Algoritmi greedy
V parte

Progettazione di Algoritmi a.a. 2019-20
Matricole congrue a 1
Docente: Annalisa De Bonis

105

105

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 .

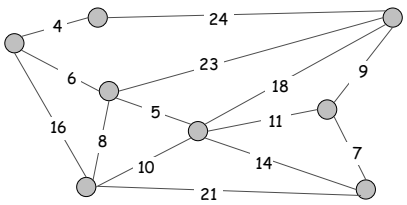
- Esempi di applicazione alla progettazione di una rete sono.
  - Reti telefoniche, idriche, televisive, stradali, di computer

106

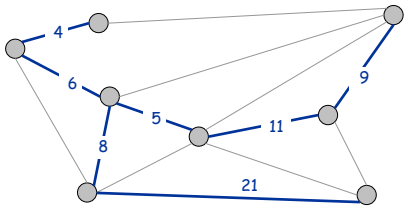
Minimo albero ricoprente (Minimum spanning tree o in breve MST)

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

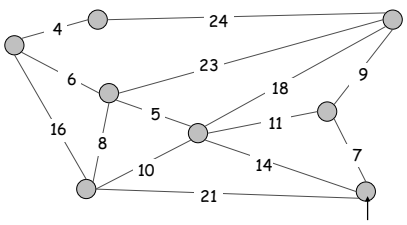
PROGETTAZIONE DI ALGORITMI A.A. 2019-20
A.DE BONIS

107

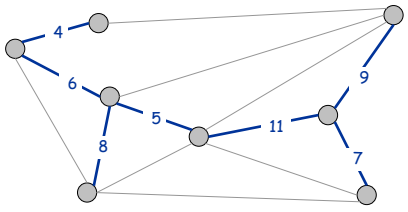
107

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

PROGETTAZIONE DI ALGORITMI A.A. 2019-20
A.DE BONIS

108

108

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

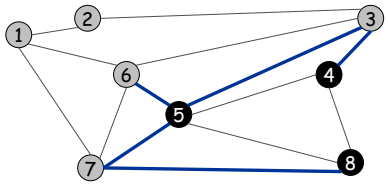
- Il problema di trovare un minimo albero ricoprente non può essere risolto con un algoritmo di forza bruta
- Teorema di Cayley. Ci sono n^{n-2} alberi ricoprenti del grafo completo K_n.

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.



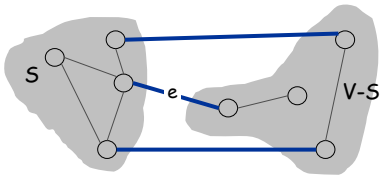
Taglio [S,V-S] = ({ 4, 5, 8 }, { 1,2,3,6,7 })
Archi che attraversano [S,V-S] D = 5-6, 5-7, 3-4, 3-5, 7-8

111

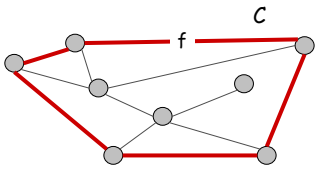
111

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



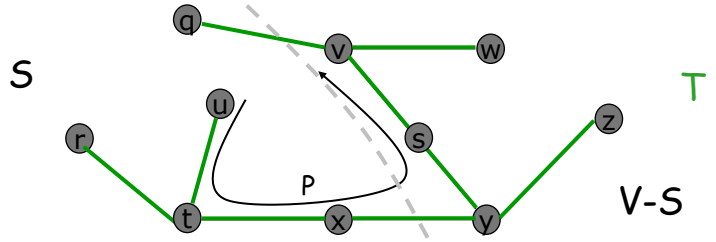
f non è nello MST

112

112

Proprietà del taglio

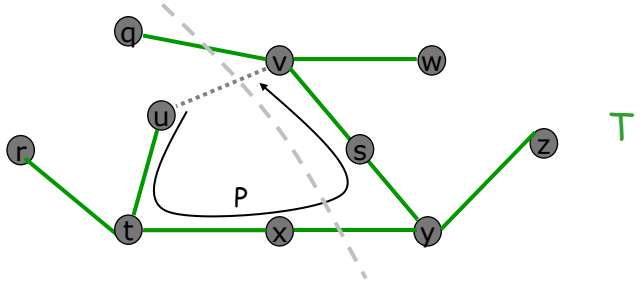
- Stiamo assumendo (per il momento) che i costi degli archi siano a due a due distinti
- Proprietà del taglio. Sia S un qualsiasi sottoinsieme di nodi e sia e=(u,v) l'arco di costo minimo che attraversa il taglio [S,V-S]. Ogni minimo albero ricoprente contiene e.
- Dim. (nel caso in cui i costi degli 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

Proprietà del taglio

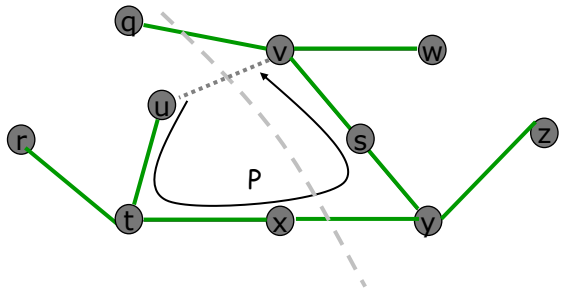
- Siccome u e v sono ai lati opposti del taglio [S,V-S] allora il percorso P da u a v in T deve comprendere un arco f=(x,y) che attraversa il taglio [S,V-S]



Continua nella prossima slide

Proprietà del taglio

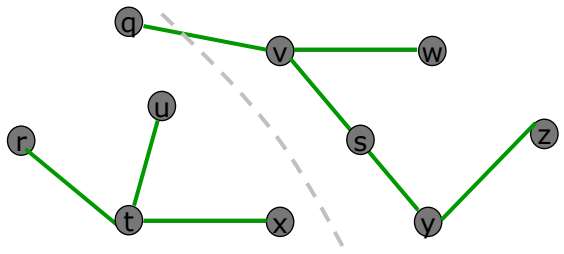
- Poichè (x,y) ≠ (u,v) e poichè entrambi attraversano il taglio e (u,v) è l'arco di peso minimo tra quelli che attraversano il taglio allora si ha $c_e < c_f$



115

Proprietà del taglio

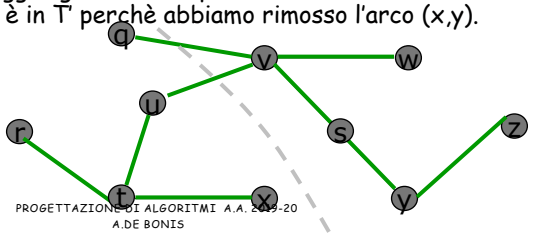
- Poichè (x,y) ≠ (u,v) e poichè entrambi attraversano il taglio e (u,v) è l'arco di peso minimo tra quelli che attraversano il taglio allora si ha $c_e < c_f$
- f=(x,y) si trova sull'unico percorso P che connette u a v in T
- Se togliamo f=(x,y) da T dividiamo T in due alberi, uno contente u e l'altro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



116

Proprietà del taglio

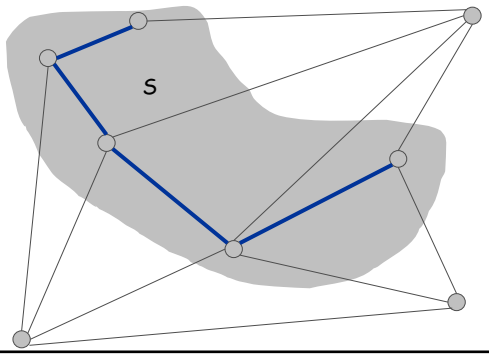
- Poichè (x,y) ≠ (u,v) e poichè entrambi attraversano il taglio e (u,v) è l'arco di peso minimo tra quelli che attraversano il taglio allora si ha $c_e < c_f$
- f=(x,y) si trova sull'unico percorso che connette u a v in T
- Se togliamo f=(x,y) da T dividiamo T in due alberi, uno contente u e l'altro contenente v
- Se introduciamo e=(u,v) riconnettiamo i due alberi ottenendo un nuovo albero ricoprente T' = T-{f}∪{e} dove tutte le coppie di nodi che prima erano connessee 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).



117

Algoritmo di Prim

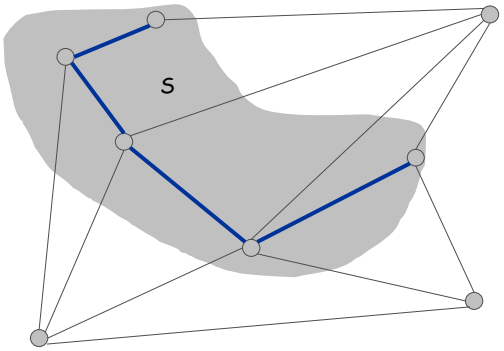
- Algoritmo di Prim. [Jarník 1930, Prim 1957, Dijkstra 1959,]
- Ad ogni passo T è un sottoinsieme di archi dello MST
- S= insieme di nodi di T
- Inizializzazione: Pone in S un qualsiasi nodo u. Il nodo u sarà la radice dello MST
- Ad ogni passo aggiunge a T l'arco (x,y) di costo minimo tra tutti quelli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



118

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

119

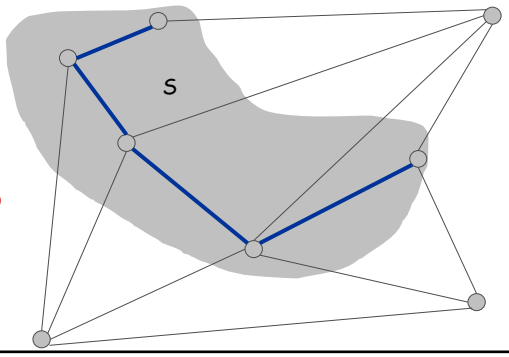
119

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



120

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 Prim con coda a priorità:
- $O(n^2)$ con array o lista non ordinati;
- $O(m \log n + n \log n)$ con heap. Siccome nel problema dello MST il grafo è connesso allora $m \geq n-1$ e $O(m \log n + n \log n) = O(m \log n)$

```
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
```

121