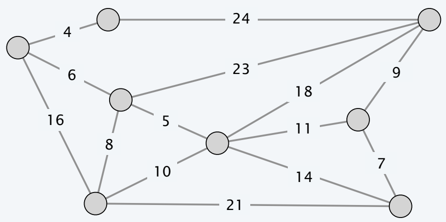
Minimo sottografo ricoprente

Dato un grafo connesso $G = (V, E)$ con costi positivi sugli archi c_e, un minimo sottografo ricoprente è un insieme di archi $E' \subseteq E$ tale che:

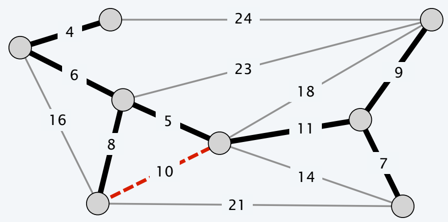
- $G' = (V, E')$ è connesso
- $c(E') = \sum_e c_e \le c(F)$ per ogni altro insieme di archi $F \subseteq E$ tale che $H = (V, F)$ è connesso.



Minimo sottografo ricoprente

Dato un grafo connesso $G = (V, E)$ con costi positivi sugli archi c_e, un minimo sottografo ricoprente è un insieme di archi $E' \subseteq E$ tale che:

- $G' = (V, E')$ è connesso
- $c(E') = \sum_e c_e \le c(F)$ per ogni altro insieme di archi $F \subseteq E$ tale che $H = (V, F)$ è connesso.

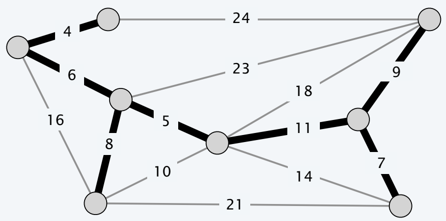


Osservazione: $G' = (V, E')$ non contiene cicli, quindi è un albero

Minimo albero ricoprente

Dato un grafo connesso $G = (V, E)$ con costi positivi sugli archi c_e, un minimo albero ricoprente è un insieme di archi $E' \subseteq E$ tale che:

- $G' = (V, E')$ è un albero
- $c(E') = \sum_e c_e \le c(F)$ per ogni altro insieme di archi $F \subseteq E$ tale che $H = (V, F)$ è un albero.



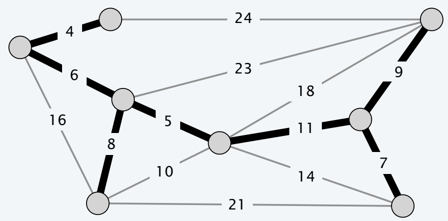
MST di costo = 50 = 4 + 6 + 8 + 5 + 11 + 9 + 7

Osservazione: $G' = (V, E')$ non contiene cicli, quindi è un albero

Minimo albero ricoprente

Dato un grafo connesso $G = (V, E)$ con costi positivi sugli archi c_e, un minimo albero ricoprente è un insieme di archi $E' \subseteq E$ tale che:

- $G' = (V, E')$ è un albero
- $c(E') = \sum_e c_e \le c(F)$ per ogni altro insieme di archi $F \subseteq E$ tale che $H = (V, F)$ è un albero.



MST di costo = 50 = 4 + 6 + 8 + 5 + 11 + 9 + 7

non possiamo cercarlo esaustivamente

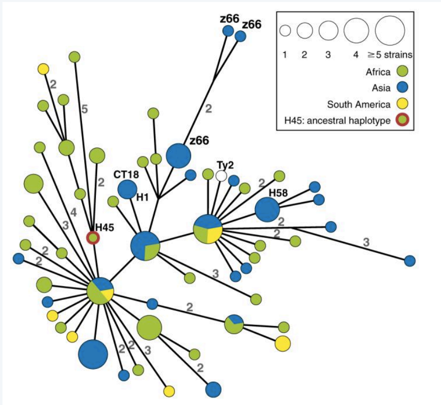
Teorema[Cayley]. Un grafo può avere fino a n^{n-2} alberi ricoprenti

Applicazioni

MST è un problema che si ritrova in numerose applicazioni.

- design di reti (telefoniche, idrauliche, stradali)
- clustering.
- compressione dati in proteomica.
- generazione di ipotesi evolutive.

MST su 59 aplotipi di Salmonella Typhi
la lunghezza degli archi rappresenta la
differenza in numero di SNP



Algoritmi greedi per MST

Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.
- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

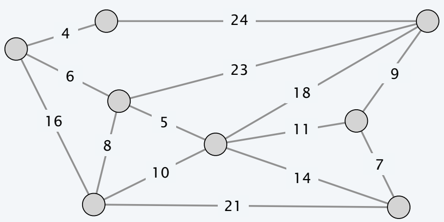
Algoritmo di Kruskal
- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo

Algoritmi greedi per MST

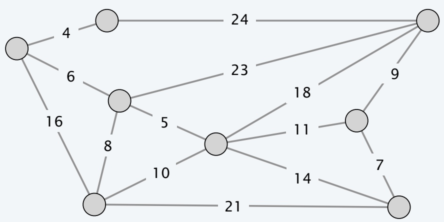
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.
- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal
- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim



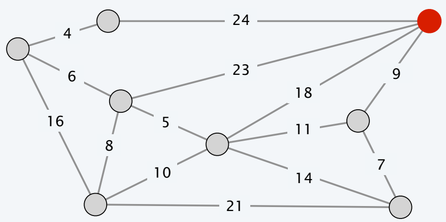
Kruskal

Algoritmi greedi per MST

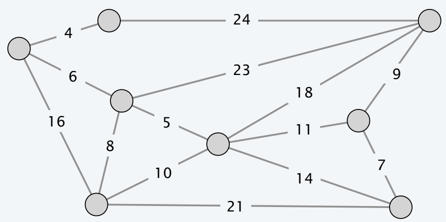
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.
- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal
- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim



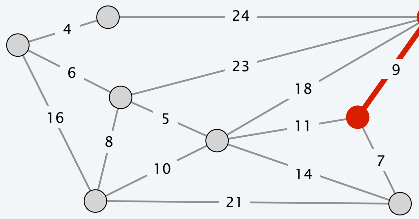
Kruskal

Algoritmi greedi per MST

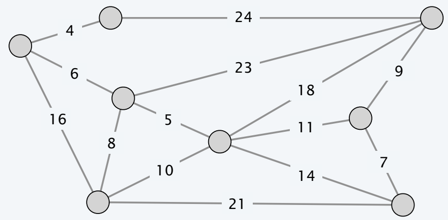
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.
- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal
- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim



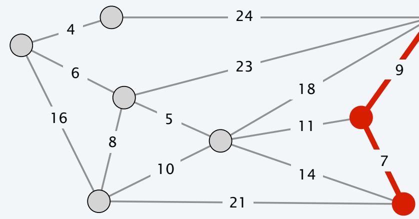
Kruskal

Algoritmi greedi per MST

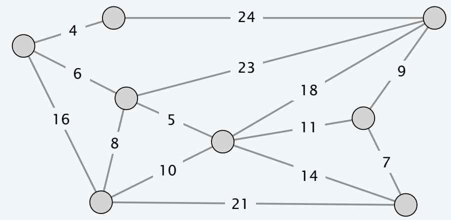
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.
- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal
- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim



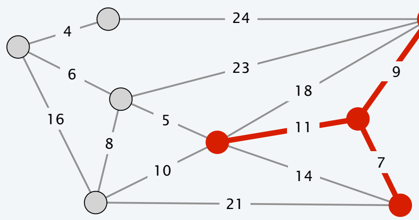
Kruskal

Algoritmi greedi per MST

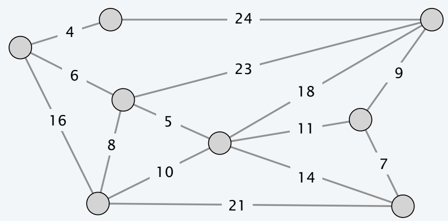
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.
- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal
- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim



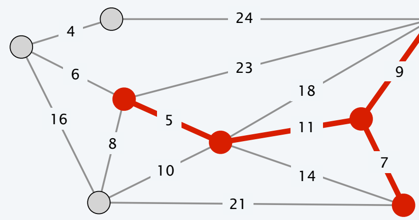
Kruskal

Algoritmi greedi per MST

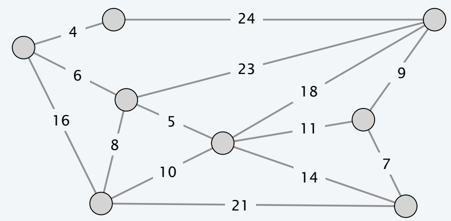
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.
- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal
- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim



Kruskal

Algoritmi greedi per MST

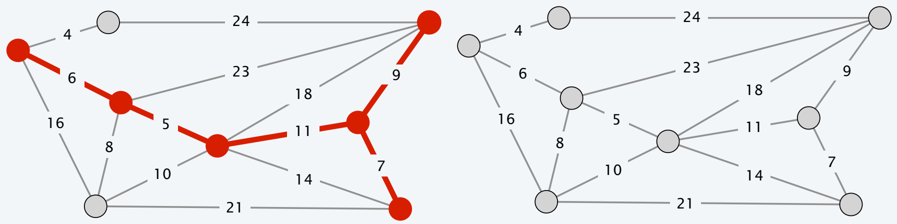
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.

- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal

- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim Kruskal

Algoritmi greedi per MST

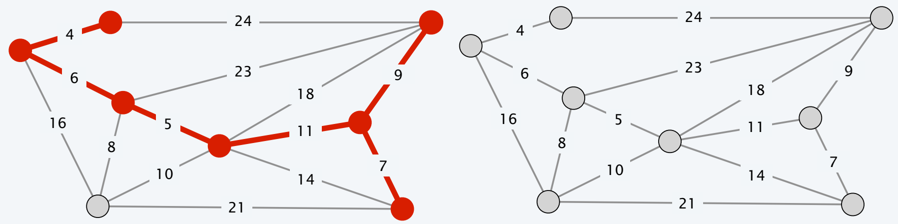
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.

- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal

- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim Kruskal

Algoritmi greedi per MST

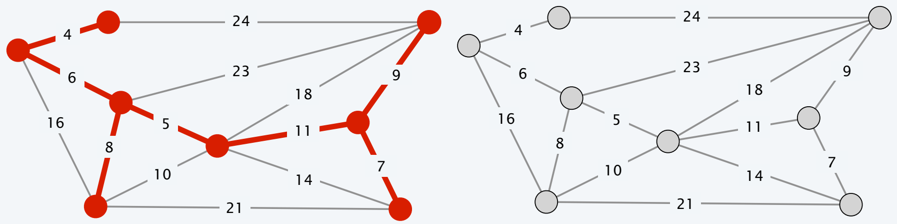
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.

- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal

- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim Kruskal

Algoritmi greedi per MST

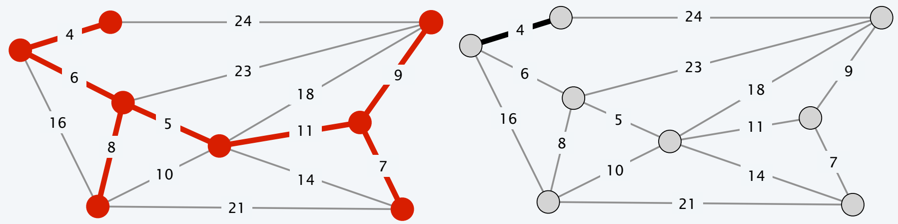
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.

- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal

- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



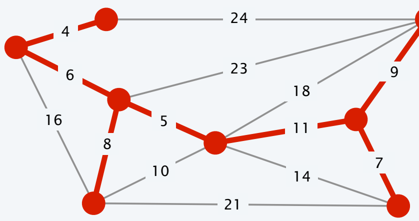
Prim Kruskal

Algoritmi greedi per MST

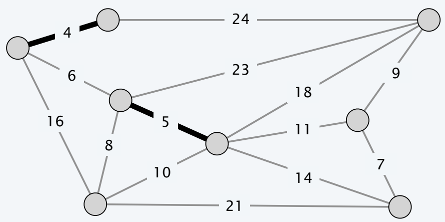
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.
- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal
- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim



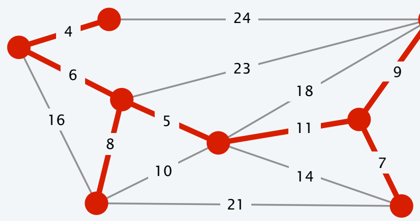
Kruskal

Algoritmi greedi per MST

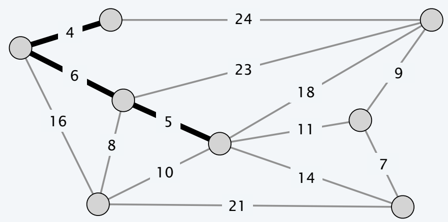
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.
- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal
- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim



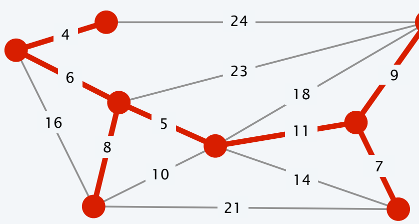
Kruskal

Algoritmi greedi per MST

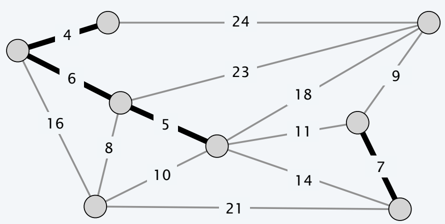
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.
- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal
- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim



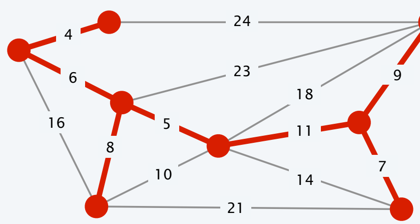
Kruskal

Algoritmi greedi per MST

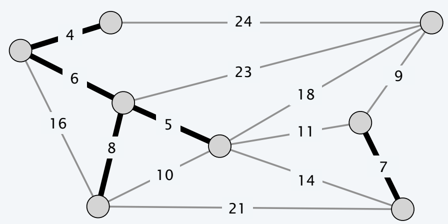
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.
- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal
- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim



Kruskal

Algoritmi greedi per MST

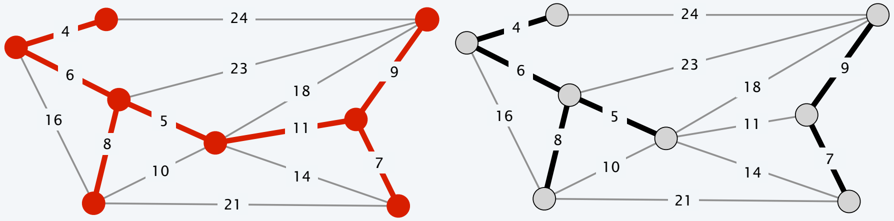
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.

- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal

- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim Kruskal

Algoritmi greedi per MST

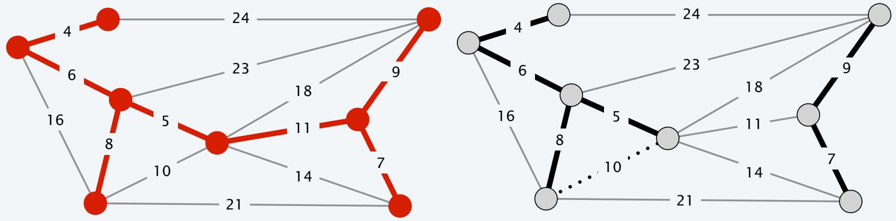
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.

- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal

- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo



Prim Kruskal

Algoritmi greedi per MST

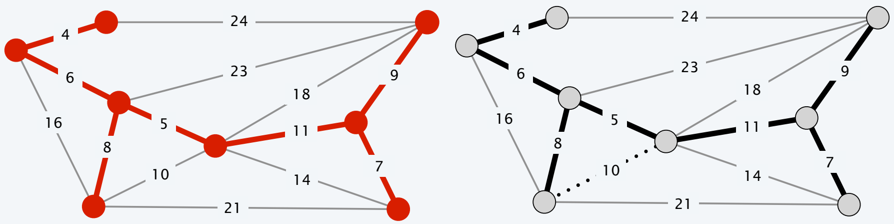
Diverse tecniche greedy forniscono una soluzione ottima.

Algoritmo di Prim.

- comincia con un singolo nodo arbitrario s: T = {s}
- aggiungi l'arco di costo minimo con una sola terminazione in T

Algoritmo di Kruskal

- comincia con T = ∅
- considera gli archi in ordine crescente di costo, aggiungi un arco a T se non crea un ciclo

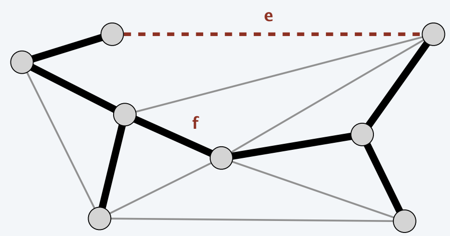


Prim Kruskal

Osservazioni

Cicli.

- Aggiungendo un arco e ad un albero (ricoprente) T crea un ciclo C.
- Eliminando un arco $f \in C$ da $T \cup \{ e \}$ otteniamo un nuovo albero ricoprente.



T = (V, F)

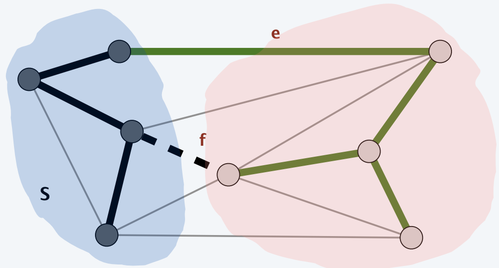
Osservazione. Se $c_e < c_f$, allora T NON è un MST.

Ottimalità dell'algoritmo di Prim

Teorema. L'algoritmo di Prim produce un MST

Dim

- sia f il primo arco scelto da Prim che non è in un MST
- S è il sottoalbero costruito da Prim che è parte di un MST
- gli archi verdi sono la restante parte di un tale MST T
- se aggiungiamo f creiamo un ciclo



- $c_f < c_e$ (perché scelto da Prim) quindi T - e ∪ f ha costo inferiore a T

Algoritmo di Prim: implementazione

L'algoritmo di Prim può essere implementato in tempo $O(m \log n)$.

- quasi identica a quella dell'algoritmo di Dijkstra.
  [$d(v)$ = minimo costo di un arco tra v e S]

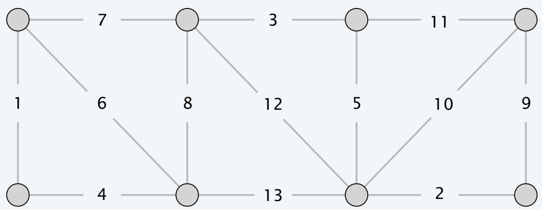
```
PRIM (V, E, c)

Crea una coda a priorità S.
s ← un qualsiasi nodo di V.
FOR EACH v ≠ s : d(v) ← ∞; d(s) ← 0.
FOR EACH v : Insert (S, v, d(v)).
WHILE (S is not empty)
   (u, d[u]) ← Extract-min(S).
   FOR EACH arco (u, v) ∈ Adj[u]:
      IF d(v) > c(u, v)
         Decrease-val(S, v, c(u, v)).
         d(v) ← c(u, v).
```


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

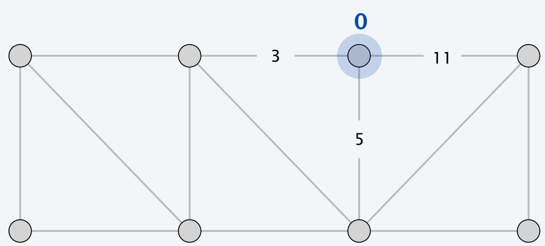


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

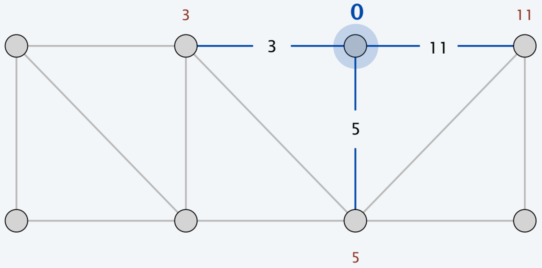
Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.



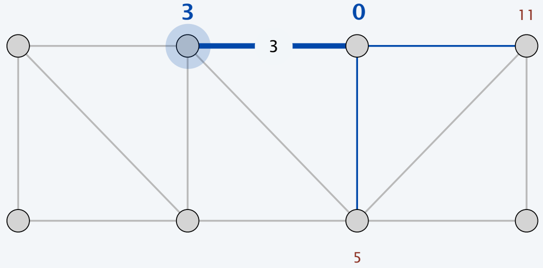
Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.
Ripeti $n - 1$ volte:
- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.



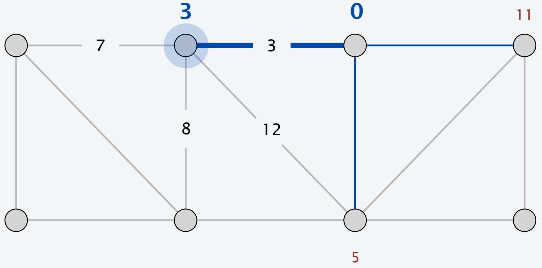
Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.
Ripeti $n - 1$ volte:
- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.



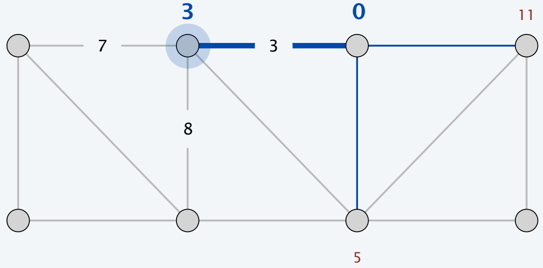
Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.
Ripeti $n - 1$ volte:
- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.



Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.
Ripeti $n - 1$ volte:
- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

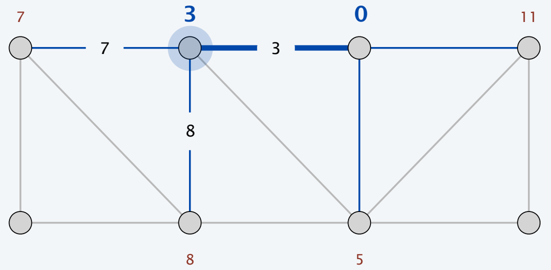


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n-1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

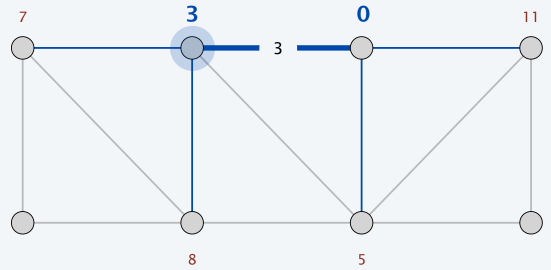


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n-1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

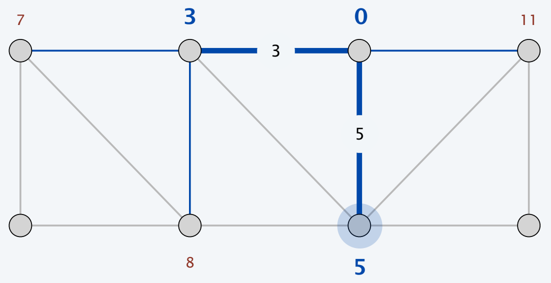


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n-1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

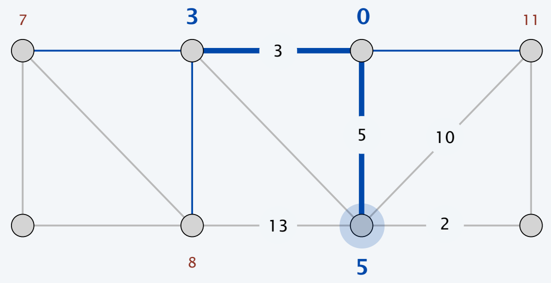


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n-1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

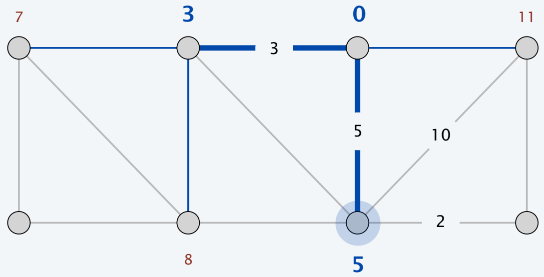


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

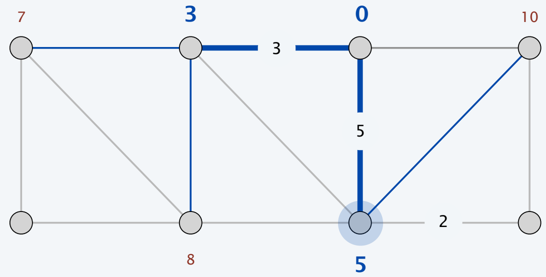


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

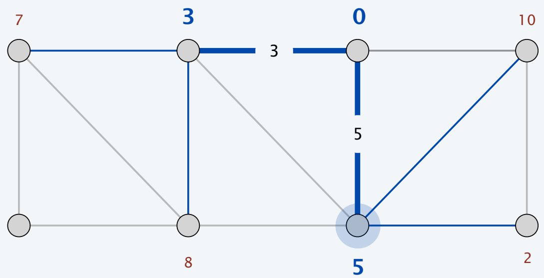


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

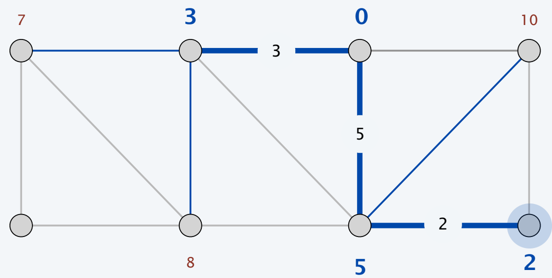


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

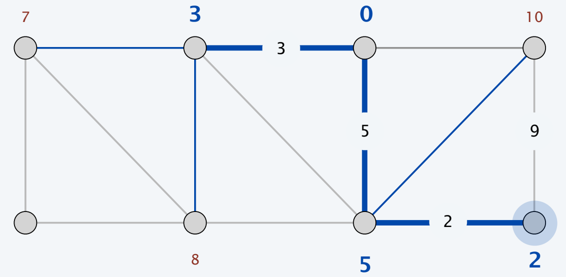


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

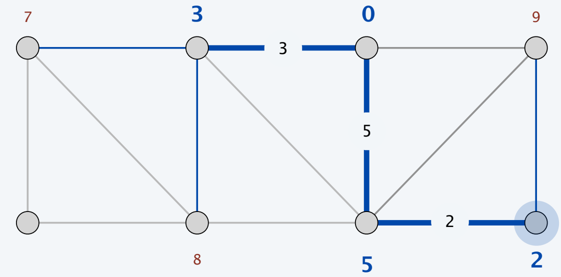


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

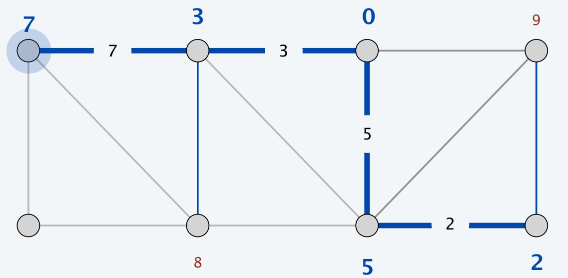


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

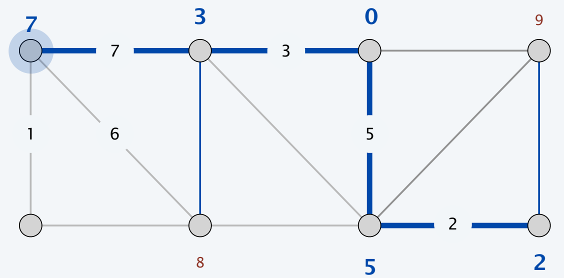


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

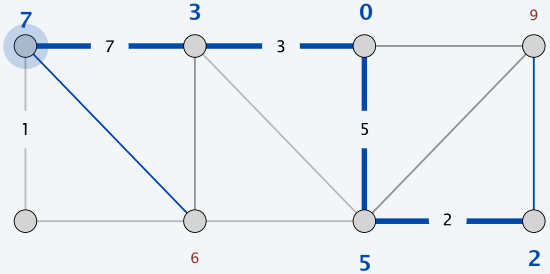


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

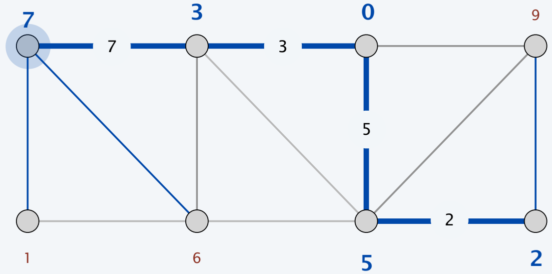


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

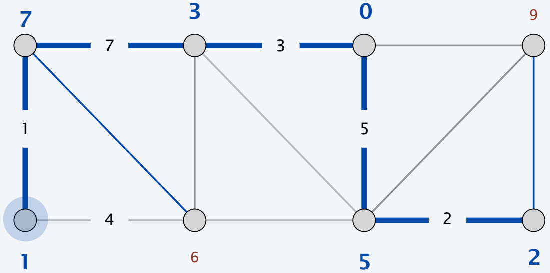


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

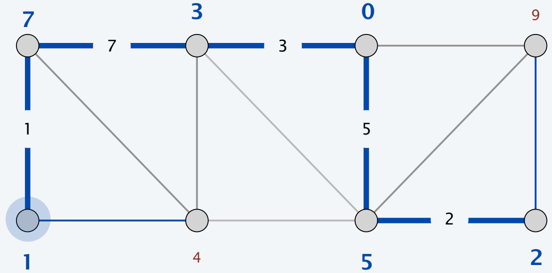


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

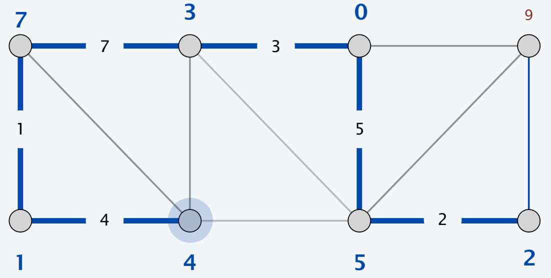


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.

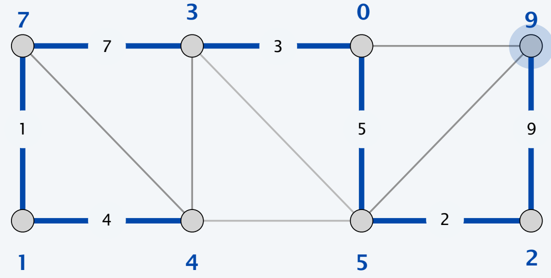


Algoritmo di Prim - esempio

Comincia con S = un nodo qualsiasi.

Ripeti $n - 1$ volte:

- Aggiugi all'albero l'arco di minimo costo con un solo vertice in S.
- Aggiungi il nuovo nodo ad S.



Algoritmo di Prim: implementazione

L'algoritmo di Prim può essere implementato in tempo $O(m \log n)$.

- quasi identica a quella dell'algoritmo di Dijkstra.

  [$d(v)$ = minimo costo di un arco tra v e S]

PRIM (V, E, c)

Crea una coda a priorità S.

$s \leftarrow$ un qualsiasi nodo di V.

FOR EACH $v \neq s$: $d(v) \leftarrow \infty$; $d(s) \leftarrow 0$.

FOR EACH v : *Insert* (S, v, $d(v)$).

WHILE (S *is not empty*)

    (u, $d[u]$) $\leftarrow$ *Extract-min*(S).

    FOR EACH arco $(u, v) \in Adj[u]$:

        IF $d(v) > c(u, v)$

            Decrease-val(S, v, $c(u, v)$).

            $d(v) \leftarrow c(u, v)$.

Σ_u *degree(u) = 2m*

Running time

T(n) = O(m log n)