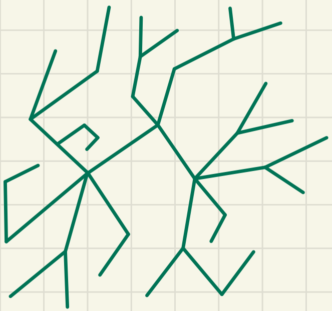
Algoritmi e Strutture Dati

Lezione 22

12 novembre 2025

Albero ricoprente minimo

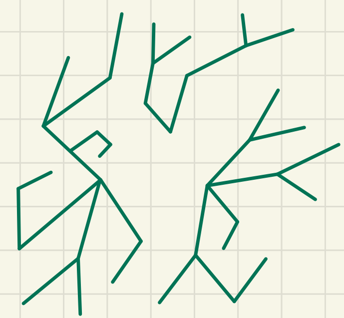
ALBERI



ALBERO. grafo NON ORIENTATO, CONNESSO e PRIVO DI CICLI

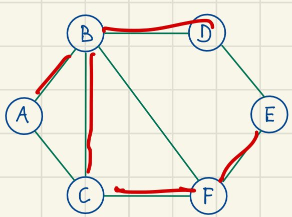
Proprietà $G=(V,E)$ non orientato e connesso è un albero sse $\#E = \#V - 1$

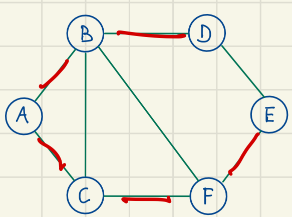
FORESTA: insieme di alberi



ALBERO di SUPPORTO o RICOPRENTE (Spanning tree)

Dato $G=(V,E)$ grafo non orientato connesso, un albero ricoprente di G è un albero $G'=(V',E')$ con $V'=V$ e $E' \subseteq E$





ALBERO RICOPRENTE MINIMO

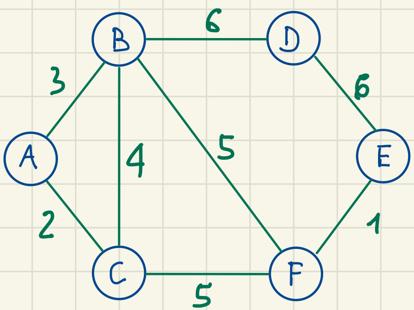
<u>Problema</u> Dato $G = (V, E)$ non orientato connesso con una funzione peso $w : E \to \mathbb{R}$ trovare un albero ricoprente $T = (V, E_T)$ di peso minimo

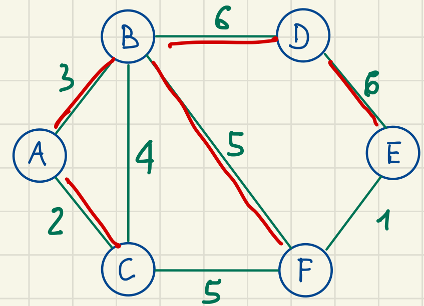
Dato $G' = (V, E')$: $\quad w(G') = \sum_{e \in E'} w(e)$

$T = (V, E_T)$ è una soluzione ottima del problema se:

- $E_T \subseteq E$, T è un albero
- $\forall\, T' = (V, E_{T'})$, se T' è un albero ricoprente allora $w(T) \leq w(T')$

ESEMPIO

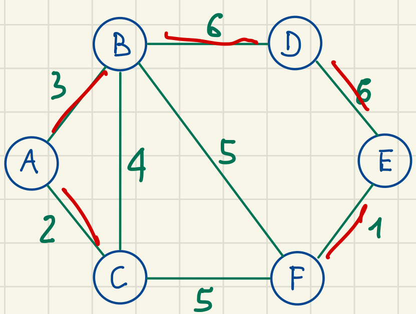




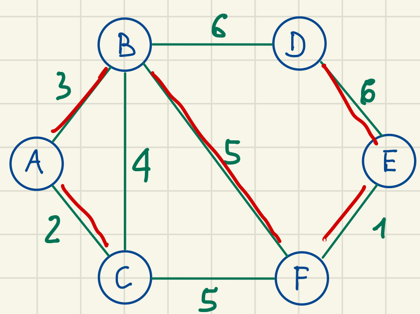
$w(T) = 22$

ricerca esaustiva

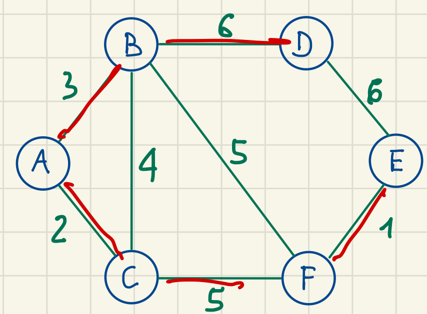
$$\binom{m}{n-1} \approx m^n$$



$w(T) = 18$



$w(T) = 17$



$w(T) = 12$

ALBERO RICOPRENTE MINIMO: una strategia greedy

Inizialmente:

$T = (V, \emptyset)$

$C = E$ insieme dei candidati

Ad ogni passo:

preleva da C l'arco di peso minimo

lo aggiungo a T se non forma cicli

con gli archi già scelti

```
ALGORITMO greedy (insieme C) → soluzione
  S ← ∅
  WHILE C ≠ ∅ DO
    x ← seleziona (C)
    C ← C − {x}
    IF S ∪ {x} è ammissibile THEN
       S ← S ∪ {x}
  RETURN S
```

ALGORITMO Kruskal (Grafo $G = (V, E, \omega)$) $\rightarrow$ albero

ordina E in maniera non decrescente in base ai pesi

$T \leftarrow (V, \emptyset)$

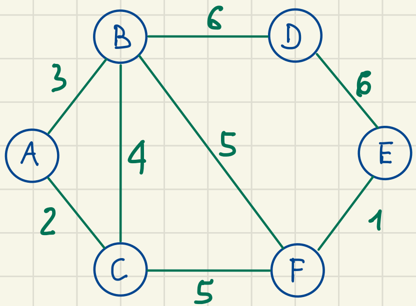
FOR EACH $(x, y) \in E$ secondo l'ordine DO

IF x e y non sono connessi in T DO

aggiungi a T l'arco (x, y)

RETURN T

ESEMPIO



(E, F)

(A, C)

(A, B)

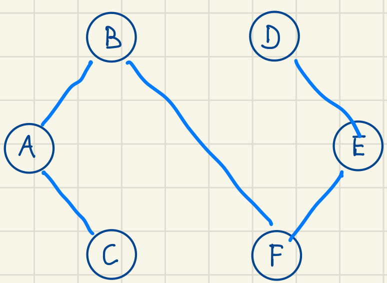
~~(B, C)~~

(B, F)

~~(C, F)~~

(D, E)

~~(B, D)~~



17

ALGORITMO Kruskal (Grafo $G=(V,E,\omega)$) → albero

ordina E in maniera non decrescente in base ai pesi

$T \leftarrow (V, \emptyset)$

FOR EACH $(x,y) \in E$ secondo l'ordine DO

  IF x e y non sono connessi in T THEN

    aggiungi a T l'arco (x,y)

RETURN T

ALGORITMO di KRUSKAL: correttezza

Teo L'algoritmo di Kruskal trova un albero ricoprente di peso minimo

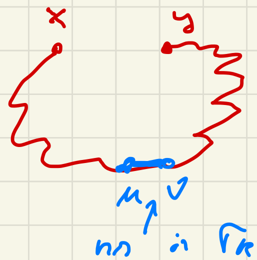
Dim T_K : albero ricoprente determinato dall'algoritmo di Kruskal

T_0: albero ricoprente minimo con il maggior numero di archi in comune con T_K

Per assurdo $T_K \neq T_0$

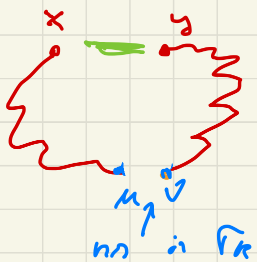
Sia (x, y) il primo arco (nell'ordine di Kruskal) t.c. (x,y) è in T_K ma non in T_0

T_0



$\exists (u,v)$ nel cammino tra x e y in T_0 t.c. (u,v) non è in T_K

$w(x,y) \leq w(u,v)$ ← ordine di Kruskal



Tolgo da T_0 (u,v) e aggiungo (x,y)

→ nuovo albero $\bar{T}$

$$w(\bar{T}) = w(T_0) - w(u,v) + w(x,y) \leq w(T_0)$$

$\bar{T}$ è un albero ricoprente minimo, con un arco in più di T_0 in comune con T_K ASSURDO

ALGORITMO Kruskal (Grafo $G=(V,E,\omega)$) → albero

ordina E in maniera non decrescente in base ai pesi

$T \leftarrow (V, \emptyset)$

FOR EACH $(x,y) \in E$ secondo l'ordine DO

IF x e y non sono connessi in T THEN

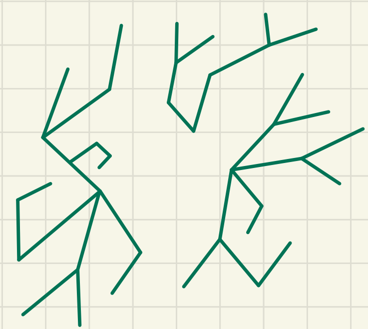
aggiungi a T l'arco (x,y)

RETURN T

rappresentazione Lista di archi

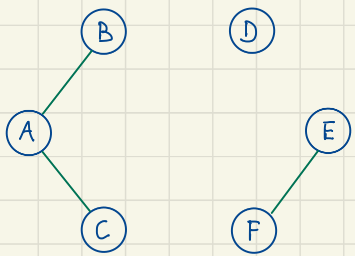
↑

Array



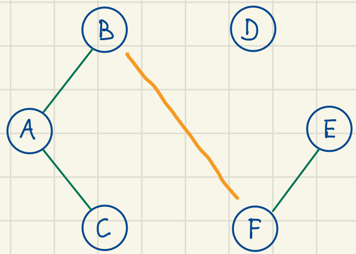
ALGORITMO di KRUSKAL: implementazione

- Organizziamo l'insieme dei vertici utilizzando una partizione
- Due vertici appartengono allo stesso elemento della partizione se e solo se sono connessi da un cammino



$\{A, B, C\}$ $\{D\}$ $\{E, F\}$

ALGORITMO di KRUSKAL: implementazione



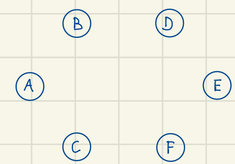
$\{A, B, C\}\ \{D\}\ \{E, F\}$

FIND(B)
FIND(C)

B, C sono connessi? SI

(B, F) sono connessi? NO

FIND(B) $\neq$ FIND(F)   UNION(..)



- Partizione iniziale: singoletti $\{A\}\ \{B\}\ \{C\}\ \{D\}\ \{E\}\ \{F\}$
- Aggiunta di un arco: unione di due elementi della partizione
- Rappresentazione del grafo: lista di archi

ALGORITMO di KRUSKAL: implementazione

- Partizione di V
  - → elemento (insieme) della partizione ⟷ albero
  - → partizione ⟷ foresta
  - → x e y connessi in T sse sono nello stesso elemento della partizione ⟷ $\mathrm{Find}(x) = \mathrm{Find}(y)$
  - → aggiunta di un arco ⟷ unione di due alberi — Union
  - → partizione iniziale ⟷ singoli vertici — Makeset

ALGORITMO di KRUSKAL: implementazione UNION/FIND

```
ALGORITMO Kruskal (Grafo G=(V,E,w)) → albero
  ordina E in maniera non decrescente in base ai pesi   ← lista d'archi
  T ← (V, ∅)
  FOR EACH vertice v∈V DO makeset(v)
  FOR EACH (x,y)∈E secondo l'ordine DO
      Tx ← FIND(x)
      Ty ← FIND(y)
      IF Tx ≠ Ty THEN
          UNION(Tx, Ty)
          aggiungi a T l'arco (x,y)
  RETURN T
```

```
ALGORITMO Kruskal (Grafo G=(V,E,w)) → albero
  ordina E in maniera non decrescente in base ai pesi
  T ← (V, ∅)
  FOR EACH (x,y)∈E secondo l'ordine DO
      IF x e y non sono connessi in T THEN
          aggiungi a T l'arco (x,y)
  RETURN T
```

ALGORITMO di KRUSKAL: complessità in Tempo

$n = \# V \qquad m = \# E$

$\boxed{1}$ Heapsort $\qquad O(m \lg m)$

$\boxed{2}$ n MakeSet $\qquad O(n)$

$\boxed{3}$ FOR-EACH $\quad m$ iterazioni

$\boxed{3a}$ # tot find : $2m$ $\qquad O(m \lg n)$

$\boxed{3b}$ # tot union : $\underline{n-1}$ $\qquad O(n)$

una per ogni arco scelto

ALGORITMO Kruskal (Grafo $G = (V, E, \omega)$) $\rightarrow$ albero

$\boxed{1}$ ordina E in maniera non decrescente in base ai pesi

$T \leftarrow (V, \emptyset)$

$\boxed{2}$ FOR EACH vertice $v \in V$ DO makeSet(v)

$\boxed{3}$ FOR EACH $(x, y) \in E$ secondo l'ordine DO

$\boxed{a}$ $T_x \leftarrow$ FIND(x)

$T_y \leftarrow$ FIND(y)] 2 find

IF $T_x \neq T_y$ THEN

$\boxed{b}$ UNION(T_x, T_y)

$T \leftarrow T \cup \{(x, y)\}$

RETURN T

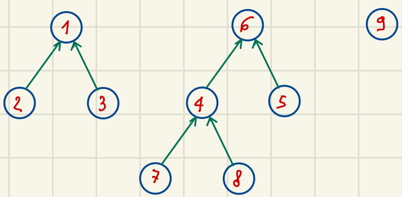
$$O(m \lg m) + O(n) + O(m \lg n) + O(n)$$

grafo connesso $\quad n - 1 \leq m \leq n^2$

$$\leq O(m \lg n^2) + O(m \lg n) = \underline{\underline{O(m \lg n)}}$$

$\lg n^2 = 2 \lg n$ $\qquad$ Tempo

QuickUnion con bilanciamento in altezza



MakeSet $O(1)$

Find $O(\lg n)$

Union $O(1)$

ALBERO RICOPRENTE MINIMO:

una strategia greedy alternativa

<u>Problema</u> Dato $G=(V,E)$ non orientato connesso con una funzione peso $w: E \to \mathbb{R}$ trovare un albero ricoprente $T=(V,E_T)$ di peso minimo

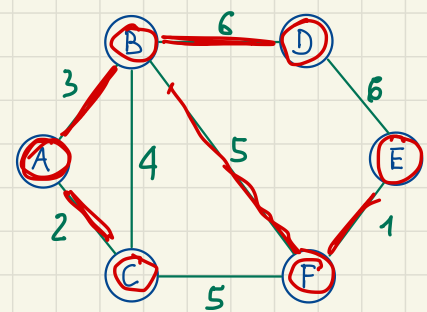
Inizialmente:

T: albero costituito da un unico vertice

Algoritmo di Prim

Ad ogni passo:

si espande T aggiungendo l'arco (x,y) di peso minimo con un vertice in T e l'altro non in T



ALGORITMO Prim (Grafo $G=(V,E,\omega)$) $\rightarrow$ albero

$T \leftarrow$ albero costituito da un unico vertice $s \in V$ qualsiasi

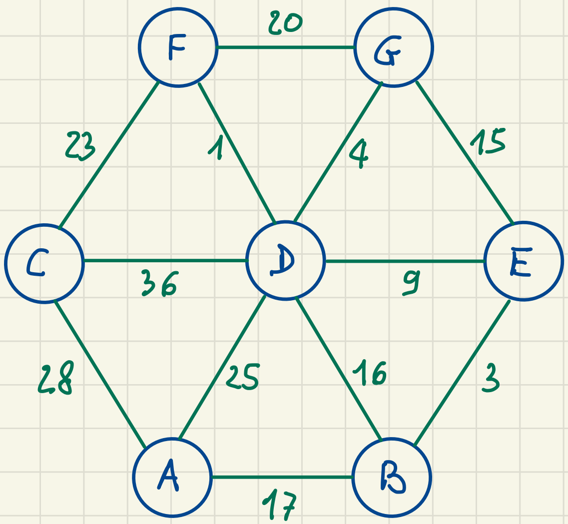
WHILE T ha meno di n nodi DO

sia (x,y) l'arco di peso minimo

con x in T e y non in T

aggiungi a T il vertice y e l'arco (x,y)

RETURN T



ALGORITMO Prim (Grafo $G = (V, E, \omega)$) $\rightarrow$ albero

T $\leftarrow$ albero costituito da un unico vertice $s \in V$ qualsiasi

WHILE T ha meno di n nodi DO

sia (x, y) l'arco di peso minimo
con x in T e y non in T

aggiungi a T il vertice y e l'arco (x, y)

RETURN T