Algoritmi e Strutture Dati

Minimo albero ricoprente

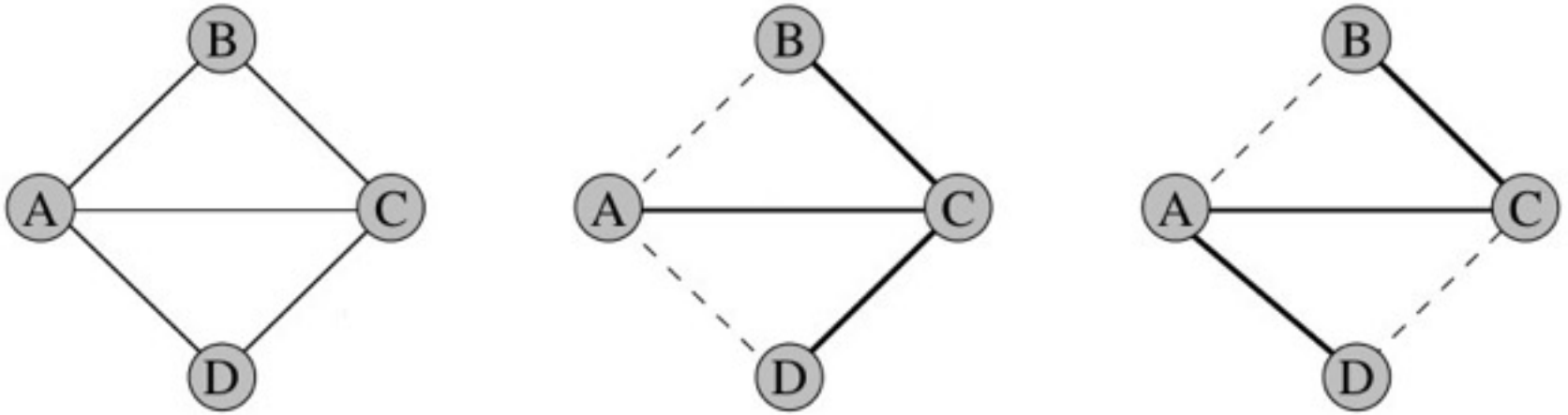
Fabio Patrizi

Albero ricoprente

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

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

Un grafo può avere più alberi ricoprenti.



Minimo albero ricoprente

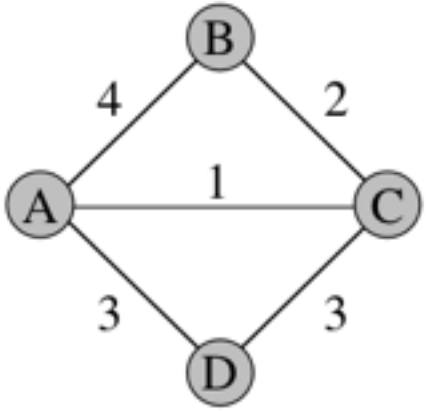
- Ad ogni arco **e∈E** associamo un **costo** (o **peso**) **w(e)**.
- Definiamo il **costo di un albero ricoprente T** di un grafo **G** come la somma dei costi dei suoi archi:

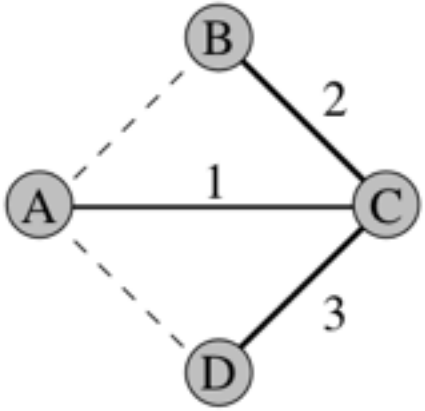
$$w(T) = \sum_{e \in E} w(e)$$

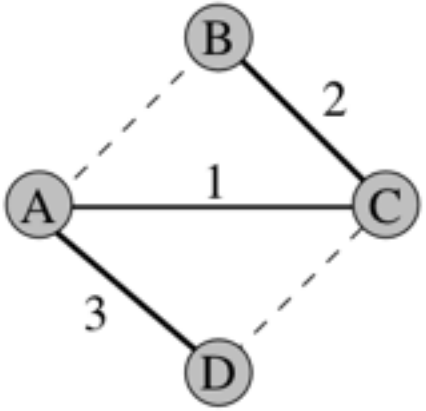
Definizione: (minimo albero ricoprente) Sia **G=(V,E)** un grafo *non orientato*, *connesso* e *pesato sugli archi*. Un **minimo albero ricoprente** di G è un albero ricoprente di G con costo *minimo*.

Esempi

Il minimo albero ricoprente non è necessariamente unico







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 diversi contesti applicativi (esempio: *noleggio connessioni di rete*)

La tecnica algoritmica *Greedy*

- Adottiamo la tecnica algoritmica ***greedy*** (*golosa*)
- Tecnica spesso utilizzata per problemi di ottimizzazione
- Prevede la costruzione di una soluzione effettuando ad ogni passo la scelta che sembra più promettente, senza preoccuparsi dell'impatto sulle scelte future
- Scelte già effettuate non possono essere cambiate (*no backtracking*)

Un approccio *Greedy*

- Costruiamo il **minimo albero ricoprente** un arco alla volta, effettuando scelte *localmente* vantaggiose:
  - includere nella soluzione archi di costo piccolo
  - escludere dalla soluzione archi di costo elevato

- Processo di *colorazione*:
  - archi blu: inclusi nella soluzione
  - archi rossi: esclusi dalla soluzione

Tagli

Taglio:

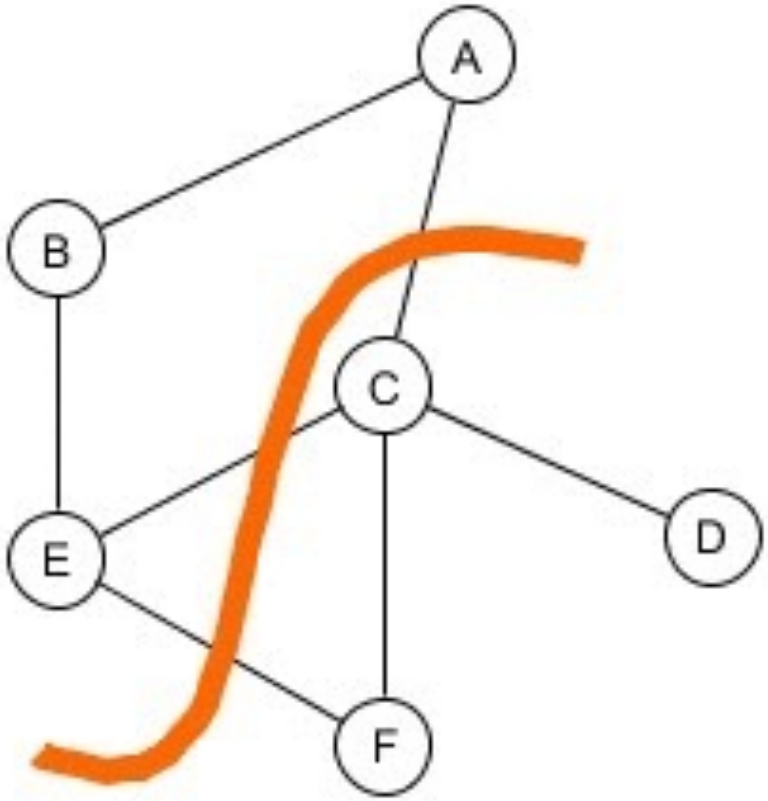
partizione dell'insieme di vertici **V** di **G** in due insiemi: **X** e **X' = V-X.**

Un arco **e = (u,v)** *attraversa* il taglio **(X,X')** se **u∈X** e **v∈X'**

Identifichiamo un **taglio** con *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 di blu

- Ogni albero ricoprente deve contenere almeno un arco del taglio
- E' conveniente 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 di rosso

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

L’approccio greedy

- L’approccio greedy applica una delle due regole ad ogni passo, finché tutti gli archi sono colorati
- Si può dimostrare che, una volta applicato l’algoritmo, esiste un minimo albero ricoprente contenente tutti gli archi blu e nessun arco rosso
- Si può inoltre dimostrare che il metodo greedy 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 diversi algoritmi con diversi costi di esecuzione

Algoritmo di Kruskal

Strategia

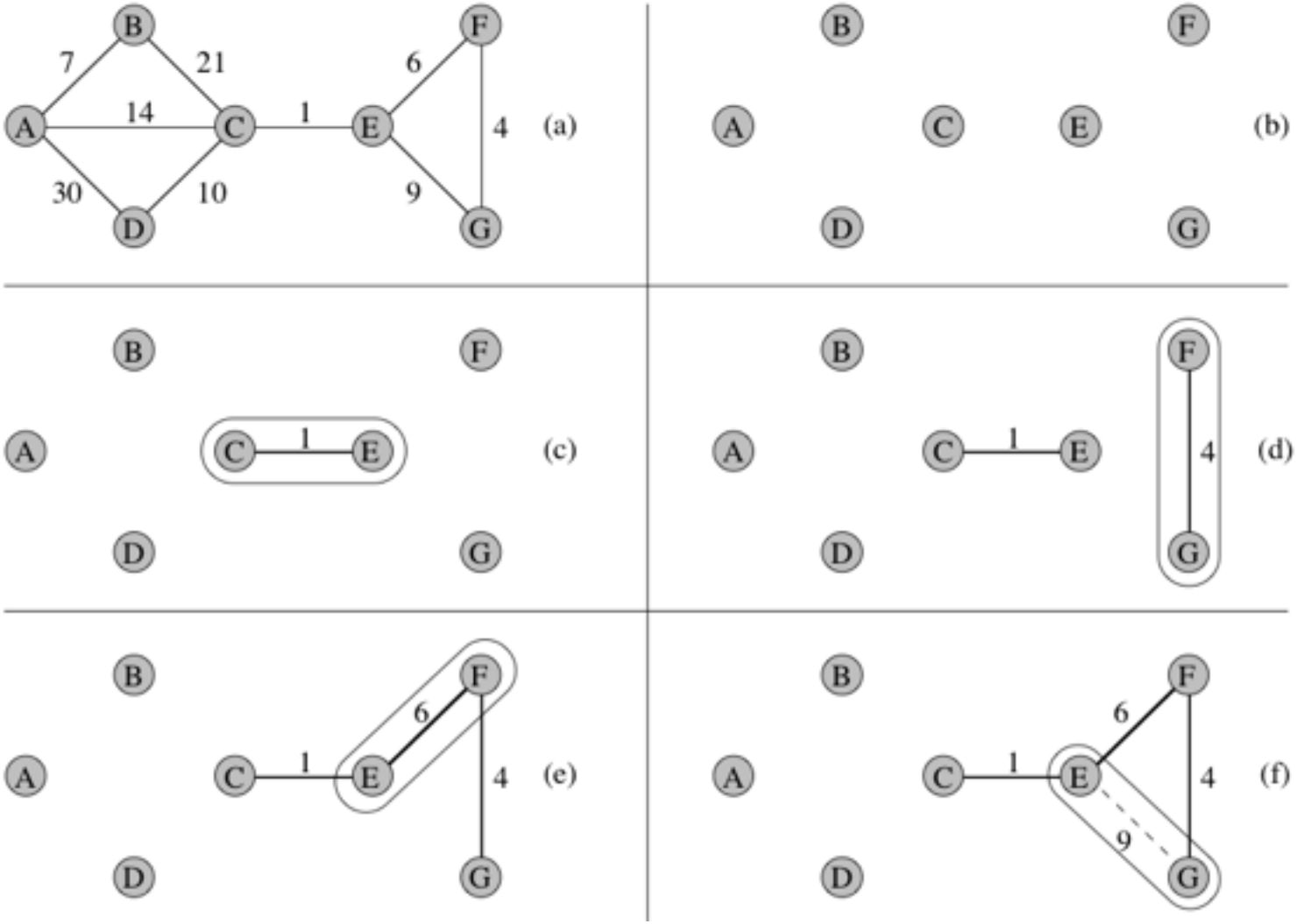
- Mantiene una foresta di alberi formati da soli archi blu (alberi blu), all'inizio tutti disgiunti.

- 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*

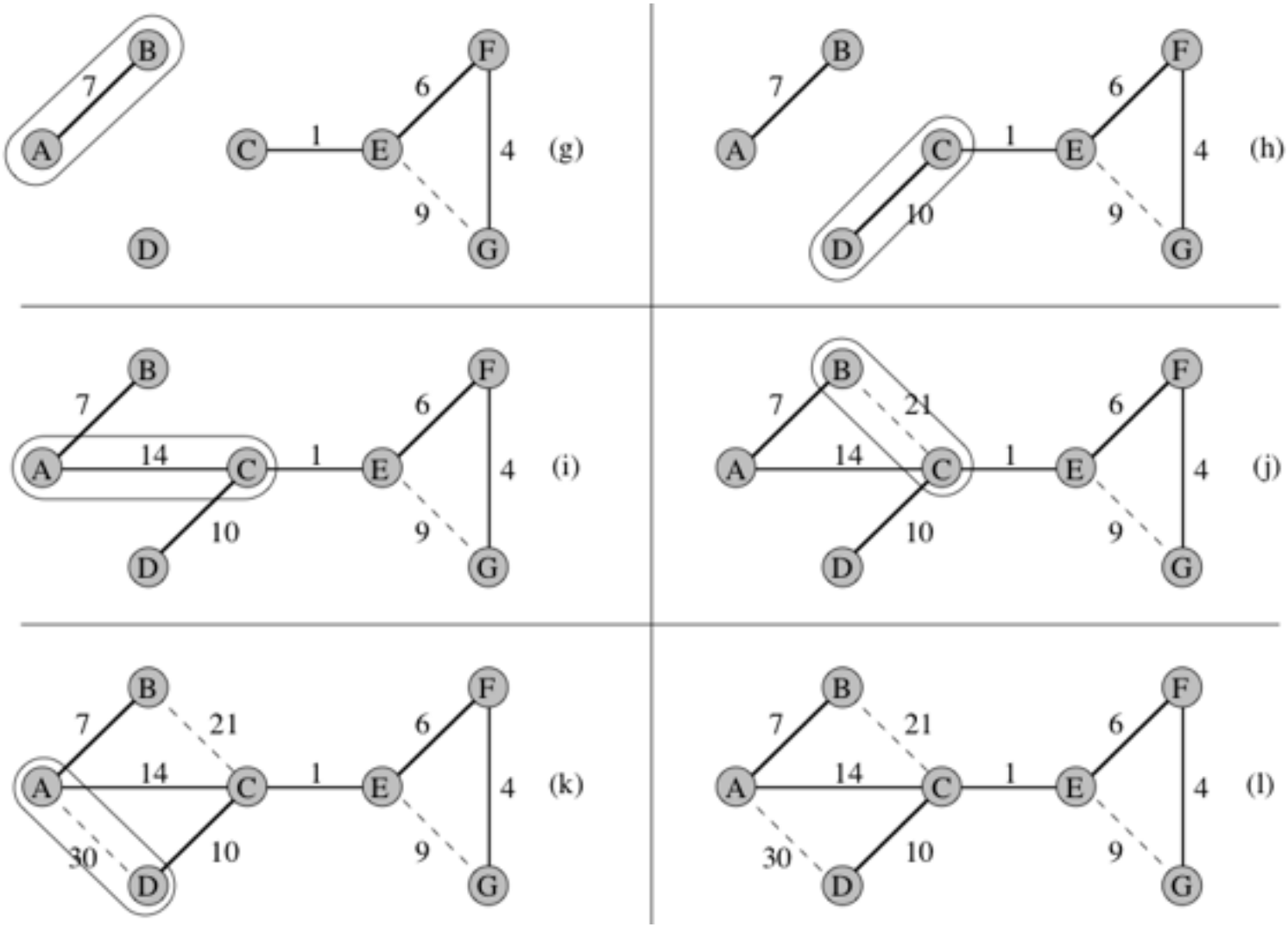
Pseudocodice

algoritmo Kruskal(grafo **G**) → albero
1. ordina gli archi di **G** secondo costi non decrescenti;
2. **T** ← 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:

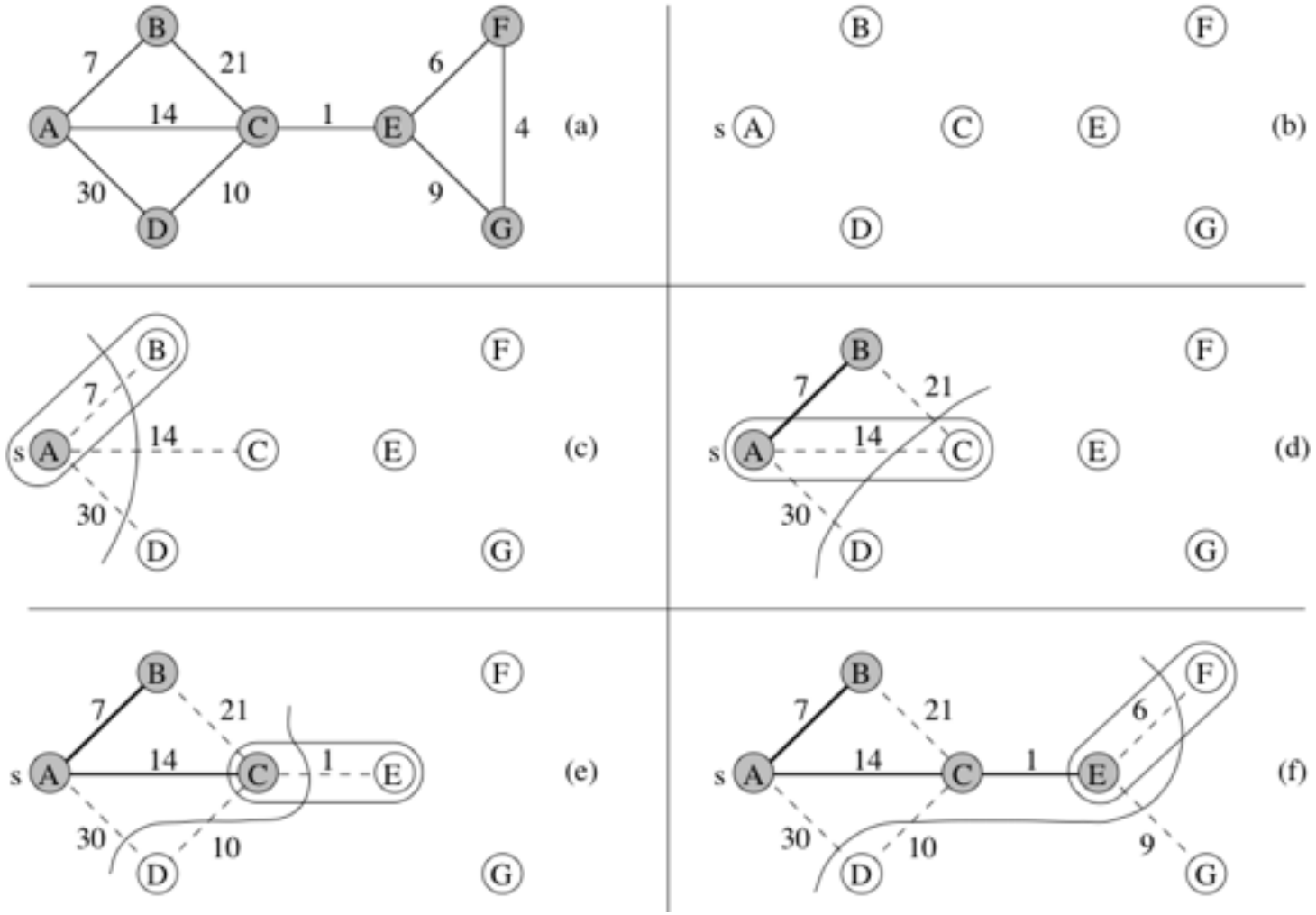
> *scegli un arco di costo minimo incidente su uno dei nodi di **T** e coloralo di **blu***

Pseudocodice

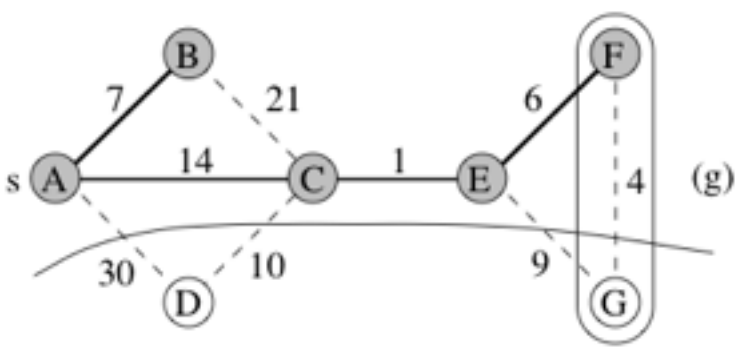
Definiamo **arco azzurro** un arco **(x,y)** tale che:
- **x∈T**
- **y∉T**

algoritmo Prim(grafo **G**) → albero
1. scegli un nodo **s**;
2. **T** ← 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**

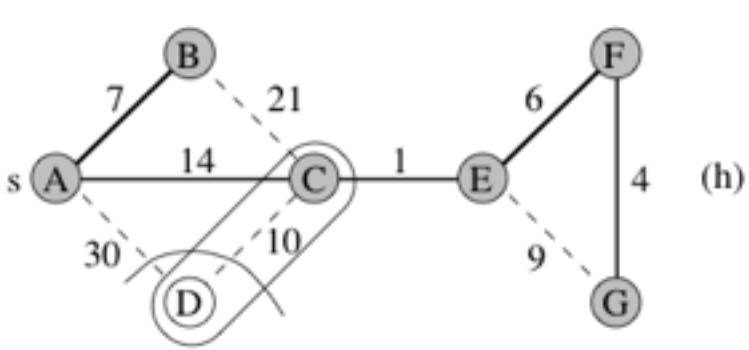
Esempio (1/2)


B
F
7
21
6
A
14
C
1
E
4
(a)
30
10
9
D
G
s
A
B
F
C
E
(b)
D
G
B
F
7
s
A
14
C
E
(c)
30
D
G
B
F
7
21
s
A
14
C
E
(d)
30
D
G
B
F
7
21
s
A
14
C
1
E
(e)
30
10
D
G
B
F
7
21
6
s
A
14
C
1
E
(f)
30
10
9
D
G

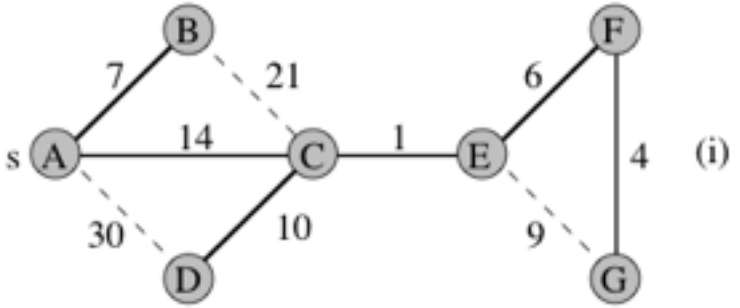
Esempio (2/2)



(g)



(h)



(i)

Analisi

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

Algoritmo di Borůvka

Strategia

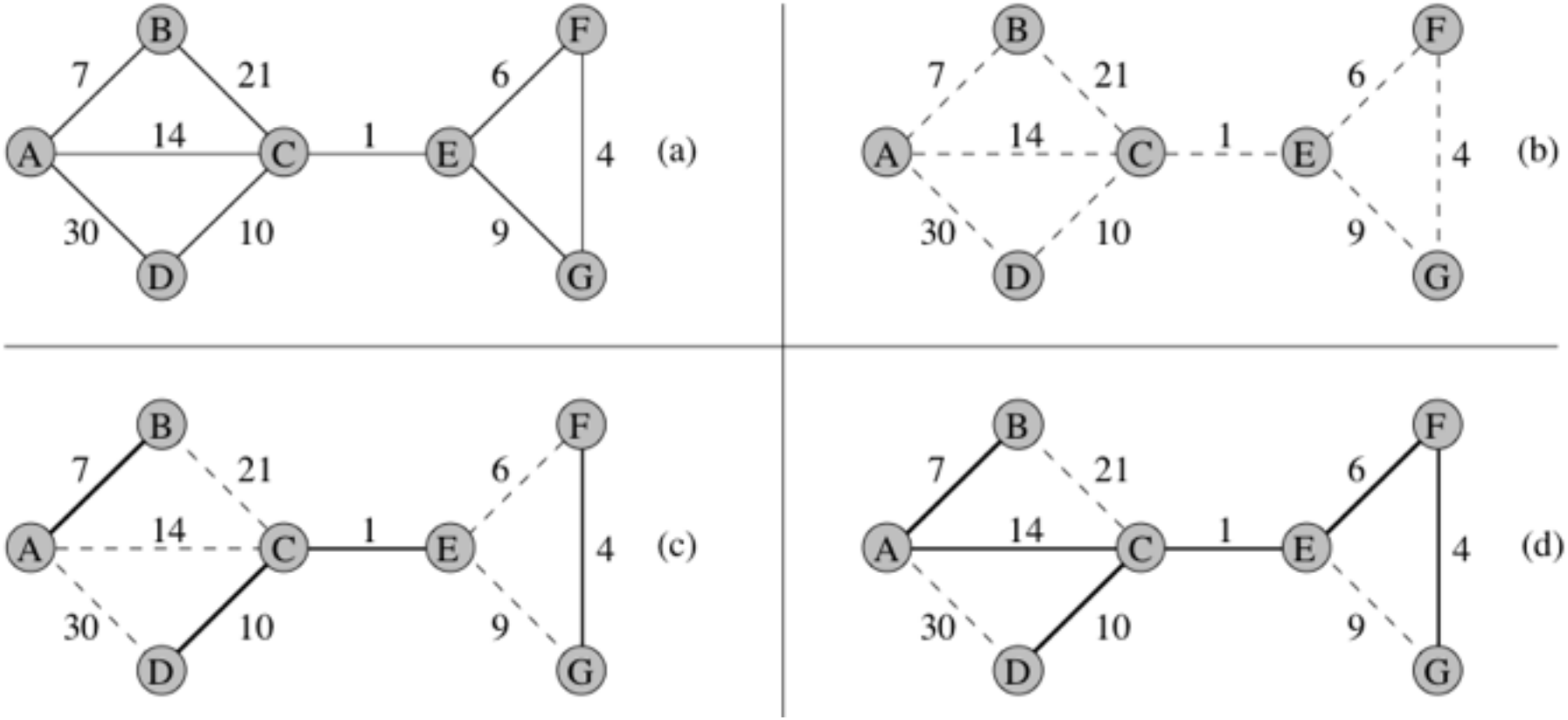
- Mantiene una foresta di alberi blu, all'inizio tutti disgiunti
- Ad ogni passo, applica la seguente regola:
  *"per ogni albero blu **T** nella foresta, scegli un arco di costo minimo incidente su **T** e coloralo blu (applica la regola del taglio)"*
- (Assumiamo che i costi degli archi siano tutti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

Pseudocodice

algoritmo Boruvka(grafo **G = (V,E)**) → albero

1. **T** ← 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**∈**C** e **v**∉**C**;
5. aggiungi l'arco **(u,v)** a **T**;
6. **return T**

Esempio


B
F
7
21
6
A
14
C
1
E
4
(a)
30
10
9
D
G
B
F
7
21
6
A
14
C
1
E
4
(b)
30
10
9
D
G
B
F
7
21
6
A
14
C
1
E
4
(c)
30
10
9
D
G
B
F
7
21
6
A
14
C
1
E
4
(d)
30
10
9
D
G

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