Algoritmi greedy
IV parte

Progettazione di Algoritmi a.a. 2022-23
Matricole congrue a 1
Docente: Annalisa De Bonis

118

118

Minimo albero ricoprente (Minimum spanning tree)

- Supponiamo di voler creare una rete che interconnetta un insieme di posizioni $V = \{v_1, v_2, \ldots, v_n\}$ in modo che per ogni coppia di posizioni esista un percorso che li collega.
- Alcune coppie di posizioni possono essere collegate direttamente.
- Stabilire un collegamento diretto tra una coppia di posizioni ha un costo che dipende da vari parametri.
- L'obiettivo è di utilizzare esattamente n-1 di questi collegamenti diretti tra coppie di posizioni in modo da connettere l'intera rete e da minimizzare la somma dei costi degli n-1 collegamenti stabiliti .
- Esempi di applicazione alla progettazione di una rete sono.
  - Reti telefoniche, idriche, televisive, stradali, di computer

PROGETTAZIONE DI ALGORITMI A.A. 2022-23
A. De Bonis

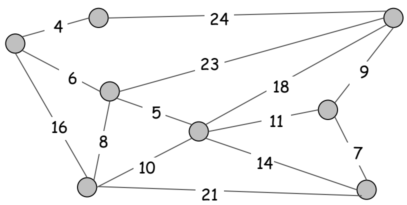
119

Minimo albero ricoprente (Minimum spanning tree o in breve MST)

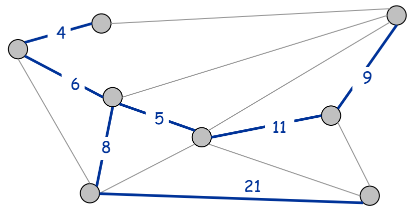
- Grafo non direzionato connesso $G = (V, E)$.
- Per ogni arco e, c_e = costo dell'arco e (c_e numero reale).

- **Def. Albero ricoprente (spanning tree).** Sia dato un grafo non direzionato connesso $G = (V, E)$. Uno spanning tree di G è un sottoinsieme di archi $T \subseteq E$ tale che $|T|=n-1$ e gli archi in T non formano cicli (in altre parole T forma un albero che ha come nodi tutti i nodi di G).
- **Def.** Sia dato un grafo non direzionato connesso $G = (V, E)$ tale che ad ogni arco e di G è associato un costo c_e .Per ogni albero ricoprente T di G , definiamo il costo di T come $\sum_{e \in T} c_e$

.



$G = (V, E)$

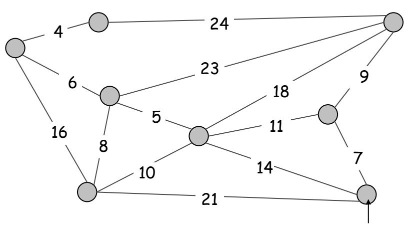


T, $\Sigma_{e \in T}\, c_e = 64$

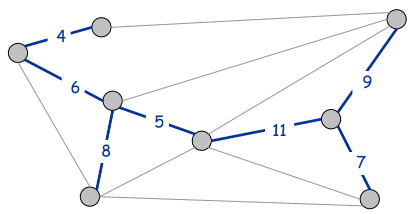
120

Minimo albero ricoprente (Minimum spanning tree o in breve MST)

- Input:
  - Grafo non direzionato connesso $G = (V, E)$.
  - Per ogni arco e, c_e = costo dell'arco e.
- **Minimo albero ricoprente.** Sia dato un grafo non direzionato connesso $G = (V, E)$ con costi c_e degli archi a valori reali. Un minimo albero ricoprente è un sottoinsieme di archi $T \subseteq E$ tale che T è un albero ricoprente di costo minimo.



$G = (V, E)$



T, $\Sigma_{e \in T}\, c_e = 50$

121

Minimo albero ricoprente (Minimum spanning tree o in breve MST)

- Il problema di trovare un minimo albero ricoprente non può essere risolto con un algoritmo di forza bruta
- Teorema di Cayley. Ci sono n^{n-2} alberi ricoprenti del grafo completo K_n.

Algoritmi greedy per MST

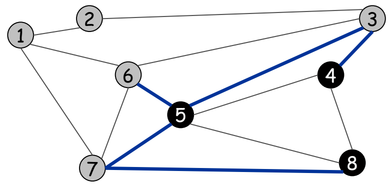
- Kruskal. Comincia con T = ϕ. Considera gli archi in ordine non decrescente di costo. Inserisce un arco e in T se e solo il suo inserimento non determina la creazione di un ciclo in T

- Inverti-Cancella. Comincia con T = E. Considera gli archi in ordine non crescente dei costi. Cancella e da T se e solo se la sua cancellazione non rende T disconnesso.

- Prim. Comincia con un certo nodo s e costruisce un albero T avente s come radice. Ad ogni passo aggiunge a T l'arco di peso più basso tra quelli che hanno esattamente una delle due estremità in T (se un arco avesse entrambe le estremità in T, la sua introduzione in T creerebbe un ciclo)

Taglio

- Taglio. Un taglio è una partizione [S,V-S] dell'insieme dei vertici del grafo.
- Insieme di archi che attraversano il taglio [S,V-S]. Sottoinsieme D di archi che hanno un'estremità in S e una in V-S.



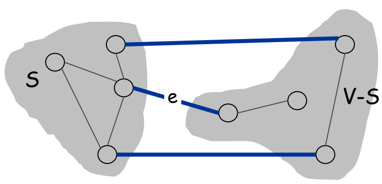
Taglio [S,V-S] = ({ 4, 5, 8 }, { 1,2,3,6,7 })
Archi che attraversano [S,V-S] D = 5-6, 5-7, 3-4, 3-5, 7-8

124

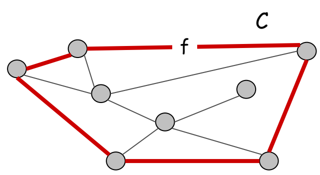
124

Algoritmi Greedy

- Proprietà del taglio. Sia S un qualsiasi sottoinsieme di nodi e sia e un arco di costo minimo che attraversa il taglio [S,V-S] . Esiste un albero ricoprente che contiene e.
- Proprietà del ciclo. Sia C un qualsiasi ciclo e sia f un arco di costo massimo in C. Esiste un minimo albero ricoprente che non contiene f.



e è nello MST



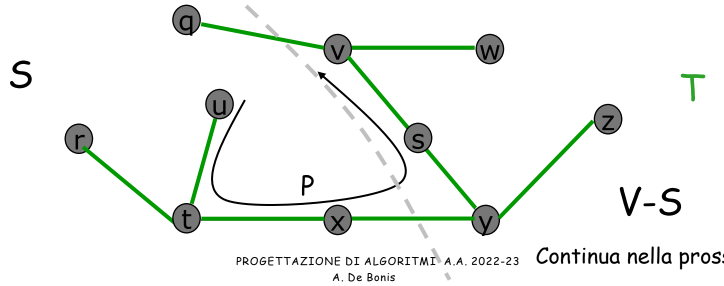
f non è nello MST

125

125

Proprietà del taglio

- Proprietà del taglio. Sia S un qualsiasi sottoinsieme di nodi e sia e=(u,v) un arco di costo minimo che attraversa il taglio [S,V-S]. Esiste un minimo albero ricoprente che contiene e.
- Dim. Dimostriamo che esiste un minimo albero ricoprente che contiene e=(u,v).
- Sia T un minimo albero ricoprente. Se e=(u,v)∈ T allora la dimostrizione termina. Supponiamo ora che e=(u,v) ∉T e facciamo vedere che è possibile ottenere a partire da T un altro albero ricoprente T' che contiene e=(u,v) e che ha lo stesso costo di T (T' sara` quindi anch'esso minimo).
- Sia P il percorso da u a v in T. In T non ci sono altri percorsi da u a v altrimenti ci sarebbe un ciclo

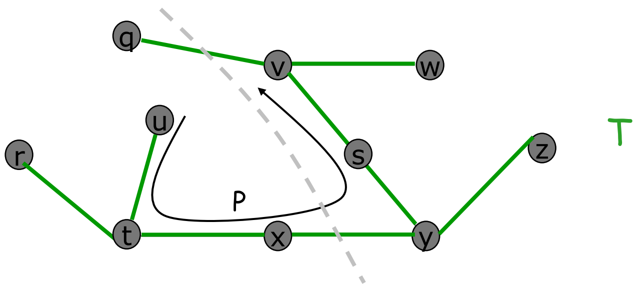


Continua nella prossima slide

126

Proprietà del taglio

- Siccome u e v sono ai lati opposti del taglio [S,V-S] allora il percorso P da u a v in T deve comprendere un arco f=(x,y) che attraversa il taglio [S,V-S]

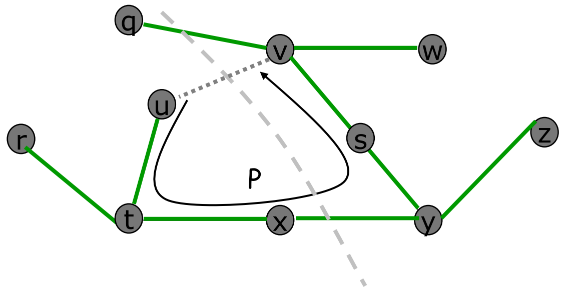


Continua nella prossima slide

127

Proprietà del taglio

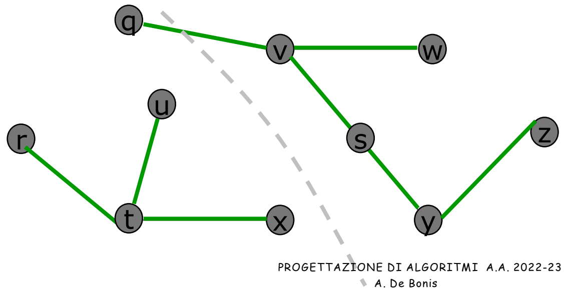
- Poichè (x,y) ≠ (u,v) e poichè entrambi attraversano il taglio e (u,v) ha peso minimo tra quelli che attraversano il taglio allora si ha $c_e \leq c_f$



128

Proprietà del taglio

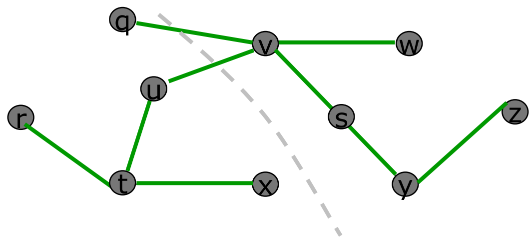
- Poichè (x,y) ≠ (u,v) e poichè entrambi attraversano il taglio e (u,v) è l'arco di peso minimo tra quelli che attraversano il taglio allora si ha $c_e \leq c_f$
- f=(x,y) si trova sull'unico percorso P che connette u a v in T
- Se togliamo f=(x,y) da T dividiamo T in due alberi, uno contente u e l'altro contenente v



129

Proprietà del taglio

- Se introduciamo e=(u,v) riconnettiamo i due alberi ottenendo un nuovo albero ricoprente T' = T-{f}∪{e} dove tutte le coppie di nodi che prima erano connesse da un percorso contenente (x,y), ora sono connesse da un percorso che passa attraverso (u,v)
- Il costo di T' è $c(T')=c(T)-c_f+c_e \leq c(T)$. Siccome stiamo assumendo che T è un **minimo albero** ricoprente allora non puo` valere il "minore" ma vale l'uguaglianza → $c(T')=c(T)-c_f+c_e =c(T)$.
- Si noti che T' non contiene cicli perché l'unico ciclo su cui si potrebbe venire a trovare (u,v) quando lo aggiungiamo a T è quello formato da P e dall'arco (u,v) ma il percorso P non è in T' perchè abbiamo rimosso l'arco (x,y).

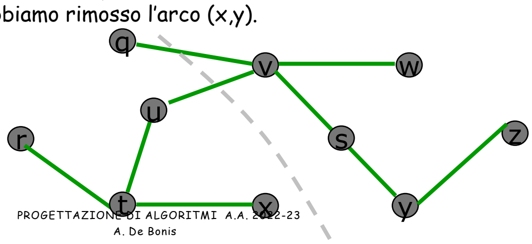


130

Proprietà del taglio (sintesi della dimostrazione)

- Poichè (x,y) ≠ (u,v) e poichè entrambi attraversano il taglio e (u,v) è l'arco di peso minimo tra quelli che attraversano il taglio allora si ha $c_e \leq c_f$
- f=(x,y) si trova sull'unico percorso che connette u a v in T
- Se togliamo f=(x,y) da T dividiamo T in due alberi, uno contente u e l'altro contenente v
- Il costo di T' è $c(T')=c(T)-c_f+c_e \leq c(T)$. Siccome stiamo assumendo che T è un **minimo albero** ricoprente allora non puo` valere il "minore" ma vale l'uguaglianza → $c(T')=c(T)-c_f+c_e =c(T)$.
- Il costo di T' è $c(T')=c(T)-c_f+c_e \leq c(T)$. Siccome stiamo assumendo che T è un minimo albero ricoprente allora non puo` valere il "minore stretto" ma vale l'uguaglianza è $c(T')=c(T)-c_f+c_e =c(T)$.

▪Si noti che T' non contiene cicli perché l'unico ciclo su cui si potrebbe venire a trovare (u,v) quando lo aggiungiamo a T è quello formato da P e dall'arco (u,v) ma il percorso P non è in T' perchè abbiamo rimosso l'arco (x,y).



131

Osservazione per il caso in cui c'è un unico arco di costo minimo che attraversa il taglio

- Se per un certo taglio [S,V-S], e=(u,v) è l'unico arco di costo minimo che lo attraversa allora ogni minimo albero ricoprente deve contenere l'arco e=(u,v).
- Se infatti assumessimo che T è un minimo albero ricoprente e che e=(u,v) $\notin$ T allora ripetendo la stessa dimostrazione che abbiamo fatto prima questa volta pero` con $c_e < c_f$ avremmo:
- $c(T')=c(T)-c_f+c_e<c(T)$ e arriveremmo a contraddire il fatto che T è minimo albero ricoprente.

- Si noti che se i costi del grafo sono a due a due distinti allora per ciascun taglio esiste un unico arco di costo minimo che attraversa il taglio e questo deve necessariamente essere nel minimo albero ricoprente→ esiste un unico minimo albero ricoprente.

Osservazione per il caso in cui ci sono due o piu` archi di costo minimo che attraversano il taglio

- Supponiamo che per un certo taglio [S,V-S] esistano due o piu` archi con costo minimo che lo attraversano.
- Sia (u,v) uno di questi archi e assumiamo di essere nel caso in cui (u,v) non fa parte dello MST T .
- Nella parte di dimostrazione in cui si considera il caso in cui (u,v) non fa parte dello MST T, la disuguaglianza in $c(T')=c(T)-c_f+c_e \leq c(T)$ in realta` vale con il segno di uguale dal momento che per ipotesi T è uno MST e di conseguenza il suo costo c(T ) non puo` essere inferiore a quello di T'
  $c(T)-c_f+c_e = c(T) \rightarrow -c_f+c_e = 0 \rightarrow -c_f = c_e$.
- In altre parole, se e=(u,v) non è nello MST T allora lo MST deve contenere un altro arco f=(x,y) che, come abbiamo visto nella dimostrazione della proprieta`, deve attraversare anch'esso [S,V-S] e che, per quanto osservato al punto precedente, ha lo stesso costo di e=(u,v)

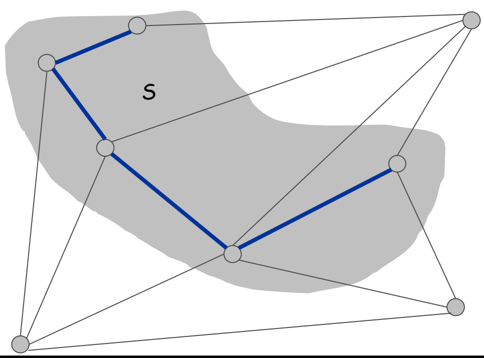
→f=(x,y) è anch'esso un arco di costo minimo che attraversa [S,V-S].

Un utile riformulazione della proprieta` del taglio

- Guardando bene la dimostrazionne della proprieta` del taglio e quanto osservato nelle due slide precedenti si vede che la proprieta` puo` essere riformulata come segue:
- Proprieta` del taglio: Sia [S,V-S] un taglio del grafo e sia (u,v) un arco di costo minimo che attraversa il taglio. Puo` accadere una delle due seguenti cose:
  1. (u,v) è l'unico arco di costo minimo che attraversa il taglio e in questo caso (u,v) fa parte di ogni MST.
  2. Oltre a (u,v) esistono uno piu` archi di costo minimo che attraversano [S,V-S]. In questo caso preso un qualsiasi MST T, puo` accadere o che (u,v) sia gia` in T o che T non contenga (u,v) . In quest'ultimo caso T contiene un arco (x,y) diverso da (u,v) e con lo stesso costo di (u,v) che attraversa [S,V-S] e che puo` essere sostituito con (u,v) in modo da ottenere un nuovo MST che contiene (u,v) e tutti gli archi di T diversi da (x,y).

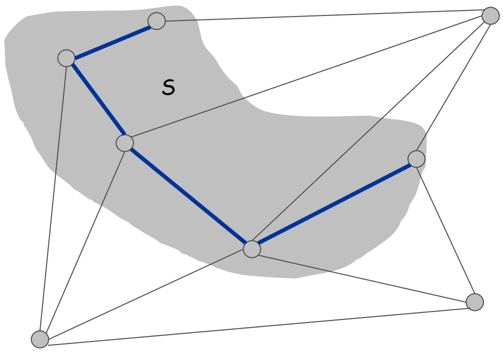
Algoritmo di Prim

- Algoritmo di Prim. [Jarník 1930, Prim 1957, Dijkstra 1959,]
- Ad ogni passo T è un sottoinsieme di archi dello MST
- S= insieme di nodi di T
- Inizializzazione: Pone in S un qualsiasi nodo u. Il nodo u sarà la radice dello MST
- Ad ogni passo aggiunge a T un arco (x,y) di costo minimo tra tutti quelli che congiungono un nodo x in S ad un nodo y in V-S (scelta greedy)
- Termina quando S=V



Correttezza dell'algoritmo di Prim

- L'algoritmo di Prim ad ogni passo inserisce in T un arco di costo minimo tra quelli che attraversano il taglio [S,V-S]. Quindi per ogni arco e di T esiste un taglio per cui e è un arco di costo più piccolo tra quelli che lo attraversano.
- La proprietà del taglio implica che esiste un minimo albero ricoprente che contiene ciascuno degli archi selezionati dall'algoritmo → ogni arco selezionato è contenuto in un MST



continua nella prossima slide

136

136

Correttezza dell'algoritmo di Prim

- Potrebbe sorgere un obiezione: chi mi dice che gli archi selezionati facciano pero` parte di uno stesso MST?
- Supponiamo che fino ad un certo punto Prim abbia selezionato un certo insieme di archi A e che questi siano effettivamente tutti all'interno di uno stesso MST. Per come funziona Prim, A è un albero che connettera` un certo sottoinsieme S di nodi del grafo. Potrebbe accadere che selezionando un arco (u,v) di costo minimo che attraversa il taglio [S,V-S], come fa Prim, A∪{(u,v)} non sia piu` un sottoinsieme di MST?
- Osserviamo che cio` non puo` accadere se (u,v) è l'unico arco di costo minimo che attraversa il taglio in quanto in questo caso (u,v) fa sicuramente parte di ogni MST e quindi anche di quelli che contengono gli archi di A (si veda slide 134)
- Consideriamo allora il caso in cui ci sono piu` archi di costo minimo che attraversano [S,V-S] e sia T uno MST che contiene tutti gli archi di A (deve esisterne almeno uno) perche' stiamo assumendo che A è un sottoinsieme di un MST.
- Il punto 2 della proprieta` del taglio (slide 134) ci dice che o (u,v) è nello MST T o che esiste un arco (x,y) che attraversa [S,V-S] che puo` essere rimpiazzato con (u,v) in modo che il nuovo albero sia ancora un MST e contenga tutti gli altri di T diversi da (x,y)→ otteniamo un nuovoMST che contiene sia (u,v) che tutti gli archi di A.

continua nella prossima slide

137

137

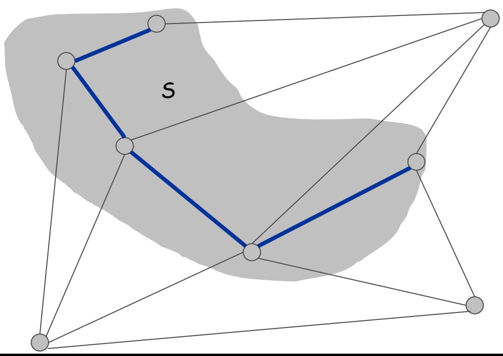
Correttezza dell'algoritmo di Prim

- Ci resta da dimostrare che al termine dell'algoritmo di Prim, l'albero T costruito dall'algoritmo è un albero ricoprente, cioè che T effettivamente connette ogni nodo di V.

- Ciò è un'ovvia conseguenza del fatto che l'algoritmo si ferma solo quando S=V, cioè quando ha attaccato tutti i vertici all'albero, si ha che T è un albero

In conclusione T è un albero ricoprente che contiene esclusivamente archi che fanno parte dello MST e quindi T è lo MST.

NB: quando i costi sono a due a due distinti c'è un unico MST in quanto per ogni taglio c'è un unico arco di costo minimo che lo attraversa



Implementazione dell'algoritmo di Prim con coda a priorità

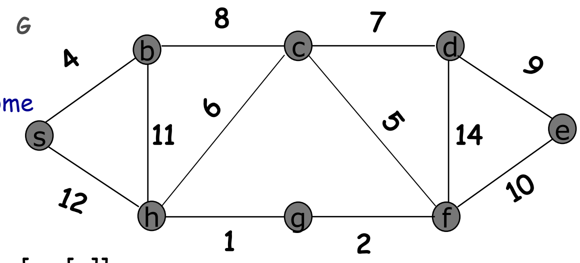
- Mantiene un insieme di vertici esplorati S.
- Per ogni nodo non esplorato v, mantiene a[v] = costo dell'arco di costo più basso tra quelli che uniscono v ad un nodo in S
- Mantiene coda a priorità Q delle coppie (a[v],v) $v \notin S$
- Stessa analisi dell'algoritmo di Dijkstra con coda a priorità:
- $O(n^2)$ con array o lista non ordinati;
- O(m log n+nlog n) con heap. Siccome nel problema dello MST il grafo è connesso allora $m \geq n-1$ e O(m log n+nlog n) =O(mlog n)

```
Prim's Algorithm (G,c)
Let S be the set of explored nodes
For each u not in S, we store the cost a[u]
Let Q be a priority queue of pairs (a[u],u) s.t. u is not in S

For each u in V insert (Q, ∞,u)  in Q EndFor
While (Q is not empty)
 (a[u],u)<-- ExtractMin(Q)
  Add u to S
  For each edge e=(u,v)
   If  ((v not in S) && (cₑ< a[v]))
     ChangeKey(Q,v, cₑ)

EndWhile
```

139

Un esempio

s = nodo scelto come radice



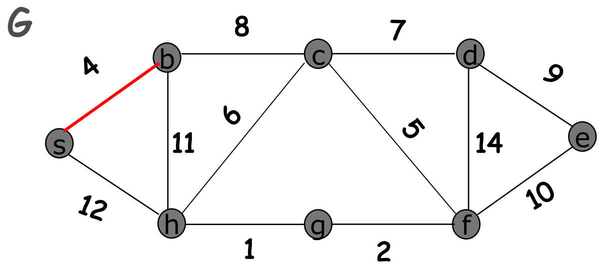
[u,a[u]]

Q = {[s, ∞],[b ,∞],[c, ∞],[d, ∞],[e, ∞],[f, ∞],[g, ∞],[h, ∞]}

Si estrae s da Q e si aggiornano i campi a dei nodi adiacenti ad s

Q = {[b ,4],[c, ∞],[d, ∞],[e, ∞],[f, ∞],[g, ∞],[h, 12]}

Un esempio



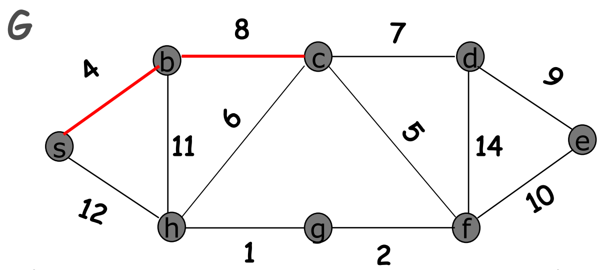
Q = {[b,4],[c, ∞],[d, ∞],[e, ∞],[f, ∞],[g, ∞],[h, 12]}

- Si estrae b da Q.

- Si aggiornano i campi a dei nodi adiacenti a b che si trovano in Q

Q = {[c,8],[d, ∞],[e, ∞],[f, ∞],[g, ∞],[h, 11]}

Un esempio



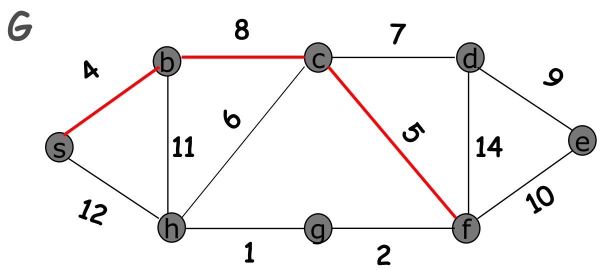
Q = {[c, 8],[d, ∞],[e, ∞],[f, ∞],[g, ∞],[h, 11]}

- Si estrae c da Q.

- Si aggiornano i campi a dei nodi adiacenti a c che si trovano in Q

Q = {[d, 7],[e, ∞],[f, 5],[g, ∞],[h, 6]}

Un esempio

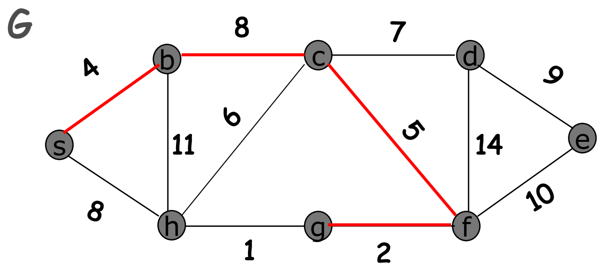


Q = {[d, 7],[e, ∞],[f, 5],[g, ∞],[h, 6]}

Si estrae f da Q e si aggiornano i campi a dei nodi adiacenti a f che si trovano in Q

Q = {[d, 7],[e, 10],[g, 2],[h, 6]}

Un esempio

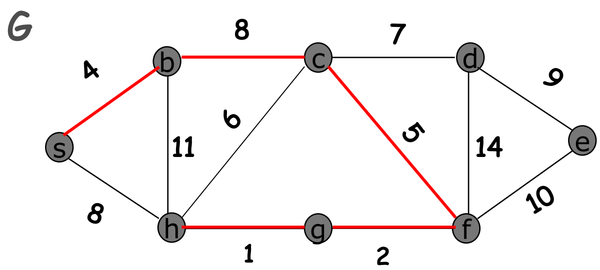


Q = {[d, 7],[e, 10],[g, 2],[h, 6]}

Si estrae g da Q e si aggiornano i campi a dei nodi adiacenti a g che si trovano in Q

Q = {[d, 7],[e, 10],[h, 1]}

Un esempio

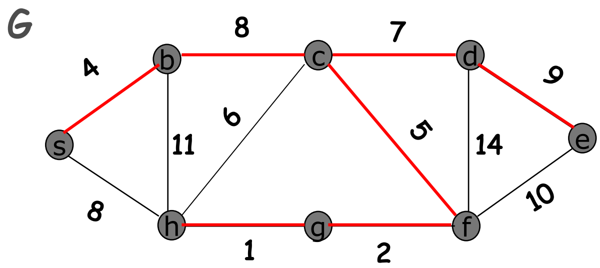


Q = {[d, 7],[e, 10],[h, 1]}

Si estrae h da Q e si aggiornano i campi a dei nodi adiacenti a h che si trovano in Q

Q = {[d, 7],[e, 10]}

Un esempio



Q = {[d, 7],[e, 10]}

Si estrae d da Q e si aggiorna il campo a di e

Q = {[e,9]}

Si estrae e da Q

Algoritmo di Prim

- Esercizio: modificare il codice dell'algoritmo di Prim in modo che l'algoritmo restituisca l'insieme di T degli archi che fanno parte dello MST.