Algoritmi greedy
V parte

Progettazione di Algoritmi a.a. 2023-24
Matricole congrue a 1
Docente: Annalisa De Bonis

90

90

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. 2023-24
A. De Bonis

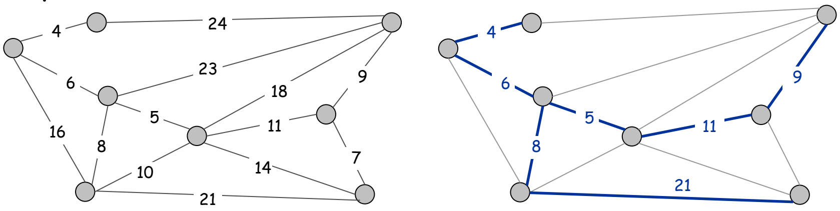
91

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 $\sum_{e \in T} c_e$

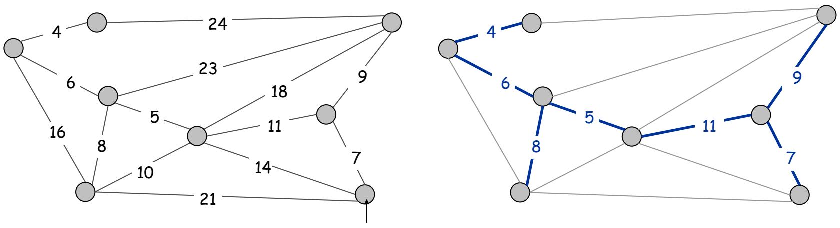
.



$G = (V, E)$ $\quad\quad$ T, $\Sigma_{e\in T}\, c_e = 64$

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.



$G = (V, E)$ $\quad\quad$ T, $\Sigma_{e\in T}\, c_e = 50$

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

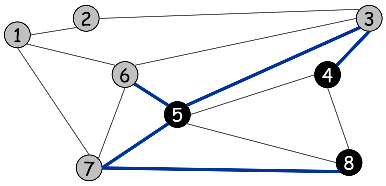
- Kruskal. Comincia con $T = \phi$. Considera gli archi in ordine non decrescente di costo. Inserisce un arco e in T se e solo il suo inserimento non determina la creazione di un ciclo in T

- 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.



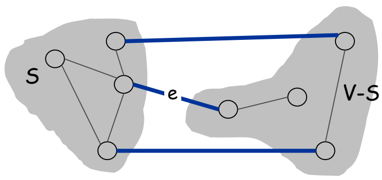
Taglio [S,V-S] = ({ 4, 5, 8 }, { 1,2,3,6,7 })
Archi che attraversano [S,V-S] D = 5-6, 5-7, 3-4, 3-5, 7-8

96

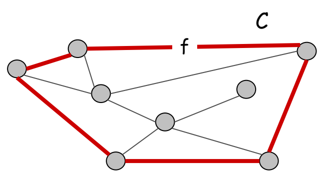
96

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 minimo 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

97

97

Proprieta` del taglio

Teorema (proprieta` del taglio):

Sia G non direzionato, connesso, pesato.

1. Sia A un sottoinsieme di archi di un MST del grafo G,
2. Sia [S,V-S] un taglio del grafo tale che nessun arco di A attraversa [S,V-S]
3. Sia (u,v) un arco di costo minimo che attraversa il taglio.

E` possible aggiungere (u,v) ad A in modo che A continui ad essere un sottoinsieme di archi di un MST.

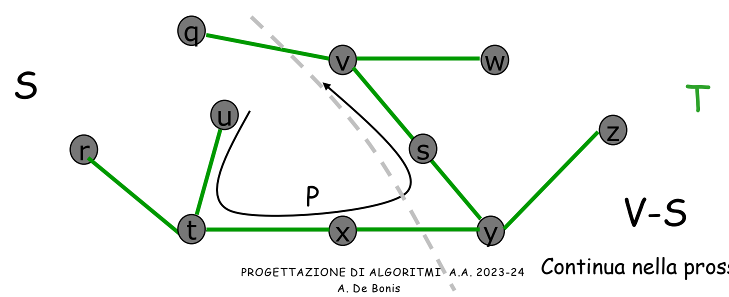
Dimostrazione:

- Dobbiamo far vedere che esiste un MST che contiene sia gli archi di A che l'arco (u,v).
- Per l'ipotesi 2, esiste un MST T che contiene tutti gli archi di A.

98

Proprietà del taglio

- Se e=(u,v)∈ T allora la dimostrazione 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 tutti gli archi di A∪{(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.

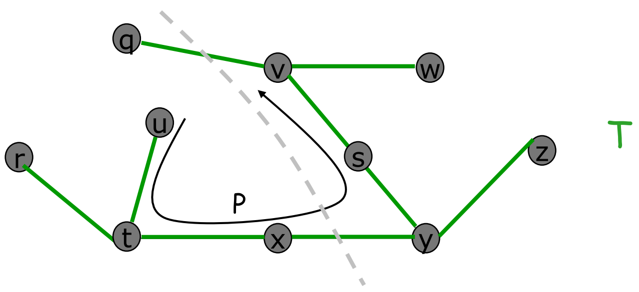


Continua nella prossima slide

99

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]

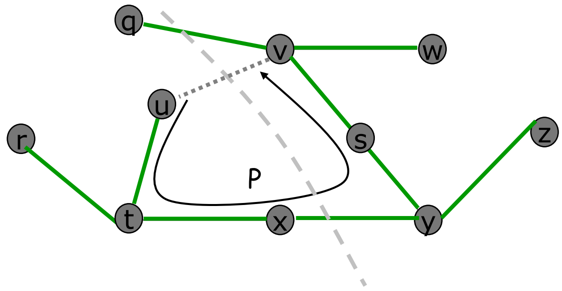


Continua nella prossima slide

100

Proprietà del taglio

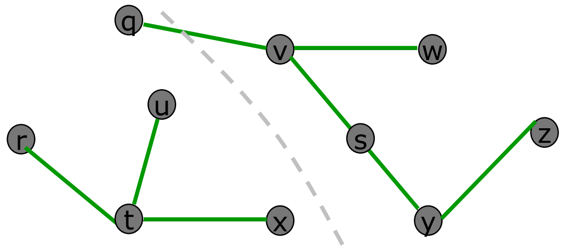
- Poichè (x,y) ≠ (u,v) e poichè entrambi attraversano il taglio e (u,v) ha peso minimo tra quelli che attraversano il taglio allora si ha $c_e \leq c_f$



101

Proprietà del taglio

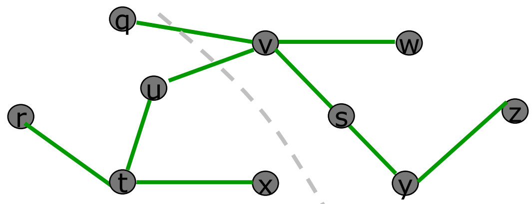
- Poichè (x,y) ≠ (u,v) e poichè entrambi attraversano il taglio e (u,v) è l'arco di peso minimo tra quelli che attraversano il taglio allora si ha $c_e \leq c_f$
- f=(x,y) si trova sull'unico percorso P che connette u a v in T
- Se togliamo f=(x,y) da T dividiamo T in due alberi, uno contente u e l'altro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



102

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).
- Si noti infine che l'arco (x,y) eliminato da T non e` in A perche' A non contiene archi che attraversano [S,V-S] → T' contiene tutti gli archi di A.



103

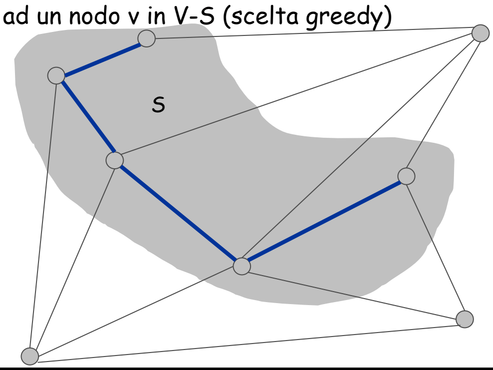
Osservazione per il caso in cui c'è un unico arco di costo minimo che attraversa il taglio

- 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.

104

Algoritmo di Prim

- Algoritmo di Prim. [Jarník 1930, Prim 1957, Dijkstra 1959,]
- L'algoritmo mantiene un sottoinsieme di archi A tale che
- A e` un albero che contiene un sottoinsieme degli archi dello MST **T** che sara` restituito alla fine dall'algoritmo.
- S= insieme di nodi di A
- **Descrizione dell'algoritmo:**
  - Inizializzazione: Pone in S un qualsiasi nodo s. Il nodo s sarà la radice di **T**
  - Ad ogni passo aggiunge a A un arco (u,v) di costo minimo tra tutti quelli che congiungono un nodo u in S ad un nodo v in V-S (scelta greedy)
  - Termina quando S=V



105

Correttezza dell'algoritmo di Prim

- Siano A ed S gli insiemi nella descrizione dell'algoritmo di Prim della slide precedente.
- Dimostriamo che ad ogni passo, l'insieme A mantenuto dall'algoritmo di Prim e` un sottoinsieme degli archi di un MST.
- All'inizio A e` vuoto per cui A e` contenuto in un qualsiasi MST.
- Supponiamo che immediatamente prima dell'esecuzione di un certo passo, A sia sottoinsieme di un MST e mostriamo che continua ad essere un sottoinsieme di un MST anche dopo.
- Osserviamo che in ogni passo e quindi anche in quello che stiamo considerando
  1. Nessun arco di A attraversa [S,V-S]. Cio` e` ovvio in quanto gli archi di A incidono solo su nodi di S.
  2. L'algoritmo aggiunge ad A, un arco (u,v) di costo minimo che attraversa il taglio [S,V-S].
- I punti 1 e 2 e il fatto che fino al passo considerato A e` un sottoinsieme di un MST implicano che l'insieme A, il taglio[S,V-S] e l'arco (u.v) soddisfano le ipotesi del teorema della proprieta` del taglio → dopo aver inserito l'arco (u,v) in A, A continua ad essere un sottoinsieme di un MST→ dopo il passo considerato A e` ancora un sottoinsieme di un MST..

continua nella prossima slide 106

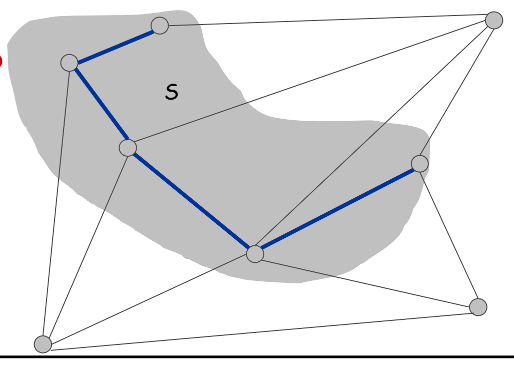
106

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

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



107

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 di un arco (u,v) di costo più basso tra quelli che uniscono v ad un nodo in S
  - pred[v] = u, dove (u,v) e` l'arco al punto precedente
- 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;
- $O(m \log n+n\log n)$ 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)
A<-- Ø,  S <-- Ø
For each u in V
        insert(Q,∞,u)
        pred(u) <-- nil
While (Q is not empty)
        (a[u],u)<-- ExtractMin(Q)
         S<-- SU{u} , A<-- S U{(pred(u),u)}
         For each edge e=(u,v)
                If  ((v not in S) && (ce< a[v]))
                        ChangeKey(Q,v, ce)
                        pred(v) <-- u
T <--A
Return T
```

108

108