Estrutura de dados

T3A2

Árvores binárias de busca

Prof. Ismar

Árvores binárias de busca

Em Ciência da computação, uma árvore binária de busca é uma estrutura de dados de árvore binária baseada em nós, onde todos os nós da sub árvore esquerda possuem um valor numérico inferior ao nó raiz e todos os nós da sub árvore direita possuem um valor superior ao nó raiz

As árvores binárias de busca têm como objetivo facilitar a procura por valores.

Operações em árvores binárias de busca

São três as operações típicas realizadas em árvores binárias de busca:

- busca,
- Inserção,
- remoção.

Operações em árvores binárias de busca

Busca

Na busca, caso o valor seja menor que a raiz, a busca deve ocorrer na subárvore à esquerda, até que seja alcançada a folha da árvore. Caso o valor buscado seja maior que a raiz, o mesmo procedimento deve ocorrer na subárvore da direita.

Implementação de busca

```
begin
        while tree <> NULO E tree (campo chave) <> x do
                if x < tree (campo chave) then tree <- esquerda de tree
                else if x > tree (campo chave) then
                        tree <- direita de tree
                else
                        return verdadeiro
                endif
        endw
        return falso
end
```

Operações em árvores binárias de busca

Inserção

O procedimento de inserção em arvores binárias inicia com uma busca pelo valor na árvore, caso o elemento não exista na árvore é alcançada a folha e, então, o valor é inserido nesta posição. Assim, a raiz é examinada e é introduzido um novo nó na sub-árvore da esquerda, caso o valor novo seja menor do que a raiz, ou a direita, caso o valor novo seja maior.

Implementação de inserção

```
Input: Arvore tree, elemento x

begin
        no <- NULO
        atual <- raiz de tree
        while atual <> NULO do
                no <- atual
                if x <atual then
                        atual <- esquerda de tree
                else
                        atual <- direita de tree
                endif
        endw
        Árvore vazia, primeiro elemento é a raiz
        if no = NULO then
                raiz de tree <- x
        else
                if x < no (campo chave) then
                        esquerda de no <- x
                else
                        direita de no <- x
                endif
        endif
end
```

Operações em árvores de binárias busca

Remoção

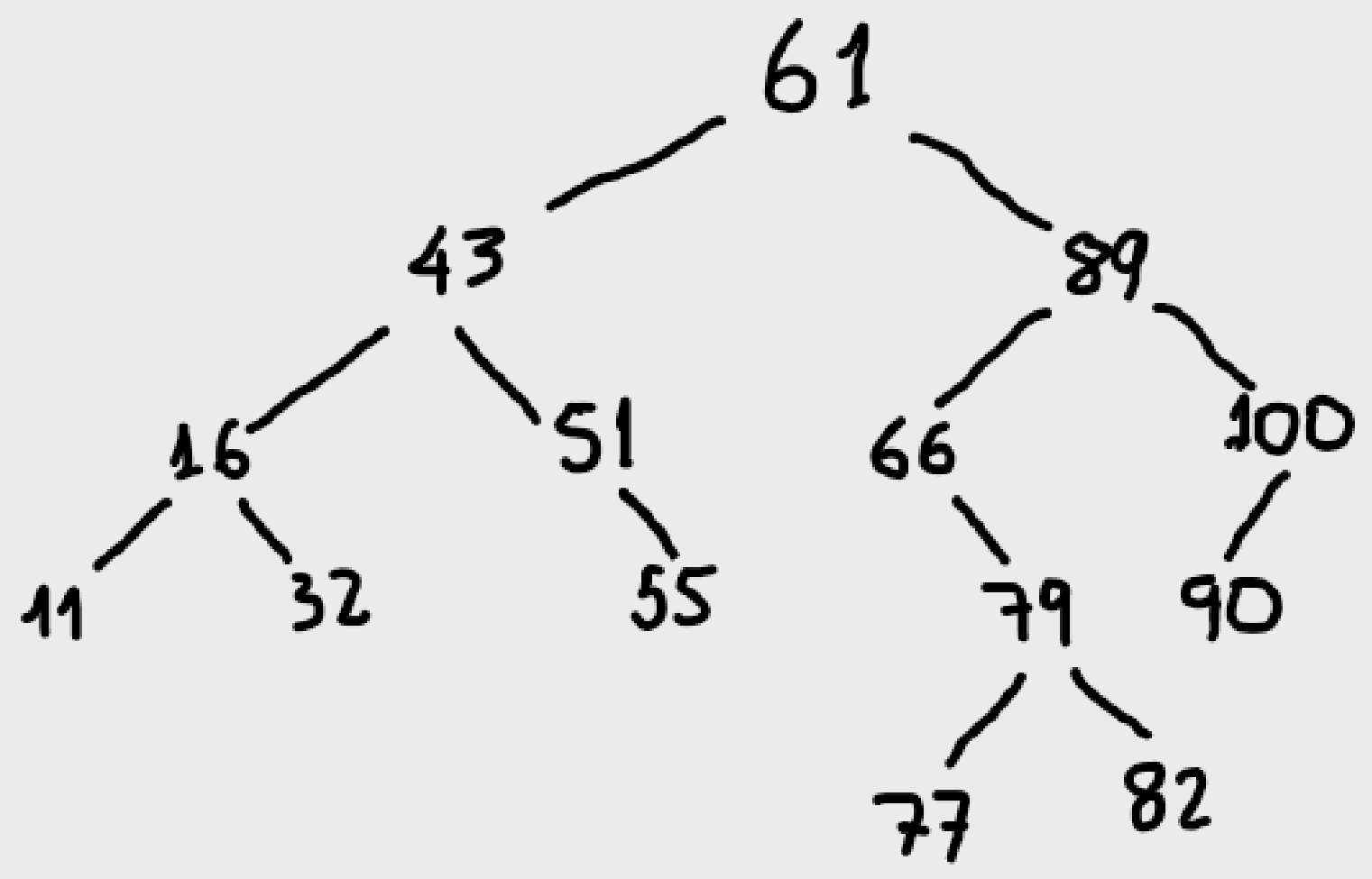
No caso do algoritmo de exclusão, tem algumas particularidades que o tornam mais complexo. Existem três situações que devem ser consideradas:

- Exclusão de folha;
- Exclusão de nó com um filho;
- Exclusão de nó com dois filhos.

Operações em árvores binárias de busca

Remoção de folha

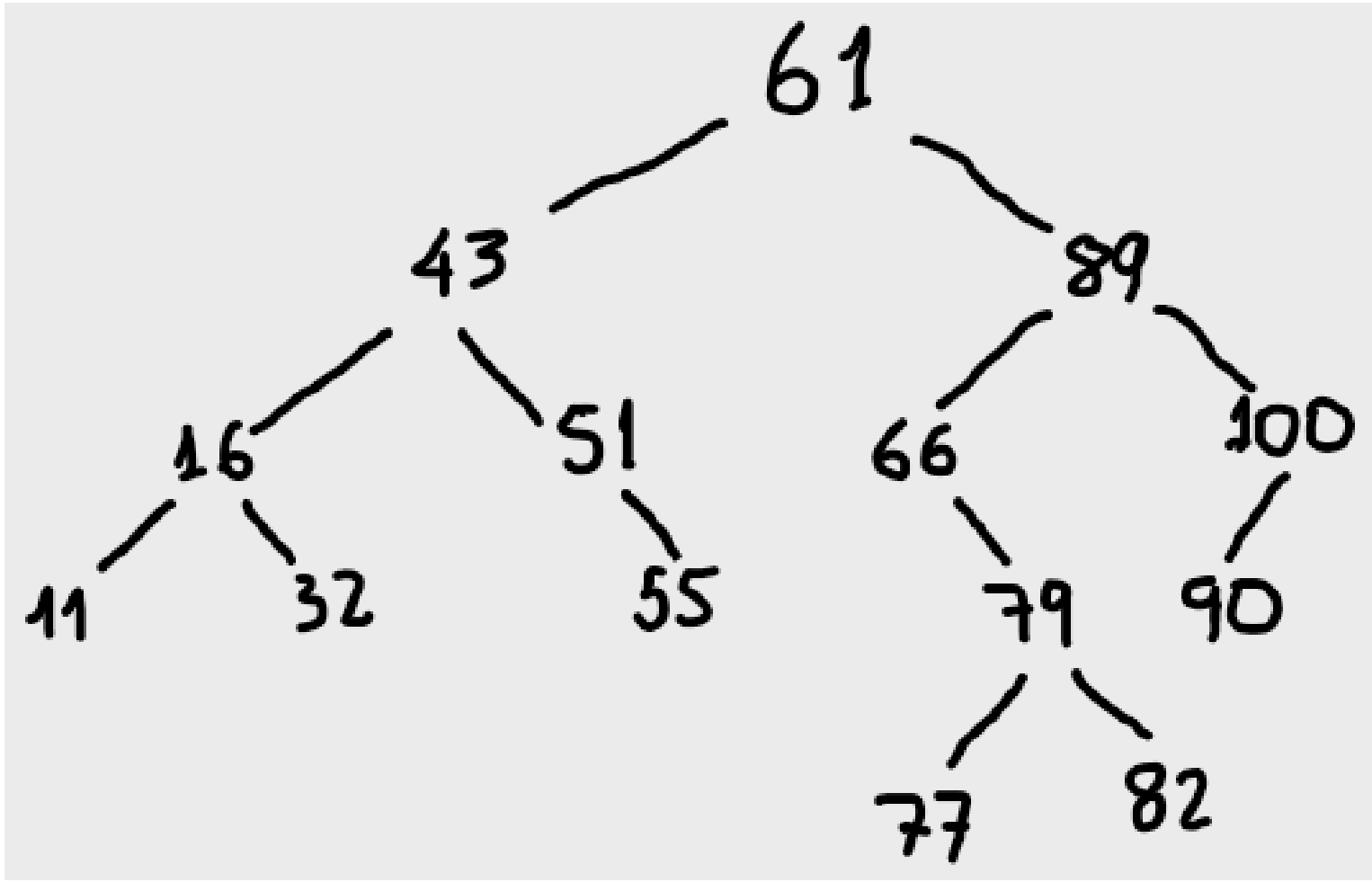
Executar o algoritmo de busca e, ao encontrar a folha, basta removê-la.



Operações em árvores binárias de busca

Remoção de nó com 1 filho

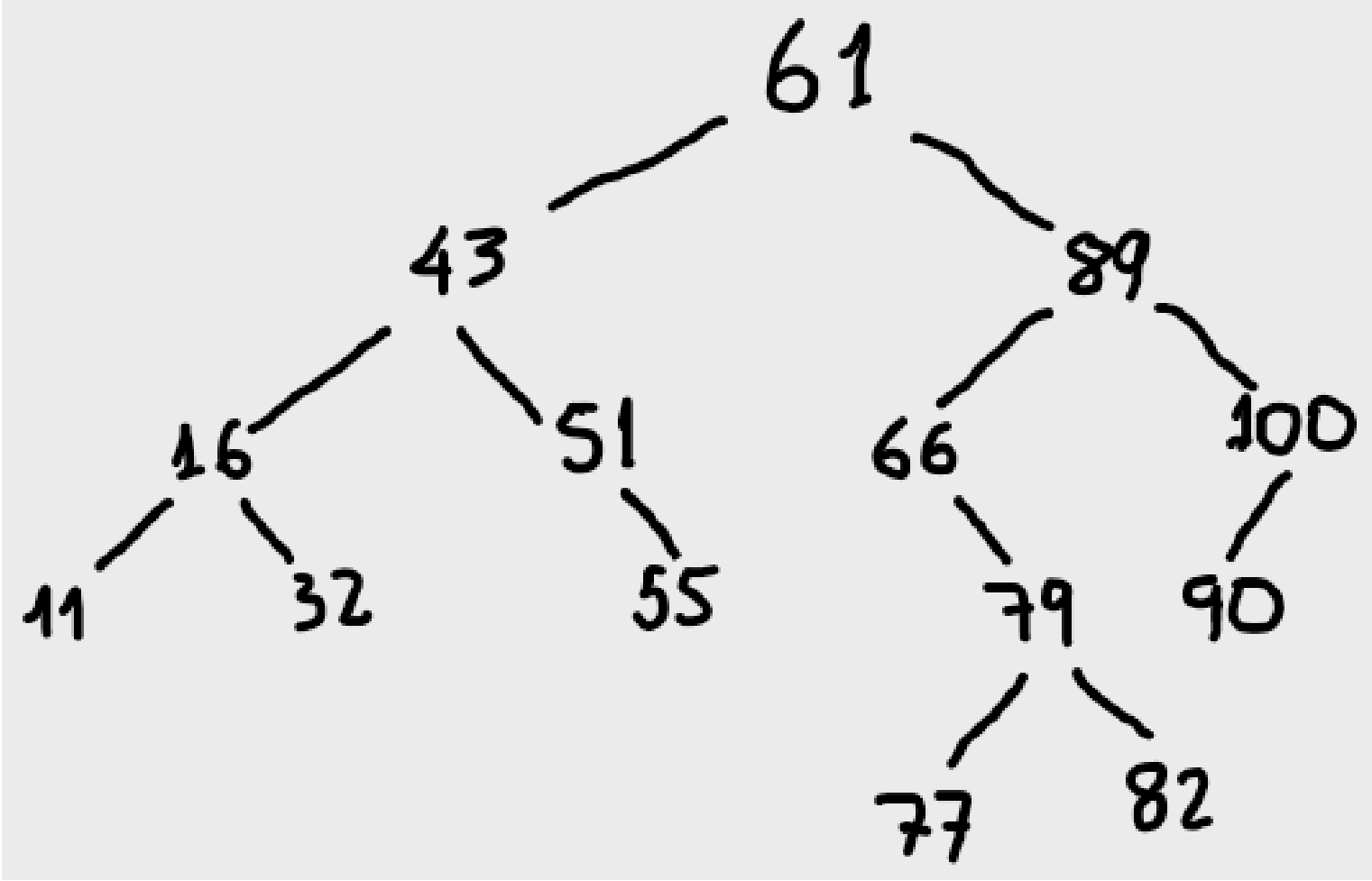
Se o nó a ser excluído possuir apenas 1 filho, o filho assume a posição do pai.



Operações em árvores binárias de busca

Remoção de nó com 2 filhos

Se o nó a ser excluído possuir 2 filhos, o nó deve ser substituído pelo valor do seu sucessor (o menor dentre os maiores).



Exercícios

Os seguintes nós são inseridos nesta ordem em uma árvore binária de busca:

50 30 70 20 40 60 80 15 25 35 45 36

Desenhe a árvore que resulta.

Percorrendo árvores binárias de busca

Percorrer uma árvore binária também é um procedimento comum e consiste em visitar todos os nós existentes na árvore a partir de algum critério. Essa operação também pode ser chamada de travessia da árvore. Existem três possibilidades de travessia:

- pré-ordem;
- pós-ordem;
- in-ordem.

Pré-ordem

Percorrer uma árvore usando o critério pré-ordem, também conhecido como profundidade, significa que o percurso segue os nós até chegar às folhas do lado esquerdo e depois repete o mesmo critério para o lado direito.

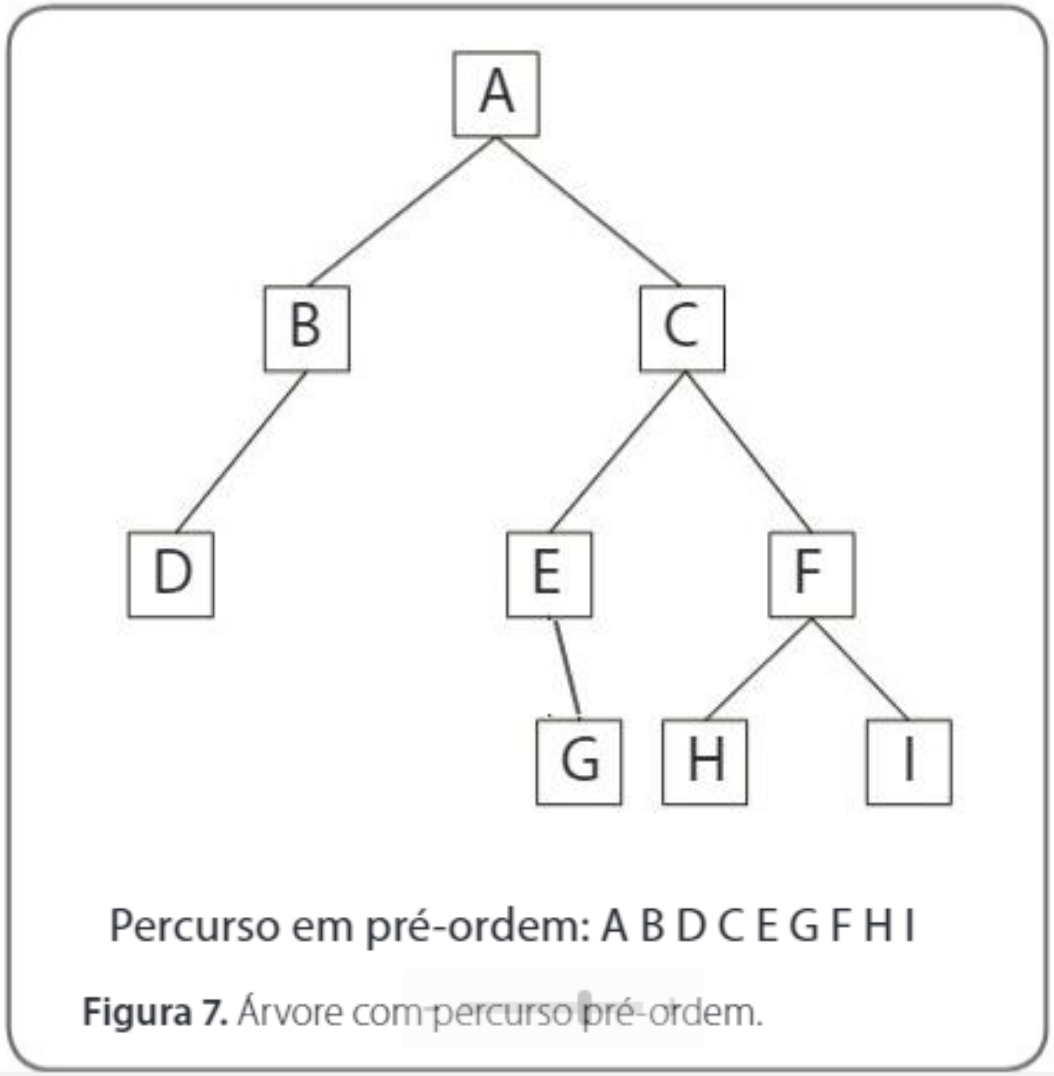


Figura 7. Árvore com percurso pré-ordem.

Pós-ordem

No critério pós-ordem, os filhos são processados antes do nó.

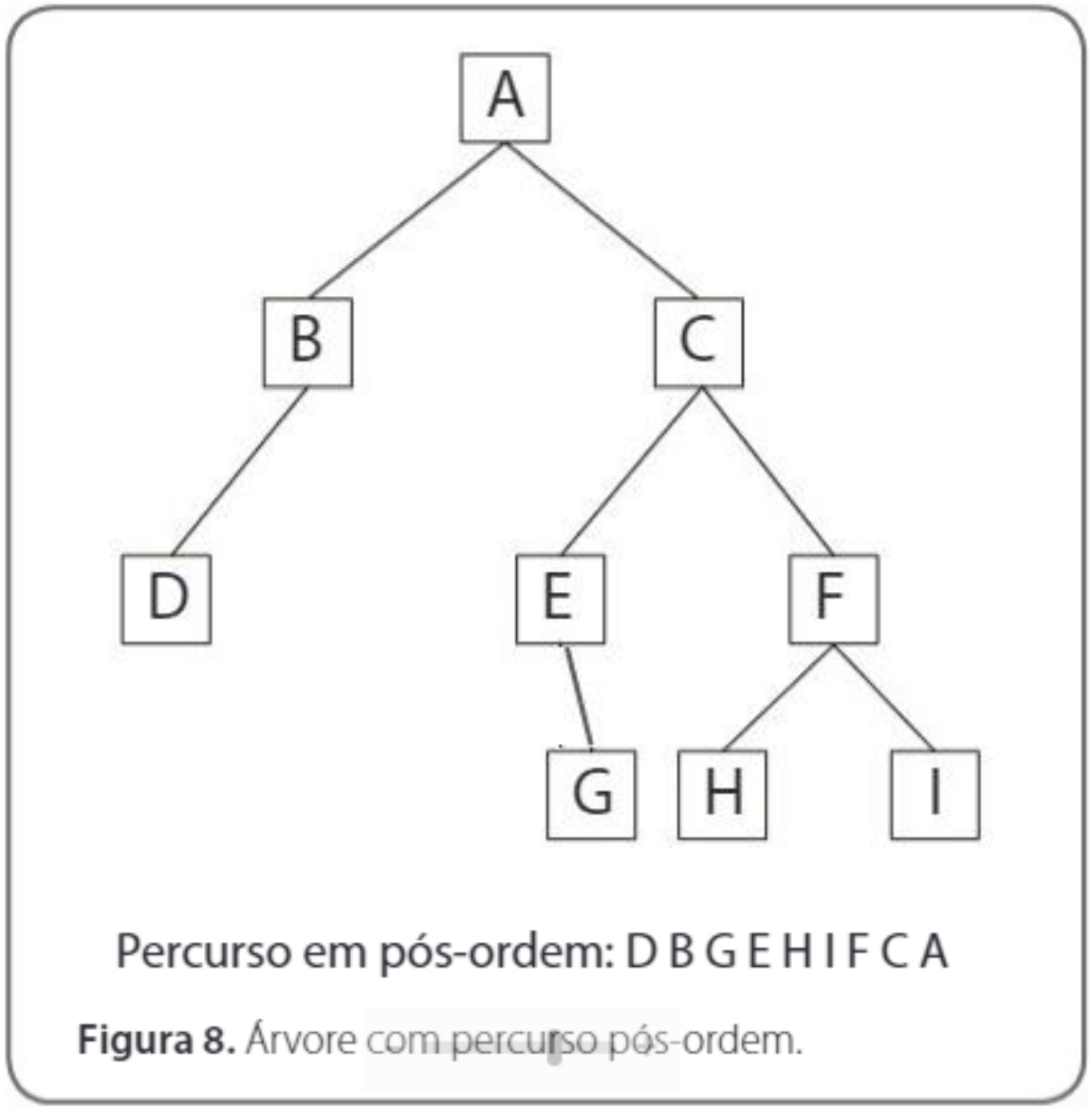


Figura 8. Árvore com percurso pós-ordem.

In-ordem

No critério in-ordem, também conhecido como travessia simétrica, tem a seguinte ordem de processamento: primeiro o filho à esquerda, depois o nó e depois o filho à direita.

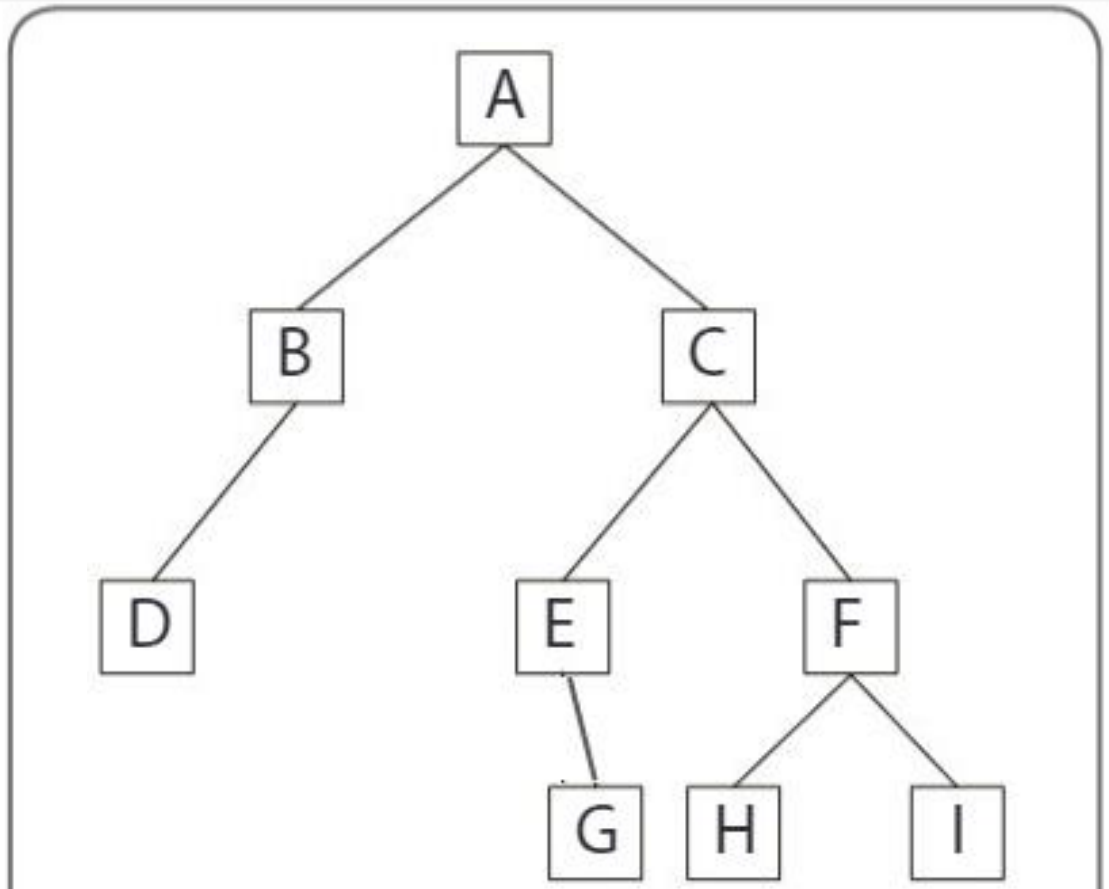


Figura 9. Árvore com percurso in-ordem.


Obrigado pela atenção

T3A3

Estrutura de dados (TEMPORADA III)

Episódio: Construções de árvores binárias

Prof. Ismar

Uma árvore binária é um conjunto de nós, tal que:

- Existe um nó r, denominado raiz, com zero ou mais subárvores;
- Os nós raízes dessas subárvores são filhos de r;
- Os nós internos da árvore são nós com filhos;
- As folhas ou os nós externos da árvore são os nós sem filhos.

Uma árvore binária é:

- Uma árvore vazia;
- Ou um nó raiz com duas subárvores: a da direita e a da esquerda.

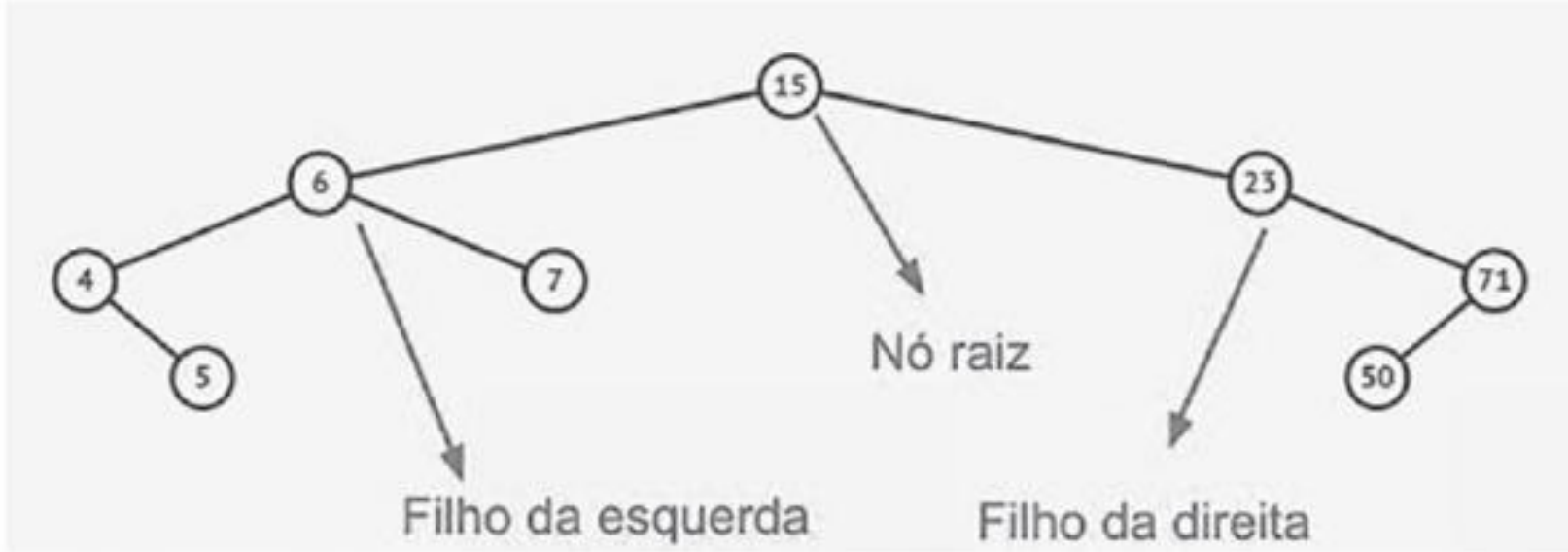


Figura 1. Representação de uma árvore binária.

Estrutura básica

O nó é a estrutura básica para a implementação da árvore.

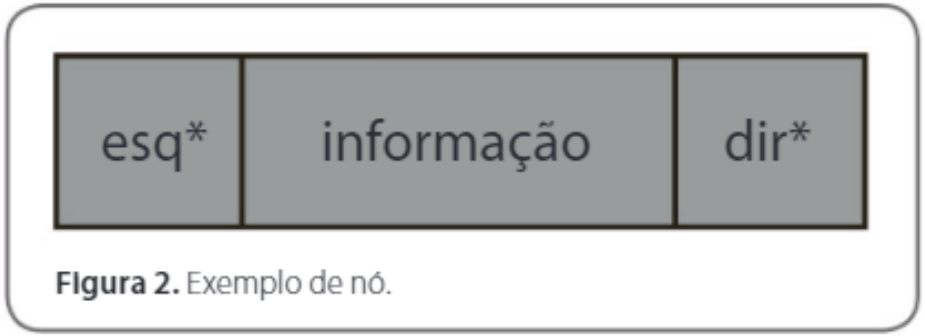


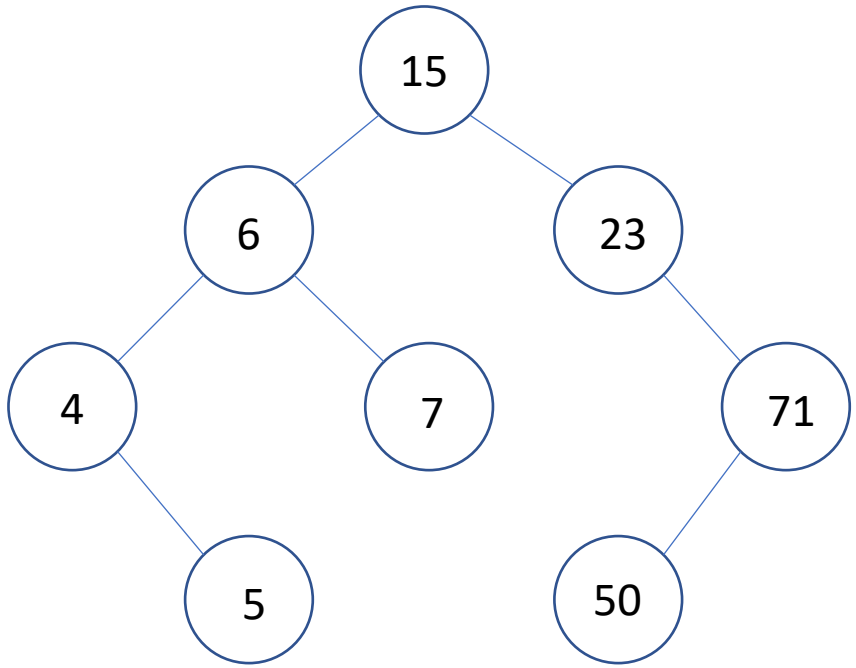
FIgura 2. Exemplo de nó.

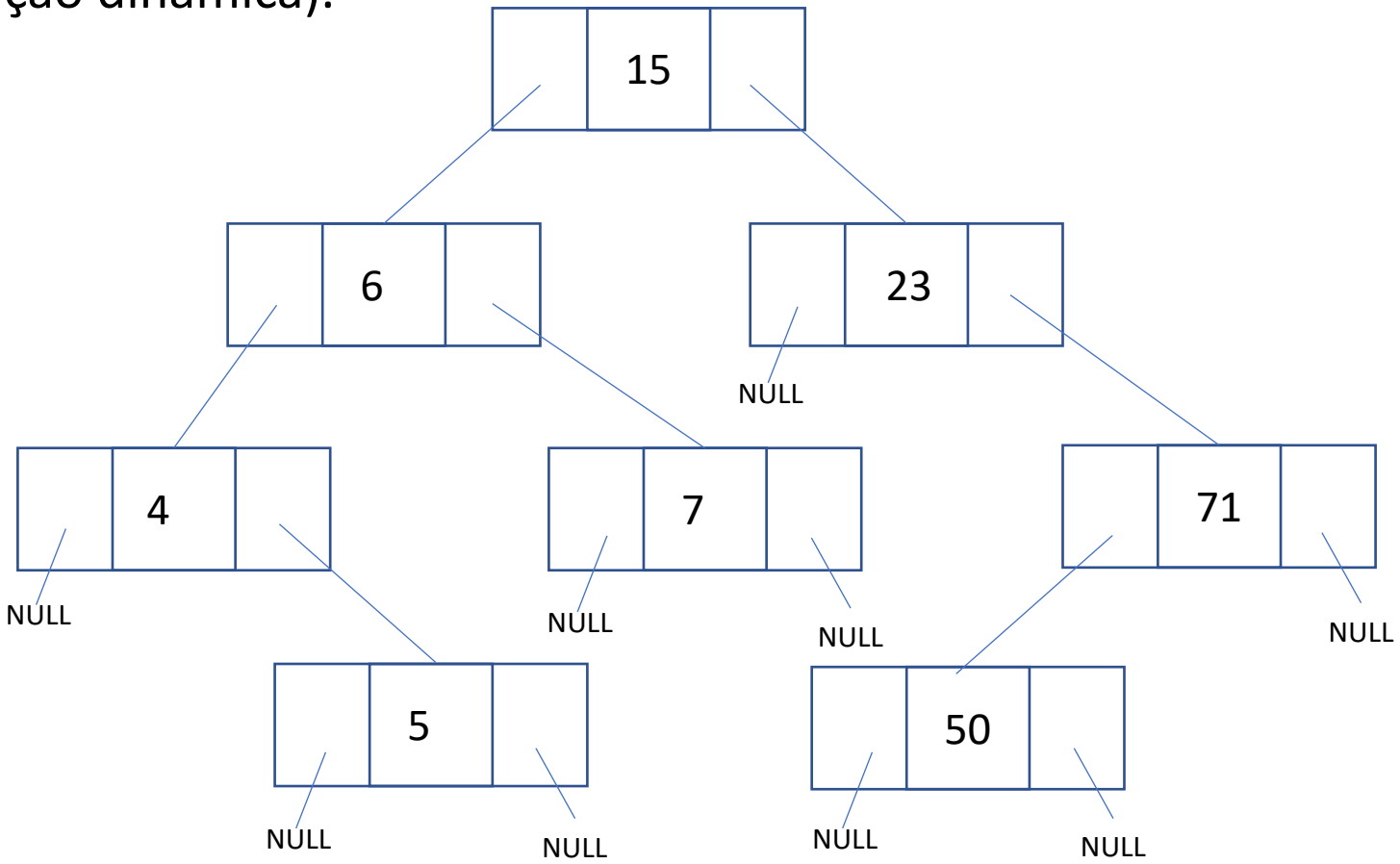
Um ponteiro apontando para o filho da esquerda

Um ponteiro apontando para o filho da direita

Implementação usando lista encadeada

Pode-se implementar uma árvore binária usando uma lista encadeada (alocação dinâmica).





Construção de árvores binárias

Implementação usando lista encadeada

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

//defini a estrutura
struct no{
    int info;
    struct no *esq,*dir;
};

typedef struct no* arvBin;

arvBin* criaArvBin();

int main(){
    arvBin* raiz;
    raiz=criaArvBin();
    return(0);
}


arvBin* criaArvBin(){
    arvBin* raiz = (arvBin*) malloc(sizeof(arvBin));
    if(raiz != NULL)
        *raiz = NULL;
    return raiz;
}
```

```
59  int insereArvBin(arvBin *raiz, int valor){
60      // se ocorreu algum erro na criacao da arvore retorna 0, indicando erro
61       if(raiz == NULL)
62          return 0;
63      //cria um novo no
64      struct no* novo;
65      //aloca memoria de forma dinamica
66      novo = (struct no*) malloc(sizeof(struct no));
67      //se ocorreu algum erro na alocaao de memoria do novo no, retorna 0 indicando erro
68      if(novo == NULL)
69          return 0;
70      //guarda a informacao no novo no e faz seus ponteiros apontarem para null
71      novo->info = valor;
72      novo->dir = NULL;
73      novo->esq = NULL;
74
```

```
75      //agora vamos encontrar onde vamos inserir o no
76      //se o nosso no especial estiver apontando para NULL, significa q temos uma arvore vazia
77      //o nosso unico no eh o no q acabou de ser criado
78      if(*raiz == NULL)
79          *raiz = novo;
80      //caso contrario vamos avaliar onde vamos colocar o novo no
81      else{
82          //para isso vamos usar um no auxiliar atual
83          struct no* atual = *raiz;
84          //e outro ant de anterior ao atual
85          struct no* ant = NULL;
86          //enquanto o atuall não for NULL, ou seja não for um nó-folha
87          while(atual != NULL){
88              //anterior vai guardar o end d atual
89              ant = atual;
90              //se o valor q queremos inserir for igual a um existente na arvore,nao inserimos
91              if(valor == atual->info){
92                  free(novo);
93                  return 0;//elemento já existe
94              }
95              //se for maior, vamos percorrer a arvore da direita
96
97              if(valor > atual->info)
98                  atual = atual->dir;
99              //menor a arvore da esquerda
100             else
101                 atual = atual->esq;
102         }
103         //isso tudo eh feito ate acharmos a um no folha
104         //o anterior vai nos guiar se vamos adicionar o novo no a direita ou a esquerda
```

```
105             if(valor > ant->info)
106                 ant->dir = novo;
107             else
108                 ant->esq = novo;
109         }
110         return 1;
111 }
112
```


Obrigado pela atenção

T3A4

Estrutura de dados (TEMPORADA III)

Episódio: Balanceamento de Árvore

Prof. Ismar

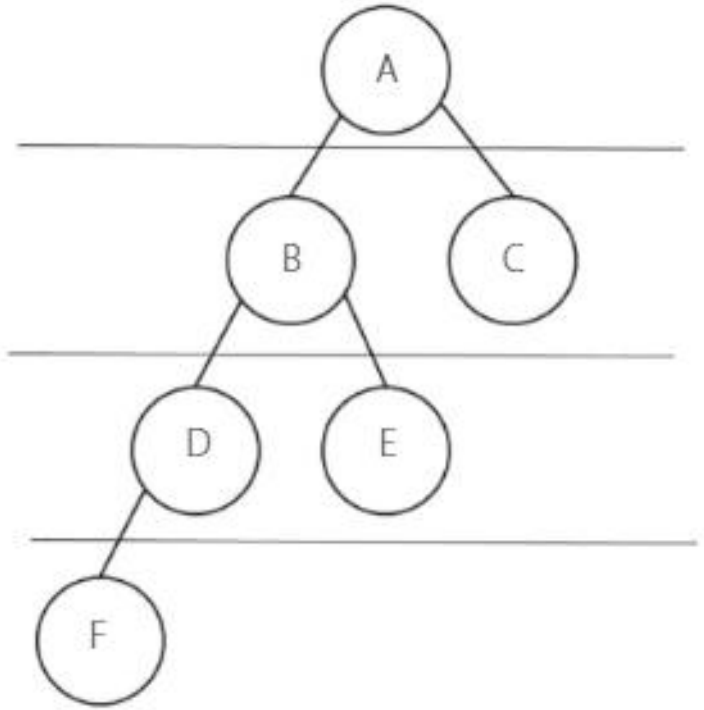
Introdução

Numa árvore, à medida em que se acrescenta nós, aumenta o tempo computacional. A solução para esse problema é a árvore AVL, a árvore balanceada. O nome AVL se deve ao fato de ter sido criada pelos soviéticos Adelson, Velsky e Landis em 1962.

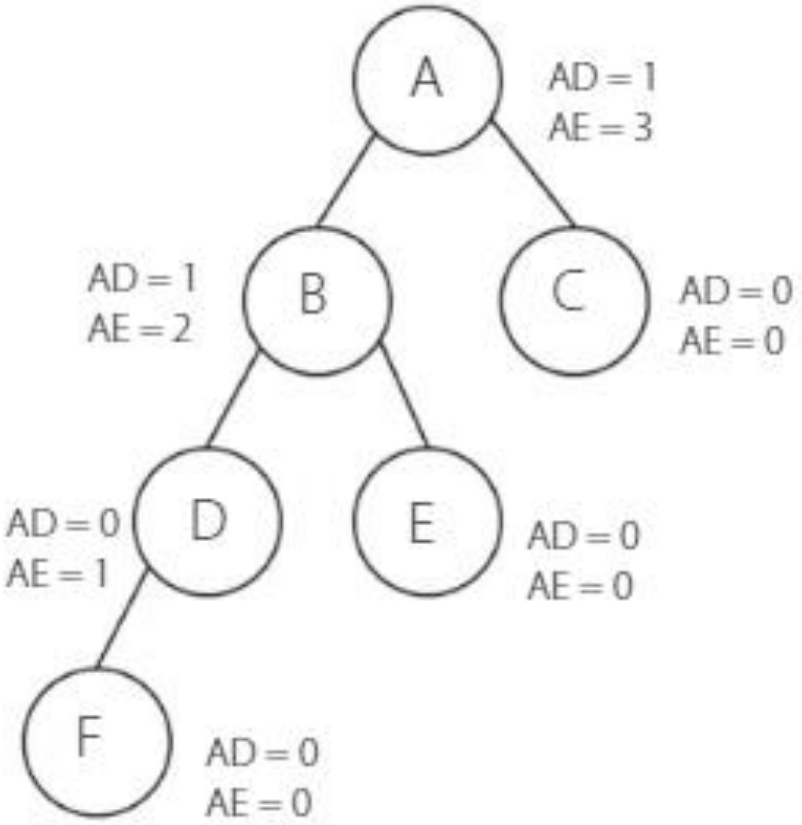
O que indica se uma árvore está ou não balanceada é a altura de suas subárvores à esquerda e à direita.

Balanceamento (árvore AVL)

A propriedade de balanceamento consiste em manter as subárvores esquerda e direita **de cada nó**, na mesma altura, podendo diferir no máximo em 1 nível.



Cálculo do Fator de Balanceamento



FB = Altura Esquerda – Altura Direita

FB = AE – AD

Uma subárvore balanceada tem FB entre -1 e 1

Balanceamento

Uma das vantagens da árvore AVL é o fato de ser possível que se faça o balanceamento local, isto é, envolvendo somente os nós com FB fora do intervalo permitido.

Balanceamento de árvores binárias

(Árvores AVL)

Requisitos:

- Árvore Binária de Busca
- FB = (-1, 0, 1)

Prof. Ismar