



# Algoritmi e Strutture Dati

## Minimo albero ricoprente

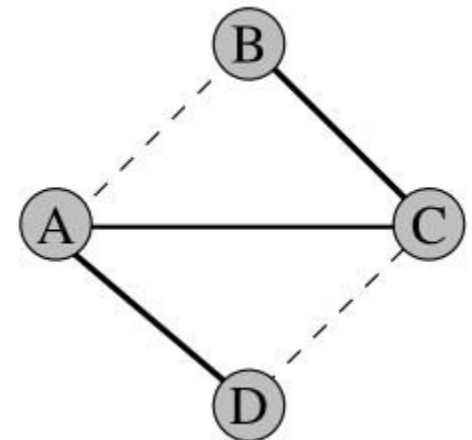
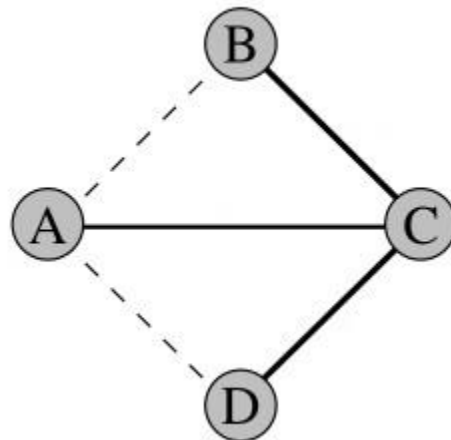
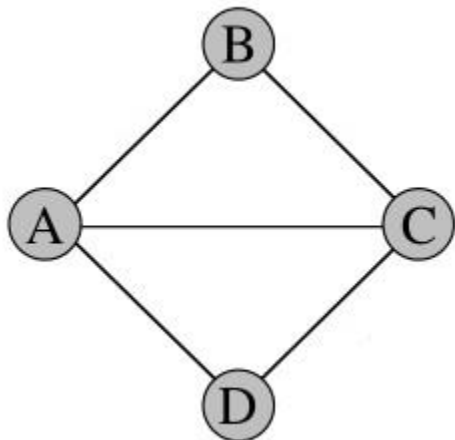
Domenico Fabio Savo

# Albero ricoprente

Sia  $G=(V,E)$  un grafo *non orientato e connesso*. Un **albero ricoprente** di  $G$  è un *sottografo*  $T \subseteq G$  tale che:

- **$T$  è un albero;**
- **$T$  contiene tutti i vertici di  $G$ .**

Un grafo può avere più alberi ricoprenti.



# Minimo albero ricoprente

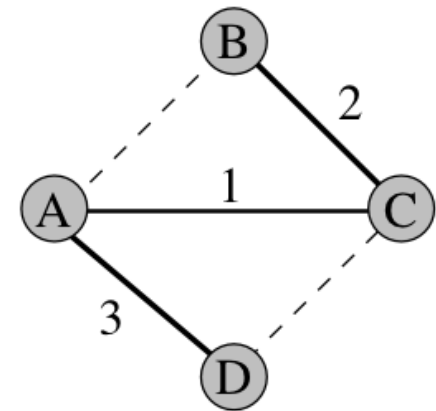
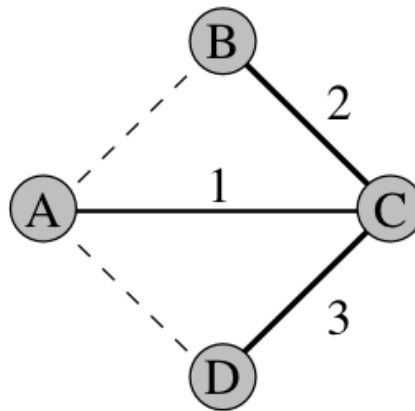
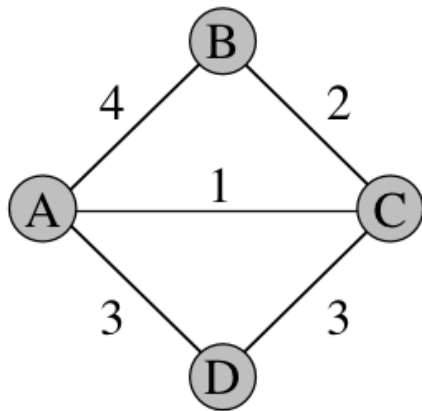
- Ad ogni arco  $e \in E$  associamo un **costo** (o **peso**)  $w(e)$ .
- Definiamo come **costo di un albero ricoprente T** di un grafo  $G$  come la somma dei costi dei suoi archi:

$$w(T) = \sum_{e \in E} w(e)$$

**Definizione:** (minimo albero ricoprente) Sia  $G=(V,E)$  un grafo *non orientato, connesso e pesato sugli archi*. Un **minimo albero ricoprente** di  $G$  è un albero ricoprente di  $G$  con costo *minimo*.

# Esempi

Il minimo albero ricoprente non è necessariamente unico





# Proprietà dei minimi alberi ricoprenti

# Il problema

**Il problema:** Dato un grafo  $G = (V, E)$  *non orientato*, *connesso* e *pesato sugli archi* trovare un **minimo albero ricoprente** di  $G$

Problema di *ottimizzazione* molto studiato che si presenta in molti contesi applicativi (ES: *noleggio connessioni di rete*).

# La tecnica algoritmica *Golosa*

- Per risolvere il problema del **minimo albero ricoprente** adotteremo la tecnica algoritmica *golosa* (*greedy*).
- Questo tipo di tecnica è tipicamente utilizzata per risolvere problemi di ottimizzazione.
- La tecnica golosa prevede la costruzione di una soluzione effettuando ad ogni passo la scelta che, allo stato attuale delle cose, sembra la più promettente, senza preoccuparsi per il futuro (per questo la tecnica è detta golosa).
- Questa tecnica non prevede la possibilità di avere ripensamenti.

# Un approccio “goloso” al problema

- Costruiremo un **minimo albero ricoprente** un arco alla volta, effettuando scelte localmente “golose”. Ad esempio:
  - **includere** nella soluzione archi di **costo piccolo**
  - **escludere** dalla soluzione archi di **costo elevato**
- Formalizzeremo il processo come un processo di colorazione:
  - archi **blu**: inclusi nella soluzione
  - archi **rossi**: esclusi dalla soluzione



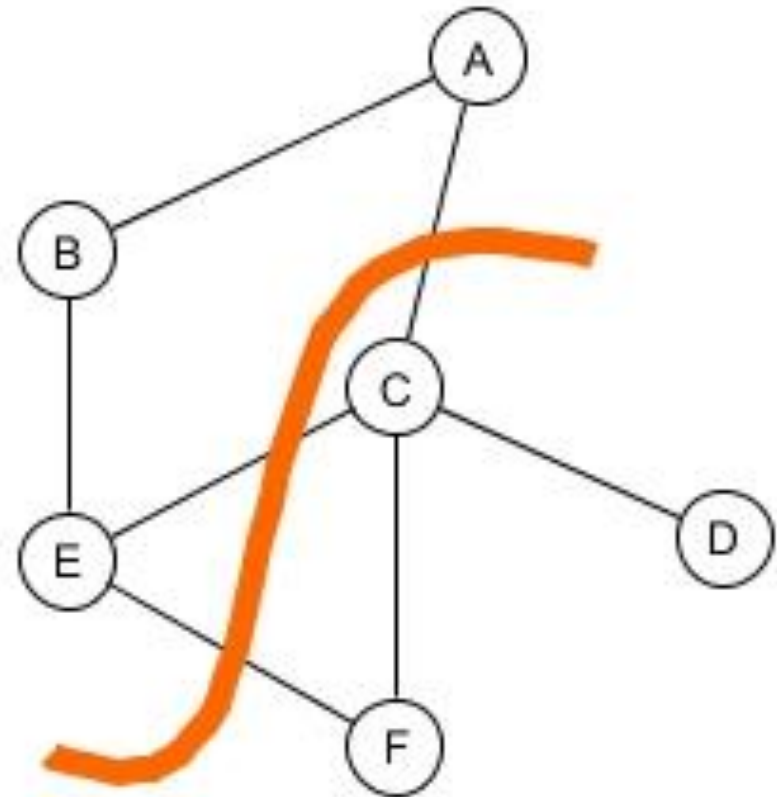
# Tagli

**Definizione** (Taglio): Dato un grafo non orientato  $G=(V,E)$ , un **taglio** in  $G$  è una partizione dei vertici  $V$  in due insiemi:  $X$  e  $\overline{X} = V-X$ . Diciamo che un arco  $e = (u,v)$  *attraversa* il taglio  $(X,\overline{X})$  se  $u \in X$  e  $v \in \overline{X}$

Con un piccolo abuso di notazione indicheremo un **taglio** come *l'insieme di archi che lo attraversano*

# Esempio: taglio

Taglio  $\{(A,C), (E,C), (E,F)\}$



# Regola del taglio (regola blu)

Scegli un taglio che non contiene archi blu. Tra tutti gli archi non colorati del taglio, scegline uno di costo minimo e coloralo blu.

- Ogni albero ricoprente deve infatti contenere almeno un arco del taglio
- E' naturale includere quello di costo minimo

# Regola del ciclo (regola rossa)

Scegli un ciclo che non contiene archi rossi. Tra tutti gli archi non colorati del ciclo, scegline uno di costo massimo e coloralo rosso.

- Ogni albero ricoprente deve infatti escludere almeno un arco del ciclo
- E' naturale escludere quello di costo massimo

# L'approccio “goloso”

- L'approccio goloso applica una delle due regole ad ogni passo, finché tutti gli archi sono colorati
- Si può dimostrare che esiste sempre un minimo albero ricoprente che contiene tutti gli archi blu e non contiene nessun arco rosso.
- Si può inoltre dimostrare che il metodo goloso colora tutti gli archi.



# Tempi di esecuzione

A seconda della scelta della regola da applicare e del taglio/ciclo usato ad ogni passo, si ottengono dal metodo goloso diversi algoritmi con diversi tempi di esecuzione



# Algoritmo di Kruskal

# Strategia

- Mantiene una foresta di alberi formati da soli archi blu (alberi blu), all'inizio tutti disgiunti.
- Per ogni arco, in ordine non decrescente di costo, applica il seguente passo: "*se l'arco ha entrambi gli estremi nello stesso albero blu, applica la regola del ciclo e coloralo rosso, altrimenti applica la regola del taglio e coloralo blu*".



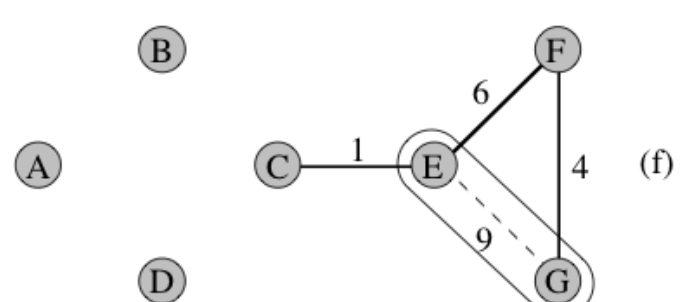
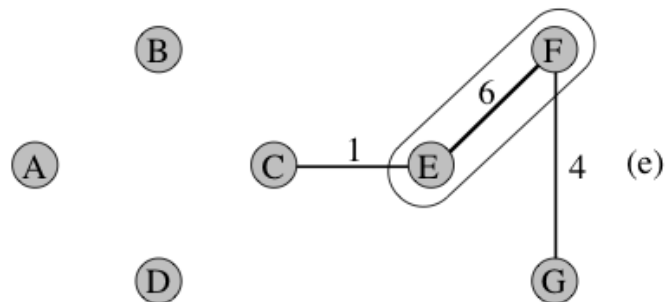
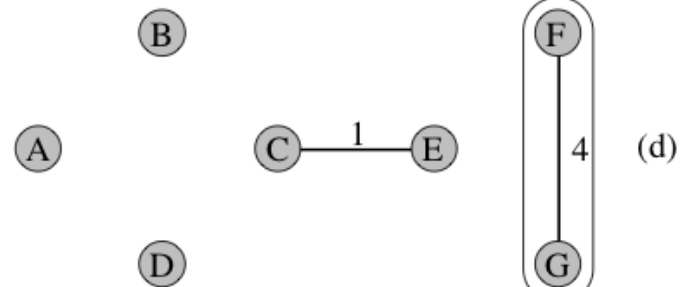
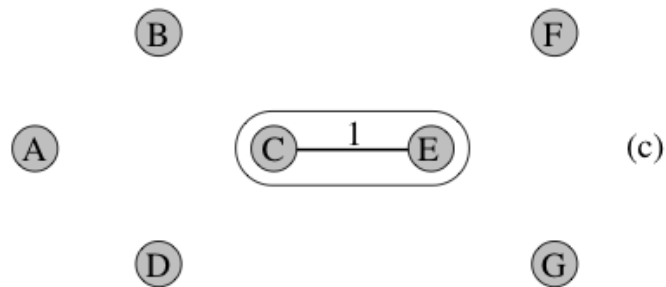
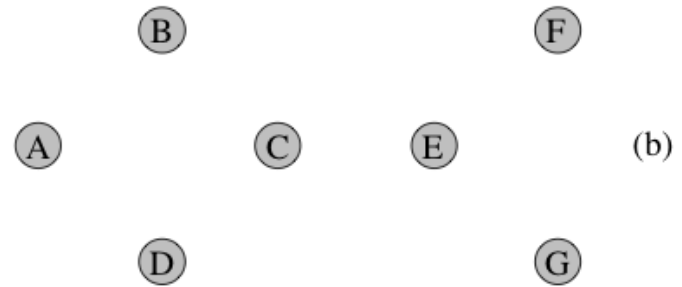
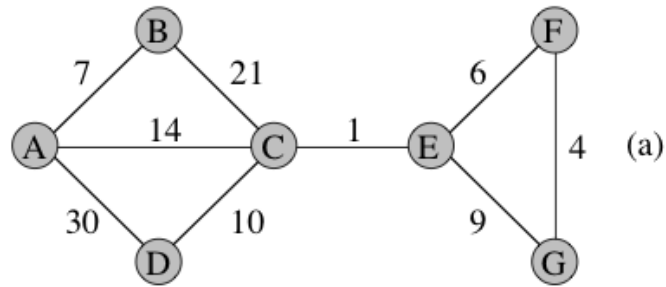
# Pseudocodice

**algoritmo** Kruskal(grafo  $G$ )  $\rightarrow$  albero

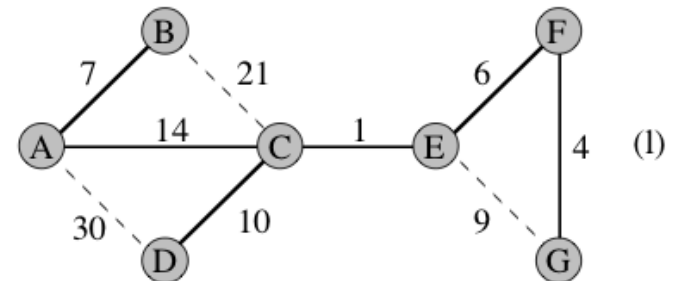
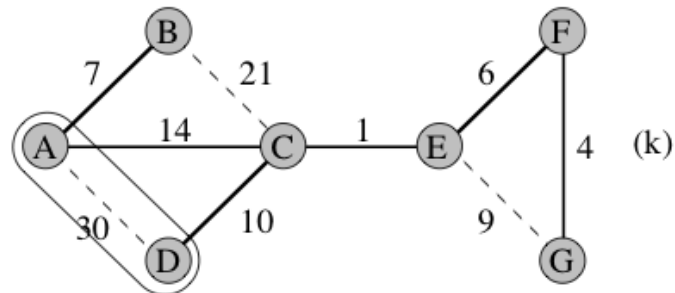
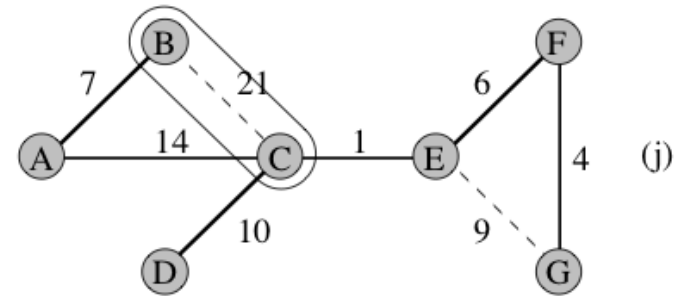
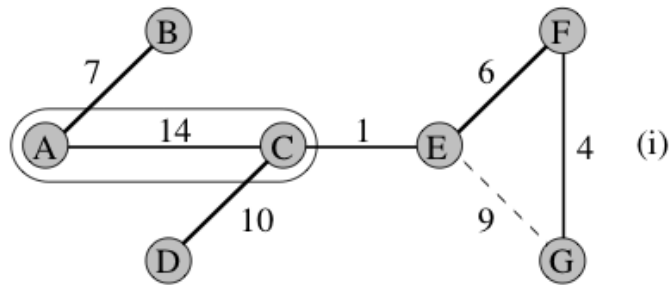
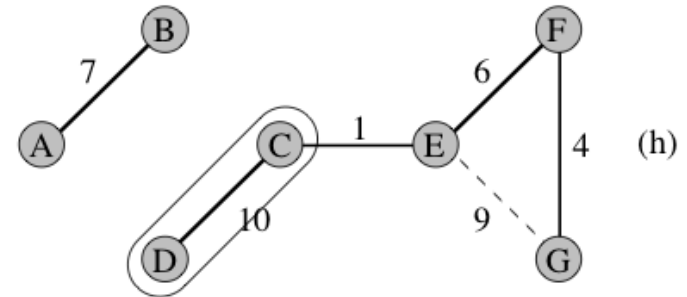
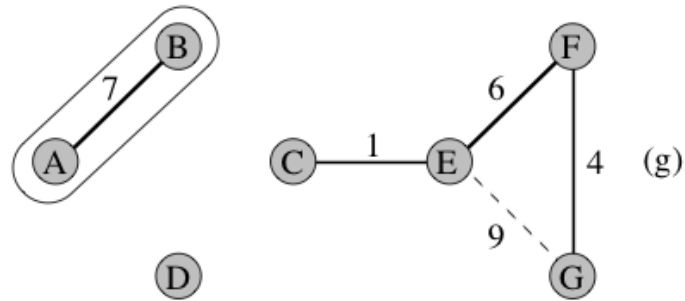
1. ordina gli archi di  $G$  secondo costi non decrescenti;
2.  $T \leftarrow$  albero vuoto;
3. **for each** (arco  $(x,y)$  di  $G$  in ordine non decrescente di costo) **do**
4.     **if**(  $x$  e  $y$  non sono connessi in  $T$ ) **then**
5.         aggiungi l'arco  $(x,y)$  a  $T$ ;
6. **return**  $T$



# Esempio (1/2)



# Esempio (2/2)



# Analisi

Il tempo di esecuzione dell'algoritmo di Kruskal è  $O(m \log n)$  nel caso peggiore

Dove  $n$  è il numero di nodi e  $m$  è il numero di archi



# Algoritmo di Prim

# Strategia

- Mantiene un unico albero blu  $T$ , che all'inizio consiste di un vertice arbitrario  $s$ .
- Applica per  $n-1$  volte il seguente passo:

*scegli un arco di costo minimo incidente su uno dei nodi di  $T$  e coloralo di **blu***

# Pseudocodice

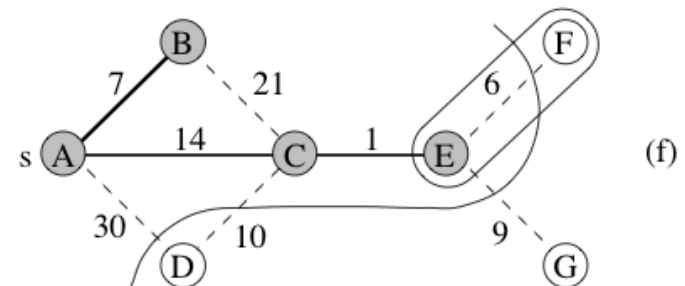
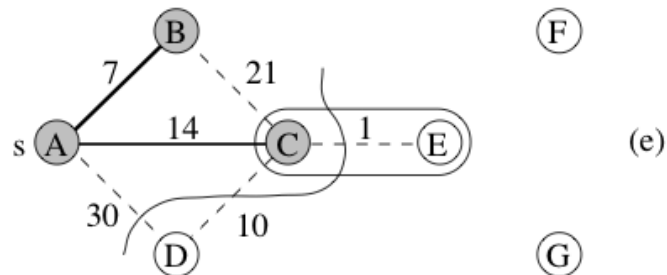
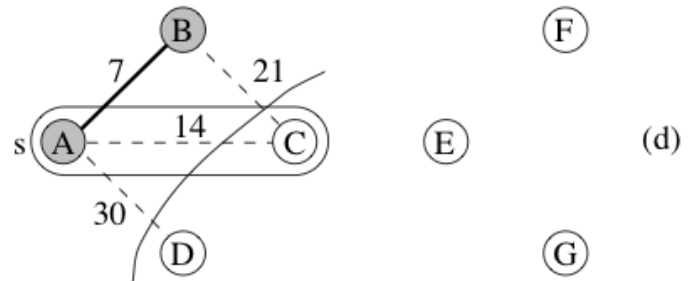
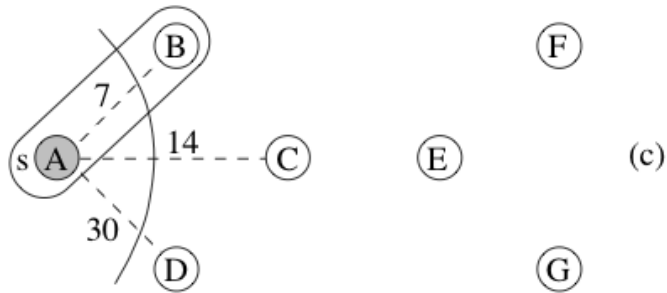
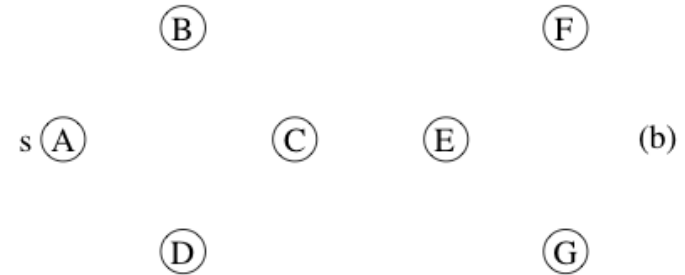
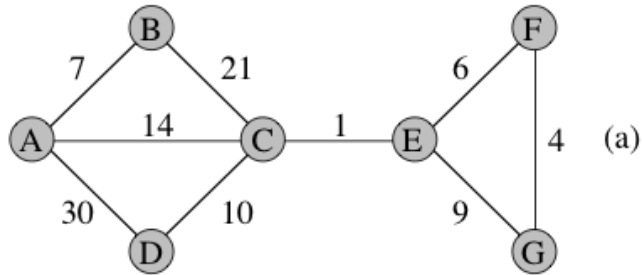
Definiamo **arco azzurro** un arco  $(x,y)$  tale che:

- $x \in T$
- $y \notin T$

**algoritmo** Prim(grafo  $G$ )  $\rightarrow$  albero

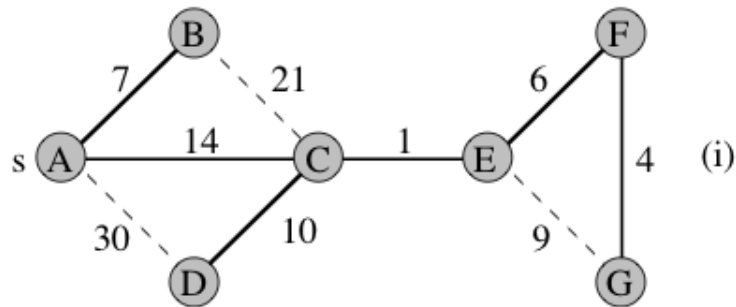
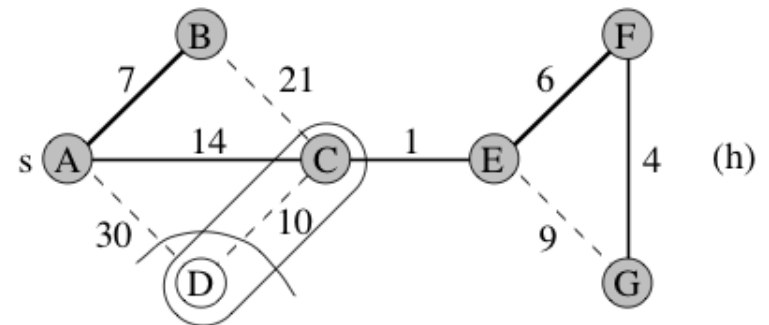
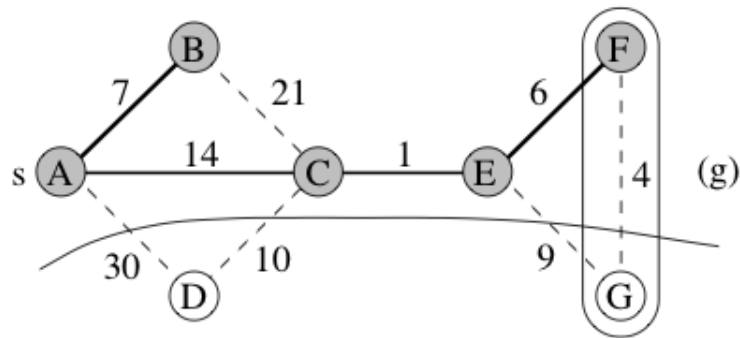
1. scegliere in modo arbitrario un nodo  $s$ ;
2.  $T \leftarrow$  albero formato dal solo nodo  $s$ ;
3. **while** (numero di nodi di  $T < n$ ) **do**
4.       trova **l'arco azzurro**  $(x,y)$  con costo minimo.
5.       aggiungi l'arco  $(x,y)$  a  $T$ ;
6. **return**  $T$

# Esempio (1/2)





# Esempio (2/2)





# Analisi

Il tempo di esecuzione dell'algoritmo di Prim è  $O(m + n \log n)$  nel caso peggiore



# Algoritmo di Borůvka

# Strategia

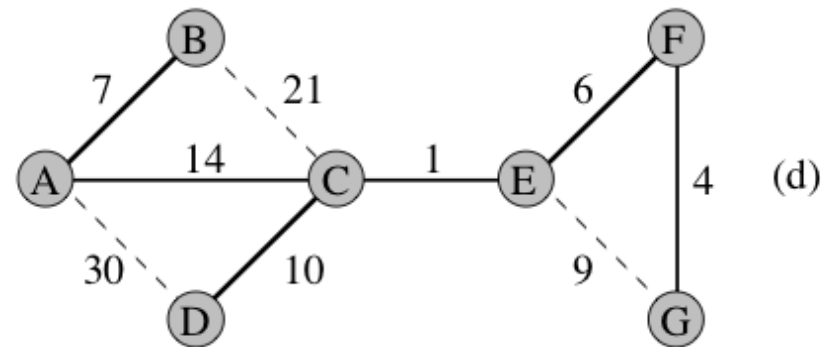
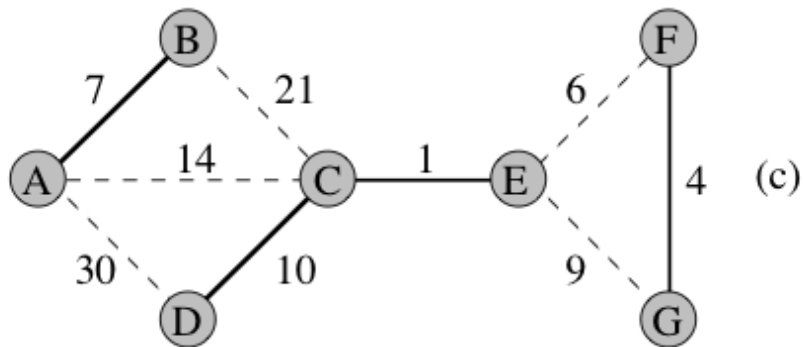
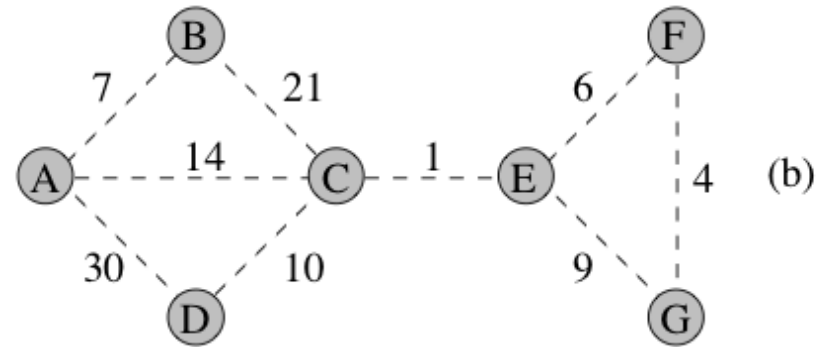
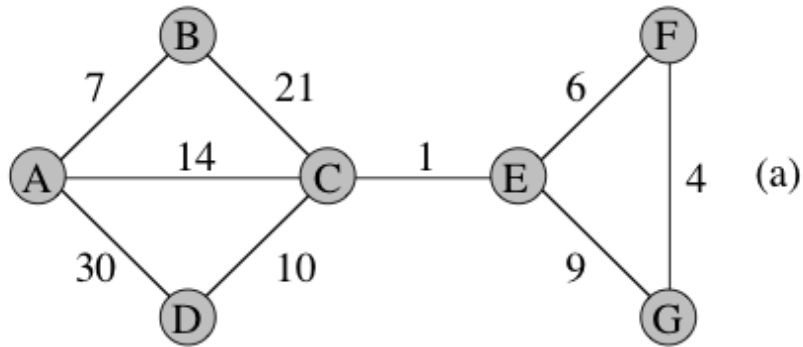
- Mantiene una foresta di alberi blu, all'inizio tutti disgiunti
- Ad ogni passo, applica la seguente regola:  
*"per ogni albero blu  $T$  nella foresta, scegli un arco di costo minimo incidente su  $T$  e coloralo blu (applica la regola del taglio)"*
- Per non rischiare di introdurre cicli, assumiamo che i costi degli archi siano tutti distinti

# Pseudocodice

**algoritmo** Boruvka(grafo  $G = (V,E)$ )  $\rightarrow$  albero

1.  $T \leftarrow$  albero vuoto;
3. **while** ( $T$  ha più di una componete connessa) **do**
4.     per ogni componente  $C$  di  $T$  scegli l'arco  $(u,v)$  di  $G$   
con costo minimo tale che  $u \in C$  e  $v \notin C$ ;
5.     aggiungi l'arco  $(u,v)$  a  $T$ ;
6. **return**  $T$

# Esempio





# Analisi

Il tempo di esecuzione dell'algoritmo di Boruvka è  $O(m \log n)$  nel caso peggiore



# Riepilogo

- Paradigma generale per il calcolo di minimi alberi ricoprenti
- Applicazione della tecnica golosa
- Tre algoritmi specifici ottenuti dal paradigma generale: Prim, Kruskal e Boruvka