# Algoritmi e Strutture Dati

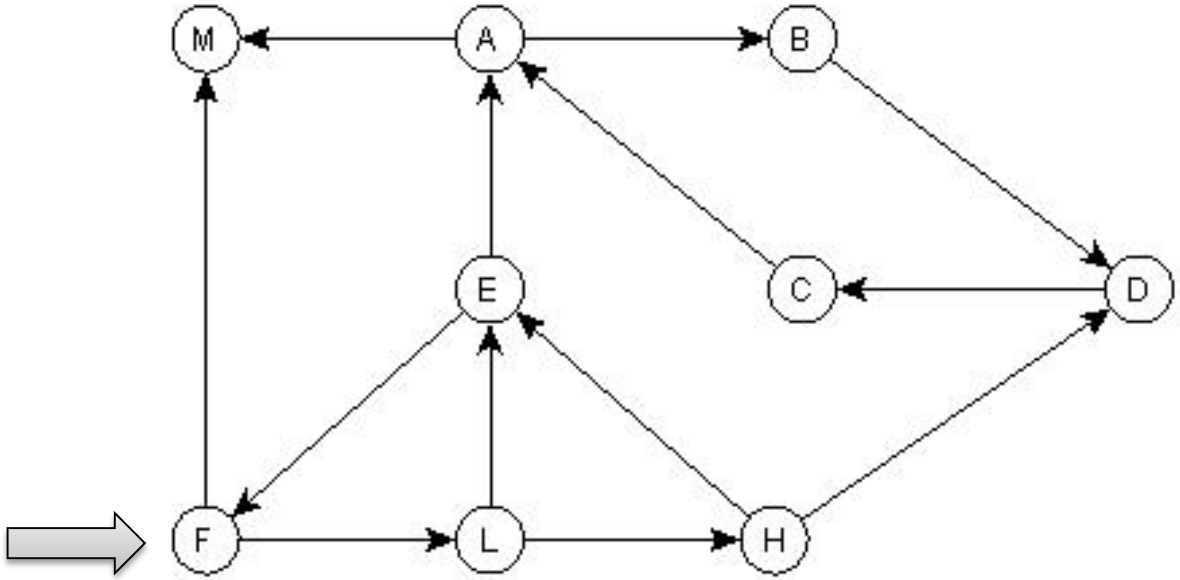
## Esercitazione 8

Domenico Fabio Savo

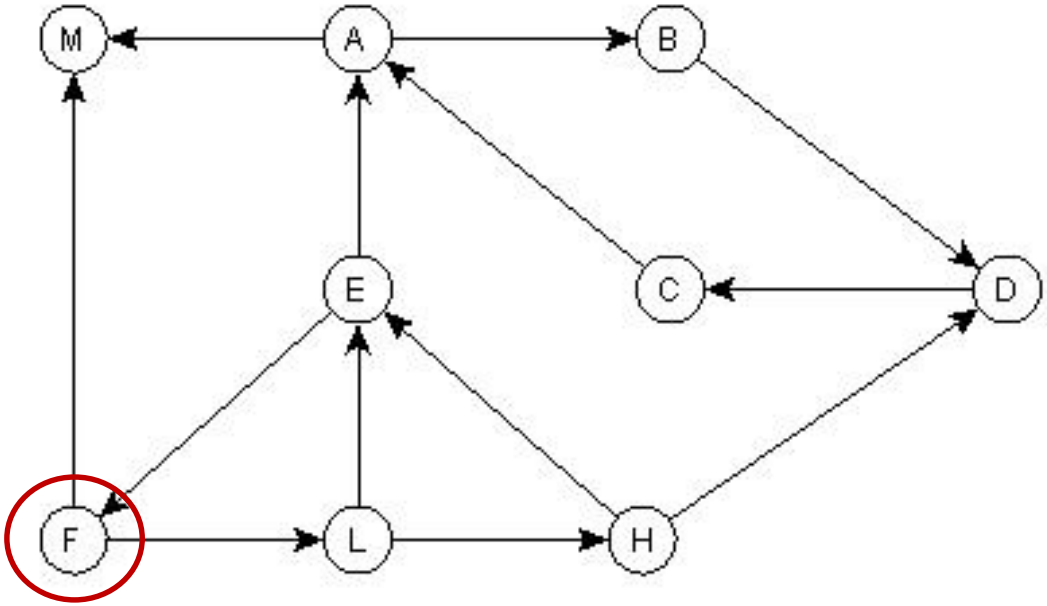
# Esercizio: visita in ampiezza

## Esercizio:

- Eseguire la **visita in ampiezza** del grafo *orientato* rappresentato in figura partendo dal nodo F.
- Presentare l'**albero BFS** prodotto dalla visita.

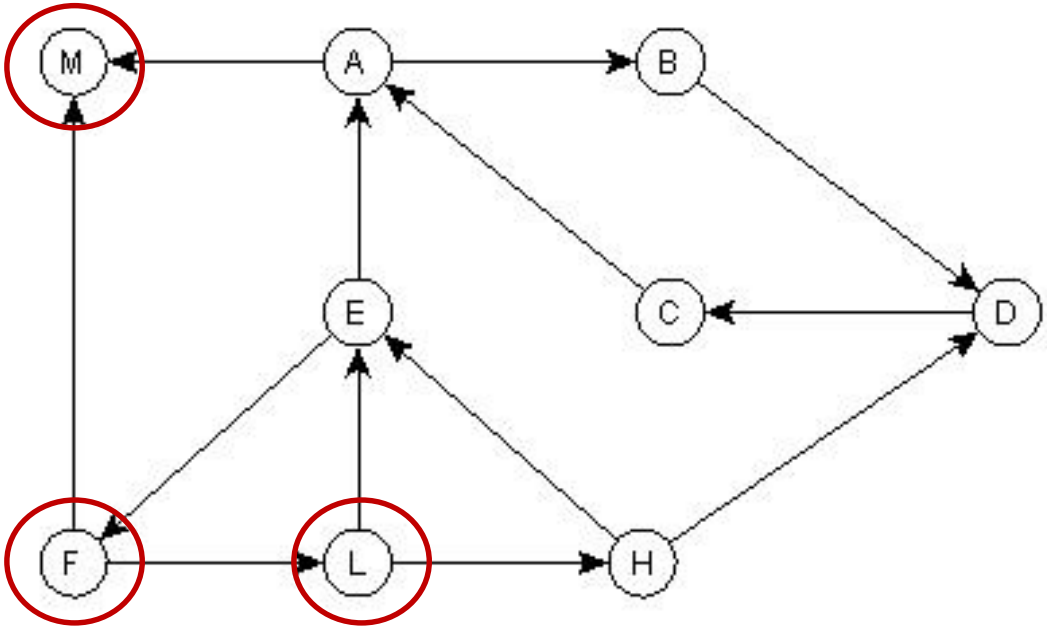


# Esercizio: visita in ampiezza

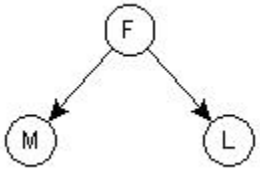



**Albero BST**


# Esercizio: visita in ampiezza

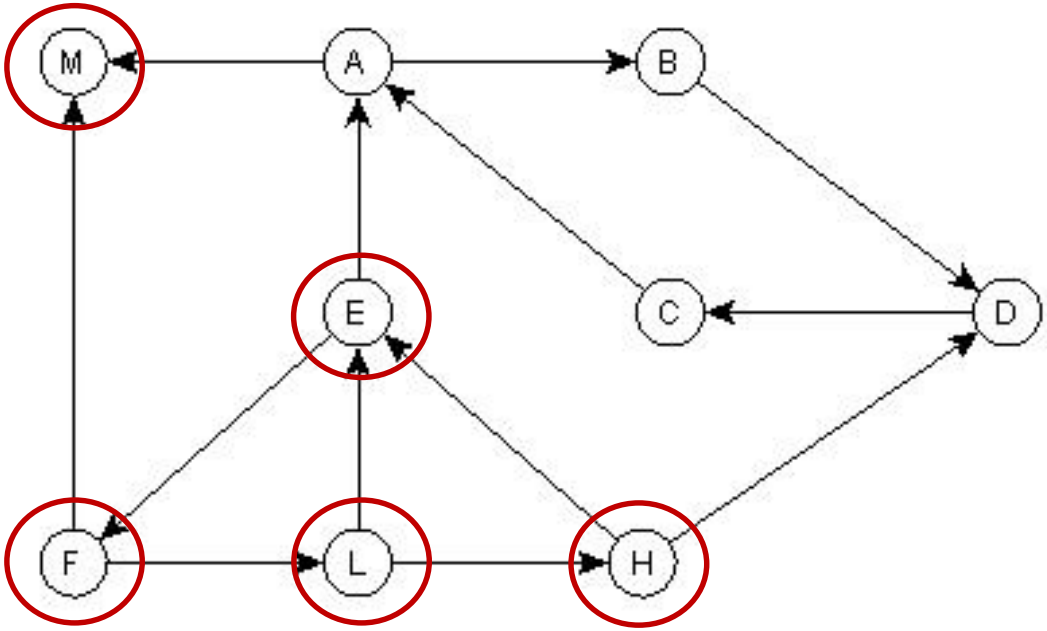



**Albero BST**



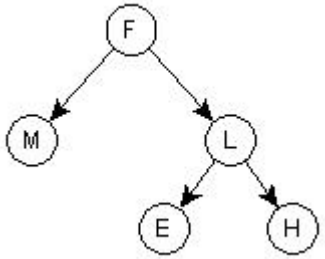

M L F

# Esercizio: visita in ampiezza

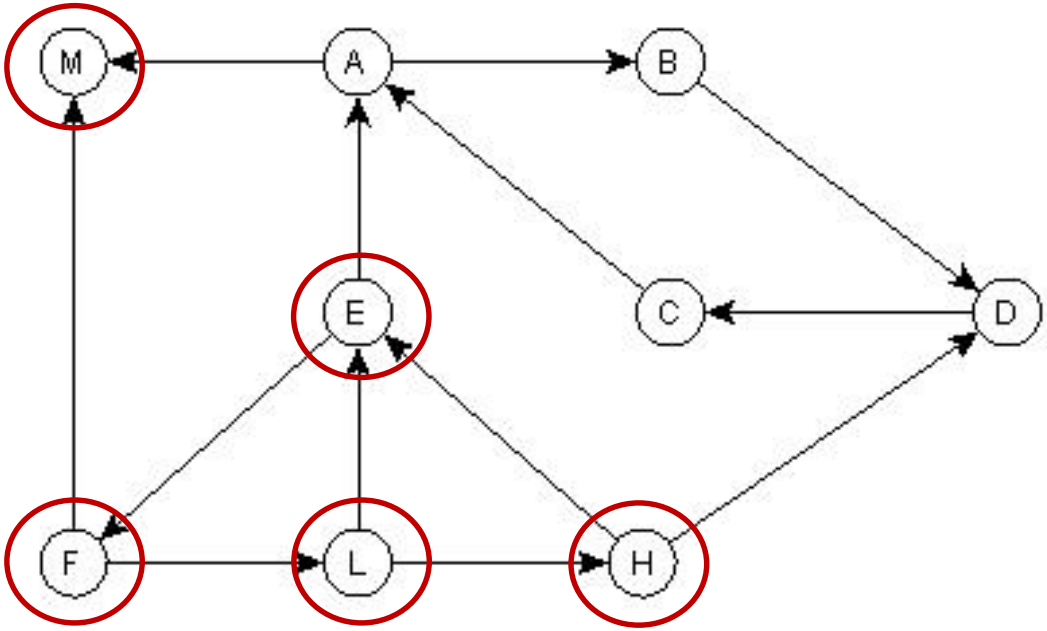



H E M L F

**Albero BST**

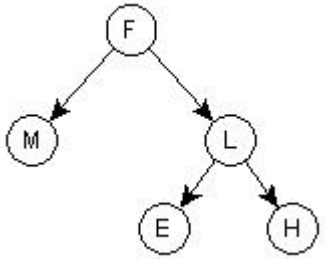


# Esercizio: visita in ampiezza

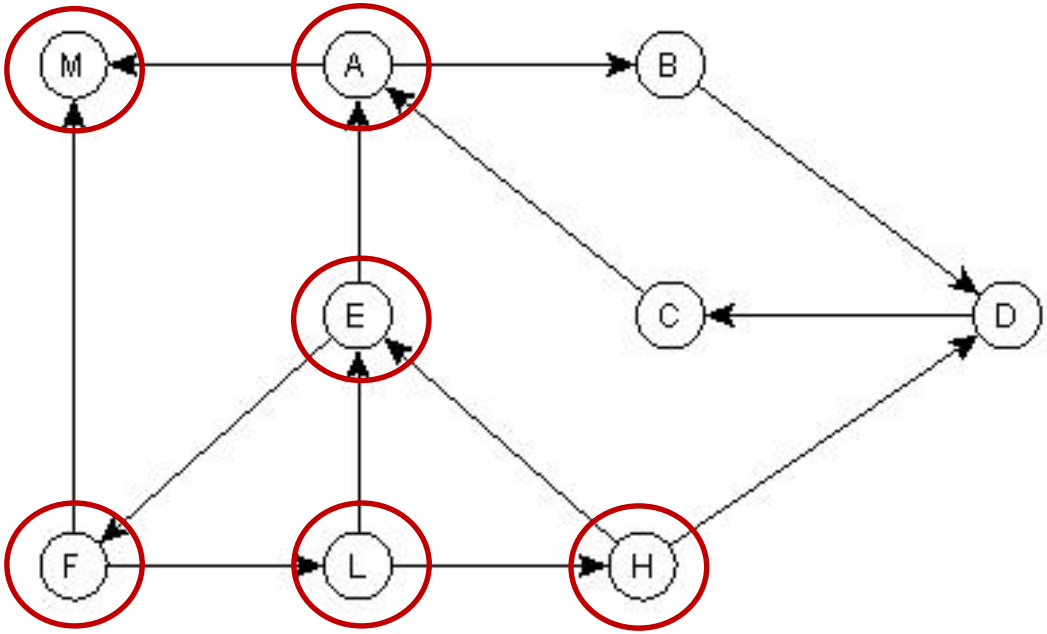



H E M L F

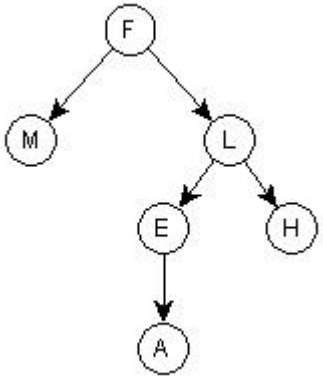
**Albero BST**



# Esercizio: visita in ampiezza

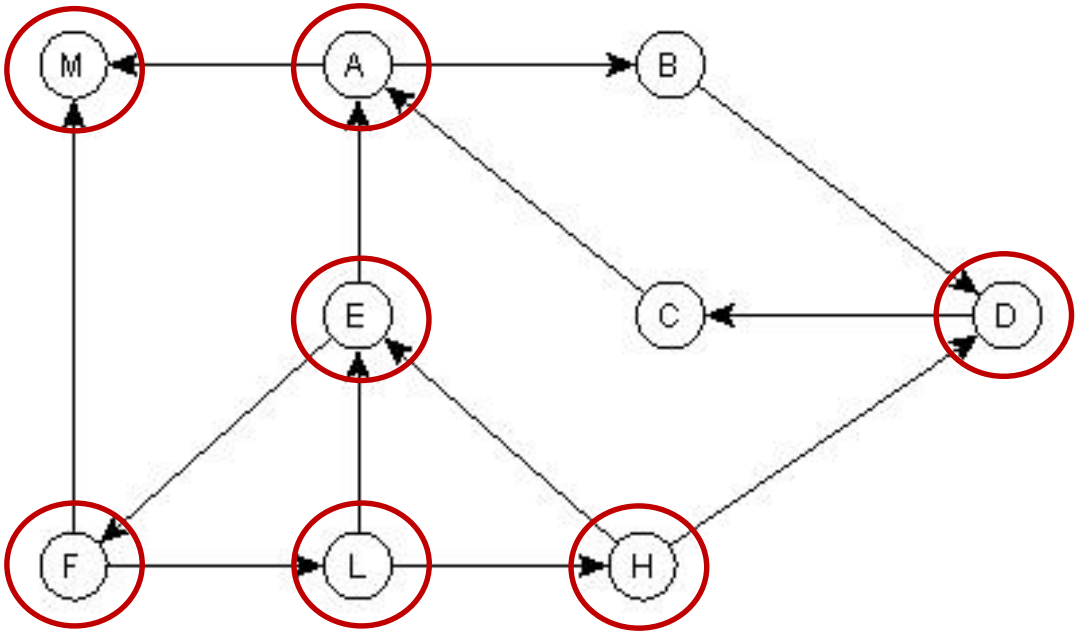



**Albero BST**

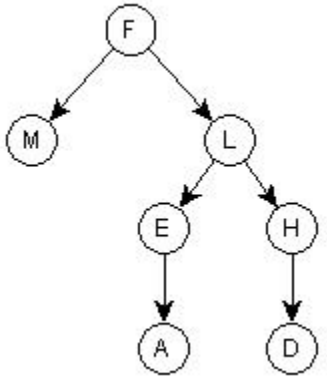



A H E M L F

# Esercizio: visita in ampiezza

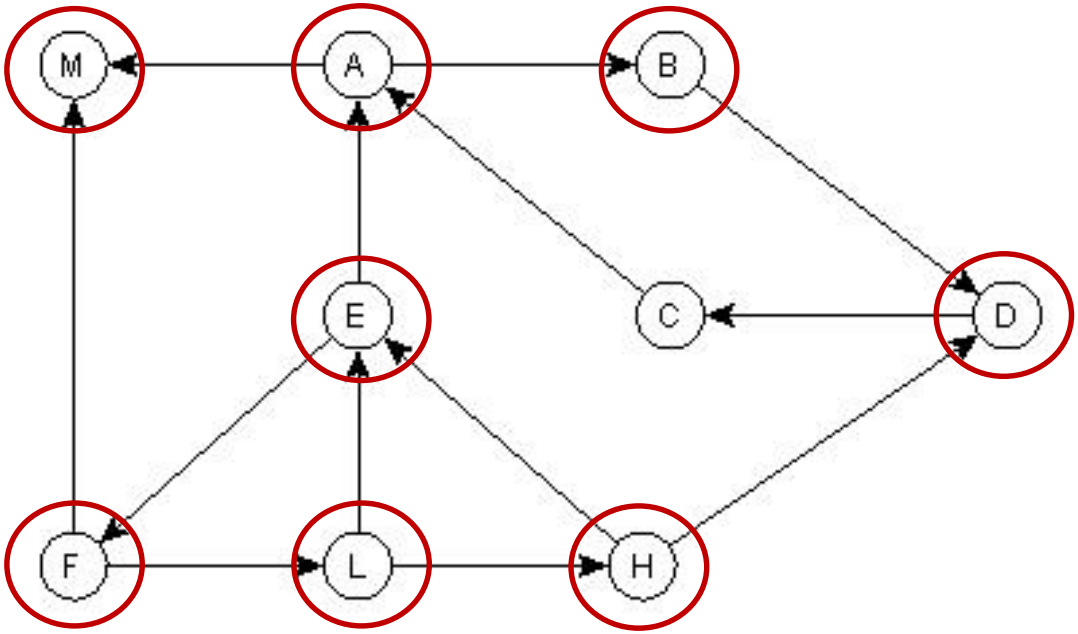



**Albero BST**

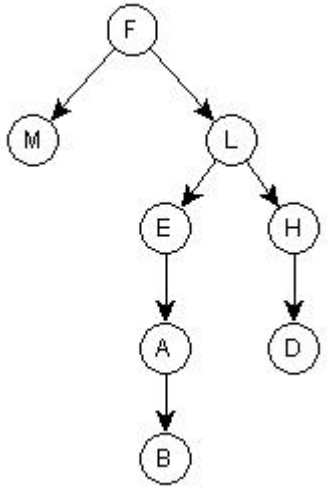



D A H E M L F

# Esercizio: visita in ampiezza

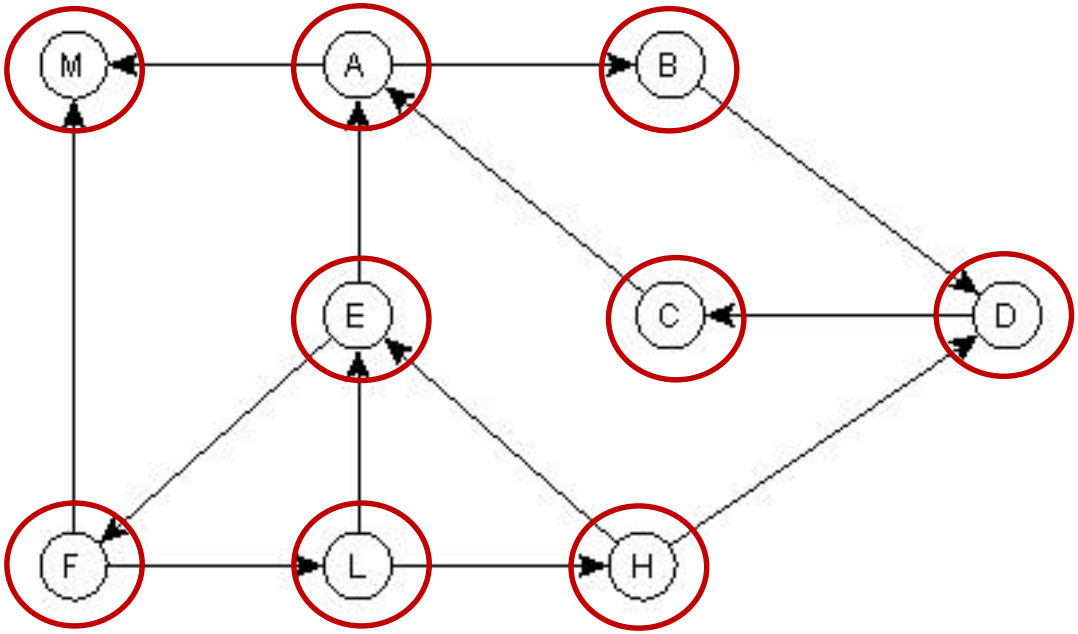



**Albero BST**



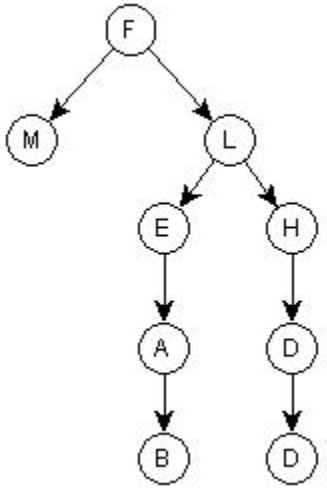

B D A H E M L F

# Esercizio: visita in ampiezza

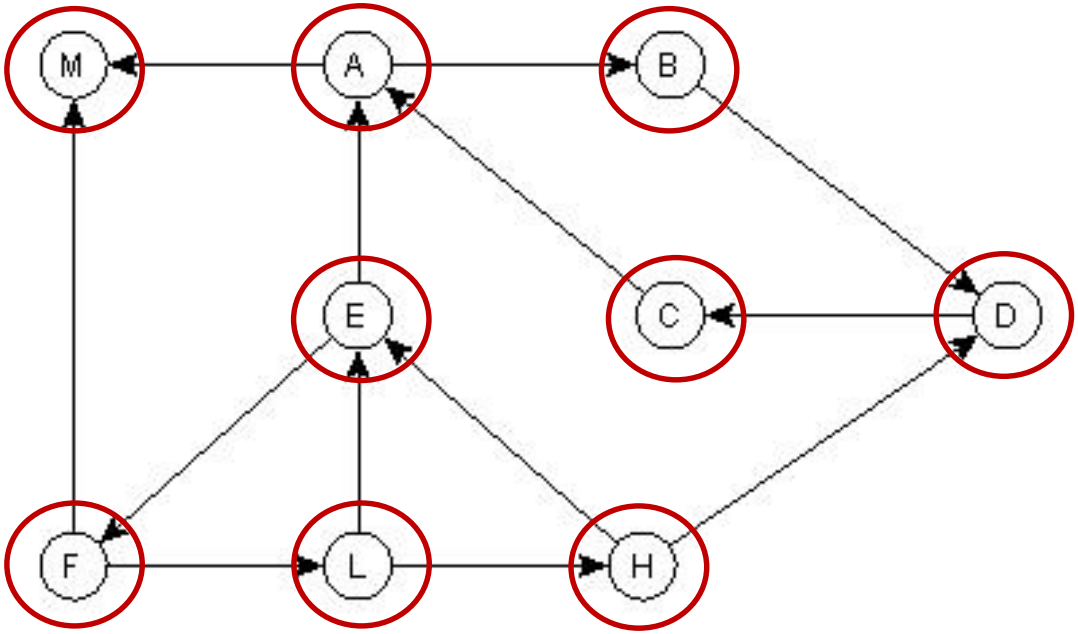



C B D A H E M L F

**Albero BST**

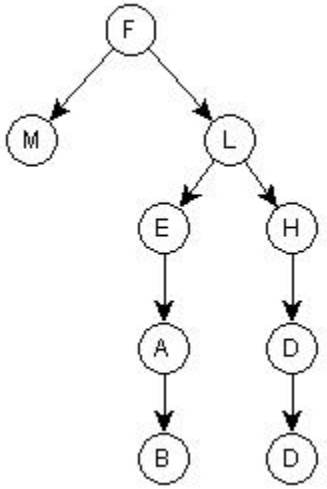


# Esercizio: visita in ampiezza

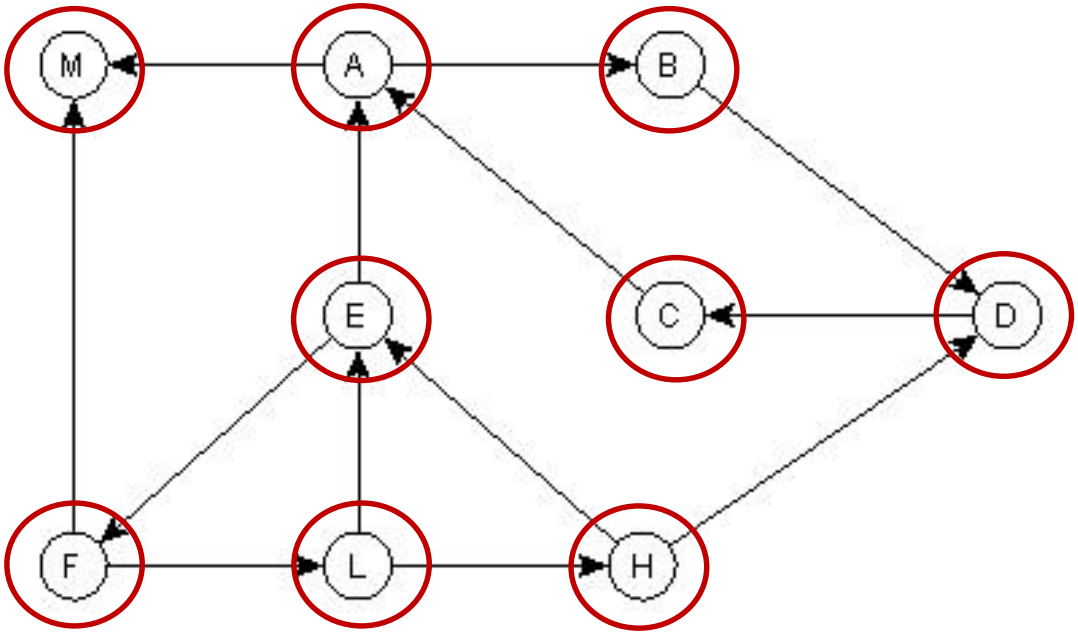



C B D A H E M L F

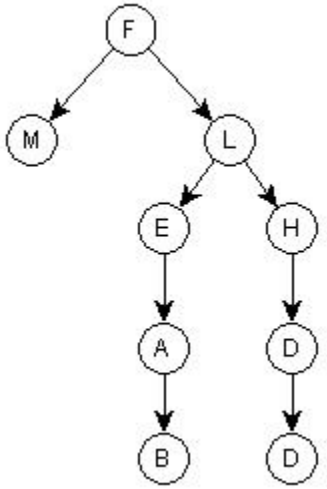
**Albero BST**



# Esercizio: visita in ampiezza




**Albero BST**



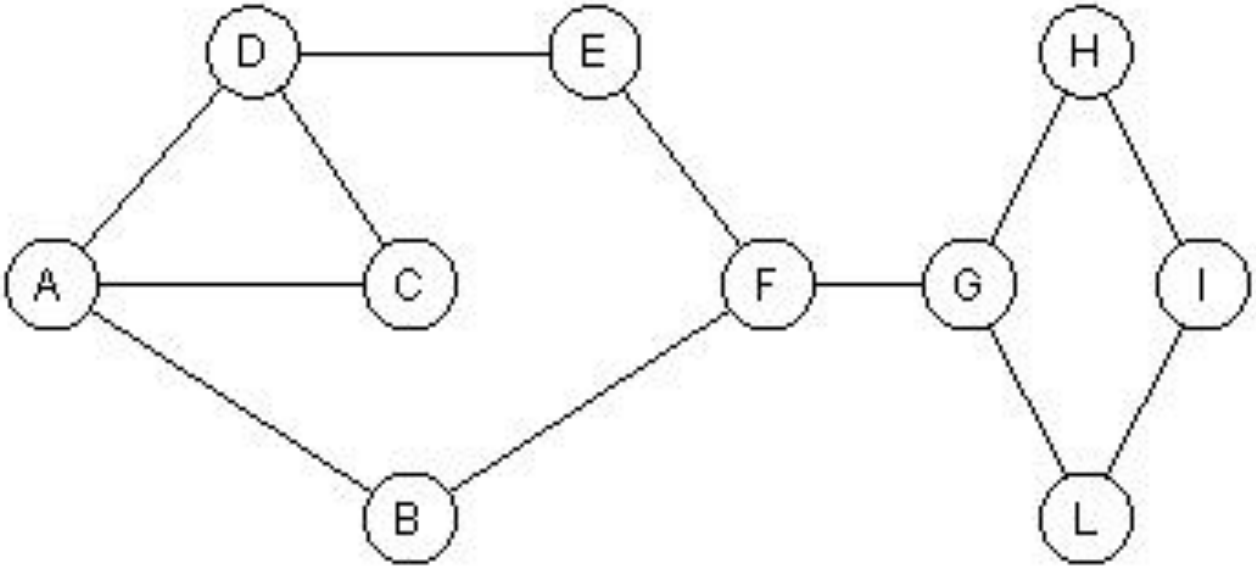

C B D A H E M L F

**La coda è vuota**!

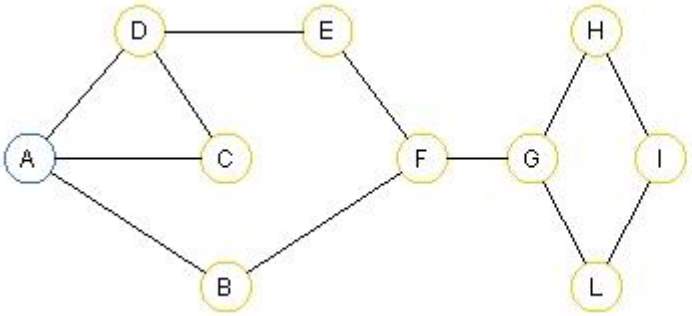
# Esercizio: visita in profondità

**Esercizio:**

- Eseguire la **visita in profondità** del grafo *non orientato* rappresentato in figura partendo dal nodo A.
- Presentare l'**albero DFS** prodotto dalla visita.



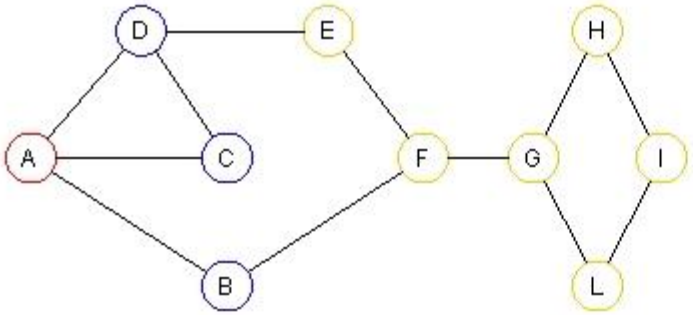
# Esercizio: visita in profondità

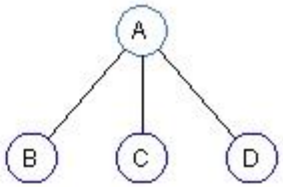




A

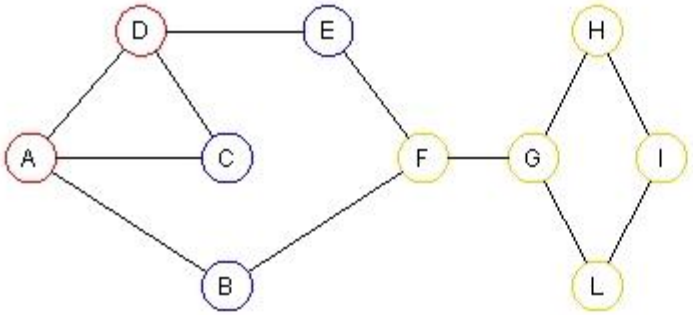
# Esercizio: visita in profondità

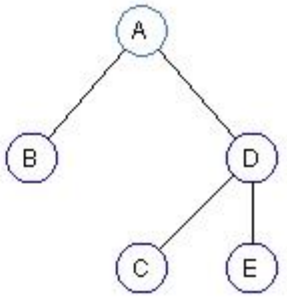



D
C
B
A

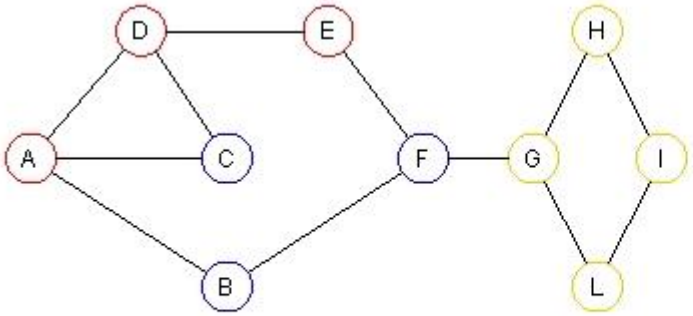
# Esercizio: visita in profondità

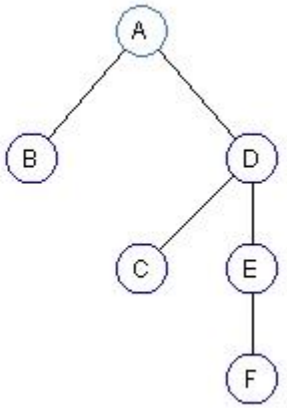



E
C
D
C
B
A

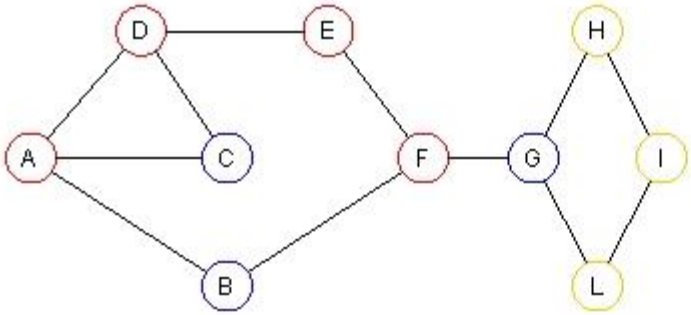
# Esercizio: visita in profondità

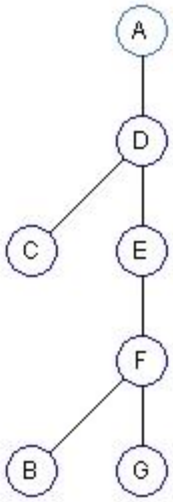



F
E
C
D
C
B
A

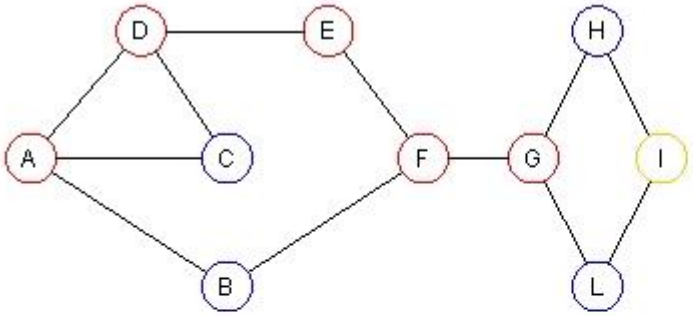
# Esercizio: visita in profondità

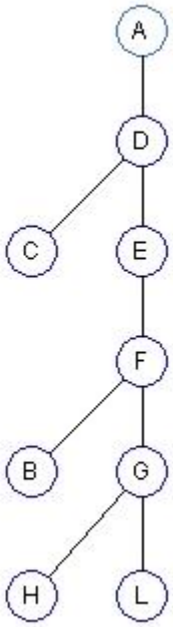



G
B
F
E
C
D
C
B
A

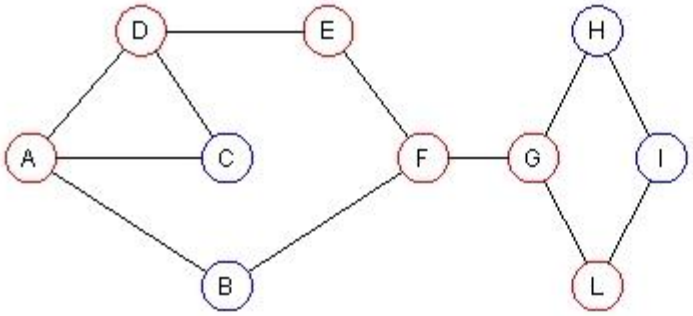
# Esercizio: visita in profondità

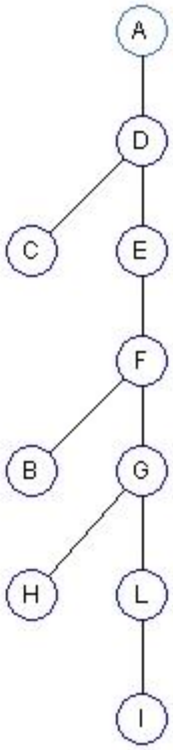



L
H
G
B
F
E
C
D
C
B
A

# Esercizio: visita in profondità

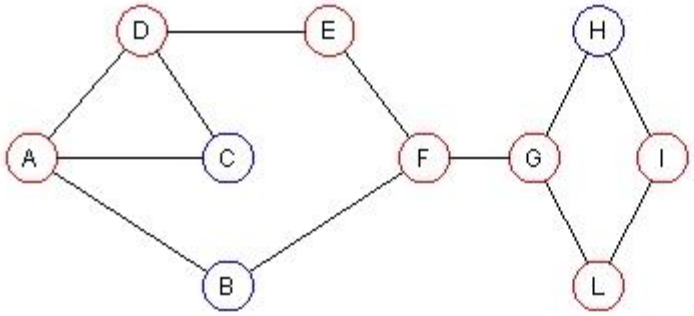


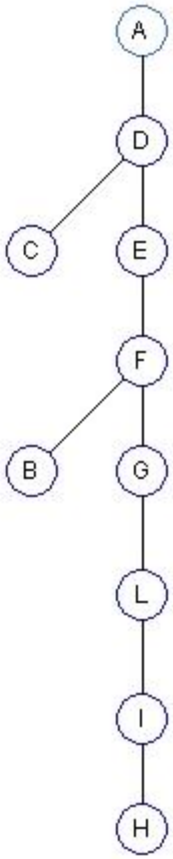




I
L
H
G
B
F
E
C
D
C
B
A

# Esercizio: visita in profondità

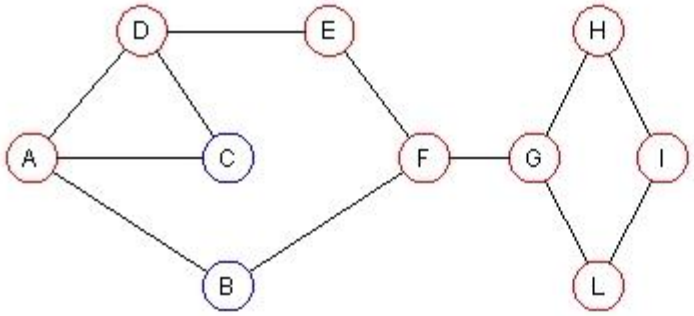
H
I
L
H
G
B
F
E
C
D
C
B
A

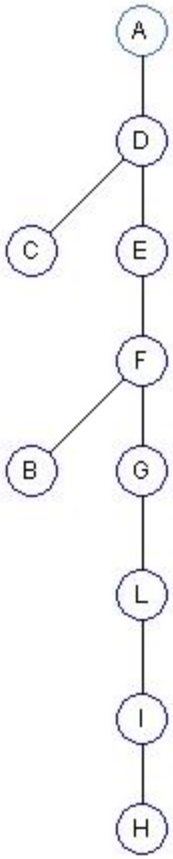


# Esercizio: visita in profondità

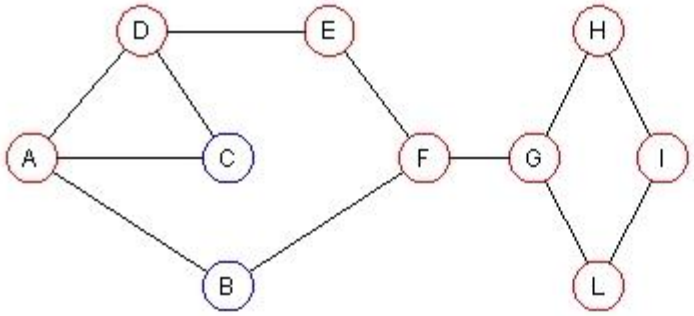
H
I
L
H
G
B
F
E
C
D
C
B
A

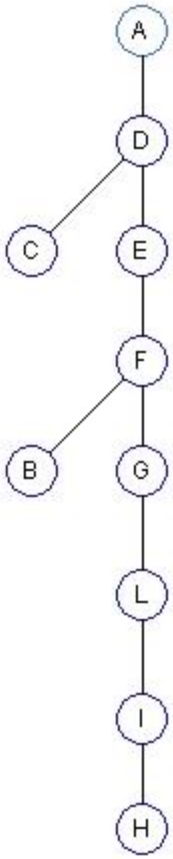


# Esercizio: visita in profondità

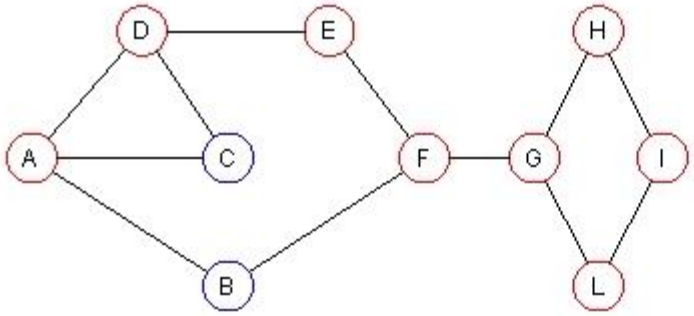
H
I
L
H
G
B
F
E
C
D
C
B
A

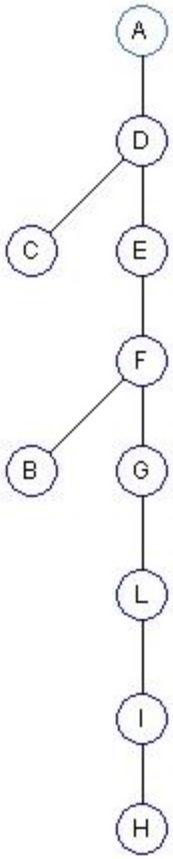


# Esercizio: visita in profondità

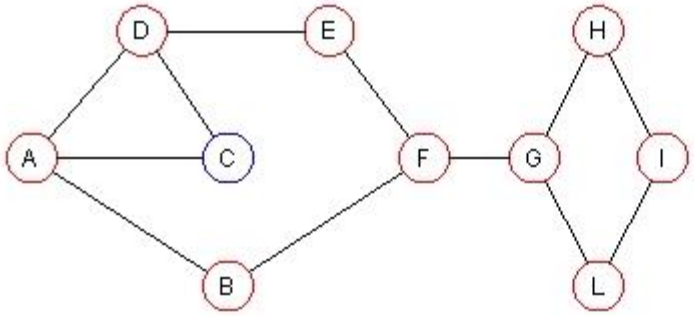
H
I
L
H
G
B
F
E
C
D
C
B
A

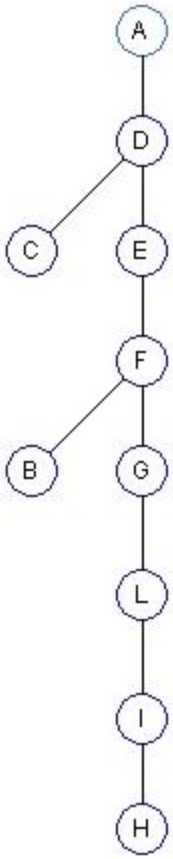


# Esercizio: visita in profondità

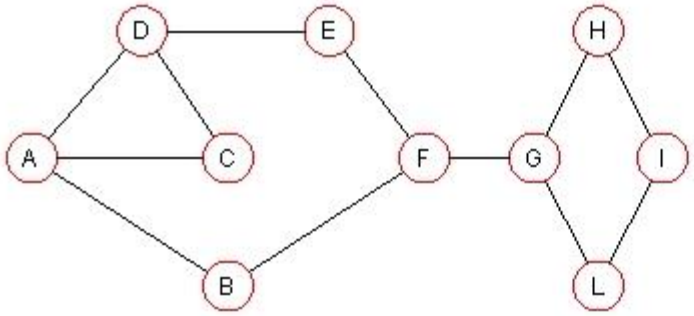
H
I
L
H
G
B
F
E
C
D
C
B
A

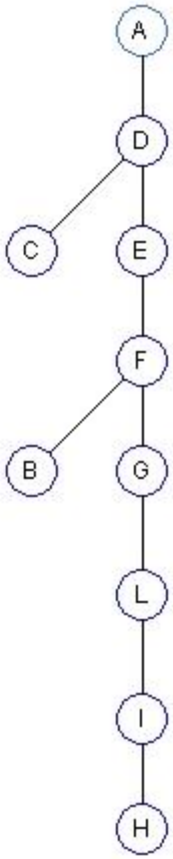


# Esercizio: visita in profondità

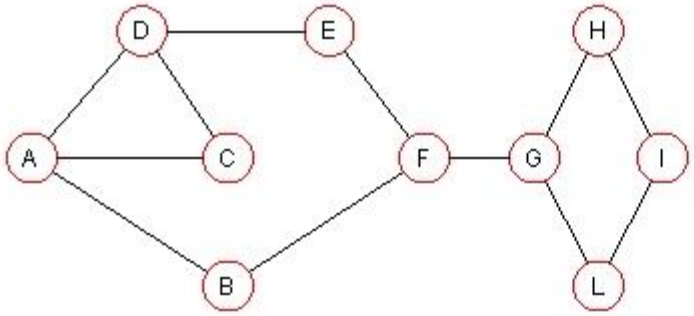
H
I
L
H
G
B
F
E
C
D
C
B
A

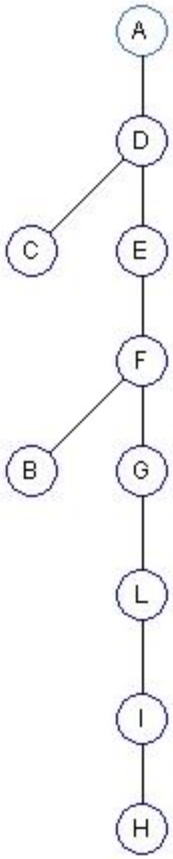


# Esercizio: visita in profondità

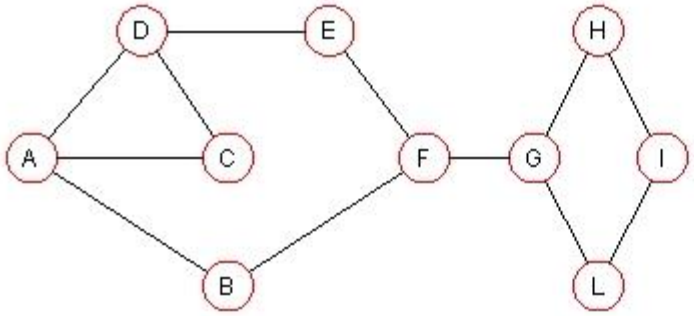
H
I
L
H
G
B
F
E
C
D
C
B
A

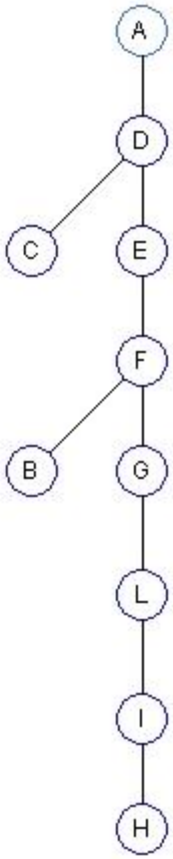


# Esercizio: visita in profondità

H
I
L
H
G
B
F
E
C
D
C
B
A

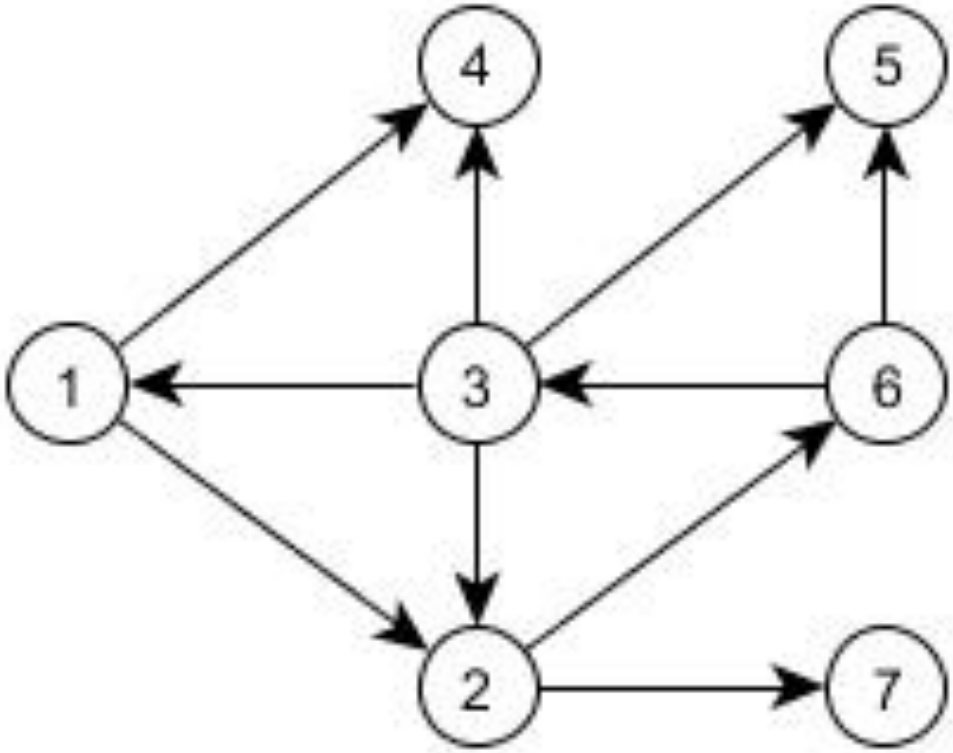



**La pila è vuota!**

# Esercizio (max grado)

Sia G = (V,E) il grafo *orientato* rappresentato in figura.



# Esercizio (max grado)

**Si chiede di:**

1. Fornire una rappresentazione del grafo basata sulla matrice di adiacenza.

2. Scrivere l'algoritmo

   **`nodoMaxGrado(array G[n][n])→ intero v`**

   che, preso in ingresso un grafo G (rappresentato come una matrice di adiacenza), restituisce l'indice (indicato come un intero) di uno dei vertici con *grado in uscita massimo*.
3. Indicare il costo computazionale dell'algoritmo proposto.

**Nota**: un vertice **v** ha *grado in uscita massimo* se **non** esiste in **G** nessun vertice **w** tale che $\delta_{out}(v) < \delta_{out}(w)$

# Esercizio (max grado)

**Soluzione:**

1. Fornire una rappresentazione del grafo con **matrice di adiacenza.**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **1** | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| **2** | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| **3** | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **6** | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| **7** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Esercizio (max grado)

**Soluzione:**

```
algoritmo nodoMaxGrado(array G[n][n]) → intero v
1.  intero v_max ← 1;
2.  intero g_max ← 0;
3.  for (int r; r = 1; r ≤ n) do
4.          int current_g ← 0;
5.          for(int c; c = 1 ; c ≤ n ) do
6.                  current_g ← current_g + G[r][c];
7.          if( current_g > g_max) then
8.                  g_max ← current_g;
9.                  v_max ← r;
10. return v_max;
```

→ Il costo dell'algoritmo è $\mathbf{O(n^2)}$