

# Estruturas de Dados II

## Árvores: conceitos, árvores binárias e percurso

**Prof. Rodrigo Minetto**

**Prof<sup>a</sup>. Juliana de Santi**

Universidade Tecnológica Federal do Paraná

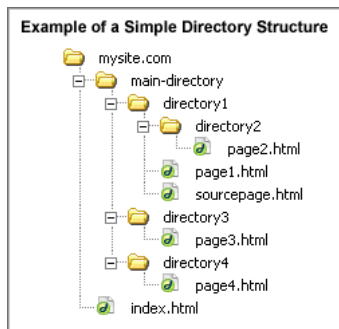
Material compilado de: Cormen, IC-UNICAMP e IME-USP

# Sumário

- 1 Árvores - conceitos
- 2 Árvore binária
- 3 Percursos
- 4 Altura
- 5 Espelhamento

# Introdução

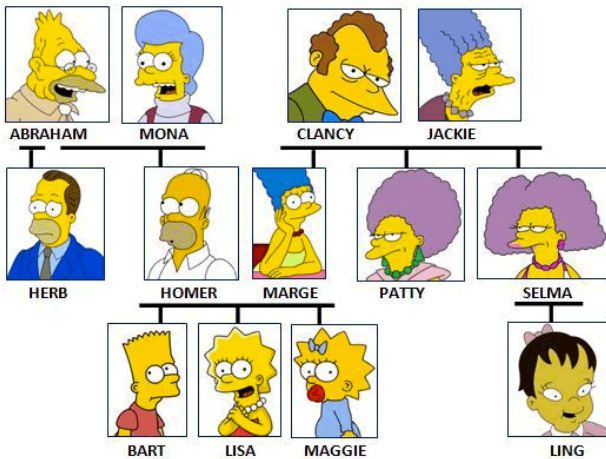
**Árvores** (na computação): **estruturas de dados** não sequenciais que organizam os dados de forma **hierárquica**.



# Introdução

## Exemplo: árvore genealógica

THE SIMPSONS

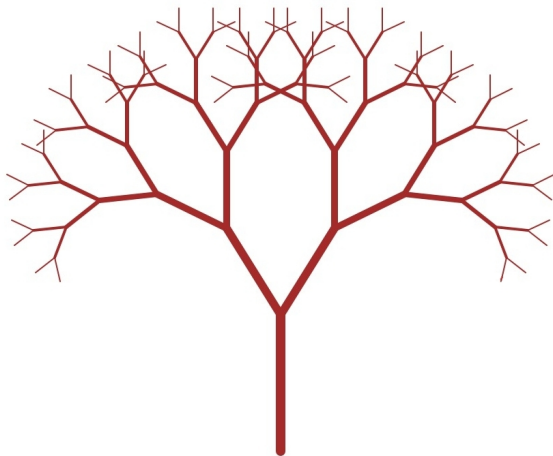


**Definição:** uma árvore é um conjunto **T** finito e não vazio, **n**  $>$  0, de objetos (estruturas elementares) chamados **nós**. Tal que **T** consiste de:

- um nó especial **r**, denominado **raiz** de  $T$ .
- $m \geq 0$  conjuntos disjuntos de nós, denominados **subárvores** ( $T_1, T_2, \dots, T_m$ ) de  $T$  (ou da raiz **r** de  $T$ ).

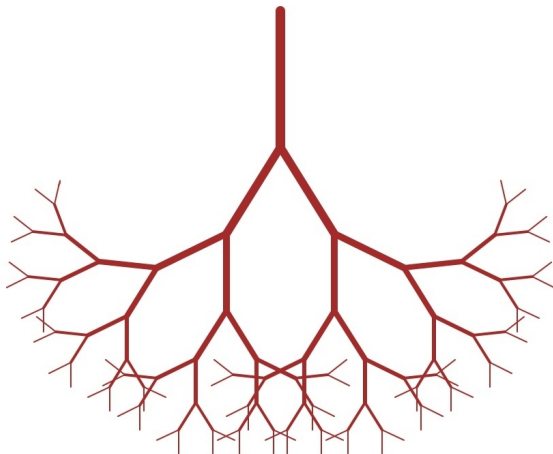
## Árvores - Conceitos

É tradicional desenhar as estruturas de árvores com a raiz para cima e as folhas para baixo.



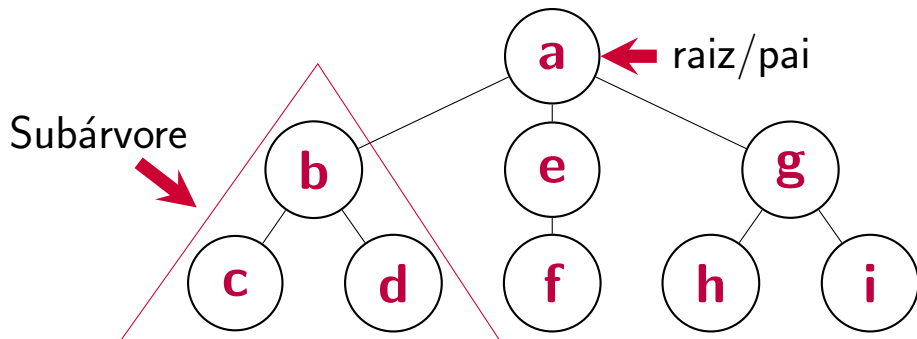
## Árvores - Conceitos

É tradicional desenhar as estruturas de árvores com a raiz para cima e as folhas para baixo.



## Árvores - Conceitos

É tradicional desenhar as estruturas de árvores com a raiz para cima e as folhas para baixo.

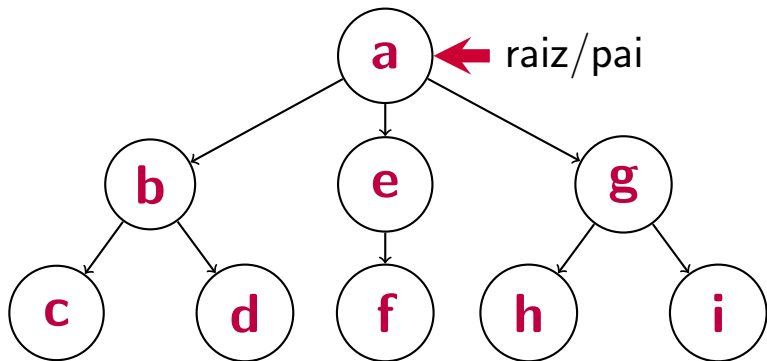


A forma mais natural de definir uma estrutura de árvore é usando a **recursividade**.



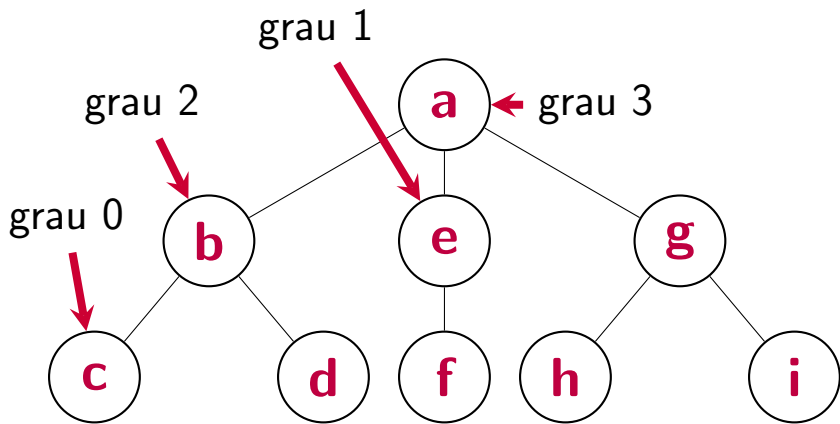
## Árvores - Conceitos

Apesar de não representarmos explicitamente a **direção dos ponteiros**, eles **apontam sempre do pai/raíz para os filhos/subárvores**.



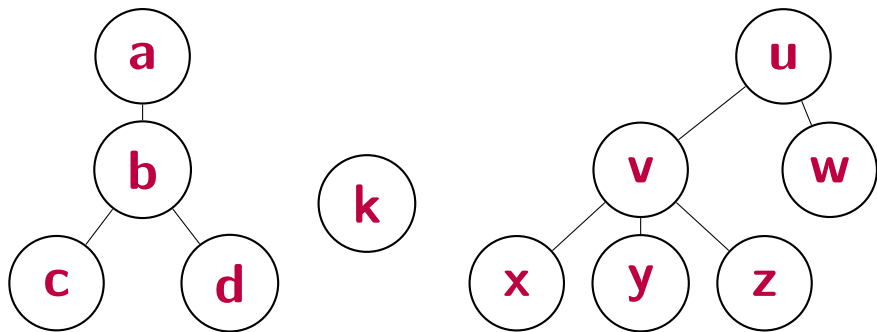
## Árvores - Conceitos

- **Grau:** número de subárvores de um nó.



- **Grau da árvore:** grau máximo entre todos os nós.
- **Nó folha ou externo:** nó com grau 0 (não tem filhos, sem subárvores).
- **Nó interno:** nó que possui filhos (subárvores).

- **Floresta:** conjunto de árvores disjuntas.



### Tipos comuns de árvores:

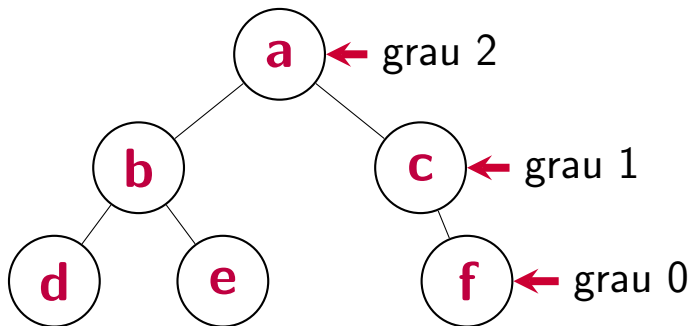
- Binária.
- Binária de pesquisa.
- AVL.
- Rubro-Negra.
- Splay-Trees.
- B.

# Sumário

- 1 Árvores - conceitos
- 2 **Árvore binária**
- 3 Percursos
- 4 Altura
- 5 Espelhamento

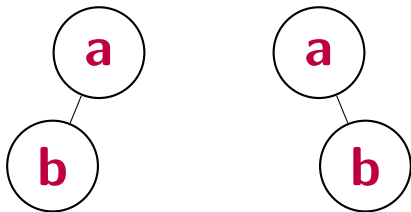
## Árvore binária

- **Árvore binária:** cada nó têm no máximo 2 filhos (ou seja têm grau 0, 1 ou 2).



## Árvore binária

Filhos são denominados subárvores da **esquerda** ou da **direita** de uma árvore maior, e qualquer uma das subárvores pode ser vazia. Assim, as duas subárvores abaixo são distintas.

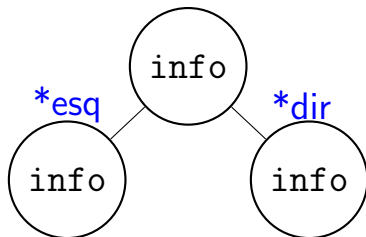




## Árvore binária - estrutura

Em C, a representação de um **nó da árvore binária** pode ser dada por:

```
typedef struct arvore {  
    char info;  
    struct arvore *esq;  
    struct arvore *dir;  
} Arvore;
```



## Árvore binária - estrutura

- Listas encadeadas são representadas por um ponteiro para o primeiro nó.
- De forma semelhante, a estrutura da árvore é representada por **um ponteiro para o nó raiz**. Através do ponteiro para o nó raiz da árvore, tem-se acesso aos demais nós.

## Árvore binária - TAD

Uma possível interface do tipo abstrato (TAD) para representar uma árvore binária pode ser definido em um arquivo **arvore.h** por:

```
Arvore*  cria_arv_vazia (void);
Arvore*  constroi_arv (char c, Arvore *e, Arvore *d);
int      verifica_arv_vazia (Arvore *a);
void     libera_arv (Arvore *a);
int      pertence_arv (Arvore *a, char c);
void     imprime_arv (Arvore *a);
```

## Árvore binária - Criar vazia

Como uma árvore é representada pelo endereço do nó raiz, uma árvore **vazia** é representada pelo valor **NULL**. Assim, a função que cria uma árvore vazia pode ser simplesmente:

```
Arvore* cria_arv_vazia (void) {  
    return NULL;  
}
```

## Árvore binária - Construir

Para **construir** árvores não-vazias, podemos ter uma função que **cria um nó** dadas a informação e duas subárvores (esquerda e direita). O valor de **retorno** é o **endereço** do nó:

```
Arvore* constroi_arv (char c, Arvore *e, Arvore *d) {  
    Arvore *no = (Arvore*)malloc(sizeof(Arvore));  
    no->info = c;  
    no->esq = e;  
    no->dir = d;  
    return no;  
}
```

## Árvore binária - Verificar se vazia

O término de uma recursão é dado ao encontrar uma **árvore vazia**, portanto é importante ter uma operação que testa esta condição:

```
int verifica_arv_vazia (Arvore *a) {  
    return (a == NULL);  
}
```

## Árvore binária - Liberar

Uma maneira de liberar a memória alocada pela estrutura da árvore é através de **recursão**:

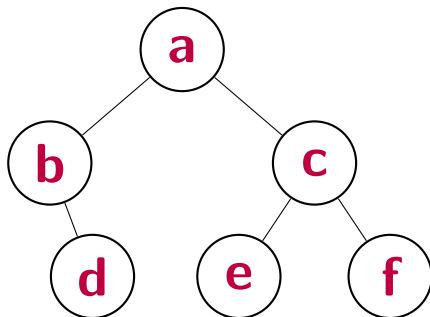
```
void arv_libera (Arvore* a) {  
    if (!verifica_arv_vazia(a)) {  
        arv_libera (a->esq);  
        arv_libera (a->dir);  
        free(a);  
    }  
}
```

**Atenção:** um cuidado a ser tomado é liberar as subárvores **antes** de liberar o nó raiz, para que o acesso a elas não seja **perdido** antes de serem removidas.



## Árvore binária - Inserir

**Exemplo:** considere a seguinte árvore binária:



## Árvore binária - Inserir

Podemos criá-la através dos seguinte passos:

```
int main (int argc, char *argv[]) {  
    Arvore *a, *a1, *a2, *a3, *a4, *a5;  
    a1 = constroi_arv('d', cria_arv_vazia(), cria_arv_vazia());  
    a2 = constroi_arv('b', cria_arv_vazia(), a1);  
    a3 = constroi_arv('e', cria_arv_vazia(), cria_arv_vazia());  
    a4 = constroi_arv('f', cria_arv_vazia(), cria_arv_vazia());  
    a5 = constroi_arv('c', a3, a4);  
    a  = constroi_arv('a', a2, a5);  
    return 0;  
}
```

## Árvore binária - Inserir

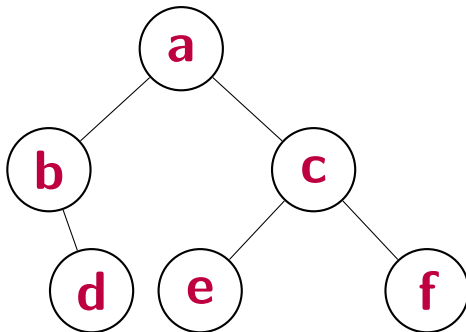
De forma alternativa, a árvore poderia ser criada com uma **única atribuição**:

```
Arvore *a = constroi_arv ('a',  
    constroi_arv('b',  
        cria_arv_vazia(),  
        constroi_arv('d',cria_arv_vazia(),cria_arv_vazia())  
    ),  
    constroi_arv('c',  
        constroi_arv('e',cria_arv_vazia(),cria_arv_vazia()),  
        constroi_arv('f',cria_arv_vazia(),cria_arv_vazia())  
    )  
);
```

## Árvore binária - Inserir

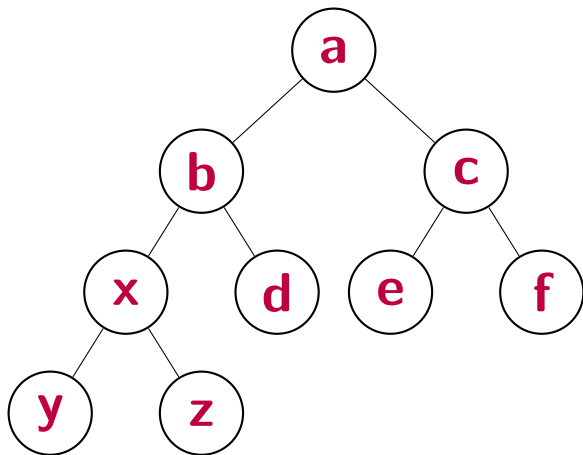
A adição de uma subárvore-esquerda em 'b' é feita por:

```
a->esq->esq = constroi_arv ('x',  
    constroi_arv('y',cria_arv_vazia(),cria_arv_vazia()),  
    constroi_arv('z',cria_arv_vazia(),cria_arv_vazia())  
);
```



## Árvore binária - Inserir

Que produz a seguinte árvore binária:



# Sumário

- 1 Árvores - conceitos
- 2 Árvore binária
- 3 Percursos**
- 4 Altura
- 5 Espelhamento

## Árvores - Percurso

Um **percurso** corresponde a uma **visita** sistemática **a cada um dos nós** da árvore. Muitas operações em árvores binárias envolvem o percurso de todas as subárvores, com a execução de alguma ação de tratamento de cada nó.

Por exemplo, se cada nó da árvore possui um campo que armazena o salário, então podemos querer **visitar** cada nó **para fazer um reajuste salarial**. Não podemos esquecer de nenhum nó, nem queremos visitar um nó mais do que uma vez. Nesse caso a ordem da visita não é importante. Mas **em algumas outras aplicações, queremos visitar os nós em certa ordem** desejada.



**Visitar** um nó significa **trabalhar com a informação** do nó (por exemplo: imprimir, modificar, etc). Contudo, **durante um percurso podemos passar várias vezes por alguns dos nós sem visitá-los.**

O momento da visita ao nó é determinado em função da **ordem de visitação**. É comum percorrer uma árvore em uma das seguintes ordens:

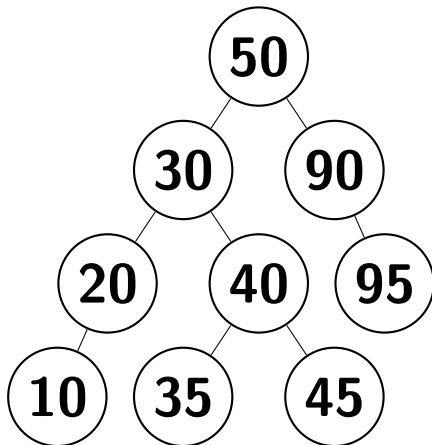
- **Pré-ordem** (R, E, D).
- **In-ordem** (E, R, D).
- **Pós-ordem** (E, D, R).

- Pré-ordem (**R, E, D**):
  - **Visitar** a **raiz**.
  - Percorrer a sua **subárvore esquerda** em pré-ordem.
  - Percorrer a sua **subárvore direita** em pré-ordem.

## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

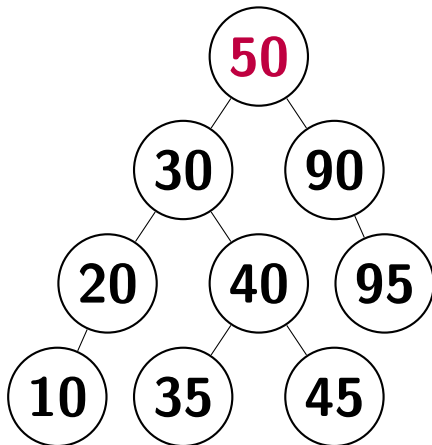
Exemplo:



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

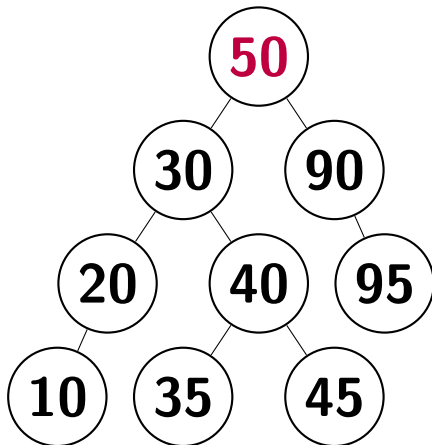
Exemplo:



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

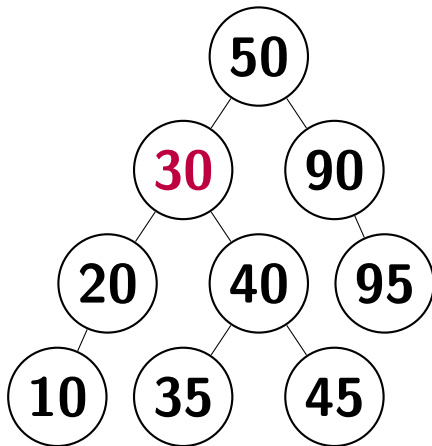
Exemplo: **50**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

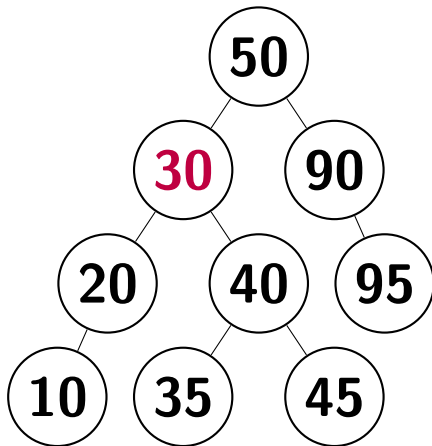
Exemplo: **50**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

Exemplo: **50,30**

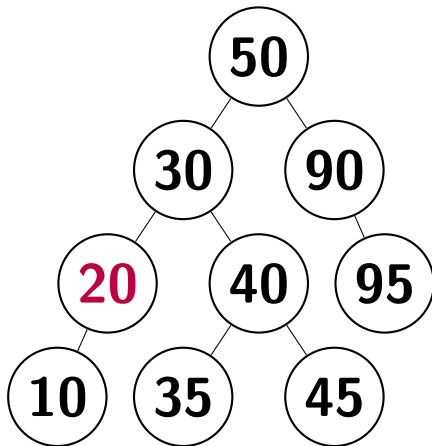




## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

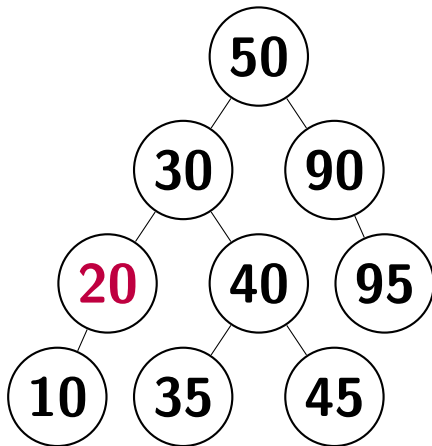
Exemplo: **50,30**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

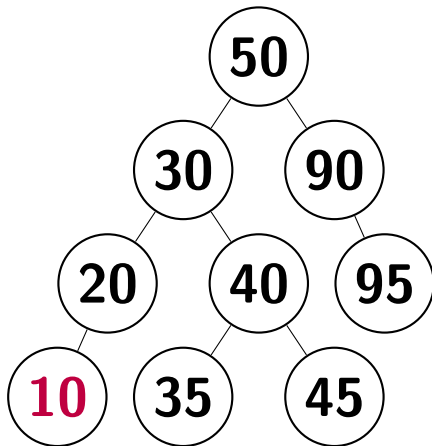
Exemplo: **50,30,20**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

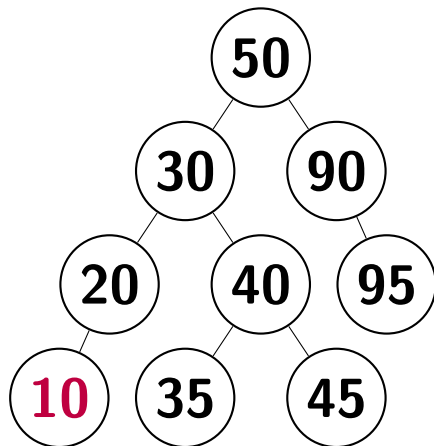
Exemplo: **50,30,20**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

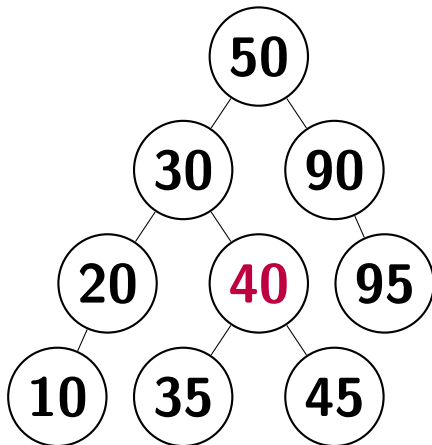
Exemplo: **50,30,20,10**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

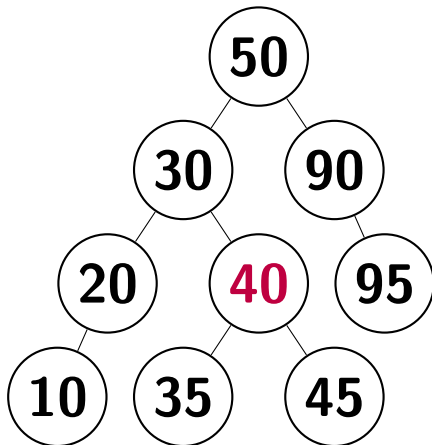
Exemplo: **50,30,20,10**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

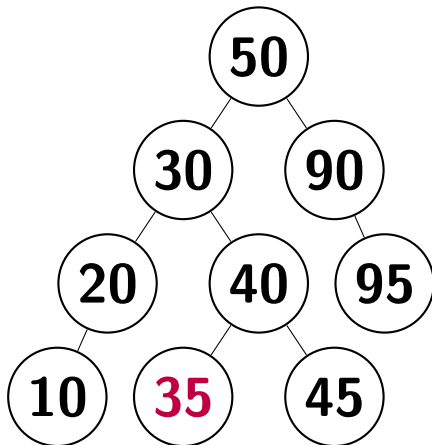
Exemplo: **50,30,20,10,40**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

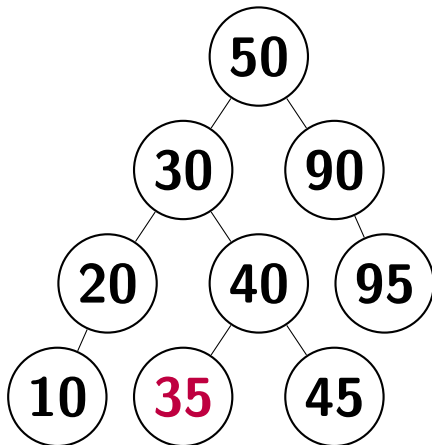
Exemplo: **50,30,20,10,40**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

Exemplo: **50,30,20,10,40,35**

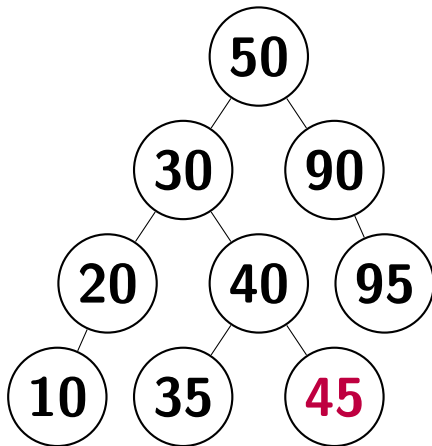




## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

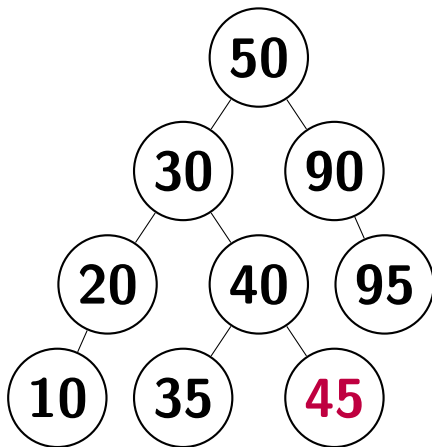
Exemplo: **50,30,20,10,40,35**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

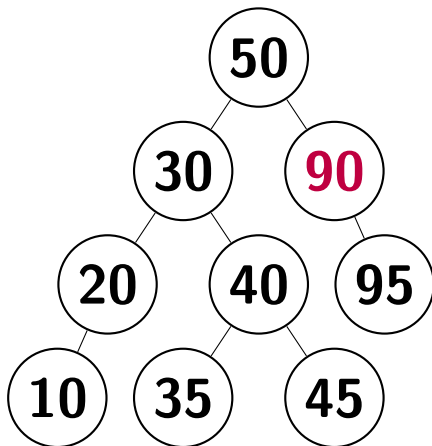
Exemplo: **50,30,20,10,40,35,45**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

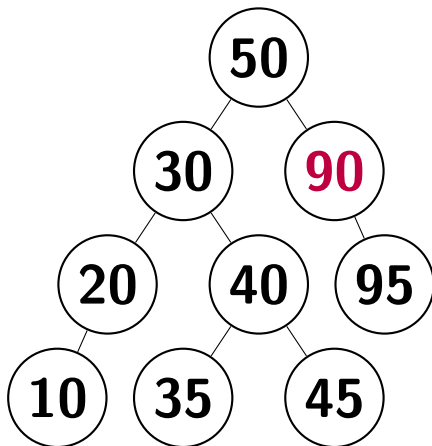
Exemplo: **50,30,20,10,40,35,45**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

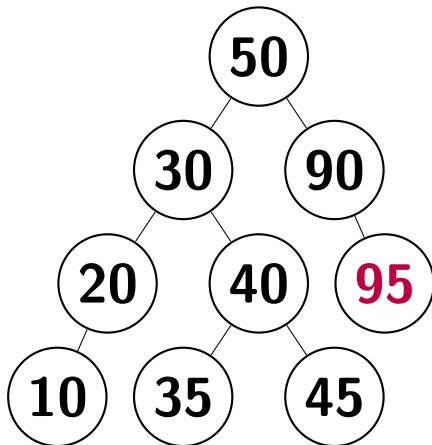
Exemplo: **50,30,20,10,40,35,45,90**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

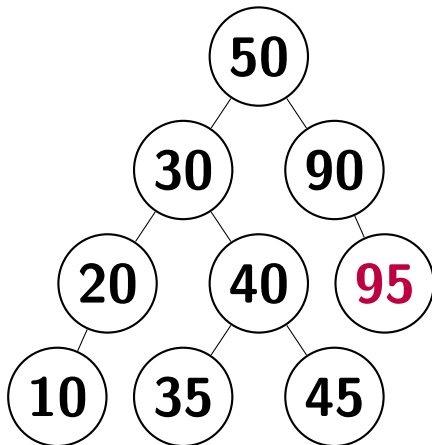
Exemplo: **50,30,20,10,40,35,45,90**



## Introdução - Percurso em árvores

Pré-ordem (R,E,D):

Exemplo: **50,30,20,10,40,35,45,90,95.**



## Árvores - Percurso

O percurso em pré-ordem segue os nós até chegar aos mais **“profundos”**, em **“ramos”** de subárvores da esquerda para a direita. É conhecida pelo nome de percurso em **produndidade**.

Percurso in-ordem: também conhecido como **ordem simétrica**.

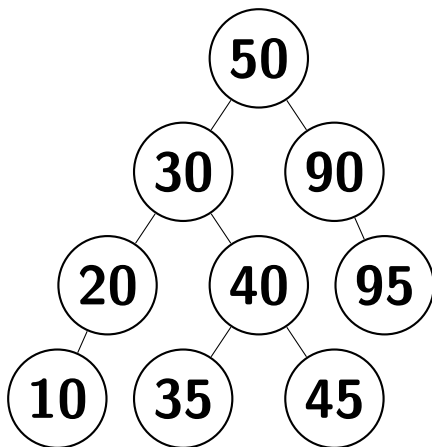
- Percurso In-ordem (**E, R, D**):
  - Percorrer a sua **subárvore esquerda** em in-ordem.
  - **Visitar** a **raiz**.
  - Percorrer a sua **subárvore direita** em in-ordem.



## Introdução - Percurso em árvores

In-ordem (E, R, D)

Exemplo: **10,20,30,35,40,45,50,90,95.**

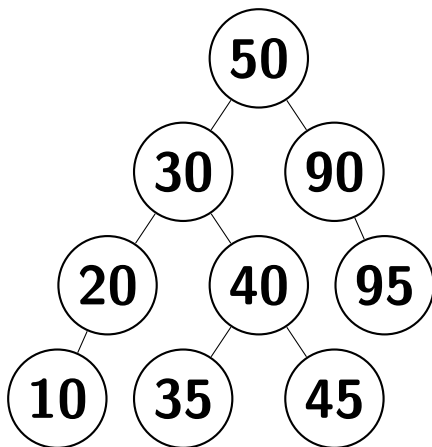


- Percurso pós-ordem (**E, D, R**):
  - Percorrer a sua **subárvore esquerda** em pós-ordem.
  - Percorrer a sua **subárvore direita** em pós-ordem.
  - **Visitar** a **raiz**.

## Introdução - Percurso em árvores

Pós-ordem (E, D, R)

Exemplo: **10,20,35,45,40,30,95,90,50.**



# Sumário

- 1 Árvores - conceitos
- 2 Árvore binária
- 3 Percursos
- 4 Altura**
- 5 Espelhamento

## Árvores - Altura

Uma propriedade fundamental de todas as árvores é que só existe um caminho da raiz para qualquer nó. Com isso, podemos definir a **altura** de uma árvore como sendo o **comprimento do caminho mais longo da raiz até uma das folhas**.

## Árvores - Altura

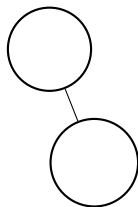
Assim, a altura de uma árvore com um único nó (a raiz) é zero e, por consequência, dizemos que a altura de uma árvore vazia é negativa e vale -1.

Exemplos:

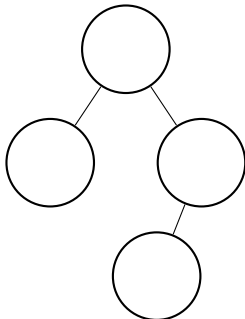
Altura = 0



Altura = 1



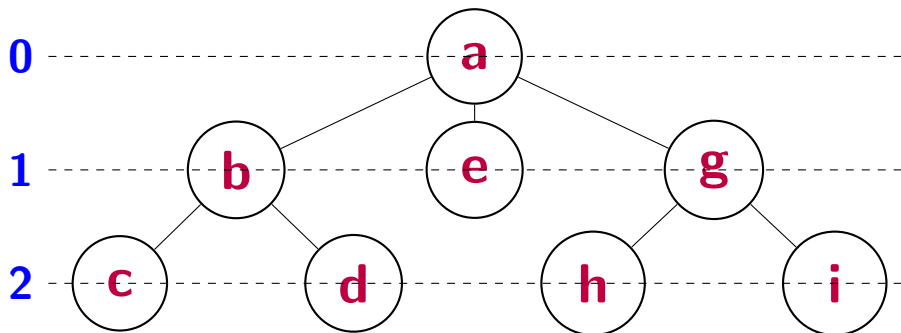
Altura = 2



## Árvores - Altura

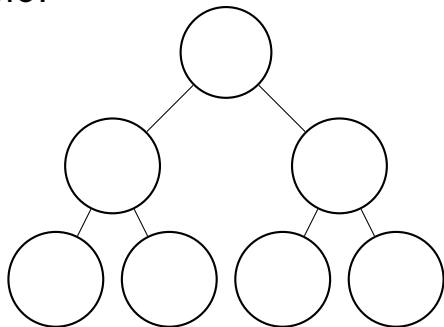
Podemos numerar os **níveis** em que os nós aparecem na árvore. A raiz está no nível 0, filhos do nó raiz no nível 1, ...

**Nível**



## Árvores - Altura

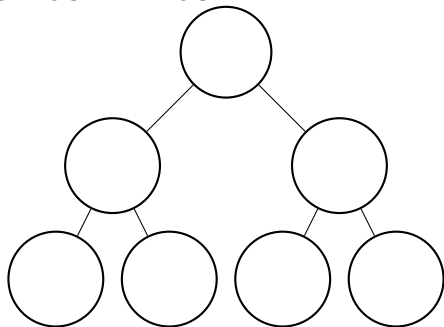
Uma árvore binária é dita **cheia (ou completa)** se todos os seus nós internos têm duas subárvores associadas e todos os nós folhas estão no último nível. Exemplo:





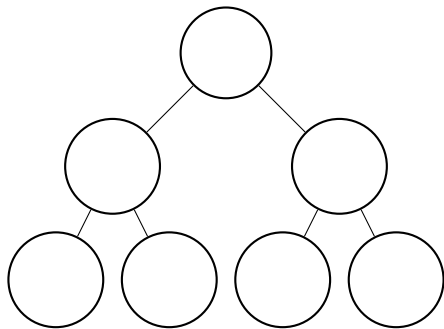
## Árvores - Altura

Note que nesse tipo de árvore temos **um** nó no nível **0**, **dois** nós no nível **1**, **quatro** nós no nível **2**, **oito** nós no nível **3**, e assim por diante. Isto é, no nível  $n$ , temos  $2^n$  nós.



## Árvores - Altura

É possível mostrar que uma árvore cheia de altura  $h$  tem um número de nós dado por  $2^{h+1} - 1$ .



## Árvores - Altura

É possível mostrar que uma árvore cheia de altura  $h$  tem um número de nós dado por  $2^{h+1} - 1$ .

- Nível 0:  $2^0 = 1$  nó
- Nível 1:  $2^1 = 2$  nós
- Nível 2:  $2^2 = 4$  nós
- ...
- Nível  $h$ :  $2^h$  nós

Total de nós:  $\sum_{i=0}^h 2^i$

## Árvores - Altura

É possível mostrar que uma árvore cheia de altura  $h$  tem um número de nós dado por  $2^{h+1} - 1$ .

Por definição (A.5 Cormen):

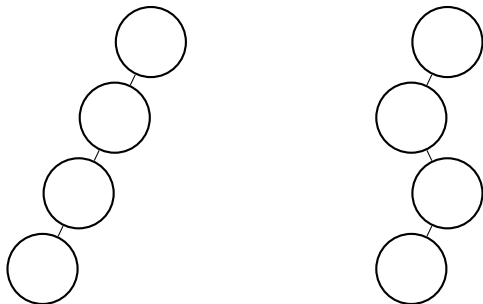
$$\sum_{k=0}^n x^k = \frac{x^{n+1} - 1}{x - 1} \quad (1)$$

Logo:

$$\sum_{i=0}^h 2^i = \frac{2^{h+1} - 1}{2 - 1} = 2^{h+1} - 1 \quad (2)$$

## Árvores - Altura

Uma árvore é dita **degenerada** se todos os seus nós internos têm uma única subárvore associada. Em uma árvore degenerada, temos um único nó em cada nível. Assim, uma árvore degenerada de altura  $h$  tem  $h + 1$  nós.



## Árvores - Altura

A altura de uma árvore é uma medida importante na avaliação da eficiência com que visitamos os nós da árvore. Uma árvore binária com  $n$  nós tem uma altura mínima proporcional a  $\log n$  (caso da árvore cheia) e uma altura máxima proporcional a  $n$  (caso da árvore degenerada). **A altura indica o esforço computacional** necessário para alcançar qualquer nó da árvore.

# Sumário

- 1 Árvores - conceitos
- 2 Árvore binária
- 3 Percursos
- 4 Altura
- 5 Espelhamento**

## Espelhamento

Duas árvores são **espelho-similares** se elas são vazias ou se elas não são vazias e suas subárvores esquerda de cada uma são **espelho-similares** as subárvores direita da outra.

