Algoritmi greedy
V parte

Progettazione di Algoritmi a.a. 2021-22
Matricole congrue a 1
Docente: Annalisa De Bonis

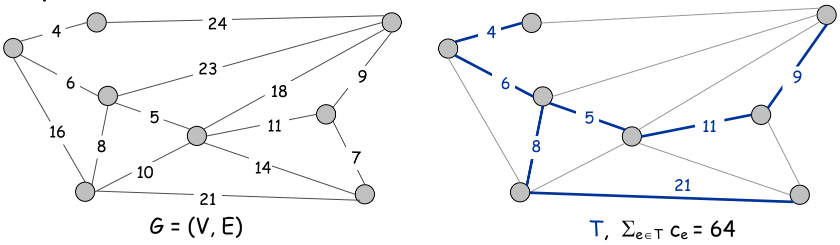
Minimo albero ricoprente (Minimum spanning tree)

- 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

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 $\sum_{e \in T} c_e$

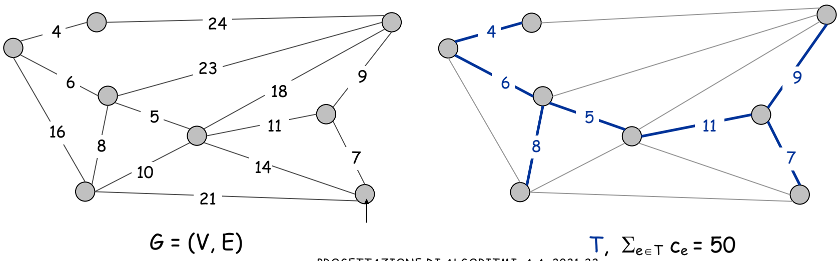


PROGETTAZIONE DI ALGORITMI A.A. 2021-22
A. De Bonis

119

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.



PROGETTAZIONE DI ALGORITMI A.A. 2021-22
A. De Bonis

120

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

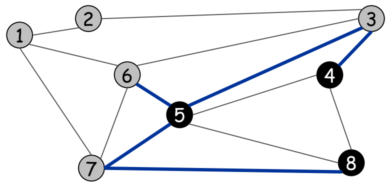
- Kruskal. Comincia con $T = \phi$. Considera gli archi in ordine non decrescente di costo. Inserisce un arco e in T se e solo il suo inserimento non determina la creazione di un ciclo in T

- 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.



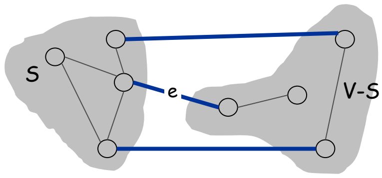
Taglio [S,V-S] = ({ 4, 5, 8 }, { 1,2,3,6,7 })
Archi che attraversano [S,V-S] D = 5-6, 5-7, 3-4, 3-5, 7-8

123

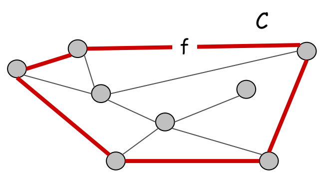
123

Algoritmi Greedy

- **Per il momento** assumiamo per semplicità che i pesi degli archi siano a due a due distinti
  - Vedremo che questa assunzione implica che il minimo albero ricoprente è unico.
- Proprietà del taglio. Sia S un qualsiasi sottoinsieme di nodi e sia e l'arco di costo minimo che attraversa il taglio [S,V-S]. Ogni minimo albero ricoprente contiene e.
- Proprietà del ciclo. Sia C un qualsiasi ciclo e sia f l'arco di costo massimo in C. Nessun minimo albero ricoprente contiene f.



e è nello MST



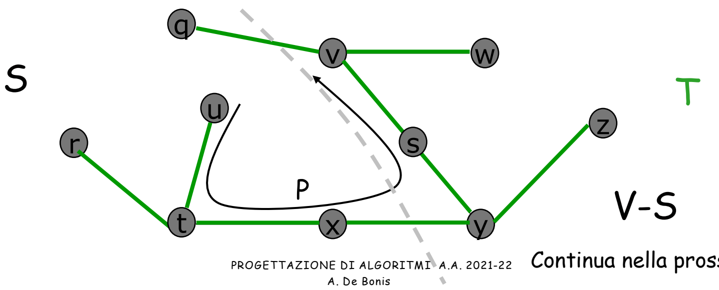
f non è nello MST

124

124

Proprietà del taglio

- Stiamo assumendo (per il momento) che i costi degli archi siano a due a due distinti
- Proprietà del taglio. Sia S un qualsiasi sottoinsieme di nodi e sia e=(u,v) l'arco di costo minimo che attraversa il taglio [S,V-S]. Ogni minimo albero ricoprente contiene e.
- Dim. (nel caso in cui i costi degli archi sono a due a due distinti)
- Sia T un albero ricoprente tale che e=(u,v) $\notin$ T. Dimostriamo che T non può essere un minimo albero ricoprente.
- 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

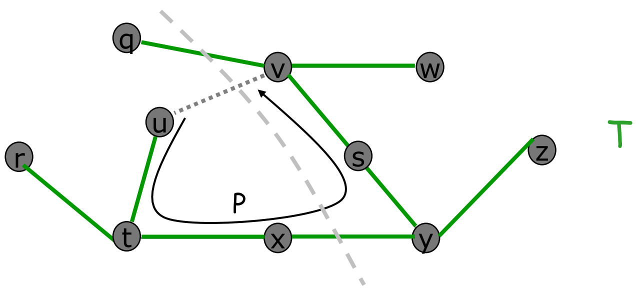


Continua nella prossima slide

125

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]

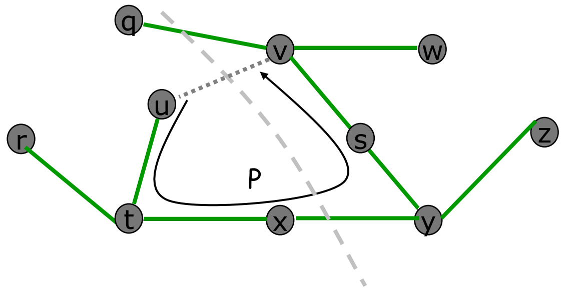


Continua nella prossima slide

126

Proprietà del taglio

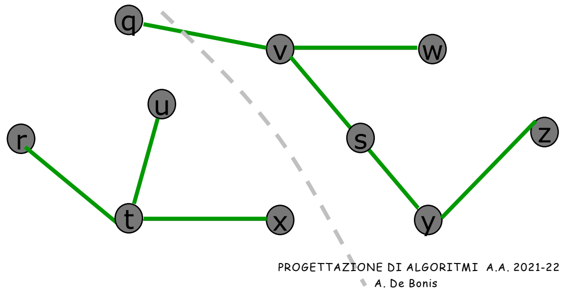
- Poichè (x,y) ≠ (u,v) e poichè entrambi attraversano il taglio e (u,v) è l'arco di peso minimo tra quelli che attraversano il taglio allora si ha $c_e < c_f$



127

Proprietà del taglio

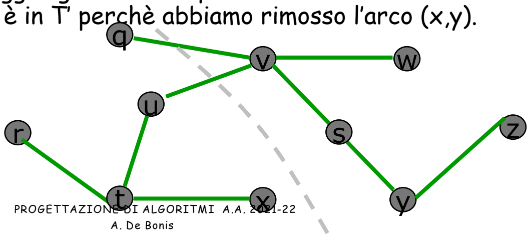
- Poichè (x,y) ≠ (u,v) e poichè entrambi attraversano il taglio e (u,v) è l'arco di peso minimo tra quelli che attraversano il taglio allora si ha $c_e < c_f$
- f=(x,y) si trova sull'unico percorso P che connette u a v in T
- Se togliamo f=(x,y) da T dividiamo T in due alberi, uno contente u e l'altro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



128

Proprietà del taglio

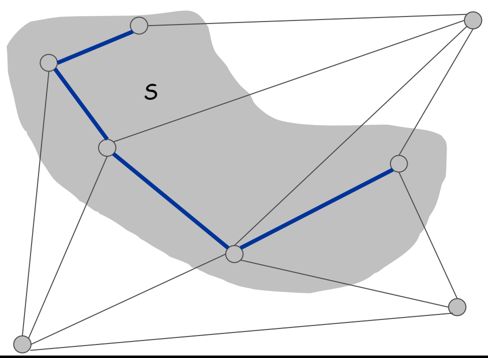
- Poichè $(x,y) \neq (u,v)$ e poichè entrambi attraversano il taglio e (u,v) è l'arco di peso minimo tra quelli che attraversano il taglio allora si ha $c_e < c_f$
- $f=(x,y)$ si trova sull'unico percorso che connette u a v in T
- Se togliamo $f=(x,y)$ da T dividiamo T in due alberi, uno contente u e l'altro contenente v
- Se introduciamo $e=(u,v)$ riconnettiamo i due alberi ottenendo un nuovo albero ricoprente $T' = T-\{f\} \cup \{e\}$ dove tutte le coppie di nodi che prima erano connesse da un percorso contenente (x,y), ora sono connesse da un parcorso che passa attraverso (u,v)
- Il costo di T' è $c(T')=c(T)-c_f+c_e<c(T)$. Ciò vuol dire che T non è un minimo albero ricoprente.
- 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).



129

Algoritmo di Prim

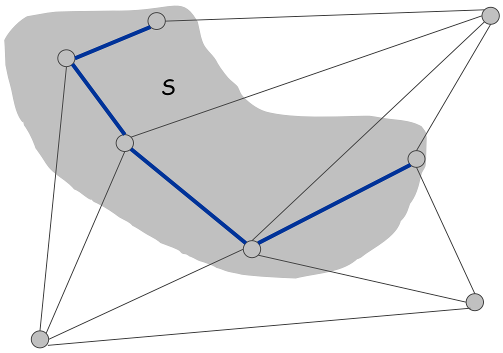
- Algoritmo di Prim. [Jarník 1930, Prim 1957, Dijkstra 1959,]
- Ad ogni passo T è un sottoinsieme di archi dello MST
- S= insieme di nodi di T
- Inizializzazione: Pone in S un qualsiasi nodo u. Il nodo u sarà la radice dello MST
- Ad ogni passo aggiunge a T l'arco (x,y) di costo minimo tra tutti quelli che congiungono un nodo x in S ad un nodo y in V-S (scelta greedy)
- Termina quando S=V



130

Correttezza dell'algoritmo di Prim

- L'algoritmo di Prim ad ogni passo inserisce in T l'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 l'arco di costo più piccolo tra quelli che lo attraversano.
- La proprietà del taglio implica che ogni albero ricoprente deve contenere ciascuno degli archi selezionati dall'algoritmo → T è un sottoinsieme di MST



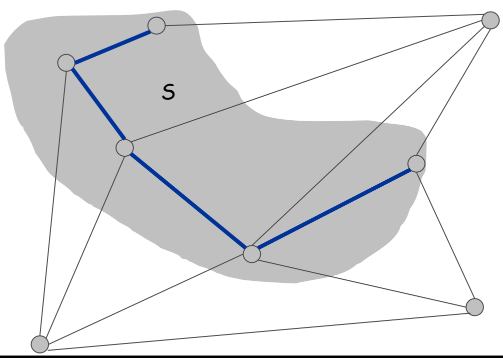
continua nella prossima slide

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

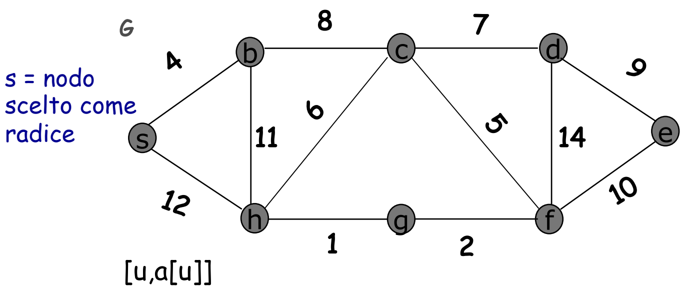
- Mantiene un insieme di vertici esplorati S.
- Per ogni nodo non esplorato v, mantiene a[v] = costo dell'arco di costo più basso tra quelli che uniscono v ad un nodo in S
- Mantiene coda a priorità Q delle coppie (a[v],v) v∉ S
- Stessa analisi dell'algoritmo di Dijkstra con coda a priorità:
- $O(n^2)$ 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≥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
```

133

133

Un esempio



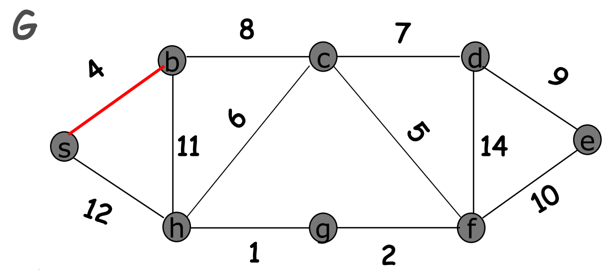
Q = {[s, ∞],[b ,∞],[c, ∞],[d, ∞],[e, ∞],[f, ∞],[g, ∞],[h, ∞]}

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]}

134

Un esempio



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

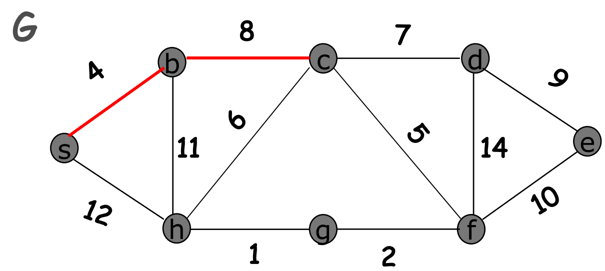
- Si estrae b da Q.

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

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

135

Un esempio



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

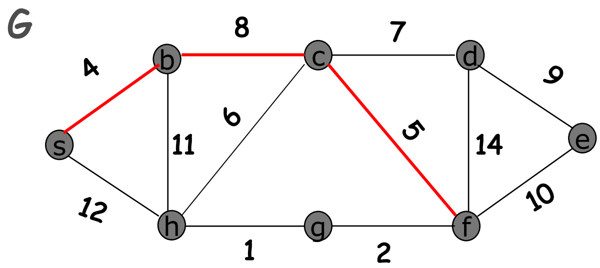
- Si estrae c da Q.

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

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

136

Un esempio

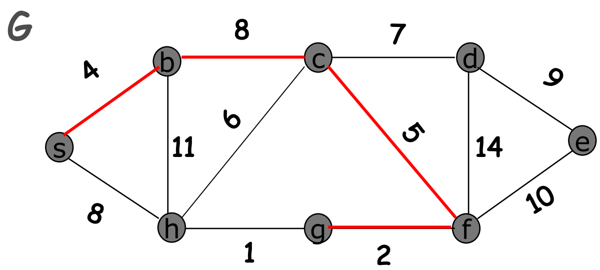


Q = {[d, 7],[e, ∞],[f, 5],[g, ∞],[h, 6]}

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

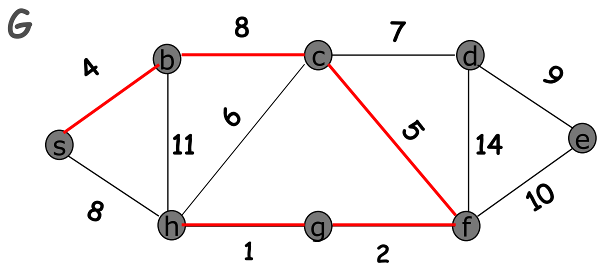


Q = {[d, 7],[e, 10],[g, 2],[h, 6]}

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



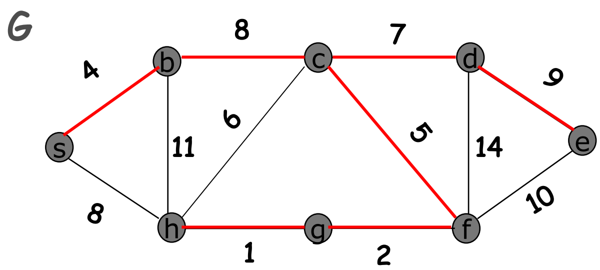
Q = {[d, 7],[e, 10],[h, 1]}

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]}

139

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

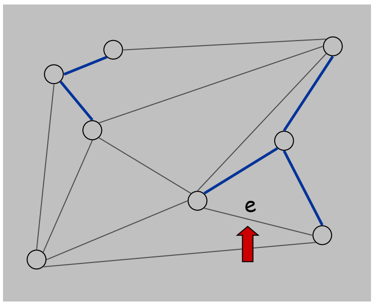
140

Algoritmo di Prim

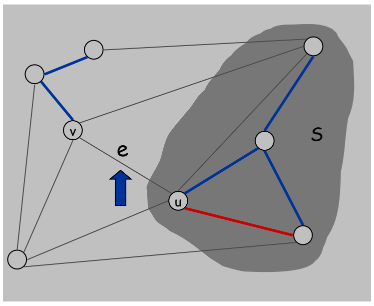
- Esercizio: modificare il codice dell'algoritmo di Prim in modo che l'algoritmo restituisca l'insieme di T degli archi che fanno parte dello MST.

Algoritmo di Kruskal

- Algoritmo di Kruskal's . [Kruskal, 1956]
  - Considera ciascun arco in ordine non decrescente di peso
  - Caso 1: Se e crea un ciclo allora scarta e
  - Case 2: Altrimenti inserisce e in T
  - NB: Siccome gli archi selezionati dall'algoritmo non creano cicli in T allora durante l'esecuzione dell'algoritmo, il grafo formato dai nodi di V insieme agli archi di T è una foresta di alberi, cioè le componenti connesse del grafo (V,T) sono alberi.



Caso 1



Caso 2

Esempio

Archi in MST : rossi (già selezionati) e verdi (non ancora selezionati)

T={(r,q),(t,u),(t,x),(v,s),(s,y)} archi dello MST già inseriti

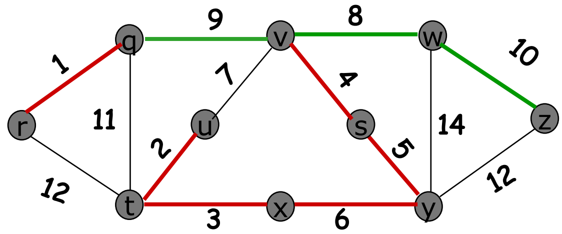
Componenti connesse (alberi) in G_T=(V,T) :

C_1=({r,q},{(r,q)})

C_2={{u,t,x,v,s,y},{(u,t),(t,x), (v,s),(s,y),(x,y)})

C_4=({w},∅)

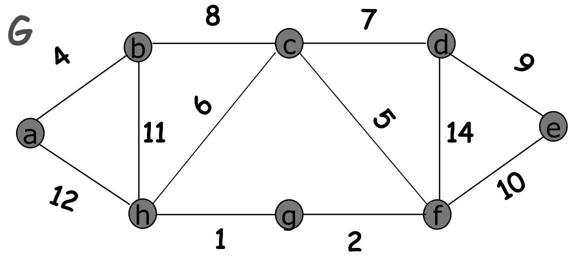
C_5=({z},∅)



Il prossimo arco selezionato è (v,w)

143

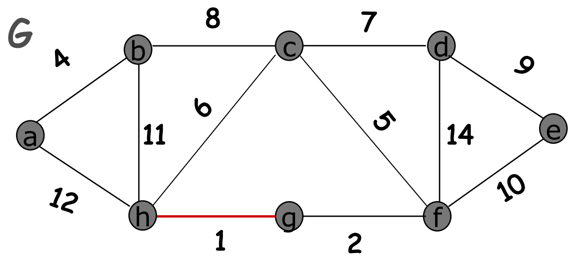
Un esempio



T = { }

144

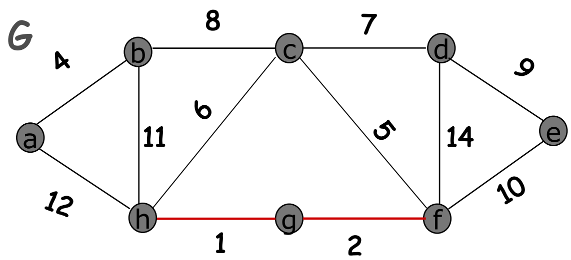
Un esempio



T = {(h,g) }

145

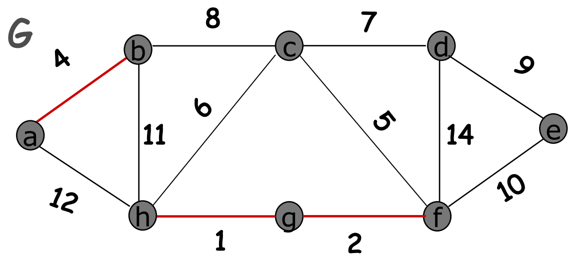
Un esempio



T = {(h,g), (g,f) }

146

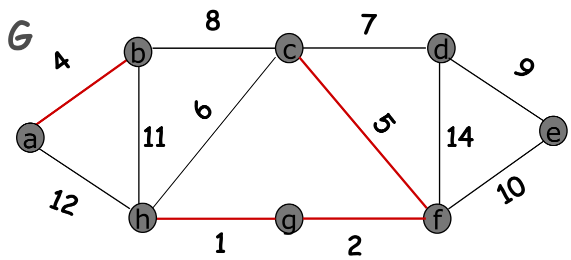
Un esempio



T = {(h,g), (g,f), (a,b) }

147

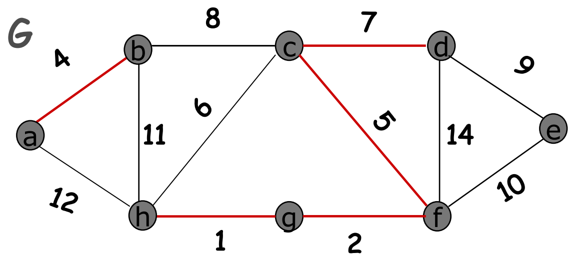
Un esempio



T = {(h,q), (g,f), (a,b),(c,f) }

148

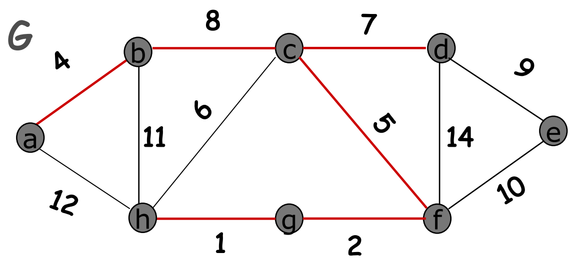
Un esempio



T = {(h,g), (g,f), (a,b),(c,f),(c,d)}

149

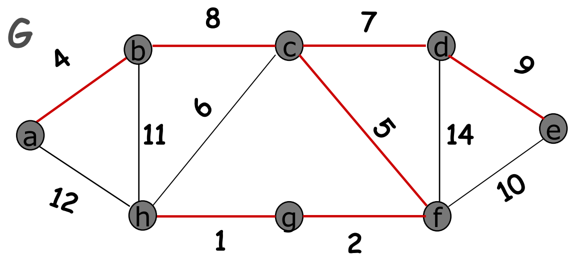
Un esempio



T = {(h,g), (g,f), (a,b),(c,f),(c,d), (b,c)}

150

Un esempio



T = {(h,g), (g,f), (a,b),(c,f),(c,d), (b,c),(d,e) }

Correttezza dell'algoritmo di Kruskal

L'insieme di archi T prodotto dall'algoritmo di Kruskal è un MST

- La dimostrazione è per il caso in cui gli archi hanno costi a due a due distinti
- Prima dimostriamo che ogni arco di T è anche un arco di MST
- Sia e=(u,v) l'arco inserito in un certo passo. **Dimostriamo che esiste un taglio per il quale** e=(u,v) è l'arco di peso minimo che lo attraversa.
  - Osserviamo che T fino a quel momento non contiene un percorso che collega **u** a **v, altrimenti introducendo e=(u,v) in T si creerebbe un ciclo e l'algoritmo scarterebbe (u,v) senza inserirlo in T.**
  - Consideriamo l'albero contenente **u** nel momento in cui e=**(u,v)** viene inserito in T. Chiamiamo S l'insieme dei vertici contenuti in questo albero, cioè l'insieme dei nodi connessi ad u fino a quel momento. Ovviamente **v** non è in S altrimenti esisterebbe un percorso da **u** a **v**. Quindi **e=(u,v)** attraversa il taglio [S, V-S] .

Correttezza dell'algoritmo di Kruskal

- Nella slide precedente abbiamo dimostrato che **e=(u,v)** attraversa il taglio [S, V-S] dove S è l'insieme dei nodi dell'albero in cui si trova u
- Osserviamo che **e=(u,v)** è l'arco di peso minimo tra quelli che attraversano il taglio [S, V-S]
  - Infatti se esistesse un arco (x,y) con x in S e y in V-S con costo inferiore a quello di (u,v), questo arco (x,y) sarebbe stato esaminato prima di (u,v) e x e y si troverebbero entrambi in S quando viene esaminato (u,v) .

Correttezza dell'algoritmo di Kruskal

- Abbiamo dimostrato che per ciascuno degli archi (u,v) selezionati dall'algoritmo di Kruskal esiste un taglio [S, V-S] per cui (u,v) è l'arco di peso minimo tra quelli che attraversano il taglio [S, V-S] .
- Quindi per la proprietà del taglio tutti gli archi selezionati dall'algoritmo sono nel minimo albero ricoprente.
- Ora dimostriamo che al termine dell'esecuzione dell'algoritmo T è un albero ricoprente.
- T è un albero ricoprente perchè
  - l'algoritmo non introduce mai cicli in T (ovvio!)
  - connette tutti i vertici
    - Se così non fosse esisterebbe un insieme W non vuoto, di al più n-1 vertici, tale che non c'è alcun arco di T che connette un vertice di W ad uno di V-W.
    - Siccome il grafo input G è connesso devono esistere uno o più archi in G che connettono vertici di W a vertici di V-W
    - Dal momento che l'algoritmo di Kruskal esamina tutti gli archi avrebbe selezionato sicuramente l'arco di costo minimo tra quelli che connettono un vertice di W ad uno di V-W. Non è quindi possibile che in T non vi sia un arco che connette un nodo in W ad uno in V-W.