ALGORITMI E STRUTTURE DATI

Prof. Manuela Montangero

A.A. 2022/23

Minimum Spanning Tree (MST)

- Albero di copertura di peso minimo -

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Minimum Spanning Tree

PROBLEMA (Minimum Spanning Tree):

INPUT: Grafo G = (V,E) indiretto connesso

funzione di peso sugli archi di G $c : E \rightarrow R$

OUTPUT: albero di copertura di peso minimo

Albero di copertura T per G=(V,E):
T = (V, E'), tale che E' ⊆ E, è un albero
(quindi |E'| = |V| -1)

Peso di un albero di copertura T=(V,E') per G=(V,E):

$$\sum_{e \in E'} c(e)$$

soluzione ammissibile: albero di copertura per G (spanning tree)

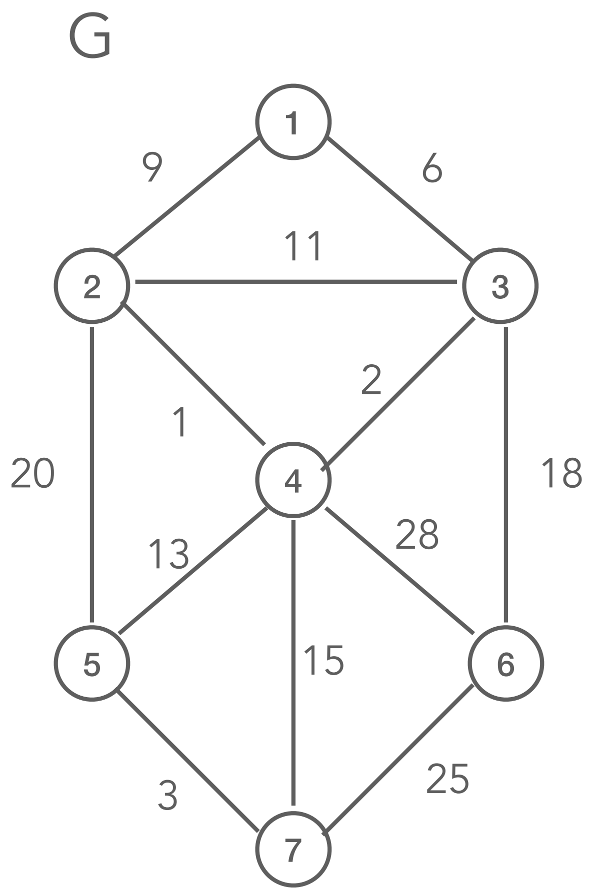
costo soluzione: peso dell'albero di copertura

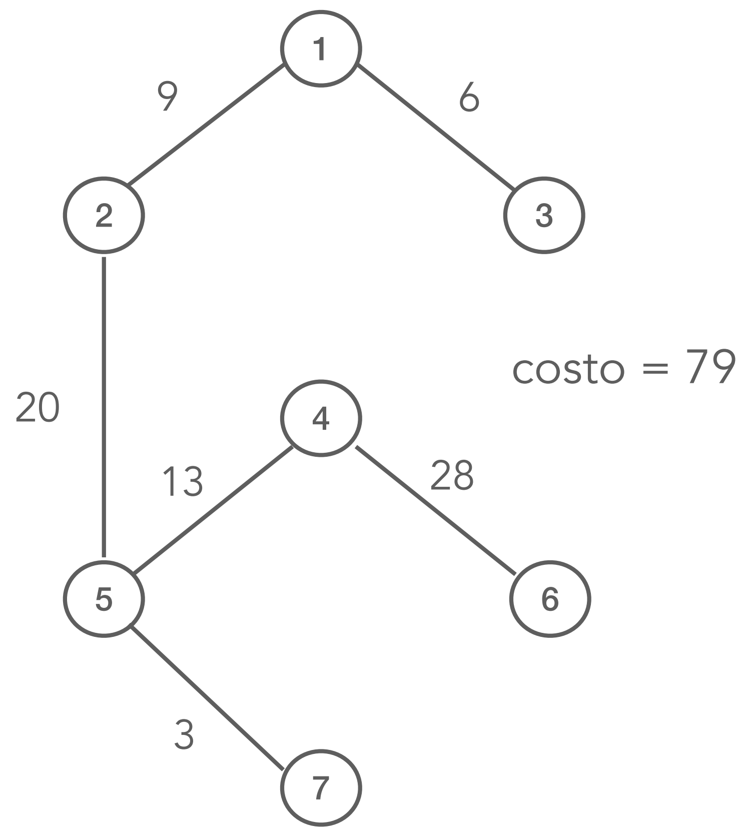
funzione obiettivo: minimo

soluzione ottima: soluzione ammissibile ottima (minimum spanning tree)

Minimum Spanning Tree

ESEMPIO

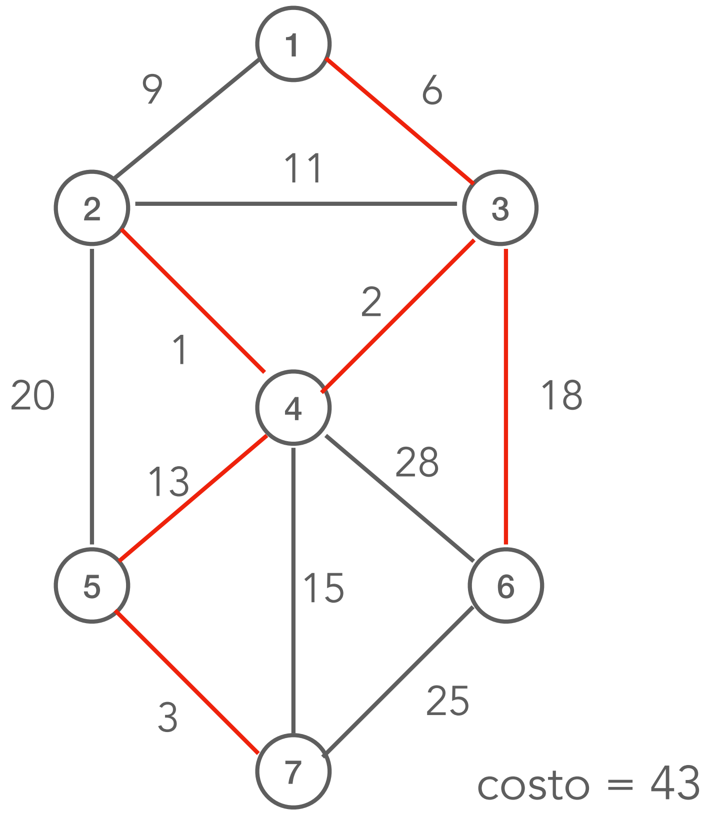




Minimum Spanning Tree

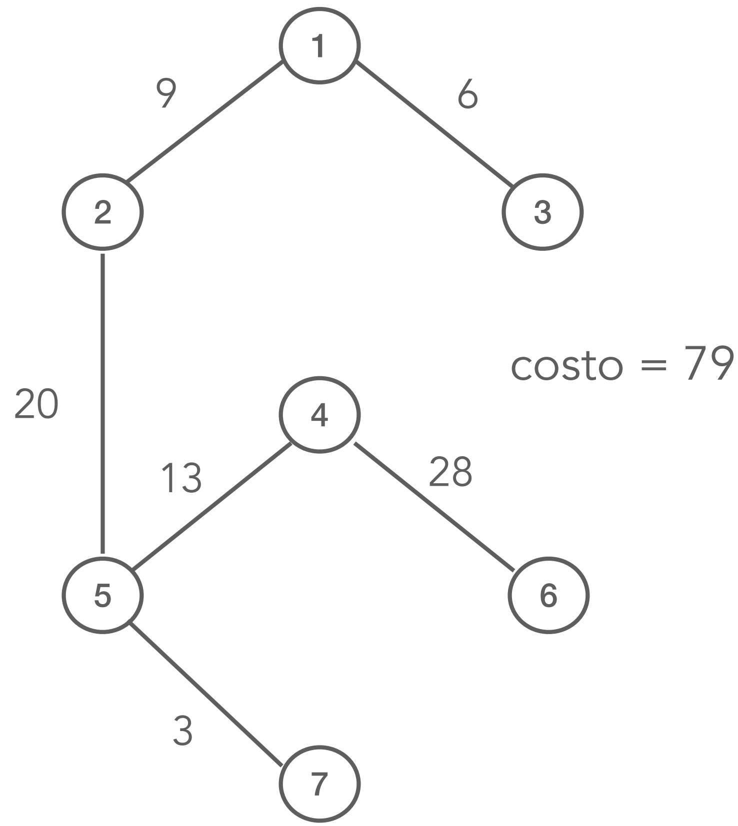
ESEMPIO

G



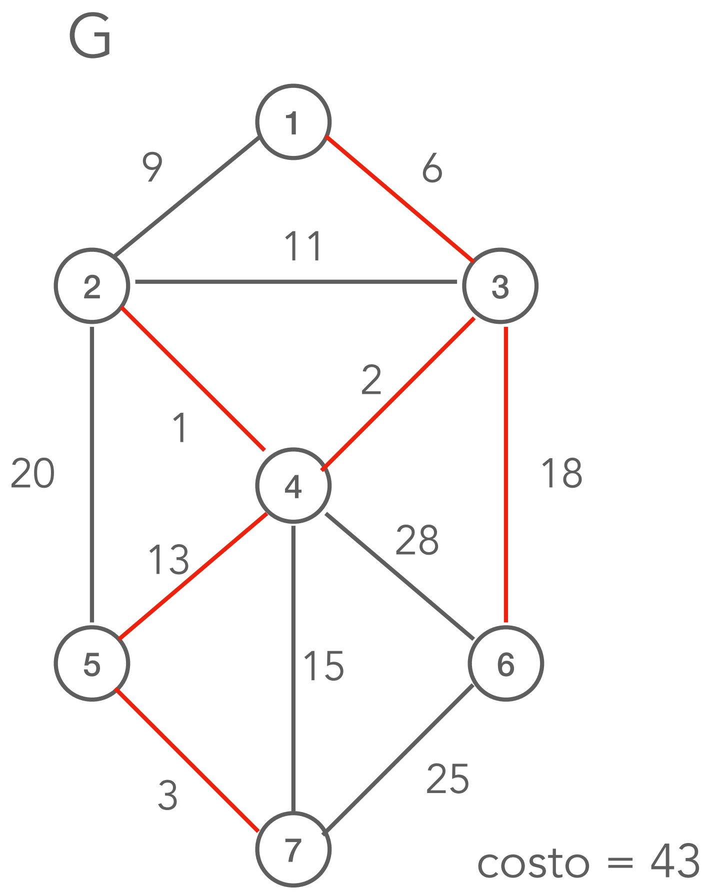
Soluzione ottima

Una soluzione ammissibile



Minimum Spanning Tree

ESEMPIO

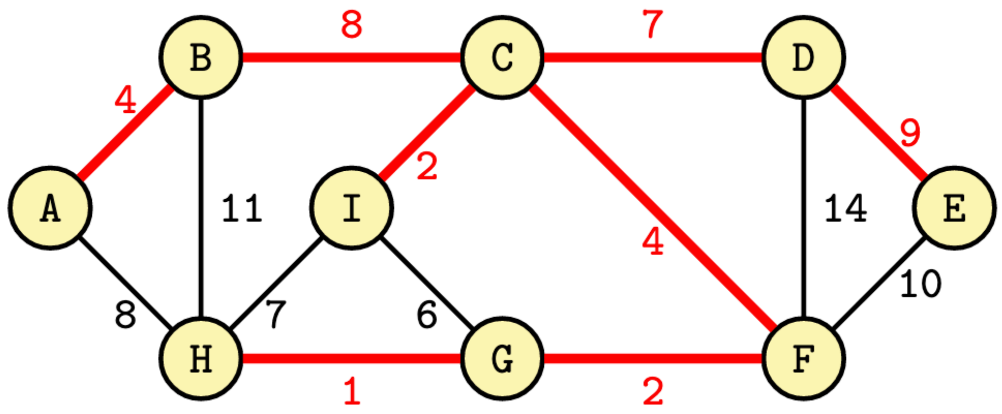


Soluzione ottima

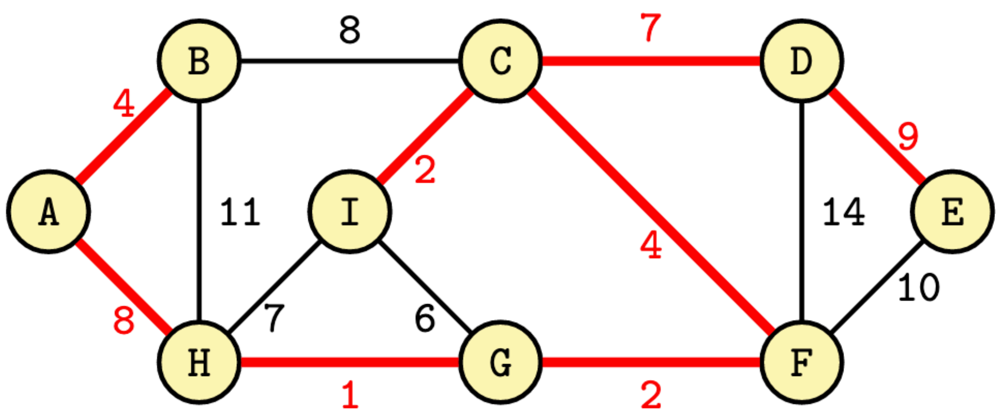
La soluzione ottima può non essere UNICA!

Minimum Spanning Tree

La soluzione ottima può non essere UNICA!



cost = 37



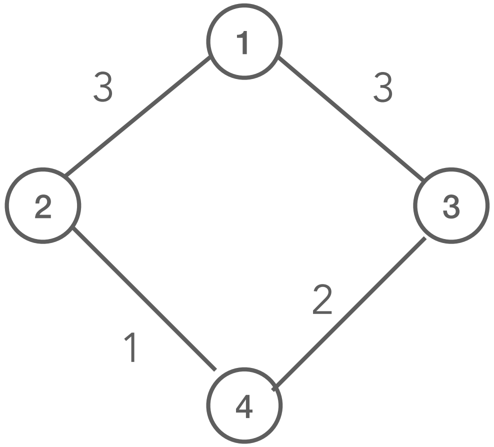
cost = 37

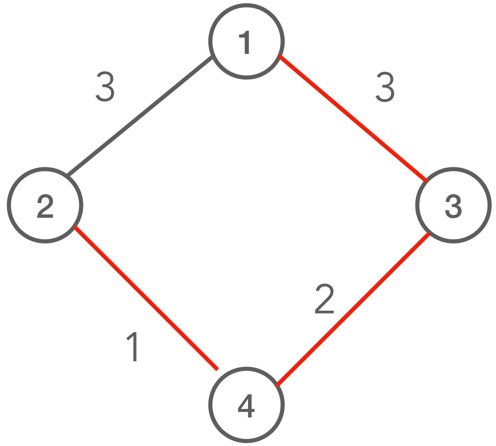
Immagini originali A. Montresor

MST VS cammini minimi

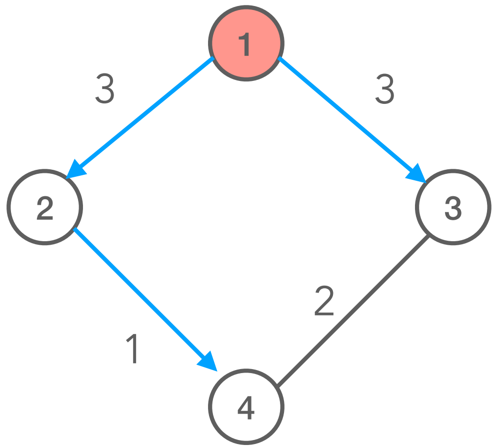
Minimum Spanning Tree e albero dei cammini minimi
NON sono la stessa cosa!

G





MST

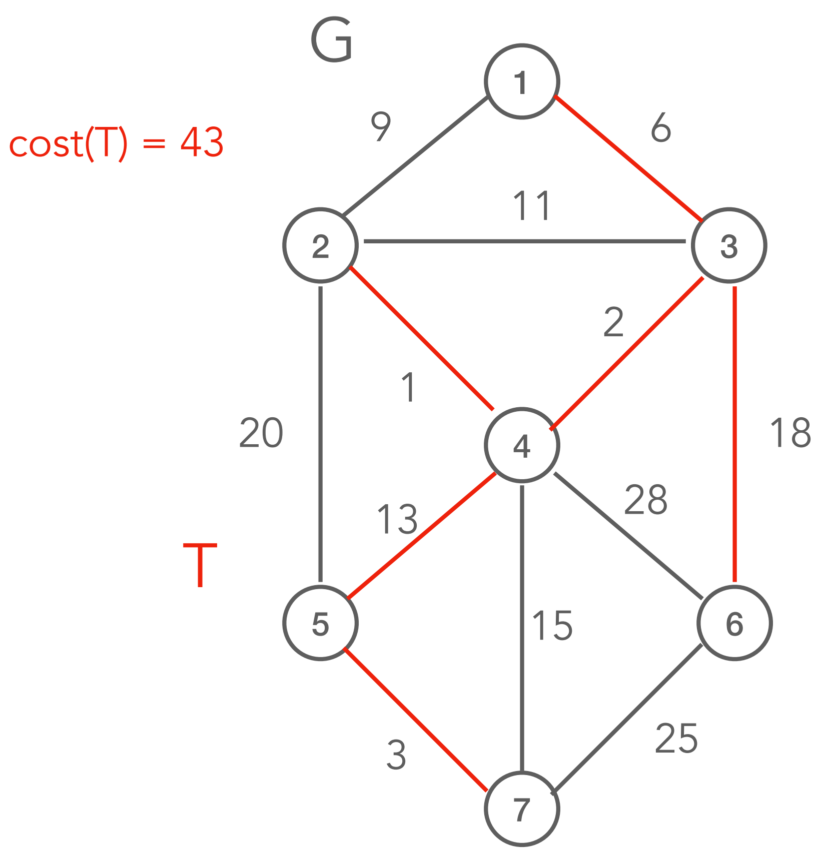


albero cammini
minimi da sorgente 1

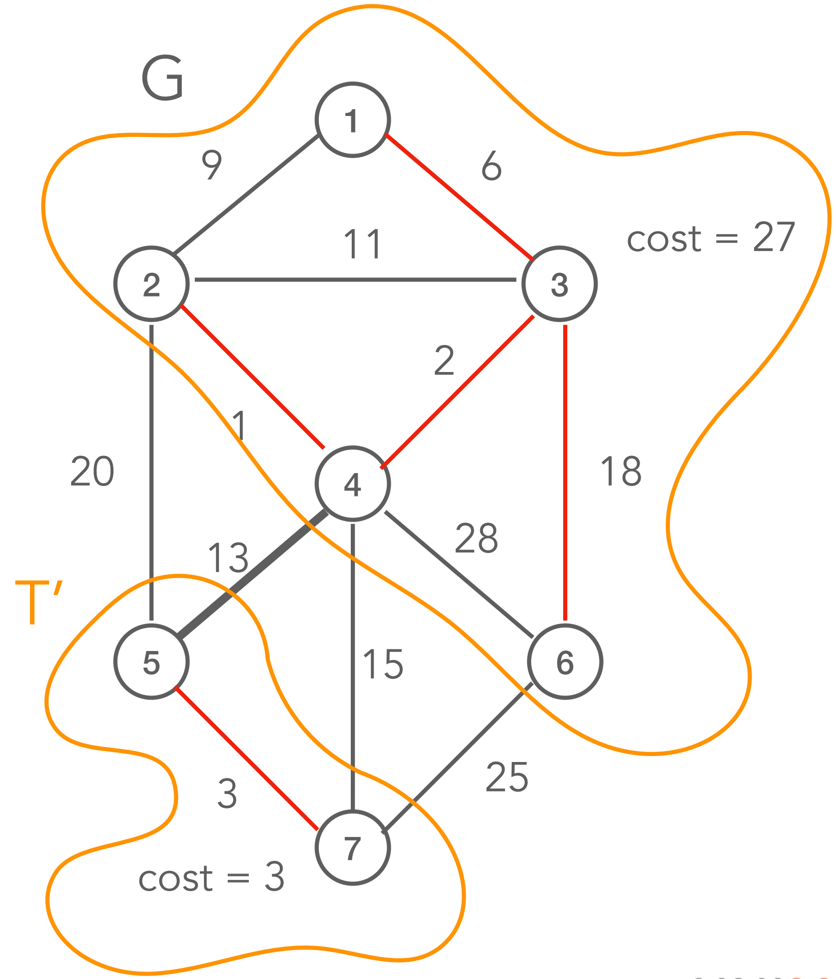
Minimum Spanning Tree

SOTTOSTRUTTURA OTTIMA

Dato un MST T = (V,E') per G=(V,E),
gli alberi in T' = (V, E\{(u,v)}), con (u,v) ∈ E' sono
MST per gli insiemi di vertici connessi in T'



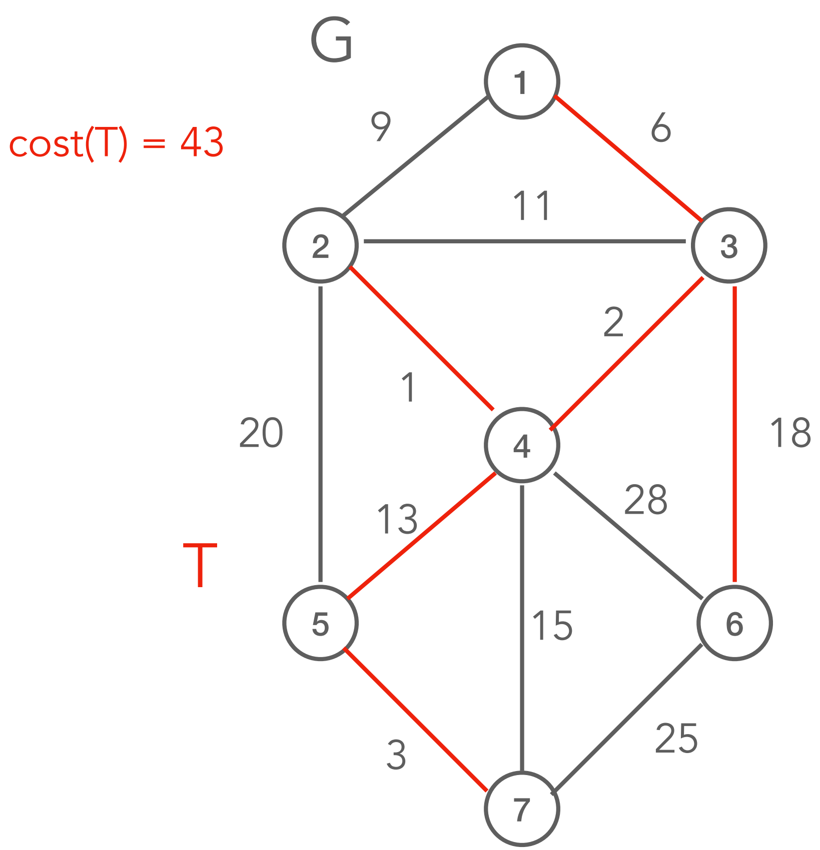
perché?



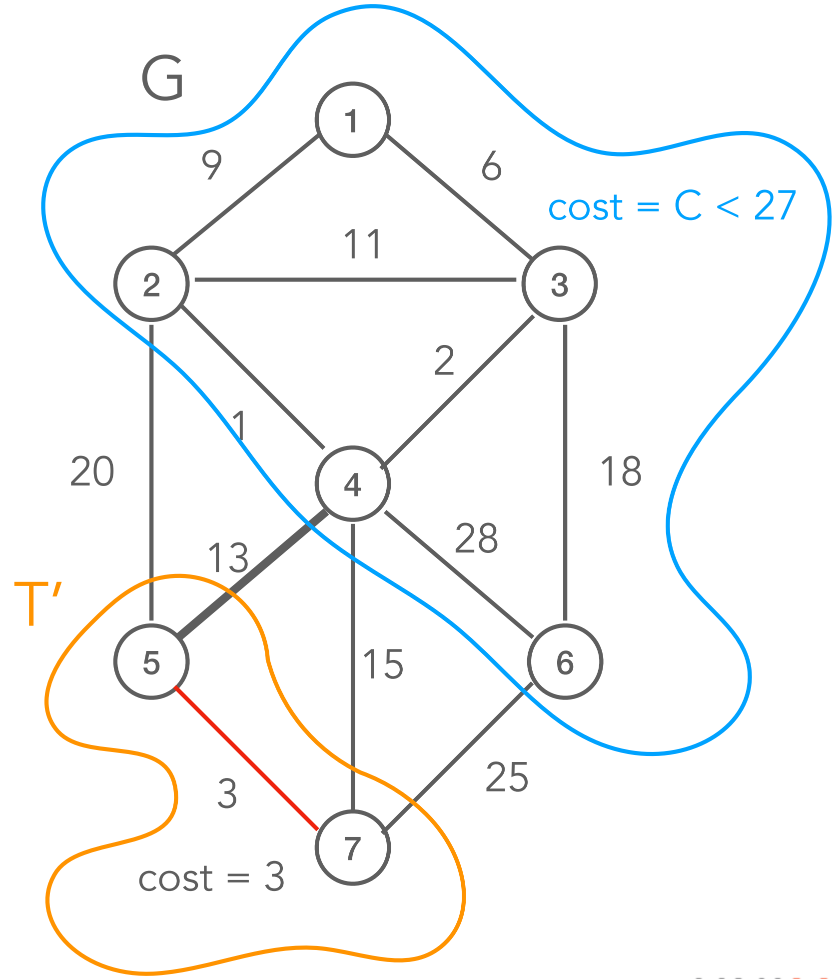
UNIMORE

Minimum Spanning Tree

SOTTOSTRUTTURA OTTIMA



supponiamo sia possibile trovare un MST per i nodi {1,2,3,4,6} di costo C < 27

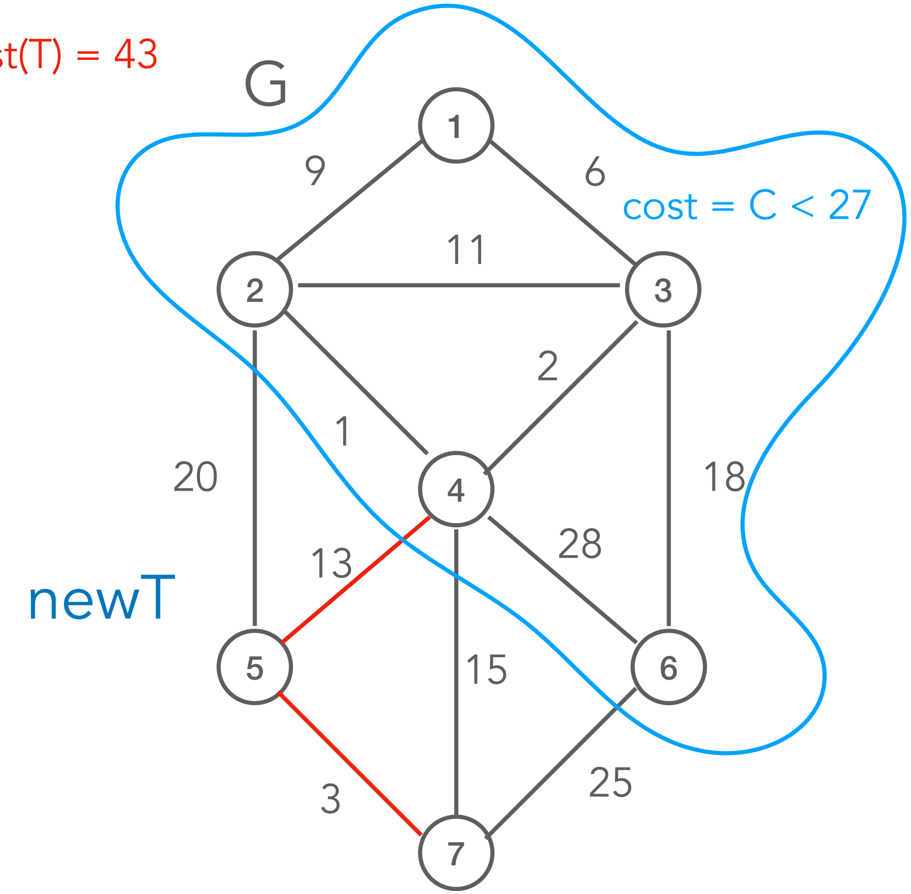


Minimum Spanning Tree

SOTTOSTRUTTURA OTTIMA

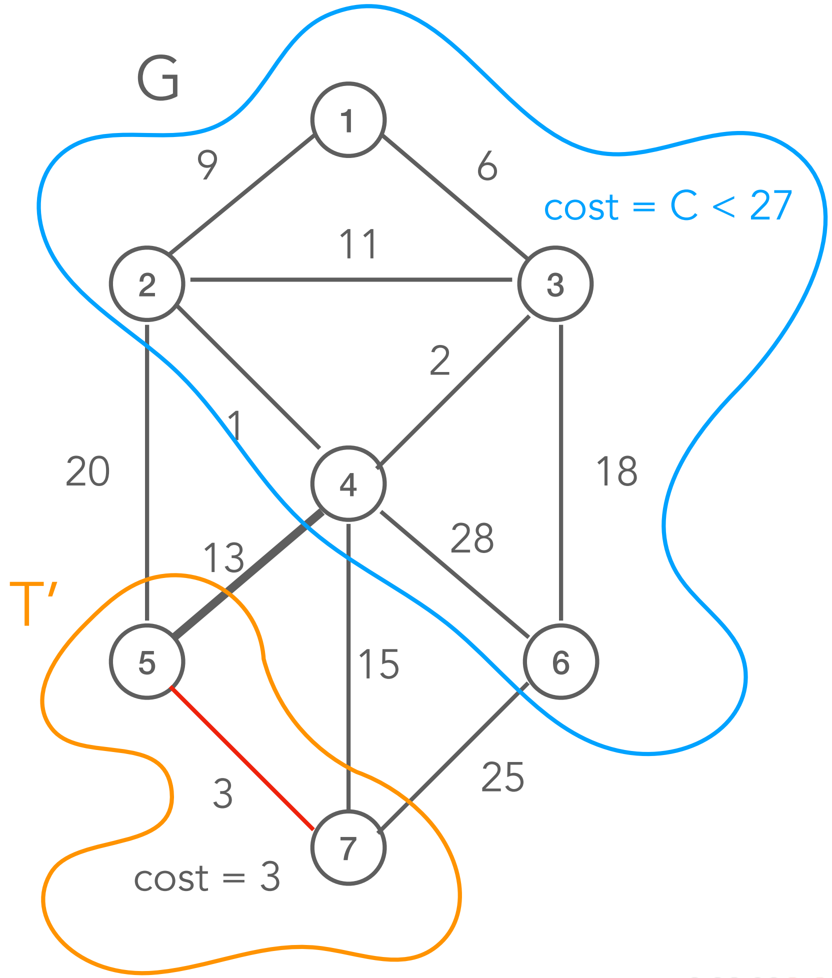
Avremmo trovato un MST di costo MINORE rispetto al MST!

cost(T) = 43



cost(newT) < 13 + 3 + 27 = 43

supponiamo sia possibile trovare un MST per i nodi {1,2,3,4,6} di costo C < 27



Minimum Spanning Tree

Possiamo pensare di usare la strategia GREEDY

dobbiamo scegliere un sottoinsieme di archi del grafo
che formino un MST

Quale scelta GREEDY?

Minimum Spanning Tree

Possiamo pensare di usare la strategia GREEDY

Algoritmo di **KRUSKAL** (1956):

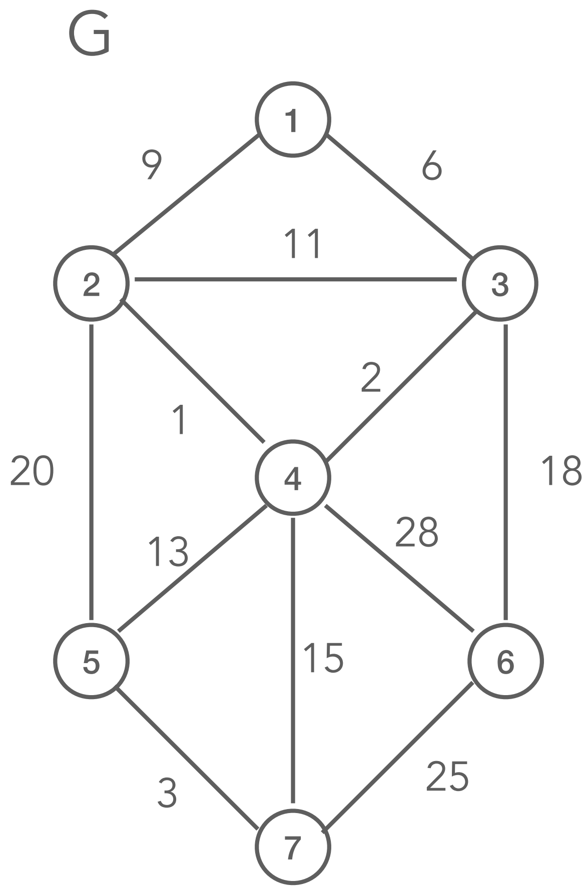
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente: ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi

Algoritmo di **PRIM** (1957):

- Scegliere un nodo sorgente
- Costruire il MST incrementalmente: ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi

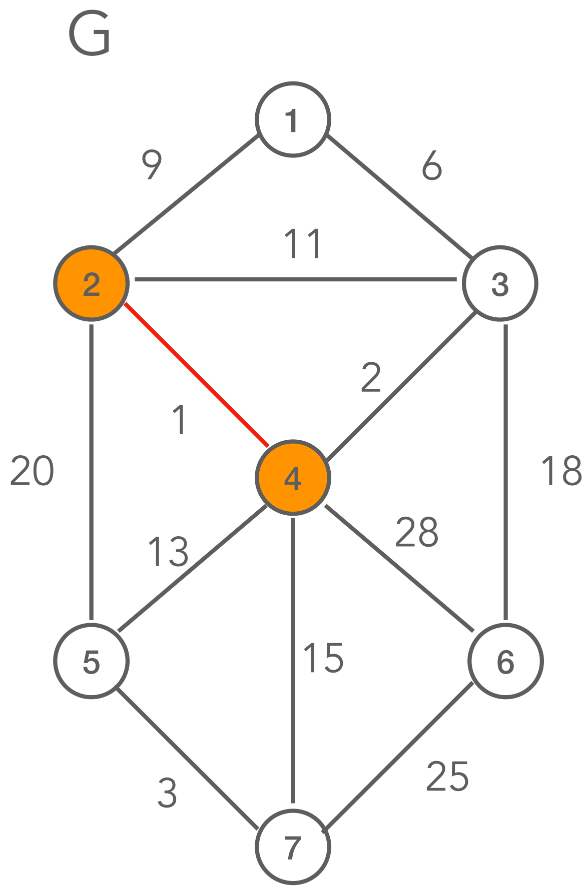
Minimum Spanning Tree

Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente:
  ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi



Minimum Spanning Tree

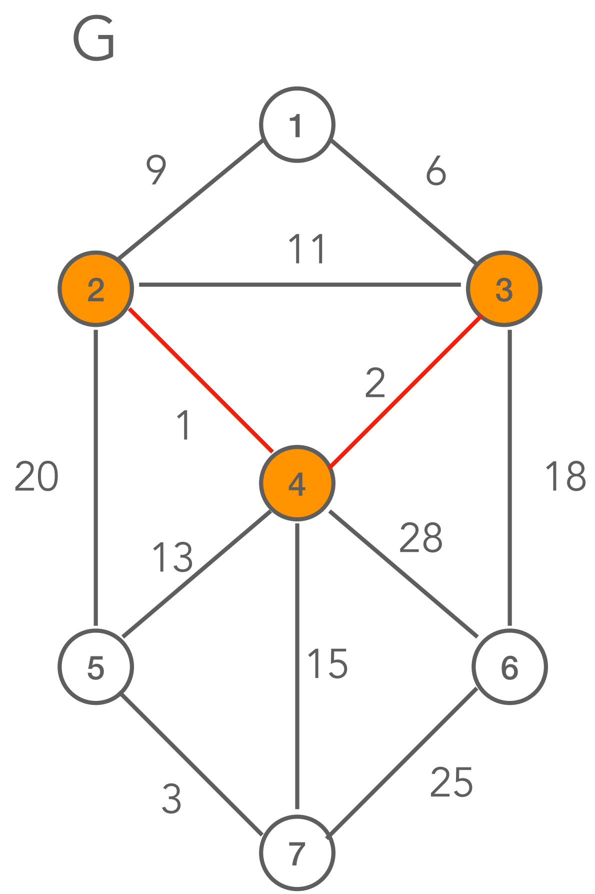
Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente:
  ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi



UNIMORE

Minimum Spanning Tree

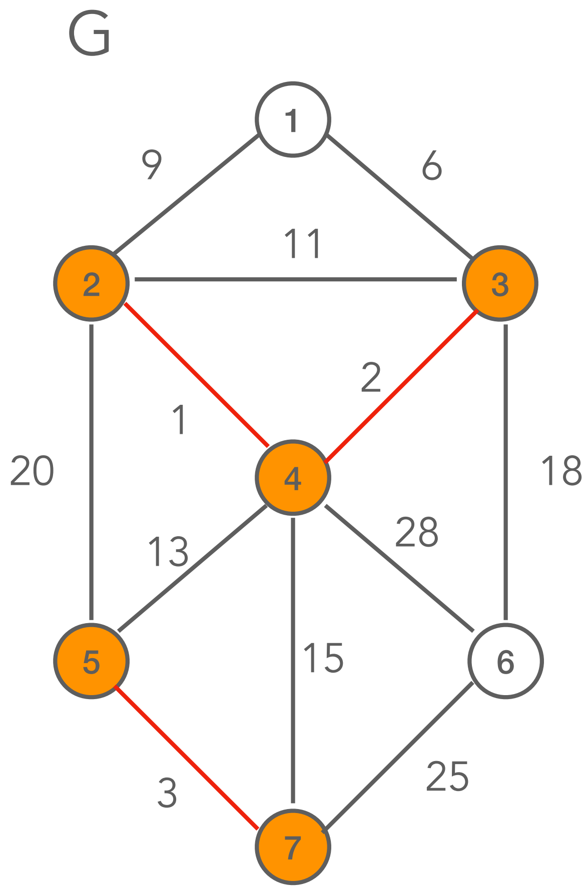
Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente:
  ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi



UNIMORE

Minimum Spanning Tree

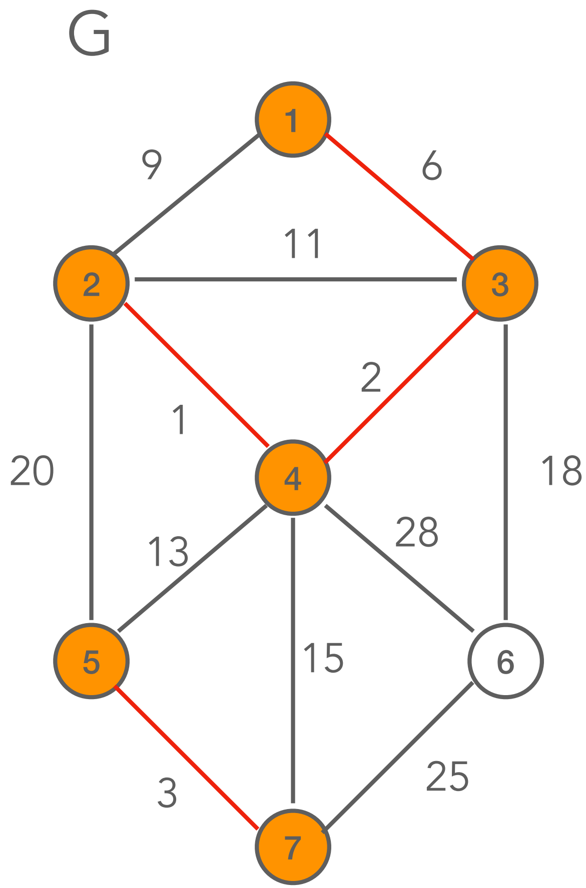
Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente:
  ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi



Minimum Spanning Tree

Algoritmo di **KRUSKAL** (1956):

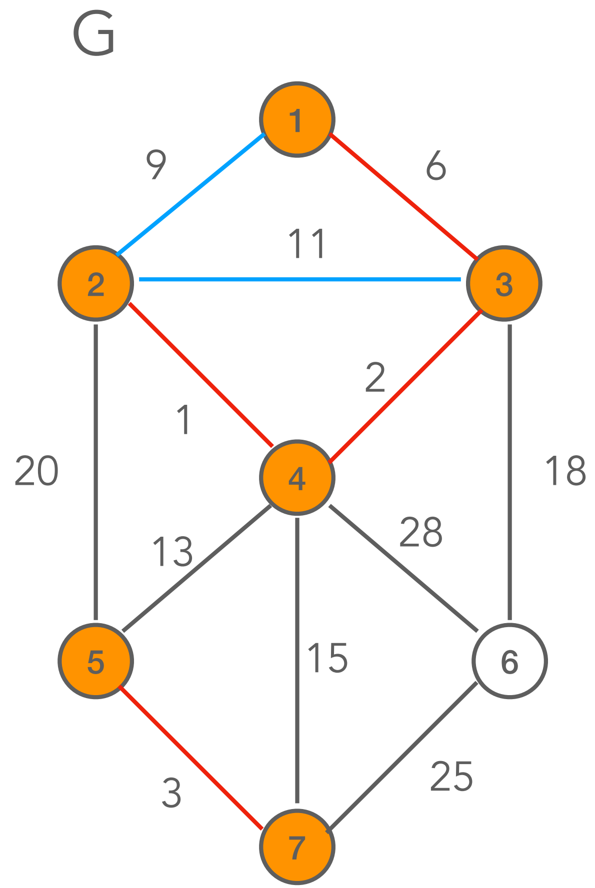
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente:
  ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi



Minimum Spanning Tree

Algoritmo di **KRUSKAL** (1956):

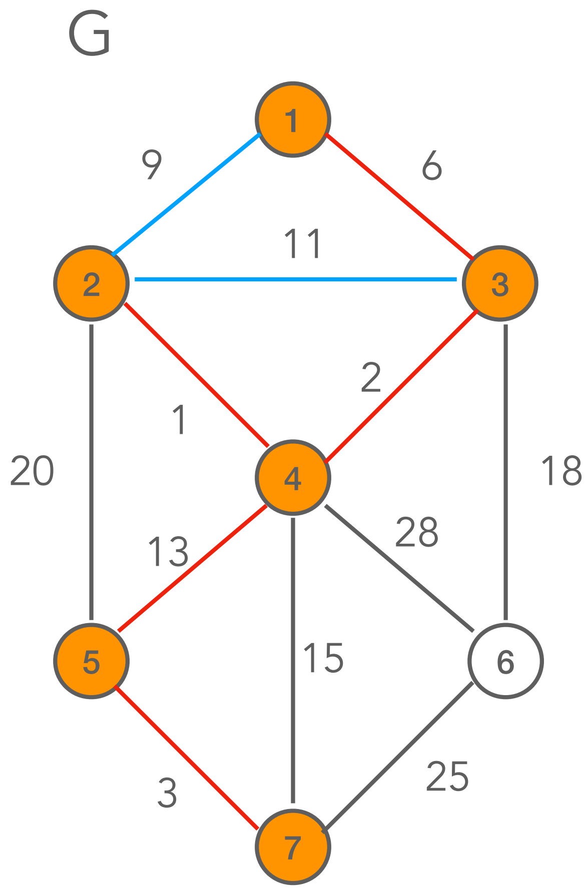
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente:
  ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi



Minimum Spanning Tree

Algoritmo di **KRUSKAL** (1956):

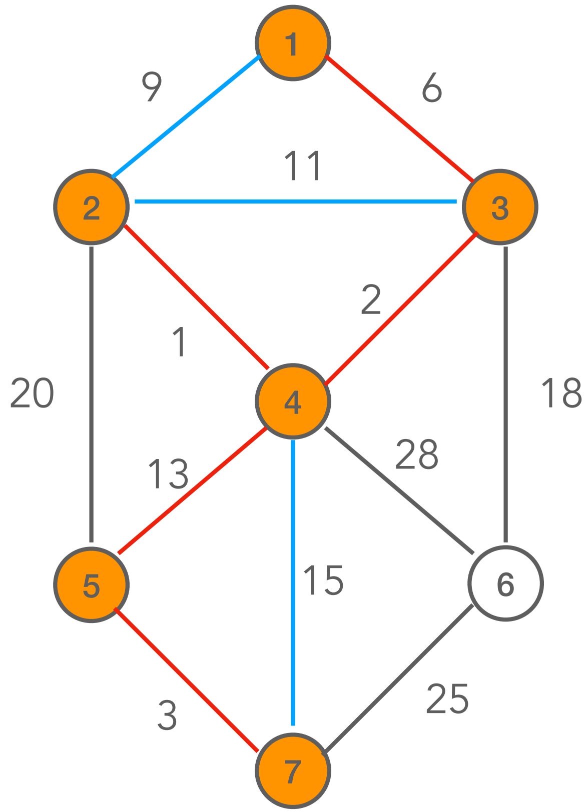
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente:
  ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi



Minimum Spanning Tree

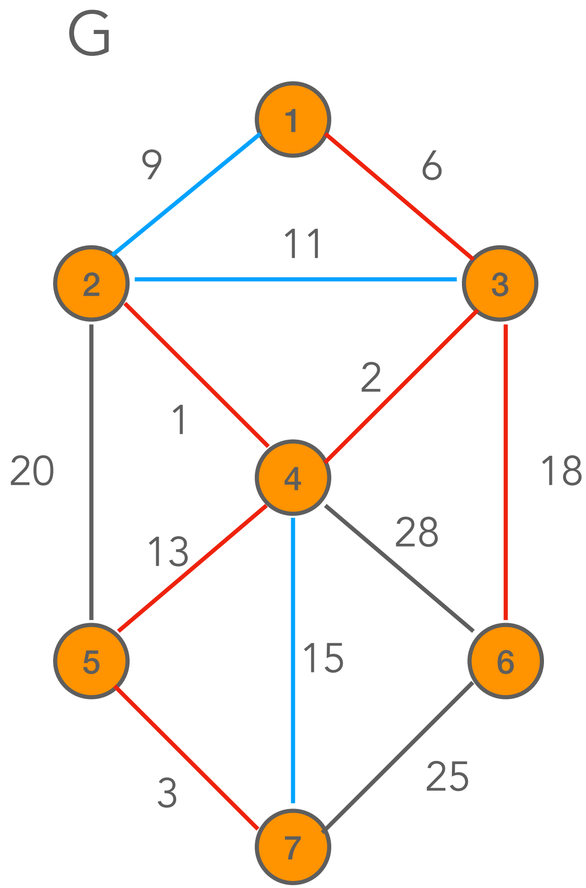
Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente:
  ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi

G



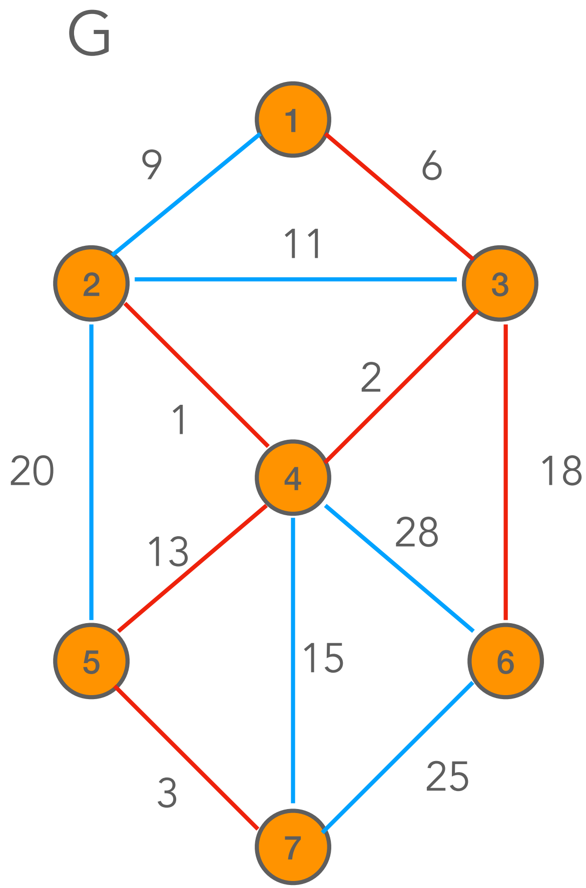
Minimum Spanning Tree

Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente:
  ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi



Minimum Spanning Tree

Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente:
  ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi



Minimum Spanning Tree

Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente:
  ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi

Scelte implementative:
- Usiamo una lista per memorizzare gli archi scelti nel MST
- Usiamo un disjoint set sull'insieme dei nodi per controllare di non formare cicli
- Terminiamo dopo aver selezionato n-1 archi

non l'abbiamo ancora visto, tornare a studiare lo pseudo-codice dopo che avremo fatto la struttura dati disjoint-set

Minimum Spanning Tree

ALGORTIMO DI KRUSKAL

```
Kruscal(G=(V,E),c)
S := make_set(V) //disjoint set
T := new_list()
ordina gli archi di E per ordine non decrescente di costo c
count := 0
while count < n - 1 do
 scegli il prossimo arco (u,v)∈ E nell'ordinamento
 if find-set(S,u) ≠ find-set(S,v)
  then p := new_list_node()
       p.val := (u,v)
       insert-head(T,p)
       union(S,u,v)
       count := count + 1
return T
```

Minimum Spanning Tree

ALGORTIMO DI KRUSKAL

```
Kruscal(G=(V,E),c)
S := make_set(V) //disjoint set O(n)
T := new_list()  O(1)                              O(m log m)
ordina gli archi di E per ordine non decrescente di costo c
count := 0
while count < n - 1 do  <------------------------ # ripetizioni O(m)
 scegli il prossimo arco (u,v)∈ E nell'ordinamento
 if find-set(S,u) ≠ find-set(S,v)
  then p := new_list_node()
       p.val := (u,v)
       insert-head(T,p)
       union(S,u,v)
       count := count + 1
return T
```

$O(\log n) + O(1) + O(\log n) \in O(\log n)$

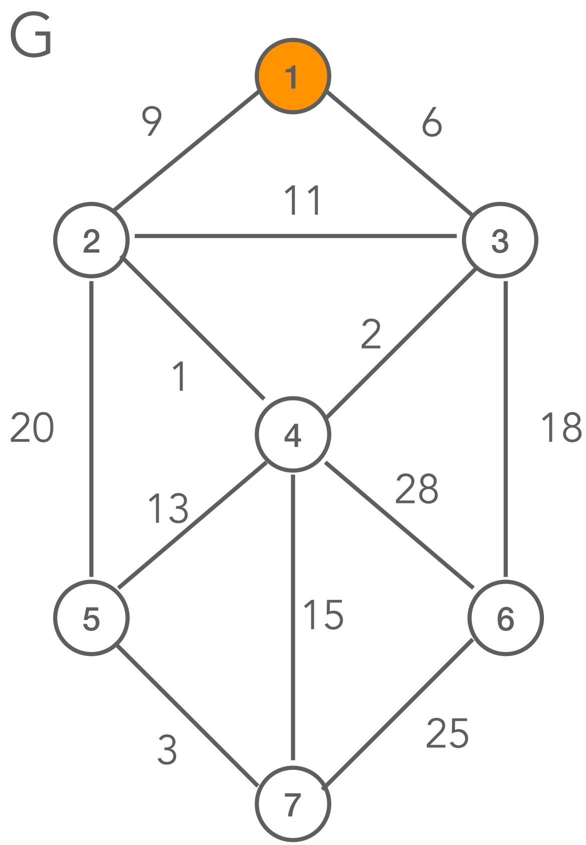
In totale: $O(n) + O(1) + O(m \log m) + O(m \log n) \in O(m \log n)$

$m \leq n^2 \Rightarrow \log m \leq 2 \log n$

Minimum Spanning Tree

Algoritmo di **PRIM** (1957):

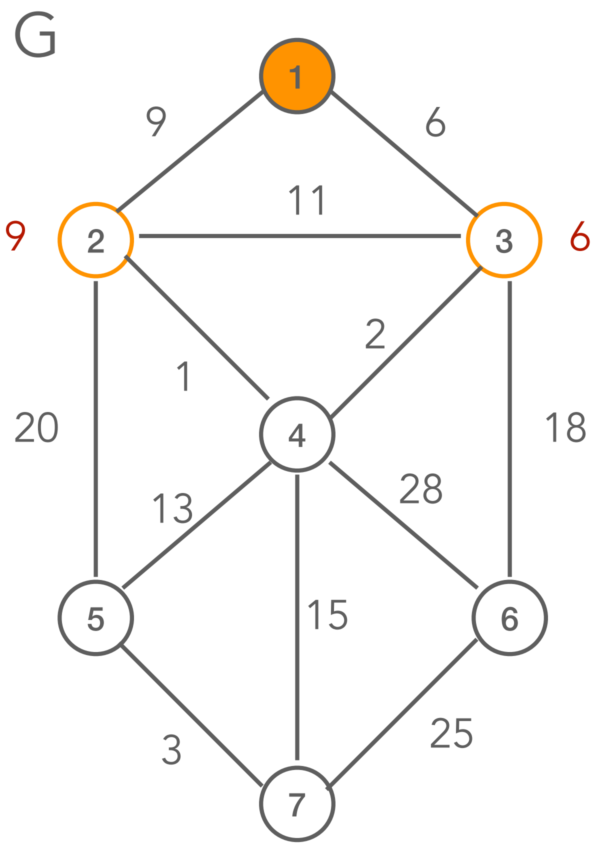
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente:
  ad ogni iterazione scegliere l’arco di peso minimo che unisca un nodo non ancora nell’albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi



Minimum Spanning Tree

Algoritmo di **PRIM** (1957):

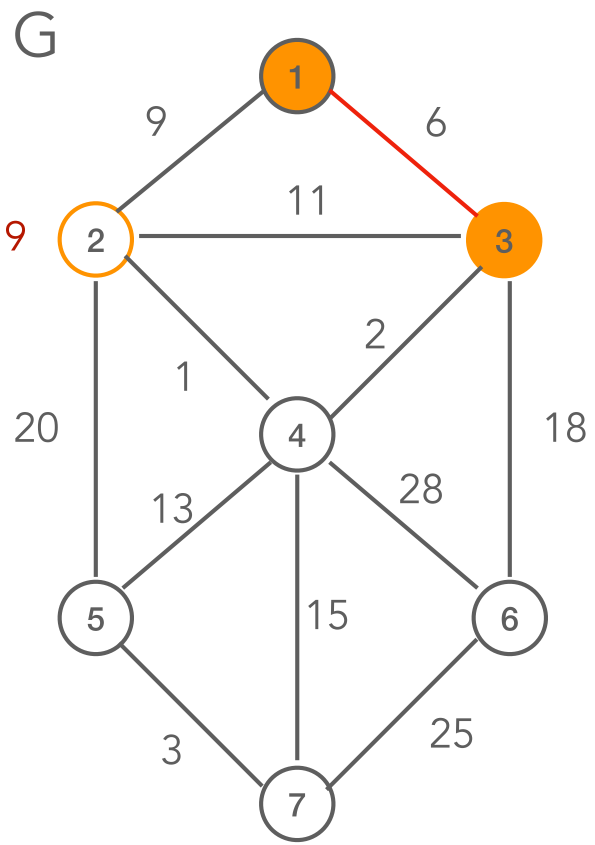
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente:
  ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi



UNIMORE

Minimum Spanning Tree

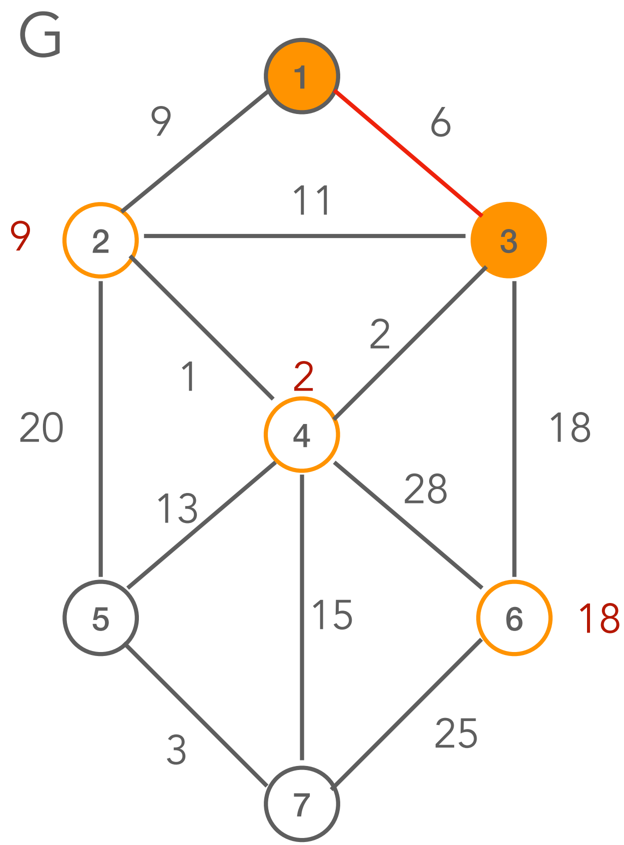
Algoritmo di **PRIM** (1957):
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente:
  ad ogni iterazione scegliere l’arco di peso minimo che unisca un nodo non ancora nell’albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi



Minimum Spanning Tree

Algoritmo di **PRIM** (1957):

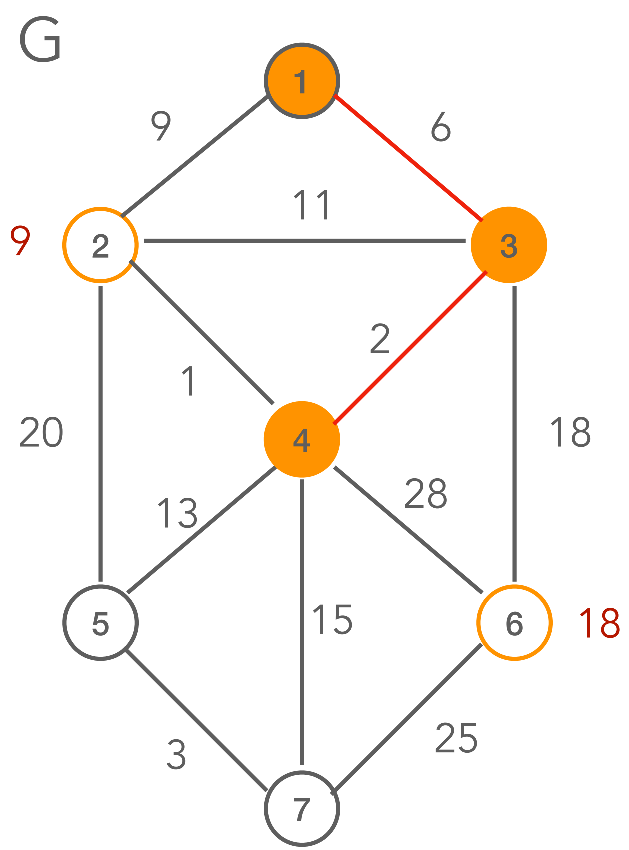
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente:
  ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi



UNIMORE

Minimum Spanning Tree

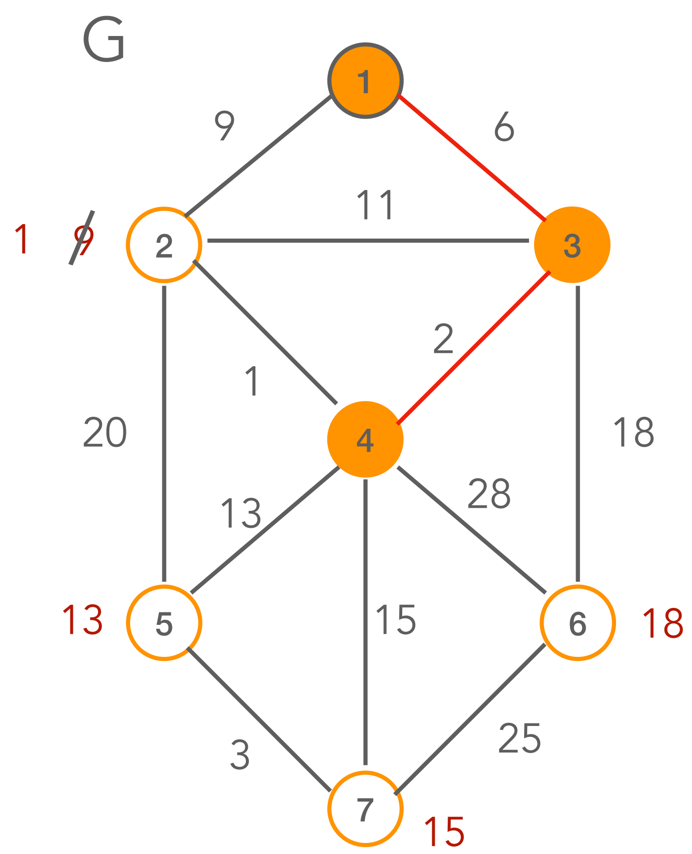
Algoritmo di **PRIM** (1957):
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente:
  ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi



UNIMORE

Minimum Spanning Tree

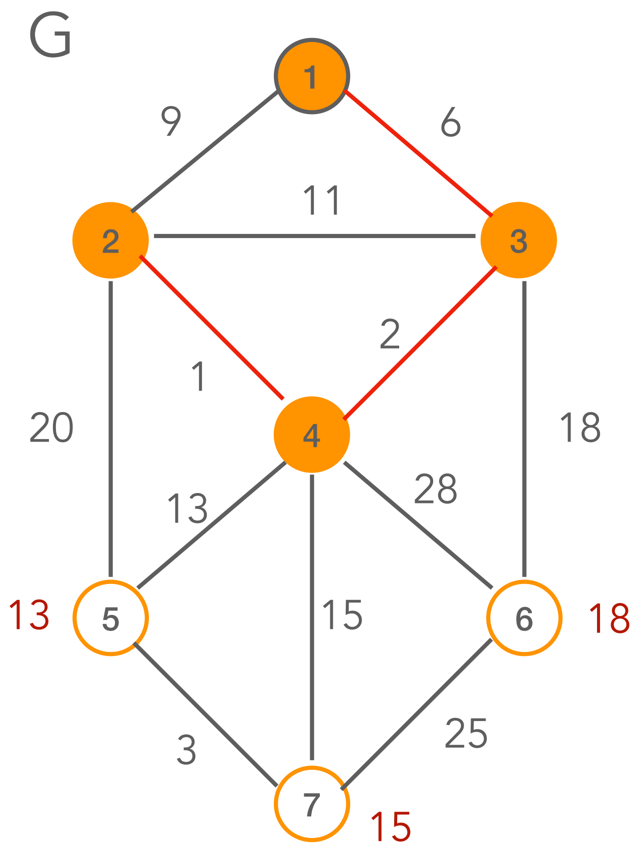
Algoritmo di **PRIM** (1957):
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente:
  ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi



UNIMORE

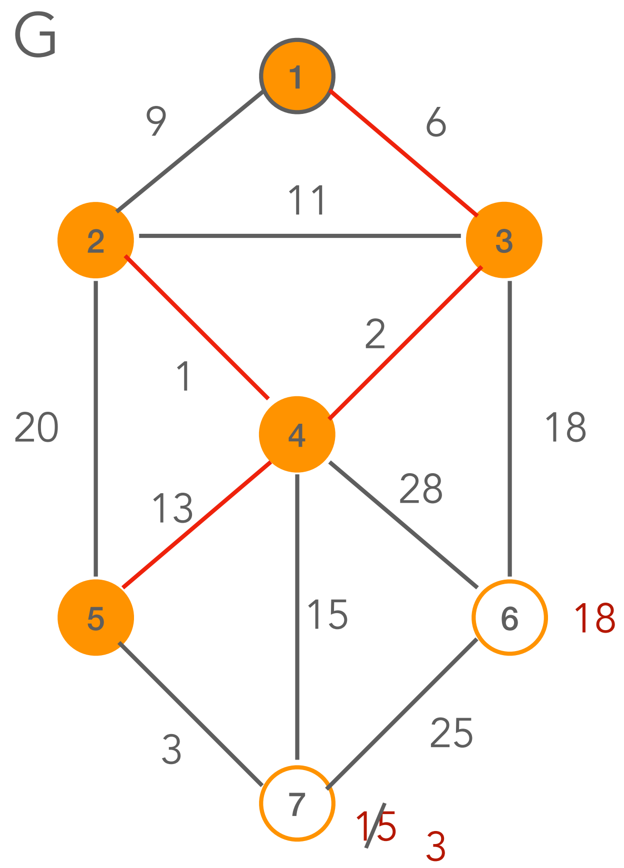
Minimum Spanning Tree

Algoritmo di **PRIM** (1957):
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente:
  ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi



Minimum Spanning Tree

Algoritmo di **PRIM** (1957):
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente:
  ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi

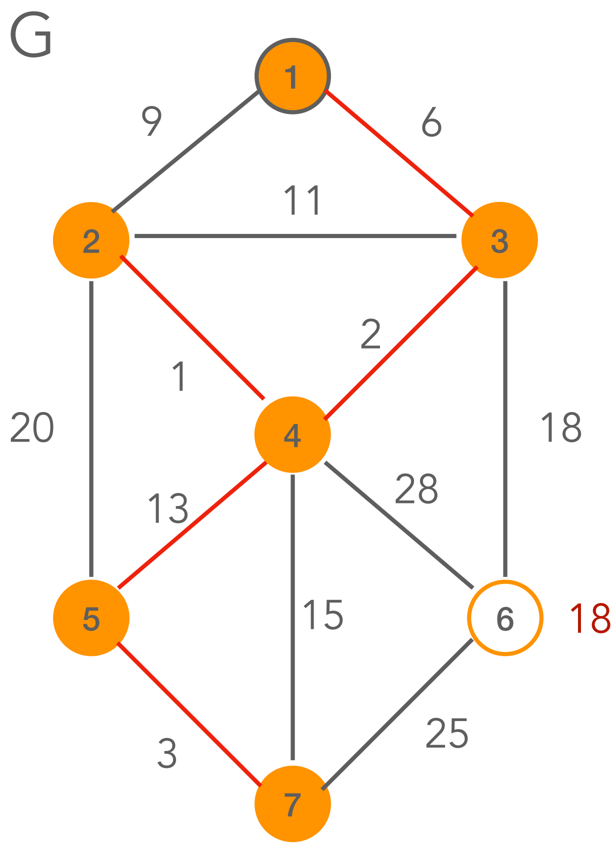


UNIMORE

Minimum Spanning Tree

Algoritmo di **PRIM** (1957):

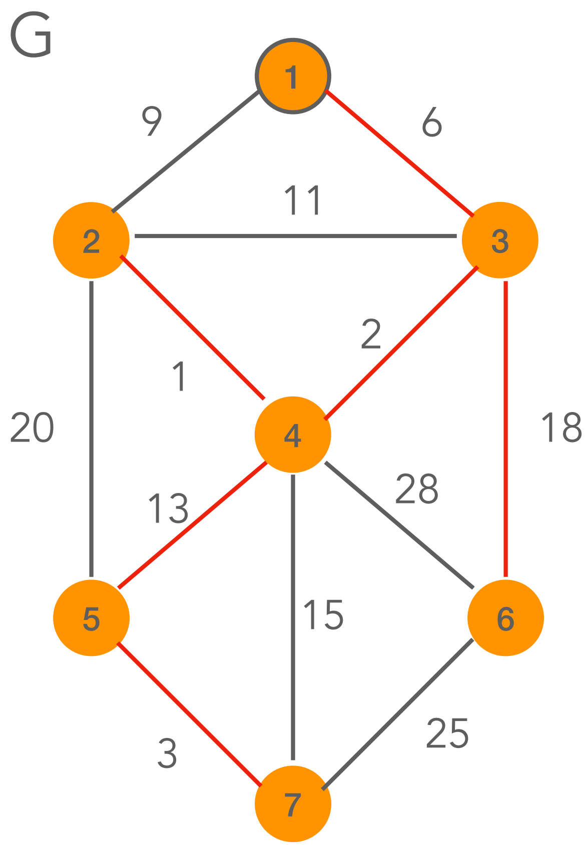
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente:
  ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi



Minimum Spanning Tree

Algoritmo di **PRIM** (1957):

- Scegliere un nodo sorgente
- Costruire il MST incrementalmente:
  ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi



Minimum Spanning Tree

Algoritmo di **PRIM** (1957):
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente:
  ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi

Scelte implementative:
- teniamo traccia del fatto che un nodo sia o meno nel MST
- teniamo traccia del costo per raggiungere un nodo
- usiamo una CODA con PRIORITÀ per scegliere il nodo in ogni iterazione
- teniamo traccia dei nodi padre

Minimum Spanning Tree

ALGORTIMO DI PRIM

```
Prim(G=(V,E),c)
for all v ∈ V do
  cost[v] := +∞
  prev[v] := NIL
  S[v] := 0
scegli un nodo s ∈ V
cost[s] := 0
S[s] := 1
Q := make_priority_queue(V') // coppie nodi di V e cost[]
while NOT is_empty_queue(Q) do
  u := DeQueue(Q)
  S[u] := 1
  for all (u,v)∈ E do
    if S[v] = 0 AND cost[v] > c(u,v)
      then
        cost[v] := c(u,v)
        prev[v] := u
        Decrease_Priority(Q,v,cost[v])
return prev[]
```

Minimum Spanning Tree

ALGORTIMO DI PRIM

```
Prim(G=(V,E),c)
for all v ∈ V do
  cost[v] := +∞
  prev[v] := NIL
  S[v] := 0
scegli un nodo s ∈ V
cost[s] := 0
S[s] := 1
Q := make_priority_queue(V') // coppie nodi di V e cost[]
while NOT is_empty_queue(Q) do
  u := DeQueue(Q)
  S[u] := 1
  for all (u,v)∈ E do
   if S[v] = 0 AND cost[v] > c(u,v)
    then
     cost[v] := c(u,v)
     prev[v] := u
     Decrease_Priority(Q,v,cost[v])
return prev[]
```

$O(n)$

$O(1)$

$O(n)$

in totale $O(n \log n)$

in totale $O(m \log n)$

In totale: $O(n) + O(1) + O(n \log m) + O(m \log n) \in O(m \log n)$

Minimum Spanning Tree

Gli algoritmi di Kruskal e Prim sono corretti?

Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente: ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi

Algoritmo di **PRIM** (1957):
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente: ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi

PRIMA DOMANDA: restituiscono degli alberi di copertura?

Minimum Spanning Tree

Gli algoritmi di Kruskal e Prim sono corretti?

Algoritmo di **KRUSKAL** (1956):

- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente: ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi

SI, per costruzione abbiamo:

- gli archi scelti non formano cicli —> è un albero
- è connesso perché lo era il grafo di partenza

Algoritmo di **PRIM** (1957):

- Scegliere un nodo sorgente
- Costruire il MST incrementalmente: ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi

SI, per costruzione

PRIMA DOMANDA: restituiscono degli alberi di copertura?

Minimum Spanning Tree

Gli algoritmi di Kruskal e Prim sono corretti?

Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente: ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi

Algoritmo di **PRIM** (1957):
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente: ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi

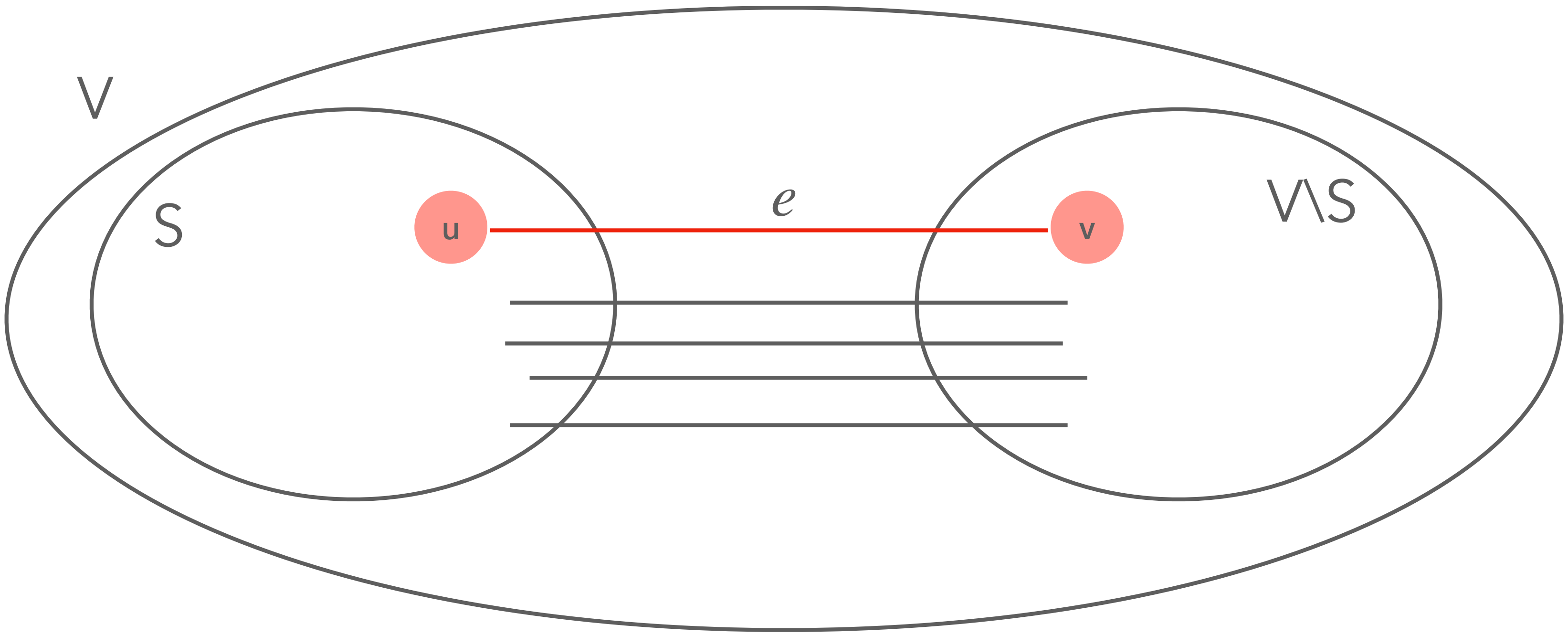
SECONDA DOMANDA: restituiscono dei MST?

UNIMORE

Minimum Spanning Tree

TEOREMA:

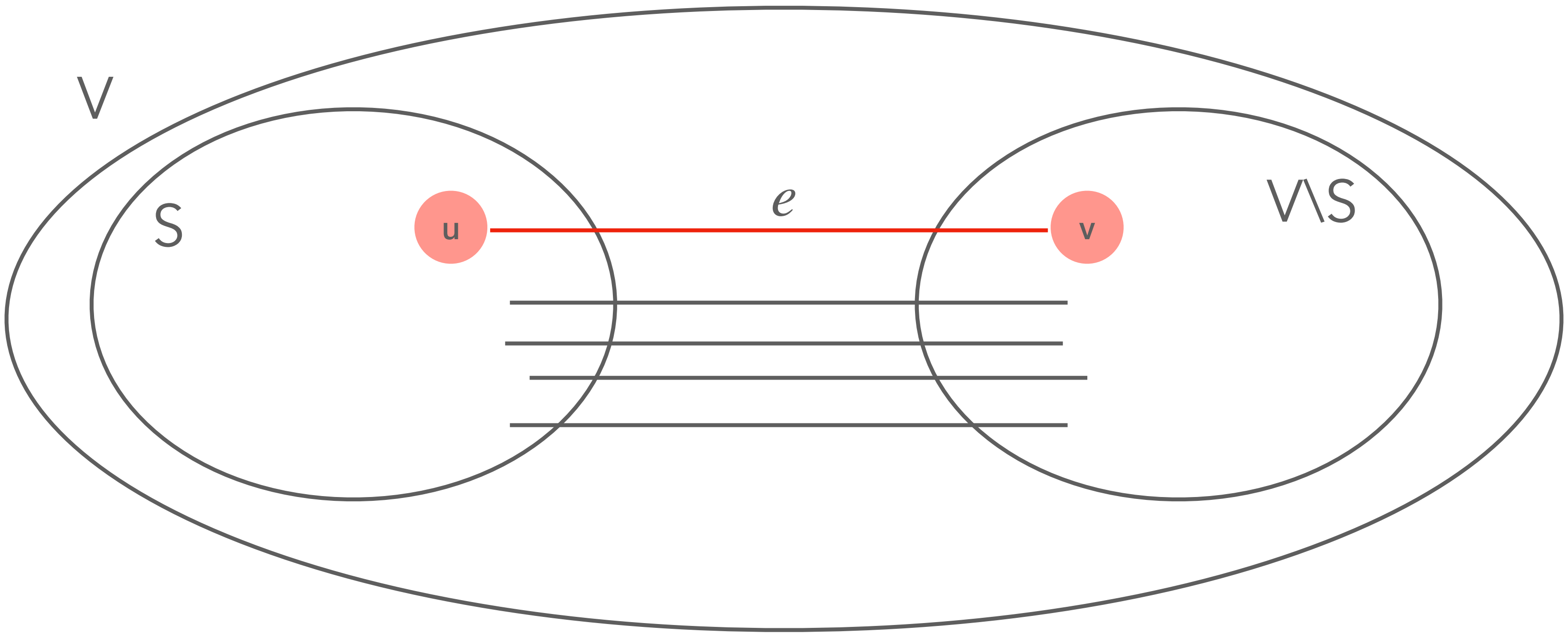
Dati un grafo $G = (V, E)$ e una funzione di costo $c : E \rightarrow R$ sugli archi, sia $S \subset V$ un sottoinsieme proprio e non vuoto di V (cioè $0 < |S| < |V|$) e sia $e = (u, v) \in E$ un arco id costo minimo tale che $u \in S$ e $v \in V \backslash S$, allora l'arco e appartiene ad un MST per G



Minimum Spanning Tree

DEFINIZONI:

* La partizione $(S, V \backslash S)$ è detta **TAGLIO** del grafo $G = (V, E)$
* L'arco $e = (u, v) \in E$ un arco di costo minimo che attraversa un taglio del grafo G è detto **SAFE**

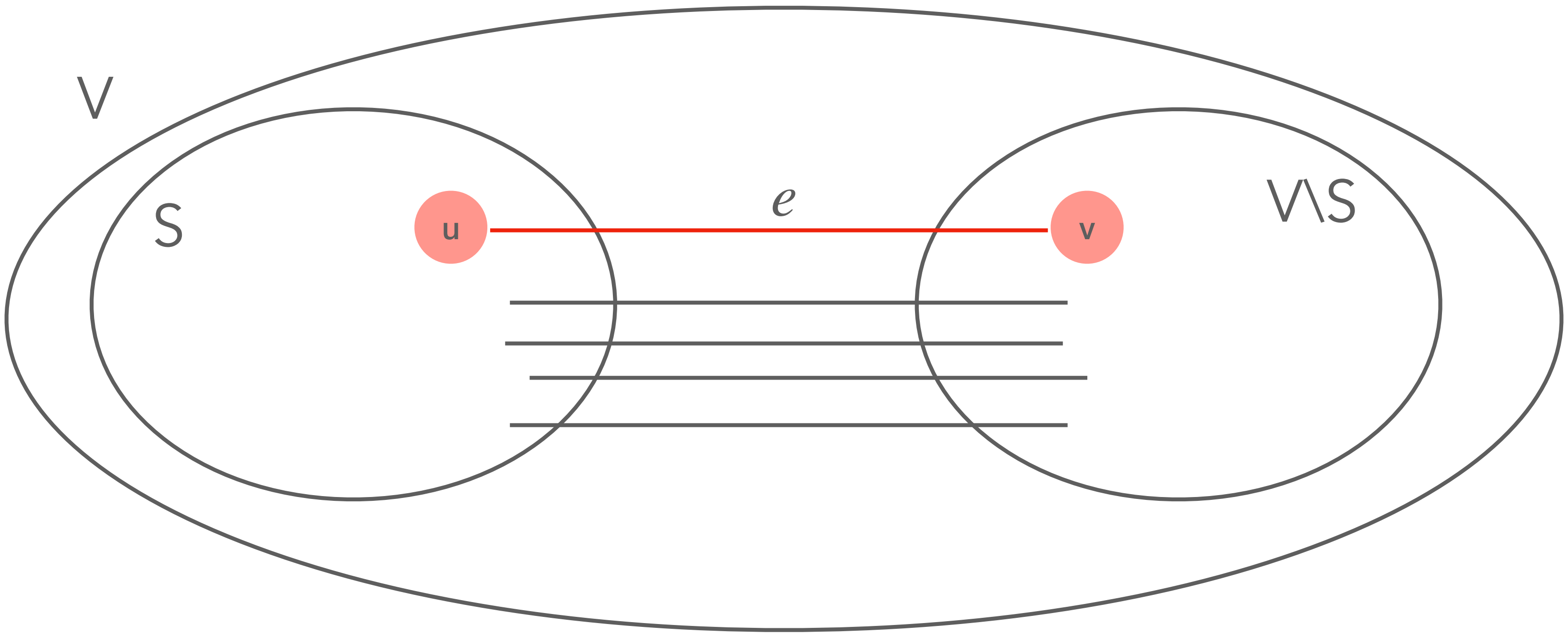


Minimum Spanning Tree

DIMOSTRAZIONE:

ci sono due casi:

- l'arco *e* appartiene ad un MST per G ← OK
- l'arco *e* NON appartiene ad un MST per G



Minimum Spanning Tree

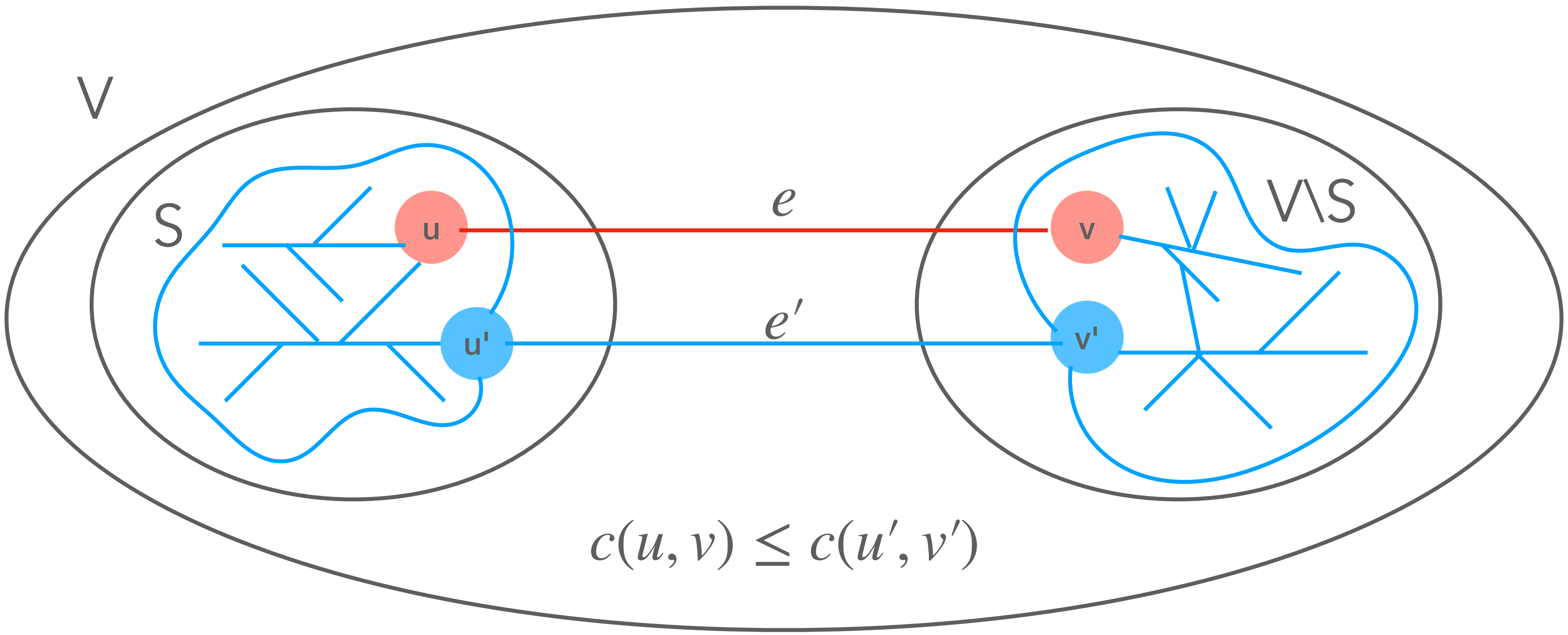
DIMOSTRAZIONE:

- l'arco e NON appartiene ad un MST per G

Nessun MST di G usa $e = (u, v)$ per connettere i nodi del taglio,

$\Downarrow$

Tutti i MST usano un solo arco $e' = (u', v')$ che attraversa il taglio diverso da e



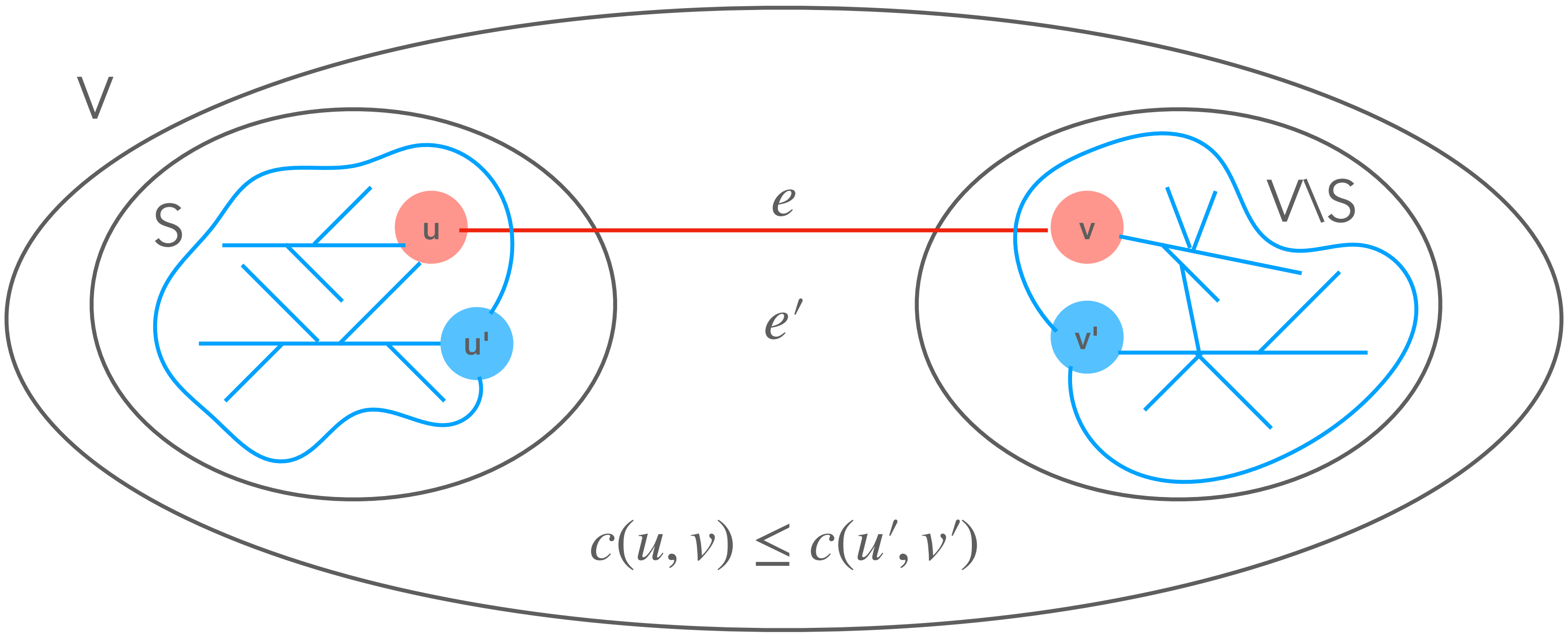
Minimum Spanning Tree

DIMOSTRAZIONE:

- l'arco e NON appartiene ad un MST per G

Sostituendo e' con e nel MST

troviamo un albero di copertura di costo non maggiore di MST



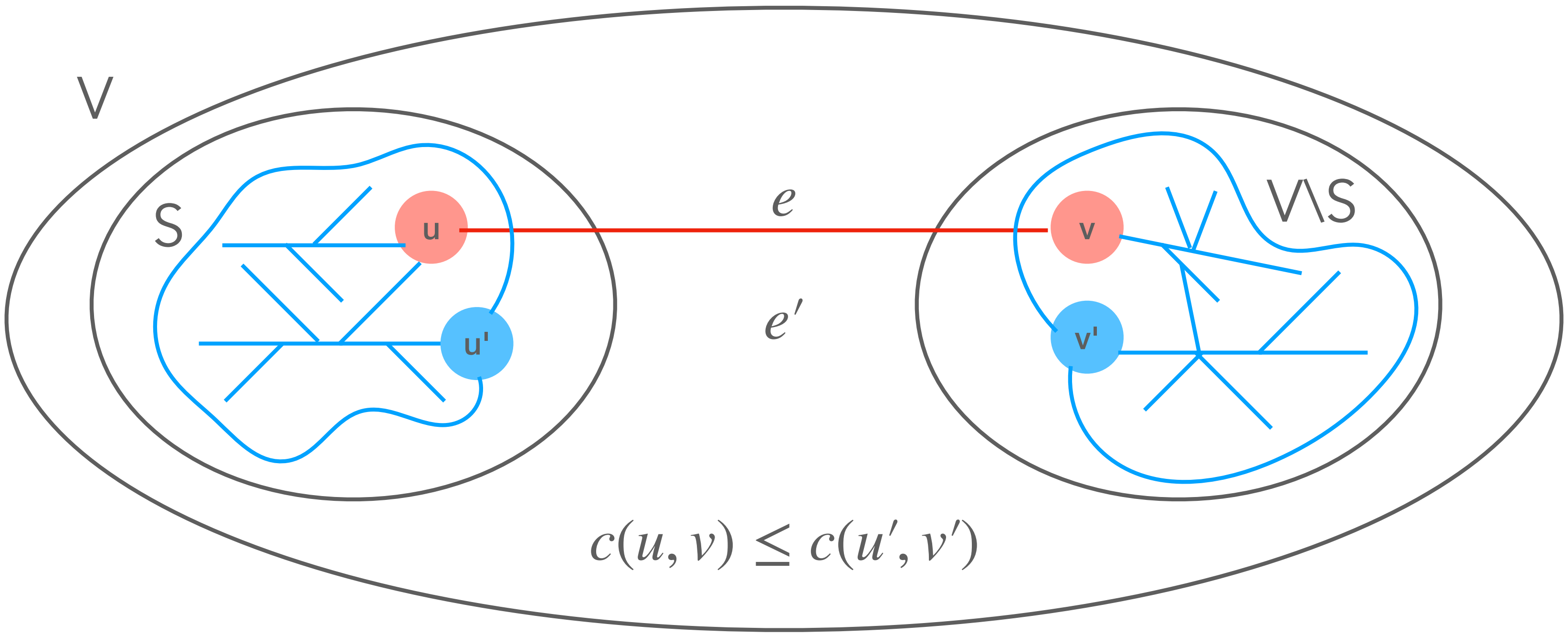
Minimum Spanning Tree

DIMOSTRAZIONE:

- l'arco e NON appartiene ad un MST per G ← NON è possibile!

Sostituendo e' con e nel MST

troviamo un albero di copertura di costo non maggiore di MST



Minimum Spanning Tree

Gli algoritmi di Kruskal e Prim sono corretti?

Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente: ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi

Algoritmo di **PRIM** (1957):
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente: ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi

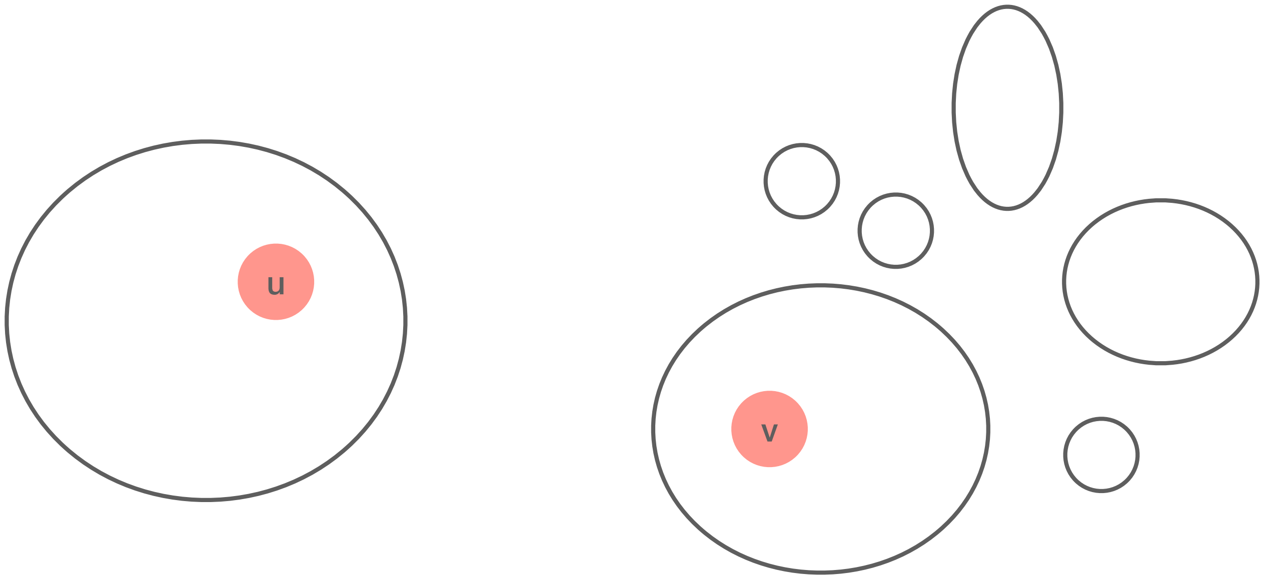
SECONDA DOMANDA: restituiscono dei MST?

SI, se ad ogni iterazione viene scelto un arco SAFE

Minimum Spanning Tree

Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente: ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi

Sceglie un arco SAFE ad ogni iterazione?



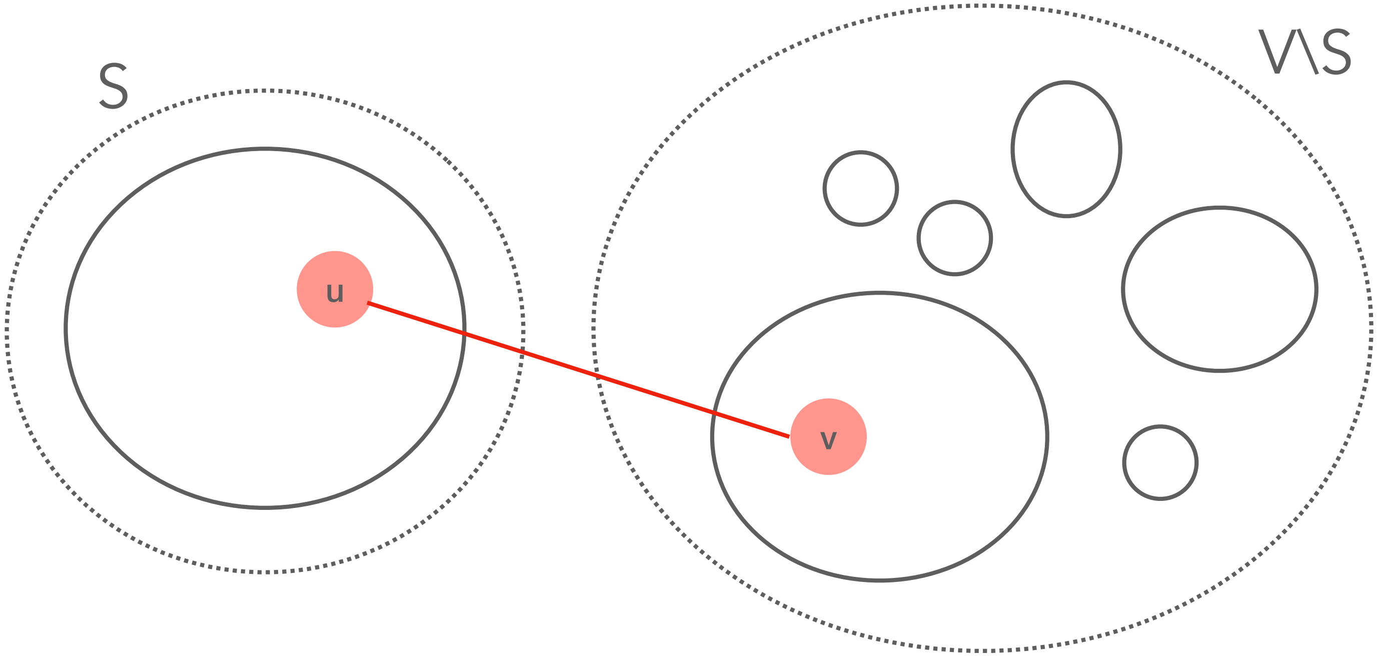
Iterazione i-esima:
si sceglie l'arco (u,v)

Minimum Spanning Tree

Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente: ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi

Sceglie un arco SAFE ad ogni iterazione?

SI



Iterazione i-esima:
si sceglie l'arco (u,v)

UNIMORE

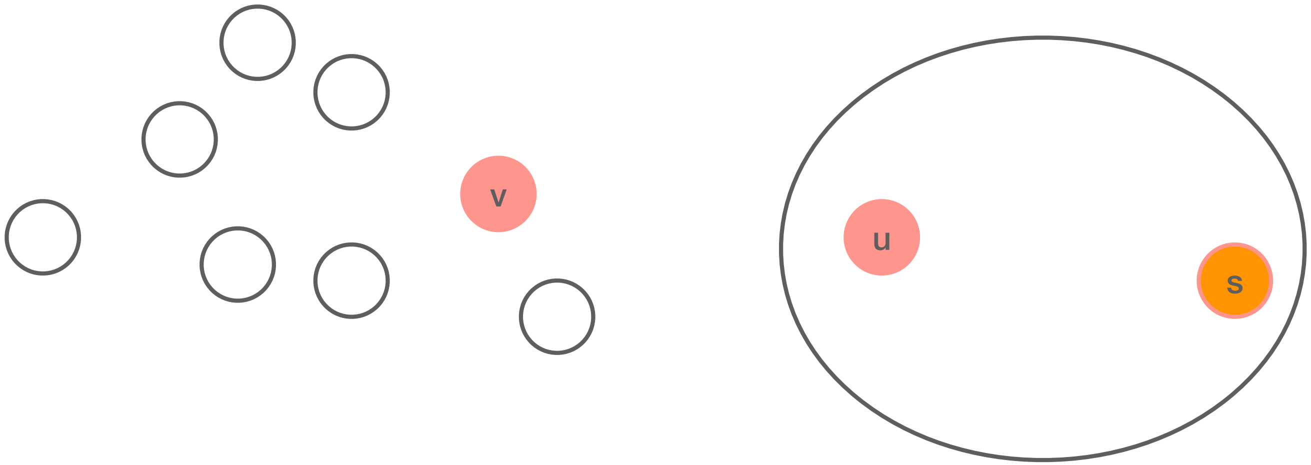
Minimum Spanning Tree

Sceglie un arco SAFE ad ogni iterazione?

Algoritmo di **PRIM** (1957):

- Scegliere un nodo sorgente
- Costruire il MST incrementalmente: ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi

Iterazione i-esima:
si sceglie l'arco (u,v)



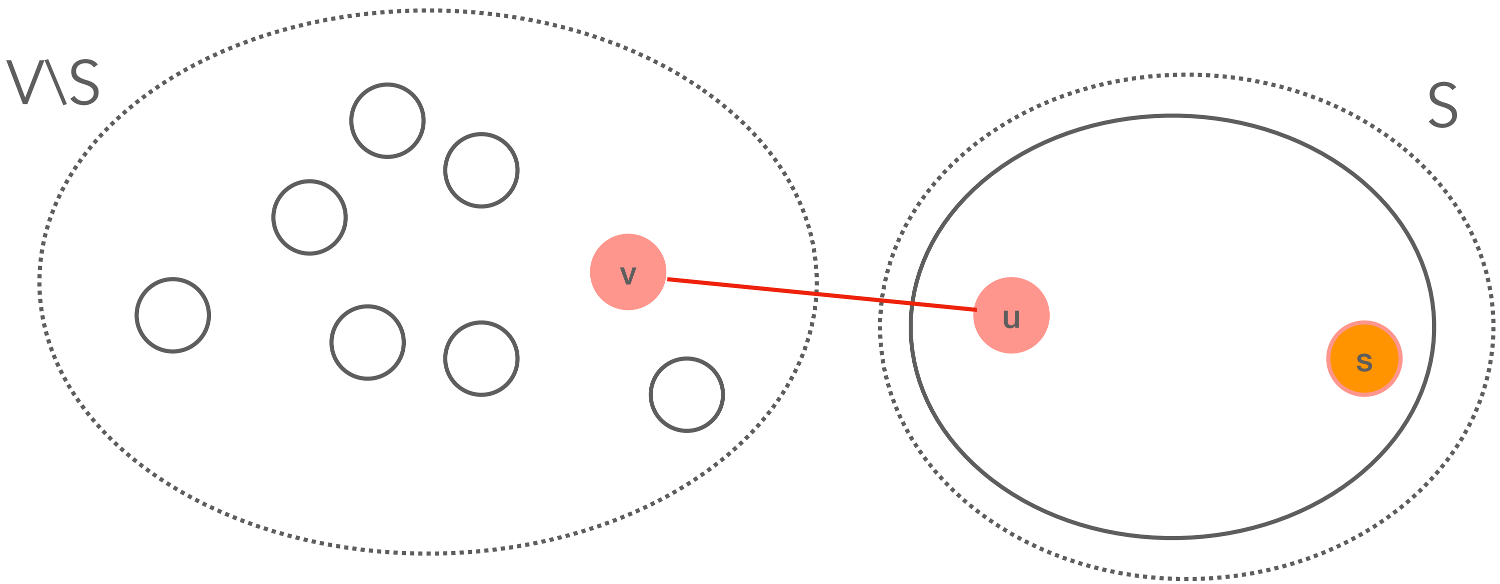
Minimum Spanning Tree

Sceglie un arco SAFE ad ogni iterazione?

Algoritmo di **PRIM** (1957):

- Scegliere un nodo sorgente
- Costruire il MST incrementalmente: ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi

Iterazione i-esima:
si sceglie l'arco (u,v)



Minimum Spanning Tree

Algoritmo di **KRUSKAL** (1956):
- Ordinare gli archi per ordine crescente di peso
- Scegliere gli archi del MST iterativamente: ad ogni iterazione scegliere l'arco di peso minimo che non crei cicli
- Termina quando tutti i nodi sono connessi

Algoritmo di **PRIM** (1957):
- Scegliere un nodo sorgente
- Costruire il MST incrementalmente: ad ogni iterazione scegliere l'arco di peso minimo che unisca un nodo non ancora nell'albero ad un nodo già nell'albero
- Termina quando tutti i nodi sono connessi

Gli algoritmi di Kruskal e Prim sono corretti