Estruturas de dados

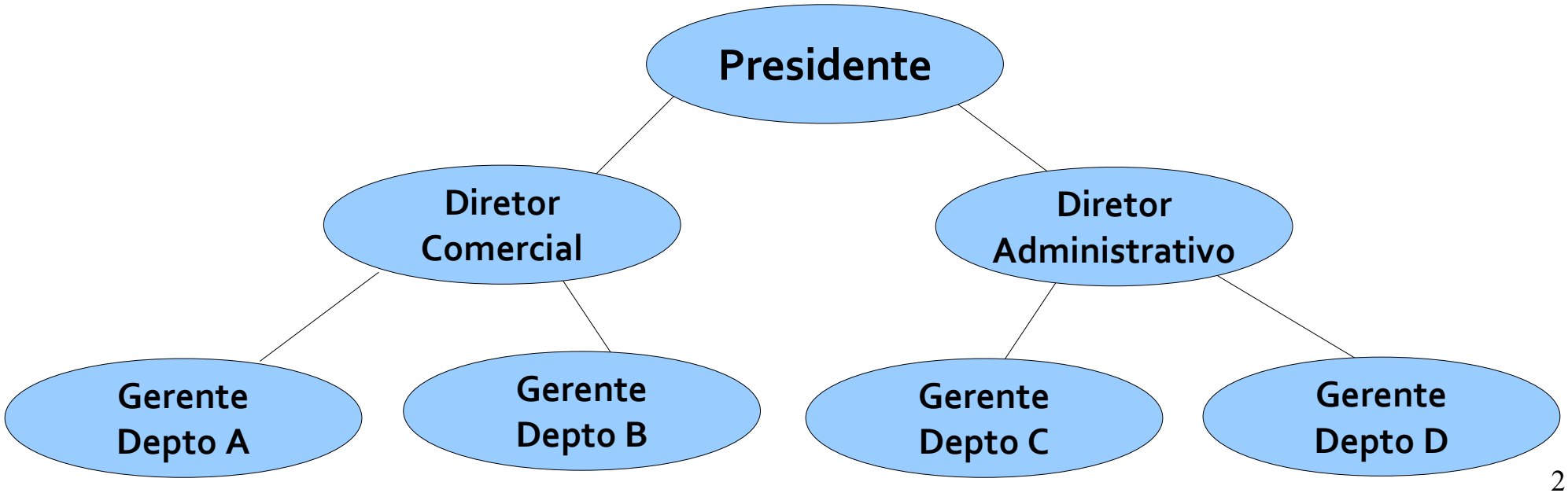
Árvores

Aula 15

Árvores

Uma árvore é uma estrutura de dados que se caracteriza por uma relação de hierarquia entre os elementos que a compõe.

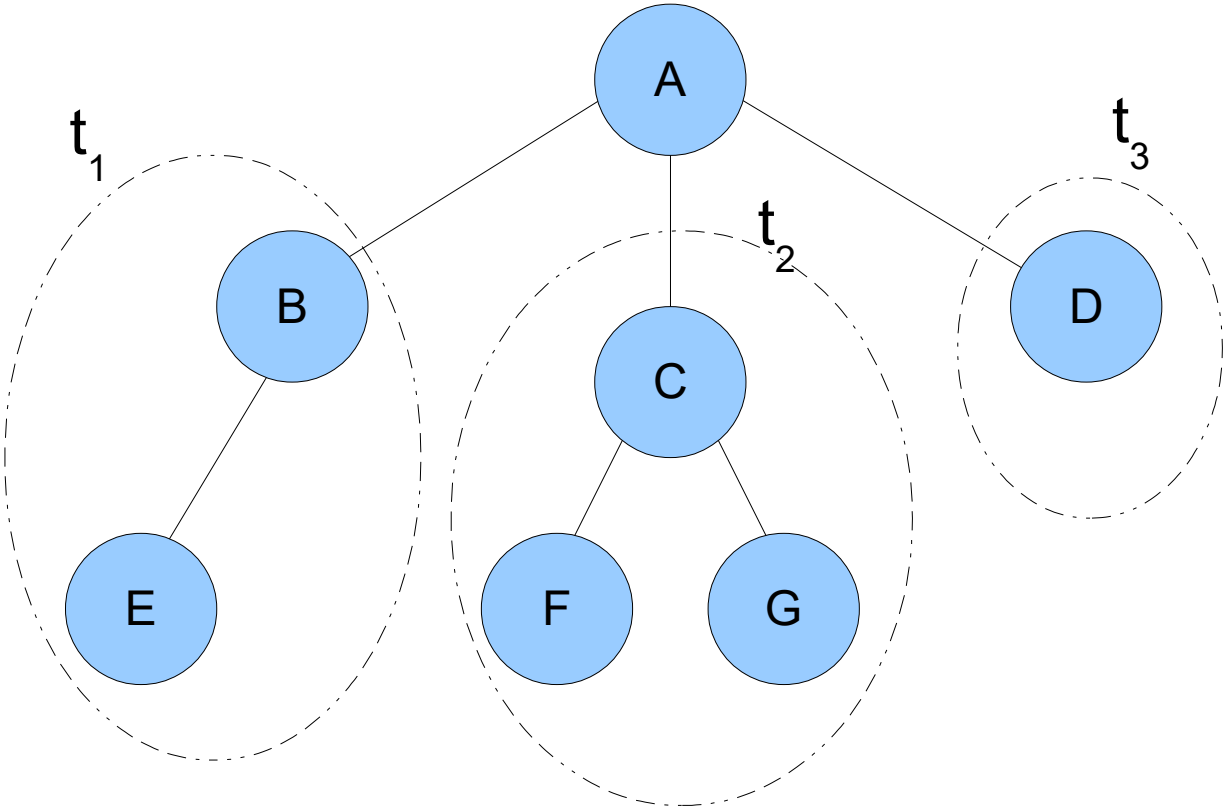
Ex: - Organograma de uma empresa.
- Divisão de um livro em capítulos, seções, tópicos.
- Estruturas de diretórios.



Árvores

Definição: É um conjunto finito T de um ou mais nodos tal que:

- Existe um nodo denominado raiz da árvore.
- Os demais nodos formam **n>=0** subconjuntos disjuntos $t_1, t_2, \ldots, t_n$, onde cada um deles é uma árvore. As árvores t_i, **(1 <= i <= n)** recebem a denominação de subárvores.



Árvores

Terminologia

raiz: Nodo de origem da árvore.

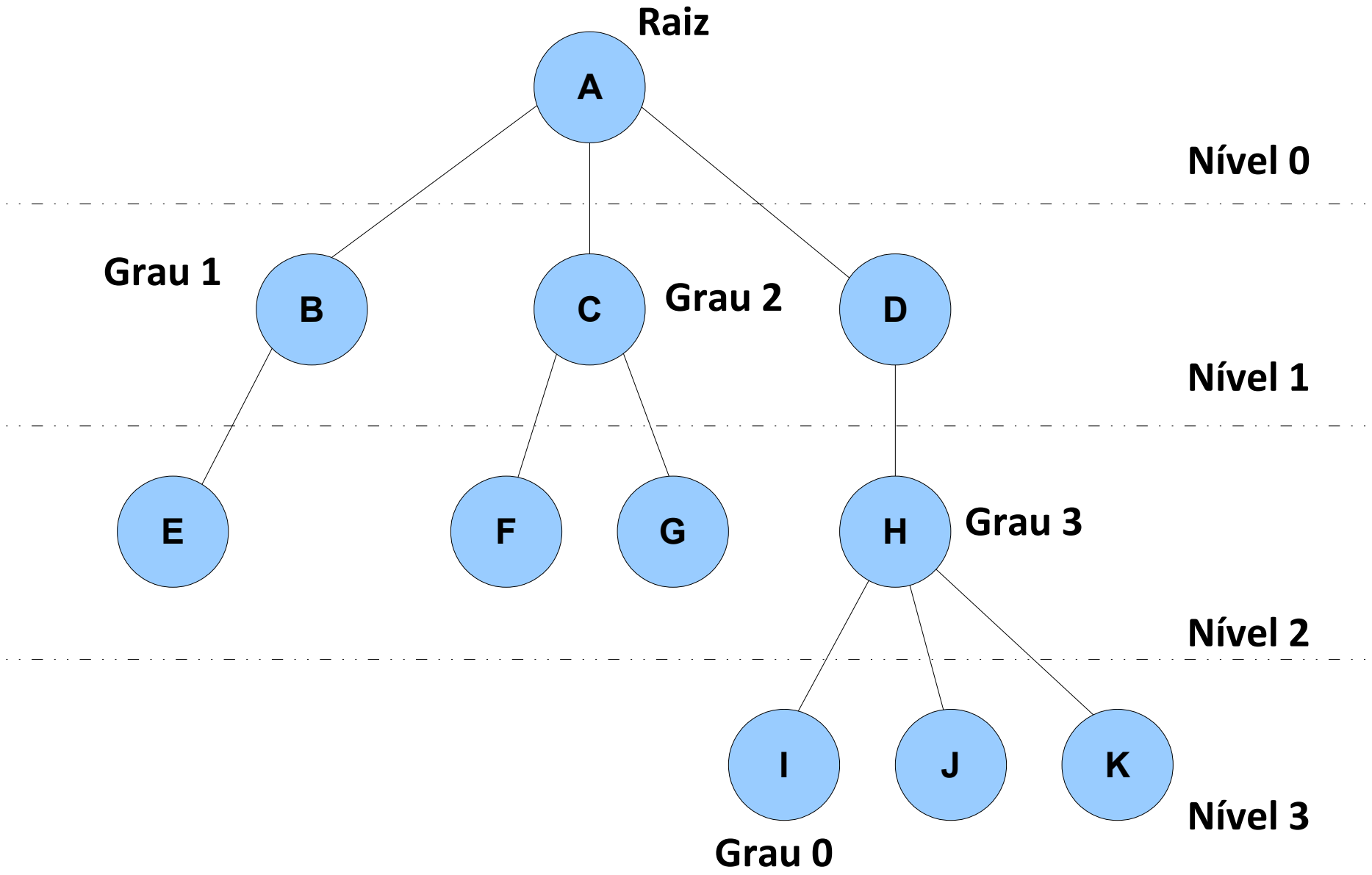
folha (nodo terminal): Nodo que não tem filhos.

grau de um nodo: número de filhos do nodo.

nível de um nodo: zero para o nodo raiz; e para os demais nodos é o número de "linhas" que o ligam ao nodo raiz.

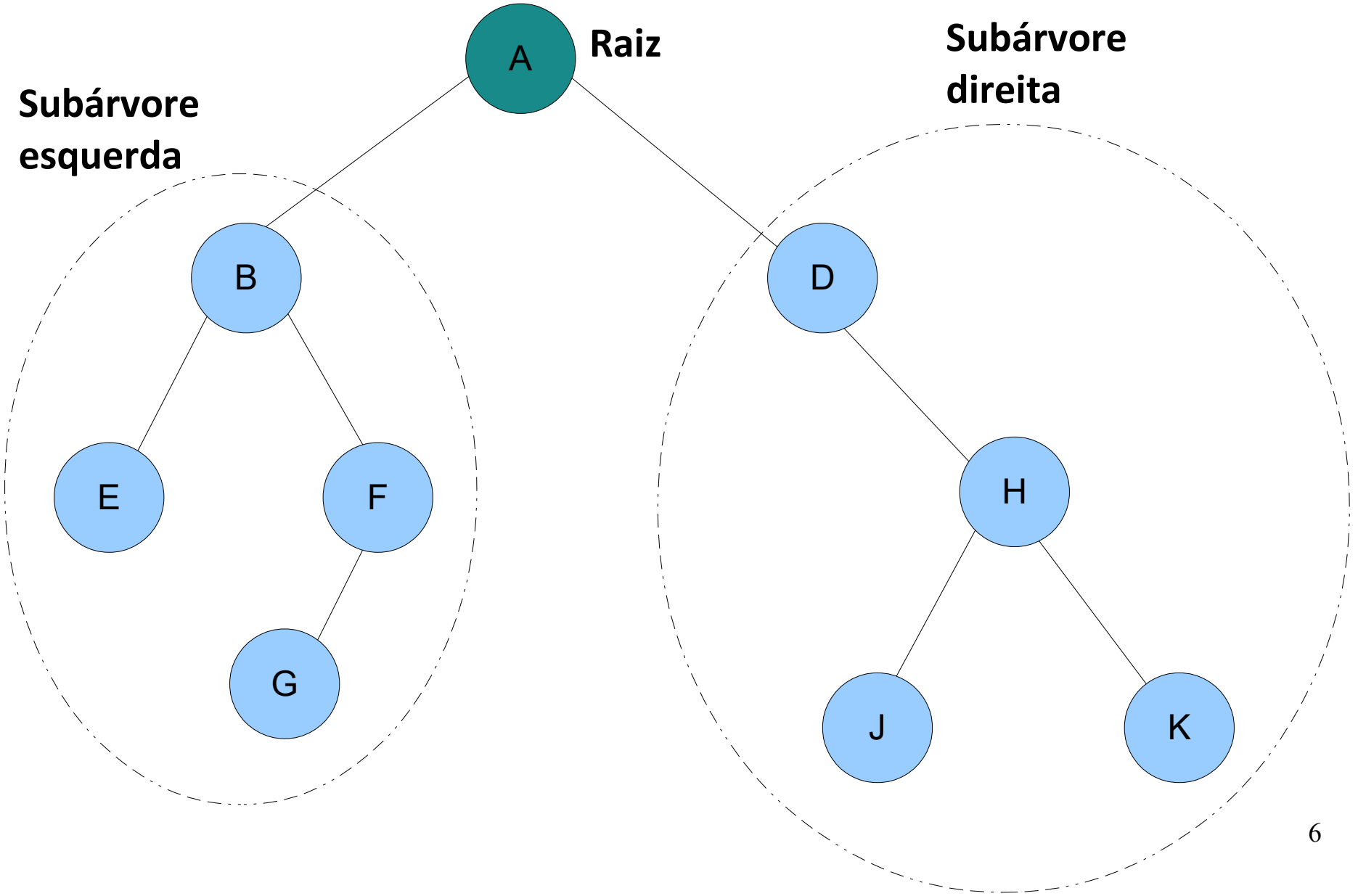
altura (ou profundidade): é o nível mais alto da árvore.

Árvores


Raiz
A
Nível 0
Grau 1
B
C
Grau 2
D
Nível 1
E
F
G
H
Grau 3
Nível 2
I
J
K
Nível 3
Grau 0

Árvores binárias

São árvores em que todos os nodos tem grau menor ou igual a dois.

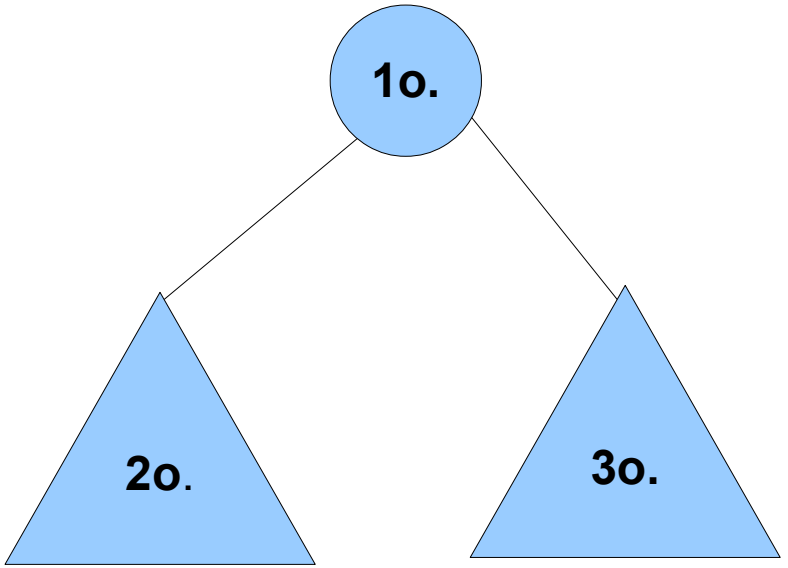


Percursos em árvores binárias

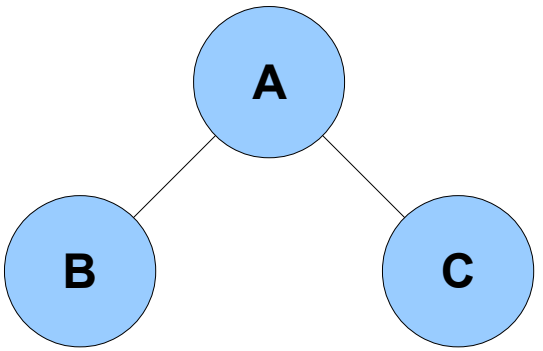
- Pré-ordem
- Em-ordem
- Pós-ordem
- Em-nível

Percurso em pré-ordem

```
inicio
    se árvore não é vazia então
        Visitar o nodo raiz
        Percorrer a subárvore esquerda
        Percorrer a subárvore direita
    fim_se
fim
```

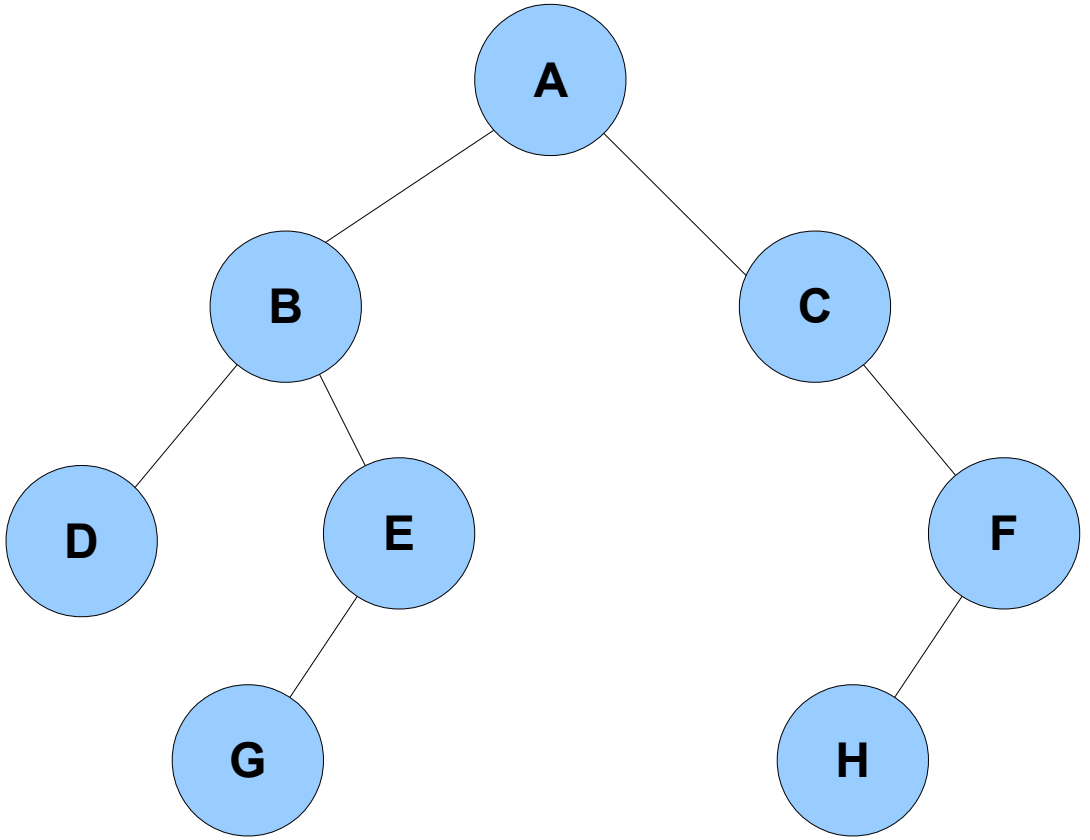


Exemplo



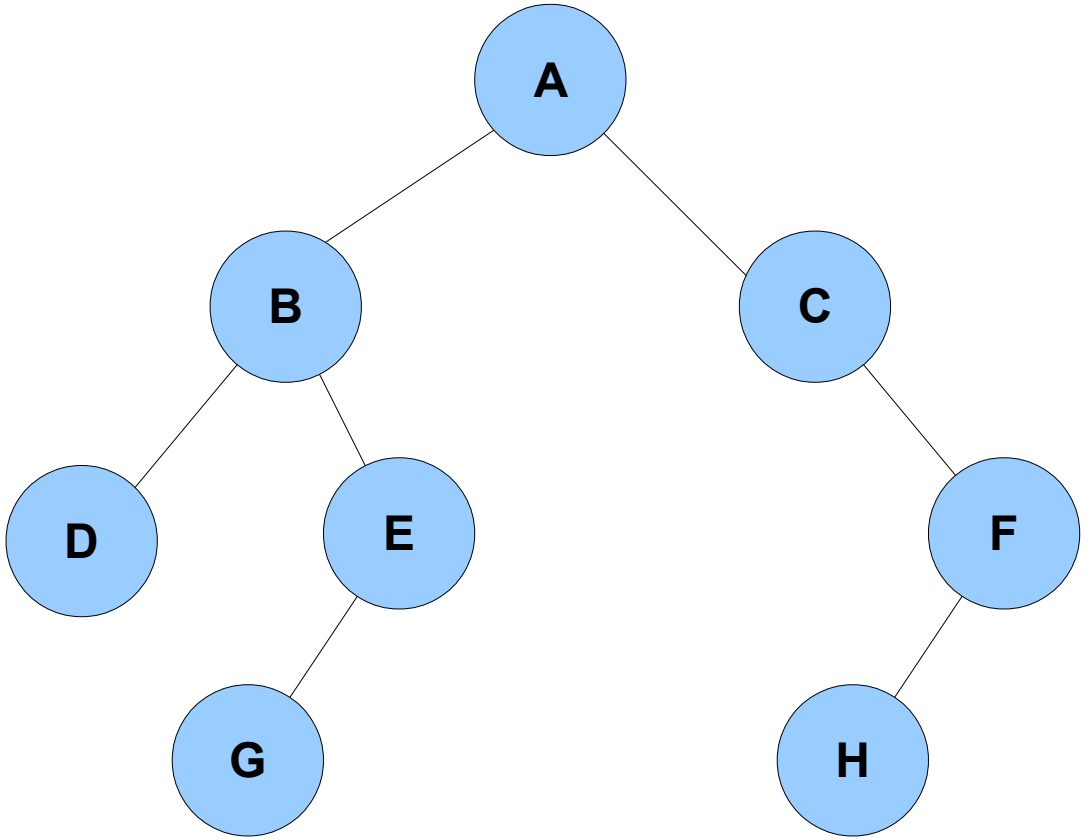
A B C

Percurso em pré-ordem



? ? ?

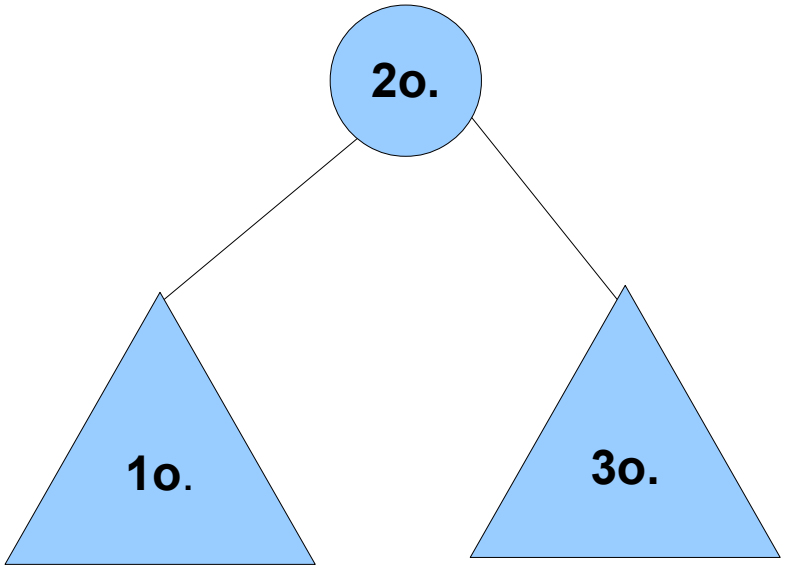
Percurso em pré-ordem



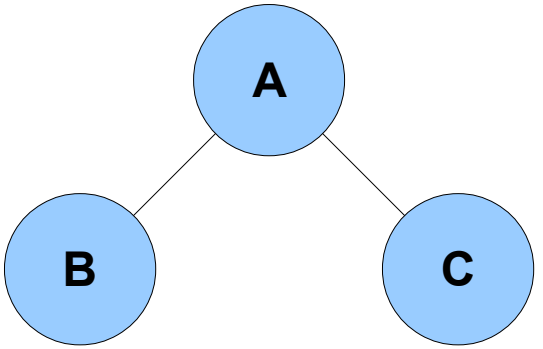
ABDEGCFH

Percurso em-ordem

```
inicio
    se árvore não é vazia então
        Percorrer a subárvore esquerda
        Visitar o nodo raiz
        Percorrer a subárvore direita
    fim_se
fim
```

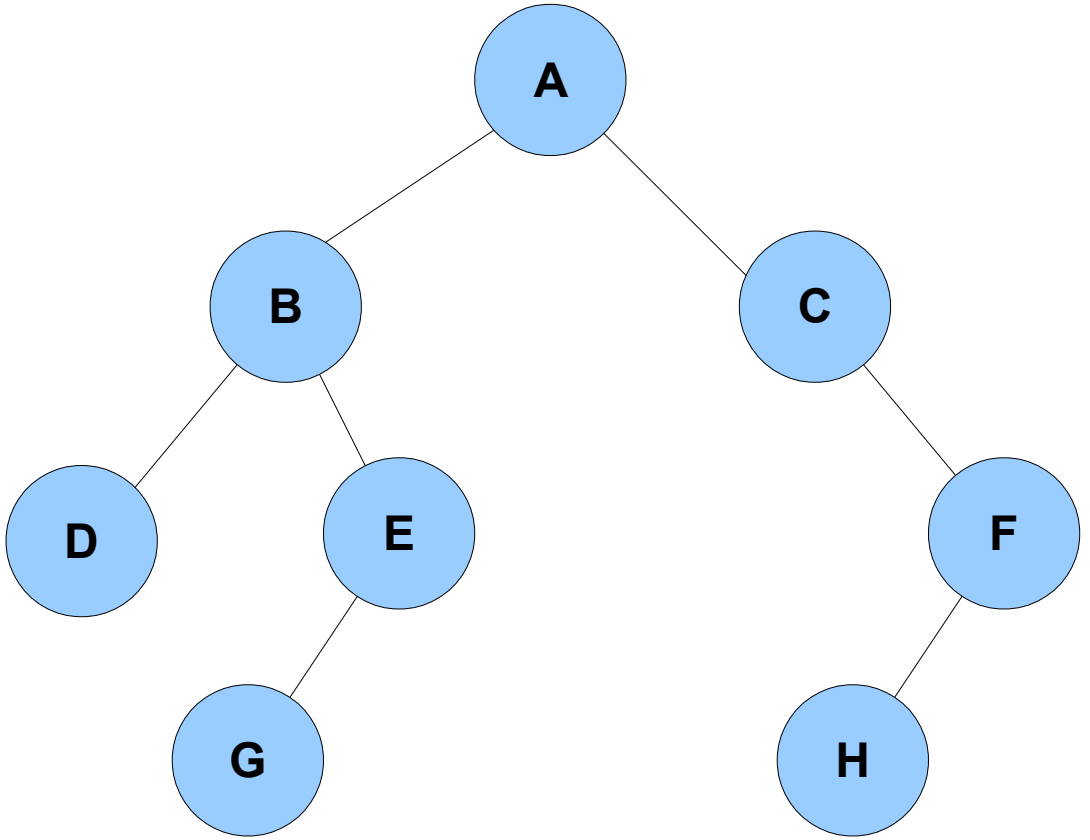


Exemplo



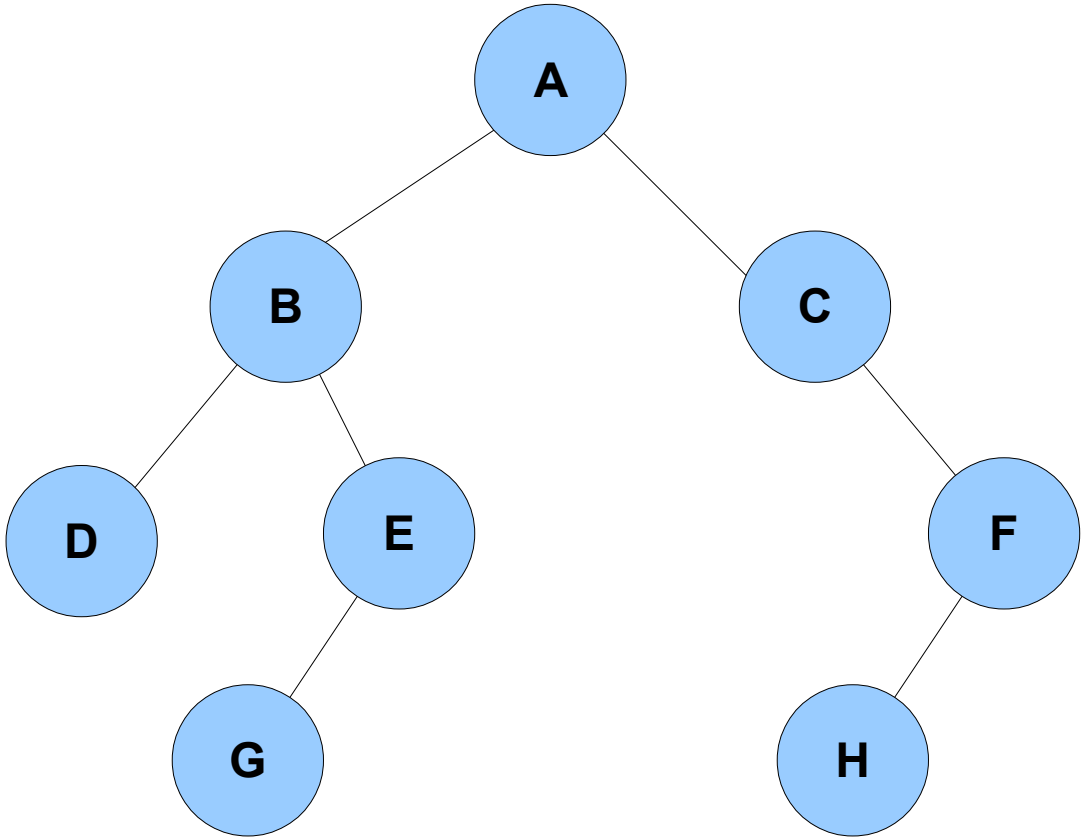
B A C

Percurso em-ordem



? ? ?

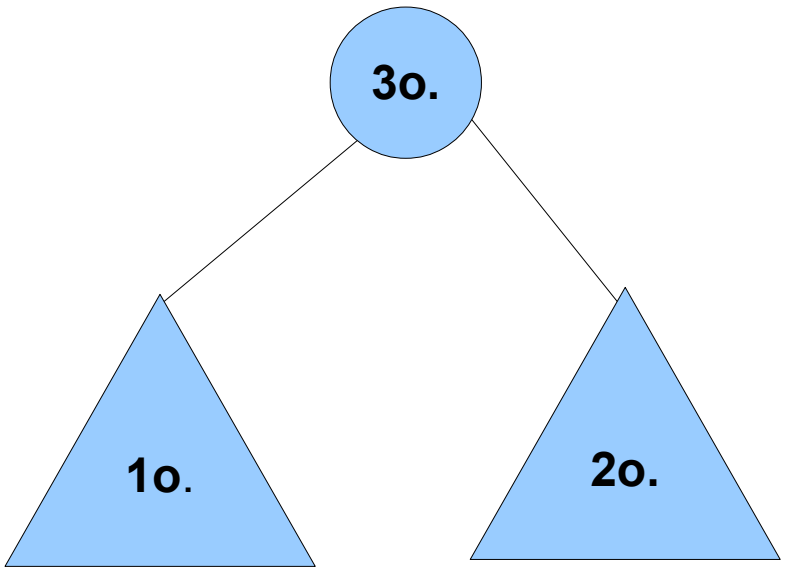
Percurso em-ordem



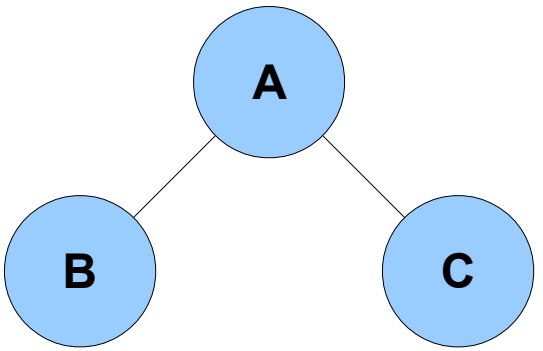
DBGEACHF

Percurso em pós-ordem

```
inicio
    se árvore não é vazia então
        Percorrer a subárvore esquerda
        Percorrer a subárvore direita
        Visitar o nodo raiz
    fim_se
fim
```

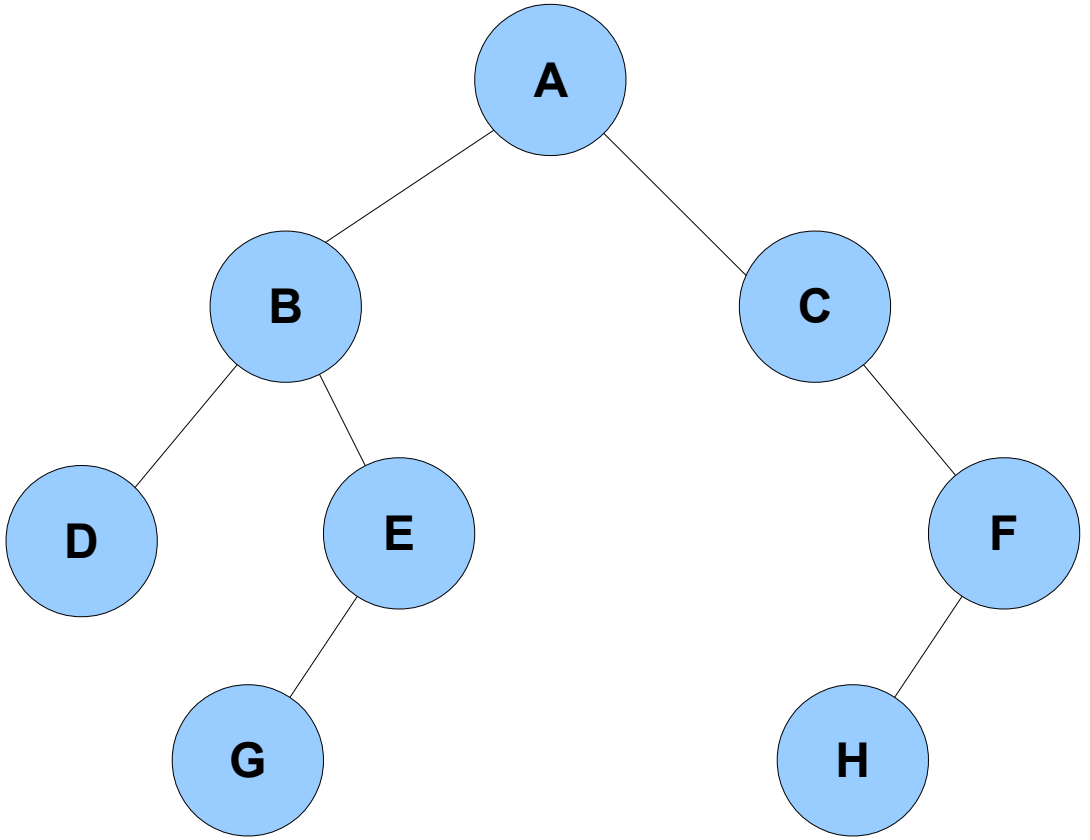


Exemplo



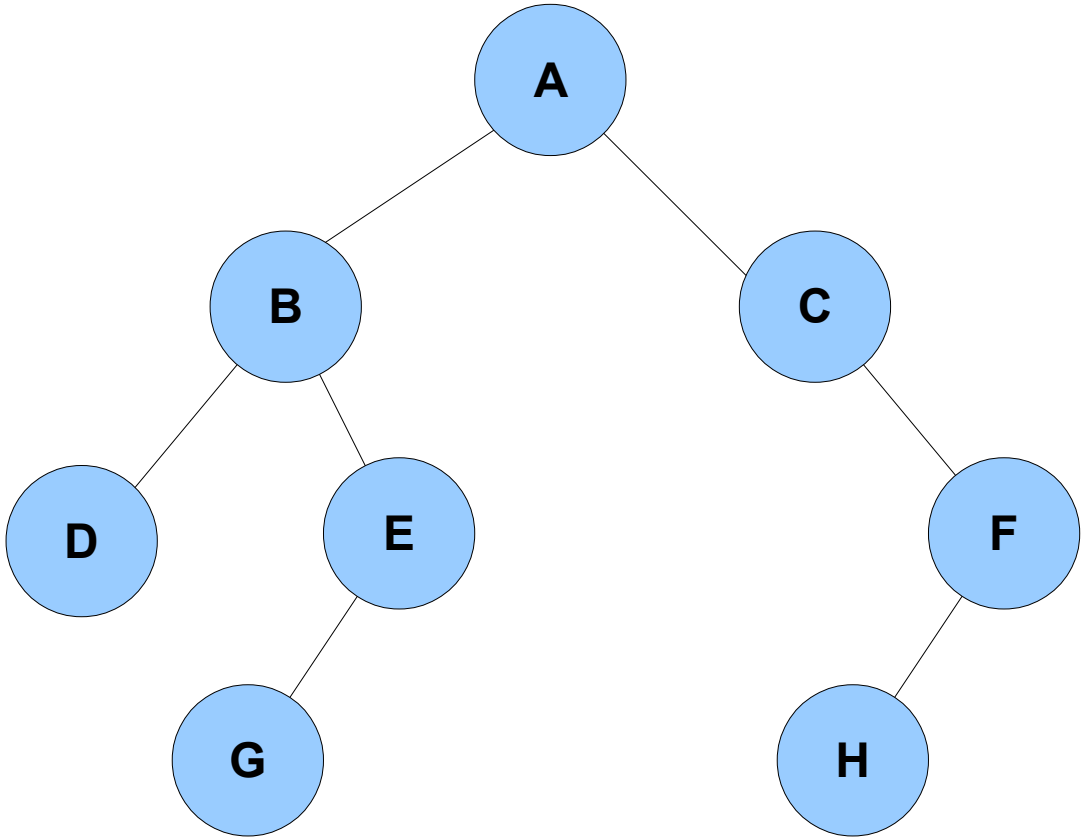
B C A

Percurso em pós-ordem



? ? ?

Percurso em pós-ordem

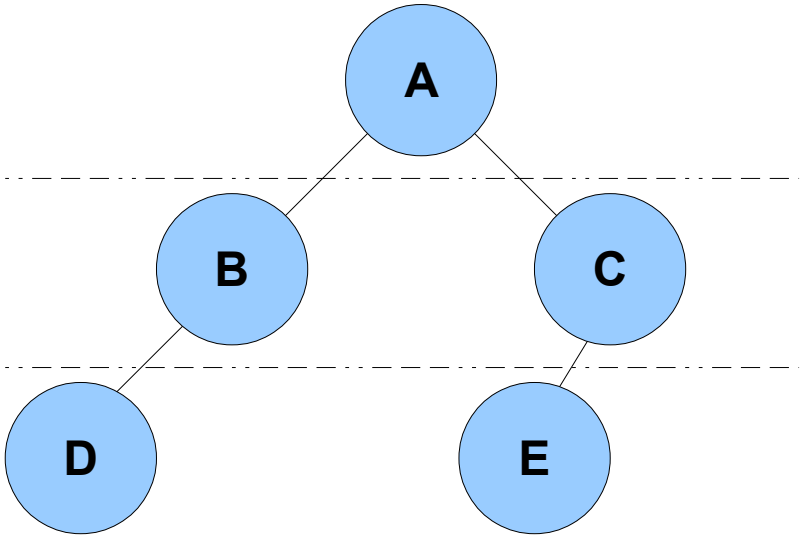


DGEBHFCA

Percurso em nível

```
inicio
    Inserir o nodo raiz em uma fila
    Enquanto a fila não estiver vazia
        Retirar o nodo T da fila
        Visitar T
        Adicionar os filhos de T na fila
    fim_Enquanto
fim
```

Exemplo

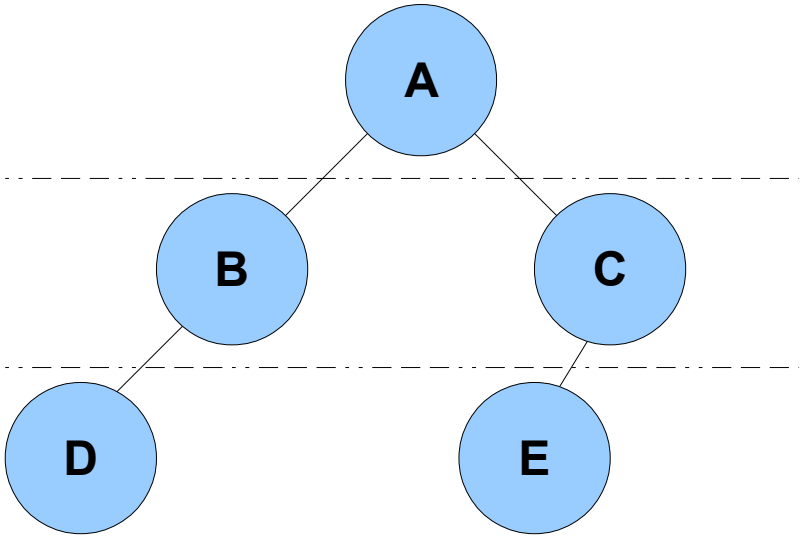


A B C D E

Percurso em nível

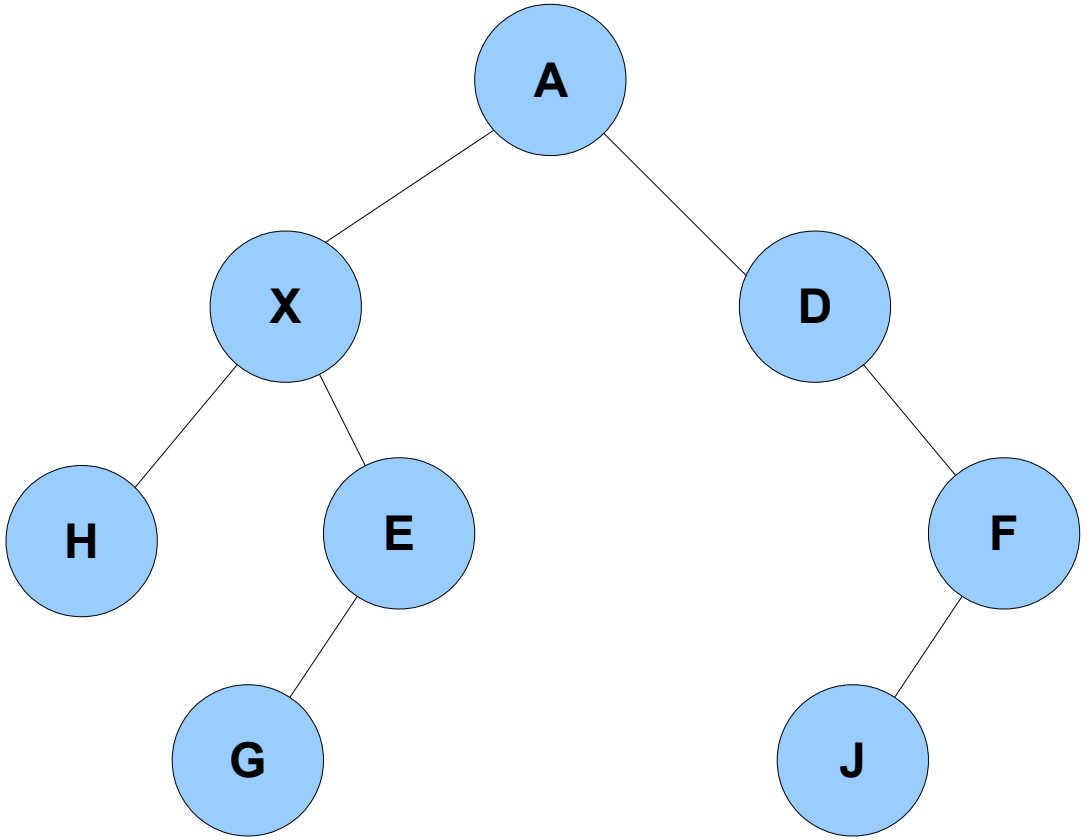
```
inicio
    Inserir o nodo raiz em uma fila
    Enquanto a fila não estiver vazia
        Retirar o nodo T da fila
        Visitar T
        Adicionar os filhos de T na fila
    fim_Enquanto
fim
```

Exemplo



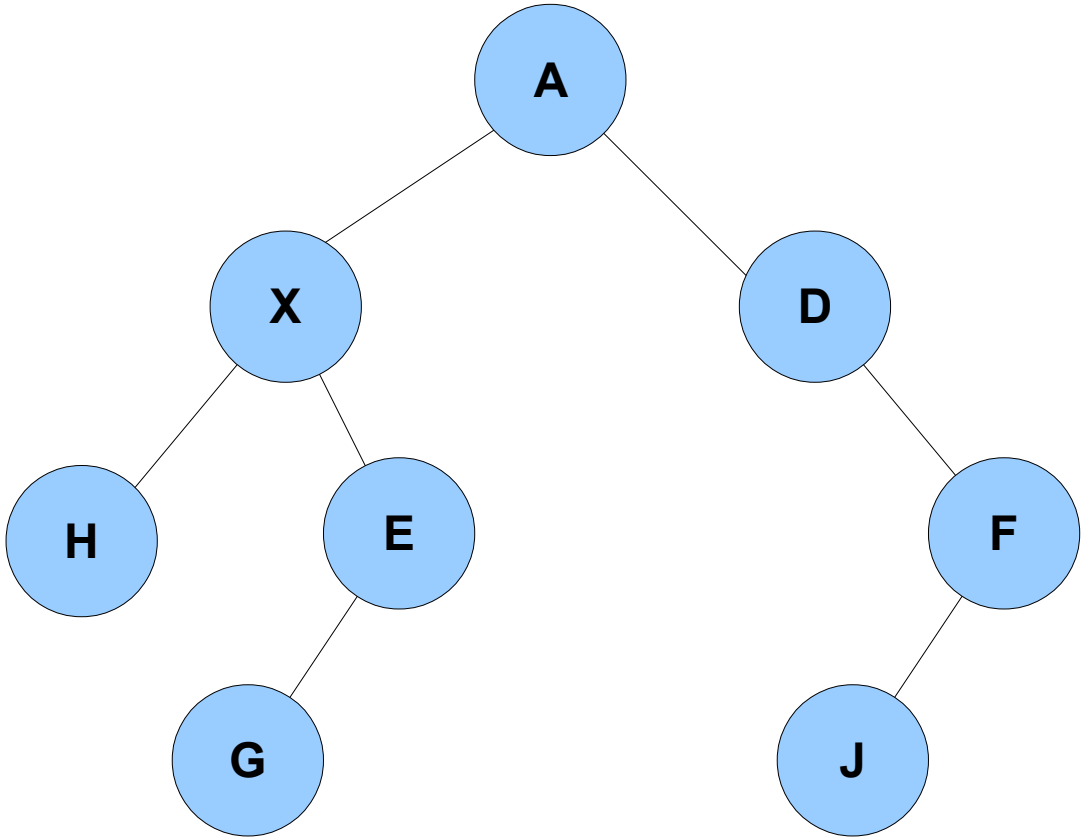
A B C D E

Percurso em nível



? ? ?

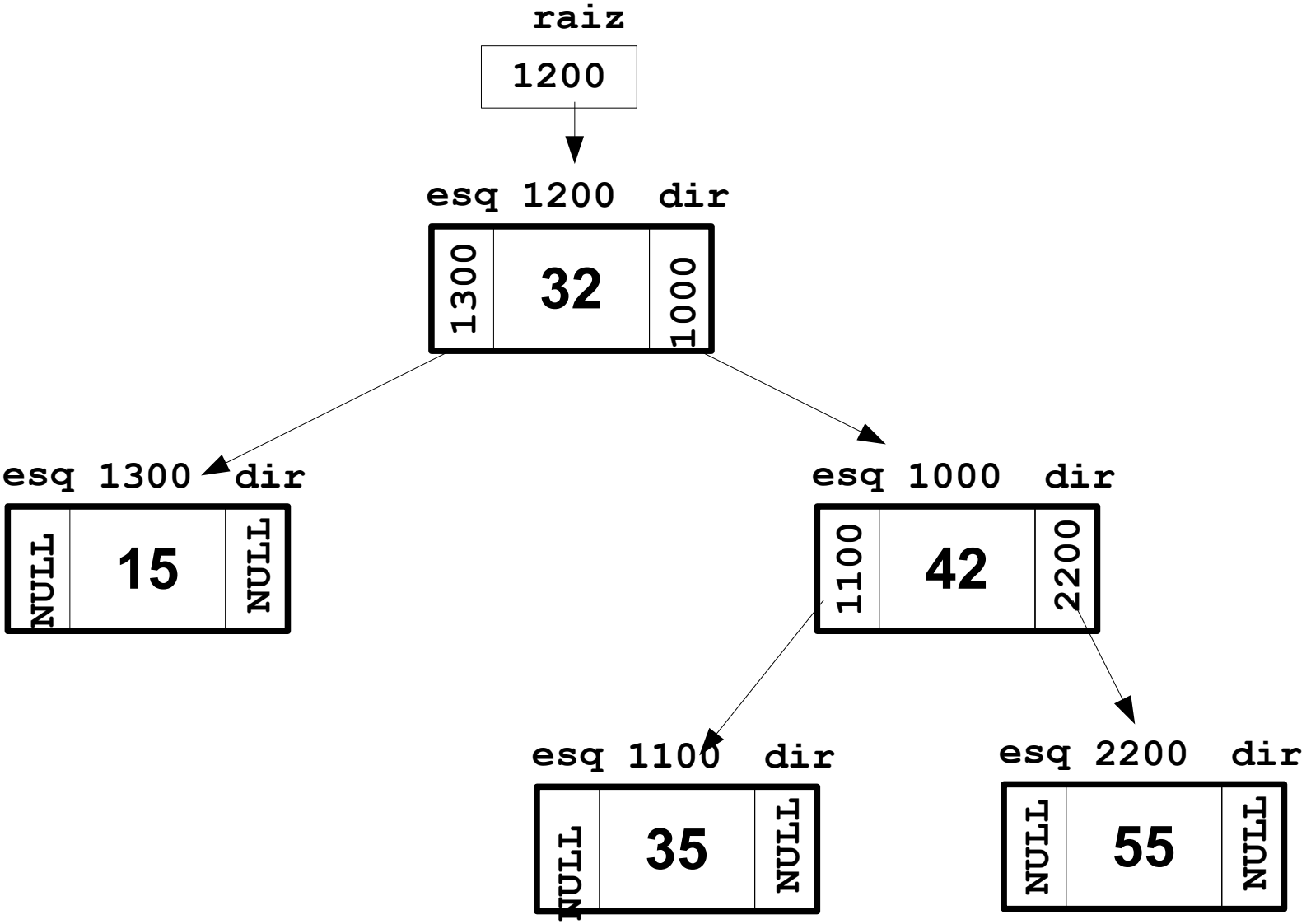
Percurso em nível



AXDHEFGJ

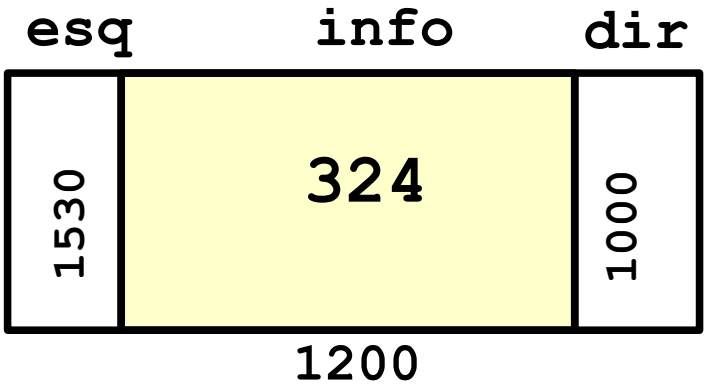
Árvores - Implementação

Representação por encadeamento:



Árvores - Implementação

Nodo para armazenar um inteiro



esq: contém o endereço do nodo filho a esquerda (**NULL** se não possui filho a esquerda).

info : contém a informação armazenada (Ex: um inteiro).

dir: contém o endereço do nodo filho a direira (**NULL** se não possui filho a direita).

Árvores

```
typedef struct {
   int cod;
   float sal;
} Dado;

typedef struct nodo Nodo;

struct nodo {
   Dado info;
   Nodo *esq;
   Nodo *dir;
};

typedef struct {
    Nodo *raiz;
} Arvore;
```

Árvore de busca binária

Os dados são distribuídos pelos nodos de forma a facilitar a pesquisa de um determinado elemento.

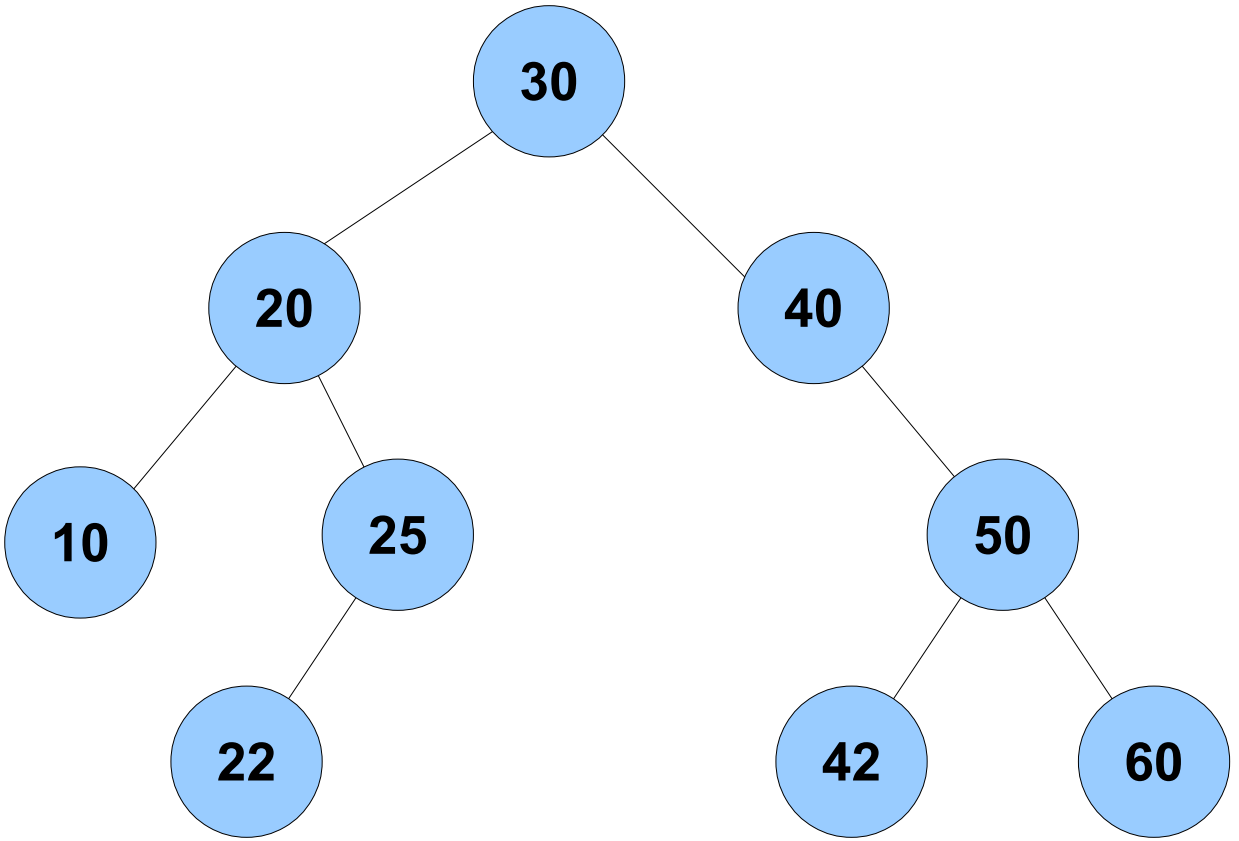
Uma árvore binária, cuja raiz armazena o elemento R, é denominada árvore de busca binária se:

1) Todo elemento armazenado na subárvore esquerda é menor que R.

2) Nenhum elemento armazenado na subárvore direita é menor que R.

3) As árvores esquerda e direita também são árvores de busca binária.

Árvore de busca binária

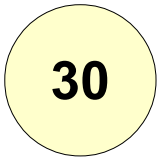


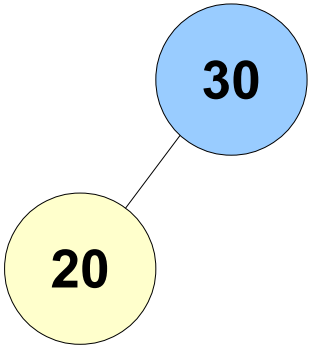
Árvore de busca binária - inserção

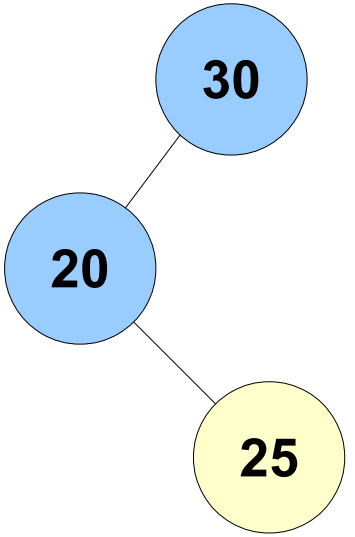
```
início
   se a árvore está vazia então
      inserir o nodo
   senão
      se o dado do nodo que será inserido é menor
         que o dado armazenado no nodo raiz então
         inserir o nodo na subárvore esquerda
      senão
         inserir o nodo na subárvore direita
      fim_se
   fim_se
fim
```

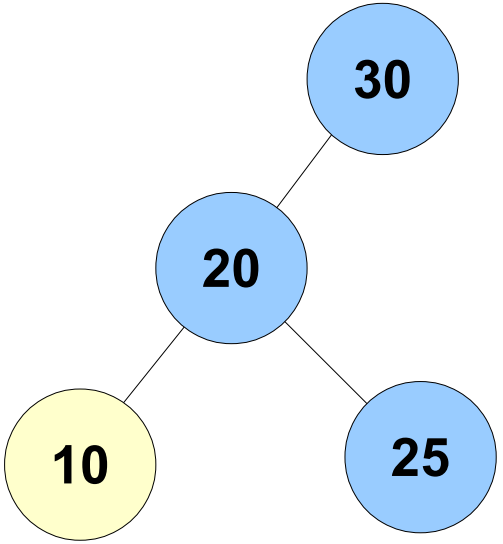
Árvore de busca binária

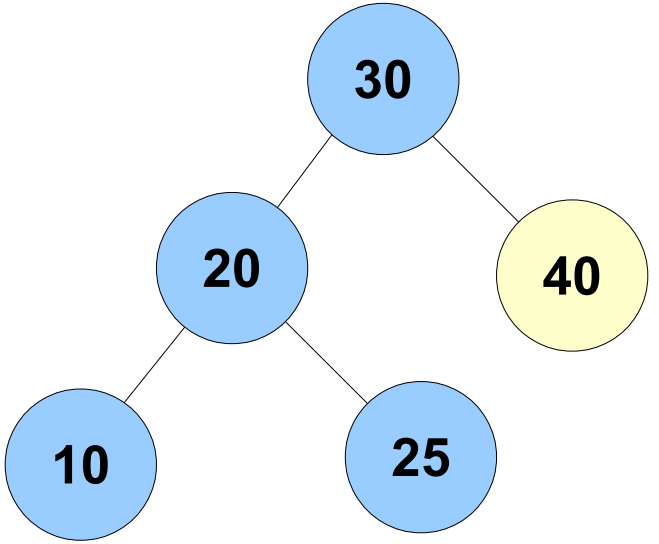
Inserir os nodos: **30 20 25 10 40** em uma árvore vazia.











Árvore de busca binária - procura

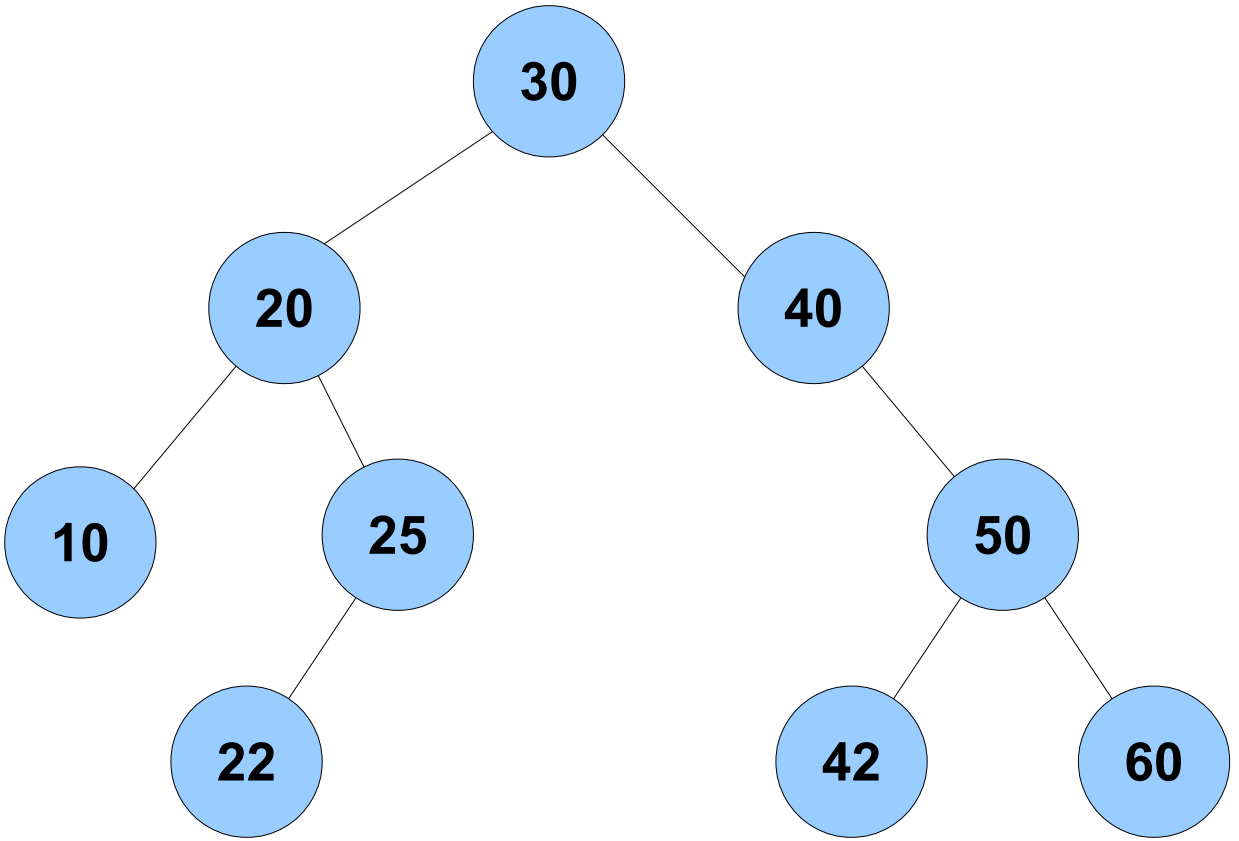
```
início
   se a árvore está vazia então
     nodo não encontrado
   senão
     se a raiz armazena o elemento procurado então
        nodo encontrado
     senão
        se o valor procurado for menor que o valor
                          armazenado no nodo raiz então
             procurar a partir da subárvore esquerda
        senão
             procurar a partir da subárvore direita
        fim_se
     fim_se
   fim_se
fim
```

Árvore de busca binária

Procurar os nodos:

27

25

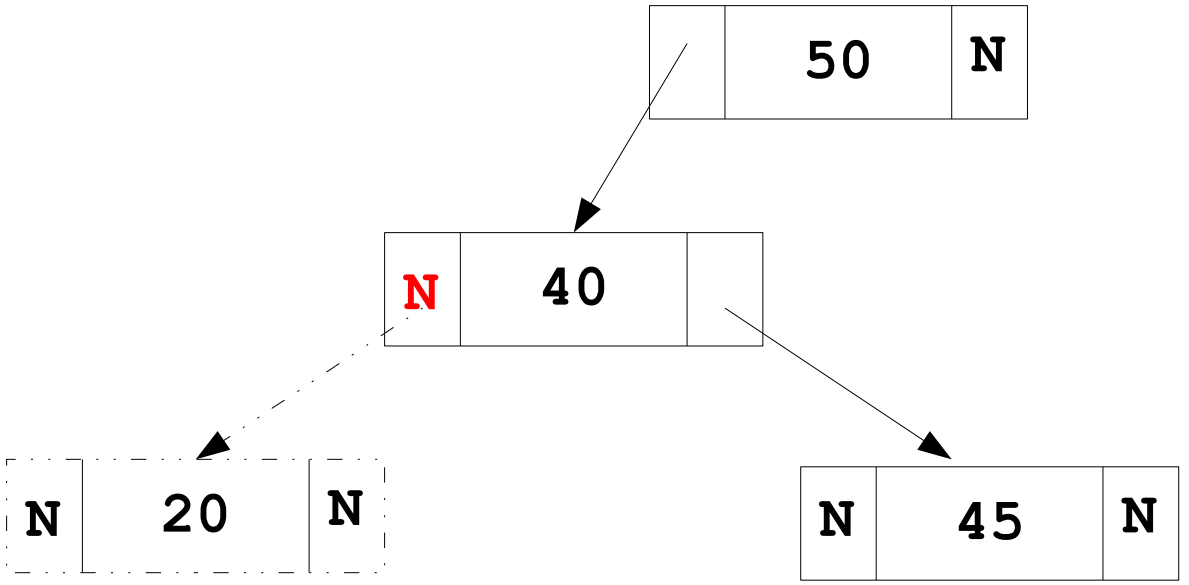


Árvore de busca binária - remoção

Procurar o elemento a ser removido. Caso seja encontrado:

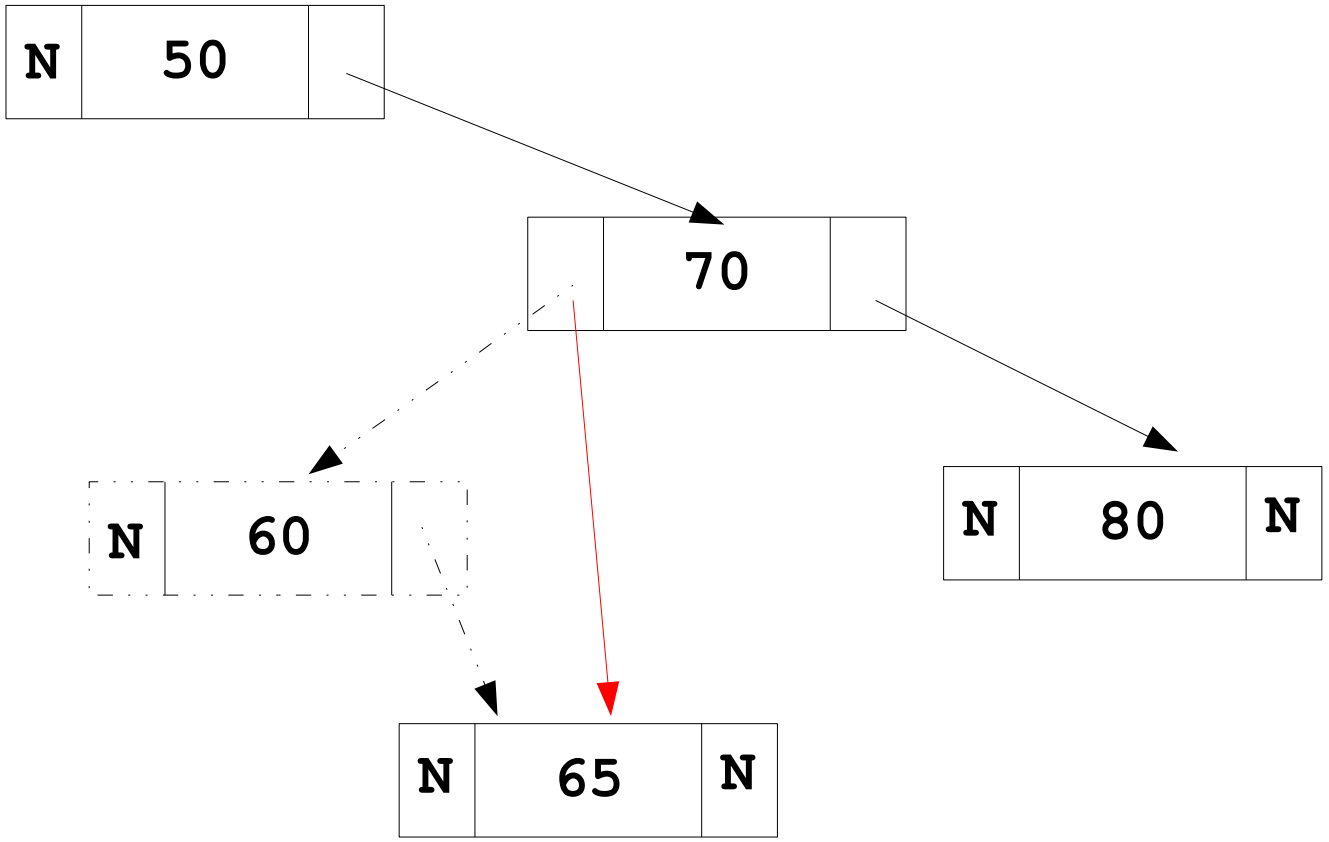
Existem 3 casos:

1) O elemento removido não possui filhos. Remover o nodo e tornar nula a raiz da subárvore.



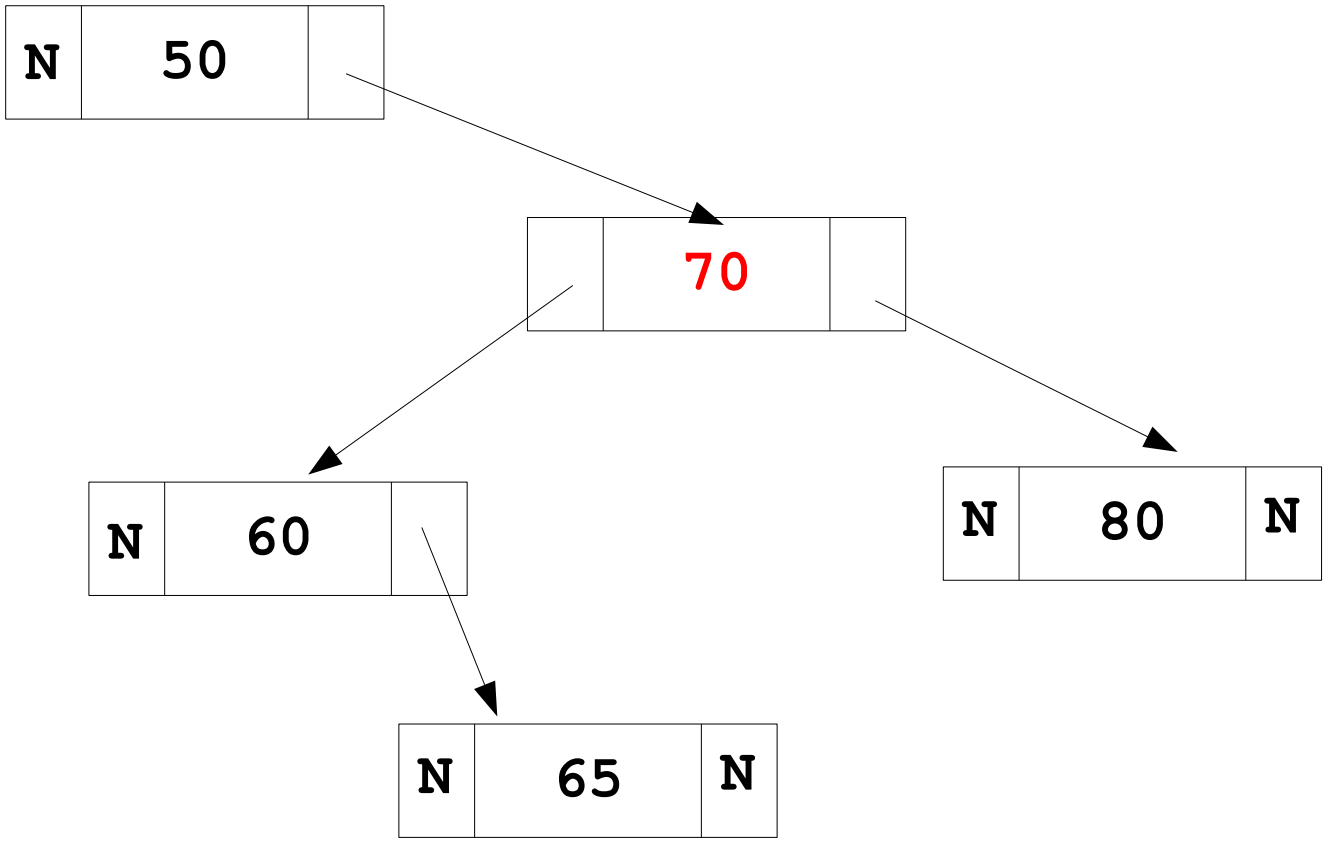
Árvore de busca binária - remoção

2) O elemento removido possui um filho. Remover o nodo substituindo-o pelo seu nodo filho.

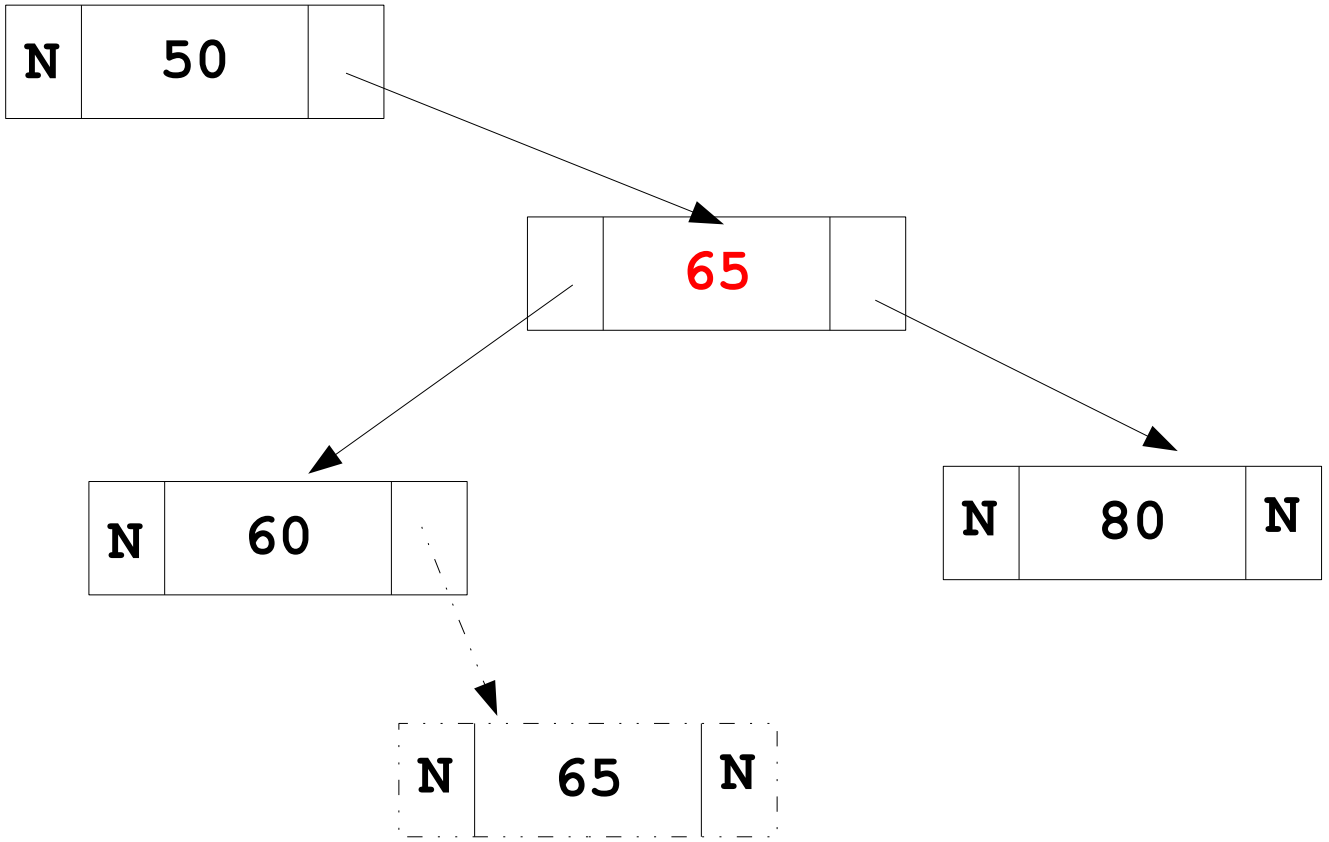


Árvore de busca binária - remoção

3) O elemento removido possui dois filhos. Procurar o nodo que armazena o maior elemento na subárvore esquerda. Este nodo será removido e o elemento armazenado será copiado para o nodo excluído.



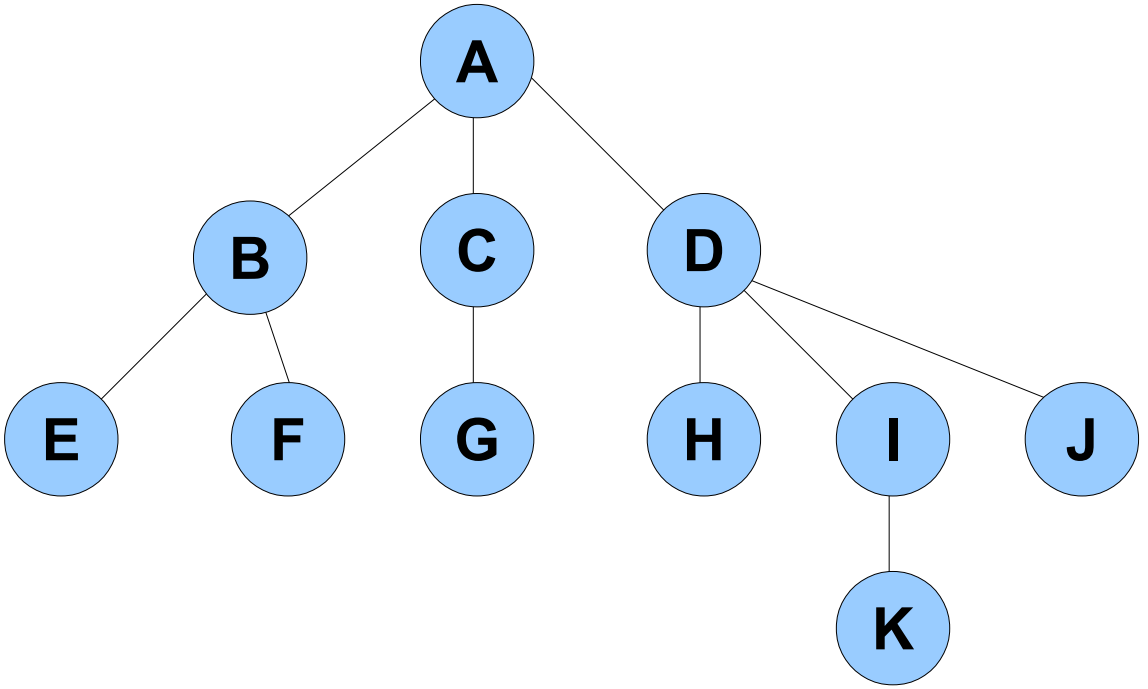
Árvore de busca binária - remoção



Árvores de grau qualquer

Para representar árvores de grau qualquer utilizar uma árvore binária fornecendo aos ponteiros **esq** e **dir** significados diferentes:

esq : passa a armazenar o endereço do primeiro filho.
dir: passa a armazenar o endereço do irmão.



Árvores de grau qualquer