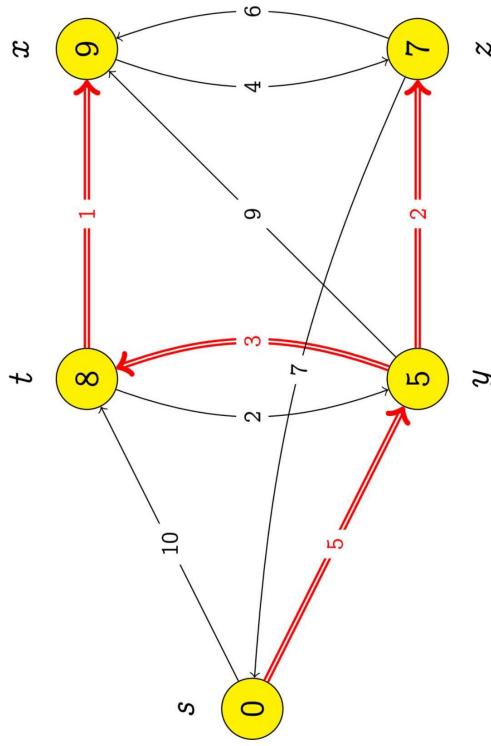
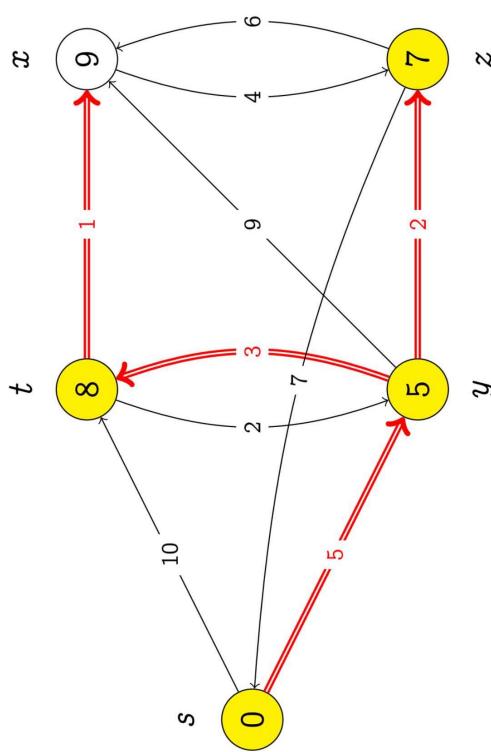
Vista a livello (bst) della via minima a peso

Algoritmo di Dijkstra: esempio



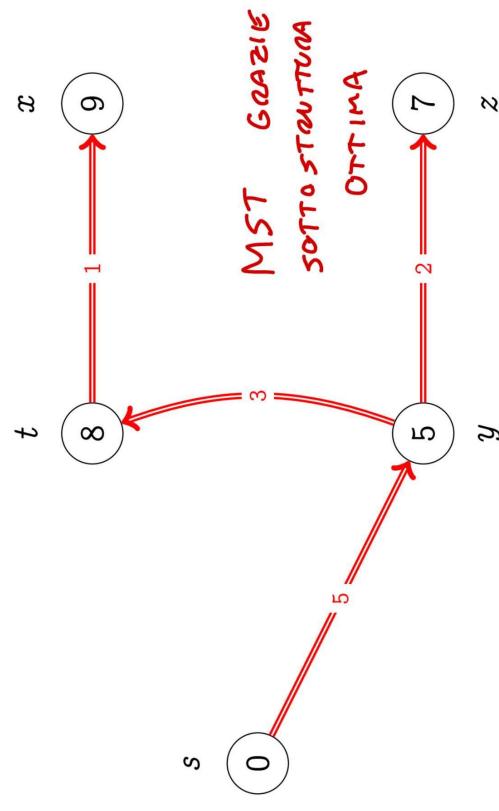
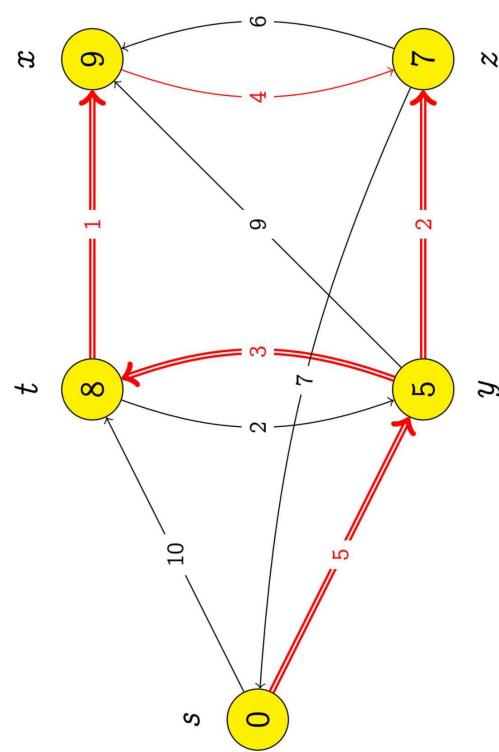
Algoritmo di Dijkstra: esempio

Algoritmo di Dijkstra: esempio



Algoritmo di Dijkstra: esempio

Algoritmo di Dijkstra: esempio



SSSP: algoritmo di Dijkstra (pesi non negativi)

Algoritmo di Dijkstra: costo computazionale

```

DIJKSTRA( $G, w, s$ )
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2   $S = \emptyset$  CIAULLI
3   $Q = G.V$ 
4  while  $Q \neq \emptyset$ 
5     $u = \text{EXTRACT-MIN}(Q)$  n volte
6     $S = S \cup \{u\}$ 
7    for each vertex  $v \in G.Adj[u]$ 
8      RELAX( $u, v, w$ ) n volte
 $\hookrightarrow$  intuitivo ( $\tau_{\text{verso}} \text{ per coerenza}$ )

```

Algoritmo di Dijkstra: costo computazionale

$\triangle G$ implementato con liste di adiacenza
 \triangle Siano $n = |V|$ e $m = |E|$
 $\triangle Q$ implementato come **vettore ordinato**
 \triangle Costo della singola Relax = $\Theta(1)$
 \triangle Costo della singola Extract-Min = $\Theta(1)$
 \triangle Passo 1 = $\Theta(n)$
 \triangle Passo 2 = $\Theta(1)$
 \triangle Passo 3 = $\Theta(n)$
 \triangle Passo 4 - 8 = $\Theta(n^2)$ **in quanto connesso**

$\triangle G$ implementato con liste di adiacenza
 \triangle Siano $n = |V|$ e $m = |E|$
 $\triangle Q$ implementato come **heap**
 \triangle Costo della singola Relax = $\Theta(\log(n))$
 \triangle Costo della singola Extract-Min = $\Theta(\log(n))$
 \triangle Passo 1 = $\Theta(n)$
 \triangle Passo 2 = $\Theta(1)$
 \triangle Passo 3 = $\Theta(n)$
 \triangle Passo 4 - 8 = $\Theta(m \log(n))$
 \triangle Costo totale = $\Theta(m \log(n))$

$m = \Theta(n) \Rightarrow n \log(n)$
 $m = \Theta(n^2) \Rightarrow n^2 \log(n)$

Algoritmo di Dijkstra: costo computazionale