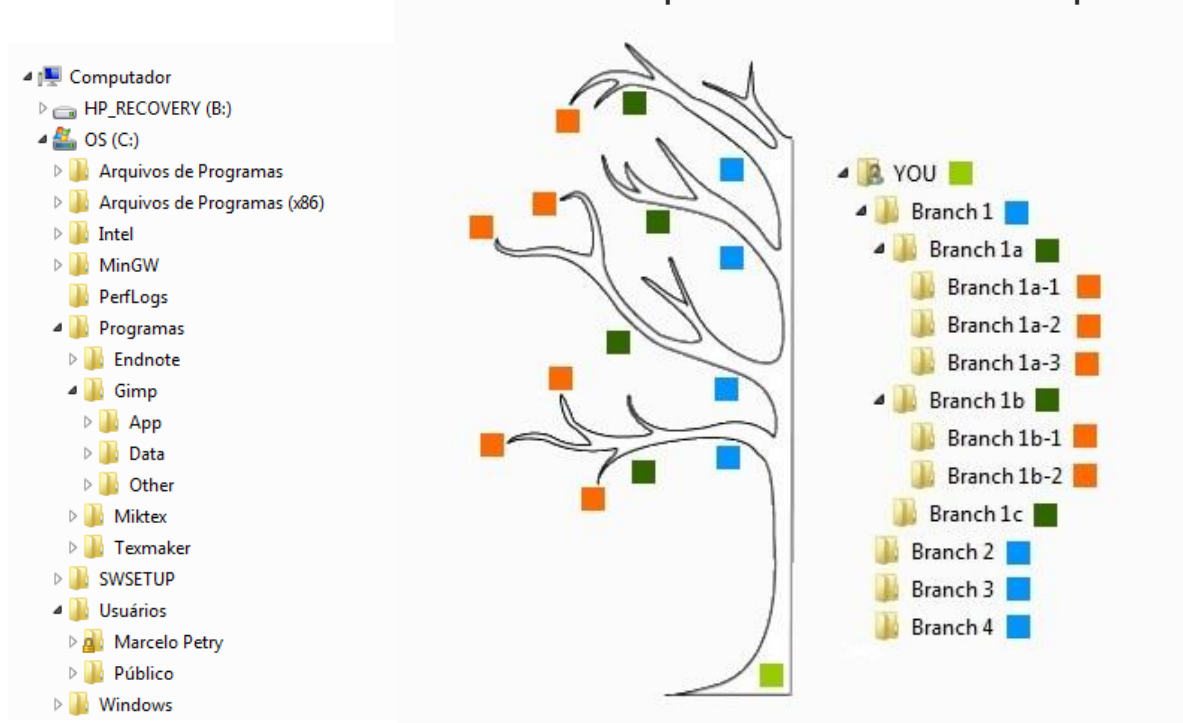


Algoritmos e estruturas de Dados A09

Árvores

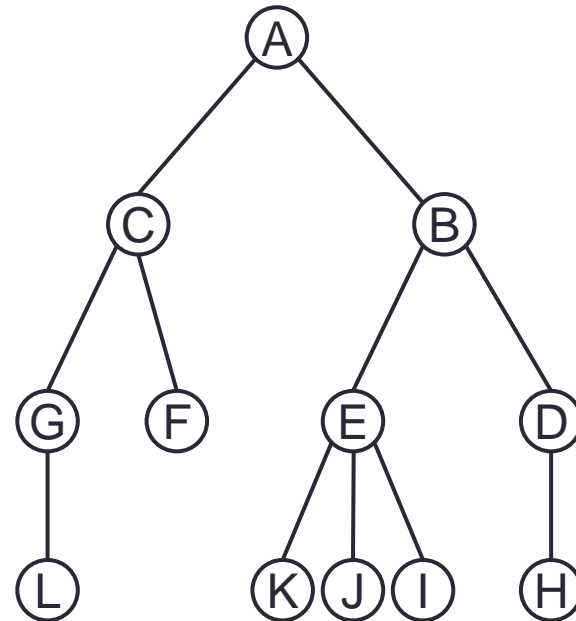
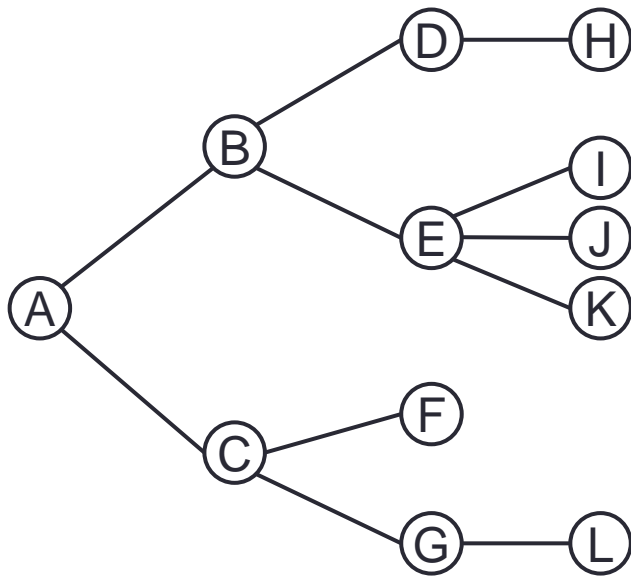
Árvores

- 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



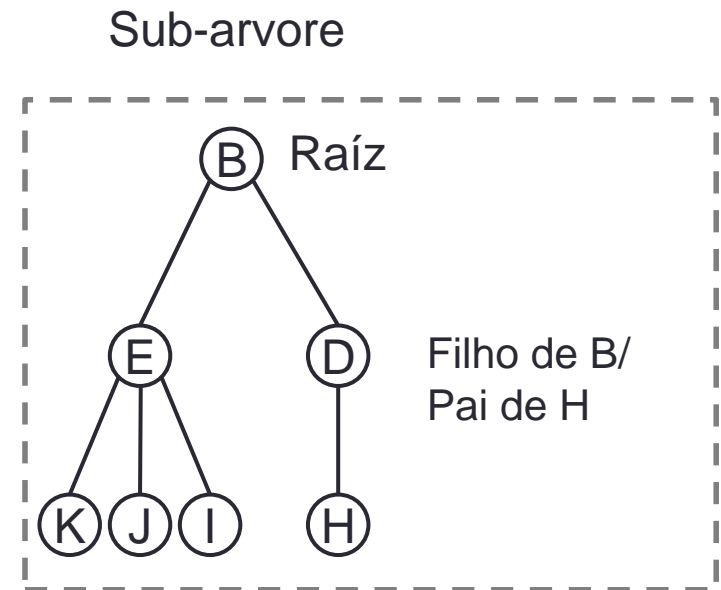
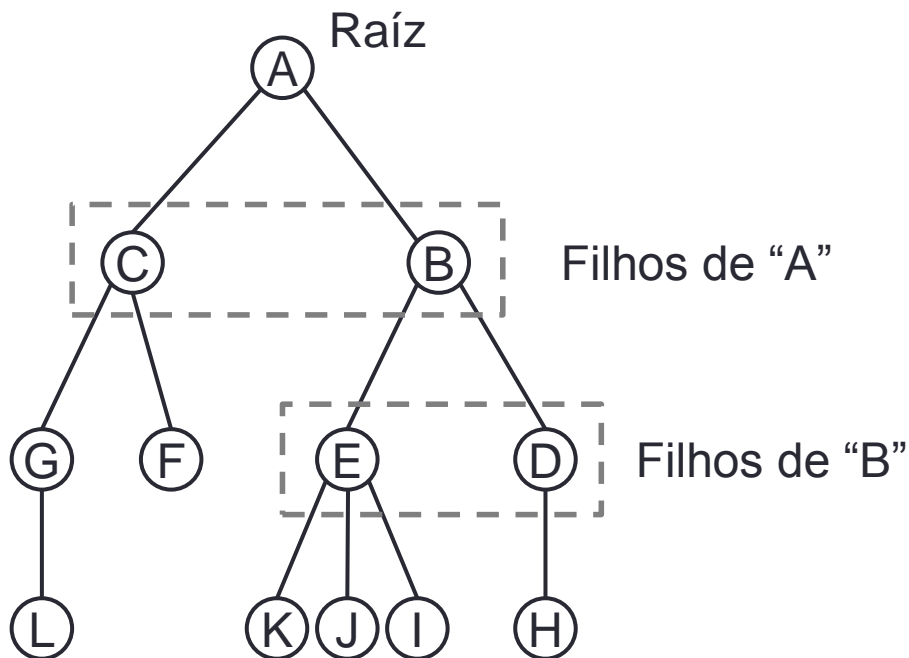
Árvores

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



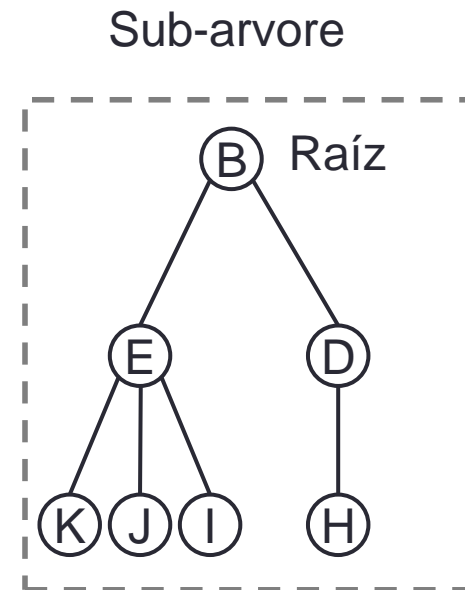
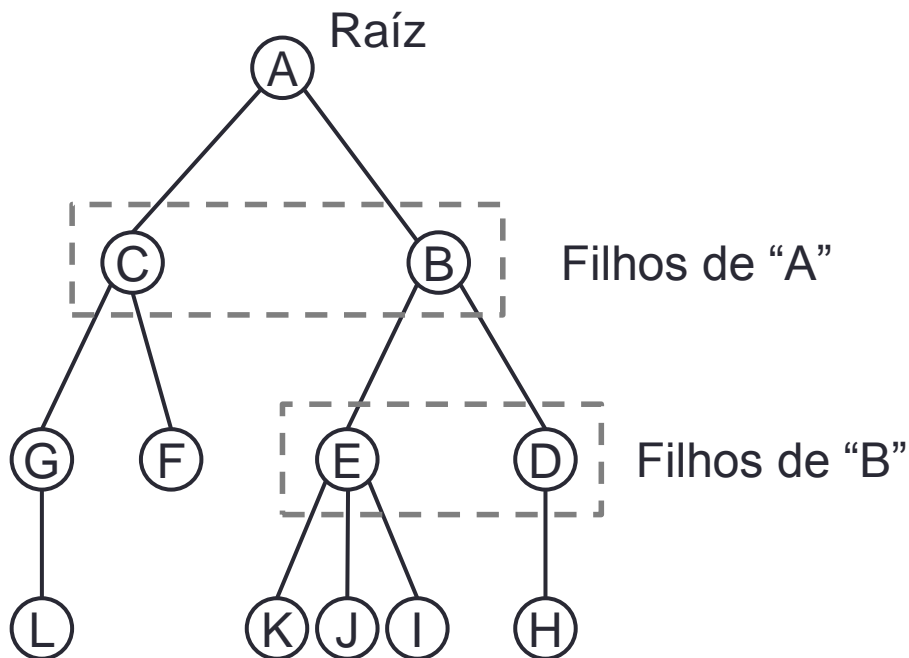
Árvores

- 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**.



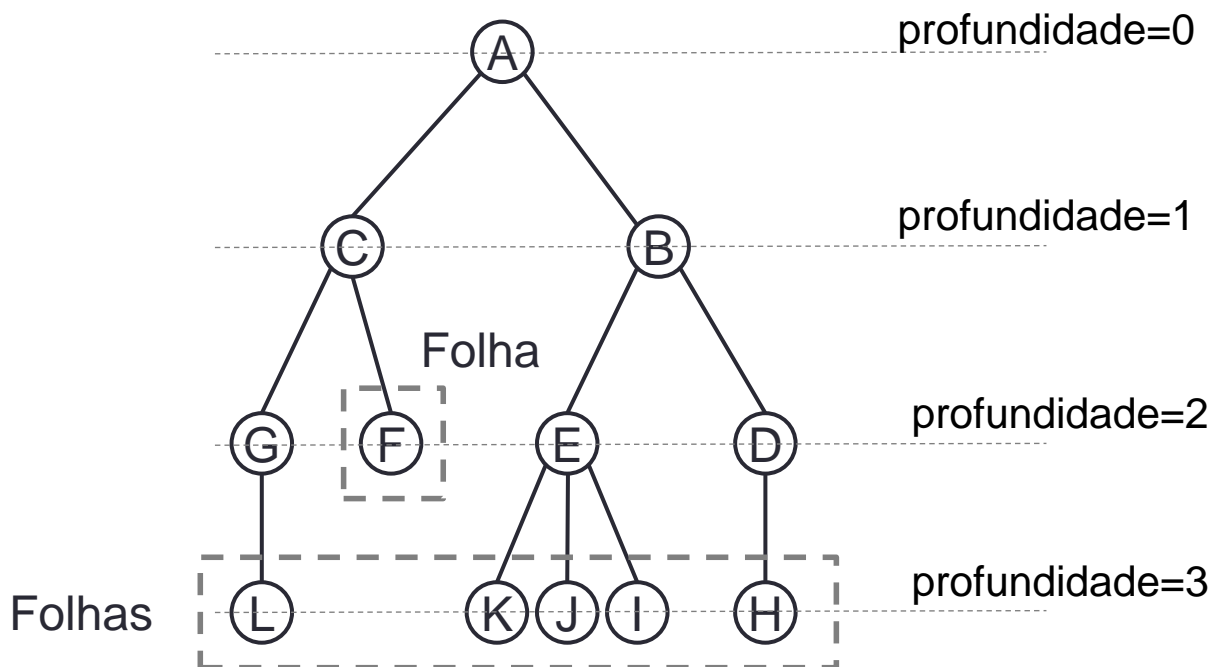
Árvores

- 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



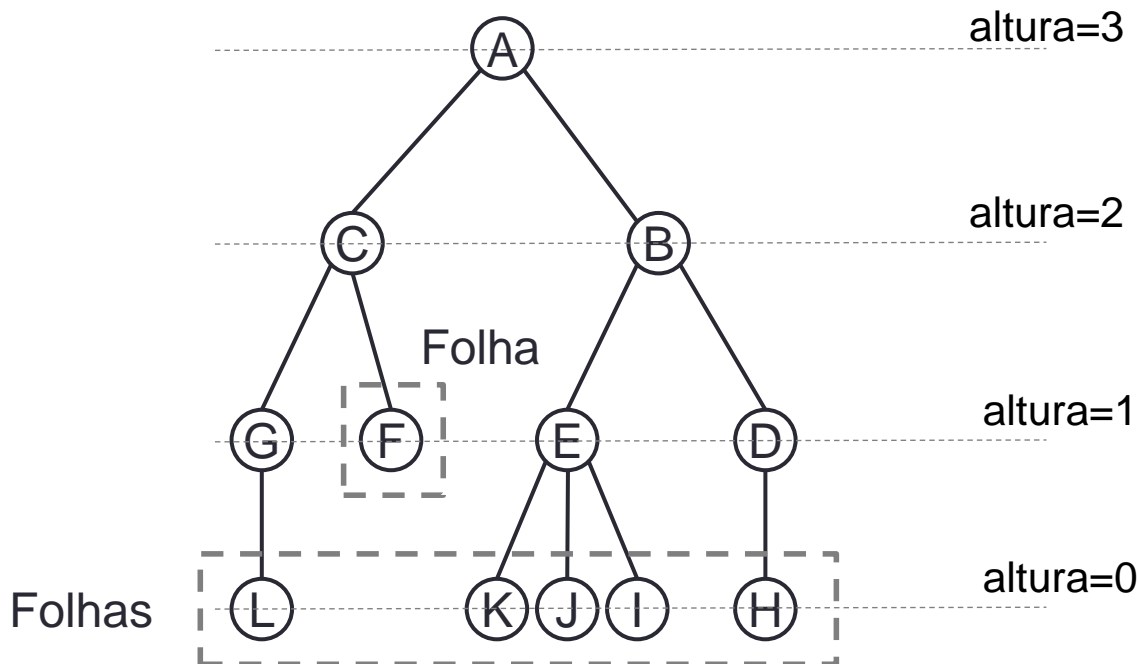
Árvores

- **Folha:** nó sem filhos.
- **Profundidade** de um nó: Comprimento da raiz até ao nó
 - Profundidade da raiz é 0
 - Profundidade de um nó é 1 + profundidade do pai



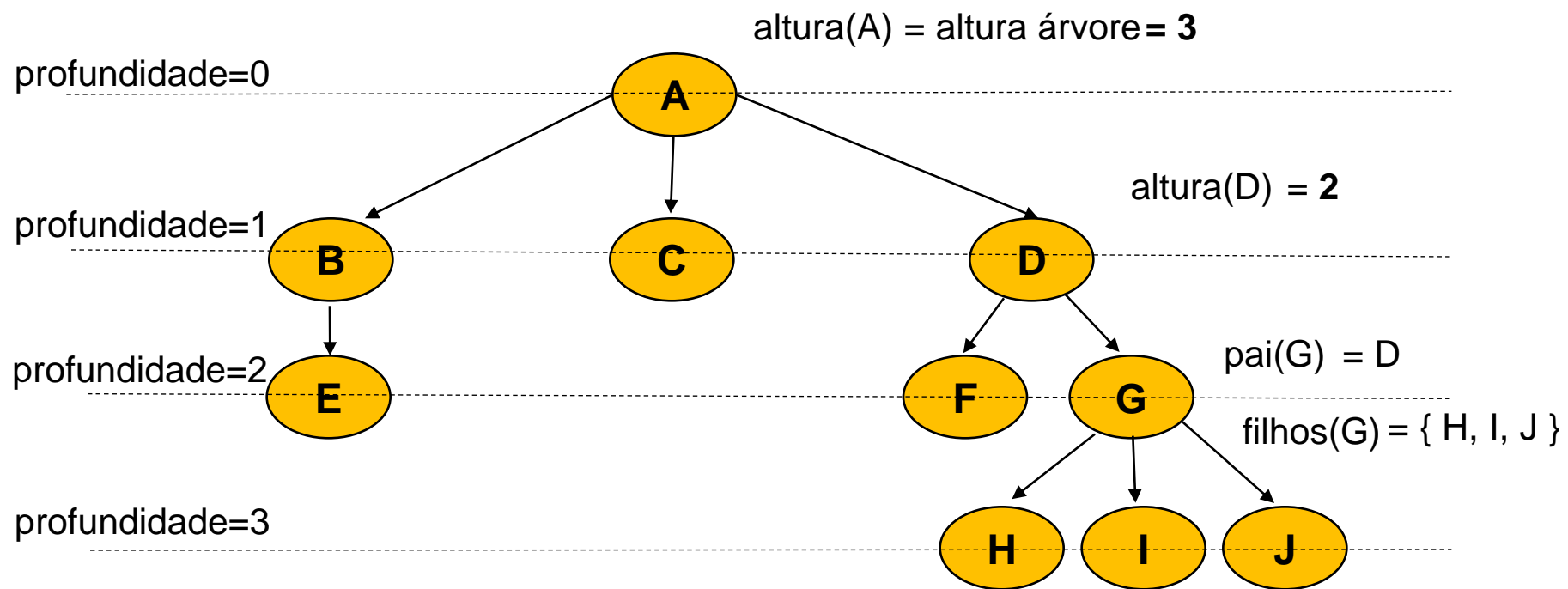
Árvores

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



Árvores

▪ Exemplo:



Árvores Binárias

- Árvore em que cada nó tem no máximo 2 filhos.

Formas de percorrer uma árvore:

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

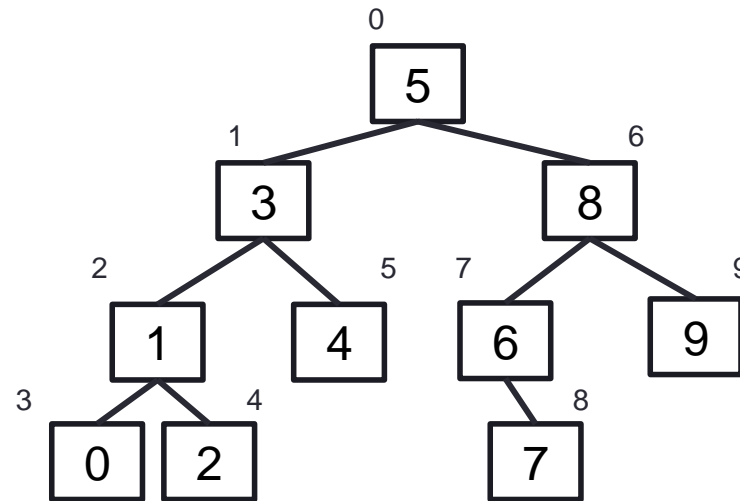
- Pré-ordem
- Em-ordem (ordem simétrica)
- Pós-ordem
- Por nível

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**).



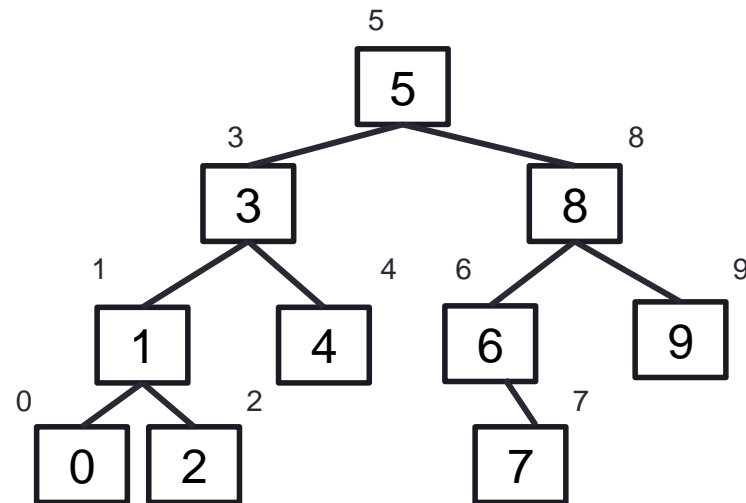
Percurso em pré-ordem: 5 3 1 0 2 4 8 6 7 9

Árvores Binárias

- **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.

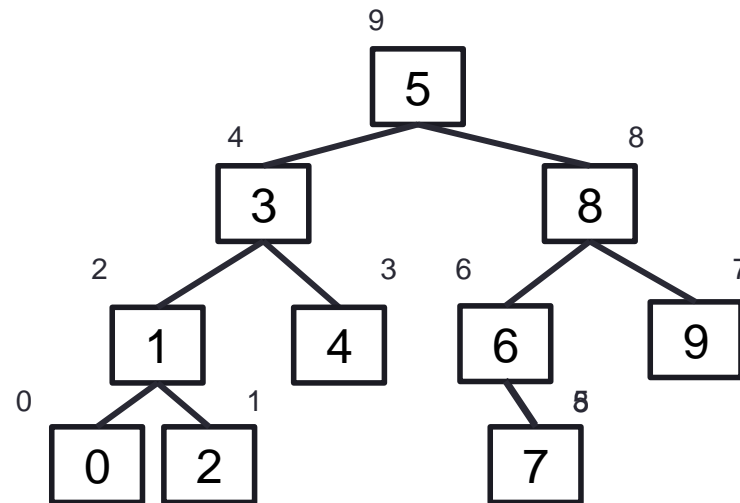


Percurso em-ordem: 0 1 2 3 4 5 6 7 8 9

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



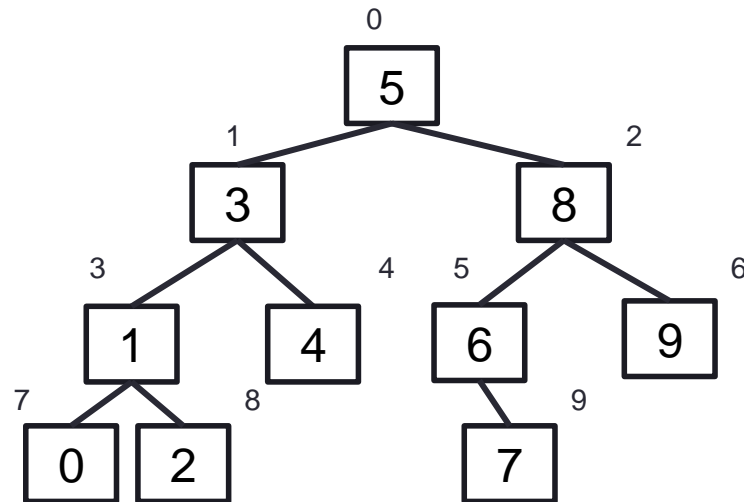
Percurso em pós-ordem: 0 2 1 4 3 7 6 9 8 5

Árvores Binárias

- *por-nível:*

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

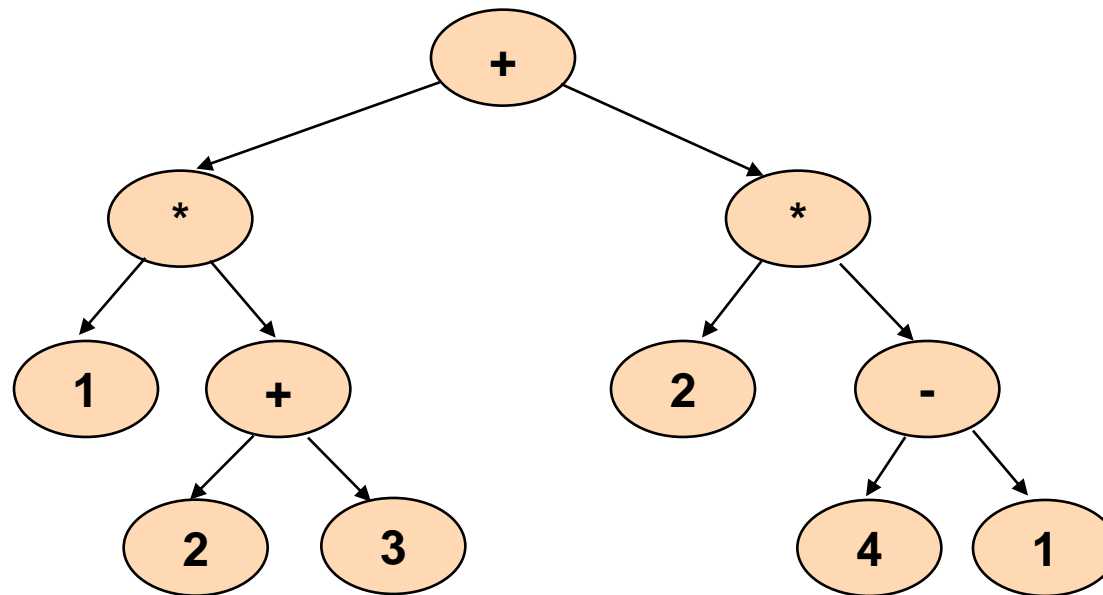
Também é conhecido como breadth-first



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

Árvores Binárias - Aplicações

- Expressões aritméticas



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

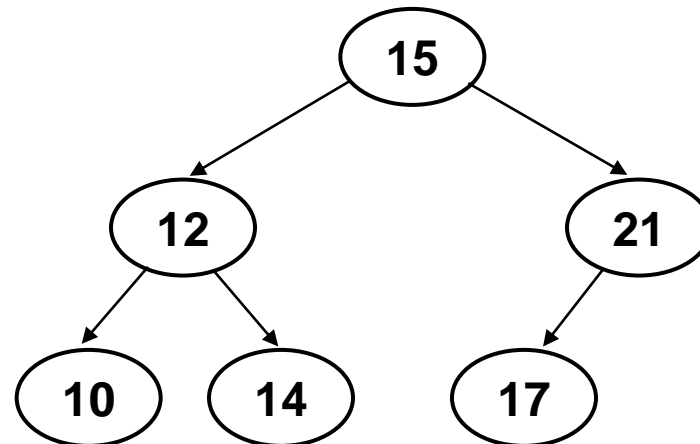
- Ordenação de dados
- Pesquisa em uma lista

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 em C

```
#include <stdio.h>
#include <stdlib.h>

typedef struct no {
    int valor;
    struct no *fe;      // filho da esquerda
    struct no *fd;      // filho da direita
} Arvore;
```

Árvores Binárias de Pesquisa em C

```
Arvore *insere_arvore(Arvore *arv, int x){

    if (arv == NULL){
        arv = (Arvore *)malloc(sizeof(Arvore));
        arv->valor = x;
        arv->fd = NULL;
        arv->fe = NULL;
    } else {
        if (x < arv->valor){
            arv->fe = insere_arvore(arv->fe,x);
        } else {
            arv->fd = insere_arvore(arv->fd,x);
        }
    }
    return arv;
}
```

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

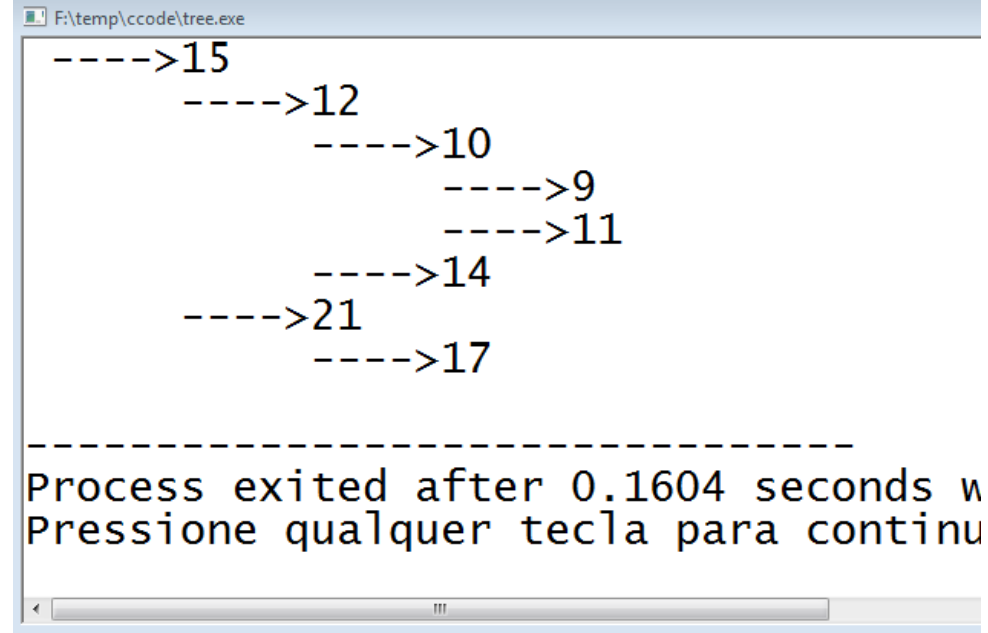
```
void libera_arvore(Arvore *arv) {  
    if(arv->fe != NULL) {  
        libera_arvore(arv->fe);  
        arv->fe = NULL;  
    }  
    if (arv->fd != NULL) {  
        libera_arvore(arv->fd);  
        arv->fd = NULL;  
    }  
  
    if(arv->fe == NULL && arv->fd == NULL) {  
        //printf("Liberando: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

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 = insere_arvore(a, 15);  
    a = insere_arvore(a, 12);  
    a = insere_arvore(a, 21);  
    a = insere_arvore(a, 10);  
    a = insere_arvore(a, 14);  
    a = insere_arvore(a, 17);  
    a = insere_arvore(a, 11);  
    a = insere_arvore(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.