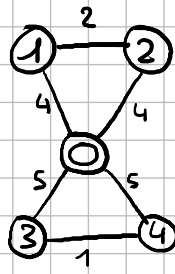
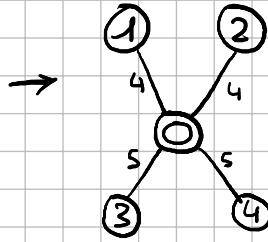


## Minimo Albero Ricoprente

Prendiamo per esempio il grafo



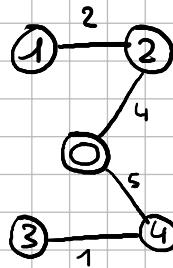
Creiamo l'albero dei cammini minimi partendo dalla sorgente 0



Peso totale: 18

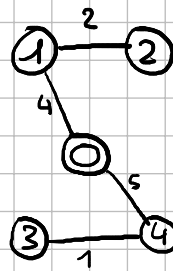
Ora creiamo il Minimo Albero

Ricoprente (MST, MINIMUM SPANNING TREE)



Peso totale: 12

oppure



anche in questo  
caso peso: 12

## Definizione:

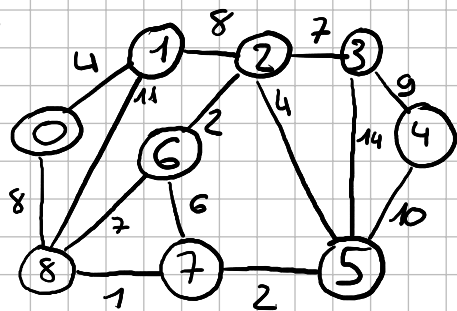
Un Minimo Albero Ricoprente è un Albero (quindi aciclico), Ricoprente (quindi da una qualsiasi sorgente posso raggiungere un qualsiasi altro nodo), connesso e dove gli archi hanno il minimo peso possibile. È valido solo per grafi non orientati!

## Teorema:

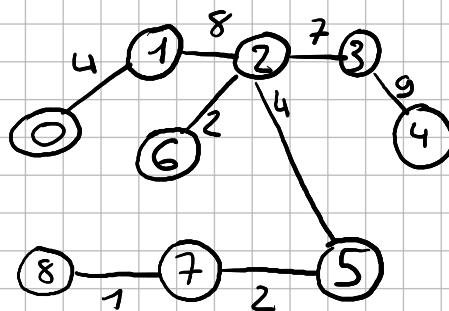
Se i pesi degli Archi di un grafo sono tutti distinti:

l' MST è unico!

$E_s$



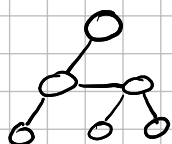
Mst



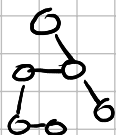
## Dimostrazione del Teorema (per assurdo)

Ho 2 archi ricorrenti:

MST1  
( $V_1, E_1$ )



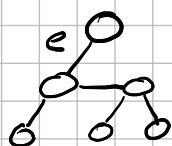
MST2  
( $V_2, E_2$ )



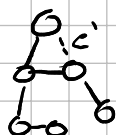
$$D = (E_1 - E_2) \cup (E_2 - E_1)$$

è arco di peso minimo di D

MST1'  
( $V_1, E_1$ )



MST2'  
( $V_2, E_2$ )



$$e \in D$$

$$\text{peso}(e') > \text{peso}(e)$$

$$\text{peso MST2}' = \text{peso}(MST2) - \text{peso}(e') + \text{peso}(e) < \text{peso}(MST2)$$

↑  
impossibile

## Come trovare l'MST?

Per prima cosa dobbiamo capire che cos'è un taglio ed

imparare il suo teorema.

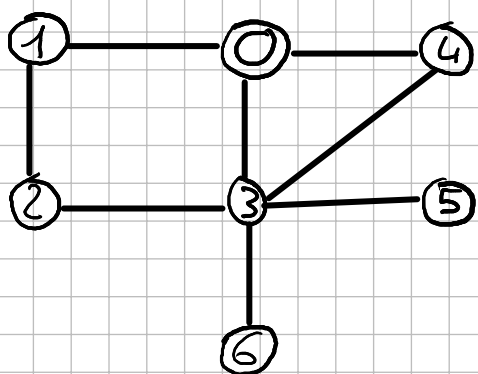
## Taglio

Dato  $G = (V, E)$  con  $\text{pesi} \in \mathbb{N}$  abbiamo

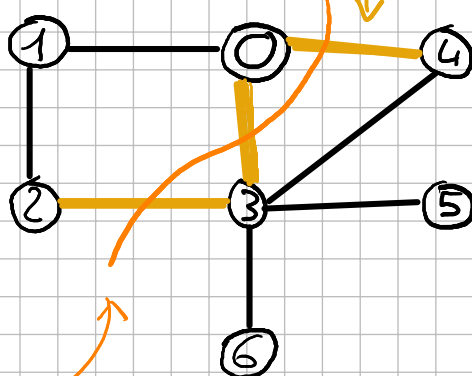
$X \subset V$  con  $X \neq V$  ed  $X \neq \emptyset$

non può  
contenere tutti i nodi

Es:



Eseguiamo un  
taglio



In questo caso abbiamo  $X = \{1, 4, 2\}$

$$V \setminus X = \{3, 4, 5, 6\}$$

$V$  meno  $X$

$(u, v)$  attraversa il taglio se  $u \in X$  e  $v \in V \setminus X$

Petto semplicemente, il taglio divide un grafo in due parti.

## Teorema del taglio

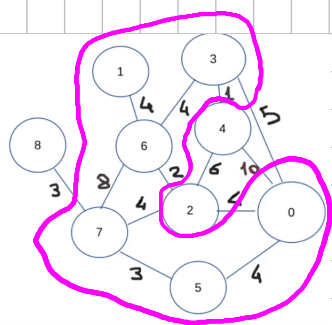
L'arco di peso minimo che attraversa un qualunque taglio fa parte dell'MST

## Dimostrazione (per assurdo)

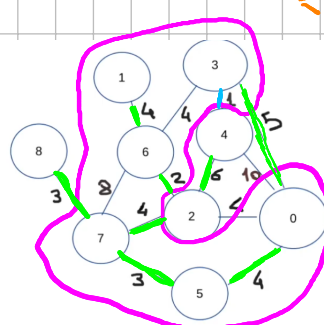
- Ipotesi:
- 1)  $G = (V, E)$
  - 2) Taglio di  $G$   $(X, V \setminus X)$  con  $e = (u, v)$  con  $u \in X$   $v \in V \setminus X$  di peso minimo
  - 3)  $T = (V, E')$  è un MST di  $G$

Tesi:  $e \in E'$

Prendiamo il grafo ed applichiamo il taglio



ora aggiungiamo  
 $\Rightarrow$  l'MST  
(archi verdi)



Da notare  
Come l'arco  
con peso

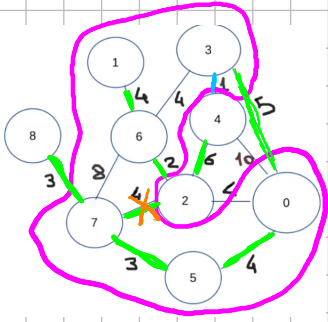
minimo sia stato  
ignorato, ovvero l'arco  
(3,4) con peso 1

Di fatto sto negando  $\rightarrow$   
la tesi, requisito fondamentale per  
la dimostrazione per assurdo

Cosa succedrebbe se aggiungessimo  $e$  all'MST?

Creiamo un ciclo!

Risolviamo questo problema togliendo un arco che attraversa  
il taglio e che ci rimuove il ciclo.





→ Togliamo per esempio l'arco  $(2,7)$ , in questo modo l'MST è di nuovo aciclico.  
 Ricorda che ora l'arco  $(3,4)$  fa parte del graf.  
 Chiamiamo  $e'$  l'arco che abbiamo tolto.

Possiamo sicuramente dire che  $\text{peso}(e) > \text{peso}(e')$  perché per definizione l'arco  $e$  è quello di peso minimo (Punto 2 dell'ipotesi)

Ma in questo modo abbiamo trovato un nuovo MST, cosa **impossibile**, quindi abbiamo confermato la tesi.

Un esempio di taglio è la frontiera negli algoritmi di visita, con

$X$  = nodi scoperti       $V \setminus X$  = nodi da scoprire.

Algoritmo di Prim - Jarník  

È un algoritmo di visita in grado di determinare un MST

Pseudo Codice (utile per la dimostrazione)

Algoritmo Prim (sorg) {

$S = \{\text{sorg}\}$

$A = \emptyset$  ← qui ci sarà l'MST

finché possibile  $(u,v)$  arco di peso minimo che attraversa la frontiera

$S = S \cup \{v\}$

$A = A \cup \{(u,v)\}$

Dobbiamo dimostrare che sia effettivamente un MST.

Non dobbiamo dimostrare che questo algoritmo ritorni un albero ricoprente perché già dimostrato dalle altre visite

Bisogna dimostrare che sia minimo

$Peso(T) = \sum_{e \in A} peso(e)$ , e questo  $\hat{=}$   $peso(MST)$ , perché l'algo  
Sceglie sempre l'arco di peso minimo.

Un arco

La sorg è indifferente, l'algoritmo ritorna sempre un MST

### Pseudo Codice completo

Algoritmo Prim(sorg) {

$S = \{sorg\}; A = \emptyset; peso = 0;$

$\forall (sorg, v) \in E \text{ minHeap.add}(\text{archi}(sorg, v), \text{peso}(sorg, v))$

while minHeap non vuoto {

$(u, v) \leftarrow \text{minHeap}$

se  $v \notin S$

$S = S \cup \{v\}$

$A = A \cup \{(u, v)\}$

$\forall (v, z) \in E \text{ minHeap.add}((v, z), \text{peso}(v, z))$

$peso = peso + \text{peso}(v, z)$

Come puoi notare è molto simile a Dijkstra, infatti:

anche il costo rimane il medesimo.

$O(m + \log m)$ , e se è denso ( $m \sim n^2$ ),  $O(m \log n)$ .

## Algoritmo di Kruskal

Questo algoritmo determina un MST.

Anche lui sfrutta il teorema del taglio ma **non** è un algoritmo di visita.

Kruskal {

finché possibile

$e = \text{arco di peso min}$

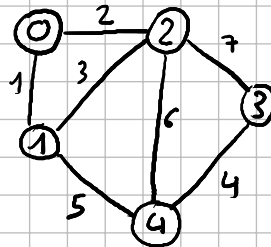
se  $A \cup \{e\}$  ciclo

scarta  $e$

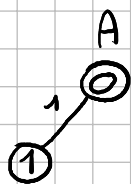
altrimenti

$A = A \cup \{e\}$

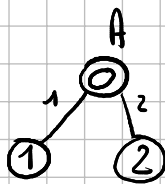
→ prima sono  
stat: caricati  
tutti gli archi



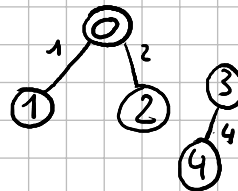
Passo 1



Passo 2

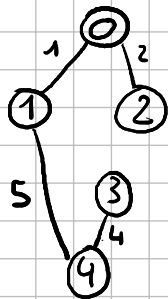


...



Da notare  
come l'arco  
di peso 3 sia  
stato scartato,  
perché avrebbe  
creato un ciclo.

... E dopo altri passi ecco l'MST



→ È normale  
che momentaneamente  
l'intero non sia  
connesso.

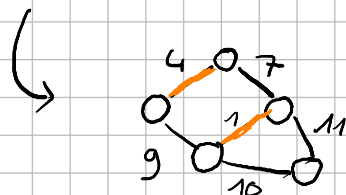
Dobbiamo controllare se l'algoritmo di Kruskal crea effettivamente un MST.

L'albero che crea:

è Ricoprente? Sì, perché l'algoritmo prende tutti i nodi

è Aciclico? Sì, perché l'algoritmo controlla questa condizione

è Connesso? Temporaneamente l'albero può risultare sconnesso, ma il grafo di partenza era connesso.



per creare un grafo non connesso dovrei scartare 7, ma sarebbe un assurdo, perché non c'è nessuna condizione che vieti questa cosa.

è Minimo? Sì, grazie al teorema del taglio, infatti ogni arco che viene aggiunto nell'albero è un arco di peso minimo che attraversa un taglio.

Quindi possiamo dire che Kruskal crea effettivamente un MST.

Pseudocodice + problemi di costo

Kruskal {

$\forall e$  minheap.add

$A = \emptyset$

while minheap non vuoto

$e = \text{minheap.remove}$

if  $(V, A \cup \{e\})$  ha cicli:

scarto  $e$

else

$A = A \cup \{e\}$

→ inserimento dei archi nell'heap + gestione heap

$O(m \cdot \log n)$

$O(\log n) \rightarrow$  gestione heap

$O(m+n)$

→ Come controllo se ha cicli? Con una visita

Costo totale:  $O(m \cdot (\log n + m \cdot m))$

↖ Algoritmo molto costoso, soprattutto per  
il controllo di Cicli; è ottimizzabile  
usando le Union Rank.