Algoritmos e Estrutura de Dados II (AE23CP-3CP)

Aula #11 - Árvores - noções básicas

Profª Luciene de Oliveira Marin
lucienemarin@utfpr.edu.br

Árvores

1. Árvores

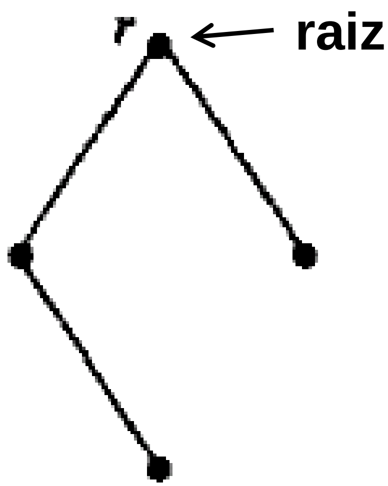
- Aplicações:
  - Armazenamento
  - Pesquisa
  - Ordenação
  - Exemplo:
    - Sistemas de Arquivos em um Sistema Operacional.
    - Representação de expressões algébricas.
    - Compiladores.

1. Árvores

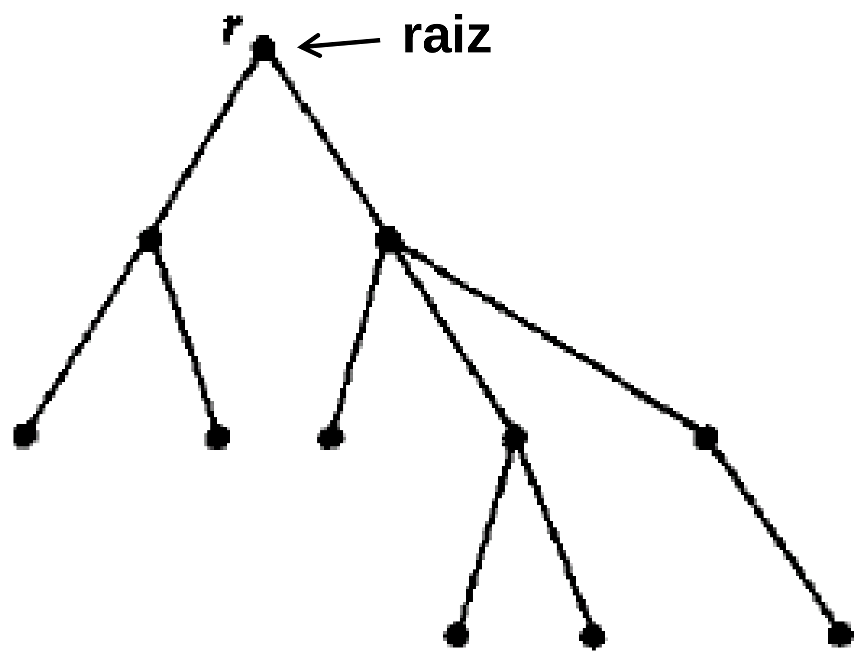
- Tipos de árvores:
  - Árvores Binária
  - AVL
  - Árvores B
  - Árvore Vermelha-Preta

1. Árvores

- Uma árvore é um grafo acíclico e conexo com um nó designado como a **raiz** da árvore.



(a)



(b)

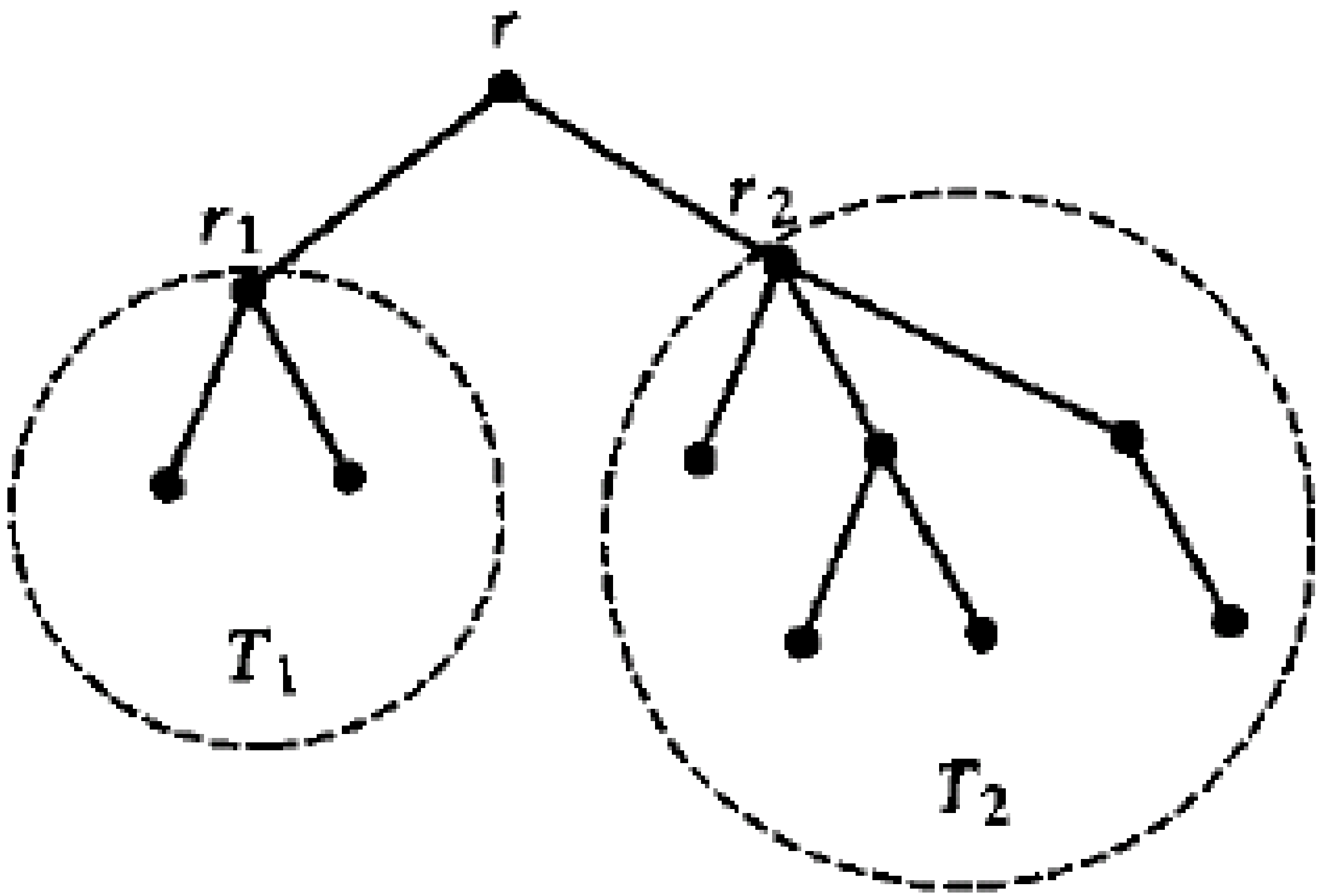
1. Árvores

- Uma árvore pode ser construída recursivamente. Um único vértice é uma árvore (este vértice é a raiz).

- Se *T_1, T_2, ..., T_t são árvores disjuntas com raízes r_1, r_2,..., r_t, o grafo formado pela ligação de um novo vértice r,* por uma única aresta a cada uma dos vértices *r_1, r_2,..., r_t constitui uma árvore de raiz r.*

- *Os vértices r_1, r_2,..., r_t são **filhos** de r, e r é **pai** de r_1, r_2,..., r_t*

1. Árvores



1. Árvores

- Como uma árvore é um grafo conexo, existe um *caminho* entre a raiz e todos os vértices da árvore; como a árvore é acíclica, este caminho é único.

- A **profundidade** de um vértice em uma árvore é o comprimento do caminho da raiz até o vértice; em particular, a raiz tem profundidade 0.

- A **altura (profundidade)** da árvore é a maior profundidade de todos seus vértices; em outras palavras, é o comprimento do maior caminho entre a raiz e um vértice.

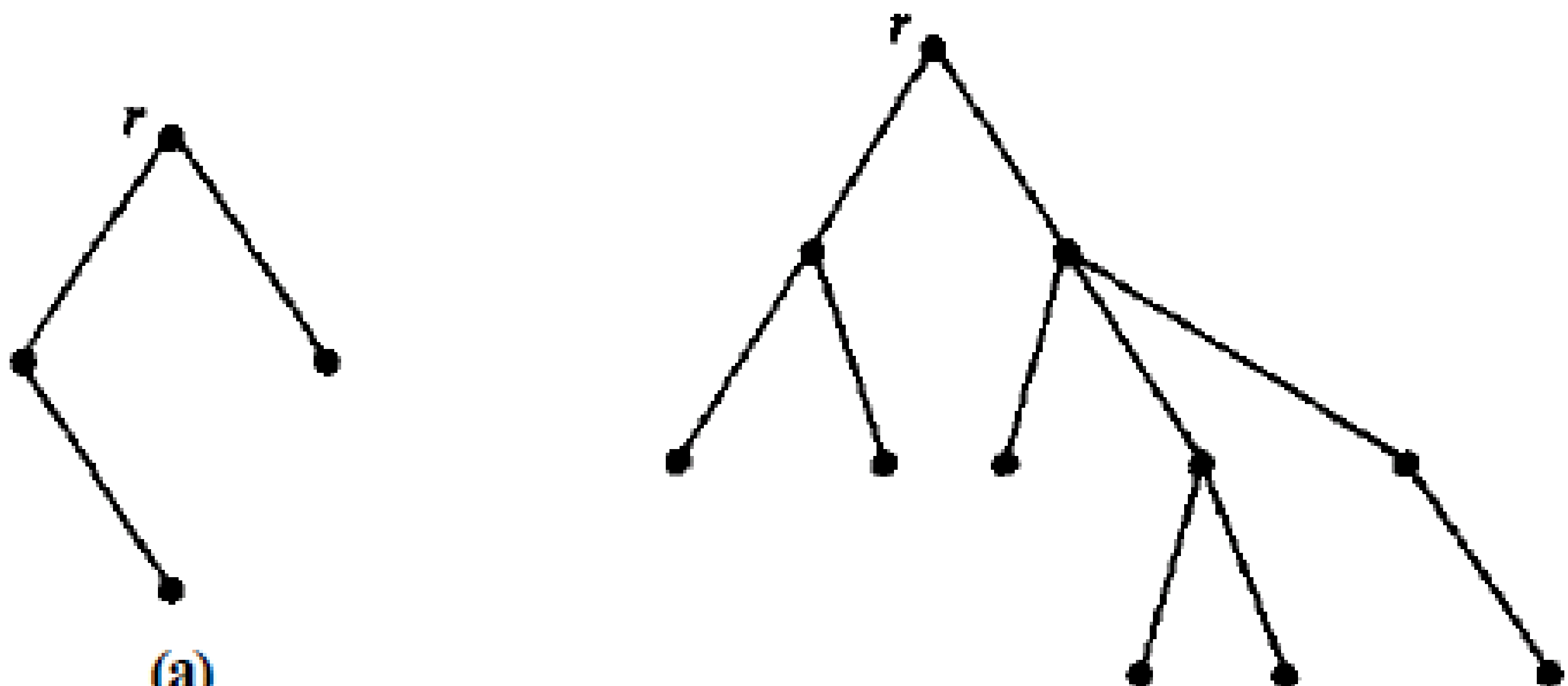
1. Árvores

- Um vértice sem filhos é chamado de **folha;** os vértices que não são folhas são chamados de **vértices internos** ou **nós internos.**

- Uma **floresta** é qualquer grafo acíclico (não necessariamente conexo);

- Portanto, uma floresta é uma coleção de árvores disjuntas.

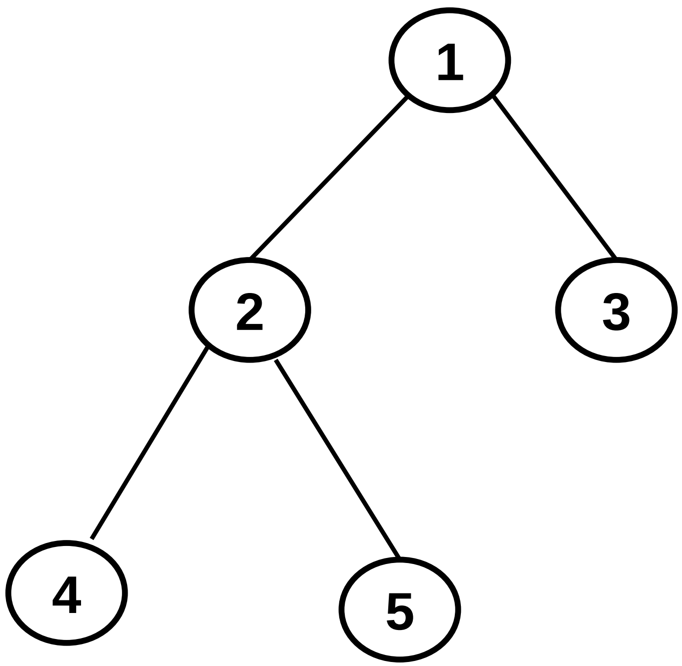
1. Árvores



(a) (b)

1. Árvores

- Exemplo:



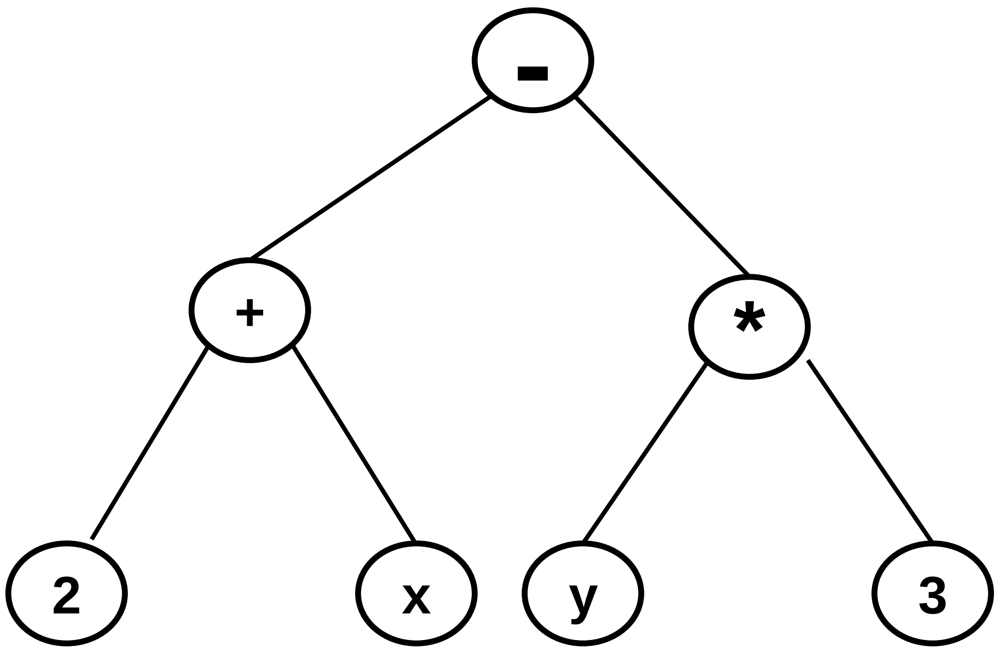
Altura da árvore = 2

Profundidade do nó 1 (raiz) = 0
Profundidade dos nós 2 e 3 = 1
Profundidade dos nós 4 e 5 = 2

Filho esquerdo de 2 = 4
Filho direito de 2 = 5

1. Árvores

Exemplo de Aplicação: expressões aritméticas.



$\equiv (2 + x) - (y * 3)$

Obs:. Caminhamento em ordem simétrica

1.1. Percurso em Árvores

Pré-Ordem:

- A raiz é visitada primeiro e depois processam-se as subárvores, da esquerda para a direita, cada uma delas em pré-ordem.

1.1. Percurso em Árvores

- Pré-ordem:

```
void pre_ordem(Arv* a)
 {
   if (a != NULL)
    {
        printf(" %c",  a -> info);
        pre_ordem(a-> esq);
        pre_ordem(a ->dir);
    }
 }
```

1.1. Percurso em Árvores

Ordem Simétrica:

- A subárvore da esquerda é percorrida em ordem simétrica, depois a raiz é visitada e depois as outras subárvores são visitadas da esquerda para a direita, sempre em ordem simétrica.
- Se a árvore for binária a raiz é visitada entre as duas subárvores.

1.1. Percurso em Árvores

Ordem Simétrica

```
void ordem_simetrica(Arv* a)
 {
   if (a != NULL){
        ordem_simetrica(a-> esq);
        printf(" %c",  a -> info);
        ordem_simetrica(a ->dir);
    }
  }
```

Percurso em Árvores

Pós-ordem:

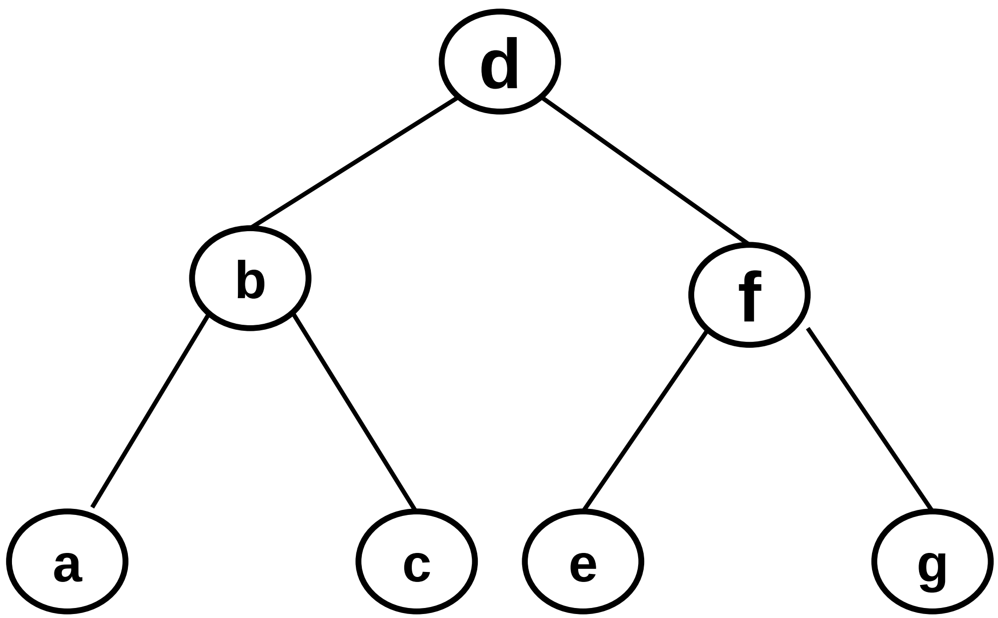
- A raiz é a última a ser visitada, após o percurso, em pós-ordem, de todas as subárvores da esquerda para a direita.

1.1. Percurso em Árvores

Pós-Ordem

```
 void pos_ordem(Arv* a)
 {
  if (a != NULL)
  {
    pos_ordem(a-> esq);
    pos_ordem(a ->dir);
    printf(" %c",  a -> info);
  }
}
```

1.1. Percurso em ÁrvoresPercurso em Árvores – Exemplo 1



- <u>Pré-ordem:</u> d, b, a, c, f, e, g
- <u>Ordem Simétrica</u>: a, b, c, d, e, f, g
- <u>Pós-ordem</u>: a,c,b,e,g,f,d

2. Árvores de Pesquisa

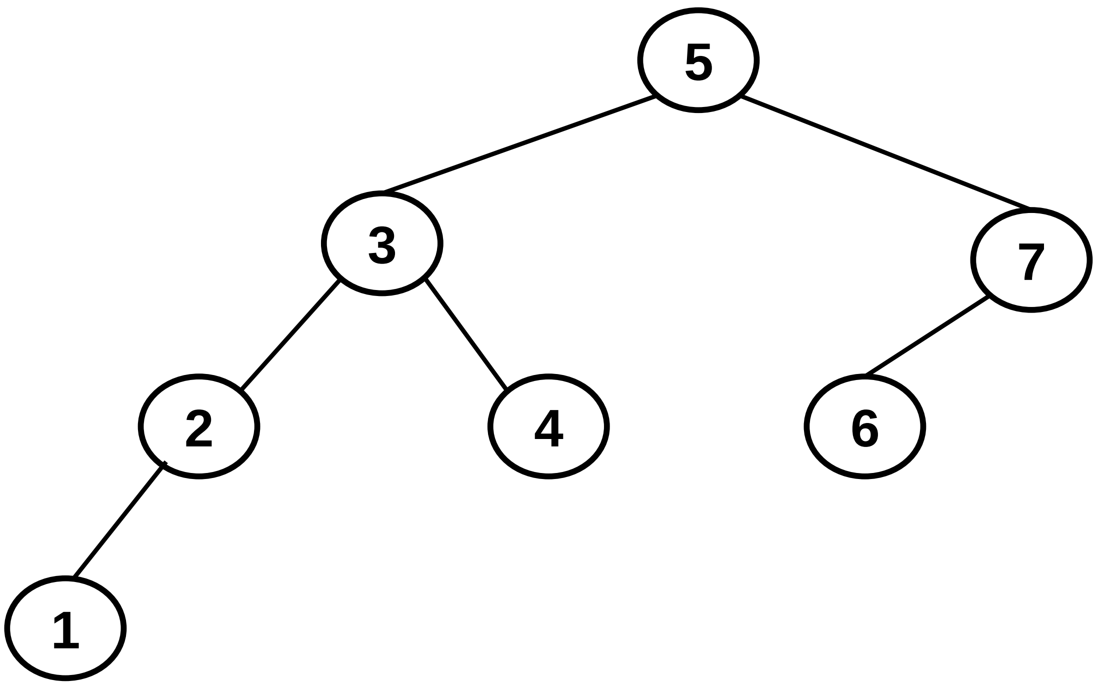
- Estrutura de dados eficiente para armazenar informações;
- É adequada ao se considerar:
  – Acesso direto e sequencial eficientes;
  – Facilidade de inserção e retirada de registros;
  – Boa taxa de utilização de memória
  – Utilização de memória primária e secundária

2.1. Árvores Binárias

- Uma *árvore binária* é definida como um conjunto finito de nós que, ou está vazio, ou consiste de um nó chamado *raiz* mais os elementos de duas subárvores distintas chamadas de *subárvores esquerda* e *direita* do nó raiz.
- Em uma árvore binária, cada nó tem no máximo duas subárvores.
- Existem apontadores para subárvores esquerda e direita em cada nó.

2.1. Árvores Binárias

- Exemplo de árvore binária

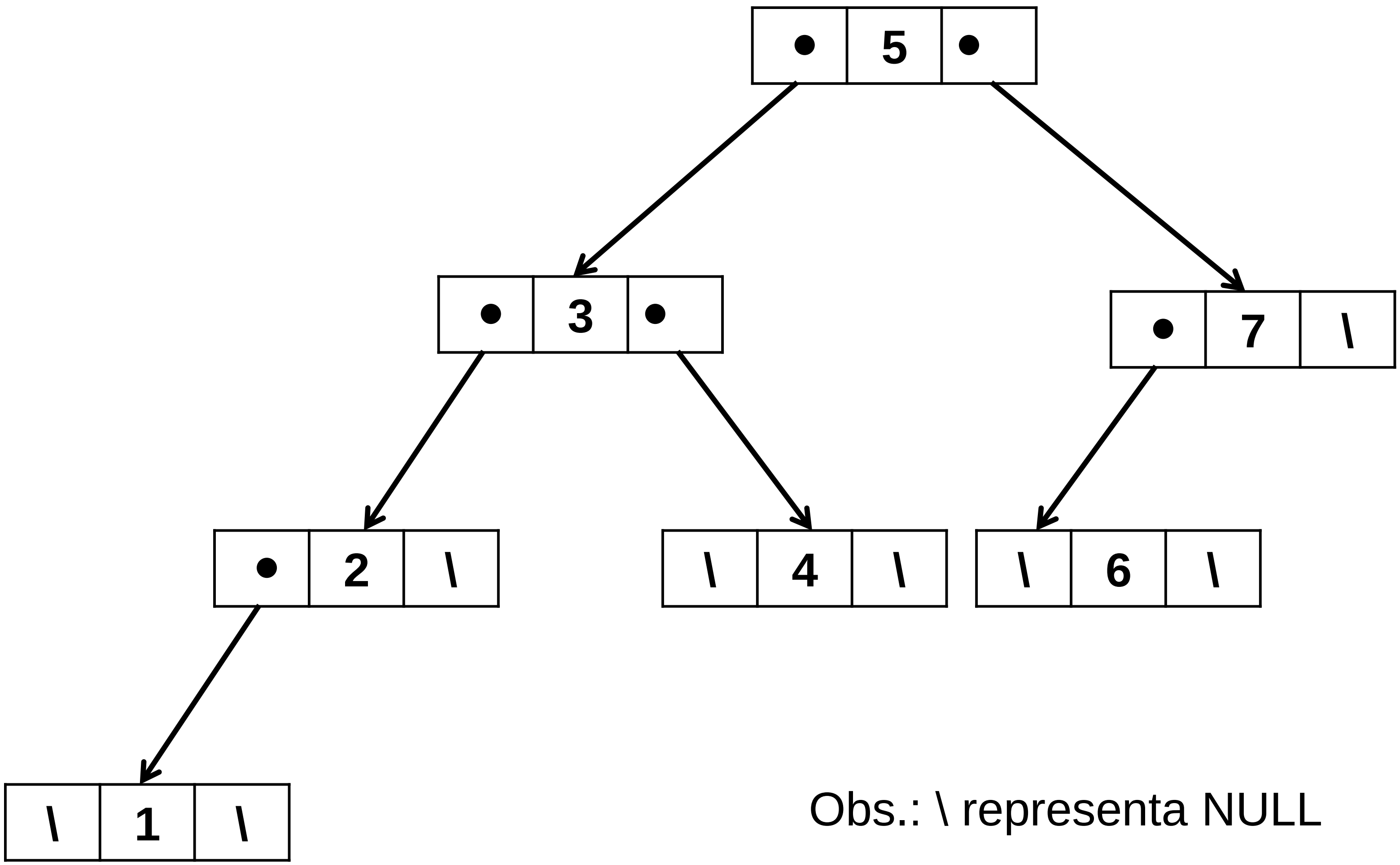


Representação de Árvores Binárias

- Esquematicamente, podemos representar um nó de uma árvore como um registro (struct) de 3 campos: informação, subárvores esquerda (sae) e subárvore direita (sad).

sae	info	sad
esquerda	**Informação**	**direita**

Representação de Árvores Binárias



Obs.: \ representa NULL

Representação de árvores

- Estrutura de Dados:

```
struct arv {
    char info;
    struct arv* esq;
    struct arv* dir;
};
typedef struct arv Arv;
```

Operações sobre árvores

- Inicialização de árvore vazia:

```
Arv* inicializa(void)
{
    return NULL;
}
```

Operações sobre árvores

- Criação de árvores não vazias:

```
Arv* cria(char c, Arv* sae, Arv* sad){
            Arv* p = (Arv*) malloc(sizeof(Arv));
      p -> info = c;
      p -> esq = sae;
      p -> dir = sad;
      return p;
 }
```

Operações sobre árvores

- Visualização: (em pré-ordem)

```
void imprime(Arv* a)
{
    if (a != NULL){
        printf("%c", a->info); /*mostra raiz */
        imprime(a->esq);    /* mostra sae */
        imprime(a->dir);    /* mostra sad */
}
```

Operações sobre árvores

Desalocar memória

```
Arv* desaloca(Arv* a) {
        if (a != NULL){
          desaloca(a->esq);
          desaloca (a->dir);
           free(a);
      }
      return NULL;
  }
```

Operações sobre árvores

- Busca: retorna um valor booleano (1 ou 0), indicando a ocorrência ou não do valor c na árvore

```
int busca(Arv* a, char c){
    if (a == NULL) return 0;
    else
       return a -> info == c
                        || busca (a -> esq, c)
                        || busca (a -> dir, c);
}
```

Operações sobre árvores

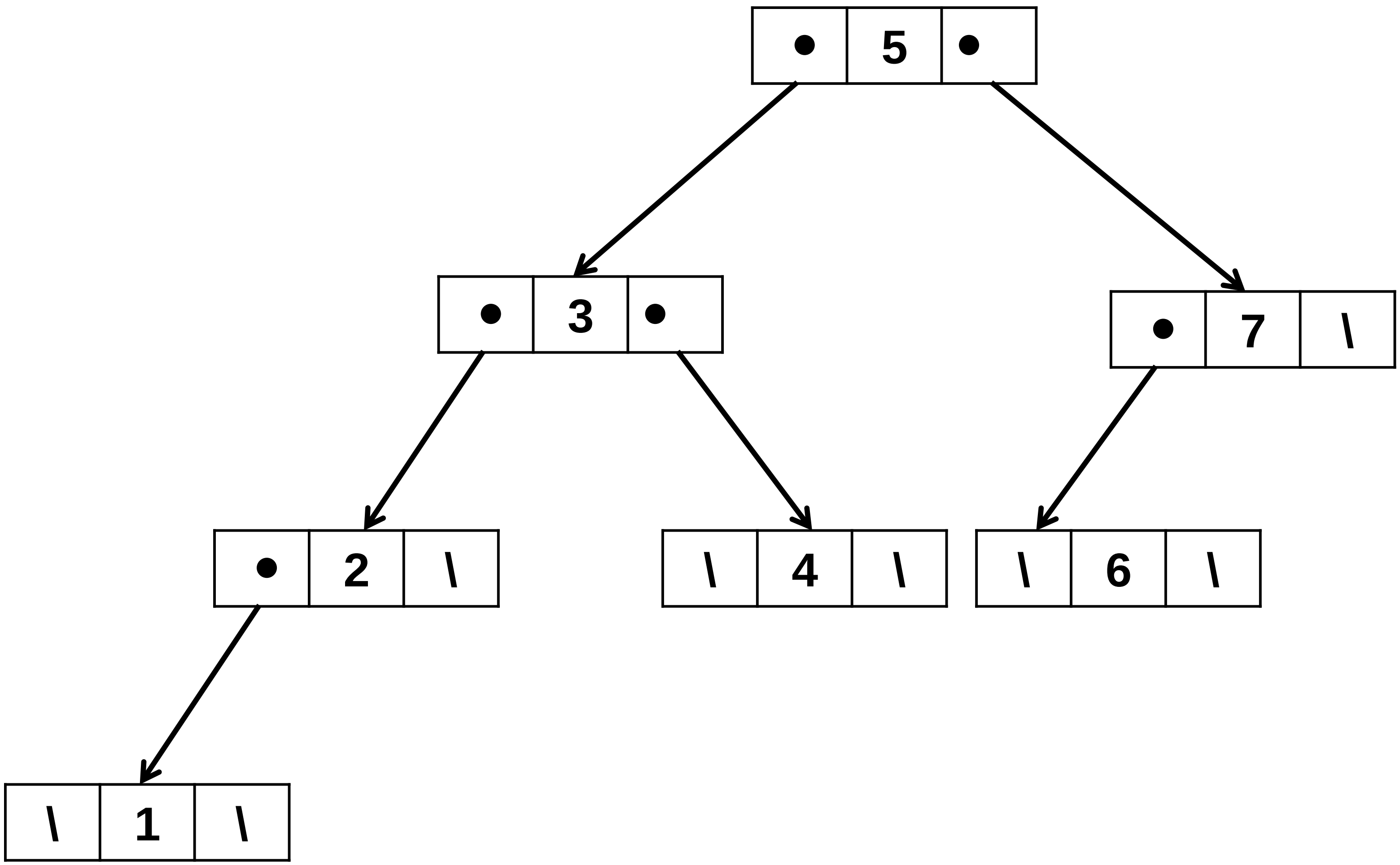
A expressão:

```
return a -> info == c
              || busca (a -> esq, c)
              || busca (a -> dir, c);
```

É equivalente a:

```
if (c == a-> info)
    return 1;
else if (busca(a -> esq, c)) return 1;
else return (busca(a -> dir, c))
```

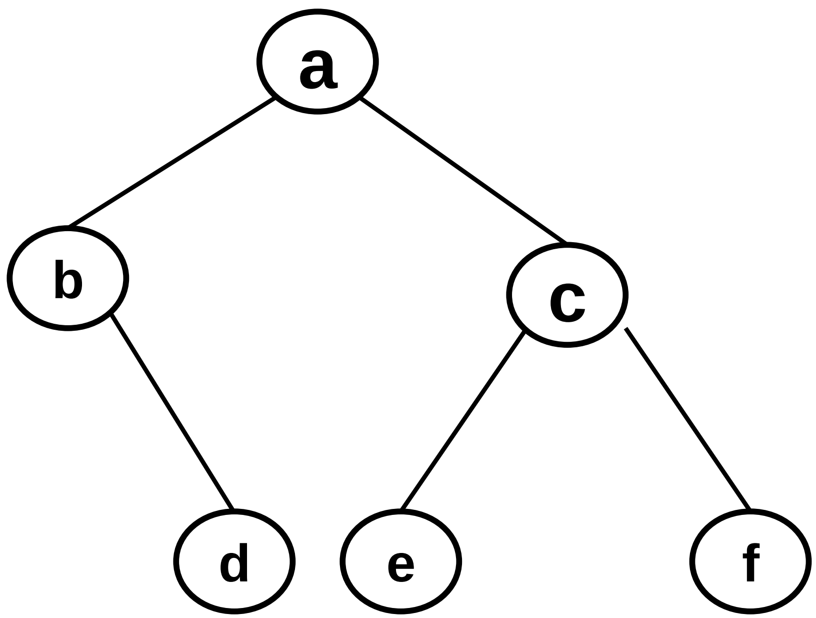
Exemplo de construção 1



Exemplo de construção 1

- Arv* a1 = cria(‘1’,inicializa(),inicializa());
- Arv* a2 = cria(‘2’,a1,inicializa());
- Arv* a3 = cria(‘4’,inicializa(),inicializa());
- Arv* a4 = cria(‘3’,a2,a3);
- Arv* a5 = cria(‘6’,inicializa(),inicializa());
- Arv* a6 = cria(‘7’,a5,inicializa());
- Arv* a7 = cria(‘5’,a4,a6);

Exemplo de Construção 2



Forma alternativa

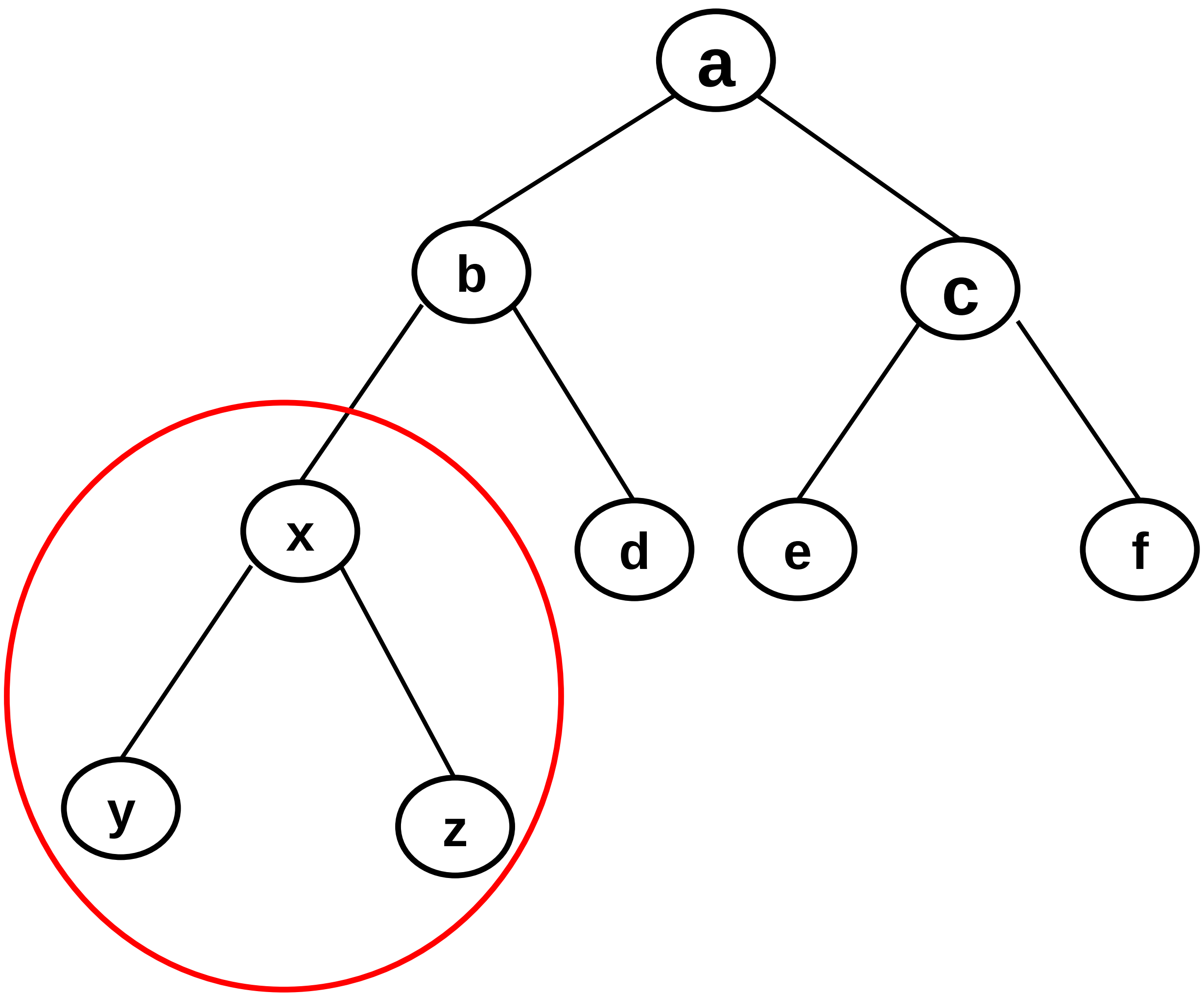
```
Arv* a = cria('a',
             cria('b',
                   inicializa(),
                   cria('d',inicializa(),inicializa())),
              cria('c',
                   cria('e',inicializa(),inicializa())
                   cria('f',inicializa(),inicializa())
                  )
              );
```

Exemplo de construção

Inserir novos elementos a uma árvore existente:

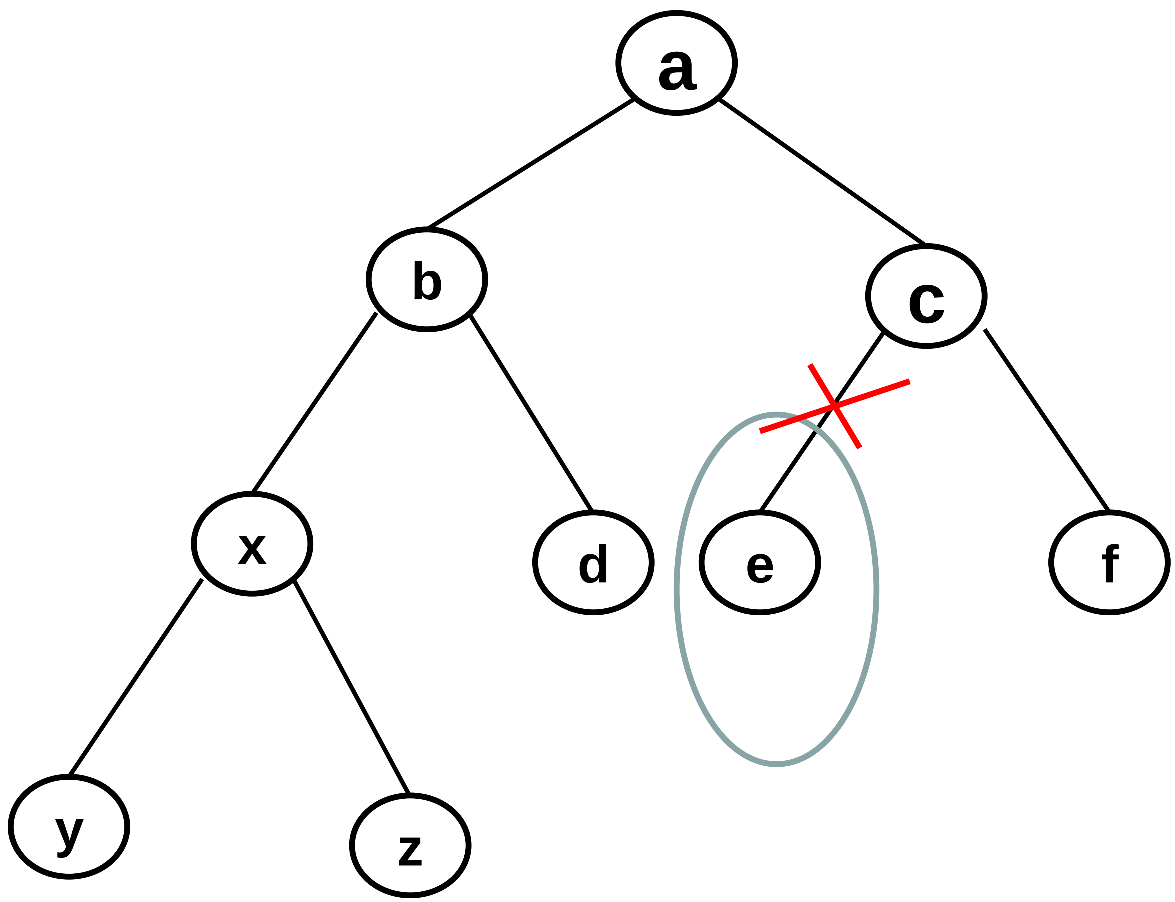
```
a -> esq -> esq = cria('x',
                      cria('y',inicializa(),inicializa()),
                      cria('z',inicializa(),inicializa()));
```

Exemplo de construção



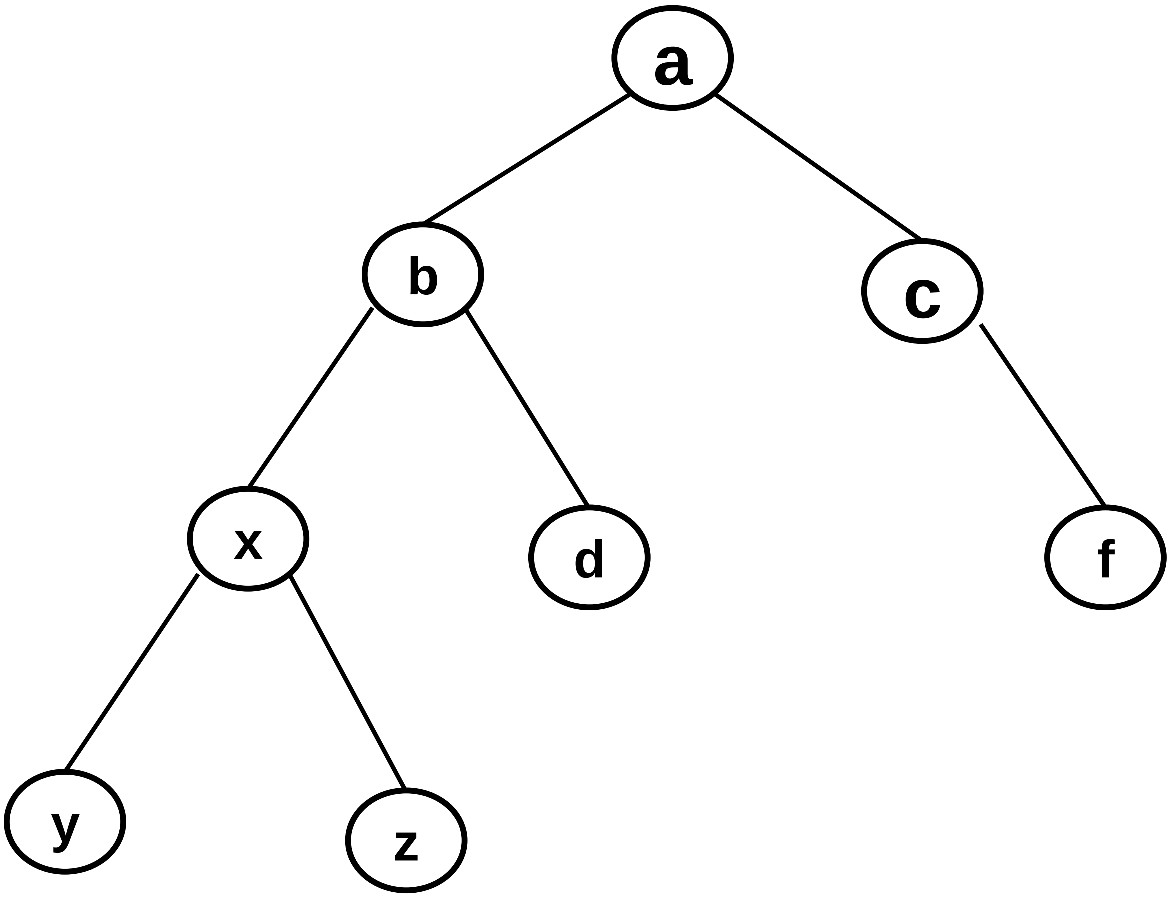
Exemplo de remoção

a -> dir -> esq = desaloca(a -> dir -> esq);



Exemplo de remoção

- Resultando em:



Árvore Binária de Pesquisa

- Impõe uma ordem entre os nós da árvore. Impondo maior rigor na inserção e remoção dos nós e aumentando a rapidez do procedimento de busca.

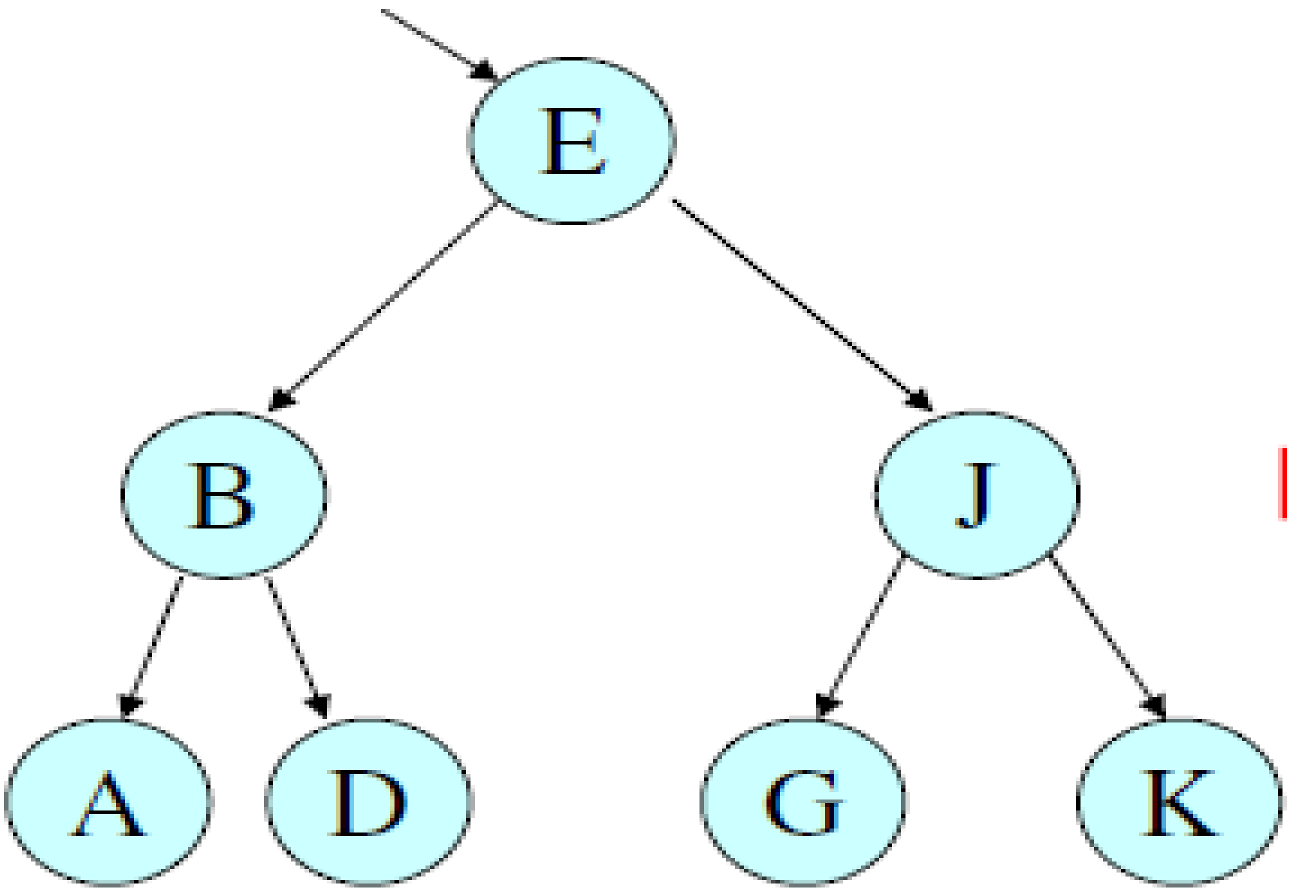
- A busca por um elemento em uma árvore binária de pesquisa tem complexidade no pior caso: $O(\log n)$

Árvore Binária de Pesquisa

- Inserção de nós:

- a) se a árvore for vazia, o nó passa a ser a raiz;
- b) caso contrário, é instalado na subárvore da esquerda, se for menor que o símbolo da raiz, ou da direita se for maior.

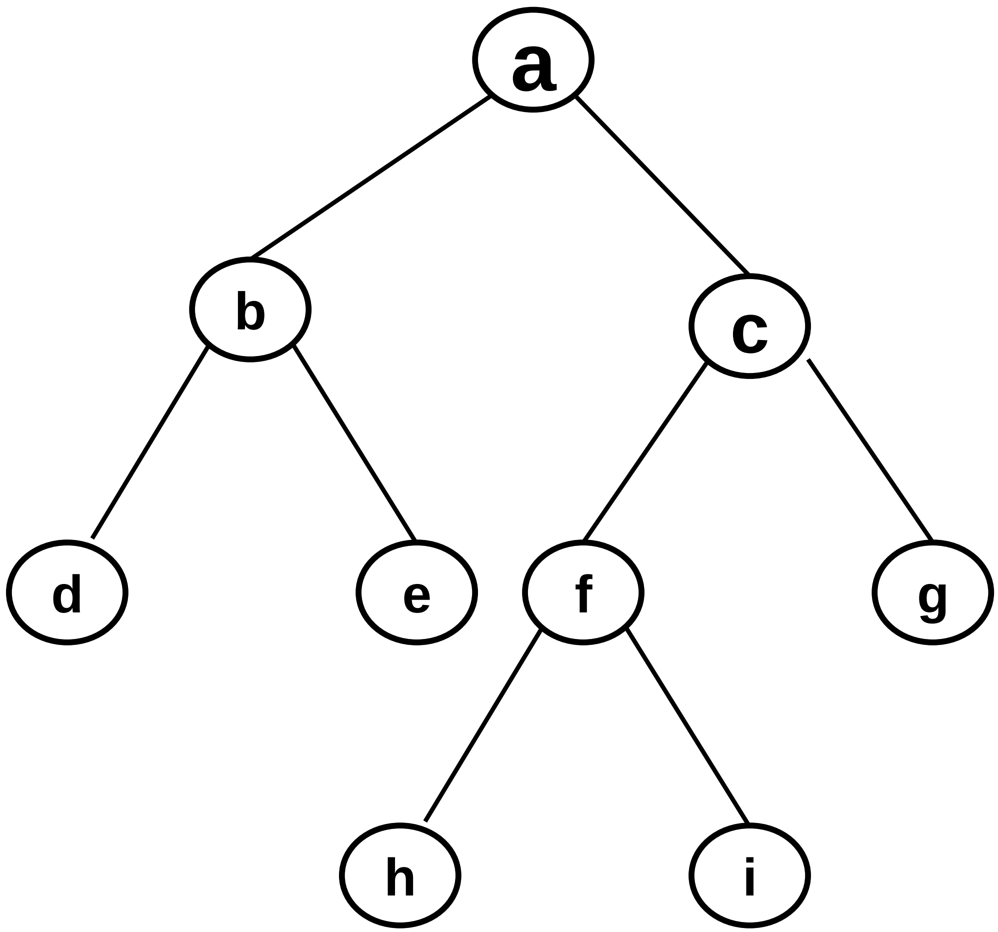
Árvore Binária de Pesquisa



Exercícios

- Para as árvores a seguir faça os 3 tipos de caminhamento

Percurso em Árvores – Exemplo 2



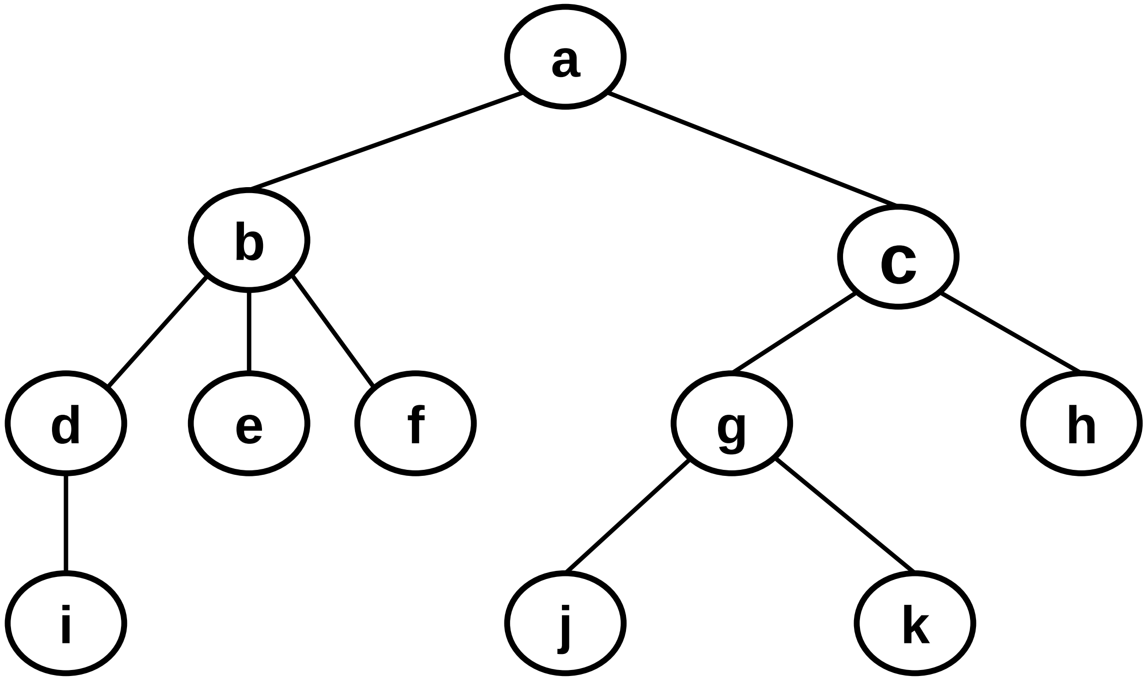
Resultado 2

- Pré-ordem: a,b,d,e,c,f,h,i,g

- Ordem Simétrica: d,b,e,a,h,f,i,c,g

- Pós-ordem: d,e,b,h,i,f,g,c,a

Percurso em Árvores – Exemplo 3



Resultado 3

Pré-ordem: a, b, d, i, e, f, c, g, j, k, h
Ordem Simétrica: i, d, b,e,f,a,j,g,k,c,h
Pós-ordem: i,d,e,f,b,j,k,g,h,c,a

Referências

- Projeto de Algoritmos – Nívio Ziviani
- Estruturas de Dados usando C – Tenenbaum
- Algoritmos: Teoria e Prática – Cormen
- Fundamentos Matemáticos para a Ciência da Computação – Judith Gersting