Algorísmica Avançada
Algorismes greedy

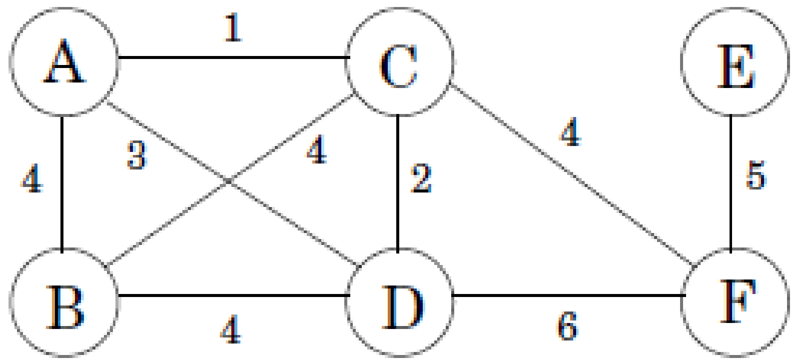
Sergio Escalera

Algorismes greedy

- Podem guanyar als escacs pensant només en la següent jugada?
- I al scrabble? → algorisme greedy?
- **Algorismes greedy troben la millor "jugada" a cada pas**

Algorismes greedy

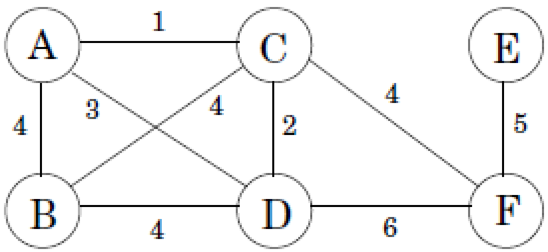
- Exemple

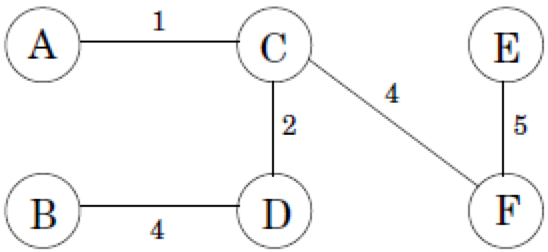


- Volem connectar els ordinadors (nodes) d'una xarxa. Les connexions són les arestes. Cadascuna té un cost. Volem el mínim cost.
  - → llavors no volem cicles
  - → volem un graf no dirigit acíclic connectat
    - → arbre !!!
  - → de mínim cost: **Minimum Spanning Tree (MST)**

Algorismes greedy

- MST amb cost 16 (un dels possibles)



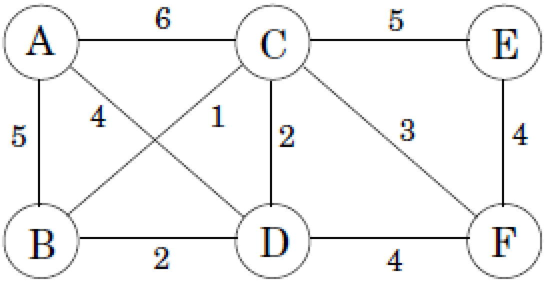


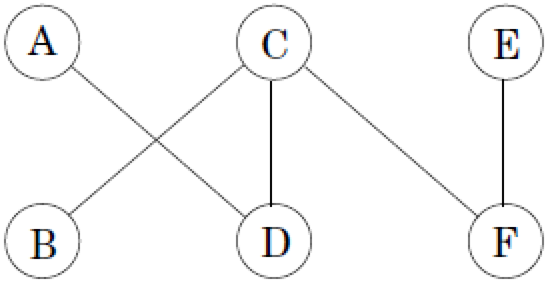
- Algorisme greedy: Kruskal
  - Començar amb arbre buit
  - Mentre no estiguin tots els nodes connectats:
    - Incloure aresta de cost mínim que no produeix un cicle

Algorismes greedy

- Cost 14!

$$B-C,\ C-D,\ B-D,\ C-F,\ D-F,\ E-F,\ A-D,\ A-B,\ C-E,\ A-C.$$





- Aquest algorisme és **òptim!**

Algorismes greedy

- **¿Per què? Propietat de tall “cut”**:
- Un tall és aquella aresta que si la traiem es genera una nova component connexa.
- El que fem amb Kruskal és anar connectant elements amb el tall de cost mínim.
- Cóm ho podem implementar eficientment?

Algorismes greedy

procedure kruskal (G, w)

Input: A connected undirected graph $G = (V, E)$ with edge weights w_e

Output: A minimum spanning tree defined by the edges X

for all $u \in V$:
    makeset(u)

$X = \{\}$

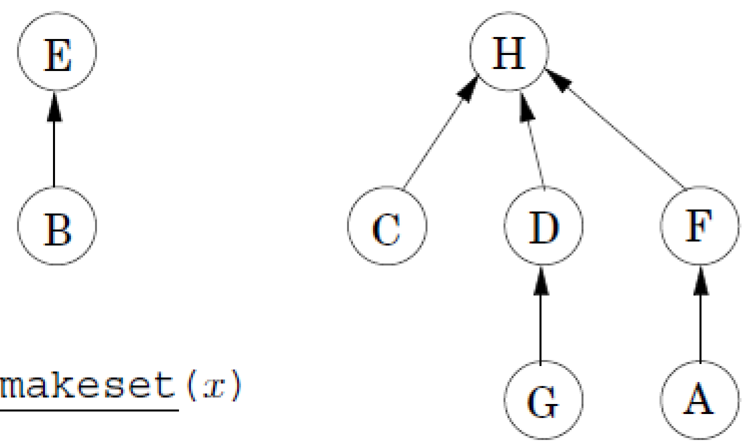
Sort the edges E by weight

for all edges $\{u, v\} \in E$, in increasing order of weight:
    if find$(u) \neq$ find(v):
        add edge $\{u, v\}$ to X
        union(u, v)

$|V|$ makeset,
$2|E|$ find
$|V| - 1$ union

Algorismes greedy

- Representació dels conjunts: arbres dirigits



<u>procedure makeset</u>(x)
$\pi(x) = x$
rank$(x) = 0$

<u>function find</u>(x)
while $x \neq \pi(x)$: $x = \pi(x)$
return x

π punter
rank: altura dins de l’arbre

Algorismes greedy

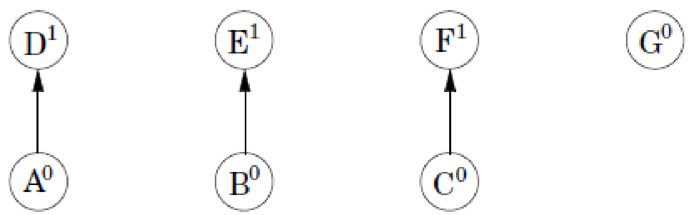
- **Makeset**: temps constant
- **Find**: segueix punters dels pares als roots, per tant el temps és proporcional a l'altura
- **Union**: com l'altura ens defineix la complexitat, posem el punter de l'arbre més curt apuntant al punter de l'arbre amb més altura

$$
\begin{array}{l}
\underline{\texttt{procedure union}}(x, y) \\
r_x = \texttt{find}(x) \\
r_y = \texttt{find}(y) \\
\texttt{if } r_x = r_y\texttt{:} \quad \texttt{return} \\
\texttt{if } \texttt{rank}(r_x) > \texttt{rank}(r_y)\texttt{:} \\
\quad \pi(r_y) = r_x \\
\texttt{else:} \\
\quad \pi(r_x) = r_y \\
\quad \texttt{if } \texttt{rank}(r_x) = \texttt{rank}(r_y)\texttt{:} \quad \texttt{rank}(r_y) = \texttt{rank}(r_y) + 1
\end{array}
$$

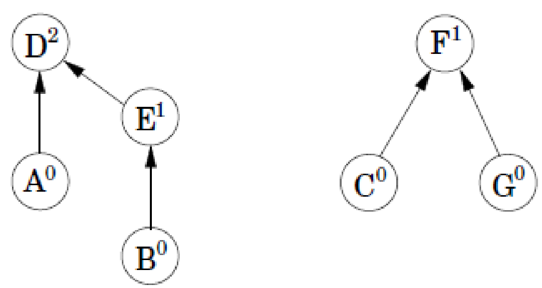
Algorismes greedy

After $\texttt{makeset}(A), \texttt{makeset}(B), \ldots, \texttt{makeset}(G)$:

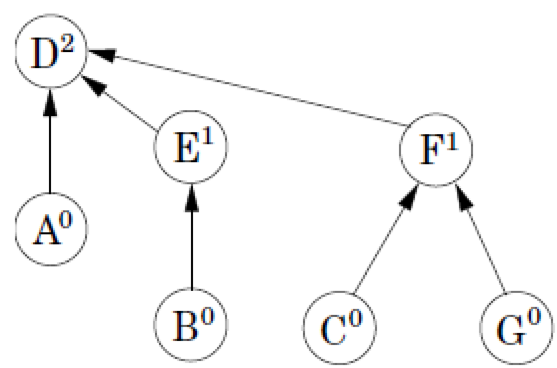
After $\texttt{union}(A, D), \texttt{union}(B, E), \texttt{union}(C, F)$:



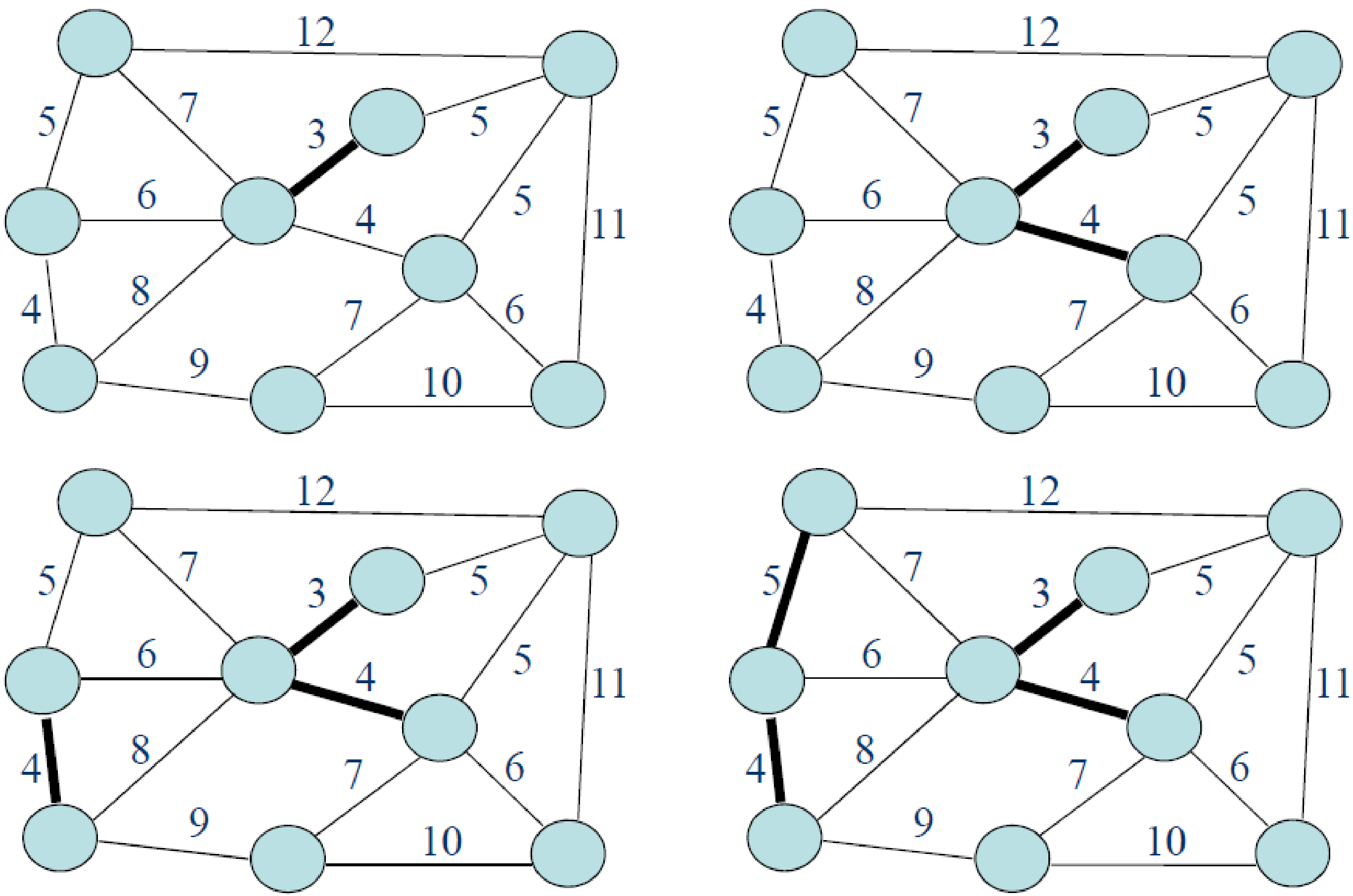
After $\texttt{union}(C, G), \texttt{union}(E, A)$:



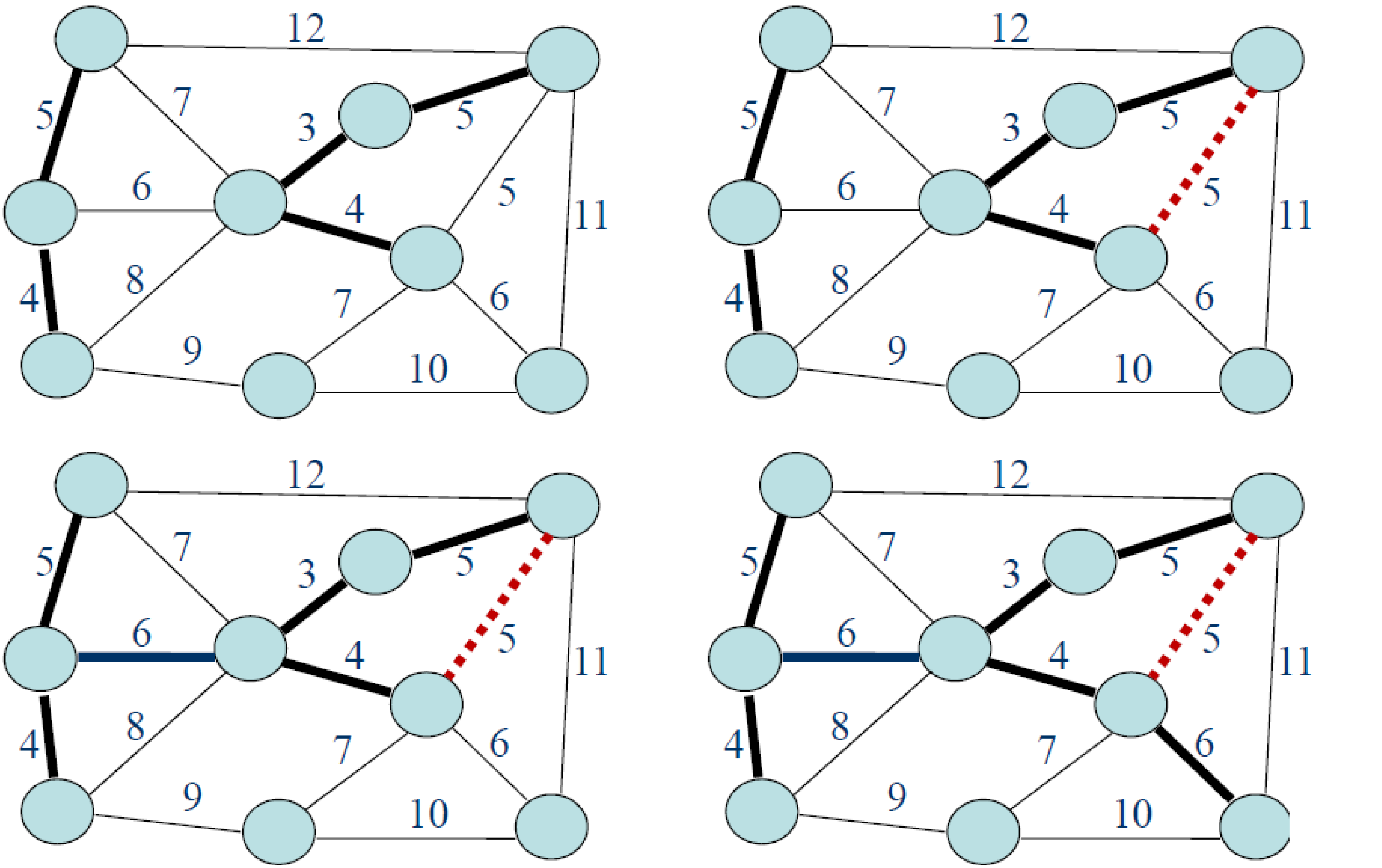
After $\texttt{union}(B, G)$:



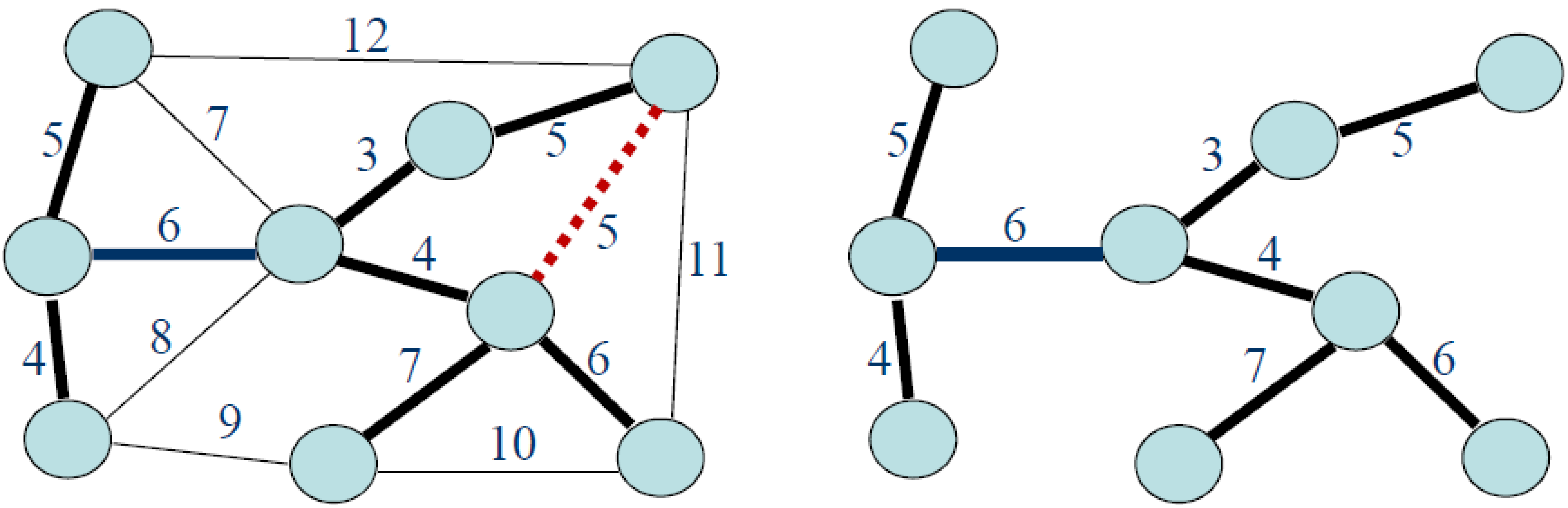
Kruskal exemple



Kruskal exemple



Kruskal exemple



Algorismes greedy

- Exemple: Algorisme de Prim
  - Alternativa a Kruskal

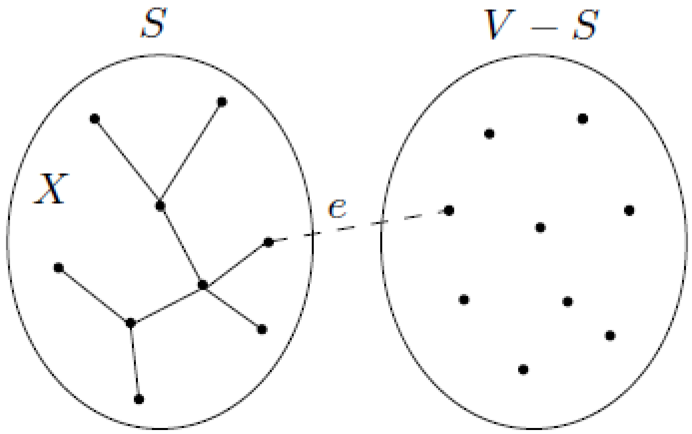
- La propietat de tall ens diu que qualsevol algorisme que segueix el següent procediment hauria de funcionar :

$X = \{\ \}$ (edges picked so far)
repeat until $|X| = |V| - 1$:
  pick a set $S \subset V$ for which X has no edges between S and $V - S$
  let $e \in E$ be the minimum-weight edge between S and $V - S$
  $X = X \cup \{e\}$

Algorismes greedy

- Prim: similar a kruskal però per nodes

$$\texttt{cost}(v) \;=\; \min_{u \in S}\; w(u,v).$$



La complexitat és similar a l'algorisme de kruskal

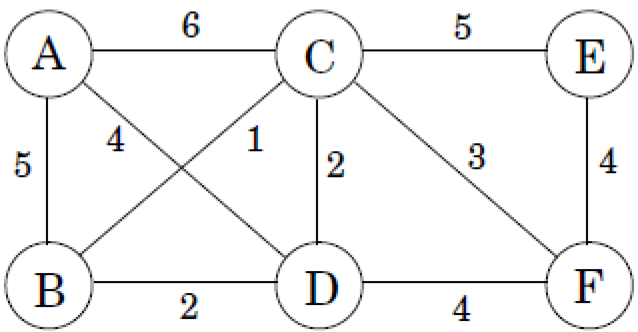
Algorismes greedy

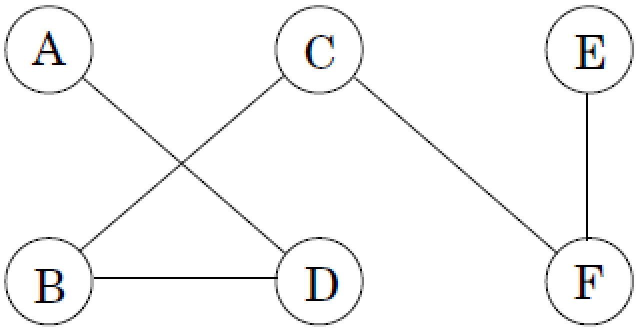
```
procedure prim(G, w)
Input:    A connected undirected graph G = (V, E) with edge weights w_e
Output:   A minimum spanning tree defined by the array prev

for all u ∈ V:
   cost(u) = ∞
   prev(u) = nil
Pick any initial node u_0
cost(u_0) = 0

H = makequeue(V)    (priority queue, using cost-values as keys)
while H is not empty:
   v = deletemin(H)
   for each {v, z} ∈ E:
      if cost(z) > w(v, z):
         cost(z) = w(v, z)
         prev(z) = v
         decreasekey(H, z)
```

Algorismes greedy

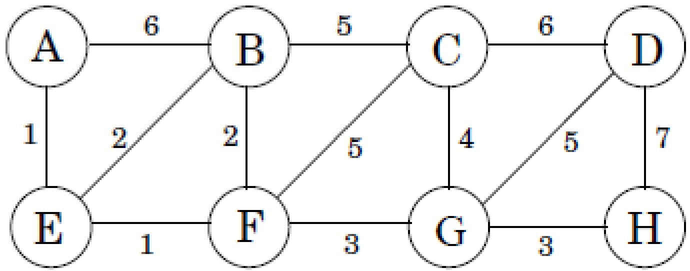




Set S	A	B	C	D	E	F
$\{\}$	$0/\text{nil}$	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
A		$5/A$	$6/A$	$4/A$	∞/nil	∞/nil
A, D		$2/D$	$2/D$		∞/nil	$4/D$
A, D, B			$1/B$		∞/nil	$4/D$
A, D, B, C					$5/C$	$3/C$
A, D, B, C, F					$4/F$	

Algorismes greedy

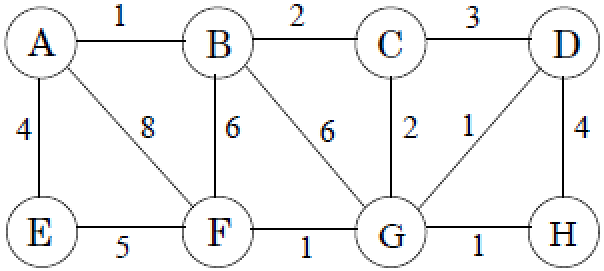
- Exercicis (1):



- A) Quin és el cost del MST?
- B) En quin ordre les arestes són incloses en el MST usant l'algorisme Kruskal?

Algorismes greedy

- Exercicis (2):



- Aplica l'algorisme Prim (order alfabètic)
  - Escriu la taula de costos intermitjos
- Aplica l'algorisme Kruskal i mostra els diferents arbres intermitjos

Algorismes greedy

- En altres casos, els algorismes greedy obtenen respostes aproximades
  - → factor d'aproximació

- No són òptimes, però no existeixen algorismes lineals que solucionen el problema
  - → Ho veurem a problemes NP

Algorismes greedy

Es disposa d'un dispositiu d'accés seqüencial, o cinta de capacitat igual a K. I per altra banda, es disposa d'n arxius de mides $M(i)$, per $i = 1, \ldots, n$. Sabem d'entrada, que la $\sum_{i=1}^{n} M(i)$ és considerablement més gran que K, de manera que per emmagatzemar tots els arxius caldrien varies cintes. I només en tenim una.

Com que el que ens interessa és poder guardar el màxim nombre de fitxers, dissenyeu un algorisme greedy $O(n^2)$, implementant-lo en una funció de capçalera `cintes(K,M)`, que maximitzi el nombre de fitxers que podem emmagatzemar a la cinta.
És òptima la solució que dóna el vostre algorisme?

Fer-ho bé vol dir fer-ho en $\Theta(n \log n)$. Si el vostre algorisme és $\Theta(n^2)$