Algorísmica Avançada
Repàs i Grafs I

Sergio Escalera
Miguel Reyes

Introducció a l'Algorísmica Avançada

- Repàs
  - Algorísmica:
    - Algorismes, notació i complexitat
    - Força bruta
    - Cerca
  - Estructura de dades
    - Arbres
    - Hashing
- Grafs

Algorismes, notació i complexitat

Què és algorisme?

Un algorisme és una seqüència finita, no ambigua i explícita, d'instruccions per a resoldre un problema. (*Wikipedia)*

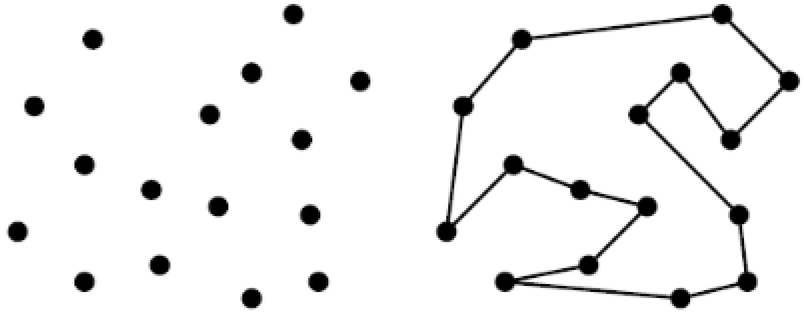
Definició II: Un algorisme és qualsevol procediment computacional que pren un (o una sèrie) de valors com a entrada (*input) i genera* algun valor (o conjunt de valors) com a sortida (*output)*.

Un algorisme és **correcte si podem demostrar que** retorna la sortida desitjada per a qualsevol entrada legal

És **eficient si es fa amb el mínim nombre de recursos** (**temps, memòria) possible.**

Algorismes, notació i complexitat

Problema I: Suposem que hem de passar per cada un d'aquests punts i volem minimitzar la distància recorreguda.



Algorismes, notació i complexitat

Solució I: Escollim un punt aleatori, i anem seleccionant el veí més proper per continuar.

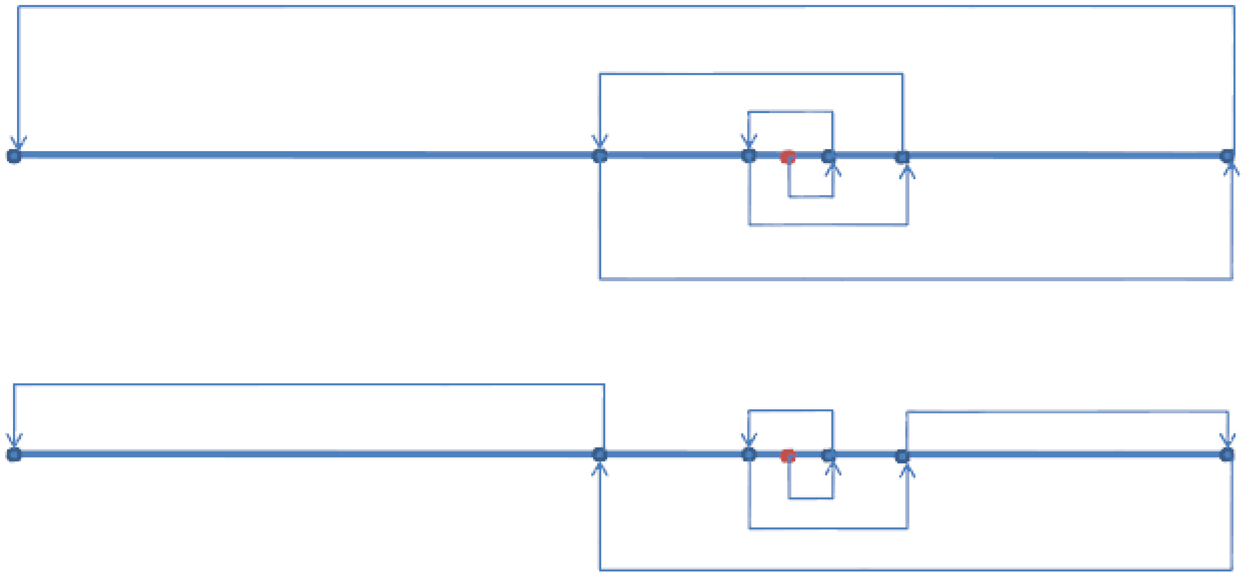
$p=p_0$
$i=0$
Mentre hi hagi punts per visitar
    $i = i + 1$
    Determinem p_i, el punt més proper a p_{i-1}
    Visitem p_i
Retorna la seqüència

Identació

Algorisme especificat en **pseudocodi**

Algorismes, notació i complexitat

És evident que no és correcte!



Algorismes, notació i complexitat

Solució II: Considerem totes les possibles ordenacions dels punts i seleccionem la més curta.

$d=\infty$
Per cada una de les $n!$ permutacions P_i dels n punts
   Si ($cost(P_i)$ <= d) llavors $d= cost(P_i)$ i $P_{min}=P_i$
Retorna P_{min}

És correcte!, però és eficient?

Algorismes, notació i complexitat

Notació per definir l'eficiència (complexitat dels algorismes)

Definición $O(g(n))$ *"o de g de n". Dadas $f = f(n)$ y $g = g(n)$, se dice que la función f pertenece al conjunto $O(g)$ si cuando n se hace muy grande, f no crece más rapidamente que g.*

$$f \in O(g) \Leftrightarrow \lim_{n \to \infty} f/g < \infty.$$

Definición $\Theta(g(n))$ *"zeta de g de n". Dadas $f = f(n)$ y $g = g(n)$, se dice que la función f pertenece al conjunto $\Theta(g)$ si cuando n se hace muy grande, f crece como g.*

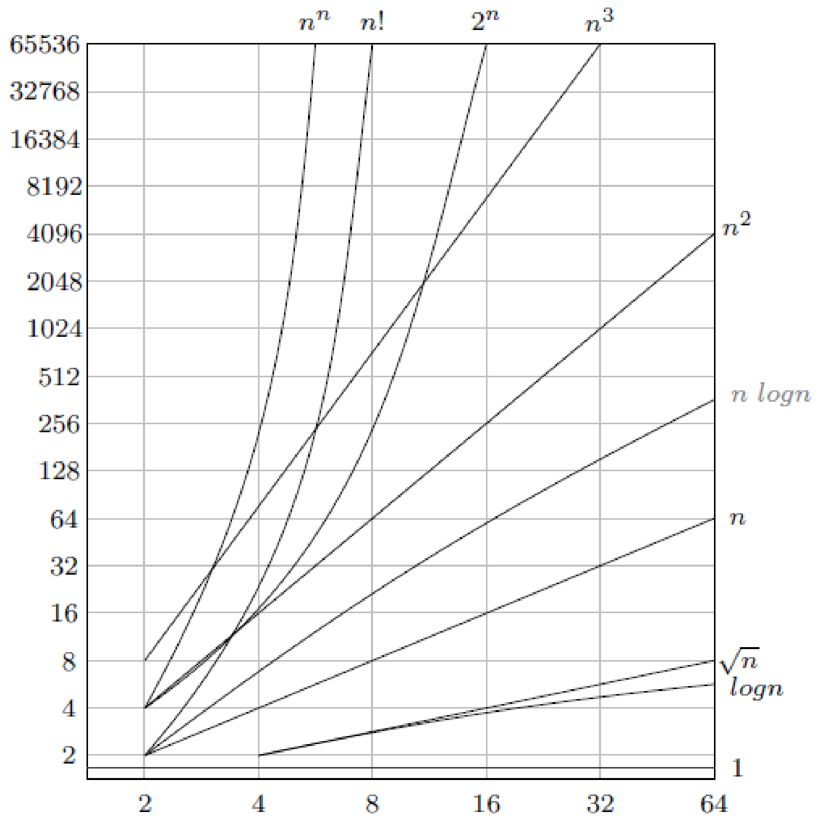
$$f \in \Theta(g) \Leftrightarrow \lim_{n \to \infty} f/g = c, \quad c \neq 0, c \in \mathbb{R}.$$

Definición $\Omega(g(n))$ *"omega de g de n". Dadas $f = f(n)$ y $g = g(n)$, se dice que la función f pertenece al conjunto $\Omega(g)$ si cuando n se hace muy grande, f no crece más lentamente que g.*

$$f \in \Omega(g) \Leftrightarrow \lim_{n \to \infty} f/g > 0$$

Algorismes, notació i complexitat

Notació per definir l'eficiència (complexitat dels algorismes)



Algorismes, notació i complexitat

Notació per definir l’eficiència (complexitat dels algorismes)

$$factorial(n) = \prod_{i=1}^{n} i$$

```
int factorial(int n)
{
    if (n≤ 1) return 1;
    return n * factorial(n-1);
}
```

“Temps”

$$T(n) = \begin{cases} \Theta(1) & si \;\; n \leq 1 \\ T(n-1) + \Theta(1) & si \;\; n > 1 \end{cases}$$

Algorismes, notació i complexitat

Notació per definir l'eficiència (complexitat dels algorismes)

$$fibonaci(n+1) = 1 + \sum_{i=1}^{n-1} fibonaci(i).$$

```
int fibonacci(int n)
{
    if (n<=2) return 1;
    return fibonaci(n-1) + fibonaci(n-2);
}
```

$$T(n) = \begin{cases} \Theta(1) & si\ n \leq 2 \\ T(n-1) + T(n-2) + \Theta(1) & si\ n > 2 \end{cases}$$

Força bruta

Cerca exhaustiva:

“***No té una solució exacta més eficient que la cerca exhaustiva: no es coneix cap algorisme exacte en temps polinòmic. D’això en diem problemes NP-hard.***”

Ho tornarem a parlar a complexitat

Cerca

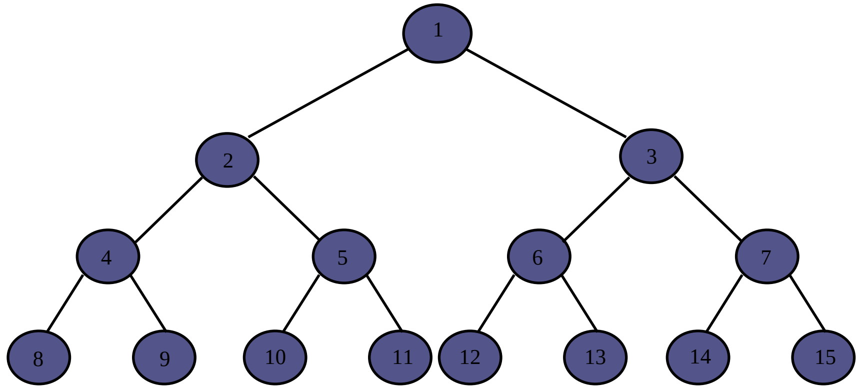
Podem fer ús dels principis d'ordenació

Cerca exhaustiva

Cerca aproximada:

- Aleatòria
- Cerca amb algorismes genètics: adaptació!

Estructura de dades - arbres



Arbre binari

Conjunt de **nodes i arestes**, on el node superior es diu el **root** i els nodes externs es diuen **fulles**

En el cas dels **arbres binaris** cada node te com a **màxim 2 arestes sortints**

Un arbre binari, al **nivell *i* té com a màxim 2^i nodes** (suposant el root i=0)

Cerca sobre arbres binaris ordenats → complexitat O(*log n*)
Recordeu els algorismes de dividir i vèncer!

Estructura de dades - arbres

Cóm podem recórrer un arbre binari?

Si denominem amb

- L: moviment a l’esquerre
- V: “visitar” el node
- R: moviment a la dreta

Tenim sis combinacions possibles de recorregut:

- LVR, LRV, VLR, VRL, RVL, RLV

Si optem per realitzar primer el moviment a l’esquerra, tenim les tres primeres possibilitats

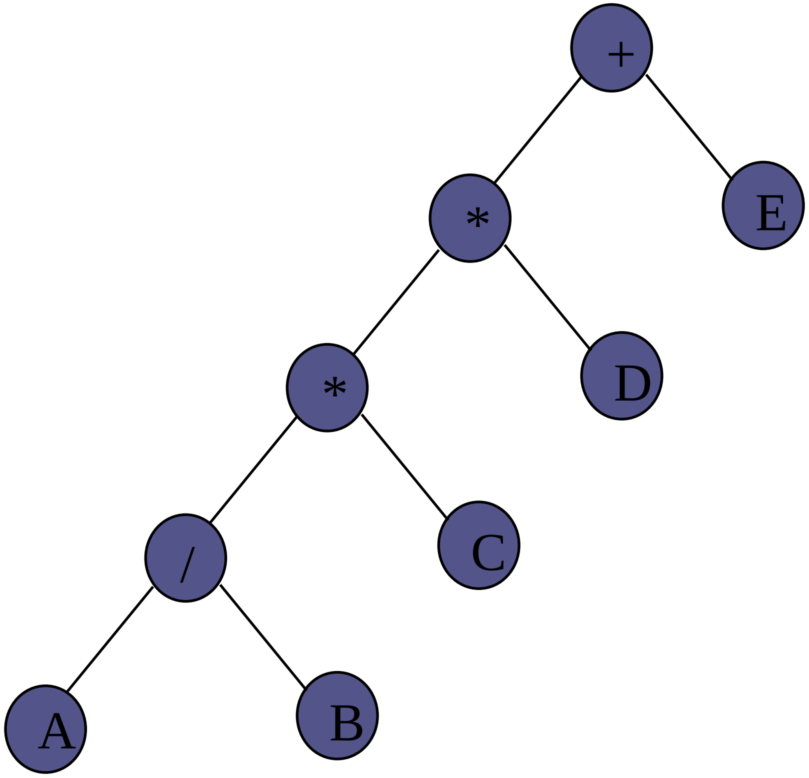
- LVR denominarem inordre
- LRV denominarem postordre, i
- VLR denominarem preordre

Estructura de dades - arbres

- Els recorreguts es corresponen amb les formes infixa, postfixa i prefixa d'escriure una expressió aritmètica en un arbre binari.

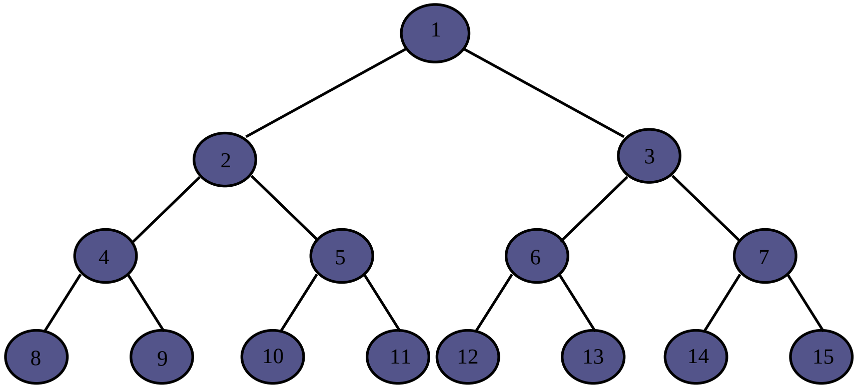
- Donat l'arbre binari, els diferents recorreguts porten a les següents formes d'escriure l'expressió:
  - Inordre: A/B*C*D+E
  - Preordre: +**/ABCDE
  - Postordre: AB/C*D*E+

- Potser no volem analitzar tots els nodes:
  - → Ho veurem a algorismes enumeratius



Estructura de dades - arbres

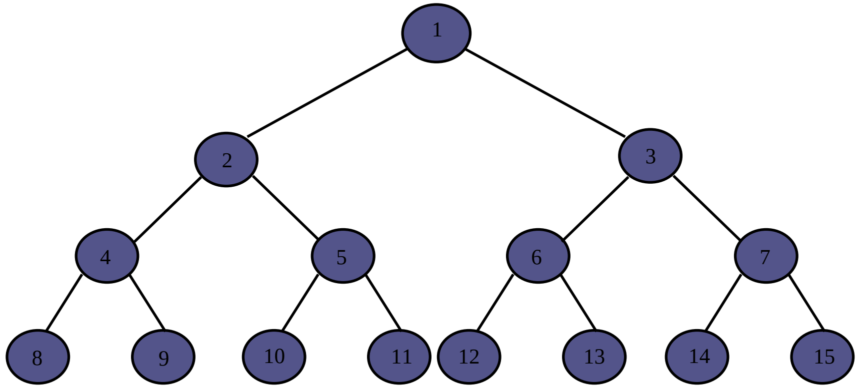
Exercicis



¿Quina és la seqüència de números segons el recorregut inordre (LVR)?

Estructura de dades - arbres

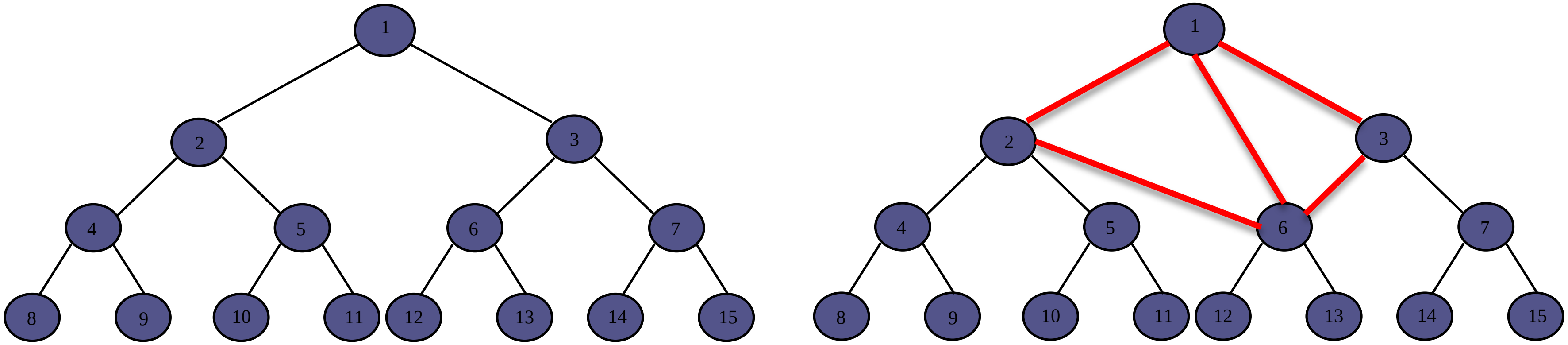
Exercicis



¿Quina és la seqüència de números segons el recorregut postordre (LRV)?

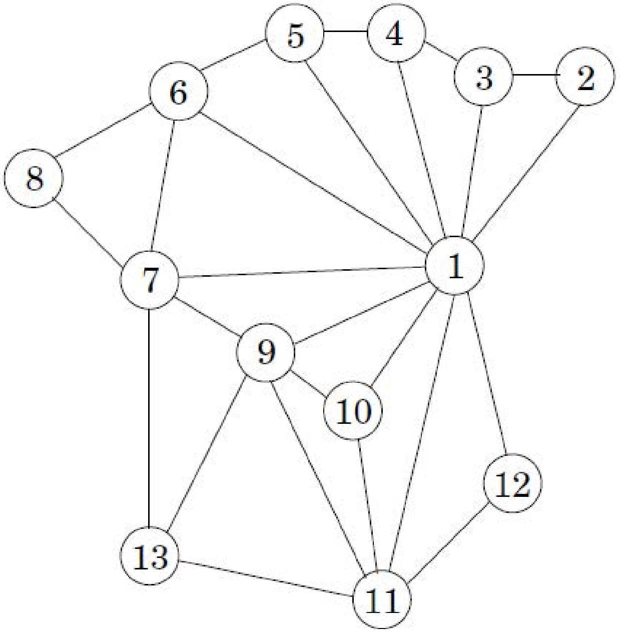
Estructura de dades - grafs

→**Eliminem la restricció del número d'arestes que surten de cada node**
→Ara **poden existir cicles**
→Tenim una representació d'un **graf**
→Tot arbre és un graf, no tot graf es un arbre (ho veurem)



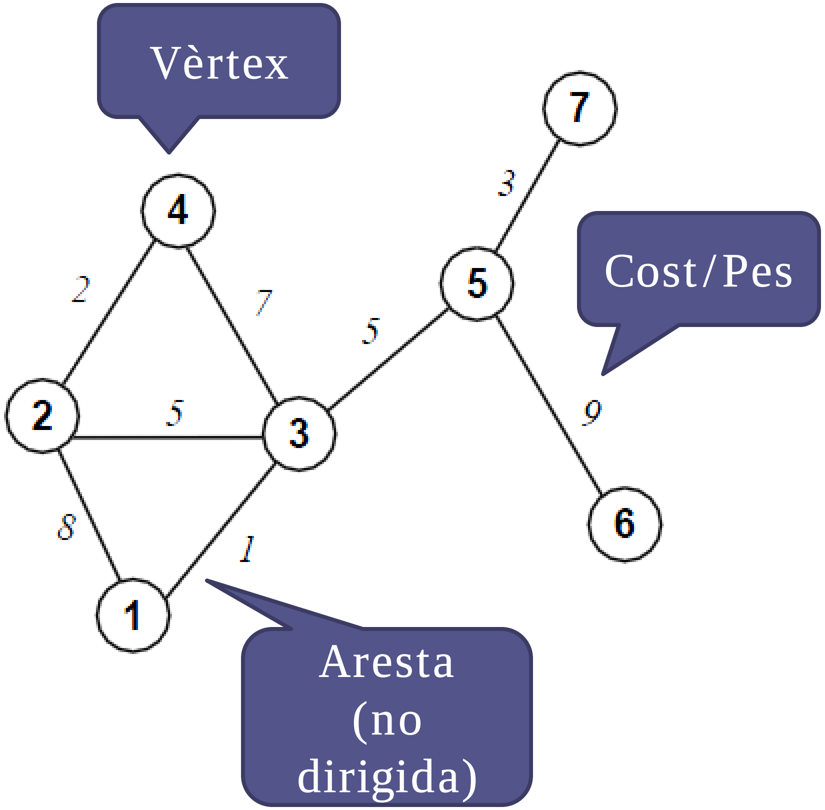
Algorismes sobre grafs

- Per a què serveix un graf?



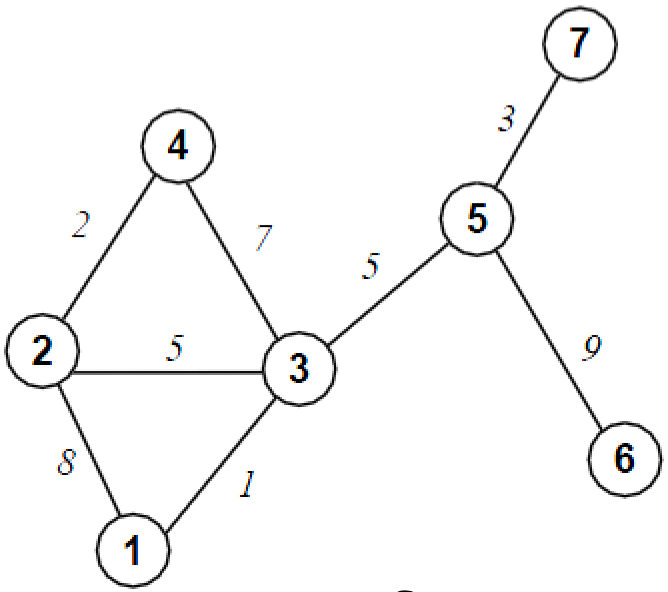
Algorismes sobre grafs

- Com representem un graf? $G=(V,E)$



Algorismes sobre grafs

- Com representem un graf? **Estructura de graf**
- Matriu d'adjacència



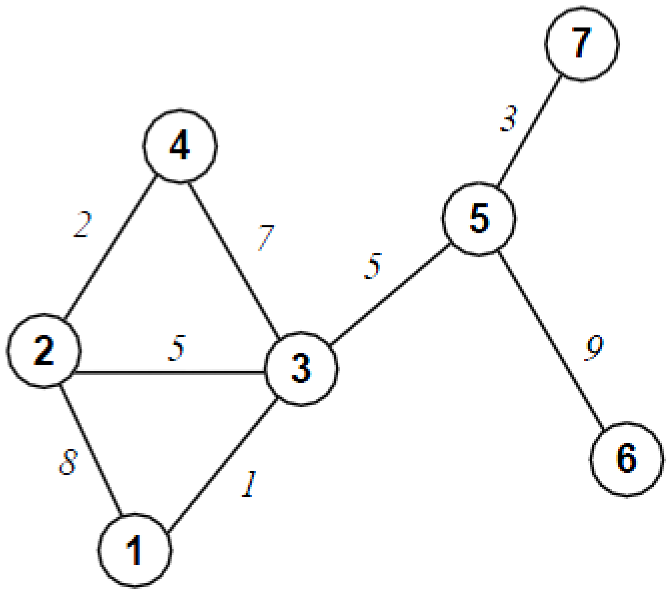
	1	2	3	4	5	6	7
1	0	8	1	∞	∞	∞	∞
2	8	0	5	2	∞	∞	∞
3	1	5	0	7	5	∞	∞
4	∞	2	7	0	∞	∞	∞
5	∞	∞	5	∞	0	9	3
6	∞	∞	∞	∞	9	0	∞
7	∞	∞	∞	∞	3	∞	0

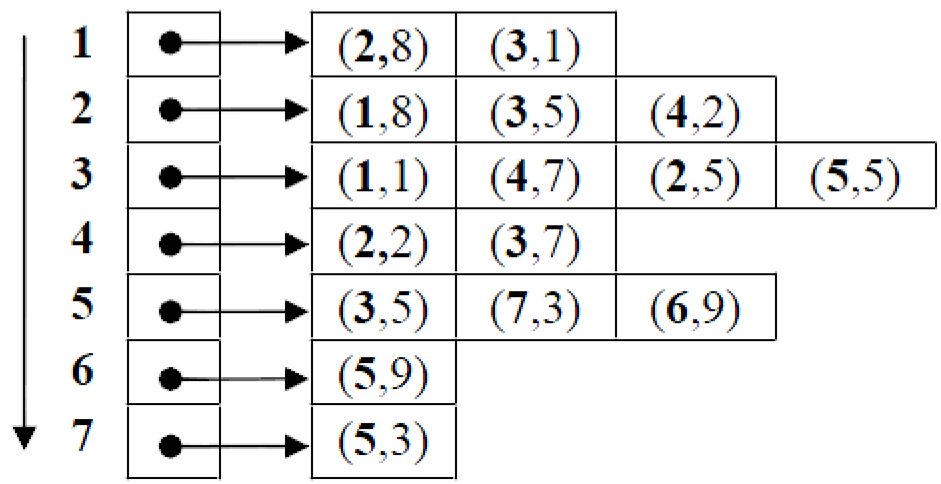
Sense pesos →

$$a_{ij} = \begin{cases} 1 & \text{if there is an edge from } v_i \text{ to } v_j \\ 0 & \text{otherwise.} \end{cases}$$

Algorismes sobre grafs

- Com representem un graf? **Estructura de graf**
- Llista d’adjacència



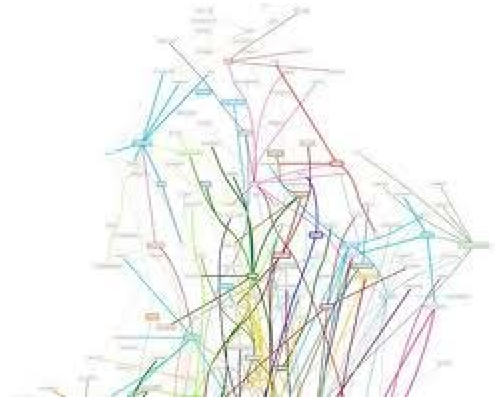


Algorismes sobre grafs

- Matriu ó llista d'adjacència?
- Matriu → $|V|^2$ posicions → un accés
- Llista → $|E|$ posicions → mínim un accés

memòria versus localització

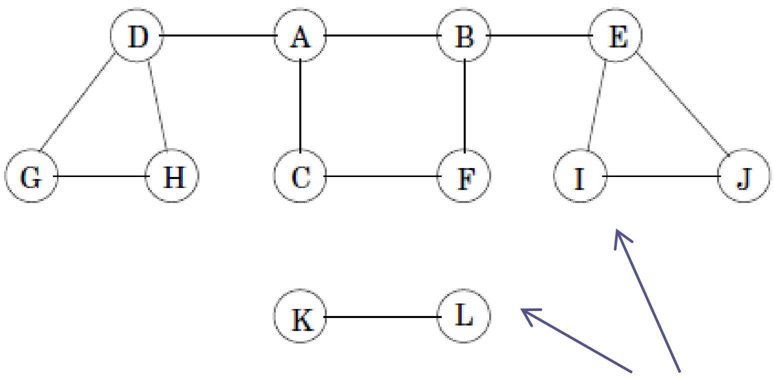
Graf **dens** versus graf **sparse**



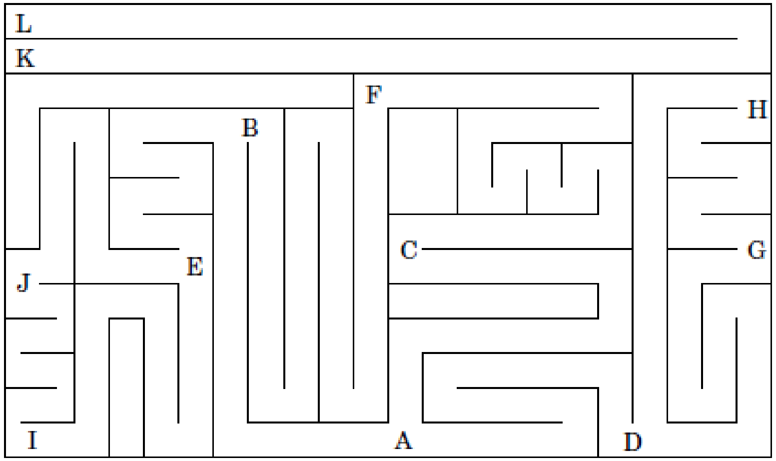
¿què faríeu servir per codificar tots els enllaços de les pàgines del www?

Algorismes sobre grafs

- Quins vèrtexs són **accessibles** des de quins?



Component connex



Ho podem veure com un laberint
Hem de guardar informació a mida que analitzem "explorem" el graf!

Algorismes sobre grafs

- Quins vèrtexs són **accessibles** des de quins?

```
procedure explore(G, v)
Input:    G = (V, E) is a graph; v ∈ V
Output:   visited(u) is set to true for all nodes u reachable from v

visited(v) = true
previsit(v)
for each edge (v, u) ∈ E:
   if not visited(u):  explore(u)
postvisit(v)
```

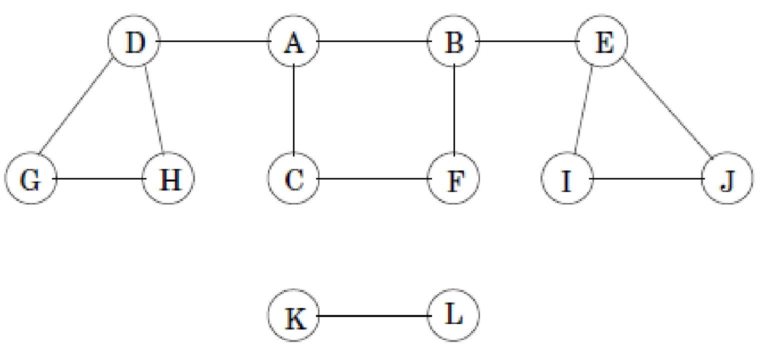
Algorismes sobre grafs

- **Recorregut topològic**
- Búsqueda en profunditat (**Depth-First Search - DFS**)

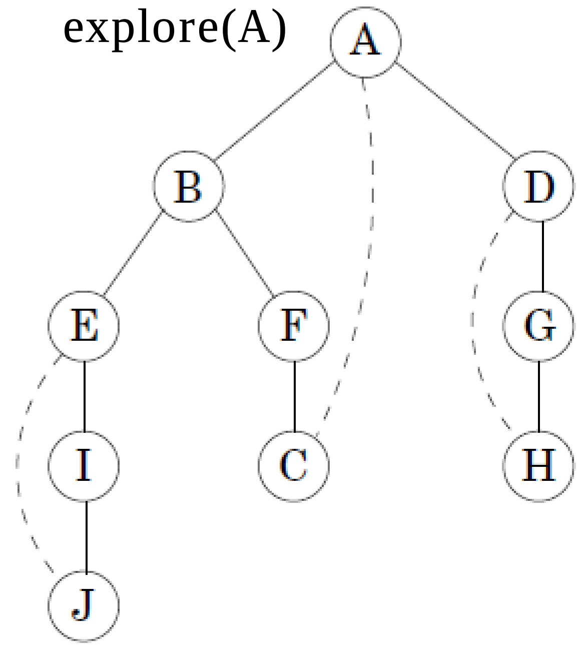
```
procedure dfs(G)

for all v ∈ V:
   visited(v) = false

for all v ∈ V:
   if not visited(v):  explore(v)
```

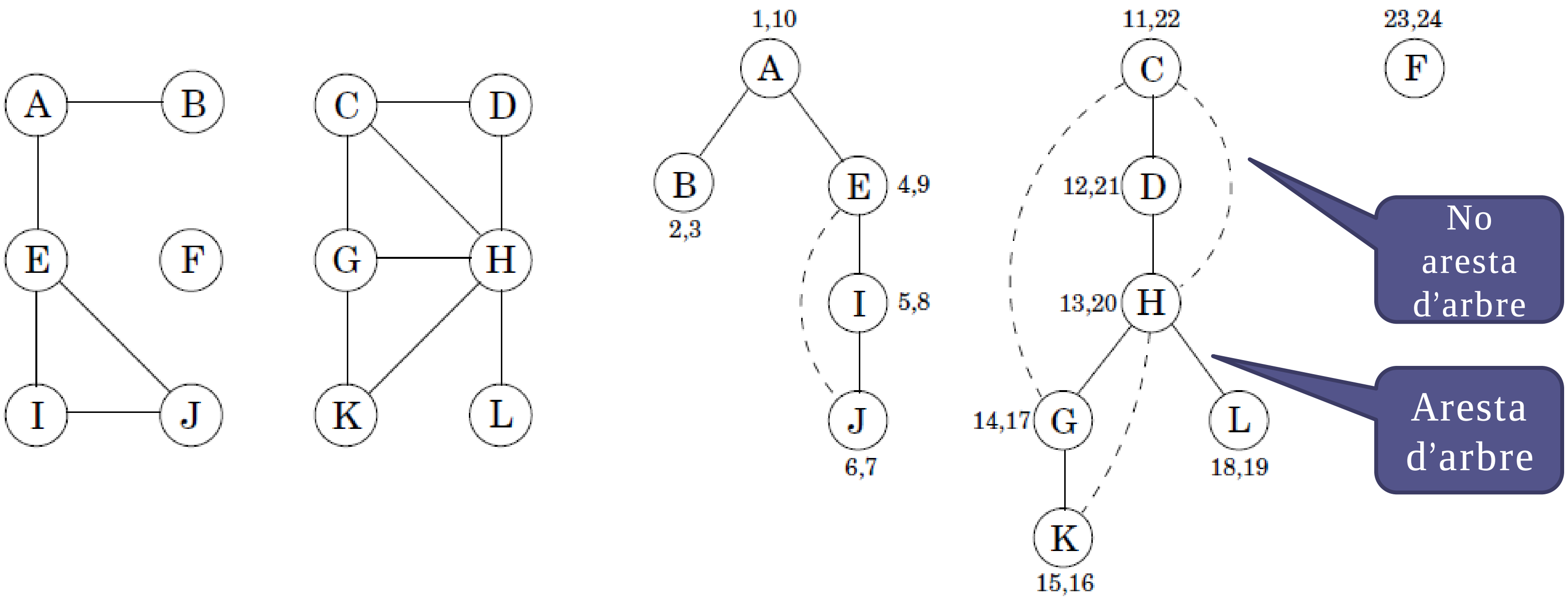


explore(A)



Algorismes sobre grafs

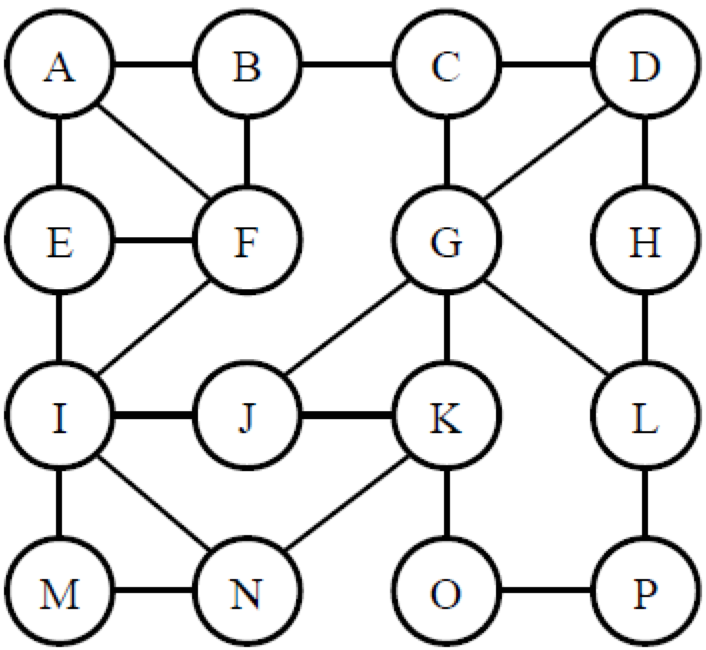
- DFS representa la connectivitat amb un **bosc d'arbres**

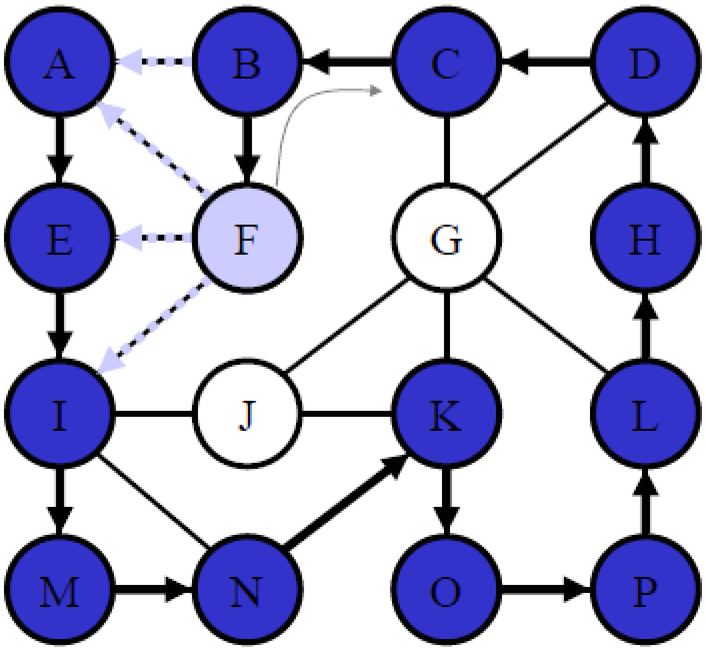


- Complexitat de DFS? $\Theta(|E|)$

Algorismes sobre grafs

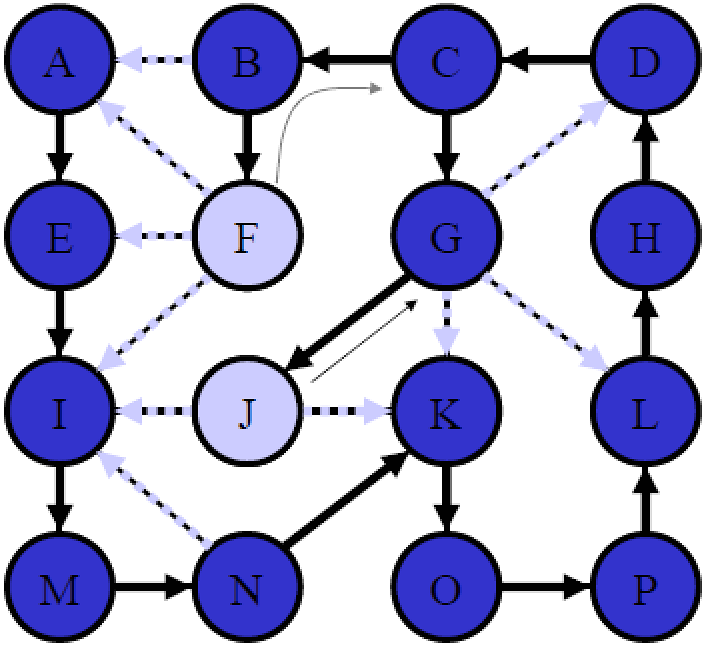
- Exemple

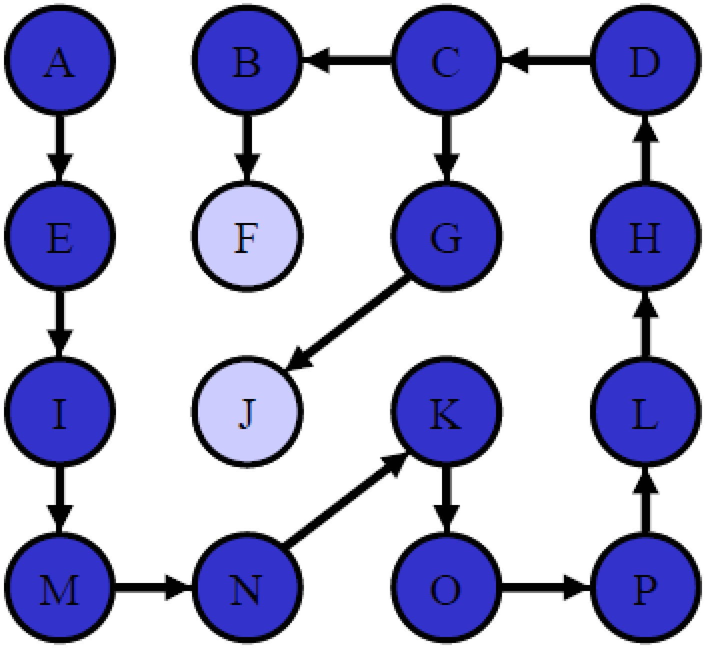




Algorismes sobre grafs

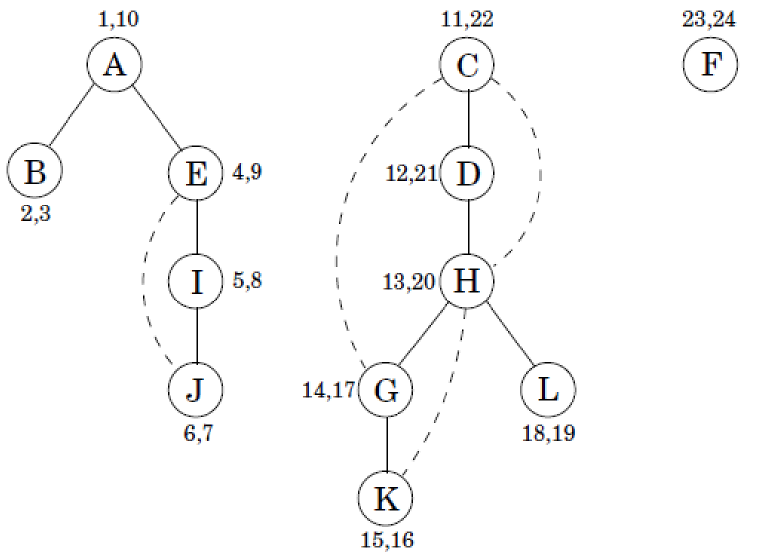
- Exemple





Algorismes sobre grafs

- DFS també ens soluciona un altre problema de grafs: els components conexos



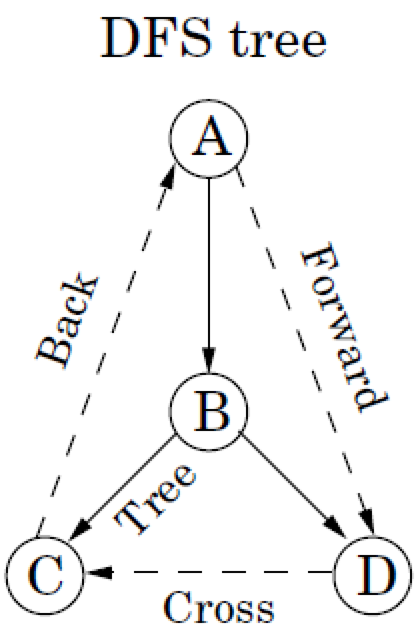
$\{A, B, E, I, J\}$ $\{C, D, G, H, K, L\}$ $\{F\}$

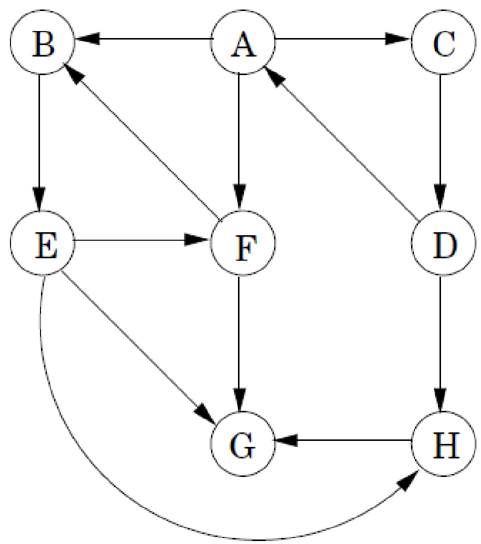
Cada crida a explore crea un nou arbre

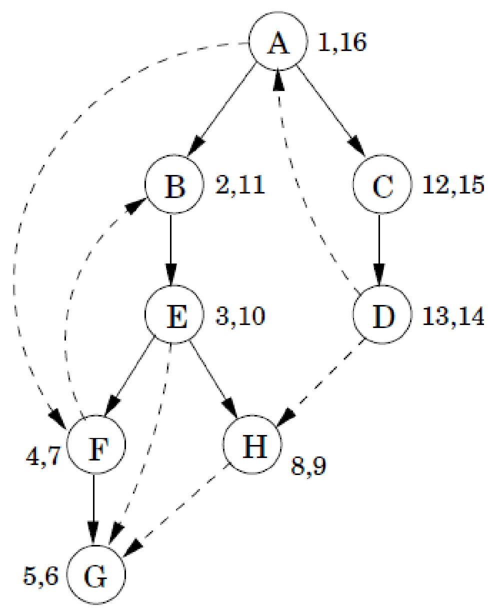
I es troba una **nova component conexa**

Algorismes sobre grafs

- Quins vèrtexs són accessibles des de quins?
- **Grafs dirigits**







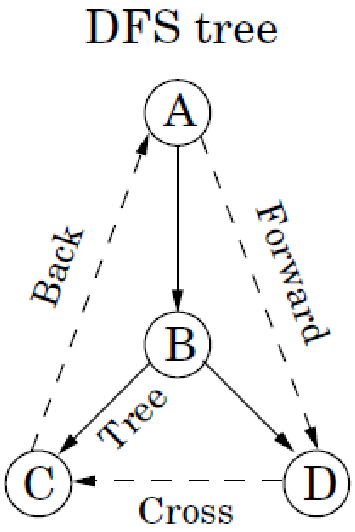
Algorismes sobre grafs

- En els grafs **no dirigits**, una **component connexa** conté com a mínim un **cicle**
- En els grafs **dirigits**, un **cicle** ha de començar i acabar en el mateix vèrtex existint connectivitat, sinó és **acíclic**

$$v_0 \to v_1 \to v_2 \to \cdots \to v_k \to v_0$$

Algorismes sobre grafs

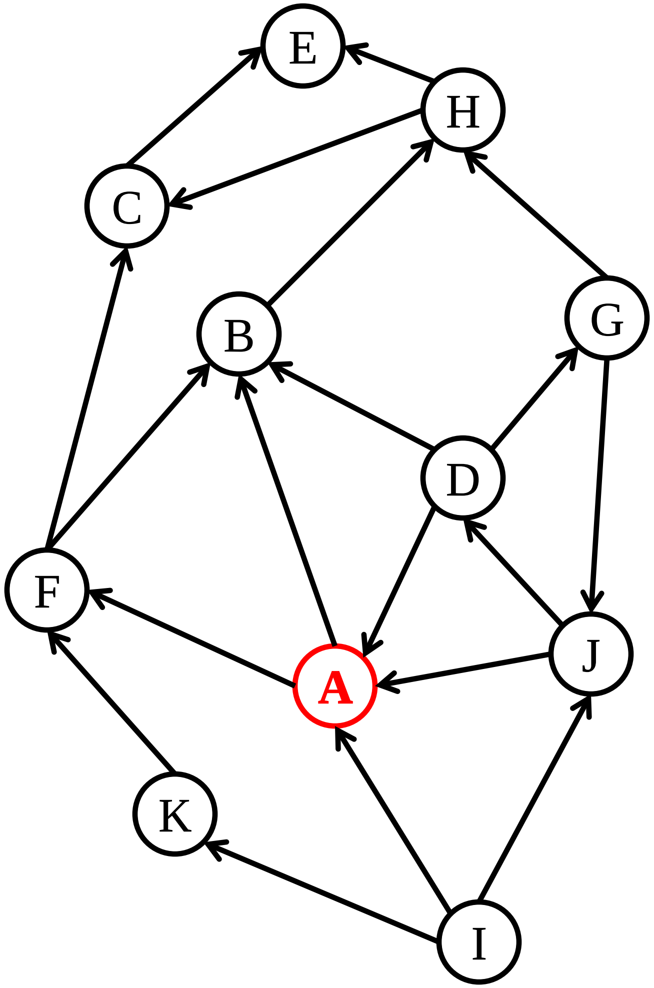
- Cicles en grafs dirigits
- Podem trobar un cicle en un graf dirigit en temps lineal? → Sí



Hem vist que DFS té complexitat lineal

Propietat: Un graf dirigit té un cicle si i només si l'algorisme DFS troba una aresta "back".

Exercici



- Fer un recorregut en DFS del graf començant amb explore(V,A)
- Identificar les components connexes