Algorísmica Avançada
Algorismes greedy I

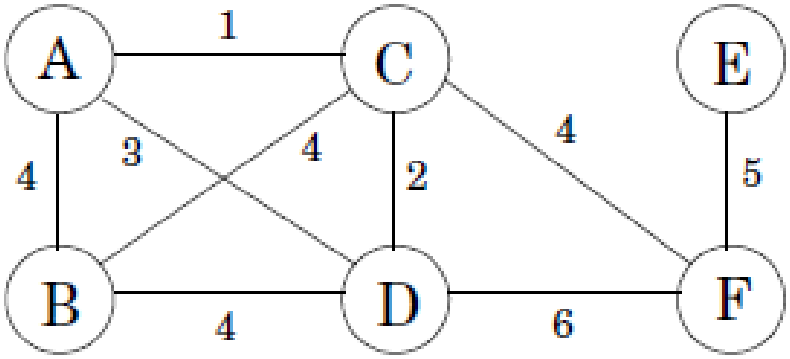
Sergio Escalera

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

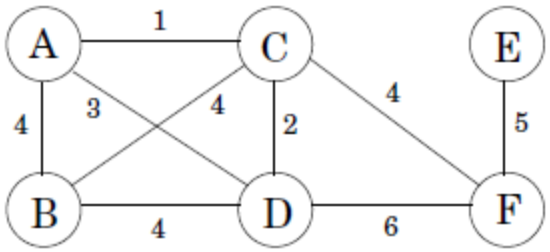
- Exemple

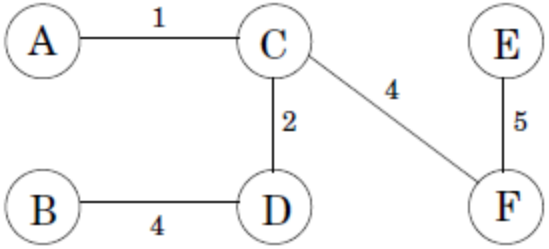


- Volem connectar els ordinadors (nodes) d'una xarxa. Les connexions són les arestes. Cadascuna té un cost. Volem el mínim cost.
  - → llavors no volem cicles
  - → volem un graf no dirigit acíclic connectat
    - → arbre !!!
  - → de mínim cost: **Minimum Spanning Tree (MST)**

Algorismes greedy

- MST amb cost 16 (un dels possibles)



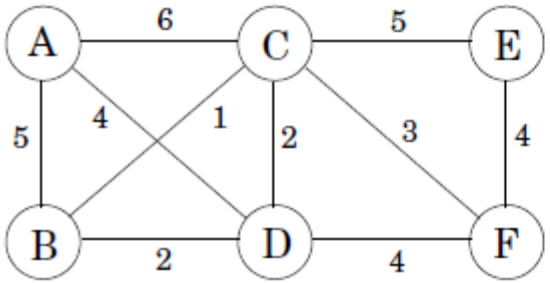


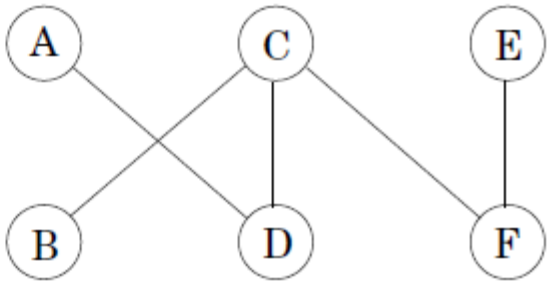
- Algorisme greedy: Kruskal
  - Començar amb arbre buit
  - Mentre no estiguin tots els nodes connectats:
    - Incloure aresta de cost mínim que no produeix un cicle

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?

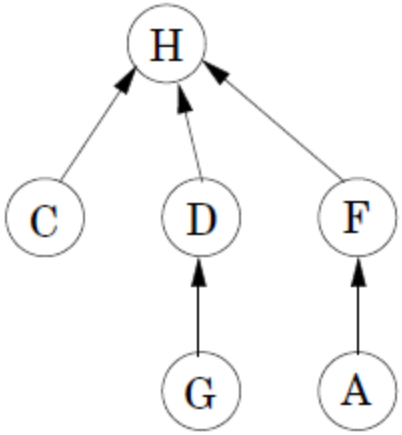
Algorismes greedy

```
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 ∈ V:
    makeset(u)

X = {}
Sort the edges E by weight
for all edges {u, v} ∈ E, in increasing order of weight:
    if find(u) ≠ 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




```
procedure makeset(x)
π(x) = x
rank(x) = 0

function find(x)
while x ≠ π(x): x = π(x)
return x
```

π punter

rank: altura dins de l'arbre

Algorismes greedy

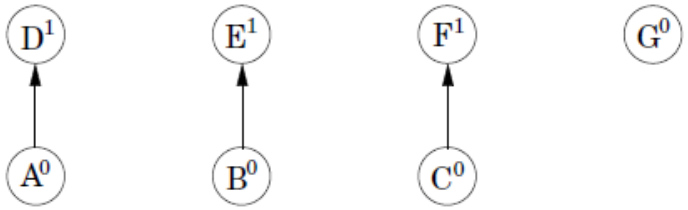
- **Makeset**: temps constant
- **Find**: segueix punters dels pares als roots, per tant el temps és proporcional a l'altura
- **Union**: com l'altura ens defineix la complexitat, posem el punter de l'arbre més curt apuntant al punter de l'arbre amb més altura

```
procedure union(x, y)
r_x = find(x)
r_y = find(y)
if r_x = r_y:   return
if rank(r_x) > rank(r_y):
    π(r_y) = r_x
else:
    π(r_x) = r_y
    if rank(r_x) = rank(r_y):   rank(r_y) = rank(r_y) + 1
```

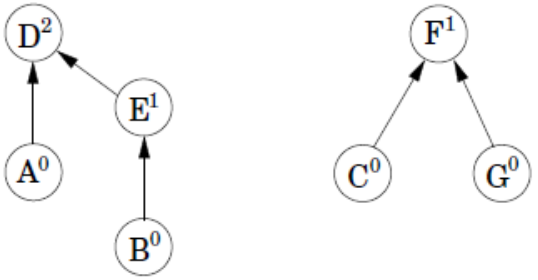
Algorismes greedy

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

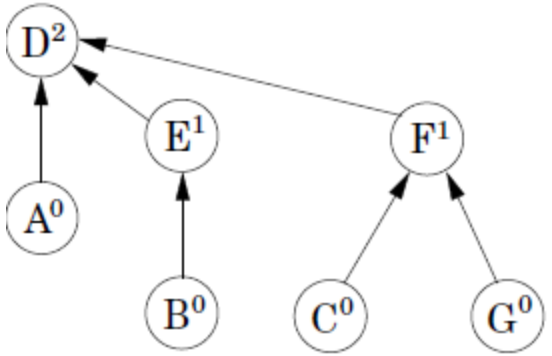
After $\texttt{union}(A, D), \texttt{union}(B, E), \texttt{union}(C, F)$:



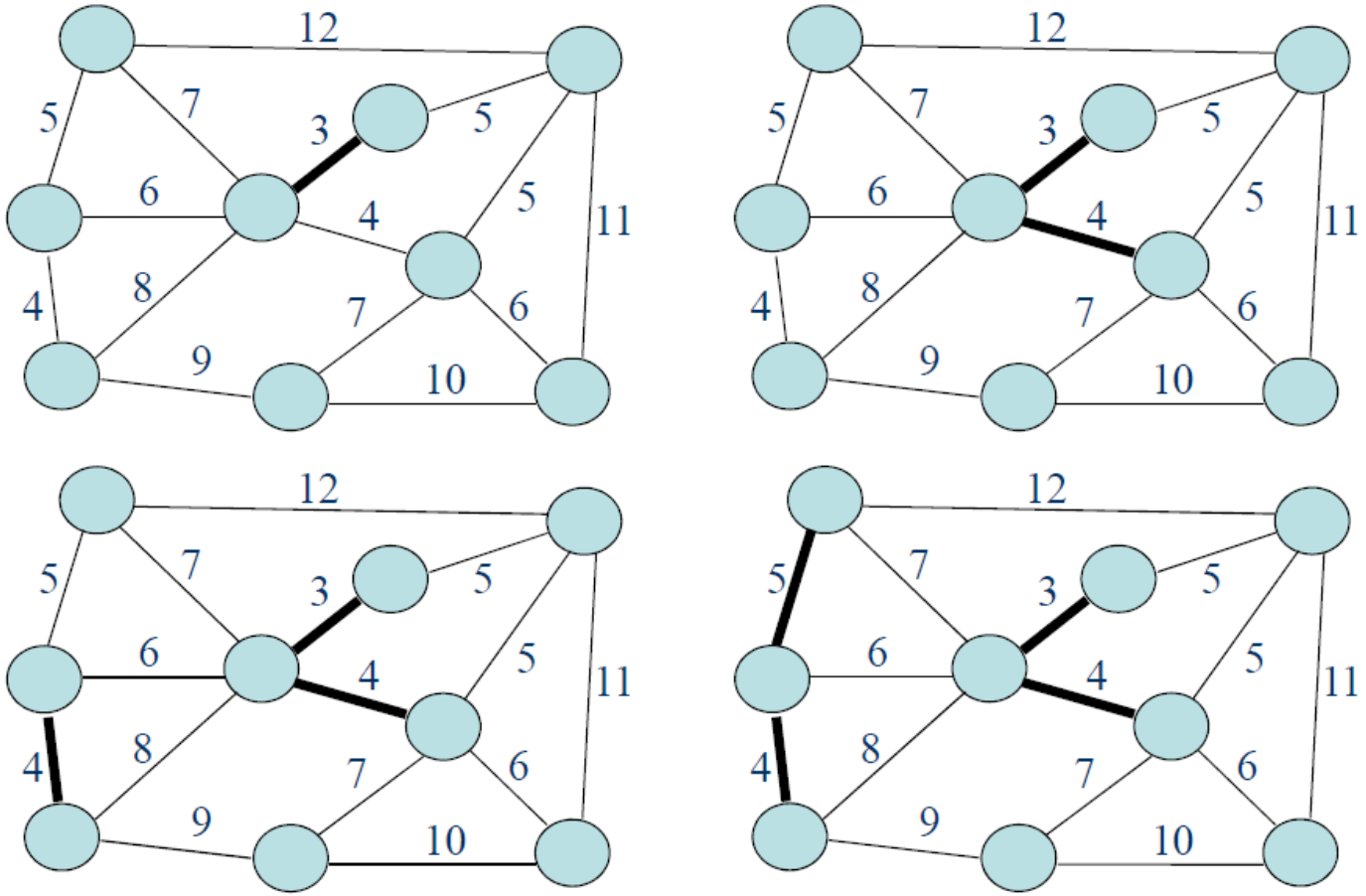
After $\texttt{union}(C, G), \texttt{union}(E, A)$:



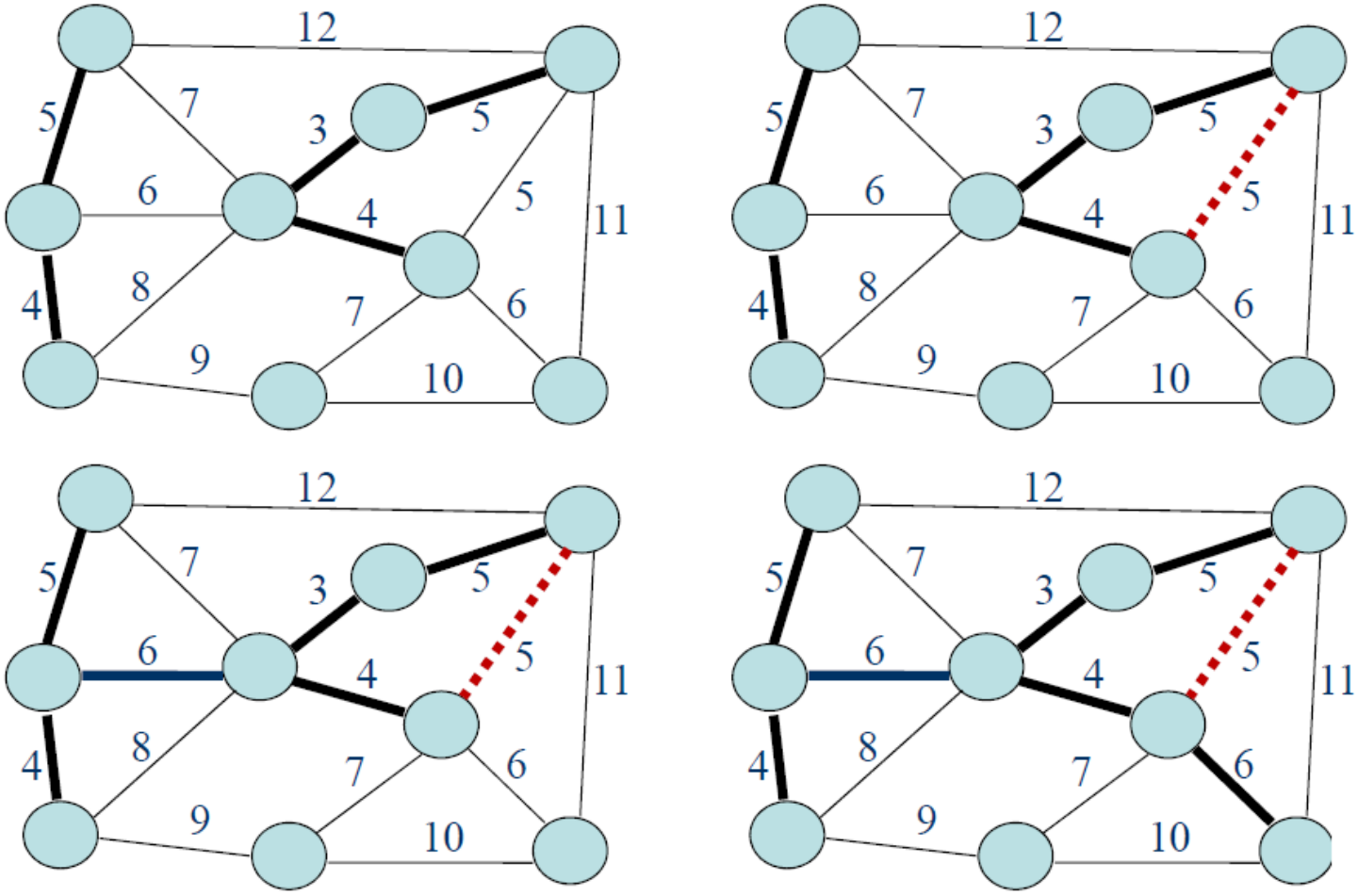
After $\texttt{union}(B, G)$:



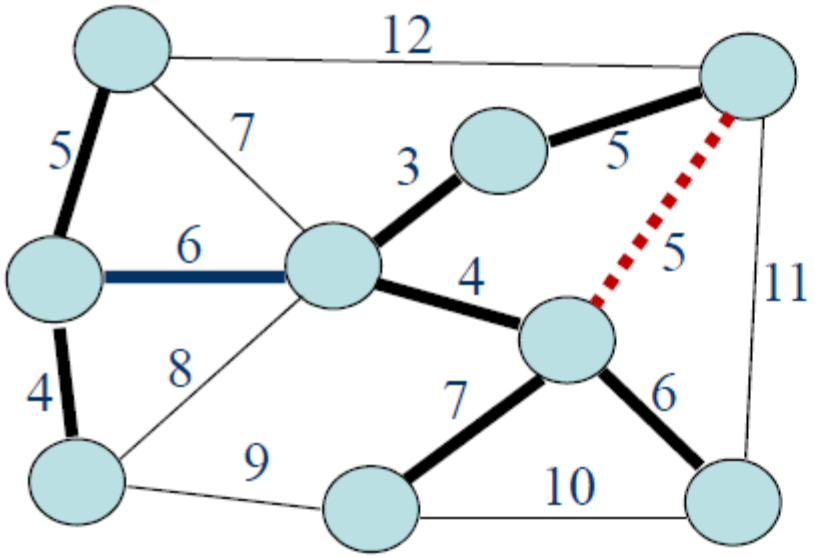
Kruskal exemple

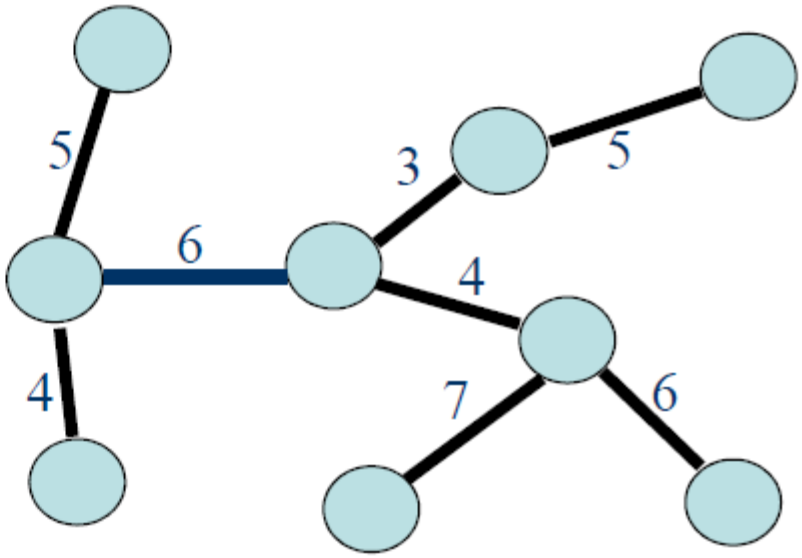


Kruskal exemple



Kruskal exemple





Algorismes greedy

- Exemple: Algorisme de Prim
  - Alternativa a Kruskal

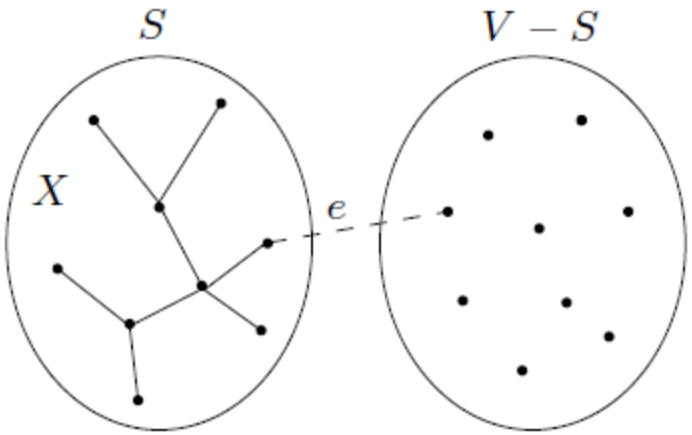
- La propietat de tall ens diu que qualsevol algorisme que segueix el següent procediment hauria de funcionar :

$X = \{\ \}$ (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

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

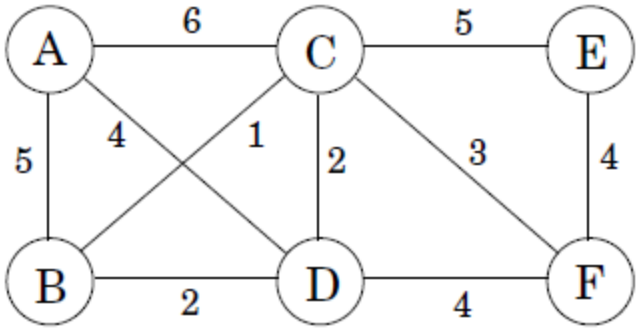


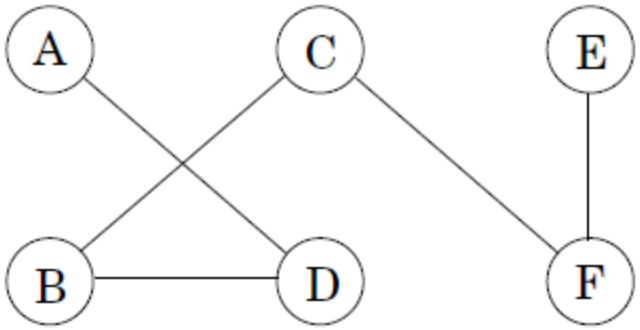
La complexitat és similar a l'algorisme de kruskal

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)
```

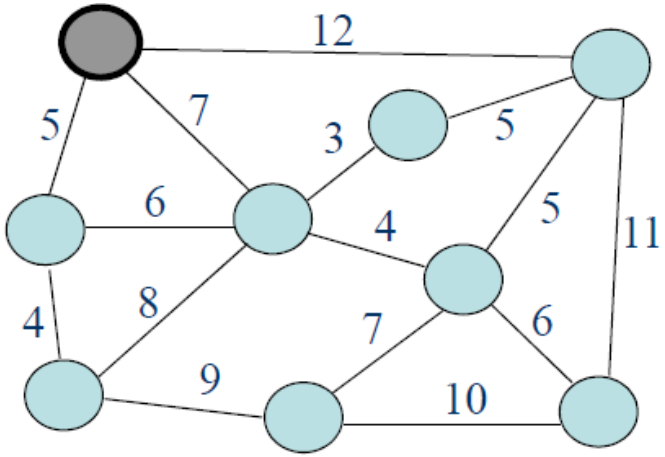
Algorismes greedy

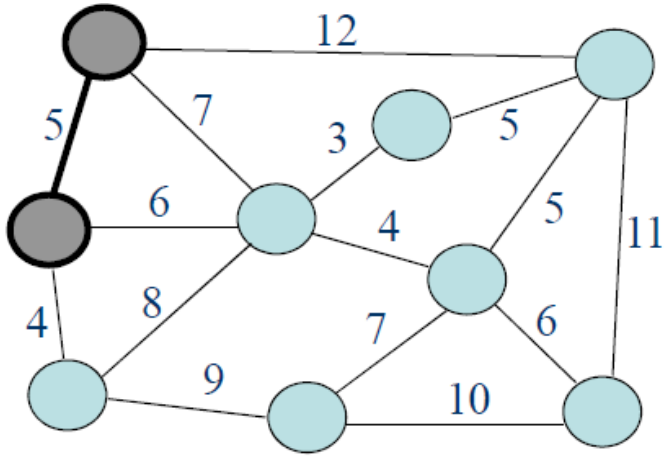


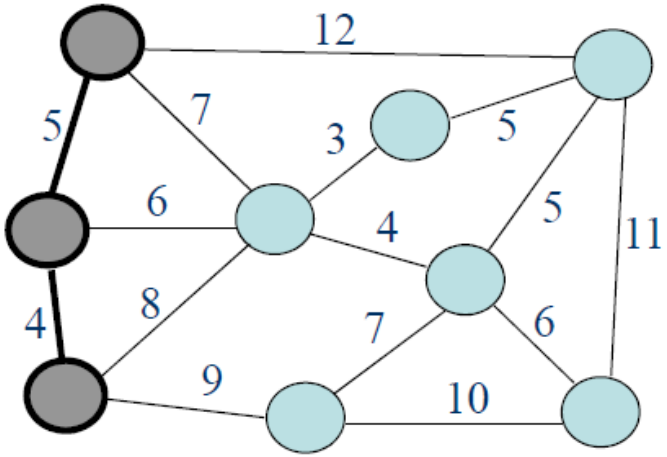


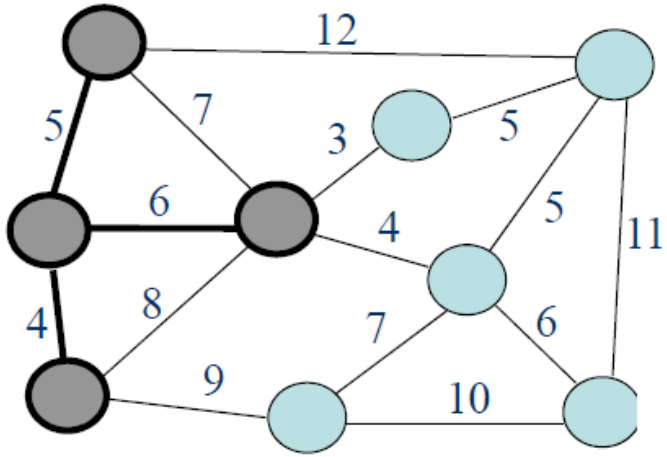
Set S	A	B	C	D	E	F
$\{\}$	0/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
A		$5/A$	$6/A$	$4/A$	∞/nil	∞/nil
A, D		$2/D$	$2/D$		∞/nil	$4/D$
A, D, B			$1/B$		∞/nil	$4/D$
A, D, B, C					$5/C$	$3/C$
A, D, B, C, F					$4/F$	

Prim exemple

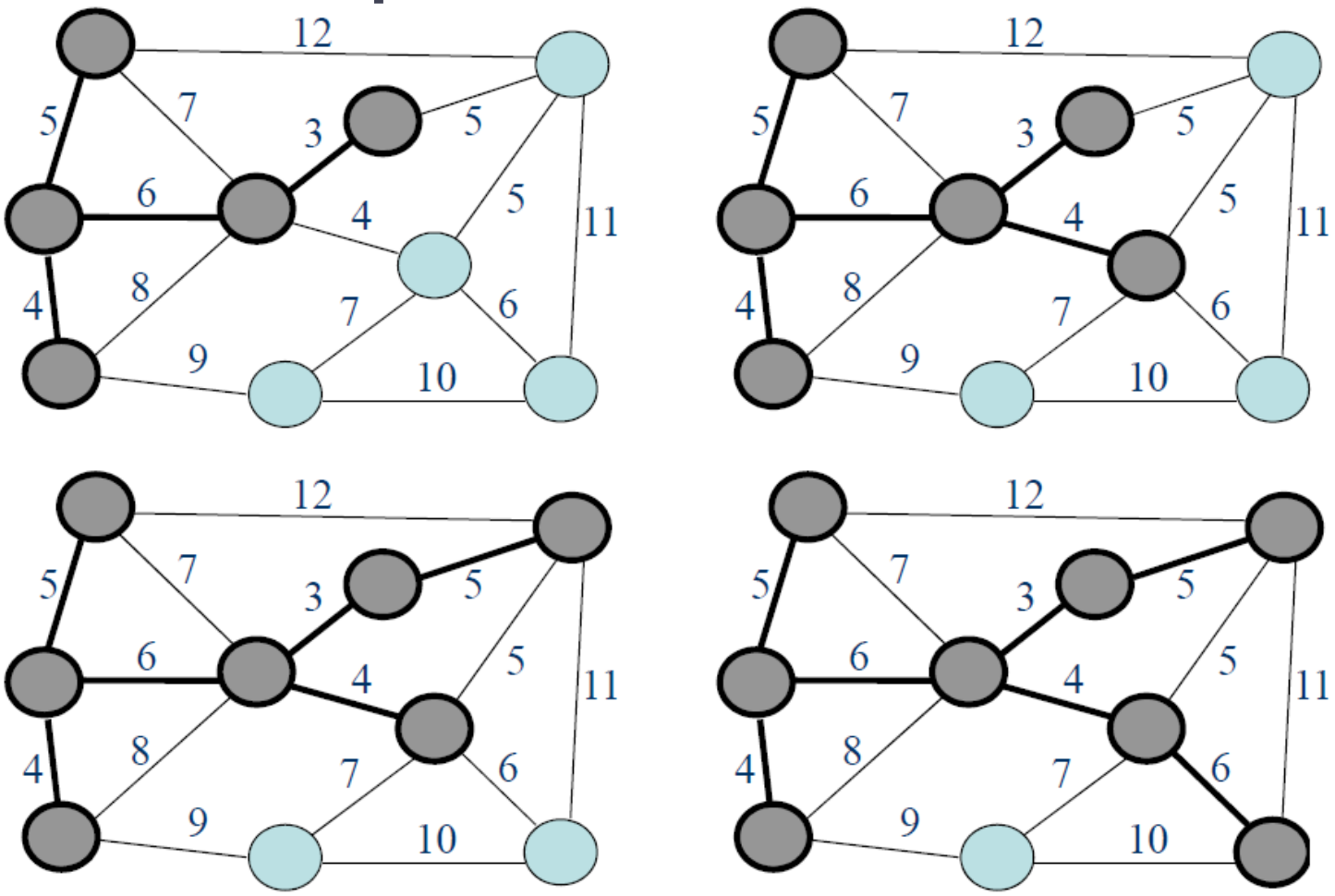




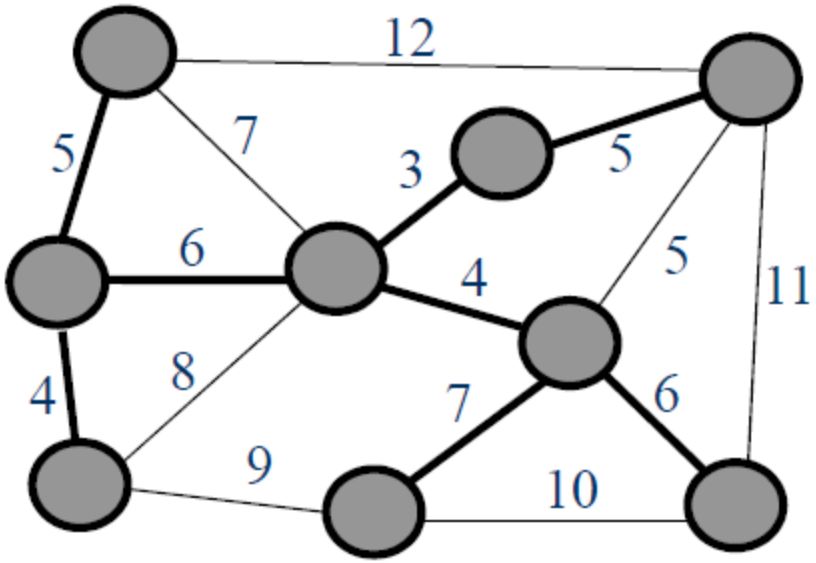


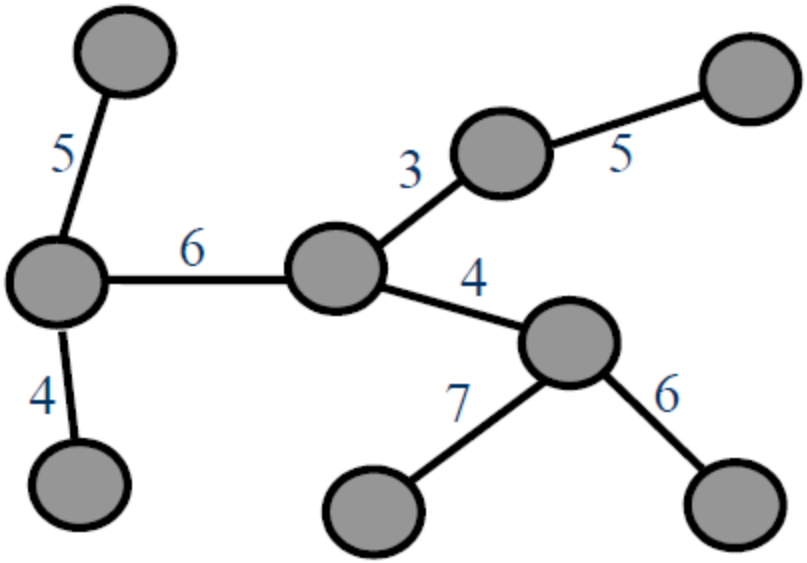


Prim exemple



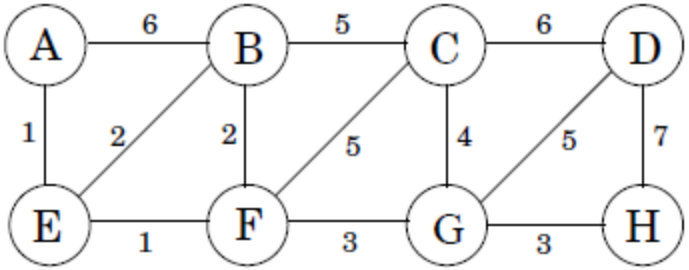
Prim exemple





Algorismes greedy

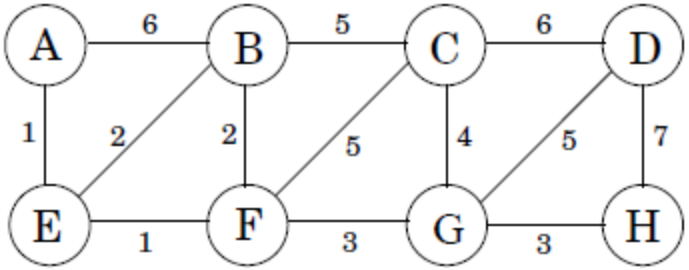
- Exercicis (1):



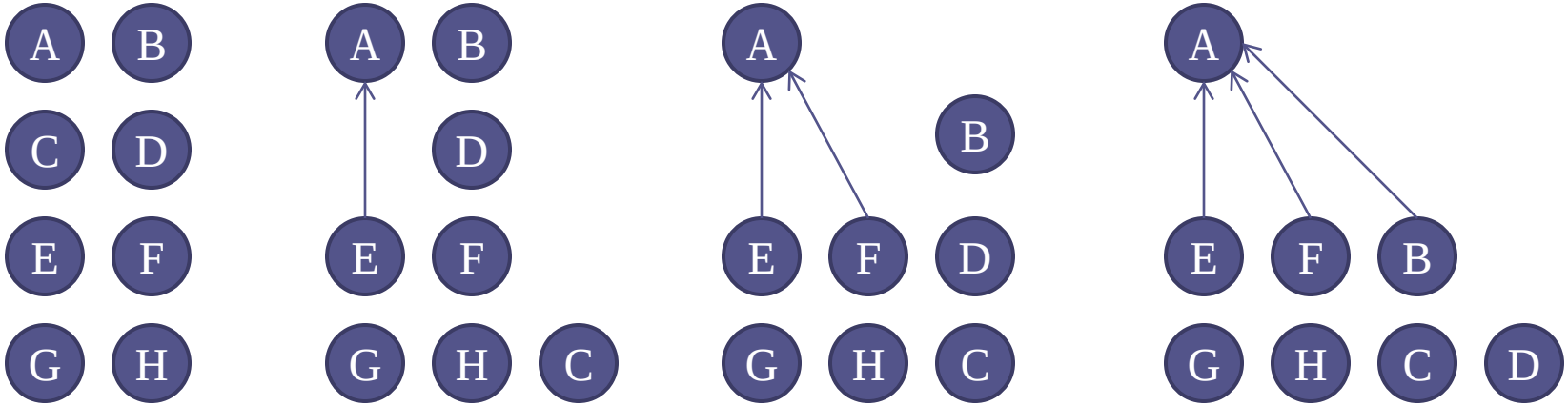
- A) Quin és el cost del MST?
- B) En quin ordre les arestes són incloses en el MST usant l'algorisme Kruskal?

Algorismes greedy

- Exercicis (1):

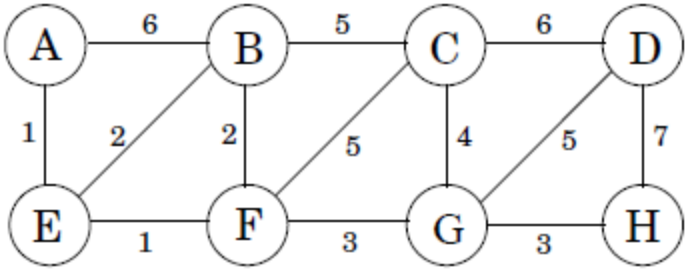


(A,E) (E,F) (E,B)

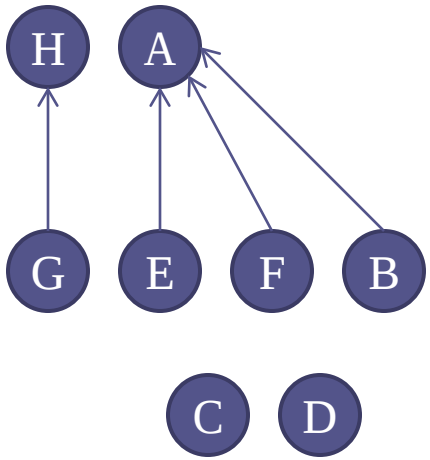


Algorismes greedy

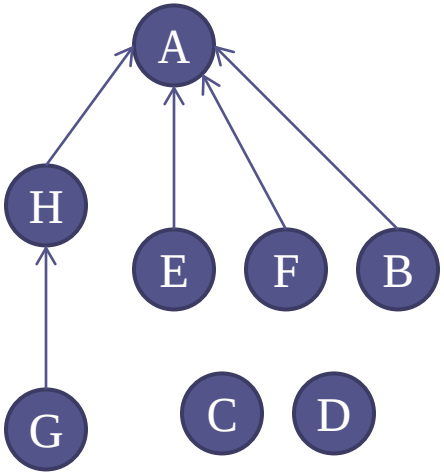
- Exercicis (1):



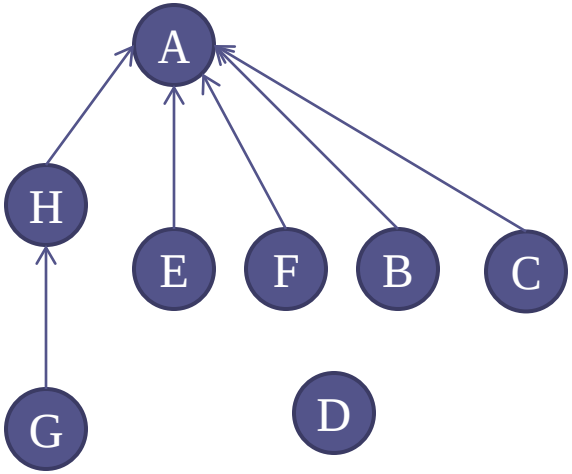
(B,F) → NO
(G,H)



(F,G)

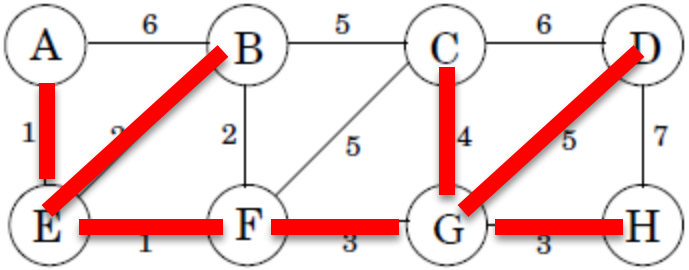


(G,C)



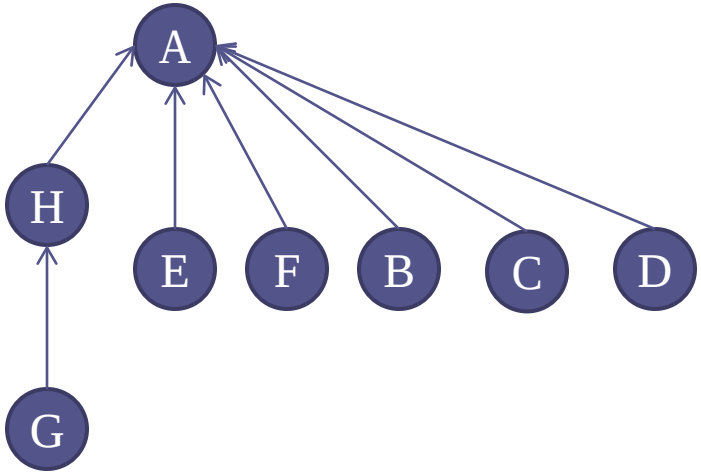
Algorismes greedy

- Exercicis (1):



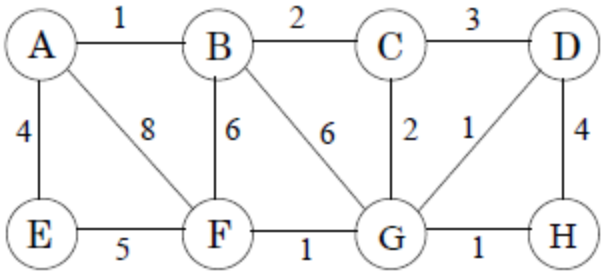
MST coste 19

(G,D)



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

Algorísmica Avançada
Algorismes greedy II

Sergio Escalera

Algorismes greedy

- Exemple: codificació Huffman

Imagina un text de quatre simbols codificat amb la següent taula de freqüències.

Symbol	Frequency
A	70 million
B	3 million
C	20 million
D	37 million

En lloc de fer servir 2 bits per cadascun, podem pensar en una altra codificació?:

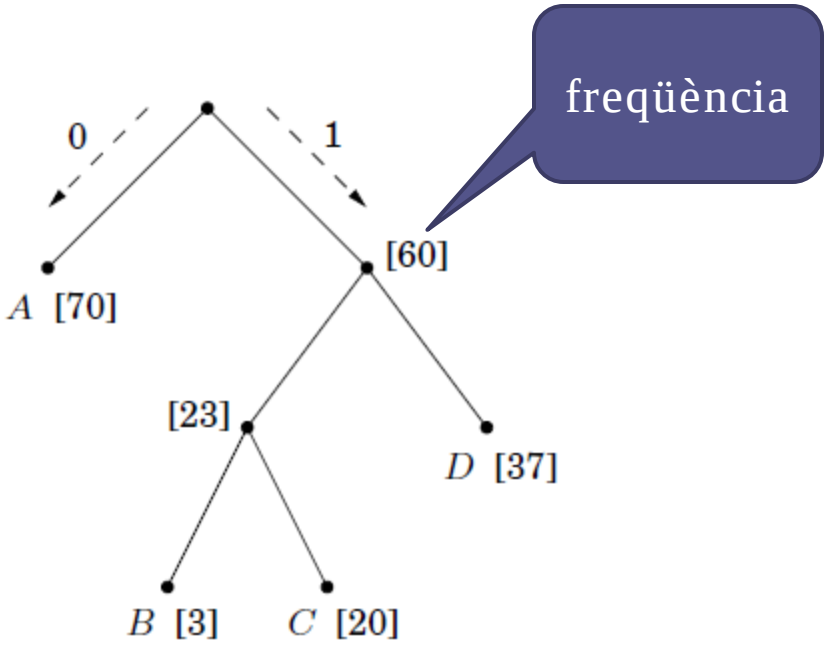
$$\{0, 01, 11, 001\}$$

La decodificació de strings com 001 és ambígua
→ podria ser també 0 i 01 !!

Algorismes greedy

- Fem un arbre lliure de prefixe!

Symbol	Codeword
A	0
B	100
C	101
D	11



- Hem aconseguit un 17% de millora en espai!

Algorismes greedy

- Per trobar aquests arbres volem minimitzar la següent funció de cost:

$$\text{cost of tree} = \sum_{i=1}^{n} f_i \cdot (\text{depth of } i\text{th symbol in tree})$$

- Tornem a un algorisme greedy!!

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

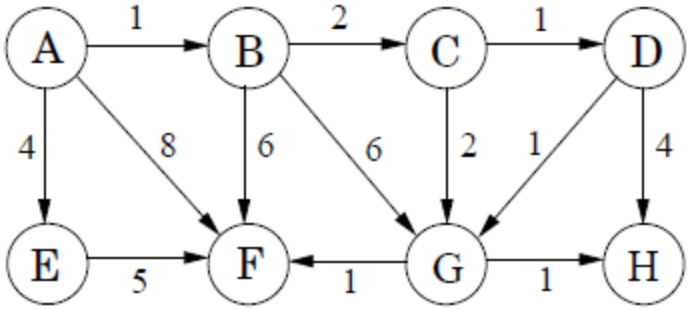
- Exercici:
  - Freqüència de lletres de l'alfabet anglès

blank	18.3%	r	4.8%	y	1.6%
e	10.2%	d	3.5%	p	1.6%
t	7.7%	l	3.4%	b	1.3%
a	6.8%	c	2.6%	v	0.9%
o	5.9%	u	2.4%	k	0.6%
i	5.8%	m	2.1%	j	0.2%
n	5.5%	w	1.9%	x	0.2%
s	5.1%	f	1.8%	q	0.1%
h	4.9%	g	1.7%	z	0.1%

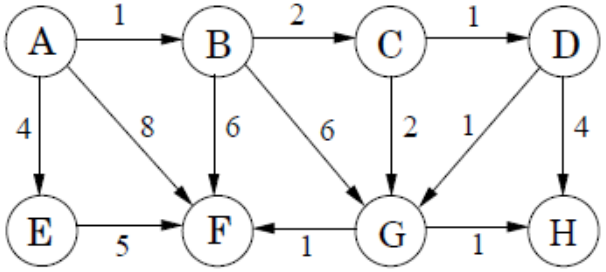
- Quina es la codificació òptima de Huffman?

Algorismes sobre grafs

- Exercici: **Dijkstra**
- Començant a A: dibuixa la taula de distàncies immediates a tots els nodes a cada iteració.
- Mostra l'arbre de camins mínims

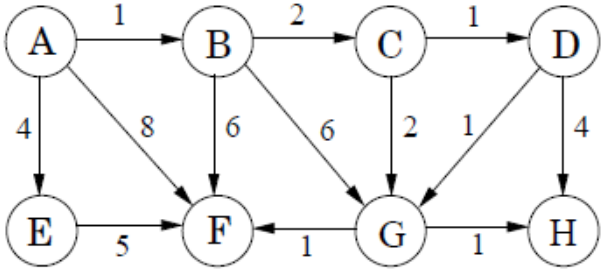


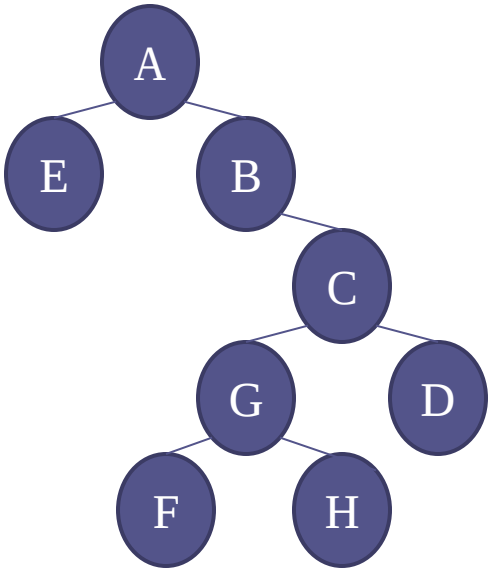
Algorismes sobre grafs



	A	B	C	D	E	F	G	H	S
1									
2									
3									
4									
5									
6									
7									
8									

Algorismes sobre grafs





	A	B	C	D	E	F	G	H	S
1	0 -	∞ -	∞ -	∞ -	∞ -	∞ -	∞ -	∞ -	[A]
2	0 -	1 A	∞ -	∞ -	4 A	8 A	∞ -	∞ -	[AB]
3	0 -	1 A	3 B	∞ -	4 A	7 B	7 B	∞ -	[A..C]
4	0 -	1 A	3 B	4 C	4 A	7 B	5 C	∞ -	[A..D]
5	0 -	1 A	3 B	4 C	4 A	7 B	5 C	8 D	[A..E]
6	0 -	1 A	3 B	4 C	4 A	7 B	5 C	8 D	[E..EG]
7	0 -	1 A	3 B	4 C	4 A	6 G	5 C	6 G	[A..G]
8	0 -	1 A	3 B	4 C	4 A	6 G	5 C	6 G	[A..H]

Algorismes sobre grafs

- Exercici: el mateix amb **Bellman-Ford**