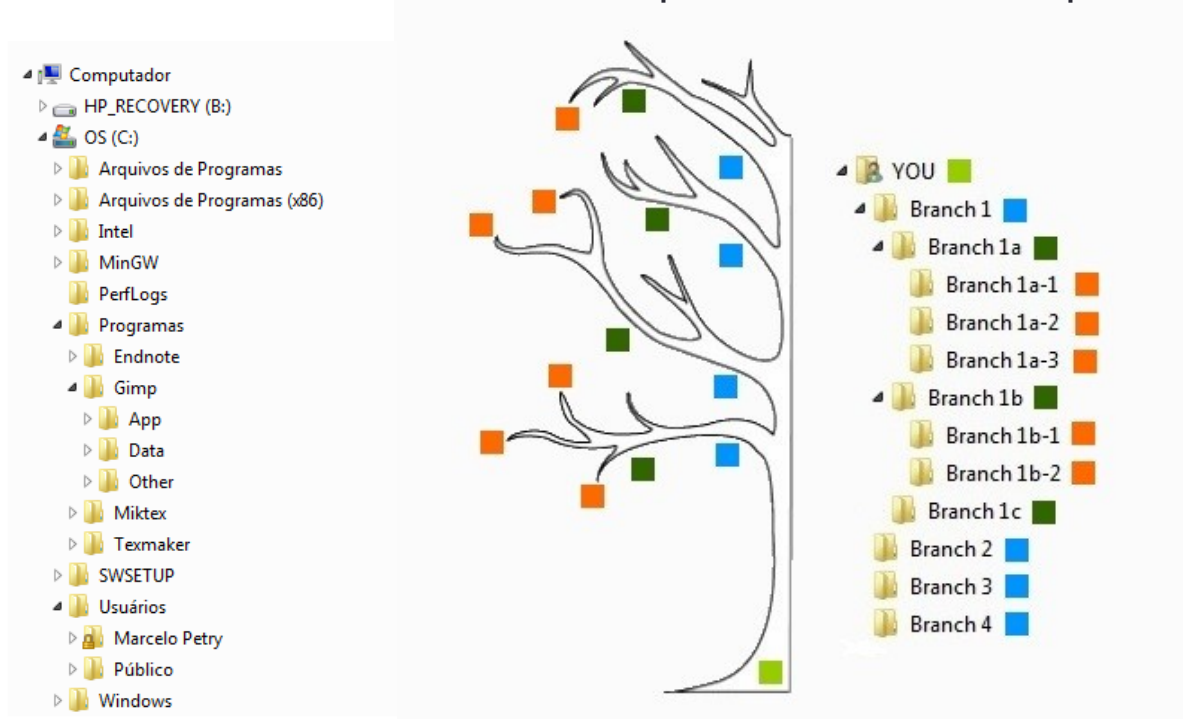


Algoritmos e estruturas de Dados

Árvores

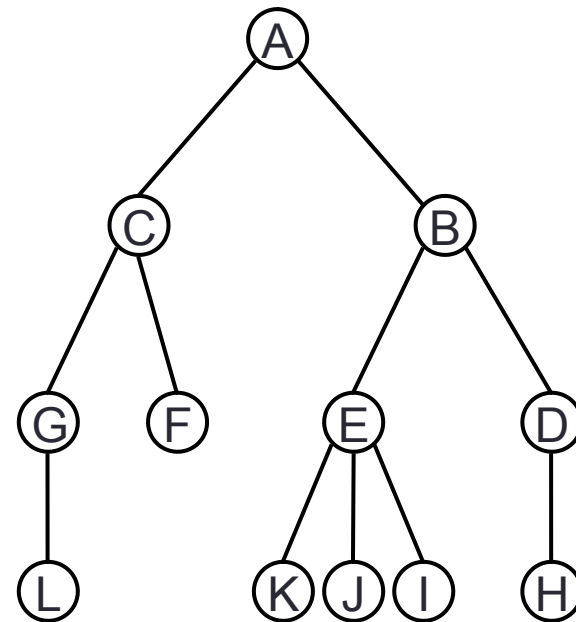
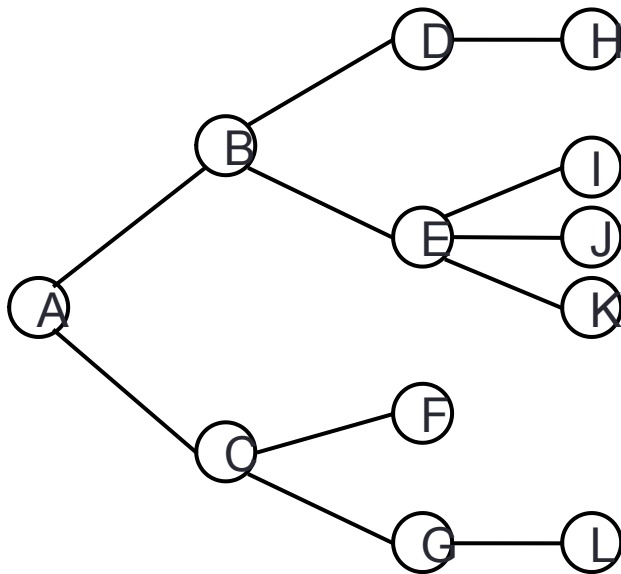
Árvores

- Vetores e listas são estruturas de dados importantes
- Mas por serem lineares não são adequadas para representar dados dispostos de maneira hierárquica
 - Ex: documentos armazenados em pastas em um computador



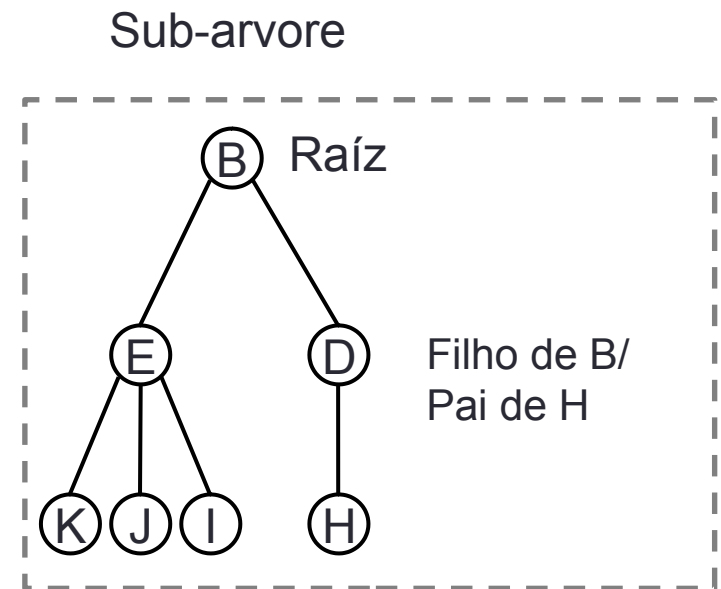
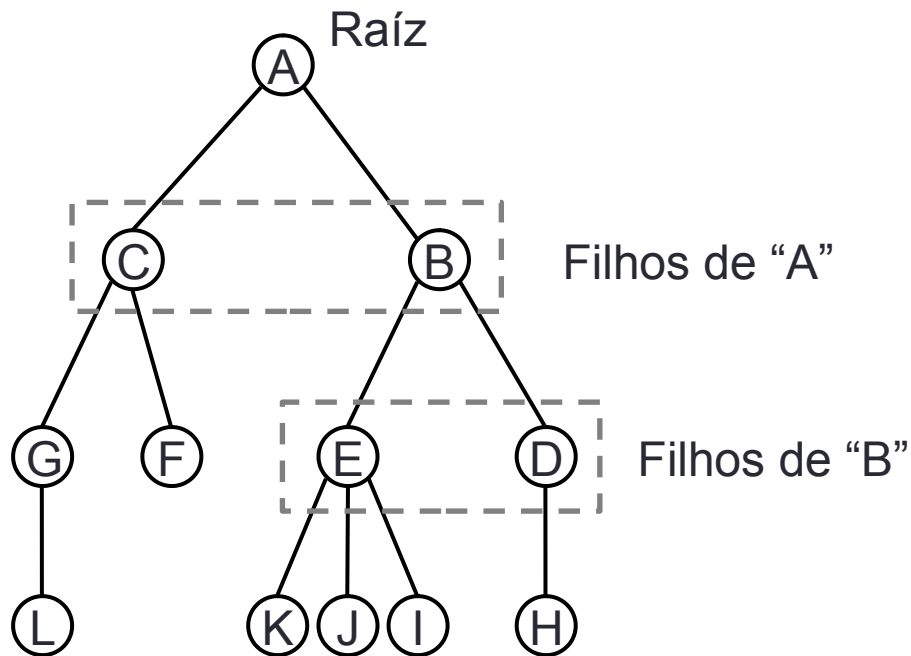
Árvores

- Uma árvore é uma estrutura na qual cada elemento possui zero ou mais sucessores.
- Porém todos os elementos possuem apenas um antecessor



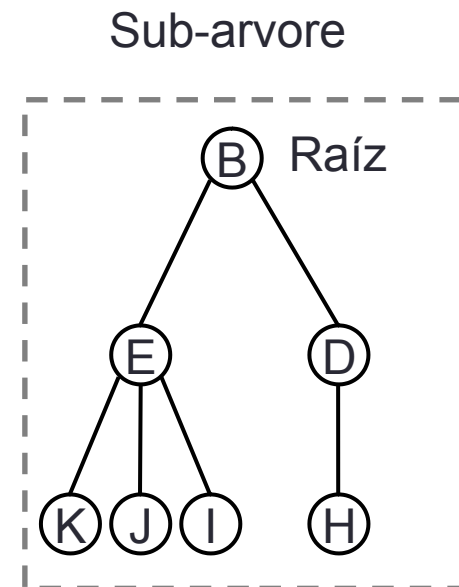
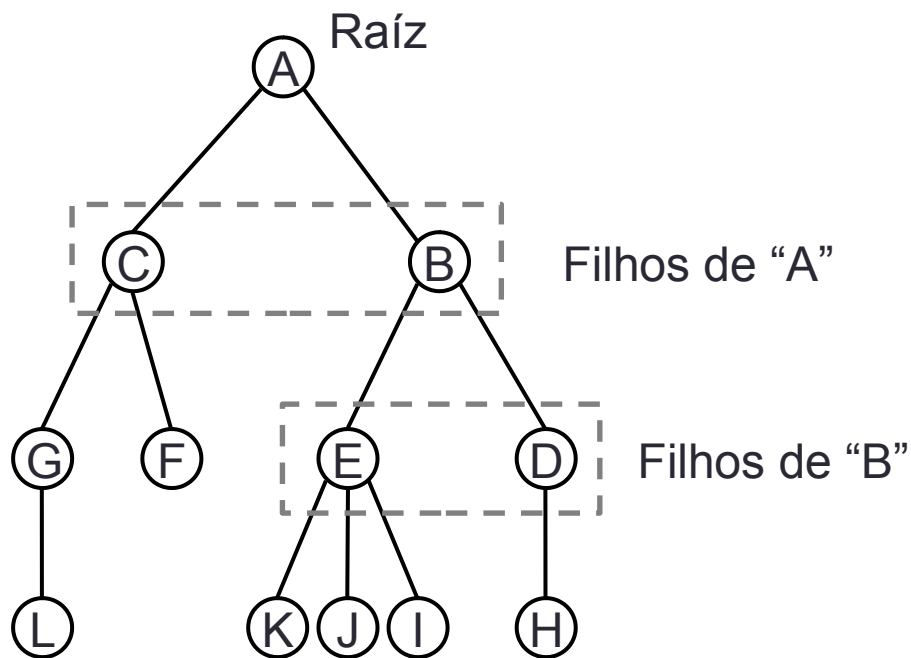
Árvores

- Qualquer elemento de uma árvore é chamado de nó.
- O 1º elemento, que dá origem aos demais, é chamado **Raíz**
- Os sucessores de um determinado nó são chamados **Filhos** (ou descendentes)
- O antecessor de um nó é chamado de **Pai**.



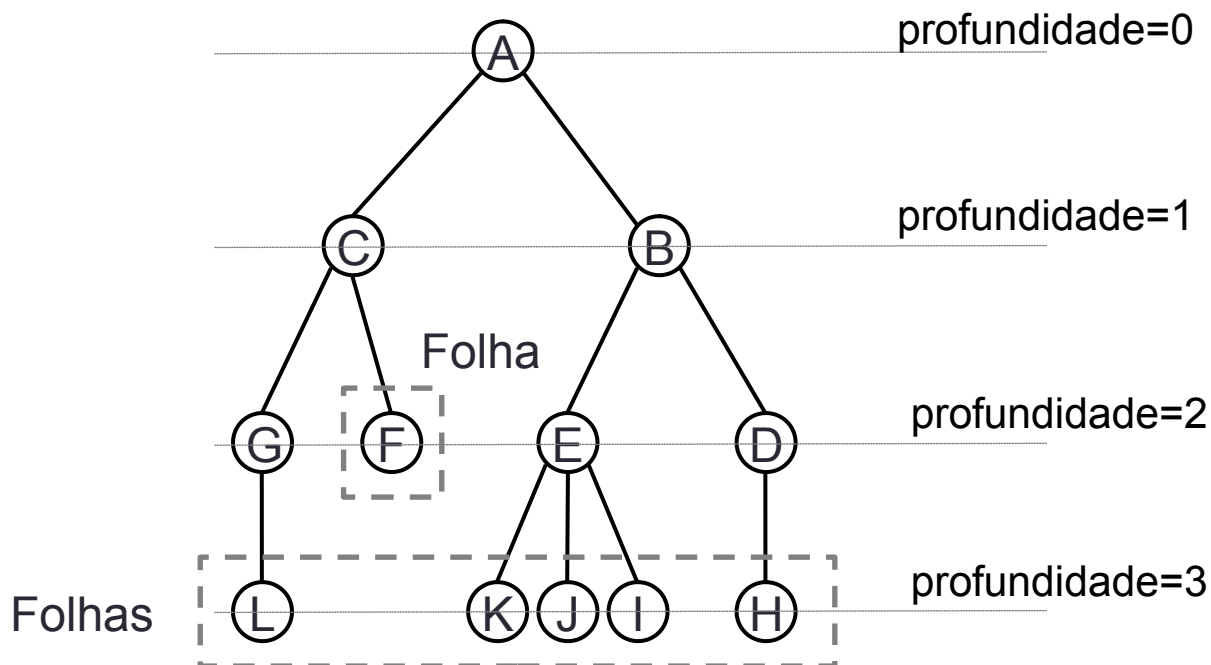
Árvores

- À exceção da raiz, todos os nós de uma árvore têm 1 (e apenas 1) pai.
 - A raiz não tem pai.
- Há um caminho único da raiz a cada nó.
 - O tamanho do caminho é o número de arestas a percorrer



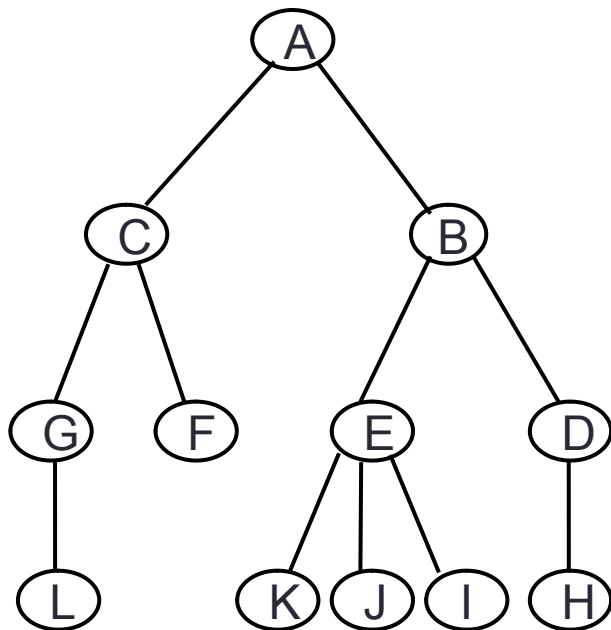
Árvores

- **Folha:** nó sem filhos.
- **Profundidade** de um nó: Comprimento da raiz até o nó
 - Profundidade da raiz é 0
 - Profundidade de um nó é $1 + \text{profundidade do pai}$



Árvores

- **Tamanho** de um nó: número de descendentes



Os nós G e D têm tamanho 1

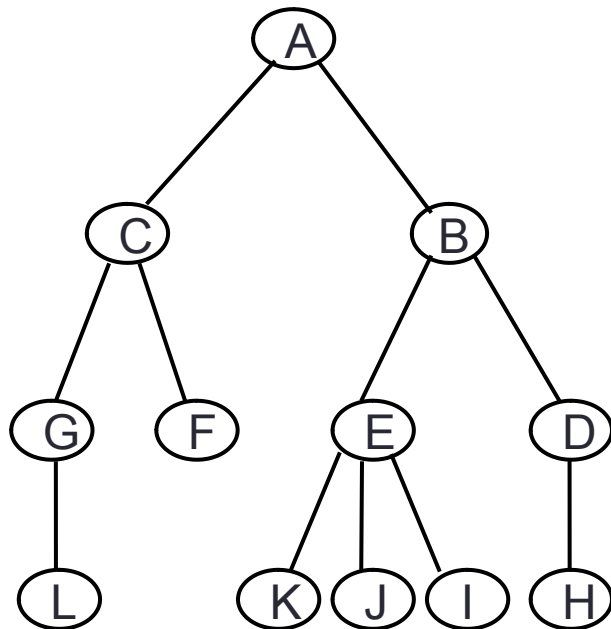
Os nós A, B e C têm tamanho 2

O nó E tem tamanho 3

Os nós L, F, K, J, I e H têm tamanho zero

Árvores

- **Altura** de um nó: comprimento do nó até à folha de maior profundidade
 - Altura de uma folha é 0
 - Altura de um nó é $1 +$ a altura do seu filho de maior altura
- **Altura da árvore: altura da raiz**



O nó A tem altura 3

Os nós C e B têm altura 2

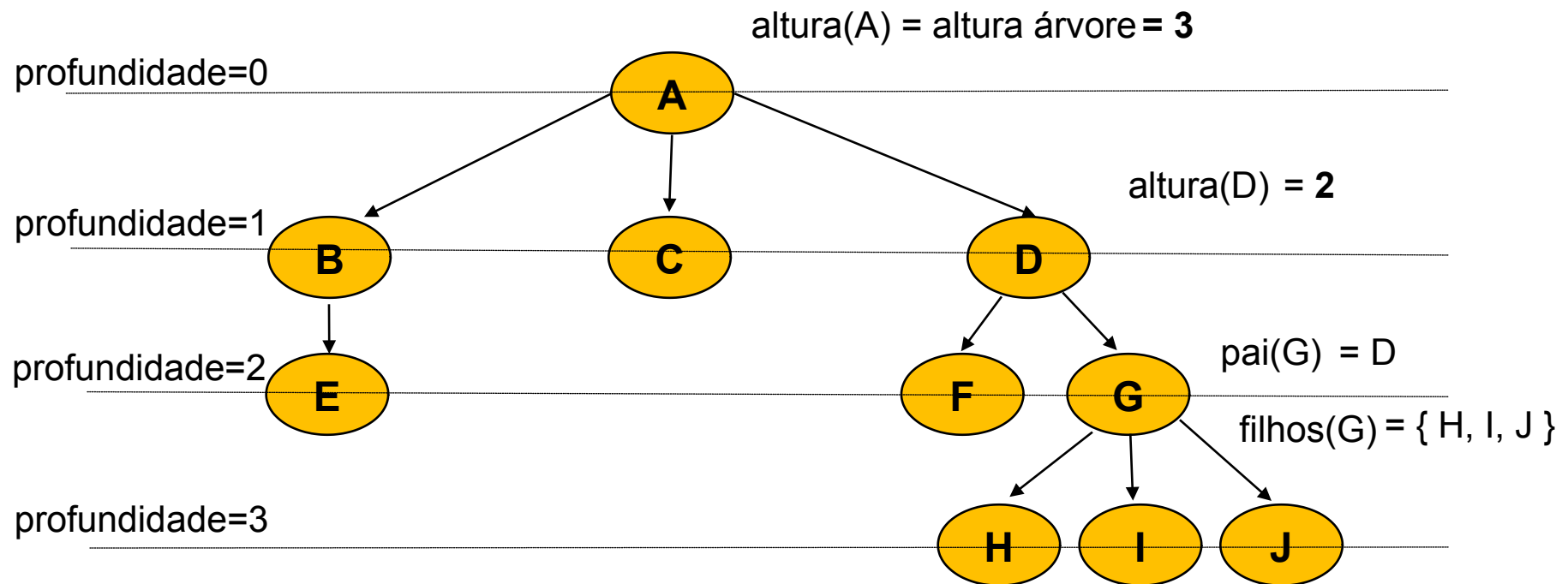
Os nós G, E e D têm altura 1

Os nós L, F, K, J, I e H têm altura zero

A árvore tem altura 3

Árvores

Exemplo:

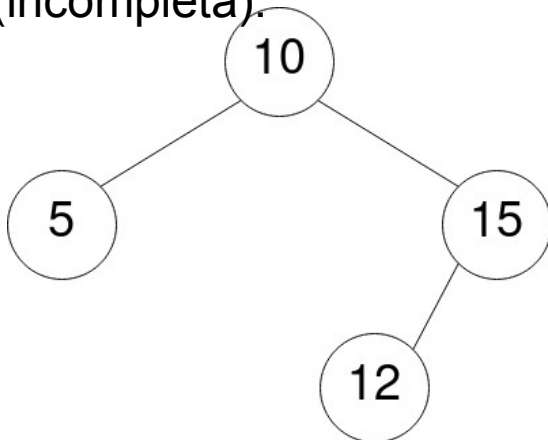


Árvores Binárias

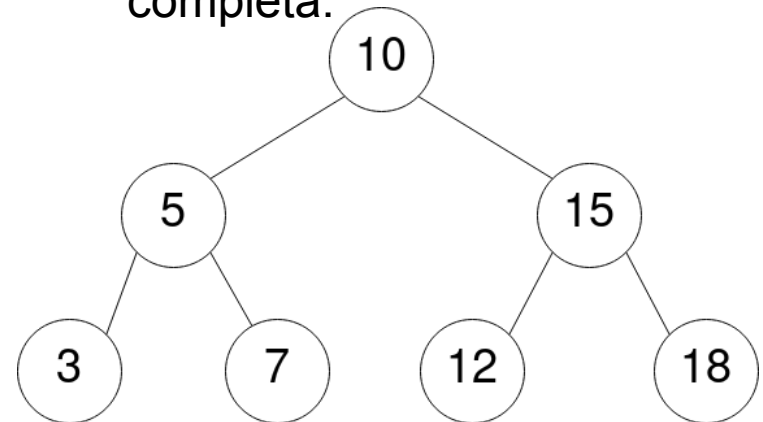
Definição: árvore em que cada nó tem no máximo 2 filhos.

Árvore binária **completa**: os nós de todos os níveis – exceto o último – têm dois filhos, enquanto os nós do último nível são folhas.

Árvore binária
(incompleta):



Árvore binária
completa:



Árvores binárias – relação entre altura e quantidade de nós

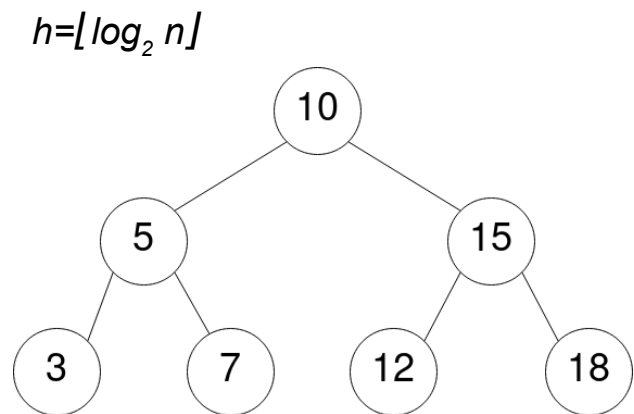
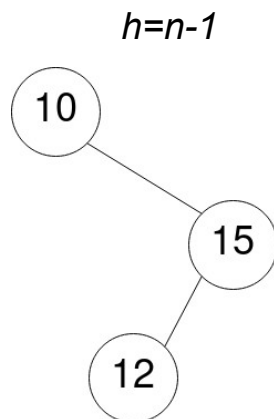
Uma árvore binária com altura h tem n nós, tal que $n-1 \leq h < \log_2 n$

Para $h=n-1$: cada nó tem um filho

Para $h=\lfloor \log_2 n \rfloor$: a árvore é completa.

Exemplos

:



Percurso em Árvores Binárias

Em algumas aplicações, é necessário percorrer uma árvore de forma sistemática, visitando cada nó da árvore uma única vez, em determinada ordem.

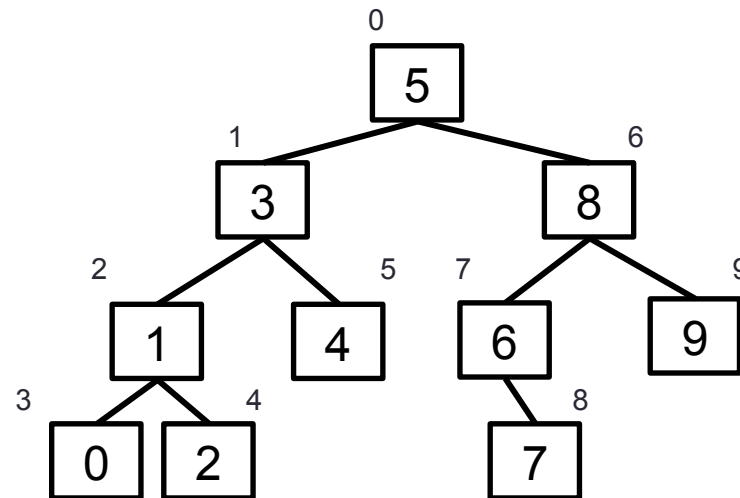
- Pré-ordem
- Em-ordem (ordem simétrica)
- Pós-ordem
- Por nível

Árvores Binárias

- **Pré-ordem (r-e-d):**

1. Visitar a raiz
2. Percorrer a sub-árvore esquerda em pré-ordem
3. Percorrer a sub-árvore direita em pré-ordem

O percurso em pré-ordem segue os nós até chegar os mais “profundos”, em “ramos” de subárvores da esquerda para a direita. É conhecida usualmente pelo nome de percurso em profundidade (**depth-first**).



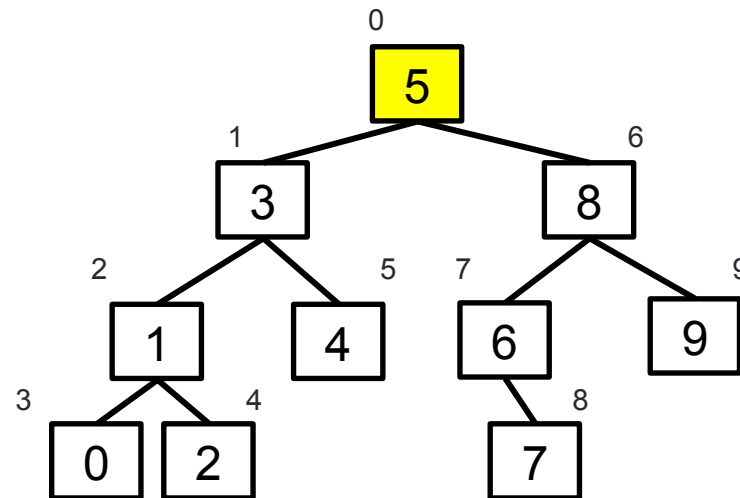
Percurso em pré-ordem: 5 3 1 0 2 4 8 6 7 9

Árvores Binárias

- **Pré-ordem (r-e-d):**

1. Visitar a raiz
2. Percorrer a sub-árvore esquerda em pré-ordem
3. Percorrer a sub-árvore direita em pré-ordem

O percurso em pré-ordem segue os nós até chegar os mais “profundos”, em “ramos” de subárvores da esquerda para a direita. É conhecida usualmente pelo nome de percurso em profundidade (**depth-first**).



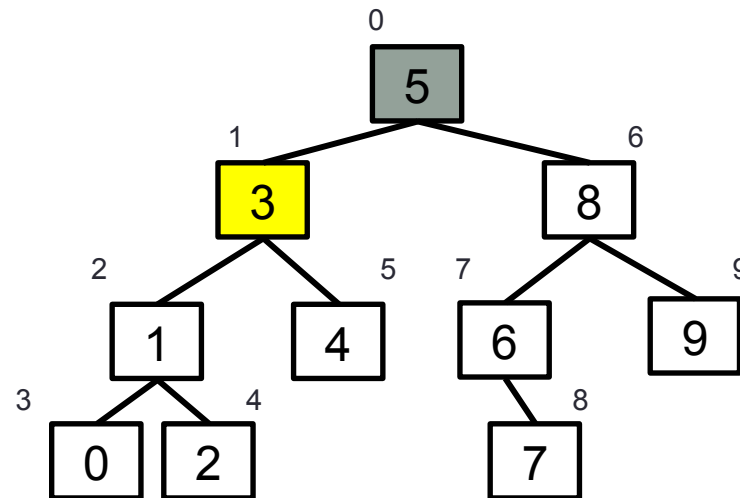
Percurso em pré-ordem: 5

Árvores Binárias

- **Pré-ordem (r-e-d):**

1. Visitar a raiz
2. Percorrer a sub-árvore esquerda em pré-ordem
3. Percorrer a sub-árvore direita em pré-ordem

O percurso em pré-ordem segue os nós até chegar os mais “profundos”, em “ramos” de subárvores da esquerda para a direita. É conhecida usualmente pelo nome de percurso em profundidade (**depth-first**).



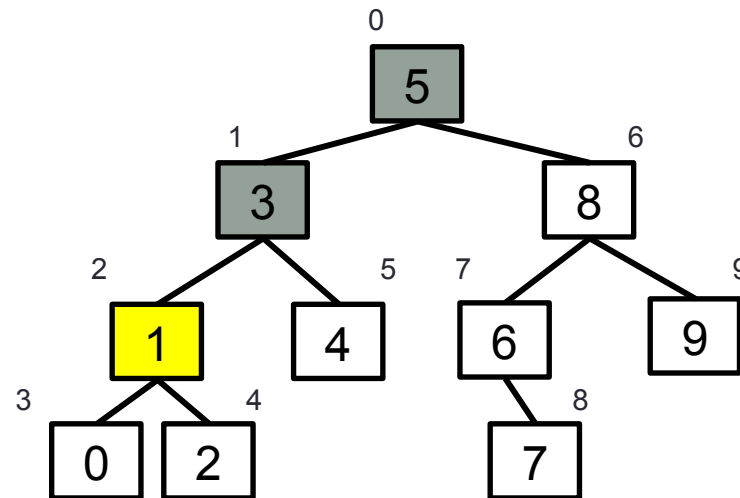
Percurso em pré-ordem: 5 3

Árvores Binárias

- **Pré-ordem (r-e-d):**

1. Visitar a raiz
2. Percorrer a sub-árvore esquerda em pré-ordem
3. Percorrer a sub-árvore direita em pré-ordem

O percurso em pré-ordem segue os nós até chegar os mais “profundos”, em “ramos” de subárvores da esquerda para a direita. É conhecida usualmente pelo nome de percurso em profundidade (**depth-first**).



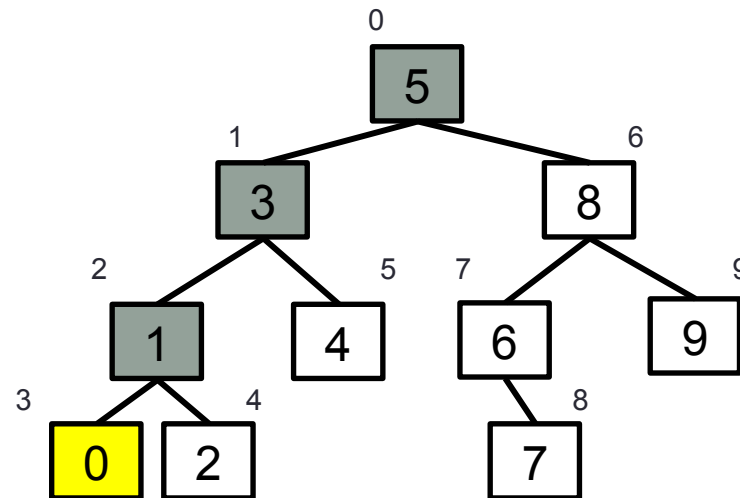
Percurso em pré-ordem: 5 3 1

Árvores Binárias

- **Pré-ordem (r-e-d):**

1. Visitar a raiz
2. Percorrer a sub-árvore esquerda em pré-ordem
3. Percorrer a sub-árvore direita em pré-ordem

O percurso em pré-ordem segue os nós até chegar os mais “profundos”, em “ramos” de subárvores da esquerda para a direita. É conhecida usualmente pelo nome de percurso em profundidade (**depth-first**).



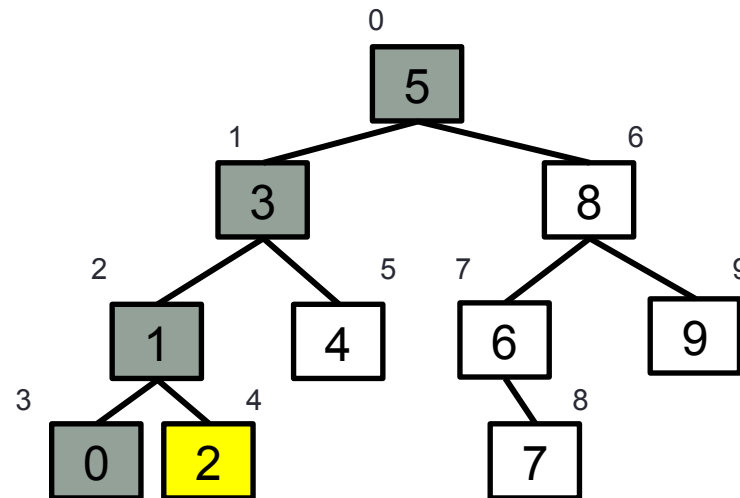
Percurso em pré-ordem: 5 3 1 0

Árvores Binárias

- **Pré-ordem (r-e-d):**

1. Visitar a raiz
2. Percorrer a sub-árvore esquerda em pré-ordem
3. Percorrer a sub-árvore direita em pré-ordem

O percurso em pré-ordem segue os nós até chegar os mais “profundos”, em “ramos” de subárvores da esquerda para a direita. É conhecida usualmente pelo nome de percurso em profundidade (**depth-first**).



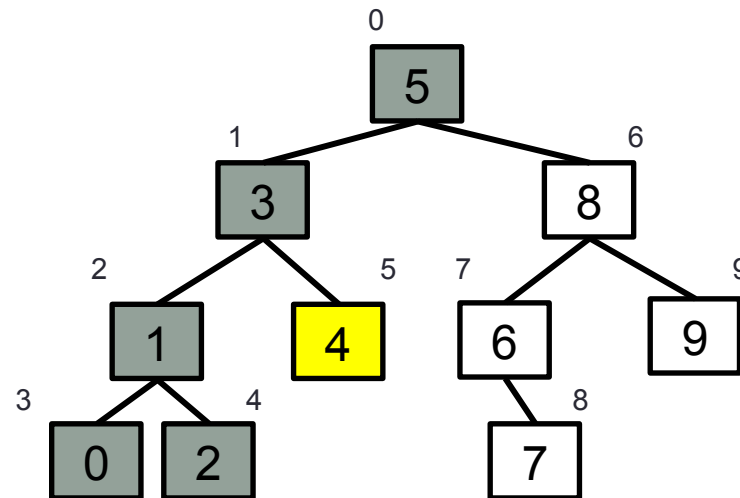
Percurso em pré-ordem: 5 3 1 0 2

Árvores Binárias

- **Pré-ordem (r-e-d):**

1. Visitar a raiz
2. Percorrer a sub-árvore esquerda em pré-ordem
3. Percorrer a sub-árvore direita em pré-ordem

O percurso em pré-ordem segue os nós até chegar os mais “profundos”, em “ramos” de subárvores da esquerda para a direita. É conhecida usualmente pelo nome de percurso em profundidade (**depth-first**).



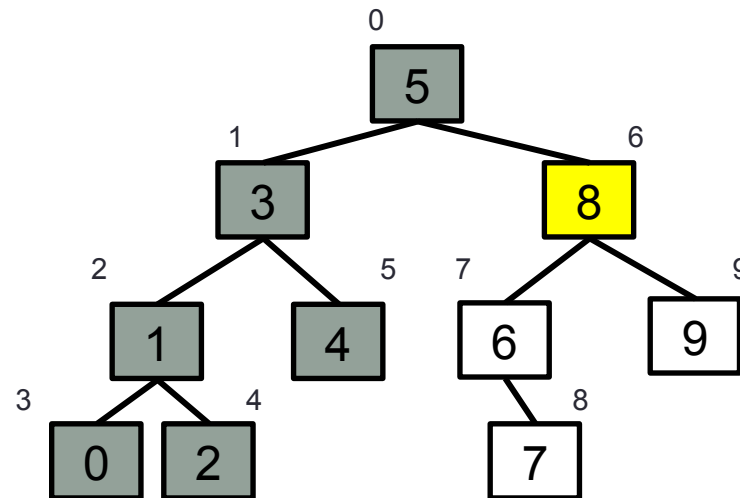
Percurso em pré-ordem: 5 3 1 0 2 4

Árvores Binárias

- **Pré-ordem (r-e-d):**

1. Visitar a raiz
2. Percorrer a sub-árvore esquerda em pré-ordem
3. Percorrer a sub-árvore direita em pré-ordem

O percurso em pré-ordem segue os nós até chegar os mais “profundos”, em “ramos” de subárvores da esquerda para a direita. É conhecida usualmente pelo nome de percurso em profundidade (**depth-first**).



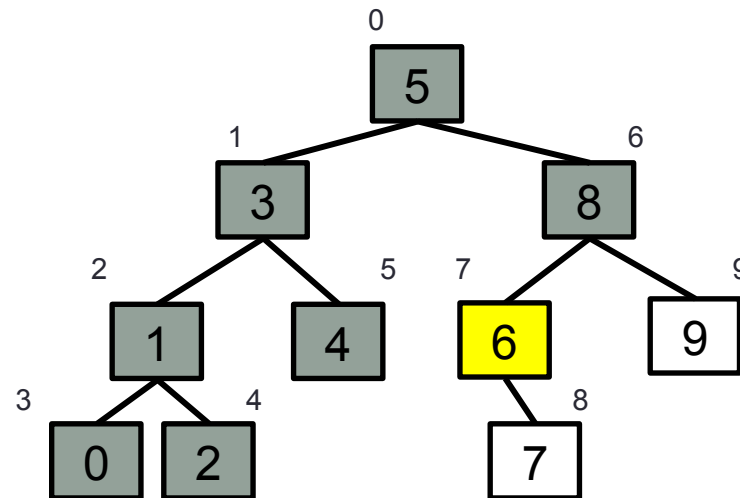
Percurso em pré-ordem: 5 3 1 0 2 4 8

Árvores Binárias

- **Pré-ordem (r-e-d):**

1. Visitar a raiz
2. Percorrer a sub-árvore esquerda em pré-ordem
3. Percorrer a sub-árvore direita em pré-ordem

O percurso em pré-ordem segue os nós até chegar os mais “profundos”, em “ramos” de subárvores da esquerda para a direita. É conhecida usualmente pelo nome de percurso em profundidade (**depth-first**).



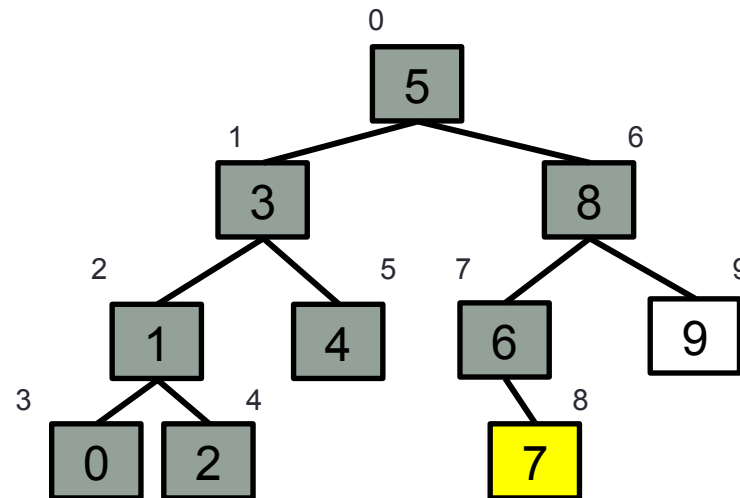
Percurso em pré-ordem: 5 3 1 0 2 4 8 6 7 9

Árvores Binárias

- **Pré-ordem (r-e-d):**

1. Visitar a raiz
2. Percorrer a sub-árvore esquerda em pré-ordem
3. Percorrer a sub-árvore direita em pré-ordem

O percurso em pré-ordem segue os nós até chegar os mais “profundos”, em “ramos” de subárvores da esquerda para a direita. É conhecida usualmente pelo nome de percurso em profundidade (**depth-first**).



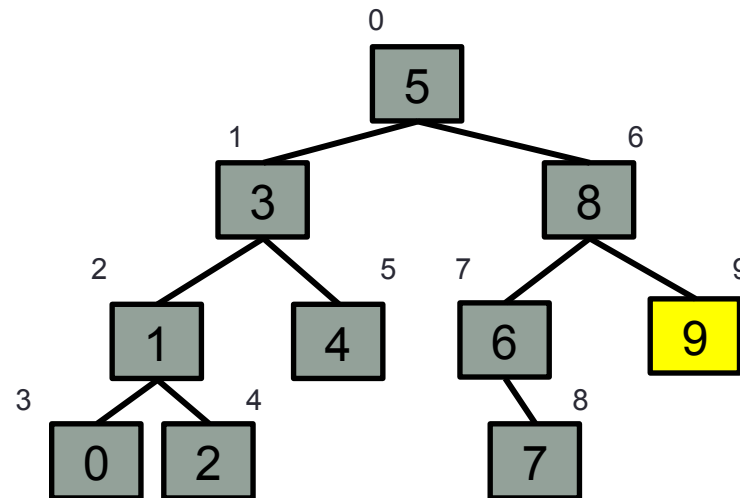
Percurso em pré-ordem: 5 3 1 0 2 4 8 6 7

Árvores Binárias

- **Pré-ordem (r-e-d):**

1. Visitar a raiz
2. Percorrer a sub-árvore esquerda em pré-ordem
3. Percorrer a sub-árvore direita em pré-ordem

O percurso em pré-ordem segue os nós até chegar os mais “profundos”, em “ramos” de subárvores da esquerda para a direita. É conhecida usualmente pelo nome de percurso em profundidade (**depth-first**).



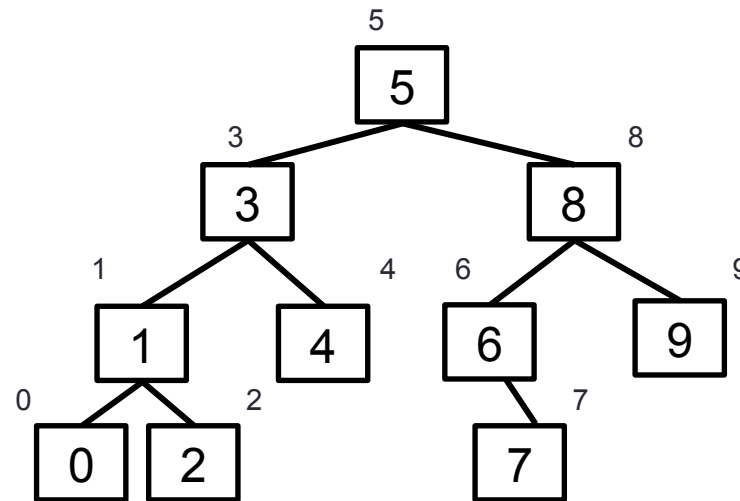
Percurso em pré-ordem: 5 3 1 0 2 4 8 6 7 9

Árvores Binárias

- **em-ordem (e-r-d):**

1. Percorrer a sub-árvore esquerda em-ordem (e-r-d)
2. Visitar a raiz
3. Percorrer a sub-árvore direita em-ordem

A em-ordem visita a raiz entre as ações de percorrer as duas sub-árvores. É conhecida também pelo nome de ordem simétrica.



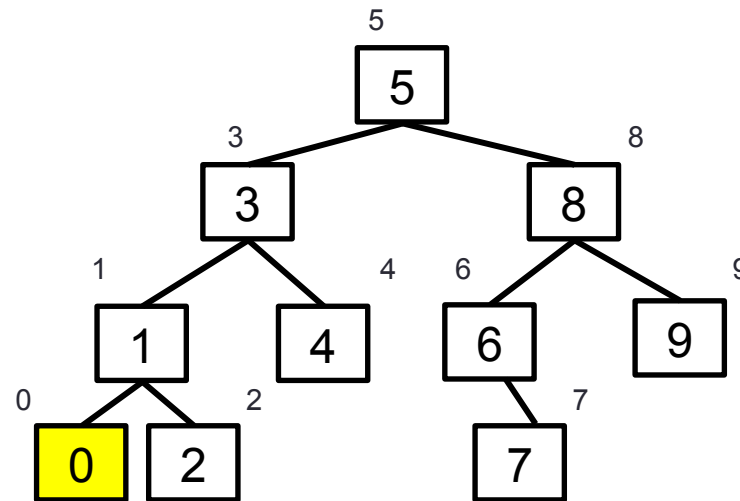
Percurso em-ordem: 0 1 2 3 4 5 6 7 8 9

Árvores Binárias

- **em-ordem (e-r-d):**

1. Percorrer a sub-árvore esquerda em-ordem (e-r-d)
2. Visitar a raiz
3. Percorrer a sub-árvore direita em-ordem

A em-ordem visita a raiz entre as ações de percorrer as duas sub-árvores. É conhecida também pelo nome de ordem simétrica.

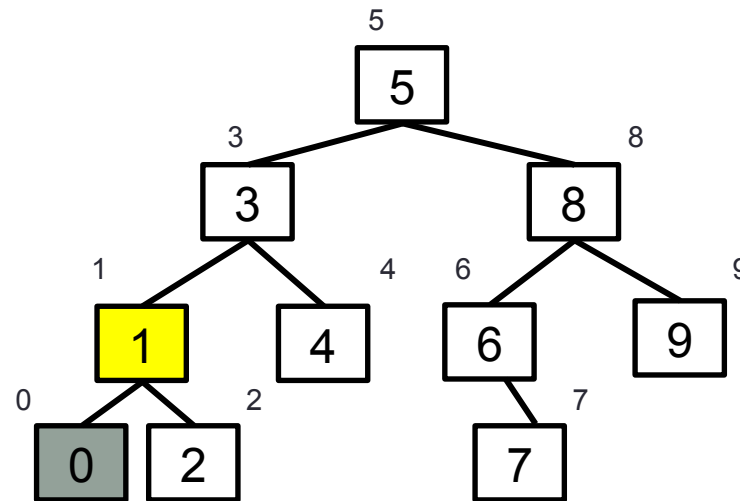


Percurso em-ordem: 0

Árvores Binárias

- **em-ordem (e-r-d):**
 1. Percorrer a sub-árvore esquerda em-ordem (e-r-d)
 2. Visitar a raiz
 3. Percorrer a sub-árvore direita em-ordem

A em-ordem visita a raiz entre as ações de percorrer as duas sub-árvores. É conhecida também pelo nome de ordem simétrica.

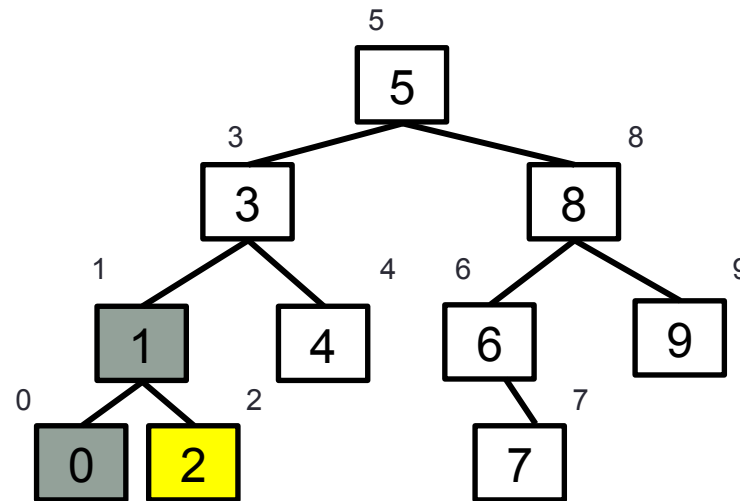


Percurso em-ordem: 0 1

Árvores Binárias

- **em-ordem (e-r-d):**
 1. Percorrer a sub-árvore esquerda em-ordem (e-r-d)
 2. Visitar a raiz
 3. Percorrer a sub-árvore direita em-ordem

A em-ordem visita a raiz entre as ações de percorrer as duas sub-árvores. É conhecida também pelo nome de ordem simétrica.



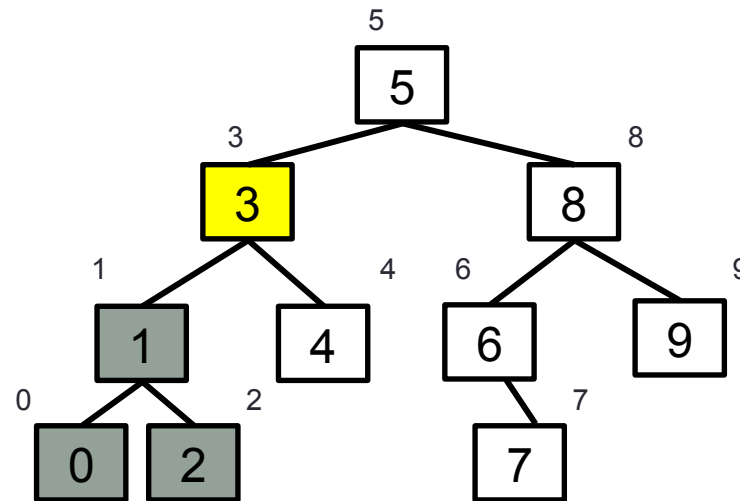
Percurso em-ordem: 0 1 2

Árvores Binárias

- **em-ordem (e-r-d):**

1. Percorrer a sub-árvore esquerda em-ordem (e-r-d)
2. Visitar a raiz
3. Percorrer a sub-árvore direita em-ordem

A em-ordem visita a raiz entre as ações de percorrer as duas sub-árvores. É conhecida também pelo nome de ordem simétrica.

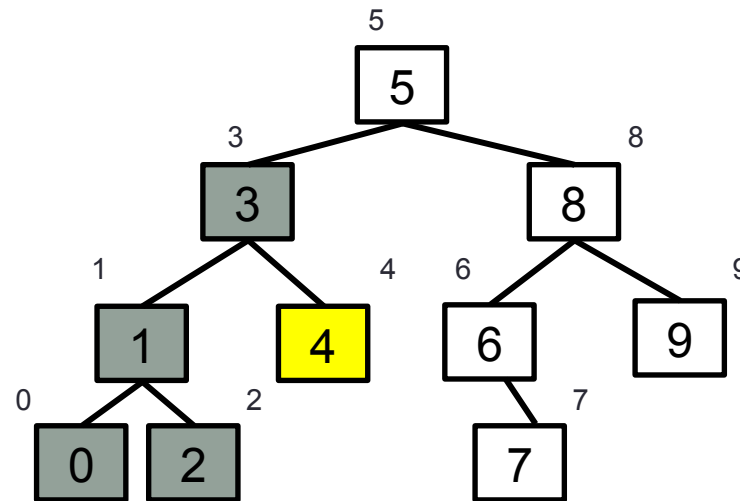


Percurso em-ordem: 0 1 2 3

Árvores Binárias

- **em-ordem (e-r-d):**
 1. Percorrer a sub-árvore esquerda em-ordem (e-r-d)
 2. Visitar a raiz
 3. Percorrer a sub-árvore direita em-ordem

A em-ordem visita a raiz entre as ações de percorrer as duas sub-árvores. É conhecida também pelo nome de ordem simétrica.

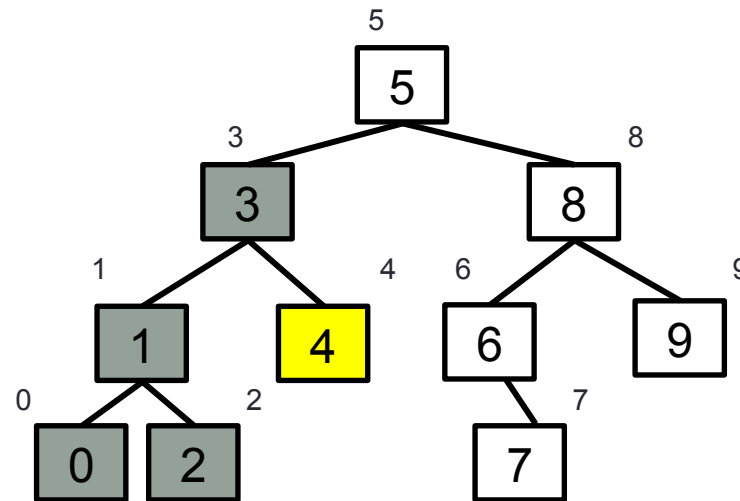


Percurso em-ordem: 0 1 2 3 4

Árvores Binárias

- **em-ordem (e-r-d):**
 1. Percorrer a sub-árvore esquerda em-ordem (e-r-d)
 2. Visitar a raiz
 3. Percorrer a sub-árvore direita em-ordem

A em-ordem visita a raiz entre as ações de percorrer as duas sub-árvores. É conhecida também pelo nome de ordem simétrica.



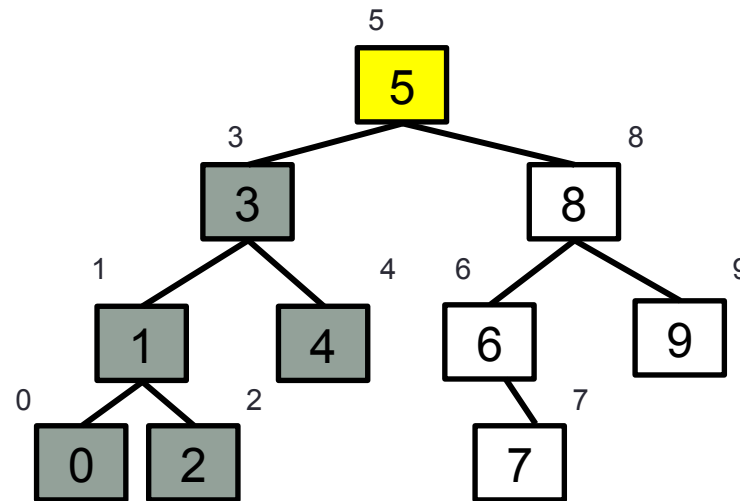
Percurso em-ordem: 0 1 2 3 4

Árvores Binárias

- **em-ordem (e-r-d):**

1. Percorrer a sub-árvore esquerda em-ordem (e-r-d)
2. Visitar a raiz
3. Percorrer a sub-árvore direita em-ordem

A em-ordem visita a raiz entre as ações de percorrer as duas sub-árvores. É conhecida também pelo nome de ordem simétrica.

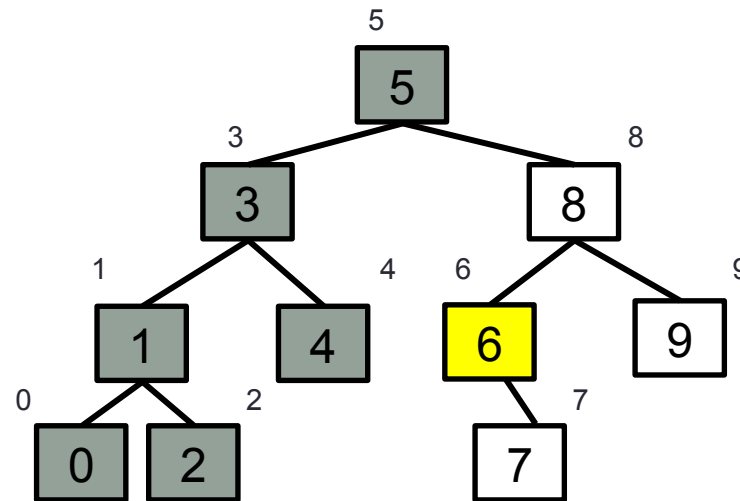


Percurso em-ordem: 0 1 2 3 4 5

Árvores Binárias

- **em-ordem (e-r-d):**
 1. Percorrer a sub-árvore esquerda em-ordem (e-r-d)
 2. Visitar a raiz
 3. Percorrer a sub-árvore direita em-ordem

A em-ordem visita a raiz entre as ações de percorrer as duas sub-árvores. É conhecida também pelo nome de ordem simétrica.

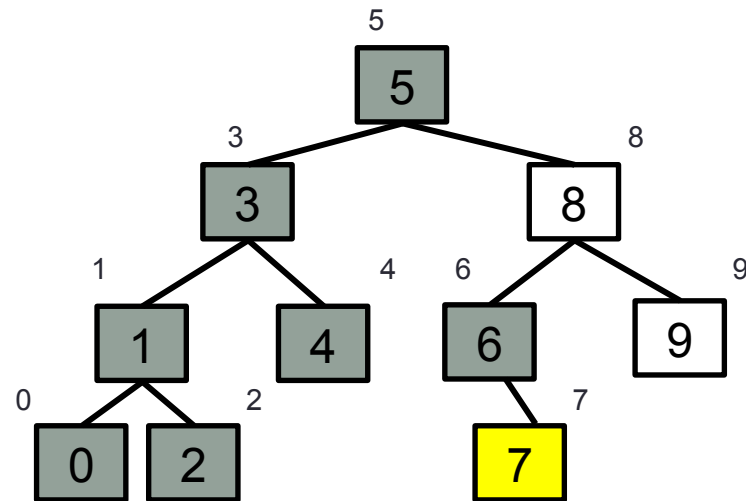


Percurso em-ordem: 0 1 2 3 4 5 6

Árvores Binárias

- **em-ordem (e-r-d):**
 1. Percorrer a sub-árvore esquerda em-ordem (e-r-d)
 2. Visitar a raiz
 3. Percorrer a sub-árvore direita em-ordem

A em-ordem visita a raiz entre as ações de percorrer as duas sub-árvores. É conhecida também pelo nome de ordem simétrica.



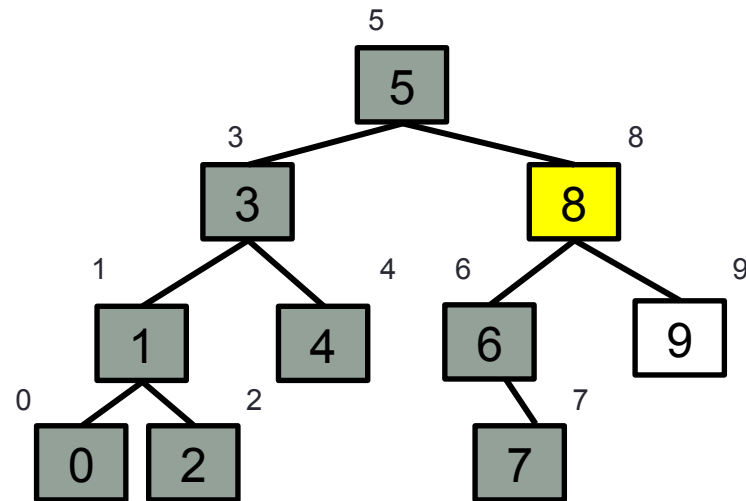
Percurso em-ordem: 0 1 2 3 4 5 6 7

Árvores Binárias

- **em-ordem (e-r-d):**

1. Percorrer a sub-árvore esquerda em-ordem (e-r-d)
2. Visitar a raiz
3. Percorrer a sub-árvore direita em-ordem

A em-ordem visita a raiz entre as ações de percorrer as duas sub-árvores. É conhecida também pelo nome de ordem simétrica.



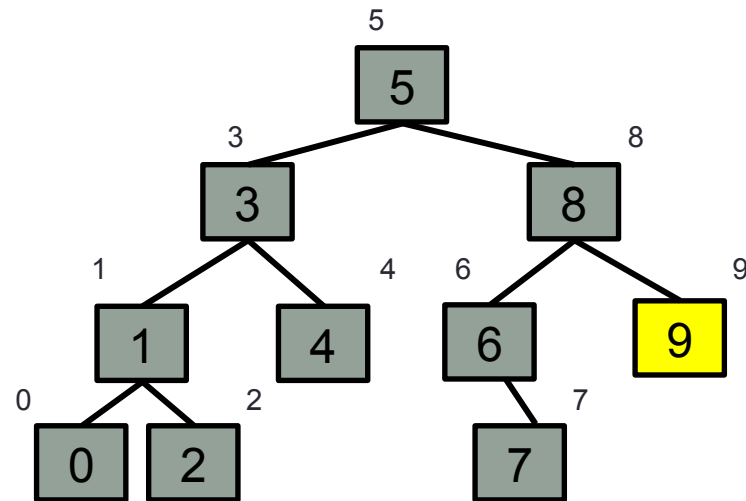
Percurso em-ordem: 0 1 2 3 4 5 6 7 8

Árvores Binárias

- **em-ordem (e-r-d):**

1. Percorrer a sub-árvore esquerda em-ordem (e-r-d)
2. Visitar a raiz
3. Percorrer a sub-árvore direita em-ordem

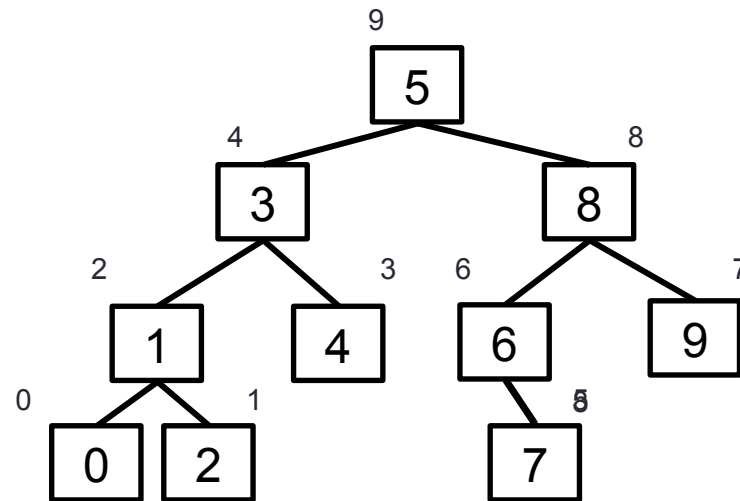
A em-ordem visita a raiz entre as ações de percorrer as duas sub-árvores. É conhecida também pelo nome de ordem simétrica.



Percurso em-ordem: 0 1 2 3 4 5 6 7 8 9

Árvores Binárias

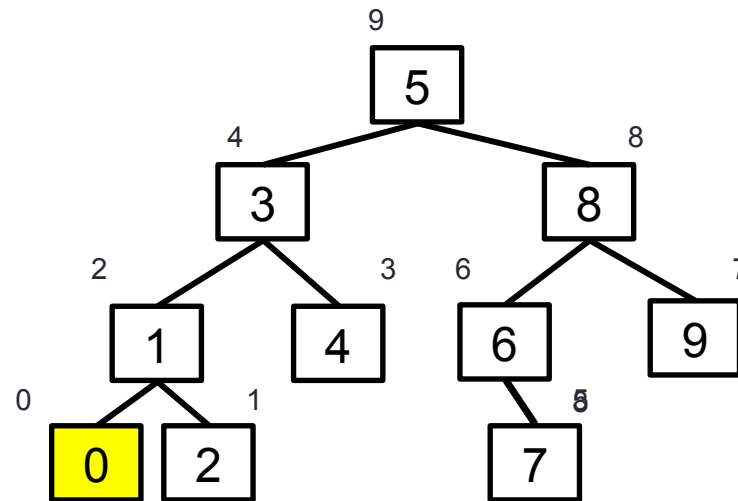
- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



Percurso em pós-ordem: 0 2 1 4 3 7 6 9 8 5

Árvores Binárias

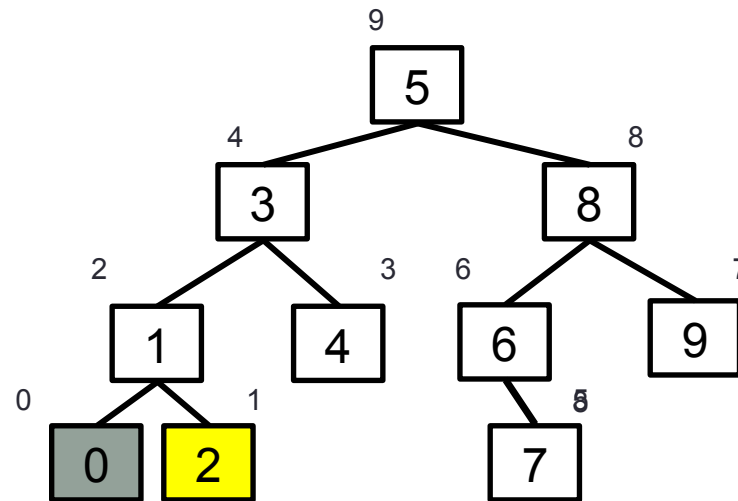
- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



Percurso em pós-ordem: 0

Árvores Binárias

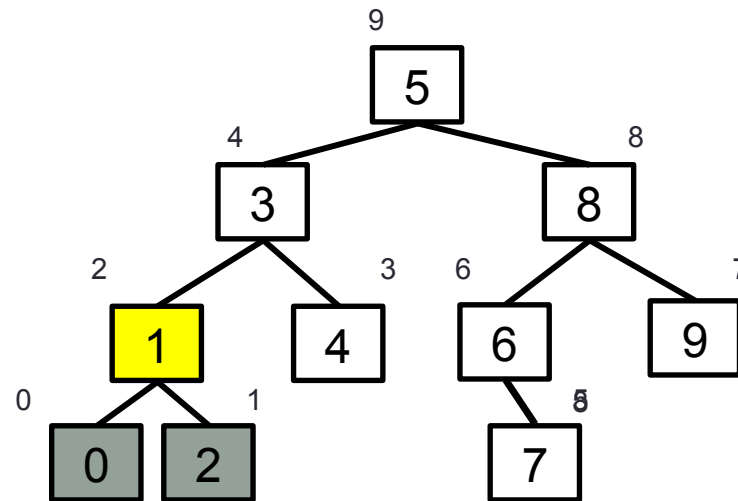
- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



Percurso em pós-ordem: 0 2

Árvores Binárias

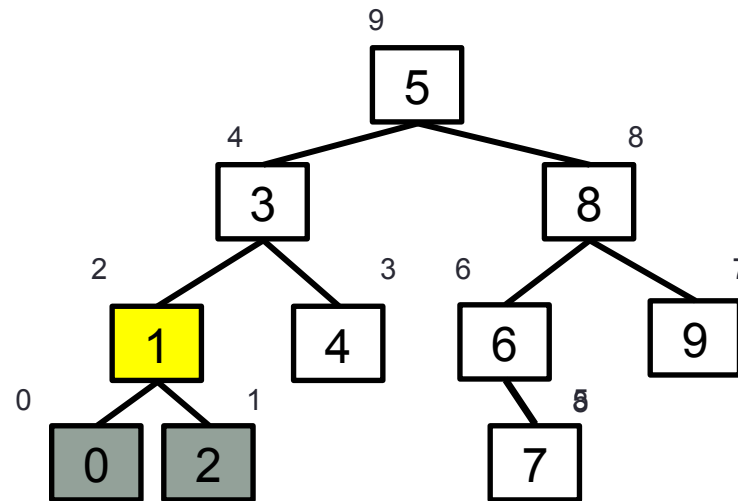
- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



Percurso em pós-ordem: 0 2 1

Árvores Binárias

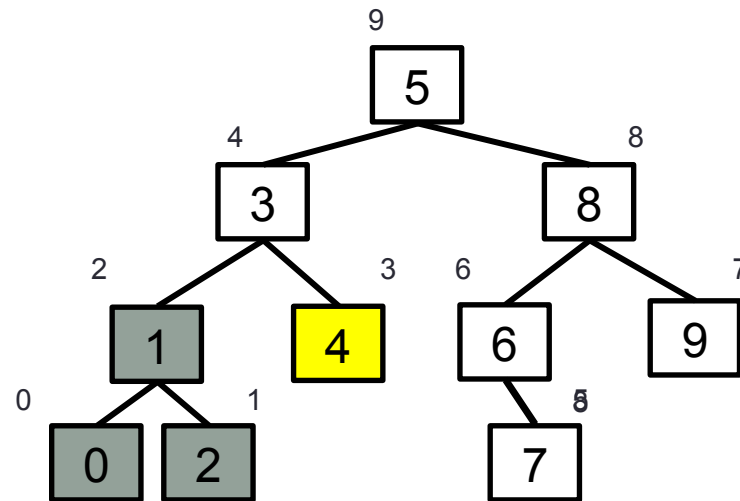
- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



Percurso em pós-ordem: 0 2 1

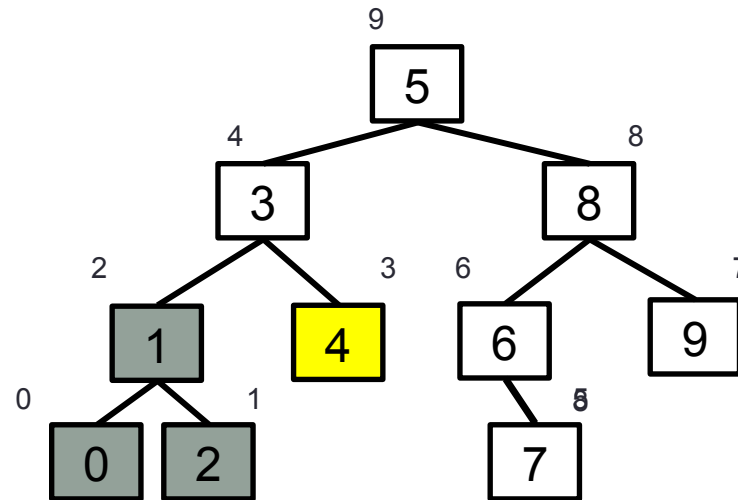
Árvores Binárias

- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



Árvores Binárias

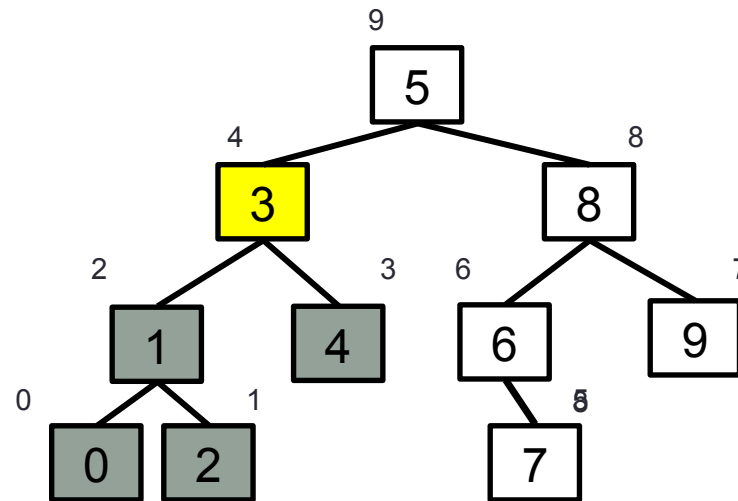
- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



Percurso em pós-ordem: 0 2 1 4

Árvores Binárias

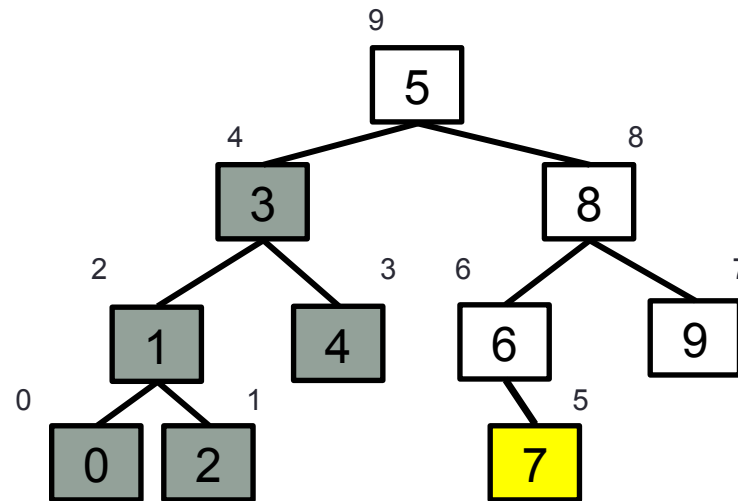
- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



Percurso em pós-ordem: 0 2 1 4 3

Árvores Binárias

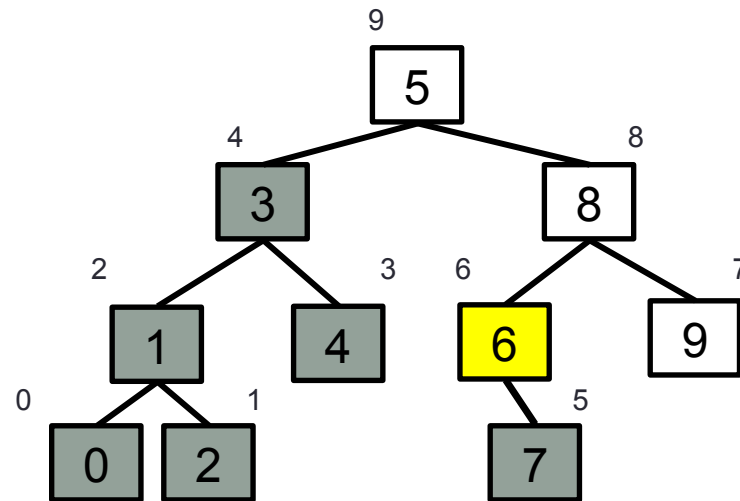
- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



Percurso em pós-ordem: 0 2 1 4 3 7

Árvores Binárias

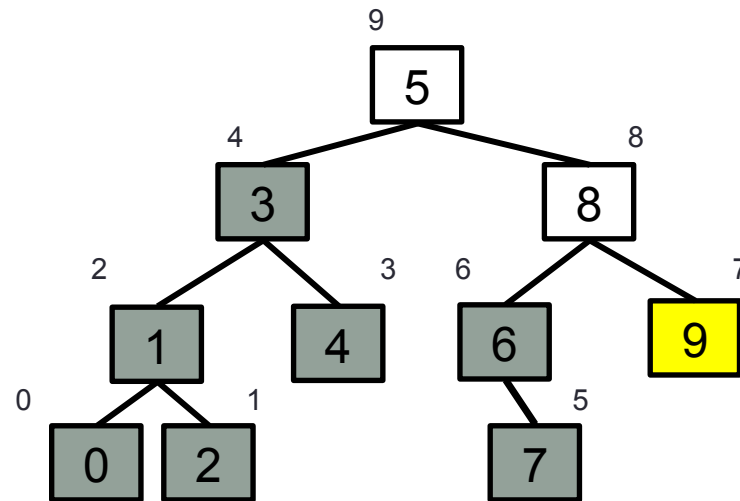
- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



Percurso em pós-ordem: 0 2 1 4 3 7 6

Árvores Binárias

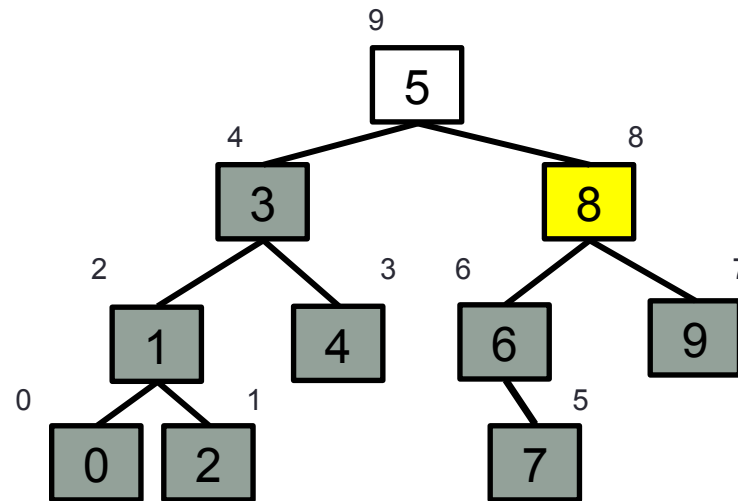
- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



Percurso em pós-ordem: 0 2 1 4 3 7 6 9

Árvores Binárias

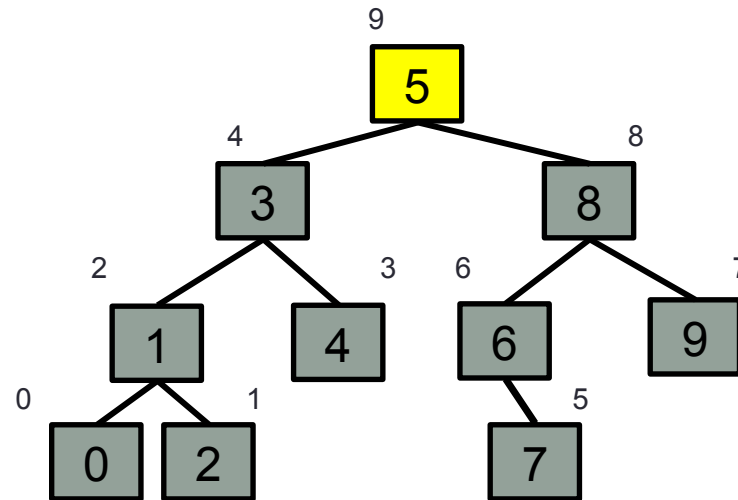
- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



Percurso em pós-ordem: 0 2 1 4 3 7 6 9 8

Árvores Binárias

- **pós-ordem (e-d-r):**
 1. Percorrer a sub-árvore esquerda em pós-ordem
 2. Percorrer a sub-árvore direita em pós-ordem
 3. Visitar a raiz



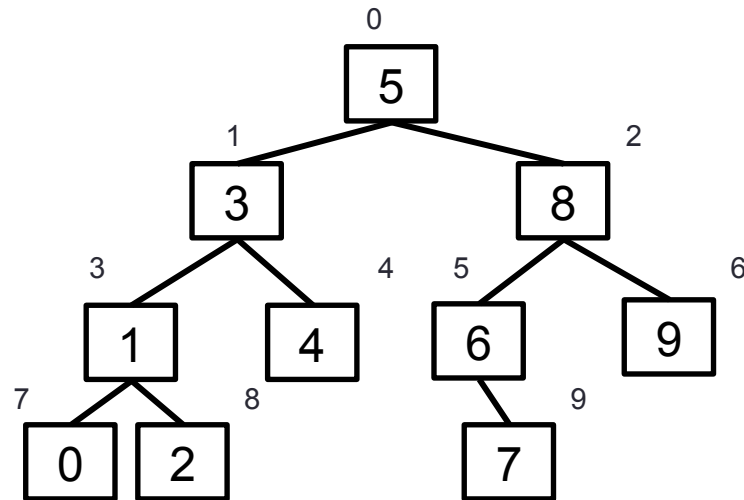
Percurso em pós-ordem: 0 2 1 4 3 7 6 9 8

Árvores Binárias

- *por-nível:*

Os nós são processados por nível (profundidade) crescente, e dentro de cada nível, da esquerda para a direita.

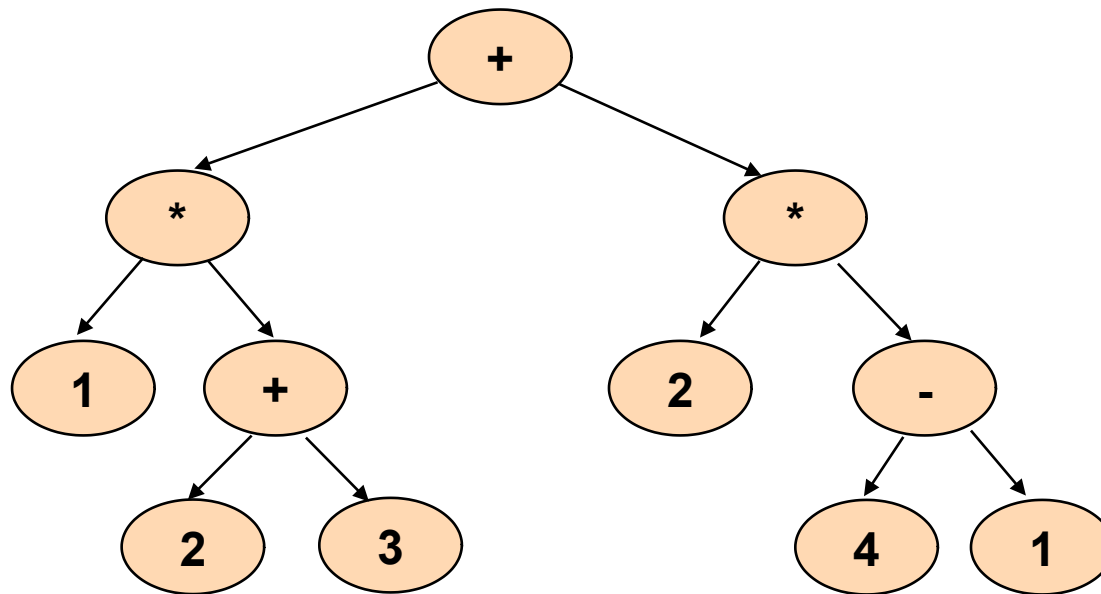
Também é conhecido como breadth-first



Percurso em por nível: 5 3 8 1 4 6 9 0 2 7

Árvores Binárias - Aplicações

- Expressões aritméticas



Expressão = $1 * (2 + 3) + (2 * (4 - 1))$

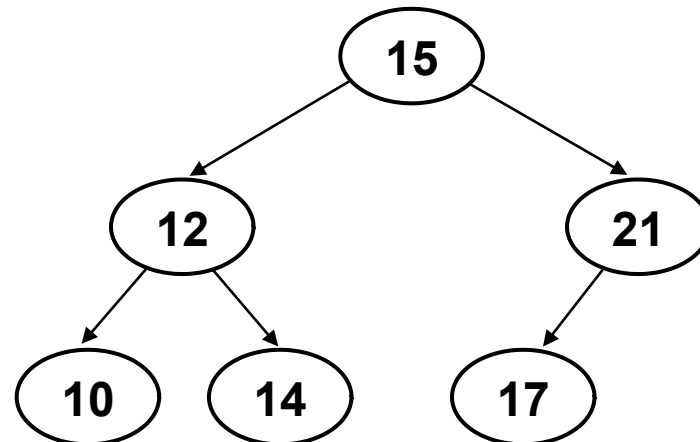
- Ordenação de dados
- Pesquisa em uma lista

Árvores Binárias de Pesquisa

Árvore binária, sem elementos repetidos, que verifica a seguinte propriedade:

- Para **cada nó**, todos os valores da sub-árvore esquerda são menores, e todos os valores da sub-árvore direita são maiores, que o valor desse nó

$$Fe < Pai < Fd$$



Árvores Binárias de Pesquisa

Pesquisa

Usa a propriedade de ordem na árvore para escolher caminho, eliminando uma sub-árvore a cada comparação.

Desce nível por nível até encontrar o nó buscado

- Quando encontra um nó maior que o buscado segue pelo ponteiro da esquerda
- Quando encontra um nó menor que o buscado, segue pelo ponteiro da direita
- Caso chegue em uma folha, o nó buscado não existe na árvore

Inserção

- Como pesquisa; novo nó inserido onde pesquisa falha

Máximo e mínimo

- Procura, escolhendo sempre a subárvore direita (máximo), ou sempre a sub-árvore esquerda (mínimo)

Remoção

- Nó folha: apagar nó
- Nó com 1 filho: filho substitui pai
- Nó com 2 filhos: elemento é substituído pelo menor da sub-árvore direita (ou maior da esquerda); o nó deste tem no máximo 1 filho e é apagado

Árvores Binárias de Pesquisa

Cada nó tem, no máximo, 2 filhos

Seja $q(n)$ a quantidade de nós no nível n . Então $q(n) = 2 \cdot q(n-1)$.

Sabe-se que $q(0)=1$

Logo: $q(1) = 2 \cdot q(0) = 2$

$q(2) = 2 \cdot q(1) = 4$

$q(3) = 2 \cdot q(2) = 8$

...

Para uma árvore de altura h :

$Q = q(0) + q(1) + \dots + q(p-1) + q(p) \rightarrow$ onde p é a profundidade máxima da árvore

$= q(0) + 2 \cdot q(0) + 2 \cdot 2 \cdot q(0) \dots \rightarrow$ na profundidade p há, no máximo, $Q(p) = 2^p \times 1$

Para uma árvore de altura h , a quantidade máxima de nós é

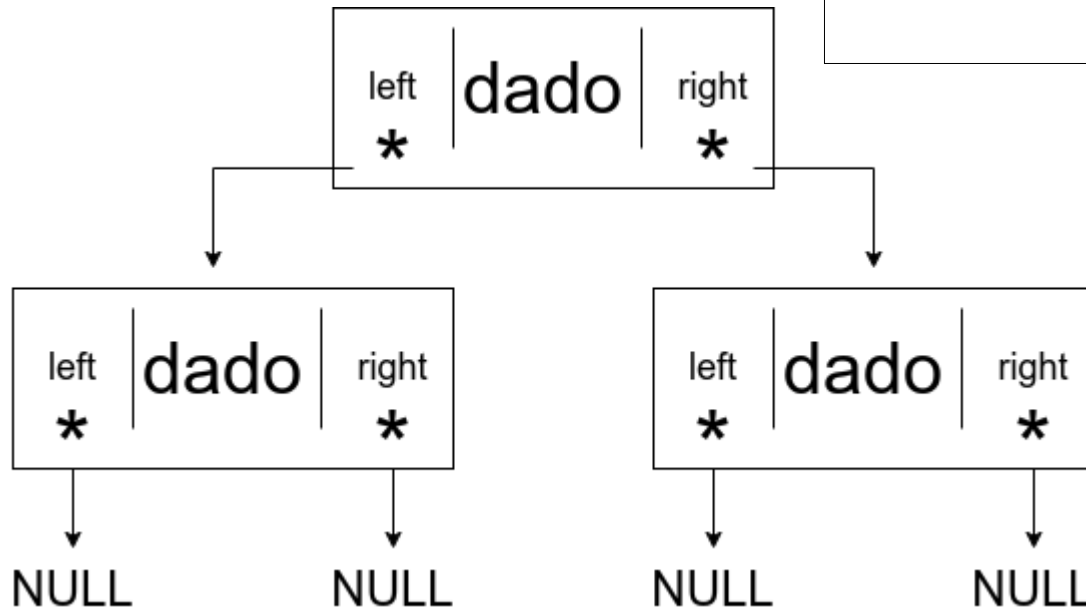
$$Q(h) = \sum_{p=0}^h 2^p$$

Árvores Binárias de Pesquisa em C

```
struct tree{  
    int dado;  
    struct tree *left; //filho da esquerda  
    struct tree *right; //filho da direita  
} typedef Tree;
```

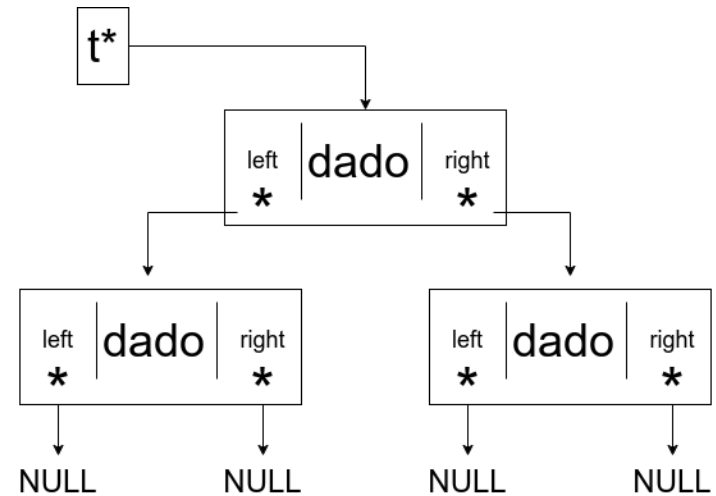
Árvores Binárias de Pesquisa em C

```
struct tree{  
    int dado;  
    struct tree *left; //filho da esquerda  
    struct tree *right; //filho da direita  
} typedef Tree;
```



Árvores Binárias de Pesquisa em C

```
Tree *insert(Tree *t, int dado) {  
    if (t==NULL) {  
        t = malloc(sizeof(Tree));  
        (*t).dado = dado;  
        (*t).left = NULL;  
        (*t).right = NULL;  
    }  
    else  
        if (dado < (*t).dado)  
            (*t).left = insert((*t).left, dado);  
        else  
            (*t).right = insert((*t).right, dado);  
    return t;  
}
```



Árvores Binárias de Pesquisa em C

```
void mostra_arvore(Arvore *arv, int nivel){
    int i;
    if (arv != NULL){
        for (i = 0; i < nivel*5; i++){
            if (i > nivel*5 - 5){
                printf("-");
            } else {
                printf(" ");
            }
        }
        printf(">%d\n", arv->valor);
        nivel++;
        mostra_arvore(arv->fe, nivel);
        mostra_arvore(arv->fd, nivel);
    }
}
```

Árvores Binárias de Pesquisa em C

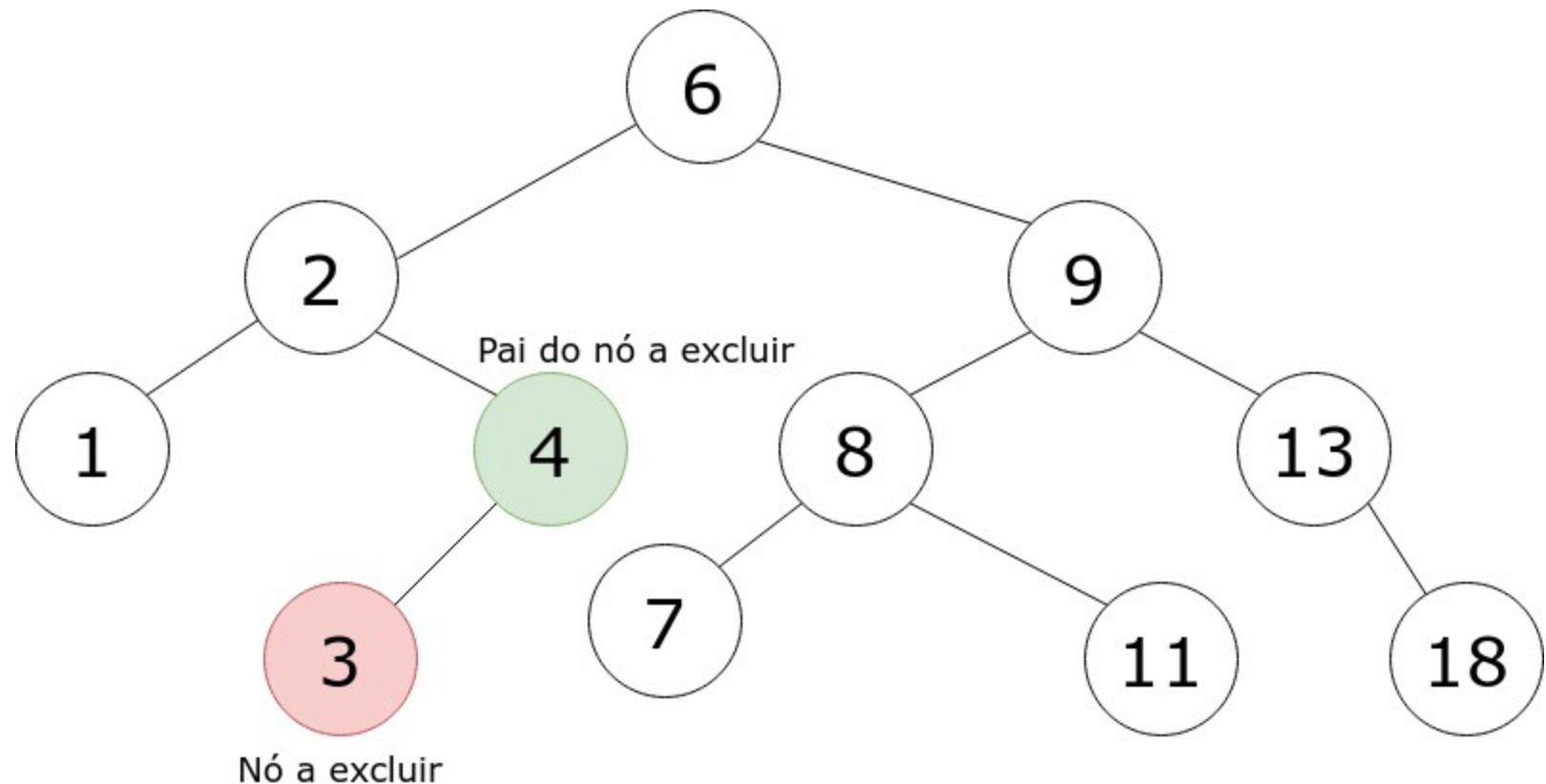
```
void libera_arvore(Arvore *arv) {  
    if(arv->fe != NULL) {  
        libera_arvore(arv->fe);  
        arv->fe = NULL;  
    }  
    if (arv->fd != NULL) {  
        libera_arvore(arv->fd);  
        arv->fd = NULL;  
    }  
  
    if(arv->fe == NULL && arv->fd == NULL) {  
        //printf("Liberando: %d\n", arv->valor);  
        free(arv);  
    }  
}
```

Exclusão em Árvores Binárias de Pesquisa

Três situações:

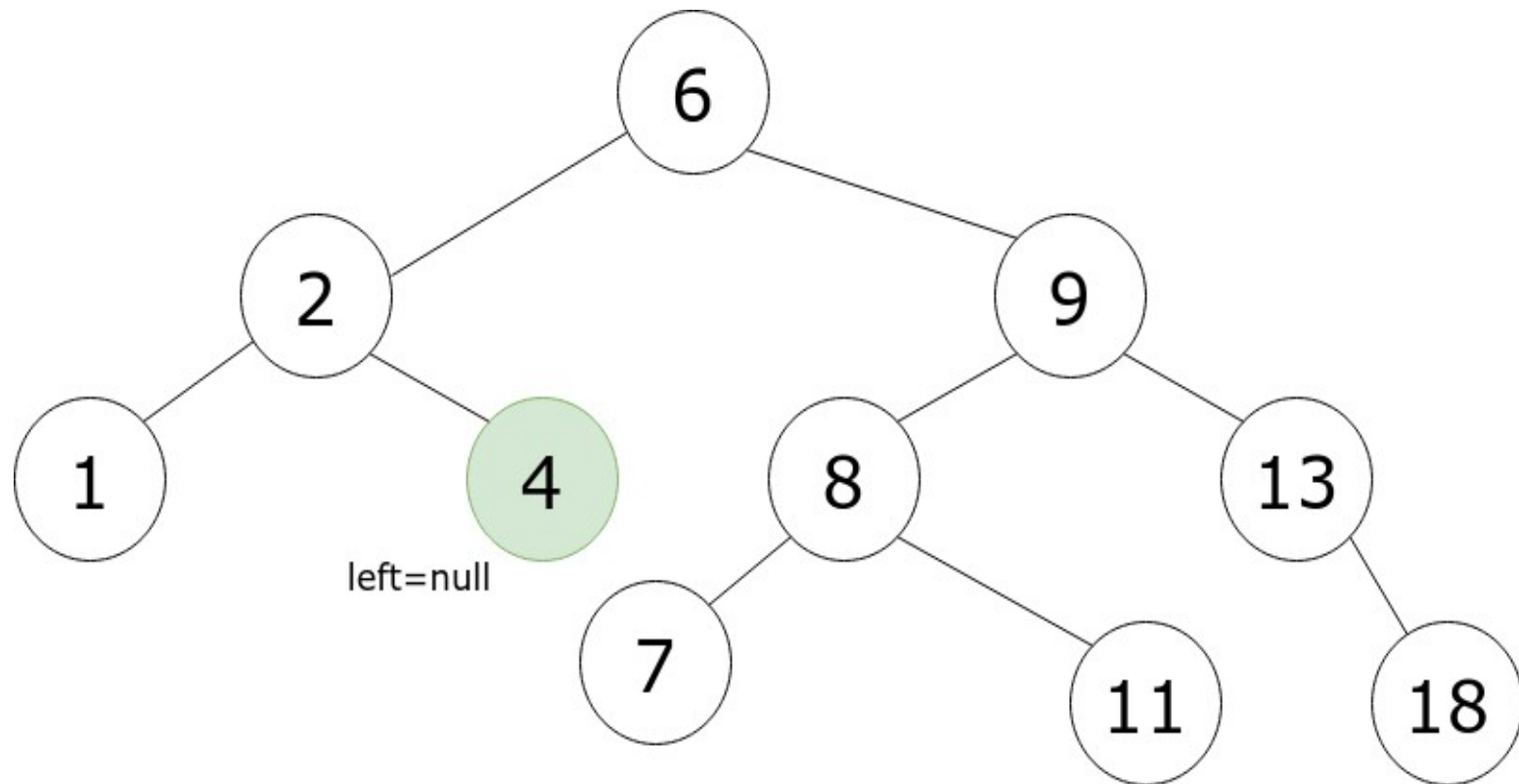
- 1.Exclusão de um nó folha;
- 2.Exclusão de um nó com uma única subárvore
- 3.Exclusão de um nó com duas subárvores

Exclusão de um nó folha

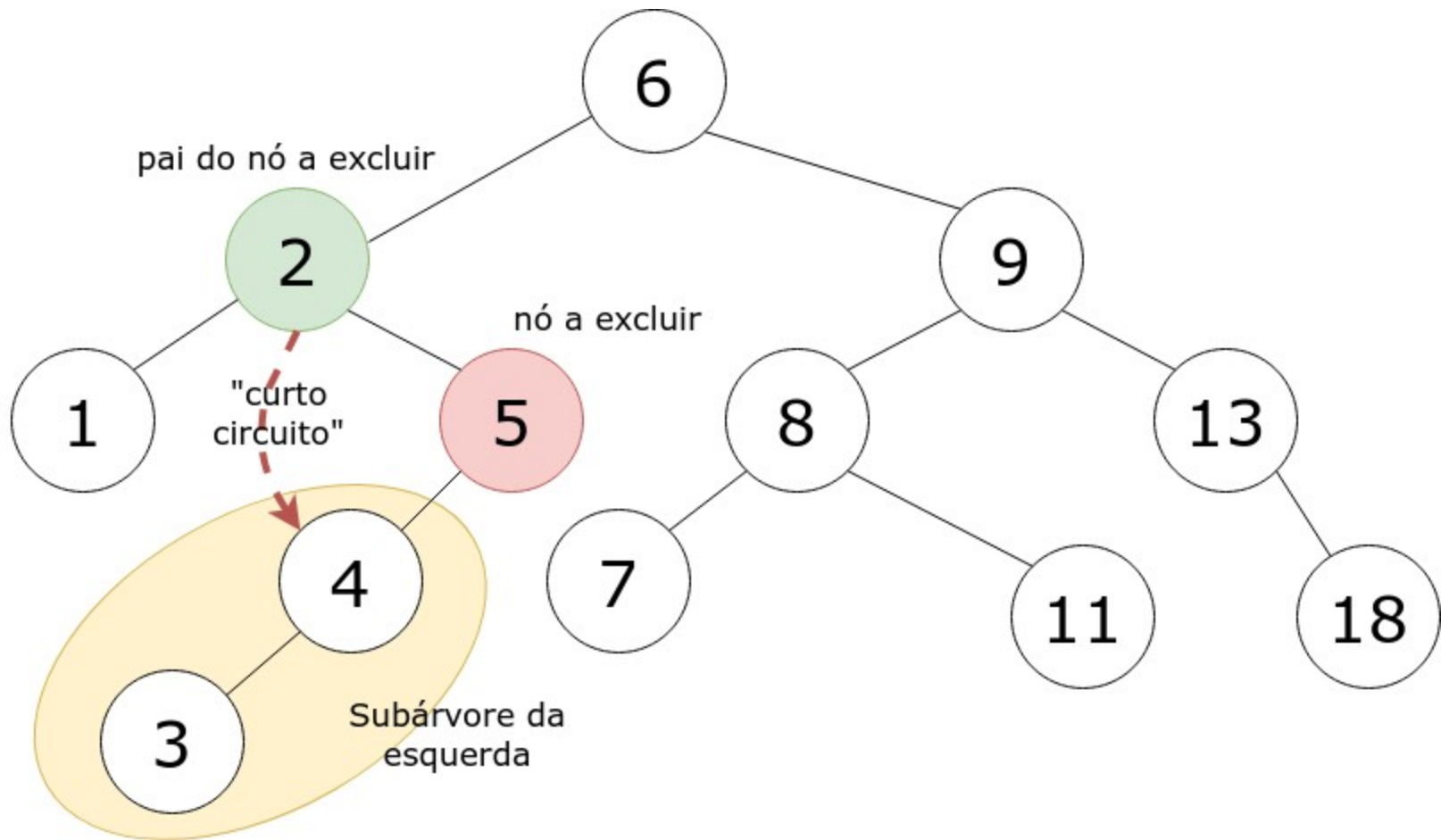


O ponteiro que apontava para o nó excluído passa a apontar para NULL

Exclusão de um nó folha

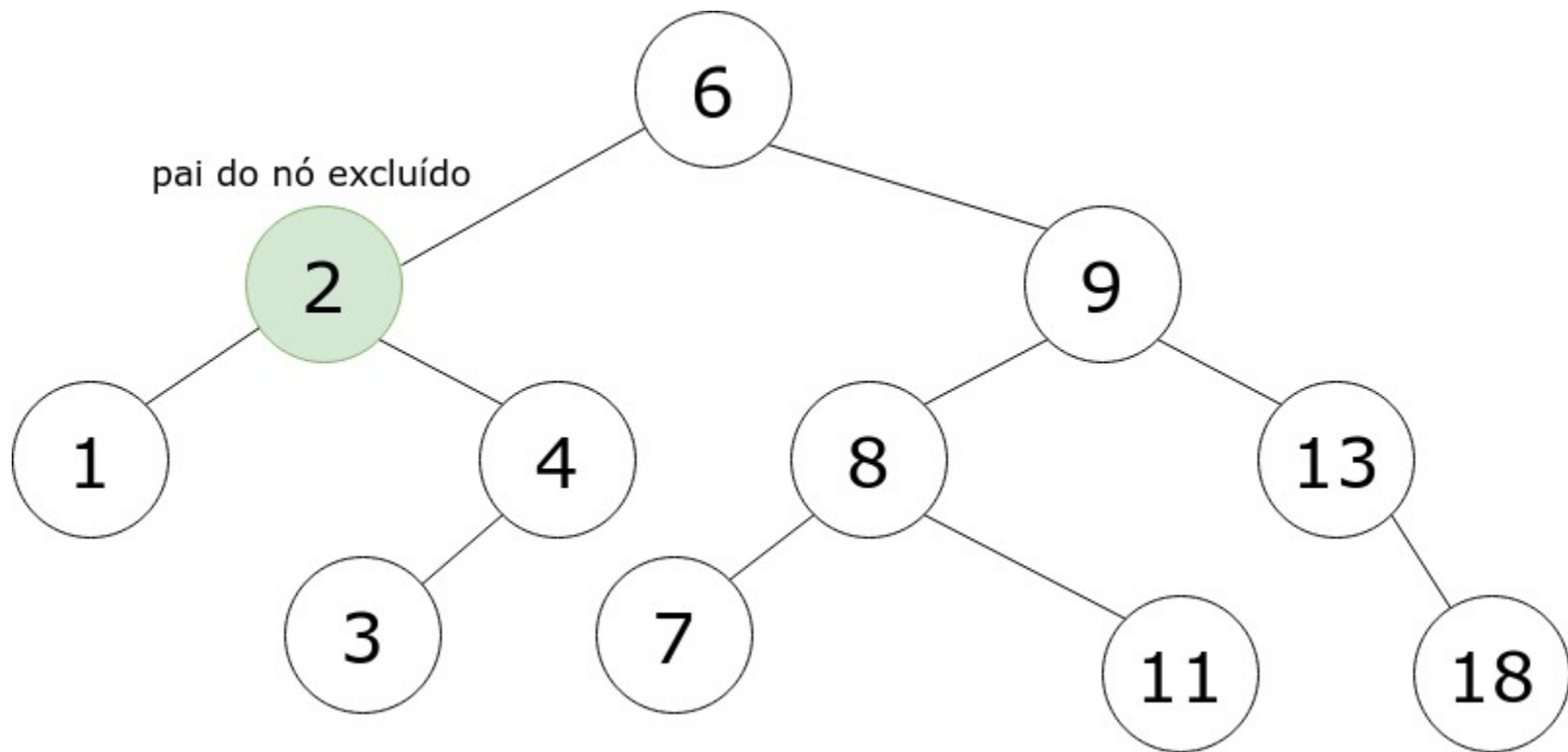


Exclusão de um nó com uma subárvore



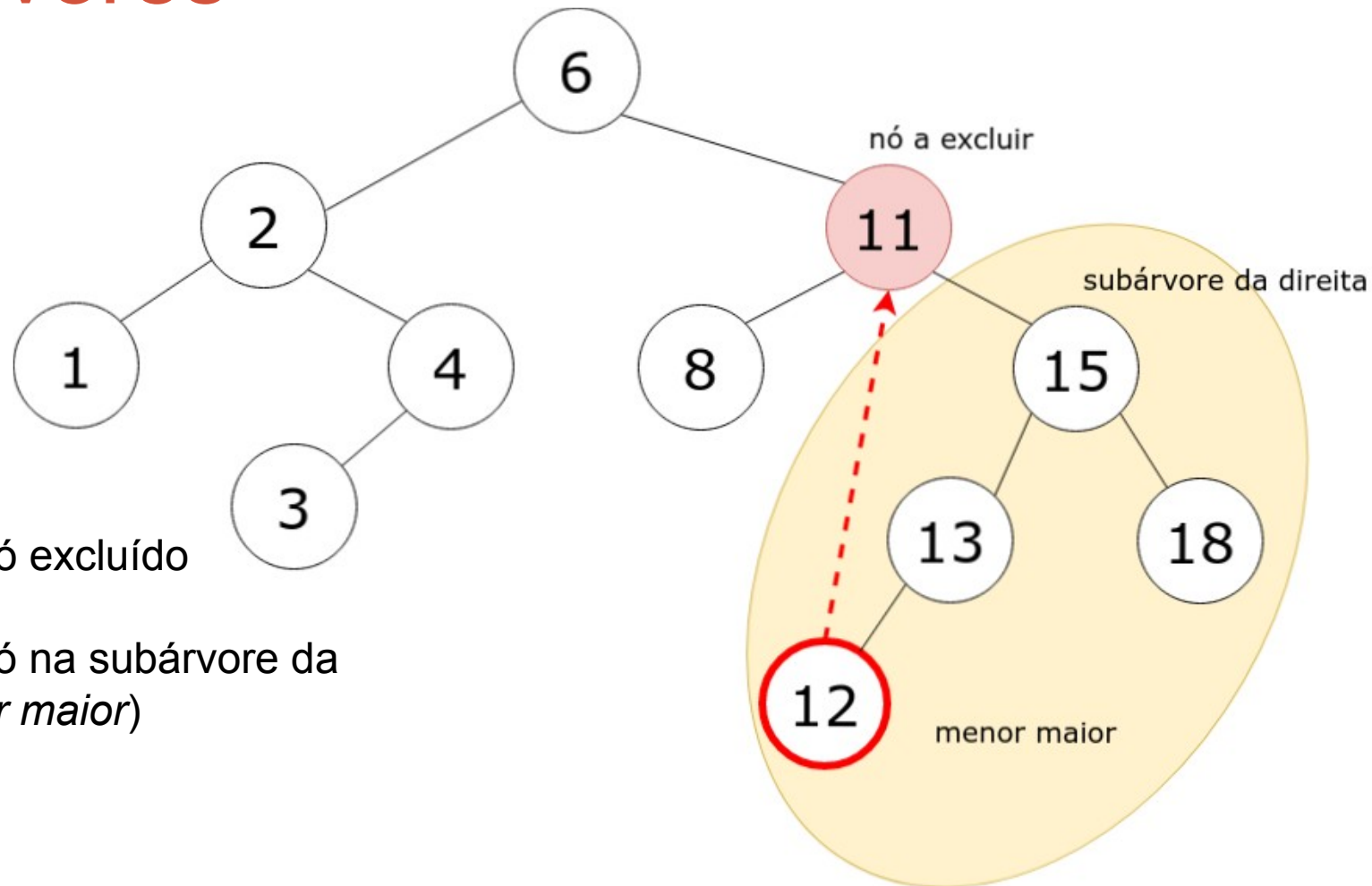
A subárvore do nó excluído torna-se subárvore de seu pai

Exclusão de um nó com uma subárvore



A subárvore do nó excluído torna-se subárvore de seu pai

Exclusão de um nó com duas subárvores



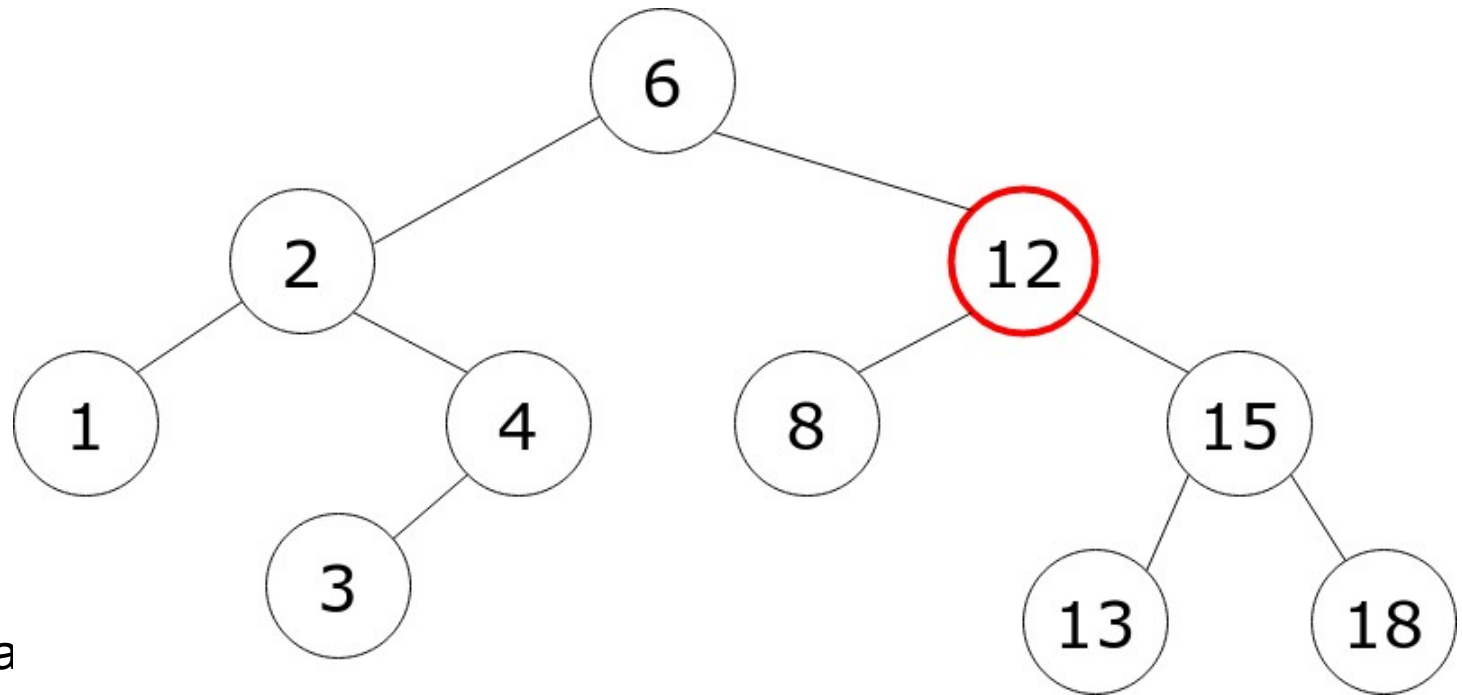
Substituir o nó excluído

pelo *menor* nó na subárvore da direita (*menor maior*)

ou

pelo *maior* nó na subárvore da esquerda (*maior menor*)

Exclusão de um nó com duas subárvores



Todos os nós na
têm valores menores que a raiz

e

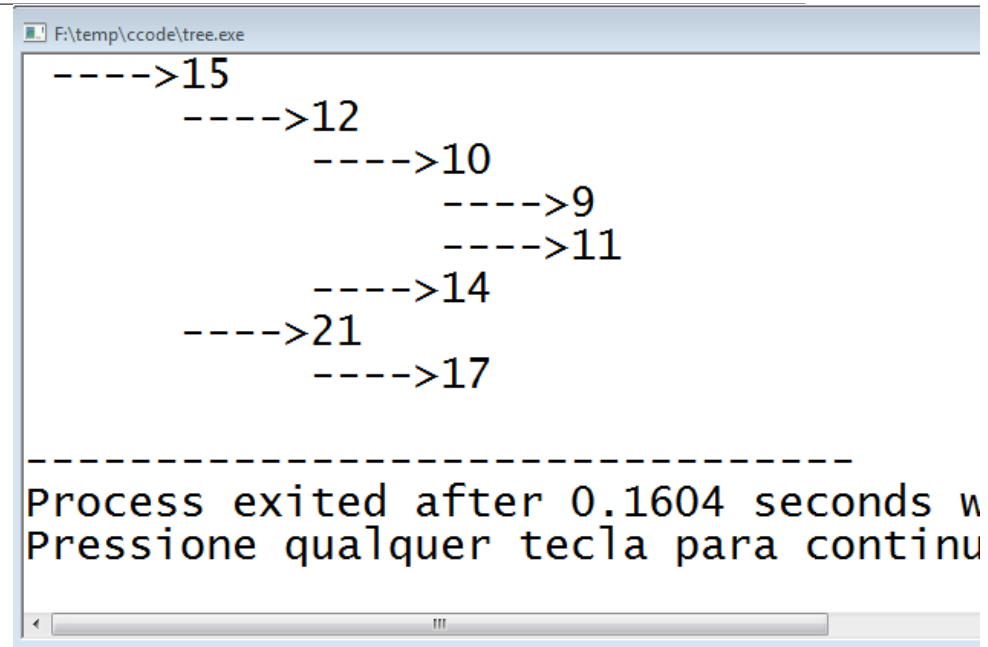
Todos os nós na subárvore da direita têm
valores maiores que a raiz

Árvores Binárias de Pesquisa em C

```
int altura_arvore(Arvore *arv) {  
    int esq, dir;  
    if (arv == NULL)  
        return -1;  
    if (arv->fe == NULL && arv->fd == NULL) {  
        return 0;  
    }  
    esq = altura_arvore(arv->fe);  
    dir = altura_arvore(arv->fd);  
    return esq > dir ? esq+1 : dir+1;  
}
```

Árvores Binárias de Pesquisa em C

```
int main() {  
    Arvore *a = NULL;  
    a = insert(a, 15);  
    a = insert(a, 12);  
    a = insert(a, 21);  
    a = insert(a, 10);  
    a = insert(a, 14);  
    a = insert(a, 17);  
    a = insert(a, 11);  
    a = insert(a, 9);  
    mostra_arvore(a, 1);  
    libera_arvore(a);  
    a = NULL;  
}
```



```
F:\temp\ccode\tree.exe  
----->15  
      ----->12  
            ----->10  
                  ----->9  
                  ----->11  
            ----->14  
      ----->21  
            ----->17  
  
-----  
Process exited after 0.1604 seconds w  
Pressione qualquer tecla para continu
```

Árvores Binárias de Pesquisa - Aplicação

Contagem de ocorrência de palavras

- Pretende-se escrever um programa que leia um arquivo de texto e apresente uma listagem ordenada das palavras nele existentes e o respetivo número de ocorrências.
- Algoritmo:
 - Guardar as palavras e contadores associados numa árvore binária de pesquisa.
 - Usar ordem alfabética para comparar os nós.

Árvores AVL

Uma **árvore AVL** é uma árvore de pesquisa binária de **altura balanceada**.

- No pior caso, a busca em uma árvore binária pode ser $O(n)$ – semelhante a uma lista.
- AVLs são sempre balanceadas – para um número n de nós, a altura sempre será $\lfloor \log(n) \rfloor$
- Criadores: Adelson-Velskii e Landis
- Para cada nó x , as alturas das subárvores esquerda e direita de x diferem em, no máximo, 1.

Árvores AVL

Inserção em AVL:

1. Igual a árvores binárias de pesquisa mas...
- 2....pode violar a propriedade de balanceamento

AVL desbalanceada requer **reestruturação**

Árvores AVL - Reestruturação

Seja **T** a raiz de uma (sub)árvore desbalanceada.

Sejam **a**, **b** e **c** os nós que compõem essa árvore, sendo que $a < b < c$.

Sejam **T0**, **T1**, **T2** e **T3** as demais subárvores de **T**, também lidos da esquerda para a direita, tal que **T0** é subárvore esquerda de **a** e **T3** é subárvore direita de **c**.

a será subárvore esquerda de **b**

c será subárvore direita de **b**

T1 será subárvore de **a**

T2 será subárvore de **c**

b será filho do pai do nodo mais alto

Árvores AVL - Exercício

Em uma árvore inicialmente vazia, inserir os seguintes elementos: 10, 20, 15, 25, 12, 5, 18, 30, 22, 35, 3, 6 mantendo a árvore balanceada.

Remover o elemento 25 mantendo a árvore balanceada

Para analisar os resultados, utilizar o simulador de árvores AVL:

<https://www.cs.usfca.edu/~galles/visualization/AVLtree.html>