Marisangila Alves, MSc
marisangila.alves@catolicasc.org.br
marisangila.com.br

Católica de Santa Catarina

2025/2

Estrutura de Dados

Árvore

Sumário

Definição

Árvore

> *As listas encadeadas, pilhas e filas são estruturas lineares de dados. Uma árvore é uma estrutura de dados não-linear e bidimensional com propriedades especiais.*
>
> (Deitel; Deitel, 2011)

> *Árvore é uma lista na qual cada elemento possui dois ou mais sucessores, porém todos os elementos possuem apenas um antecessor.*
>
> (FORBELLONE; EBERSPÄCHER, 2010)

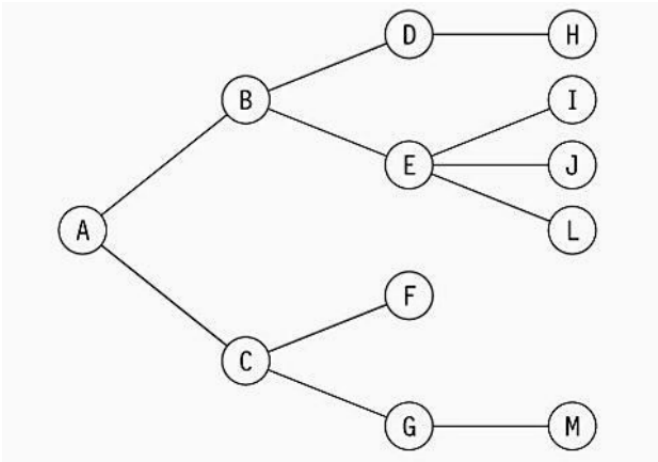
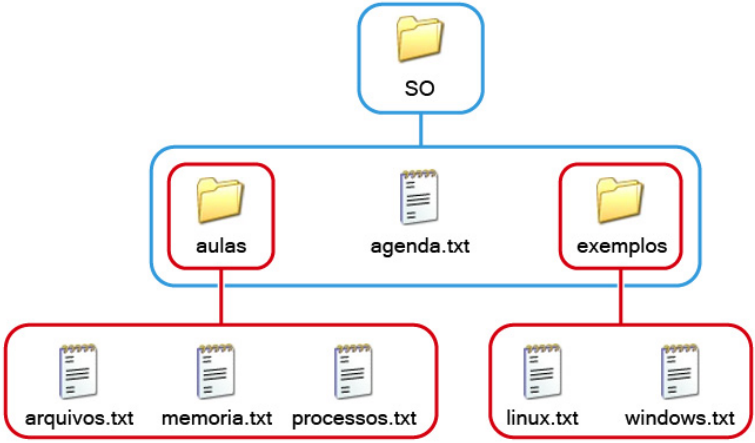


Figura 1: Exemplo de árvore (FORBELLONE; EBERSPÄCHER, 2010).

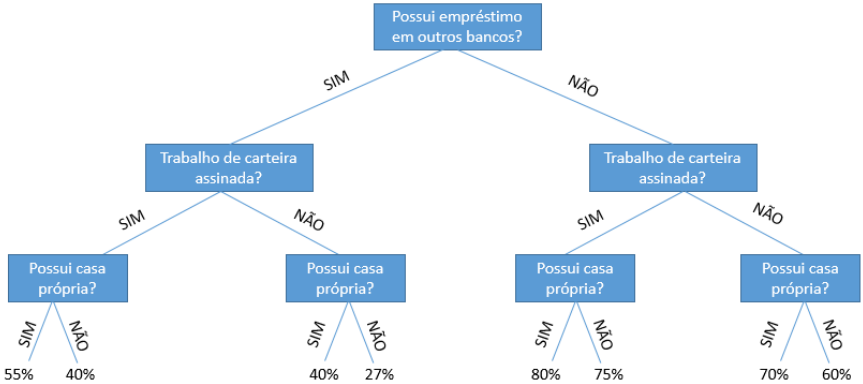
Nota:

Árvores em essência são de natureza hierarquica e recursiva!

- **Árvore de diretórios;**
- **Árvore de análise sintática de compiladores;**
- **Árvore de busca;**
- **Árvore de ordenação;**
- **Árvore de decisão;**
- **Árvore de busca de índices em banco de dados.**

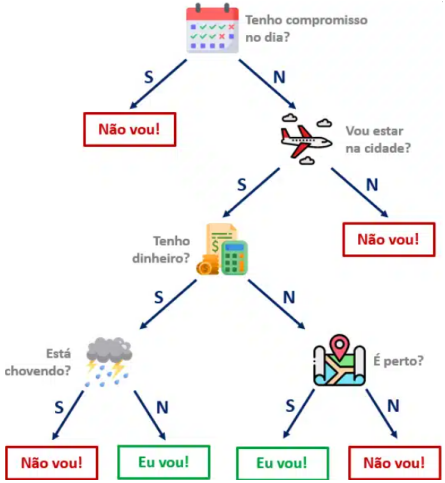

SO
aulas
agenda.txt
exemplos
arquivos.txt
memoria.txt
processos.txt
linux.txt
windows.txt

Exemplos III



Exemplo de análise de risco de crédito.

Exemplos IV


Tenho compromisso no dia?
S
N
Não vou!
Vou estar na cidade?
S
N
Não vou!
Tenho dinheiro?
S
N
Está chovendo?
É perto?
S
N
S
N
Não vou!
Eu vou!
Eu vou!
Não vou!

Terminologia

Árvores são estruturas de dados adequadas para a representação de hierarquias.

1. Um árvore pode ser definida usando recursividade;
2. Uma árvore é composta por um conjunto de nós;
3. Existe um nó `r`, denominado raiz, que contém zero ou mais subárvores, cujas (sub)raízes são ligadas diretamente a `r`;
4. Esses nós raízes das subárvores são ditos filhos do nó pai, `r`;
5. Nós com filhos são comumente chamados de nós internos e nós que não têm filhos são chamados de folhas ou nós externos;
6. **Altura:** A quantidade de níveis a partir do nó raiz até o nó mais distante é chamado de altura;
7. **Grau:** O número máximo de ramificações a partir de um nó é chamado de grau.
8. **Ordem:**

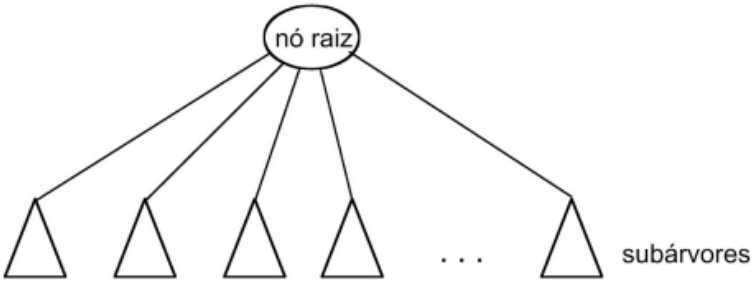
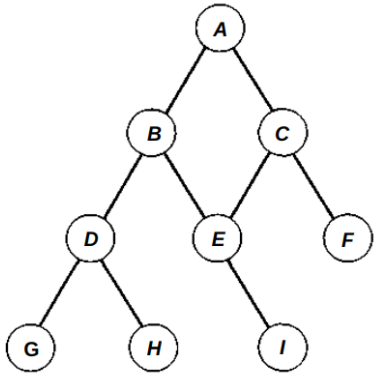


Figura 2: Exemplo de árvore (CELES; CERQUEIRA; RANGEL, 2004).

Nota:

Embora as árvores naturais cresçam com suas raízes fincadas na terra e suas folhas no ar, os cientistas de computação retratam quase universalmente as estruturas de dados em árvore com a raiz no topo e as folhas no chão (Deitel; Deitel, 2011).



Quando você percorre uma árvore a partir das folhas na direção da raiz, diz-se que você está "subindo"a árvore, e se partir da raiz para as folhas, você está "descendo"a árvore.

Classificação

Principais Tipos de Árvores I

- **Árvore Genérica (n-ária)**: Cada nó pode ter qualquer número de filhos. Usada, por exemplo, na representação de sistemas de arquivos (pastas e subpastas).
- **Árvore Binária**: Cada nó possui no máximo dois filhos (esquerda e direita).
- **Árvore Busca**: Organiza os nós de forma ordenada — à esquerda ficam os menores, à direita os maiores. Ideal para buscas, inserções e remoções rápidas.

Principais Tipos de Árvores II

- **Árvore AVL**: Árvore balanceada. Usa rotações para manter o equilíbrio após inserções e remoções.
- **Árvore B**: Árvores multi-filhas utilizadas em bancos de dados para busca eficiente em disco.
- **Árvore Heap**: Árvore usada em filas de prioridade. No Max-Heap, o maior valor está na raiz; no Min-Heap, o menor.
- **Árvore de Decisão**: Usada em *Machine Learning* para representar decisões com base em condições.

Representação

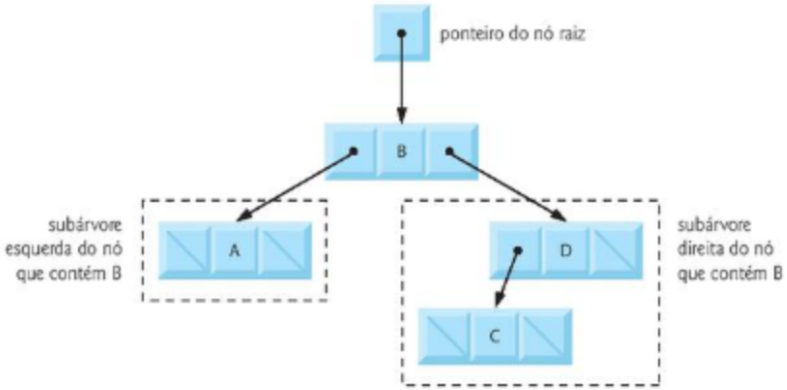


Figura 3: Lista encadeada representando uma árvore (Deitel; Deitel, 2011)

Árvore Binária

Definição I

> *Se todo nó que não é folha numa árvore binária tiver subárvores esquerda e direita não-vazias, a árvore será considerada uma árvore estritamente binária (Tenenbaum; Augenstein; Langsam, 1995).*
>
> (Tenenbaum; Augenstein; Langsam, 1995)

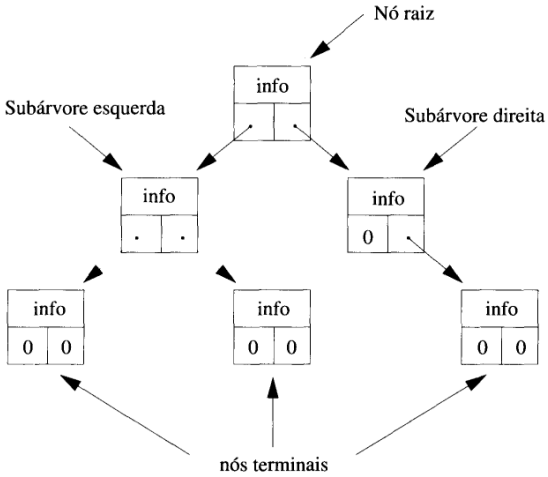
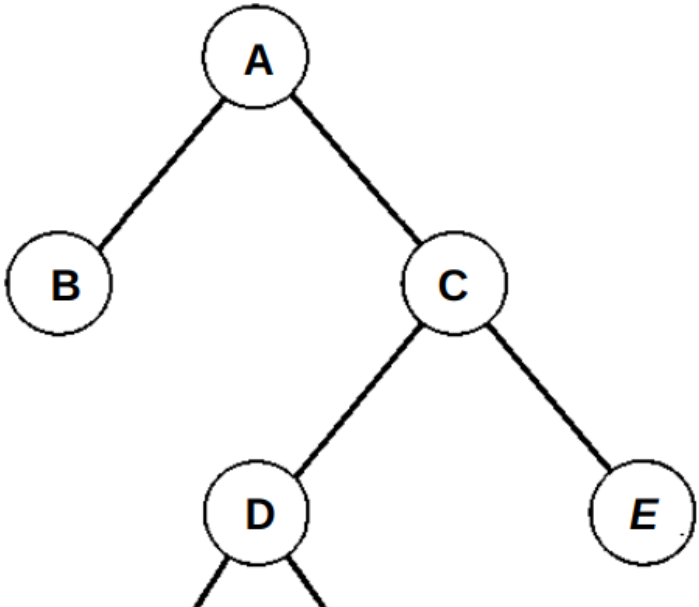


Figura 4: Estrutura de dados (Schildt, 1996)

Definição III



Árvore Binária de Busca

Definição I

- **Definição:** uma **árvore binária de busca (ABB)** é uma árvore binária na qual, para cada nó n:
  - A subárvore à **esquerda** contém apenas nós com valores **menores** que o valor em n;
  - A subárvore à **direita** contém apenas nós com valores **maiores** que o valor em n;
  - Ambas as subárvores (esquerda e direita) também devem ser **árvores binárias de busca**.
- Todos os nós de uma ABB contêm **valores distintos**.

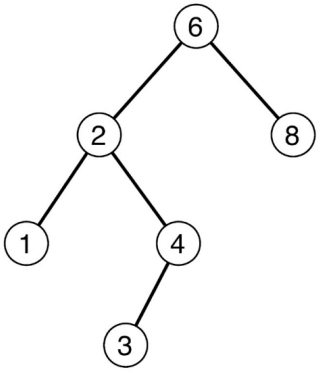


Figura 7: Árvore binária de busca (CELES; CERQUEIRA; RANGEL, 2004)

- Inserção;
- Busca de um valor;
- Remoção;
- Percurso (varredura nos nós).

Implementação

```c
typedef struct  NoAbb
{
    int dado;
    struct NoAbb *esquerda, *direita;
}NoAbb;
```

Código 2: Estrutura da arvore.

```c
NoAbb* inserir(int dado, NoAbb *no);
NoAbb* buscar(int dado, NoAbb *no);
NoAbb* remover(int dado, NoAbb *no);
NoAbb* alocar_no(int dado);
void pre_ordem(NoAbb *no);
void em_ordem(NoAbb *no);
void pos_ordem(NoAbb *no);
```

Código 3: Exemplo de protótipos das funções.

```c
NoAbb *inserir(int dado, NoAbb *no)
{
    if (no == NULL)
        return alocar_no(dado);
    if (dado < no->dado)
        no->esquerda = inserir(dado, no->esquerda);
    else
        no->direita = inserir(dado, no->direita);
    return no;
}
```

Código 4: Exemplo de inserção.

```c
NoAbb *buscar(int chave, NoAbb *no)
{
    if (no == NULL)
        return NULL;
    if (chave < no->dado)
        return buscar(chave, no->esquerda);
    if (chave > no->dado)
        return buscar(chave, no->direita);
    return no;
}
```

Código 5: Exemplo de busca.

```
1
```

Código 6: Exemplo de remoção.

```c
#include "abb.h"
#include <stdlib.h>
#include <stdio.h>
int main()
{
    NoAbb *raiz = NULL;
    raiz = inserir(10, raiz);
    raiz = inserir(30,raiz);
    raiz = inserir(50, raiz);
    raiz = inserir(20, raiz);
    raiz = inserir(5, raiz);
    printf("\n Pre-ordem!\n");
    pre_ordem(raiz);
    printf("\n Em-ordem!\n");
    em_ordem(raiz);
    printf("\n Pos-ordem!\n");
    pos_ordem(raiz);
    return 0;
}
```

Código 7: Exemplo de utilização da árvore.

- Uma árvore é uma estrutura **não sequencial**, diferentemente de uma lista.
- Não existe uma **ordem natural** para percorrer árvores, portanto, podem ser escolhidas diferentes estratégias de percurso;
- Existem duas formas gerais de **percorrer uma árvore**:
  - **Em profundidade (depth-first)**:
    - Pré-ordem;
    - Em-ordem;
    - Pós-ordem;
  - **Em largura (breadth-first)**:
    - Percorre-se **cada nível de cada vez**, da esquerda para a direita.

- Todos os métodos de percurso podem ser definidos de forma **recursiva** e se baseiam em três operações básicas:
  - Visitar o nó;
  - Percorrer a subárvore da esquerda;
  - Percorrer a subárvore da direita.

A única diferença entre estes métodos é a ordem em que estas operações são executadas.

> **Nota:**
>
> Os algoritmos funcionam da mesma forma para árvores binárias em geral (mesmo com os valores fora de ordem);

```c
void pre_ordem(NoAbb *no)
{
    if (no != NULL)
    {
        printf(" [ %d ] ", no->dado);
        pre_ordem(no->esquerda);
        pre_ordem(no->direita);
    }
}
```

Código 8: Exemplo de pré-ordem.

```c
void em_ordem(NoAbb *no)
{
    if (no != NULL)
    {
        em_ordem(no->esquerda);
        printf(" [ %d ] ", no->dado);
        em_ordem(no->direita);
    }
}
```

Código 9: Exemplo de em-ordem.

```c
void pos_ordem(NoAbb *no)
{
    if (no != NULL)
    {
        pos_ordem(no->esquerda);
        pos_ordem(no->direita);
        printf(" [ %d ] ", no->dado);
    }
}
```

Código 10: Exemplo de pós-ordem.

Nota:

- A manipulação de uma ABB pode levar ao seu desbalanceamento!
- O caso extremo é quando a árvore se torna degenerada (equivalente a uma lista encadeada);
- Neste caso, o desempenho dos algoritmos cai, tendendo à ordem linear $\rightarrow$ O(n);
- Para lidar com este problema, existem algumas implementações de ABBs autobalanceáveis;
- Organizam sua estrutura a cada manipulação;
- Garantem que a altura da ABB seja sempre O(log n);

Árvore AVL

- Esse tipo de árvore tem o nome formado pelas iniciais de seus inventores (1962): G. M. Adelson-Velskii e E. M. Landis;
- Uma árvore AVL é dita autobalanceável, pois realiza operações de balanceamento (rotações) após cada manipulação, se necessário;
- Para tal, utiliza uma métrica chamada Fator de Balanceamento.

- Dada a função A que calcula a **altura** de uma árvore;
- O **Fator de Balanceamento (FB)** de um nó é definido da seguinte forma:

$$\text{FB} = A(\text{sub-árvore esquerda}) - A(\text{sub-árvore direita})$$

- Uma **árvore AVL** é uma **árvore binária de busca (ABB)** que está balanceada, ou seja, todos os seus nós possuem fator de balanceamento igual a -1, 0 ou $+1$;
- Uma inserção ou remoção pode tornar a árvore desbalanceada, fazendo com que um ou mais nós tenham fator de balanceamento igual a -2 ou $+2$;
- Nestes casos, são realizadas operações de **rotações** para restaurar o equilíbrio da árvore.

Rotação direita simples

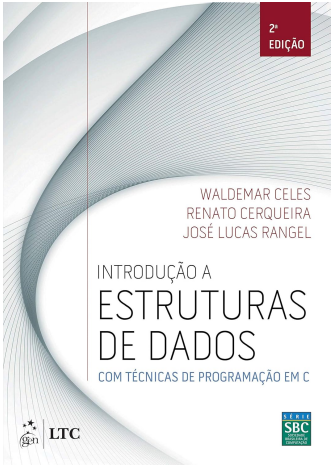
Rotação direita

Rotação esquerda simples

Rotação esquerda

Leitura Recomendada

(Deitel; Deitel, 2011) - Capítulo 16



CELES, Waldemar; CERQUEIRA, Renato; RANGEL, José Lucas. **Introdução a estruturas de dados: com técnicas de programação em C**. Rio de Janeiro: Elsevier, 2004.

DEITEL, Paul; DEITEL, Harvey. **C: Como Programar**. 6. ed. São Paulo: Pearson Universidades, 2011.

FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. **Lógica de programação: a construção de algoritmos e estrutura de dados**. 3. ed. São Paulo: Pearson Prentice Hall, 2010.

PUGA, Sandra Gavioli; RISSETTI, Gerson. **Lógica de programação e estrutura de dados: com aplicações em Java**. 2. ed. São Paulo, SP: Pearson, 2009. E-book. Disponível em: `https://plataforma.bvirtual.com.br`. Acesso em: 24 jul. 2025.

SCHILDT, Herbert. **C Completo e Total: O Guia Definitivo para Programação em C**. 3. ed. São Paulo: Makron Books, 1996. ISBN 978-8534606928.

TENENBAUM, Aaron M.; AUGENSTEIN, Moshe J.; LANGSAM, Yedidyah. **Estruturas de Dados Usando C**. 1. ed. São Paulo: Pearson Universidades, jun. 1995. ISBN 9788521201947.

Estes slides estão protegidos por uma licença Creative Commons

Este modelo foi adaptado de Maxime Chupin.

Marisangila Alves, MSc
marisangila.alves@catolicasc.org.br
marisangila.com.br

Católica de Santa Catarina

2025/2

Estrutura de Dados

Árvore