Grafi, minimo albero ricoprente

Corso di **Algoritmi e strutture dati**
Corso di Laurea in **Informatica**
Docenti: Ugo de'Liguoro, András Horváth

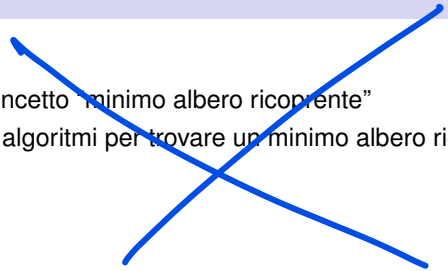
Indice

1. Definizione
2. Algoritmo generico
3. Algoritmo di Kruskal
4. Algoritmo di Prim

Sommario

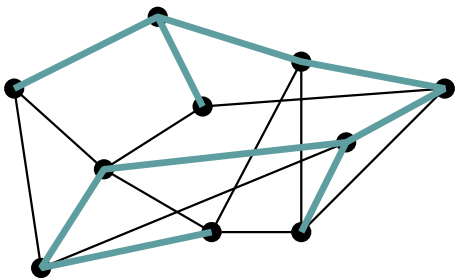
Obiettivo:

- capire il concetto "minimo albero ricoprente"
- sviluppare algoritmi per trovare un minimo albero ricoprente



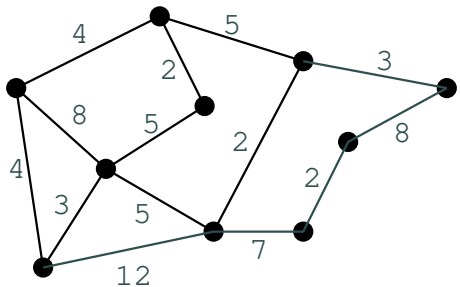
1. Albero ricoprente

- sia dato un grafo connesso e non orientato
- un albero ricoprente è un sottografo che
  - contiene tutti nodi
  - è aciclico
  - è connesso



1. Peso di un grafo

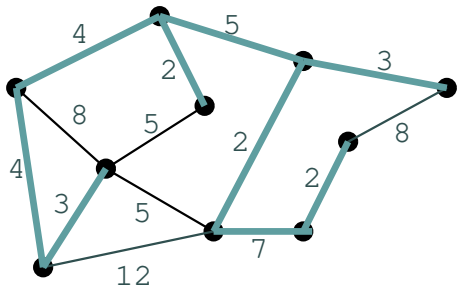
- dato un grafo $G = (V, E)$ non orientato e pesato con funziona peso W
- il peso del grafo è $W(G) = \sum_{e \in E} W(e)$



- $W(G) = 70$

1. Minimo albero ricoprente

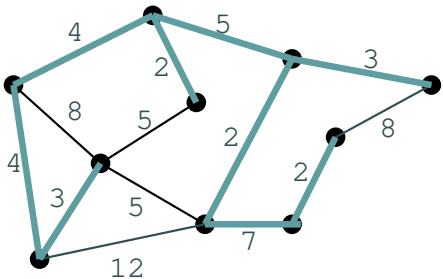
- problema: dato un grafo $G = (V, E)$ non orientato e pesato trovare un suo albero ricoprente T che abbia peso minimo



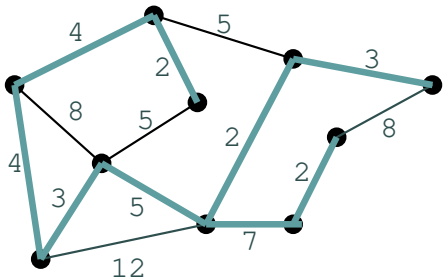
- T tale che $W(T)$ è minimo: $W(T) = 32$

1. Minimo albero ricoprente

- minimo albero ricoprente non è unico



$W(T) = 32$



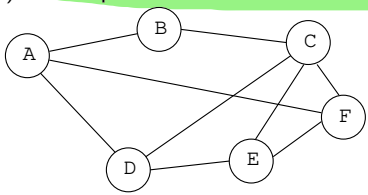
$W(T) = 32$

2. Algoritmo generico

- sia dato un grafo $G = (V, E)$
- **MINIMO_ALBERO_RICOPRENTE(G)**
  $A \leftarrow \emptyset$
  while A non è un albero ricoprente **do**
    trova un arco (u, v) che sia "sicuro" per A
    $A \leftarrow A \cup (u, v)$
- (u, v) è un arco "sicuro" per A: $A \cup (u, v)$ è un sottoinsieme degli archi di un minimo albero di copertura di G
- al termine dell'algoritmo $T = (V, A)$ è un minimo albero di copertura
- come si trova un arco "sicuro"?

2. Tagli

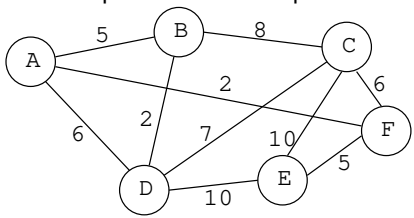
- **taglio** di un grafo $G(V, E)$: è una partizione di V in due insiemi, X e $V - X$



- $(X = \{A, C, E\}, V - X = \{B, D, F\})$
- un arco (u, v) **attraversa** il taglio $(X, V - X)$ se $u \in X, v \in V - X$
  - l'arco (B, C) è un arco che attraversa il taglio $(X = \{A, C, E\}, V - X = \{B, D, F\})$

2. Tagli

- un taglio **rispetta** un insieme di archi A se nessun arco di A attraversa il taglio
- un arco che attraversa un taglio è un **arco leggero** se è un arco di peso minimo tra quelli che attraversano il taglio



- il taglio $(X = \{A, C, E\}, V - X)$ rispetta l'insieme di archi $\{(C, E), (B, D)\}$ ma non rispetta $\{(A, F), (B, C)\}$
- l'arco (A, F) è leggero per il taglio $(X = \{A, C, E\}, V - X)$

2. Criterio per riconoscere archi “sicuri”

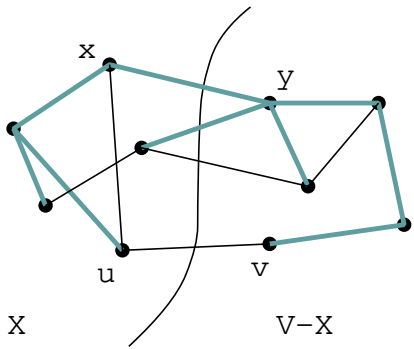
- teorema I:
  - sia $G = (V, E)$ un grafo non orientato, connesso e pesato
  - sia A un sottoinsieme di E contenuto in un qualche albero di copertura minimo
  - sia $(X, V - X)$ un qualunque taglio che rispetta A
  - sia (u, v) un arco leggero che attraversa $(X, V - X)$
  - allora l’arco (u, v) è sicuro per A

2. Dimostrazione di teorema I

- sia T un albero di copertura minimo che contiene A
- se T contiene l'arco (u, v) allora l'arco è sicuro per A
- assumiamo allora che T non contenga l'arco (u, v)
- mostreremo che (u, v) è sicuro per A costruendo un altro albero di copertura minimo T' che contenga $A \cup (u, v)$

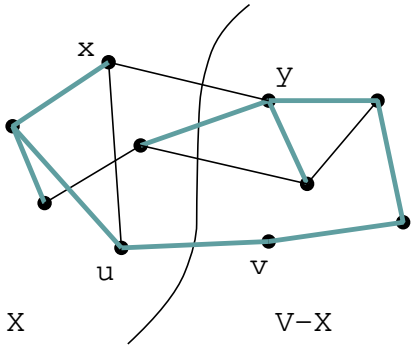
2. Dimostrazione di teorema I

- T è un albero di copertura
- in T deve esserci un cammino da u a v
- tale cammino deve contenere almeno un arco (x, y) con $x \in X, y \in V - X$



2. Dimostrazione di teorema I

- consideriamo l'albero T' con archi $E(T') = (E(T) - (x, y)) \cup (u, v)$



2. Dimostrazione di teorema I

- $W(T') = W(T) - W(x, y) + W(u, v)$
- (u, v) è un arco leggero che attraversa il taglio $(X, V - X)$
- anche (x, y) attraversa il taglio $(X, V - X)$
- per cui $W(u, v) \leq W(x, y)$ e quindi $W(T') \leq W(T)$
- T è per ipotesi un albero di copertura minimo
- per cui $W(T') = W(T)$
- T' è un albero di copertura minimo e $A \cup (u, v) \subseteq E(T')$
- quindi abbiamo dimostrato che (u, v) è un arco sicuro

2. Criterio per riconoscere archi “sicuri”

- corollario I:
  - sia $G = (V, E)$ un grafo non orientato, connesso e pesato
  - sia A un sottoinsieme di E contenuto in un albero di copertura minimo
  - sia C una componente connessa (un albero) nella foresta $G(A) = (V, A)$
  - sia (u, v) è un arco leggero che connette C a una qualche altra componente connessa di $G(A)$
  - allora l’arco (u, v) è sicuro per A

2. Criterio per riconoscere archi “sicuri”

- dimostrazione:
  - siano X i vertici di C
  - il taglio $(X, V - X)$ rispetta A
  - segue dal teorema I che l’arco (u, v) è sicuro per A

2. Algoritmi concreti

- due algoritmi:
  - di Kruskal
  - di Prim
- possono essere descritti come specializzazioni dell'algoritmo generico
- si basano sul precedente corollario
- algoritmi greedy: un arco sicuro è un arco di peso minimo, quindi il più appetibile

3. Algoritmo di Kruskal

- mantiene un sottografo non necessariamente connesso (V, A) di un MST (minimal spanning tree, minimo albero ricoprente)
- all'inizio: tutti i vertici del grafo e nessun arco
- per ogni arco, in ordine non decrescente di costo:
  - se l'arco ha entrambi gli estremi nella stessa componente connessa di (V, A) escludilo dalla soluzione
  - altrimenti, basandoti sul corollario I, includilo nella soluzione (fondendo due componenti connesse)

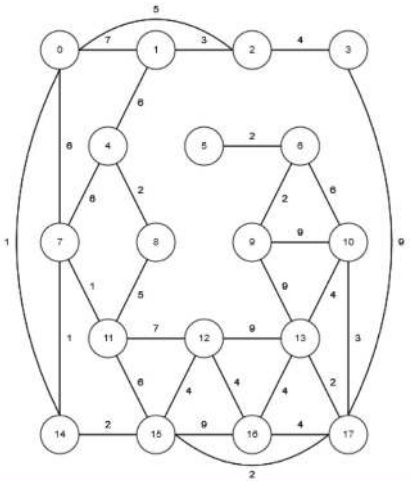
3. Algoritmo di Kruskal applicando la schema generale

- **MST_Kruskal(**G**)**
  $A \leftarrow \emptyset$
  ordina gli archi in ordine non decrescente di peso
  for $\forall (u, v)$ nell'ordine **do**
    if (u, v) è "sicuro" per A **then**
      $A \leftarrow A \cup (u, v)$
- che cosa significa "sicuro" per Kruskal?

3. Algoritmo di Kruskal

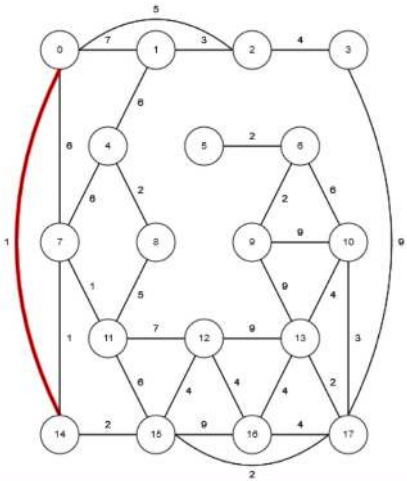
- obiettivo: la costruzione di un particolare albero, ossia un grafo connesso senza cicli
- **MST_Kruskal(**G**)**
  $A \leftarrow \emptyset$
  ordina gli archi in ordine non decrescente di peso
  for $\forall (u, v)$ nell'ordine **do**
    if (u, v) non crea ciclo in $G(A) = (V, A)$ **then**
      $A \leftarrow A \cup (u, v)$

3. Simulazione



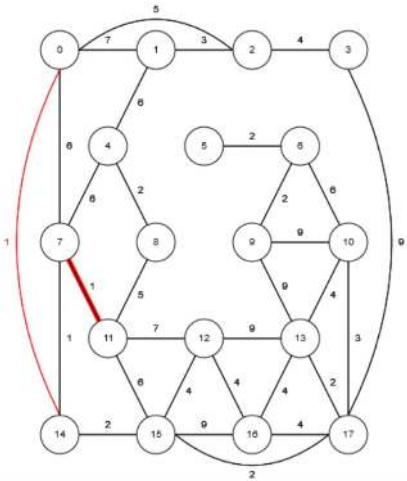
Grafo originale.

3. Simulazione



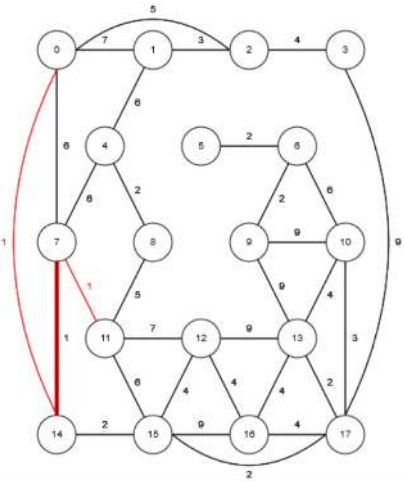
Non crea ciclo, incluso.

3. Simulazione



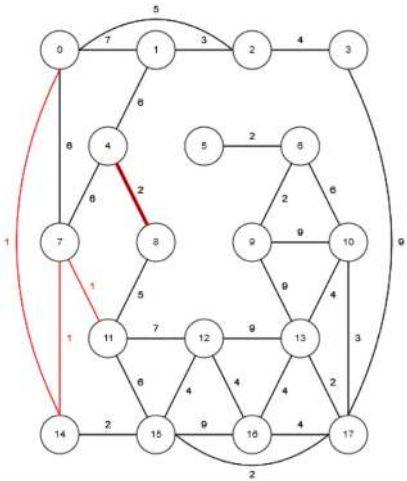
Non crea ciclo, incluso.

3. Simulazione



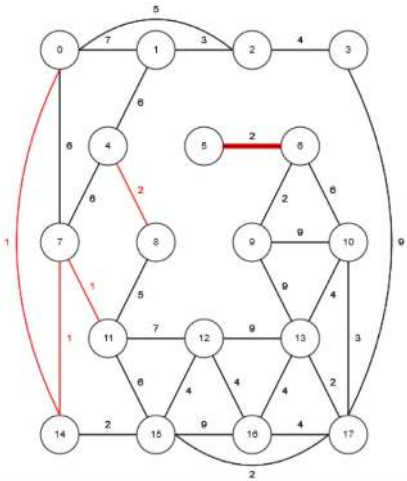
Non crea ciclo, incluso.

3. Simulazione



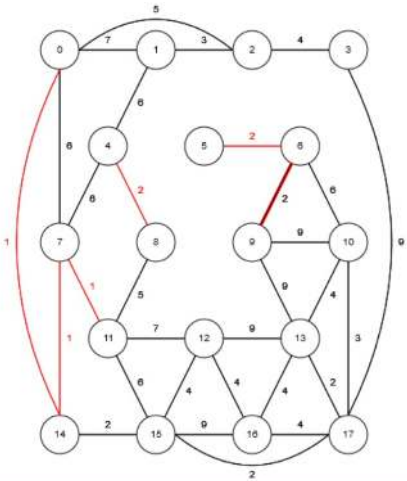
Non crea ciclo, incluso.

3. Simulazione



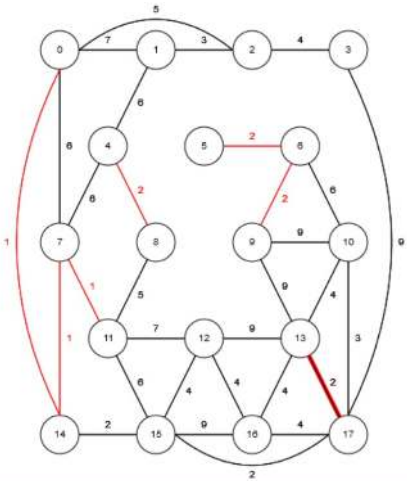
Non crea ciclo, incluso.

3. Simulazione



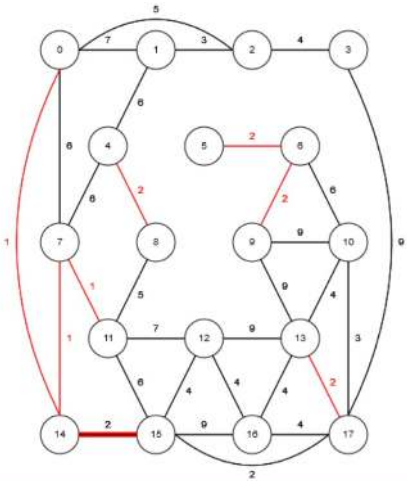
Non crea ciclo, incluso.

3. Simulazione



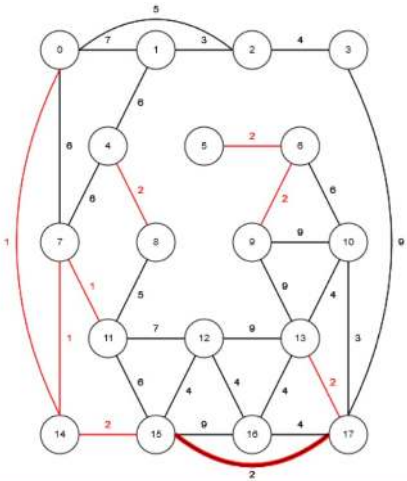
Non crea ciclo, incluso.

3. Simulazione



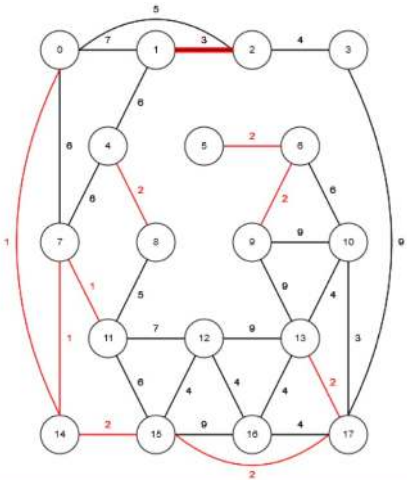
Non crea ciclo, incluso.

3. Simulazione



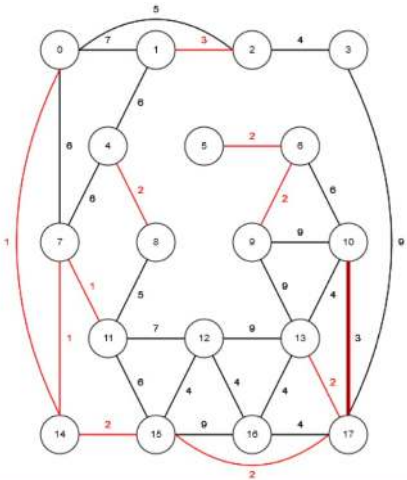
Non crea ciclo, incluso.

3. Simulazione



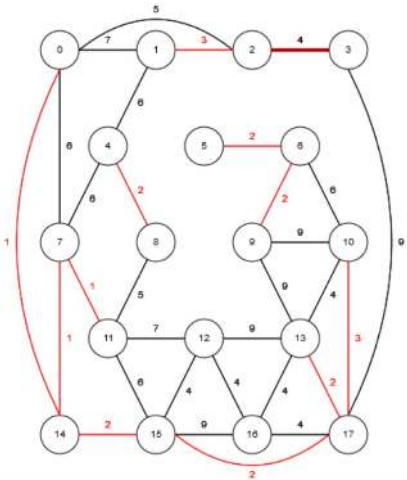
Non crea ciclo, incluso.

3. Simulazione



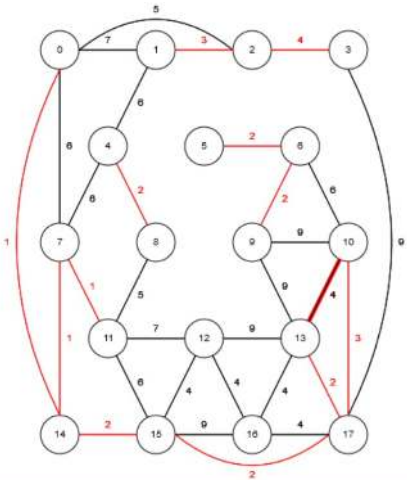
Non crea ciclo, incluso.

3. Simulazione



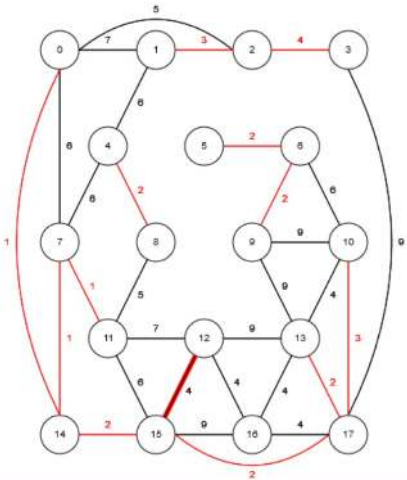
Non crea ciclo, incluso.

3. Simulazione



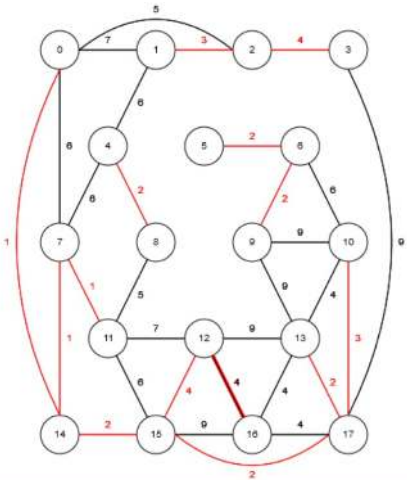
Crea ciclo, escluso.

3. Simulazione



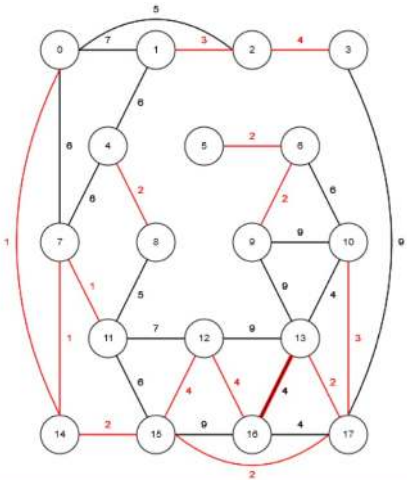
Non crea ciclo, incluso.

3. Simulazione



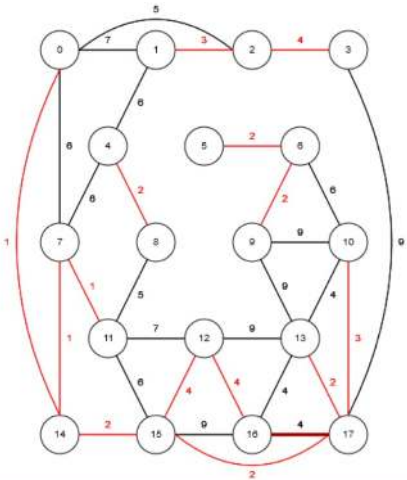
Non crea ciclo, incluso.

3. Simulazione



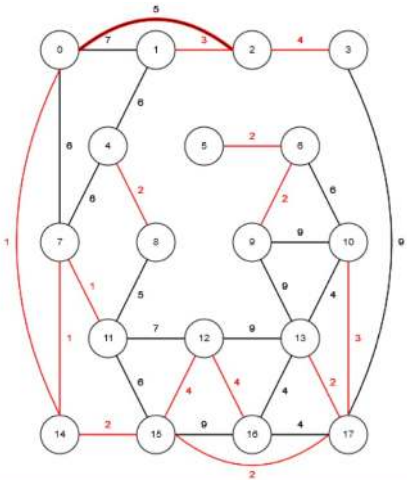
Crea ciclo, escluso.

3. Simulazione



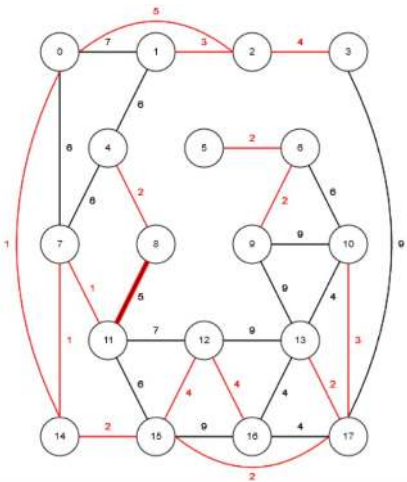
Crea ciclo, escluso.

3. Simulazione



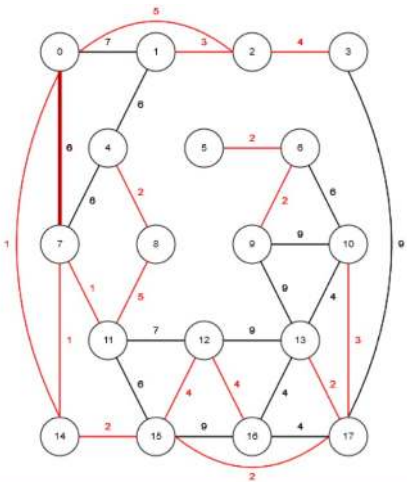
Non crea ciclo, incluso.

3. Simulazione



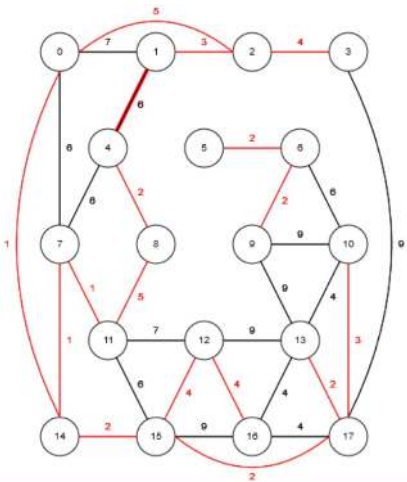
Non crea ciclo, incluso.

3. Simulazione



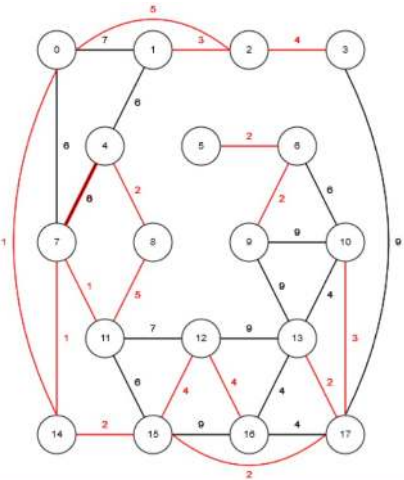
Crea ciclo, escluso.

3. Simulazione



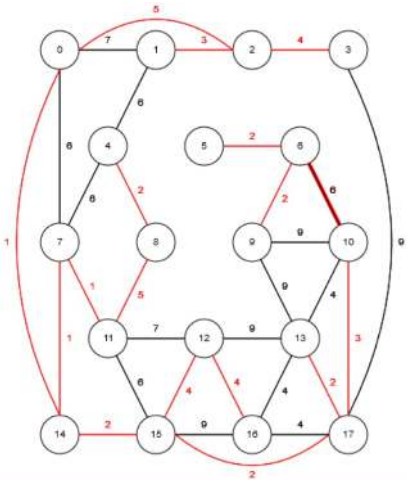
Crea ciclo, escluso.

3. Simulazione



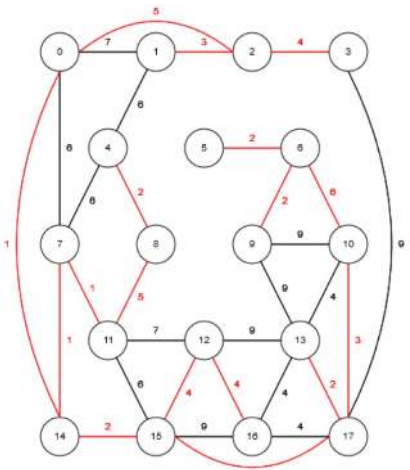
Crea ciclo, escluso.

3. Simulazione



Non crea ciclo, incluso.

3. Simulazione



Risultato finale.

3. Complessità dell'algoritmo di Kruskal

- non può essere valutata se non si conosce la struttura dati usata per rispondere alla domanda di esistenza del ciclo
- bisogna memorizzare i vertici delle componenti connesse del sottoinsieme dell albero di copertura in costruzione
- una struttura dati per insiemi disgiunti
  - collezione S di insiemi dinamici disgiunti (di vertici)
  - ogni insieme identificato da un rappresentante

3. Struttura dati che rappresenta insiemi disgiunti: operazioni

- tre operazioni
  - creazione di un nuovo insieme il cui unico elemento è x: Make_set(x)
  - unione di due insiemi che contengono x e y: Union(x, y)
  - trovare il rappresentante dell insieme contenente x: Find_set(x)
- una sequenza di $n + m$ operazioni Make_set, Find_set e Union, di cui
  - n sono operazioni di Make_set
  - m sono operazioni di union e/o find
- può essere eseguita su una UnionFind in tempo $O((n + m) \log n)$ nel caso peggiore

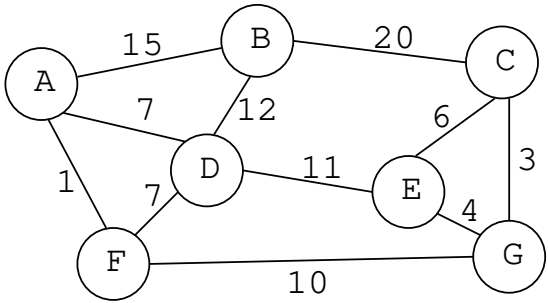
3. Algoritmo di Kruskal con insiemi disgiunti

MST_Kruskal(G**)**
  $A \leftarrow \emptyset$
  for $\forall v \in V$ **do**
    Make_set(v)
  ordina gli archi in ordine non decrescente di peso
  for $\forall (u, v) \in E$ nell'ordine **do**
    if Find(u)$\neq$Find(v) **then**
      $A \leftarrow A \cup (u, v)$
      Union(u, v)

3. Complessità nel caso peggiore

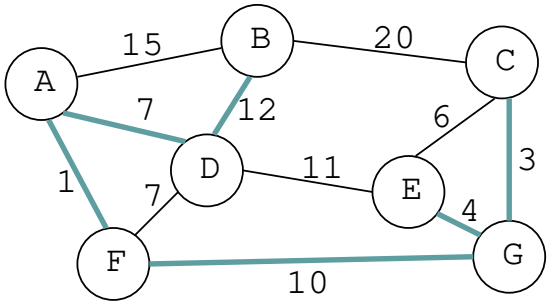
- ordinamento: $O(|E| \log |E|)$
- operazioni sulla foresta di insiemi disgiunti: $O((|V| + |E|) \log |V|)$
- il grafo è connesso: $O((|V| + |E|) \log |V|) = O(|E| \log |V|)$
- totale: $O(|E| \log |V|)$ (dato che $O(|E| \log |E|) = O(|E| \log |V|^2) = O(|E| 2 \log |V|) = O(|E| \log |V|)$)

3. Esempio



- gli archi in ordine per peso non decrescente: (a, f), (c, g), (e, g), (c, e), (a, d), (d, f), (f, g), (d, e), (b, d), (a, b), (b, c)

3. Esempio



- MST non è unico: dipende dal ordine in cui gli archi di peso uguale si considerano

3. Correttezza dell'algoritmo di Kruskal

- invariante del ciclo: A è un sottoinsieme degli archi di un MST di G:
  - l'invariante viene reso vero dall inizializzazione di A come insieme vuoto
  - corpo del ciclo **for** mantiene l'invariante:
    - se l'arco crea un ciclo non viene aggiunto
    - se non crea un ciclo allora per corollario I l'arco è "sicuro" e l'invariante viene mantenuto
- all'termine del algoritmo (V, A) è connesso (si può dimostrare per assurdo)

4. Algoritmo di Prim

- mantiene un sottografo connesso $(V - Q, A)$ di un MST
- all'inizio consiste di un nodo arbitrario:
  - Q contiene tutti i nodi tranne questo nodo iniziale
  - A è vuoto
- applica $n - 1$ volte il seguente passo
- scegli un arco (u, v) di peso minimo che collega un nodo in $V - Q$ ad un nodo in Q
  - aggiungilo ad A
  - togli da Q il vertice a cui porta l'arco

4. Algoritmo di Prim applicando la schema generale

- **MST_Prim**(G, s)
  $A \leftarrow \emptyset$
  $Q \leftarrow V - \{s\}$
  while $Q \neq \emptyset$ **do**
    scegli un arco (u, v) "sicuro" per A
    $A \leftarrow A \cup (u, v)$
    $Q \leftarrow Q - \{v\}$
- che cosa significa "sicuro" per Prim?

4. Algoritmo di Prim

- strategia di Prim: mantenere un sottografo connesso
- **MST_Prim(**G, s**)**
  $A \leftarrow \emptyset$
  $Q \leftarrow V - \{s\}$
  while $Q \neq \emptyset$ **do**
    scegli l'arco (u, v) con peso minimo e $u \in V - Q, v \in Q$
    $A \leftarrow A \cup (u, v)$
    $Q \leftarrow Q - \{v\}$

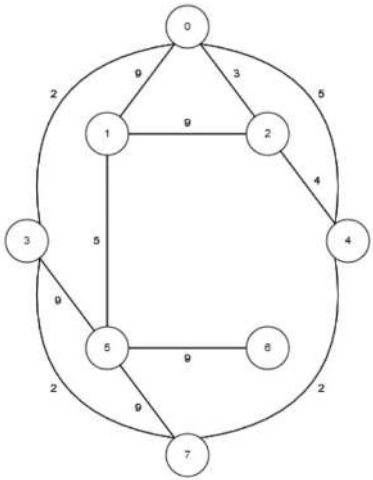
4. Algoritmo di Prim

- per facilitare la scelta dell'arco successivo memorizziamo per ogni nodo $u \in Q$ l'arco più leggero fra $V - Q$ e u
- l'attributo π è l'estremità dell'arco in $V - Q$ e d è il suo peso

MST_Prim(G, s**)**
  $Q \leftarrow V$
  for $\forall v \in V$ **do** $v.d \leftarrow \infty$
  $s.d \leftarrow 0$
  $s.\pi \leftarrow$ nil
  while $Q \neq \emptyset$ **do**
    $u \leftarrow$ nodo con d minimo in Q (tolto da Q)
    for $\forall v \in$ adj[u] **do**
      if $v \in Q$ e $W(u, v) < v.d$ **then**
        $v.d \leftarrow W(u, v)$
        $v.\pi \leftarrow u$

4. Simulazione

Vertex	Known	Cost	Path
0			
1			
2			
3			
4			
5			
6			
7			

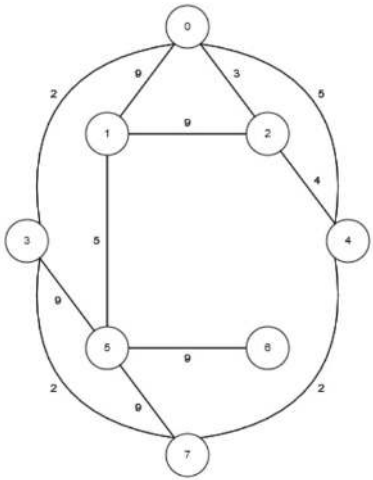


Known: true se non fa più parte di *Q*
Cost: *d*
Path: π (-1 al posto di *nil*)

4. Simulazione

Finding Cheapest Uknown Vertex

Vertex	Known	Cost	Path
0	F	0	-1
1	F	INF	-1
2	F	INF	-1
3	F	INF	-1
4	F	INF	-1
5	F	INF	-1
6	F	INF	-1
7	F	INF	-1



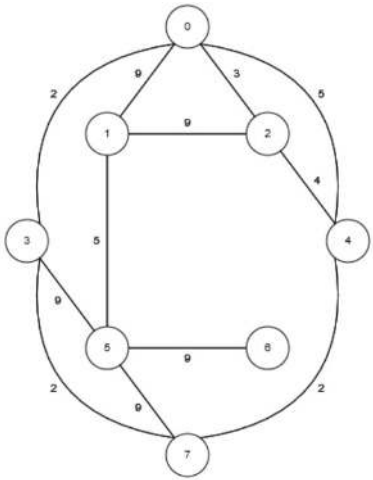
Situazione prima di cominciare il ciclo.

Nel ciclo si sceglie il nodo 0 e si aggiornano suoi adiacenti.

4. Simulazione

Finding Cheapest Uknown Vertex

Vertex	Known	Cost	Path
0	T	0	-1
1	F	9	0
2	F	3	0
3	F	2	0
4	F	5	0
5	F	INF	-1
6	F	INF	-1
7	F	INF	-1

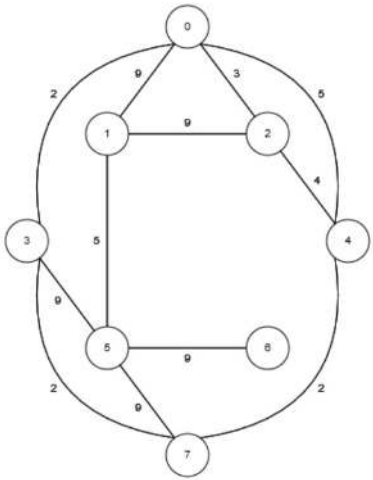


Si sceglie il nodo 3. Si aggiornano nodi 5 e 7.

4. Simulazione

Finding Cheapest Uknown Vertex

Vertex	Known	Cost	Path
0	T	0	-1
1	F	9	0
2	F	3	0
3	T	2	0
4	F	5	0
5	F	9	3
6	F	INF	-1
7	F	2	3

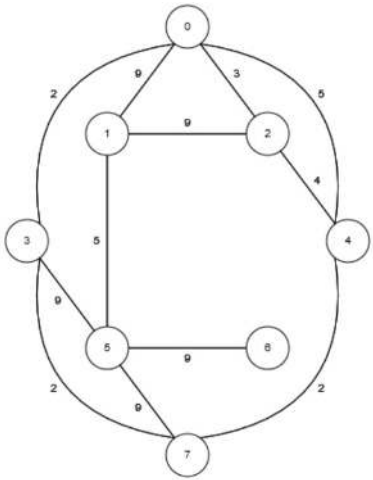


Si sceglie il nodo 7. Si aggiorna nodo 4.

4. Simulazione

Finding Cheapest Uknown Vertex

Vertex	Known	Cost	Path
0	T	0	-1
1	F	9	0
2	F	3	0
3	T	2	0
4	F	2	7
5	F	9	3
6	F	INF	-1
7	T	2	3

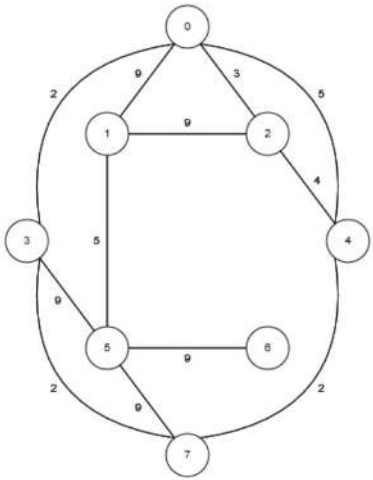


Si sceglie il nodo 4. Non si aggiorna nessun nodo.

4. Simulazione

Finding Cheapest Uknown Vertex

Vertex	Known	Cost	Path
0	T	0	-1
1	F	9	0
2	F	3	0
3	T	2	0
4	T	2	7
5	F	9	3
6	F	INF	-1
7	T	2	3

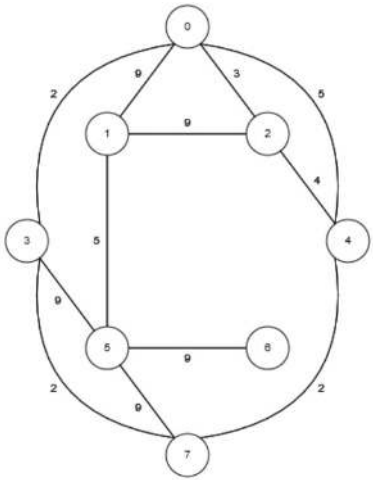


Si sceglie il nodo 2. Non si aggiorna nessun nodo.

4. Simulazione

Finding Cheapest Uknown Vertex

Vertex	Known	Cost	Path
0	T	0	-1
1	F	9	0
2	T	3	0
3	T	2	0
4	T	2	7
5	F	9	3
6	F	INF	-1
7	T	2	3

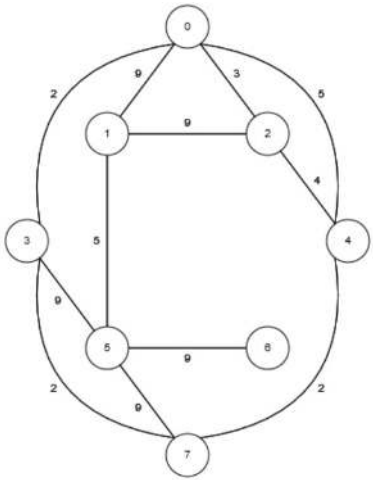


Si sceglie il nodo 1. Si aggiorna nodo 5.

4. Simulazione

Finding Cheapest Uknown Vertex

Vertex	Known	Cost	Path
0	T	0	-1
1	T	9	0
2	T	3	0
3	T	2	0
4	T	2	7
5	F	5	1
6	F	INF	-1
7	T	2	3

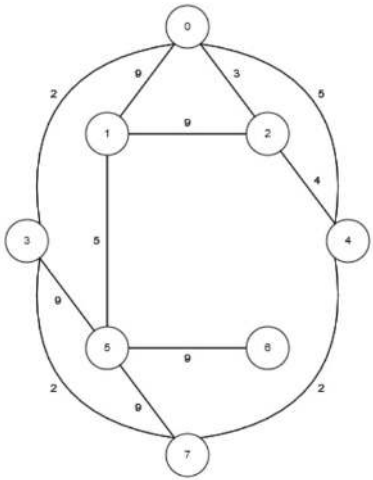


Si sceglie il nodo 5. Si aggiorna nodo 6.

4. Simulazione

Finding Cheapest Uknown Vertex

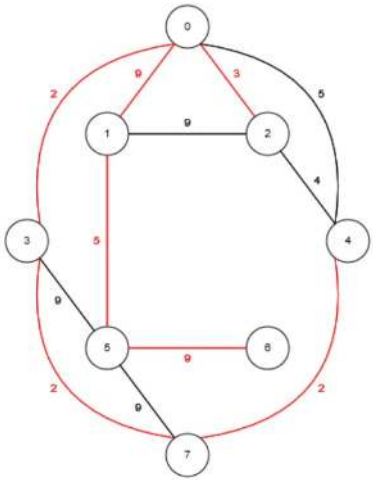
Vertex	Known	Cost	Path
0	T	0	-1
1	T	9	0
2	T	3	0
3	T	2	0
4	T	2	7
5	T	5	1
6	F	9	5
7	T	2	3



Si sceglie il nodo 6.

4. Simulazione

Vertex	Known	Cost	Path
0	T	0	-1
1	T	9	0
2	T	3	0
3	T	2	0
4	T	2	7
5	T	5	1
6	T	9	5
7	T	2	3



Situazione finale.

4. Complessità dell'algoritmo di Prim

- Q può essere implementata come coda di priorità usando un heap minimo
- le priorità sono date dall attributo d
- il costo dell algoritmo di Prim è limitato da
  - costo inizializzazione: $O(|V|)$
  - costo delle estrazioni del minimo: $O(|V| \log |V|)$
  - costo di risistemazione dello heap binario dopo il decremento eventuale delle chiavi: $O(|E| \log |V|)$
- complessità dell algoritmo con: $O((|V| + |E|) \log |V|) = O(|E| \log |V|)$

4. Correttezza dell'algoritmo di Prim

- due invarianti del ciclo:
  - invariante I: $\forall t \in Q : t.\pi \neq nil \Rightarrow t.\pi \in V - Q$
  - invariante II: $\forall t \in Q : (t.\pi, t)$ un arco di peso minimo tra t e un vertice di $V - Q$
- si dimostrano facilmente esaminando l'inizializzazione e il corpo del ciclo

4. Correttezza dell'algoritmo di Prim

- dimostriamo il seguente invariante: $(V - Q, \{(t.\pi, t) : t \neq s, t \in V - Q\})$ è un sottografo di un MST:
  - inizialmente vero
  - il corpo del ciclo mantiene l'invariante:
    - sia u il vertice estratto da Q
    - per invariante I e II, l'arco $(u.\pi, u)$ è un arco leggero che unisce le componenti connesse $(V - Q, A)$ e $(\{u\}, \emptyset)$
    - allora, per il Corollario 1, è un arco sicuro per A