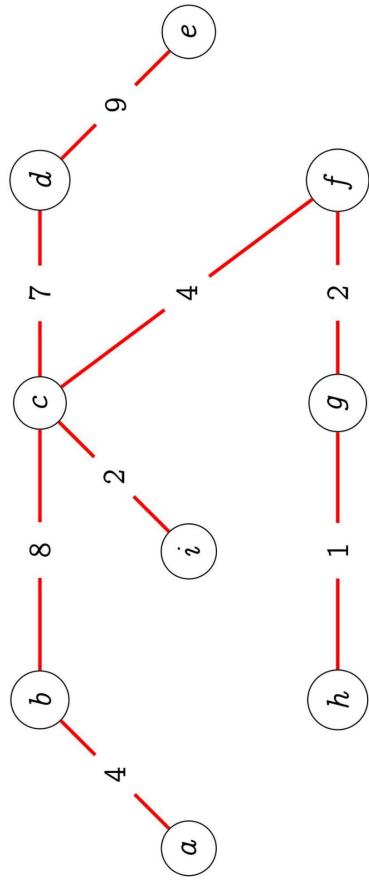
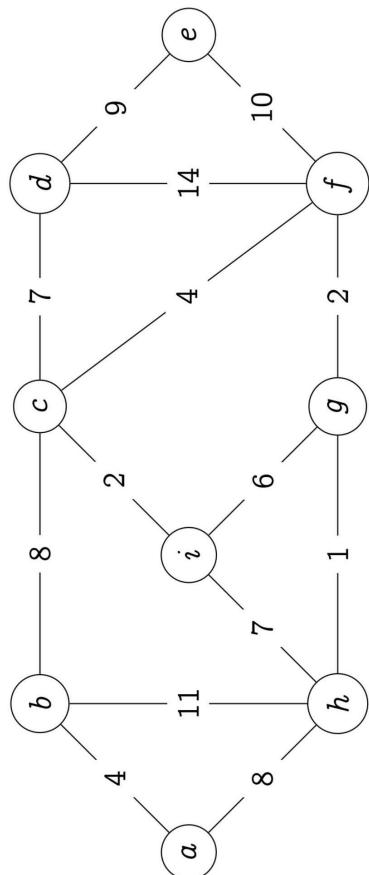


MST: esempio

MST: Esempio

MINIMUM SPANNING TREE



MST: esempio

MST: definizione

- ▷ Sia $G = (V, E)$ un grafo pesato e non orientato
- ▷ Una albero di copertura per G è un albero ottenuto da G selezionando un sottoinsieme di archi di E che toccano tutti i nodi V
- ▷ Il costo di un albero di copertura T è la somma dei pesi degli archi contenuti in T
- ▷ Una albero di copertura minimo è un albero di copertura di costo minimo
- ▷ In generale, possono esistere più MST distinti

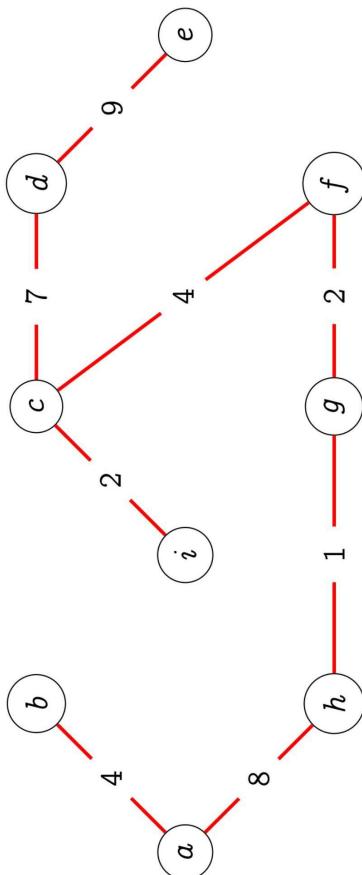
△ Sia $G = (V, E)$ un grafo pesato e non orientato

△ Una albero di copertura per G è un albero ottenuto da G selezionando un sottoinsieme di archi di E che toccano tutti i nodi V

△ Il costo di un albero di copertura T è la somma dei pesi degli archi contenuti in T

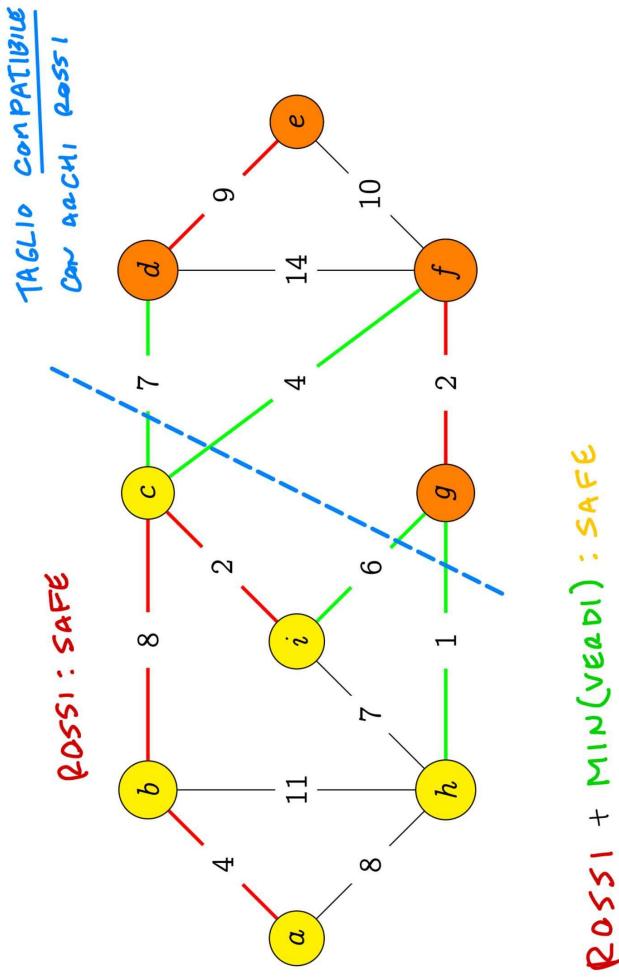
△ Una albero di copertura minimo è un albero di copertura di costo minimo

△ In generale, possono esistere più MST distinti



Archi *safe*

Esempio



Arco *safe*
Sia $G = (V, E)$ un grafo pesato e non orientato.
Sia A un sottoinsieme di archi di E . A è *safe* per G se e solo se è contenuto in almeno un *MST* per G .

Greedy
Sia $G = (V, E)$ un grafo pesato e non orientato.
Sia A un sottoinsieme di archi *safe* per G . Un arco $e \in E$ è *safe* per A se e solo se $A \cup \{e\}$ è *safe* per G .

Archi *safe*: teorema

Dimostrazione Assurdo

Assumiamo che nessun *MST* contenga $A \cup \{(u, v)\}$.
Sia T un *MST* che contiene A . Se aggiungiamo l'arco (u, v) a T creiamo un ciclo.

Visto che $u \in S$ e $v \in V - S$, nel ciclo deve comparire un arco (x, y) che attraversa il taglio.

Inoltre abbiamo che il costo di (u, v) sarà minore o uguale del costo di (x, y) .

Eliminiamo (x, y) da T ed otteniamo un *MST* che contiene (u, v) ottenendo una contraddizione.

Teorema sugli archi *safe*

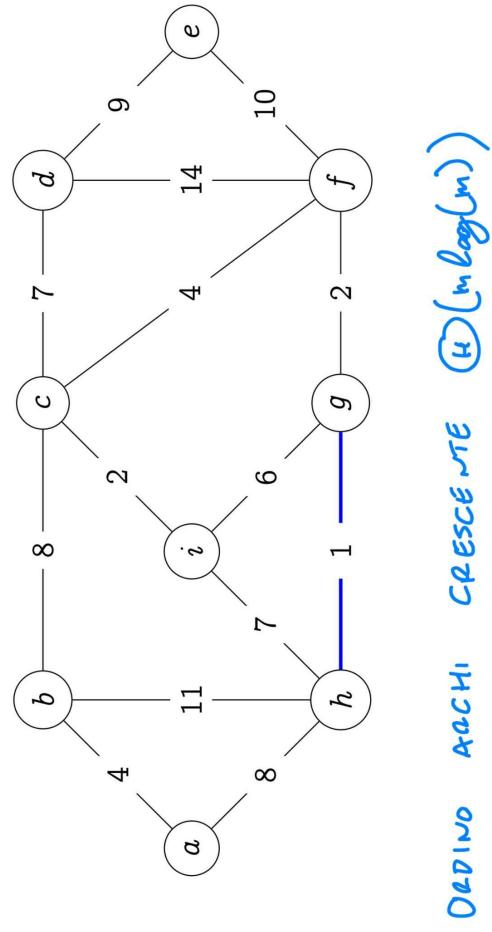
Sia $G = (V, E)$ un grafo连通的, pesato e non orientato. Sia A un insieme di archi *safe* per G . Sia $(S, V - S)$ un taglio che rispetta A e sia (u, v) un arco di costo minimo che attraversa il taglio. Allora (u, v) è un arco *safe* per A .

Algoritmo generico per MST

Numero Archi MST = Nodi - 1

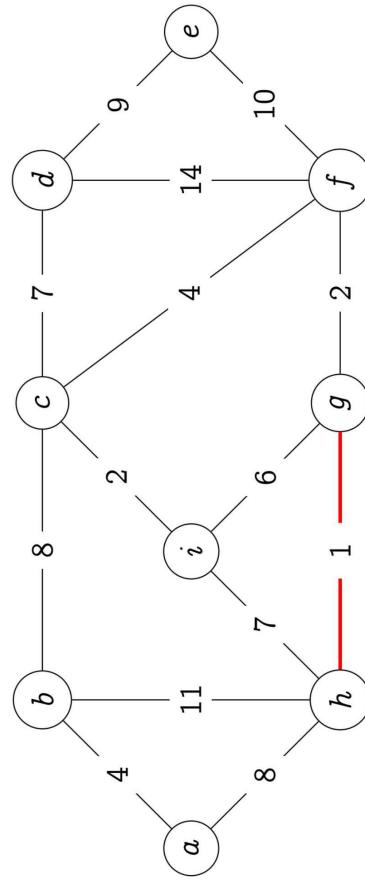
GENERIC-MST(G, w)

- 1 $A = \emptyset$
- 2 while A is not a spanning tree
 - 3 find a safe edge (u, v) for A
 - 4 $A = A \cup \{(u, v)\}$
 - 5 return A

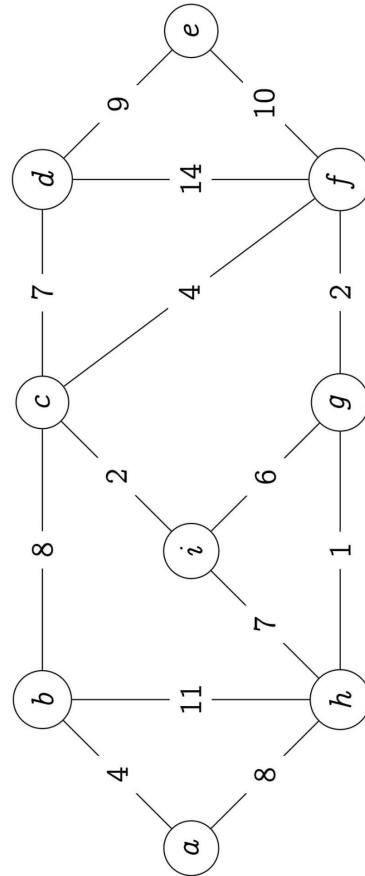


Algoritmo di Kruskal: esempio

Algoritmo di Kruskal: esempio

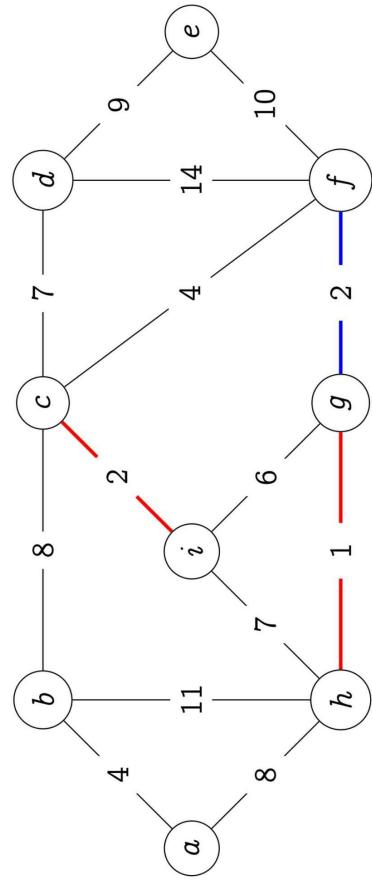
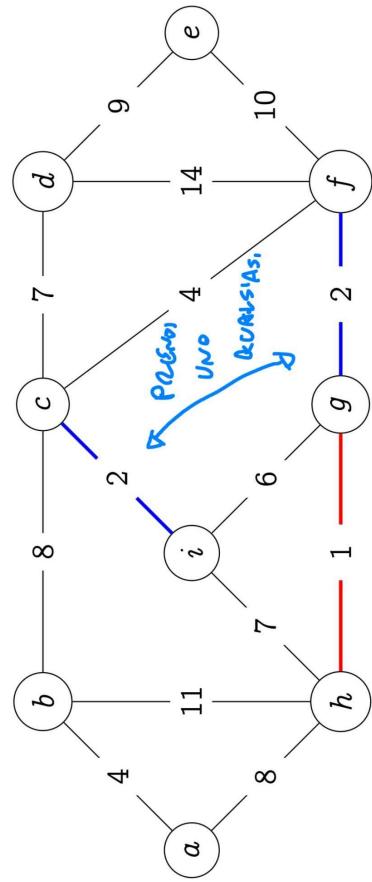


Algoritmo di Kruskal: esempio



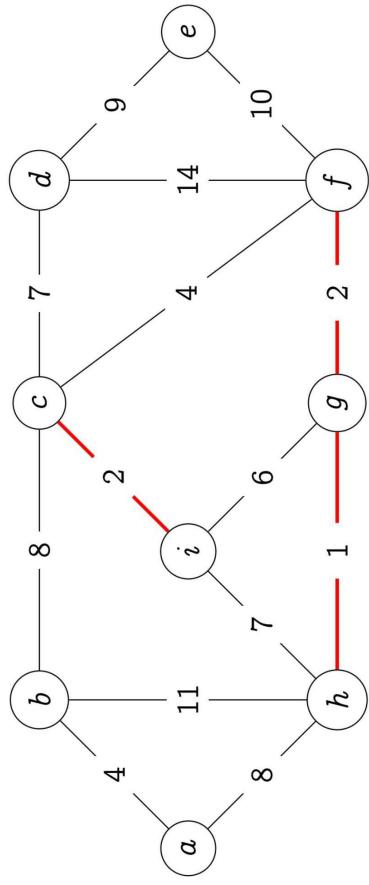
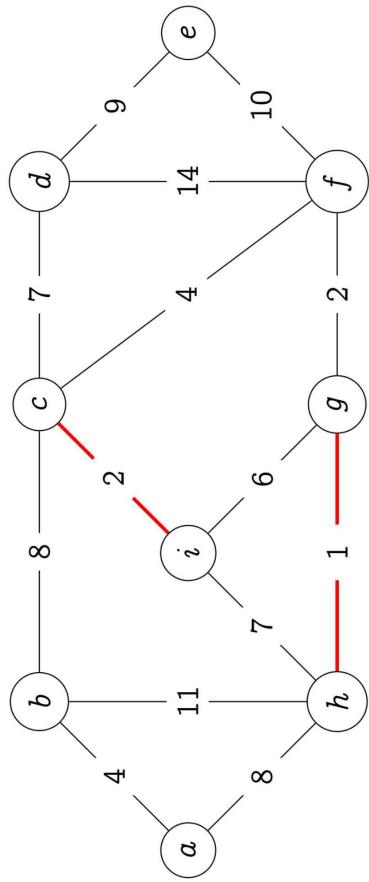
Algoritmo di Kruskal: esempio

Algoritmo di Kruskal: esempio



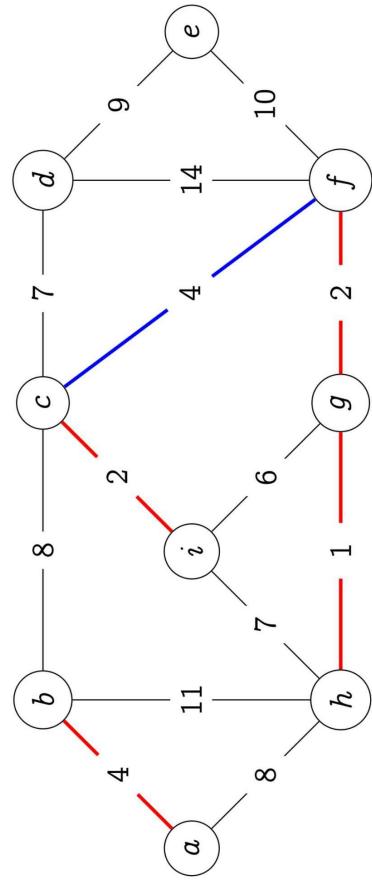
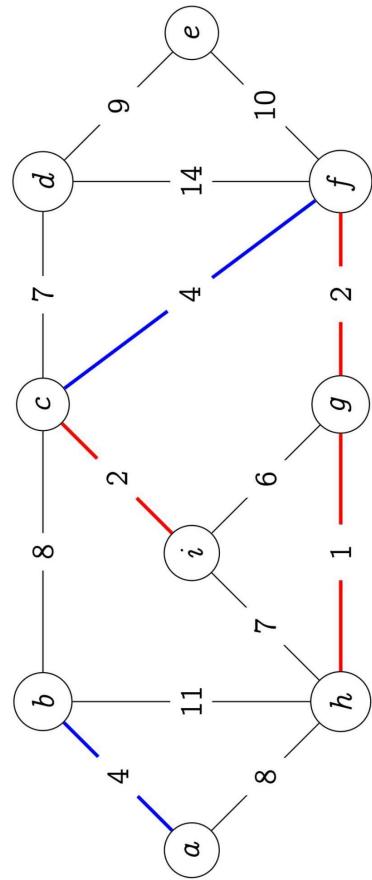
Algoritmo di Kruskal: esempio

Algoritmo di Kruskal: esempio



Algoritmo di Kruskal: esempio

Algoritmo di Kruskal: esempio

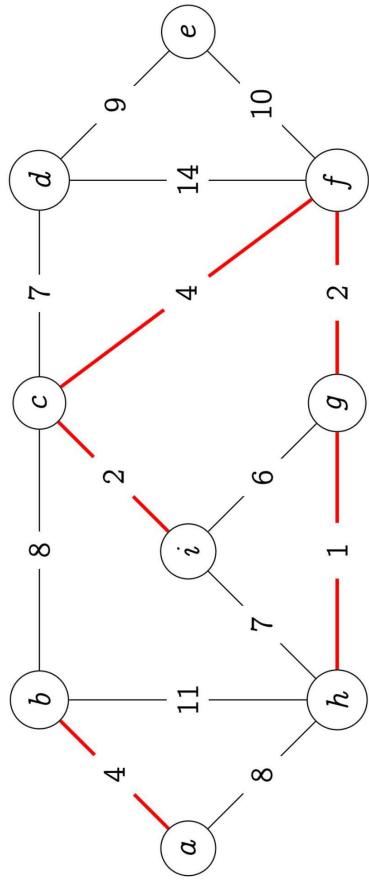
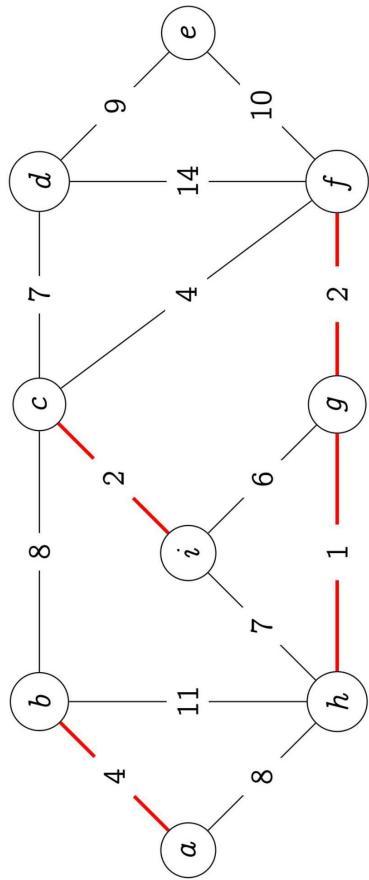


NO CICLO \Rightarrow COLLEGATI POSIZIONI DIVERSE \Rightarrow ESISTE IL TAGLIO

Ogni \Rightarrow CICLO ESTATE \Rightarrow PIÙ PICCOLE DEI PAMENTI

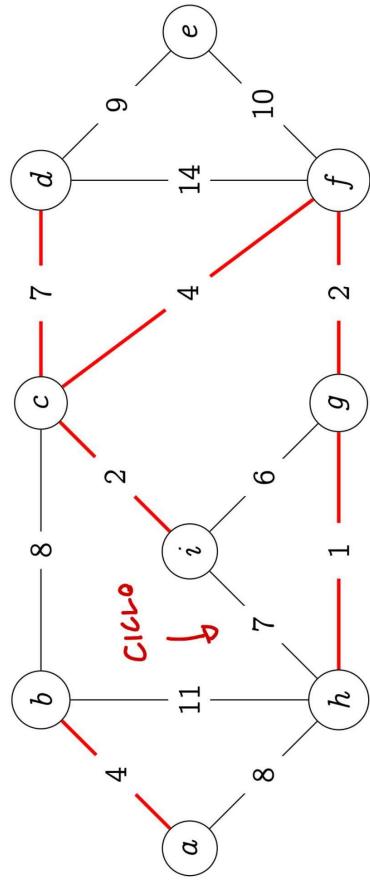
Algoritmo di Kruskal: esempio

Algoritmo di Kruskal: esempio



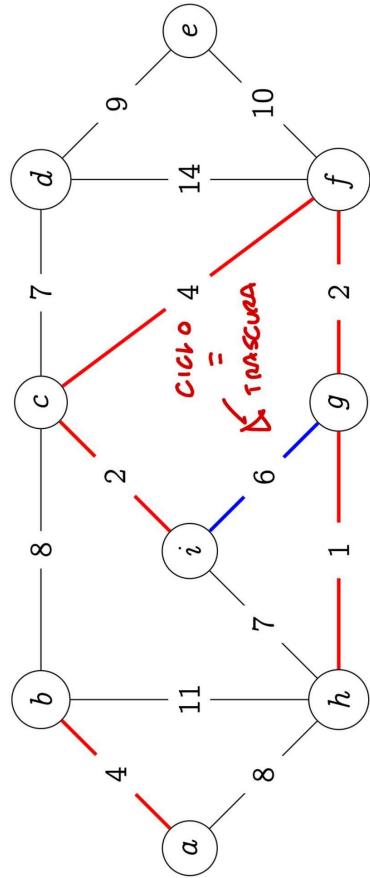
Algoritmo di Kruskal: esempio

Algoritmo di Kruskal: esempio



Algoritmo di Kruskal: esempio

Algoritmo di Kruskal: esempio

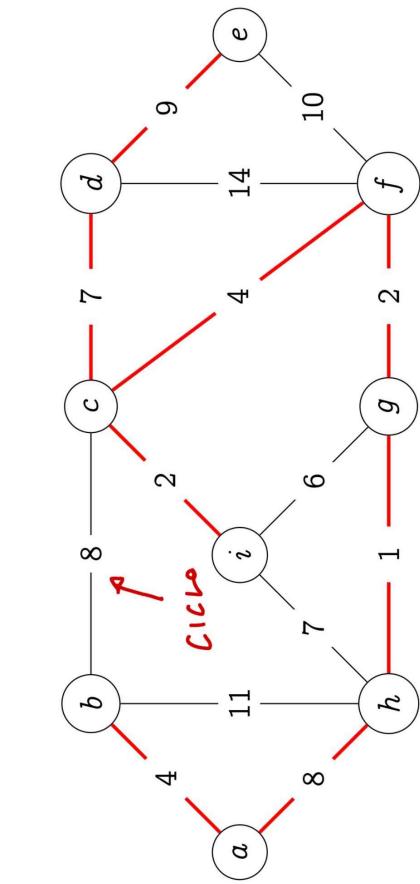


A graph diagram illustrating connections between nodes a through e . The connections and their weights are:

- a to b : weight 4
- a to c : weight 8
- b to c : weight 7
- b to d : weight 8
- c to d : weight 9
- c to e : weight 2
- c to f : weight 14
- d to e : weight 10
- d to f : weight 10
- e to f : weight 10
- f to g : weight 2
- f to h : weight 1
- g to h : weight 6
- h to i : weight 7
- i to j : weight 4
- i to k : weight 8

Algoritmo di Kruskal: esempio

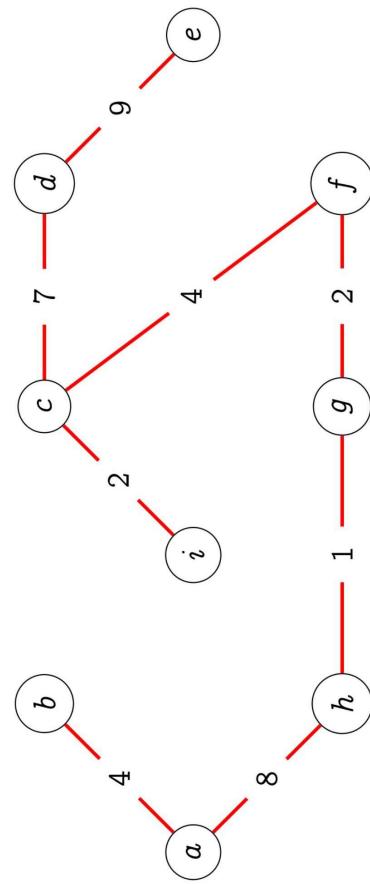
Algoritmo di Kruskal



```
MST-KRUSKAL( $G, w$ )
1    $A = \emptyset$ 
2   for each vertex  $v \in G.V$ 
3     MAKE-SET( $v$ )
4   SORT( $G.E$ ) into nondecreasing order by weight  $w$ 
5   for each edge  $(u, v) \in G.E$ 
6     taken in nondecreasing order by weight
8       if FIND-SET( $u$ )  $\neq$  FIND-SET( $v$ ) stessa classe  $=$  ciclo
9         UNION( $u, v$ )
9   return  $A$ 
```

Algoritmo di Kruskal: esempio

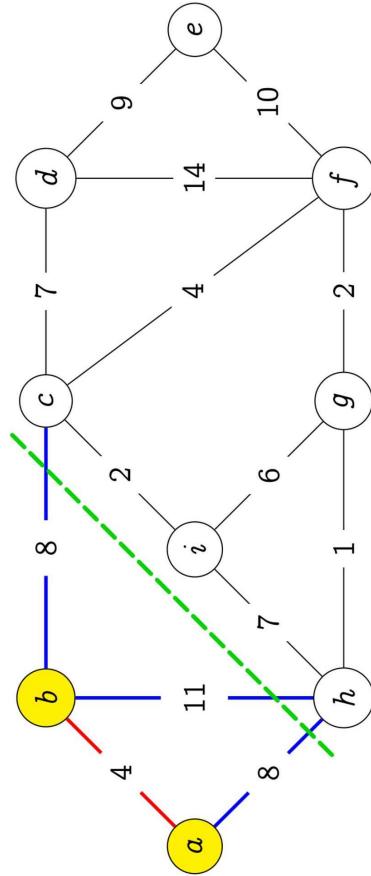
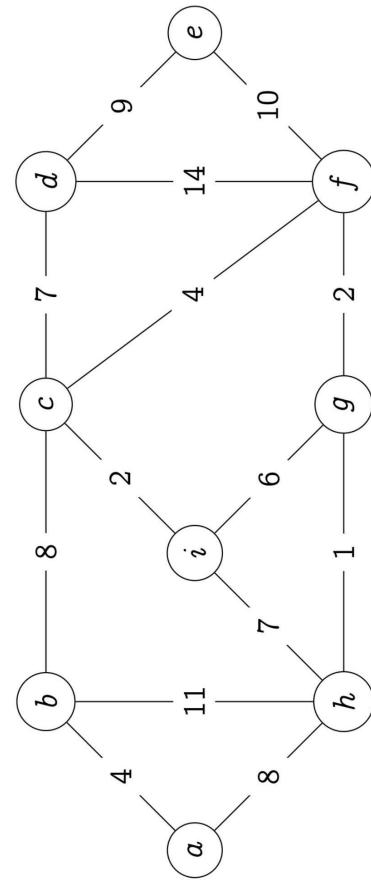
Algoritmo di Kruskal: costo computazionale



- ▷ Sia n il numero di nodi e m il numero di archi
 - ▷ Istruzioni 2 – 3: costo $\Theta(n)$
 - ▷ Istruzione 4: costo $\Theta(m \log(m))$
 - ▷ Istruzioni 5 – 8: costo $\Theta(m \log^*(m))$
 - ▷ Costo totale $\Theta(m \log(m))$

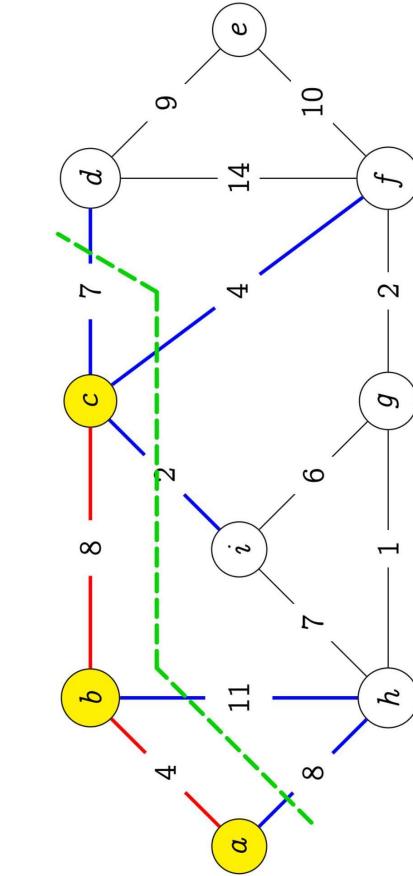
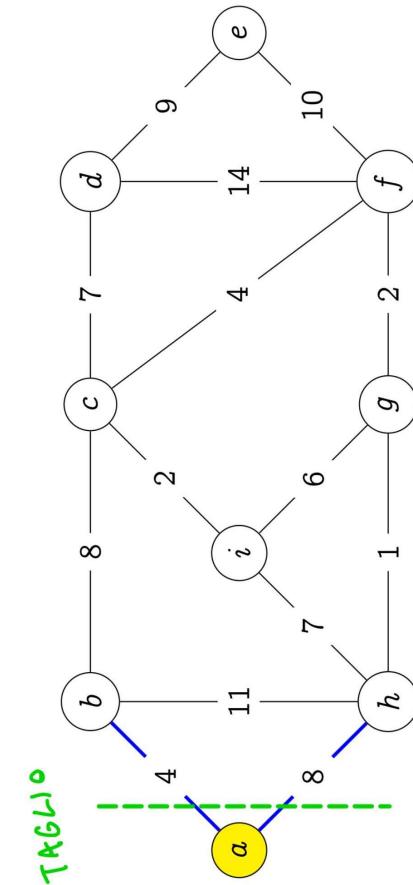
Algoritmo di Prim: esempio

Algoritmo di Prim: esempio



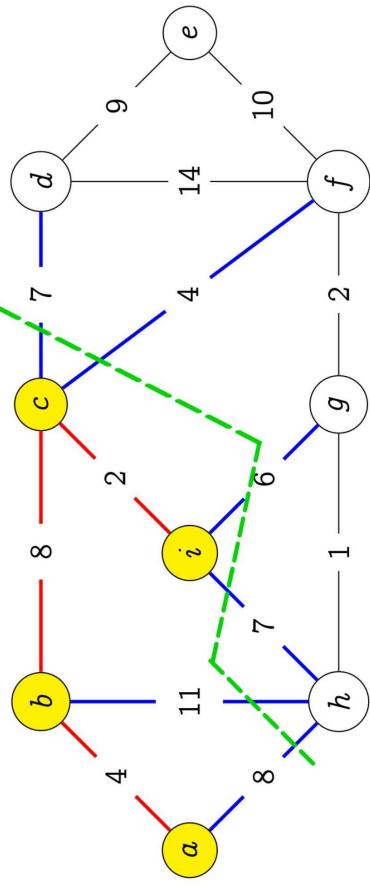
Algoritmo di Prim: esempio

Algoritmo di Prim: esempio

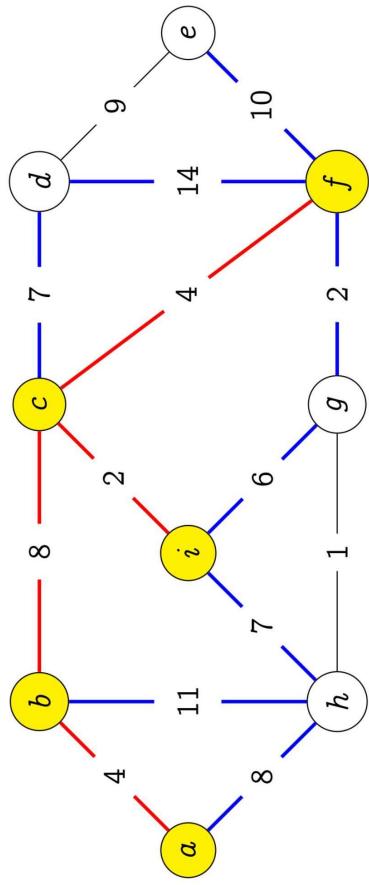


Algoritmo di Prim: esempio

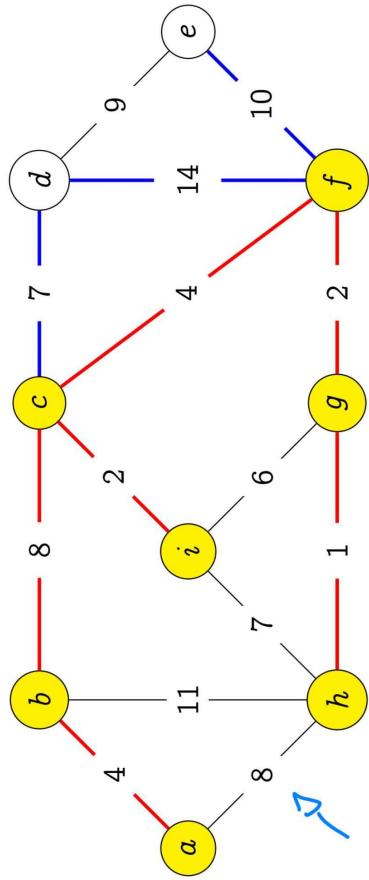
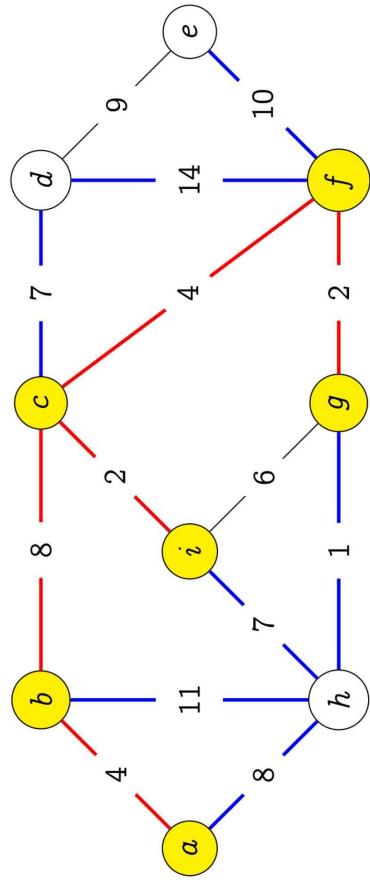
Algoritmo di Prim: esempio



Algoritmo di Prim: esempio



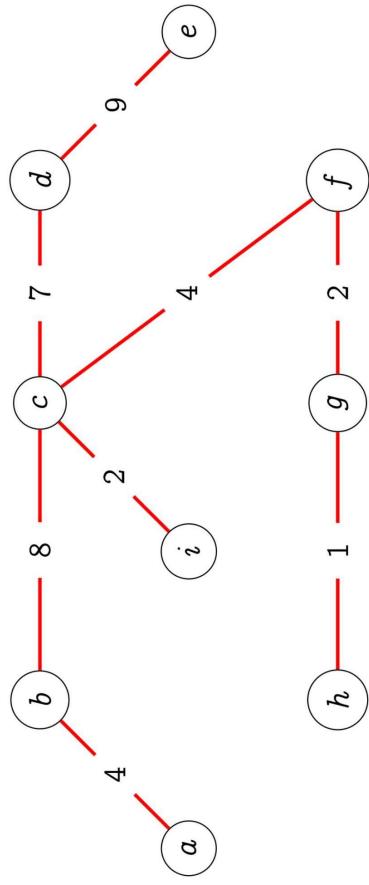
Algoritmo di Prim: esempio



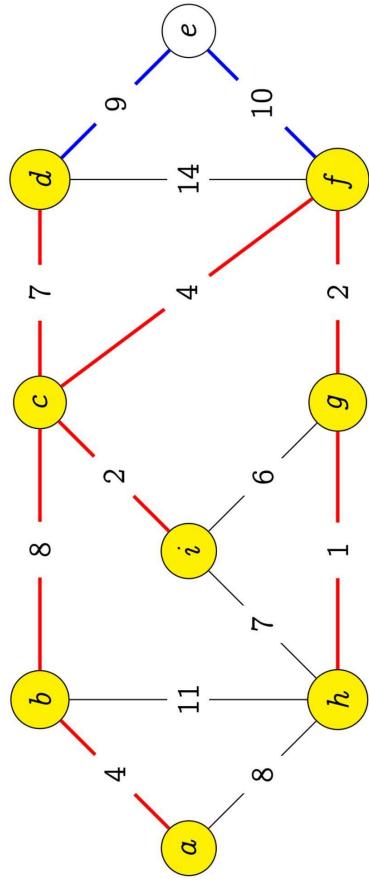
Primos i in
Quattro collegati
A modi gialli

Algoritmo di Prim: esempio

Algoritmo di Prim: esempio



Algoritmo di Prim: esempio



Siccome in genere non è invece di salvare un insieme di archi in blu, salviamo in ogni nodo il costo di inclusione nei gialli:

Ad ogni aggiunta di un nodo giallo, il valore andrà modificato solo ai nodi adiacenti

Algoritmo di Prim

Algoritmo di Prim: costo computazionale

```

MST-PRIM( $G, w, r$ )
    for each vertex  $u \in G.V$ 
         $u.key = \infty$  COSTO AGGIUNTA A GIALLI
         $u.\pi = \text{NIL}$  PADRE NEL PIREO
     $r.key = 0$ 
     $Q = G.V$  NUOVO ANCORA BIANCHI
    while  $Q \neq \emptyset$ 
         $u = \text{EXTRACT-MIN}(Q)$  RICERCA MINIMO
        for each vertex  $v \in G.Adj[u]$  GRAFO CONNESSO = O(m)
            if  $v \in Q$  and  $w(u, v) < v.key$ 
                 $v.\pi = u$ 
                 $v.key = w(u, v)$  MANTIENE STRUTTURA Q
        Possibile sempre  $v$ 
    ( $\Theta(1)$ ) con paragrafici  $Bool$ 

```

Algoritmo di Prim: costo computazionale

- ▷ Q implementata con **vettore ordinato**
 - ▷ Sia n il numero di nodi e m il numero di archi
 - ▷ Istruzioni 1 – 5: costo $\Theta(n)$ **MINIMA**
 - ▷ Istruzione 7: costo $\Theta(n \log(n))$ **$n + \log(n)$**
 - ▷ Istruzioni 8 – 11: costo $\Theta(m \log(n))$
 - ▷ Costo totale $\Theta(m \log(n))$
 - ▷ Q implementata con un **heap**
 - ▷ Sia n il numero di nodi e m il numero di archi
 - ▷ Istruzioni 1 – 5: costo $\Theta(n)$ **MINIMA**
 - ▷ Istruzione 7: costo $\Theta(n \log(n))$ **$n + \log(n)$**
 - ▷ Istruzioni 8 – 11: costo $\Theta(m \log(n))$
 - ▷ Costo totale $\Theta(m \log(n))$
- sempre $m = \Theta(n)$*

Algoritmo di Prim: costo computazionale

SALTA

- ▷ Q implementata con un **heap di Fibonacci**
- ▷ Sia n il numero di nodi e m il numero di archi
 - ▷ Istruzioni 1 – 5: costo $\Theta(n)$
 - ▷ Istruzione 7: costo $\Theta(n \log(n))$
 - ▷ Istruzioni 8 – 11: costo $\Theta(m)$
 - ▷ Costo totale $\Theta(m + n \log(n))$

Algoritmi di Kruskal e di Prim: correttezza

A seconda del rapporto nodi-archi, conviene usare un algoritmo piuttosto che l'altro

La correttezza degli algoritmi di Kruskal e di Prim deriva direttamente dal teorema sugli archi safe