ESTRUTURA DE DADOS I - (3DAD103)

Árvore Binária

Alexandre Mendonça Fava

08390615959@edu.udesc.br

Universidade do Estado de Santa Catarina – UDESC

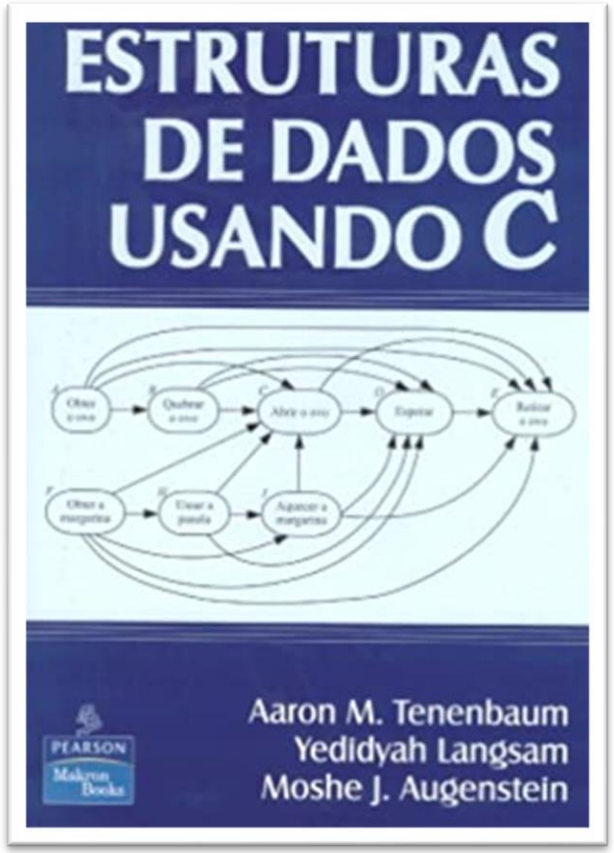
Programa de Pós-graduação em Computação Aplicada - PPGCA

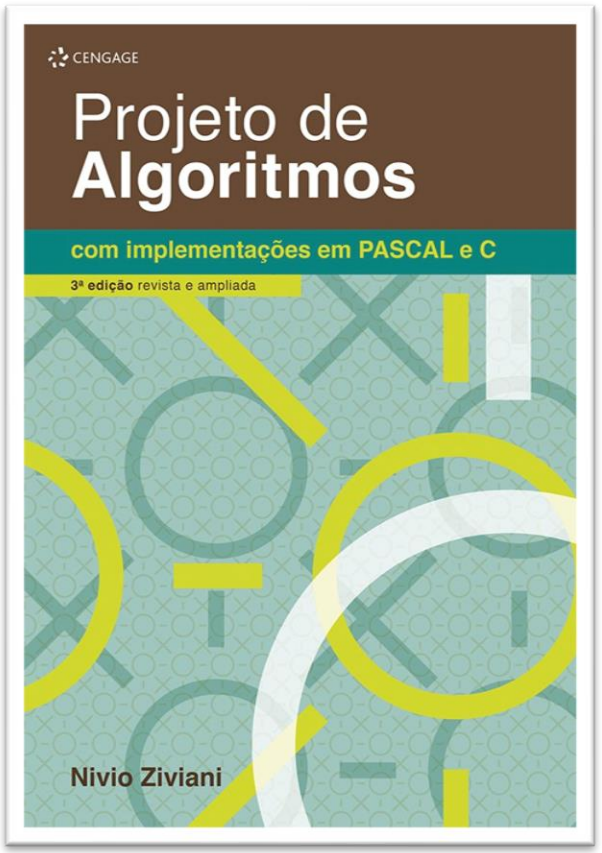
Plano de Aula

- Revisão
- Estruturas não-lineares
- Árvore
  - Definição
  - Nomenclatura
  - Aplicações
    - HeapSort
    - Busca
    - Sintaxe de Expressões
- Exercícios

Plano de Aula

- Revisão
- Estruturas não-lineares
- Árvore
  - Definição
  - Nomenclatura
  - Aplicações
    - HeapSort
    - Busca
    - Sintaxe de Expressões
- Exercícios





Revisão

Dados podem ser estruturados de inúmeras formas

Revisão

Dados podem ser estruturados de inúmeras formas

- Arranjados, organizados, ordenados, classificados...

Revisão

Dados podem ser **estruturados** de inúmeras formas

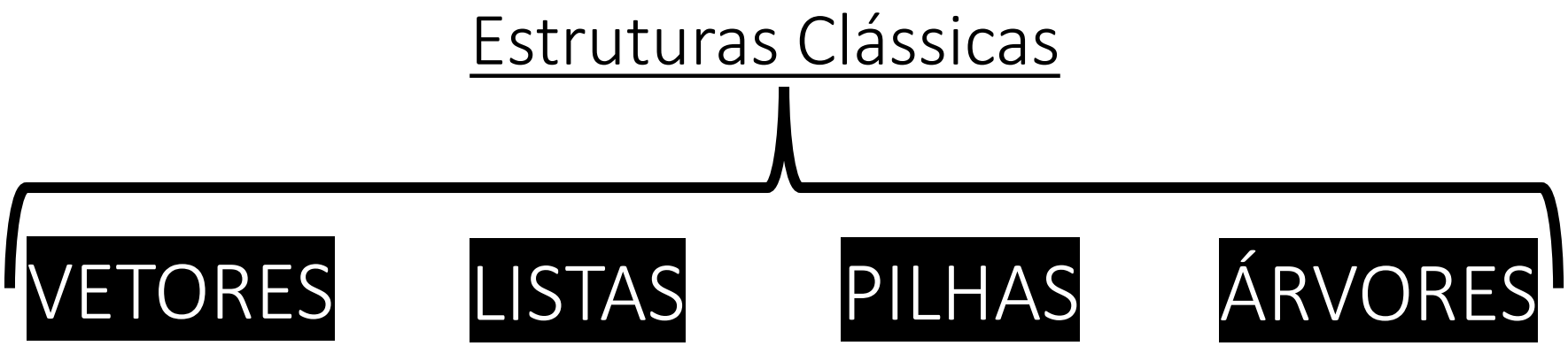
→ Arranjados, organizados, ordenados, classificados...

VETORES **LISTAS** **PILHAS** **ÁRVORES**

Revisão

Dados podem ser **estruturados** de inúmeras formas

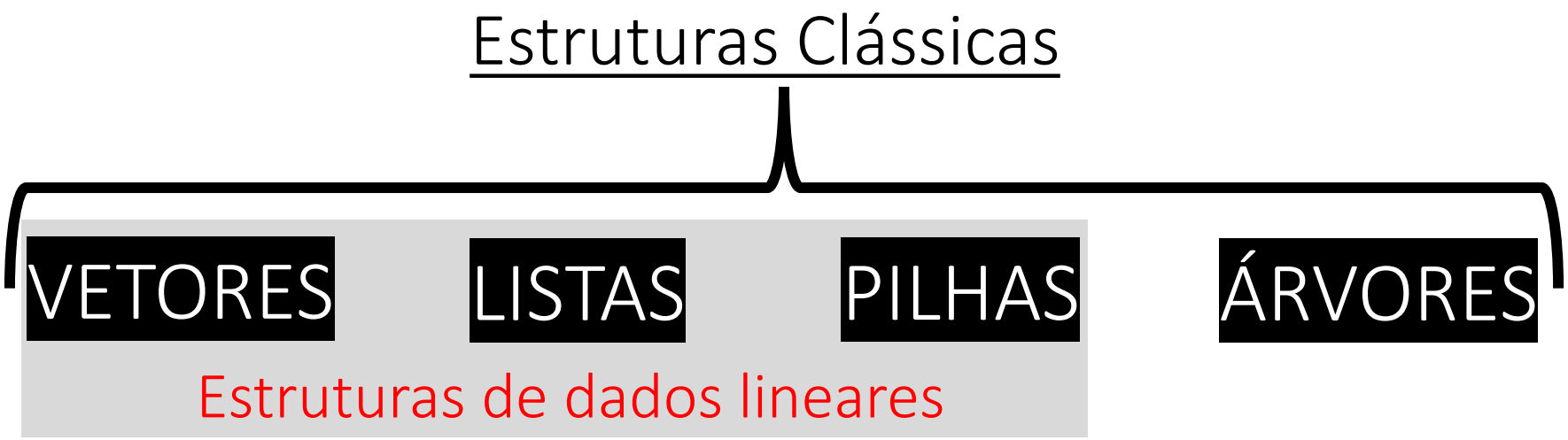
→ Arranjados, organizados, ordenados, classificados...



Revisão

Dados podem ser **estruturados** de inúmeras formas

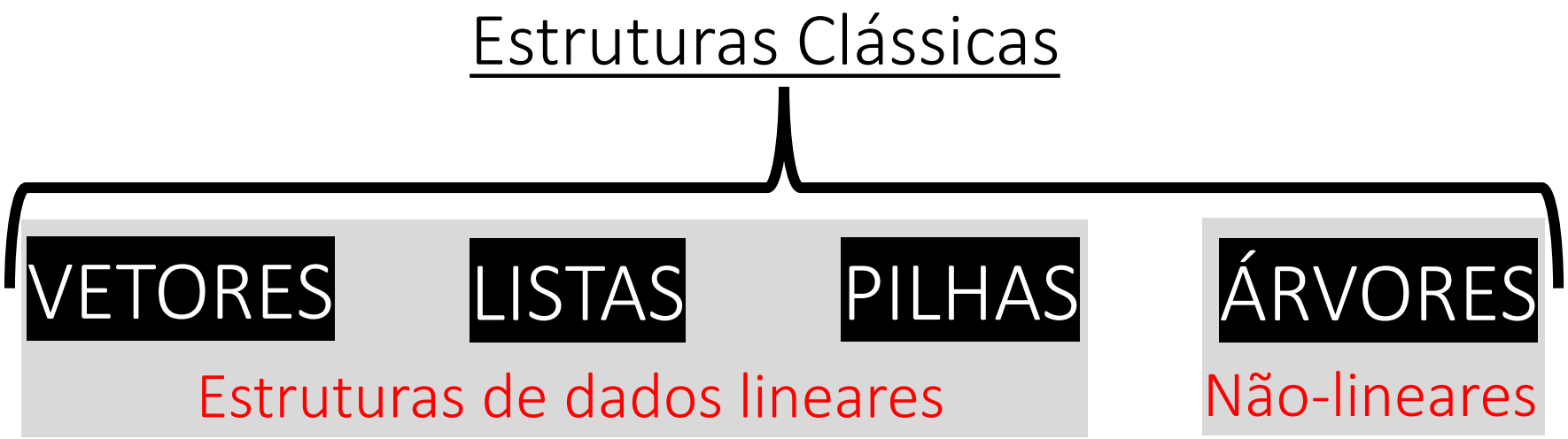
→ Arranjados, organizados, ordenados, classificados...



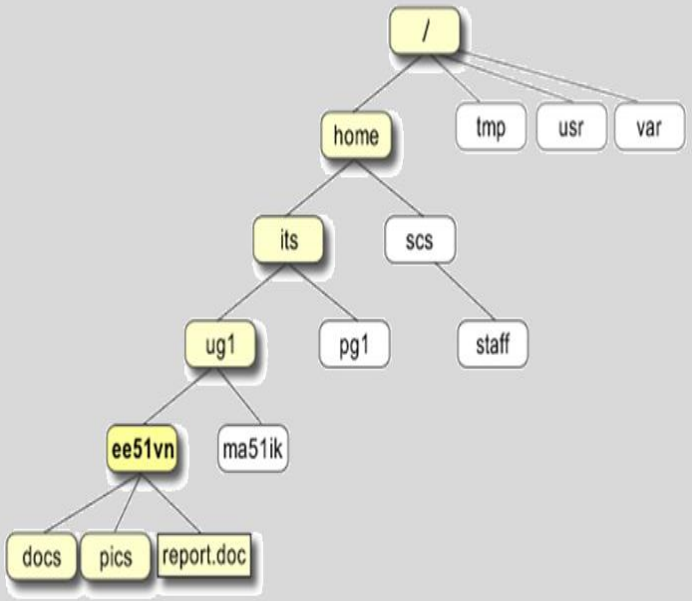
Revisão

Dados podem ser **estruturados** de inúmeras formas

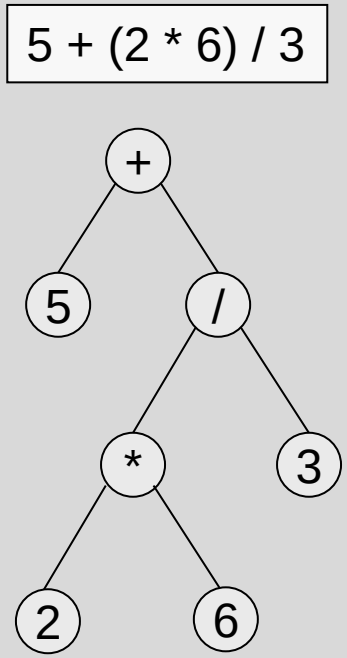
→ Arranjados, organizados, ordenados, classificados...



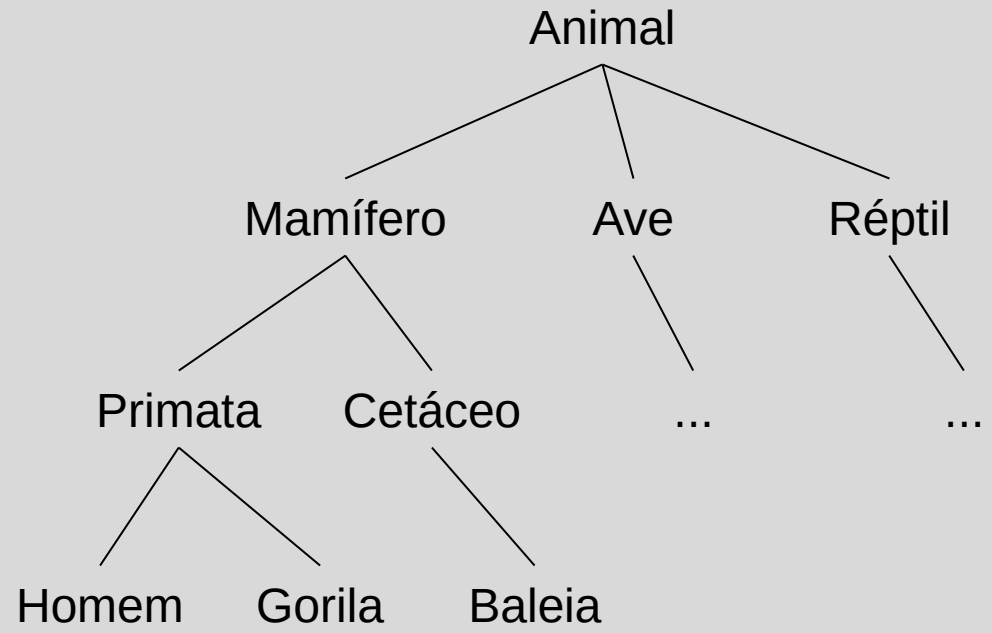
Estruturas não-lineares



Sistema de Arquivos

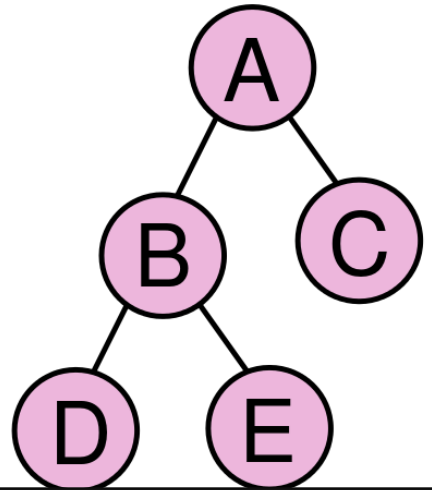


Expressões



Classificação dos seres vivos (taxonomia)

Estruturas não-lineares



Hierárquica

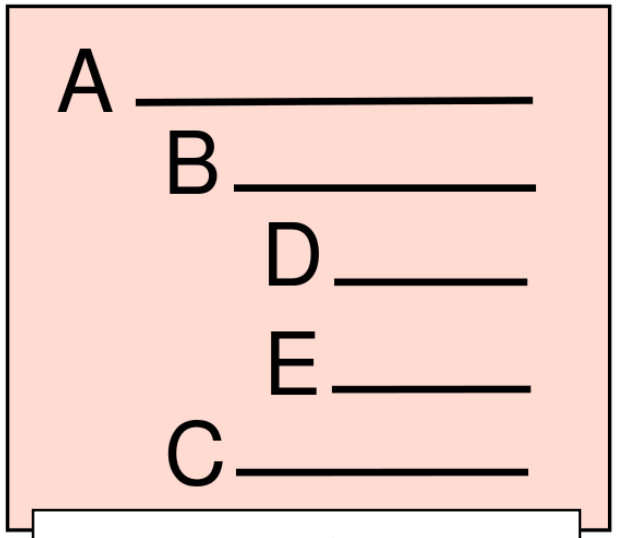


Diagrama de Barras

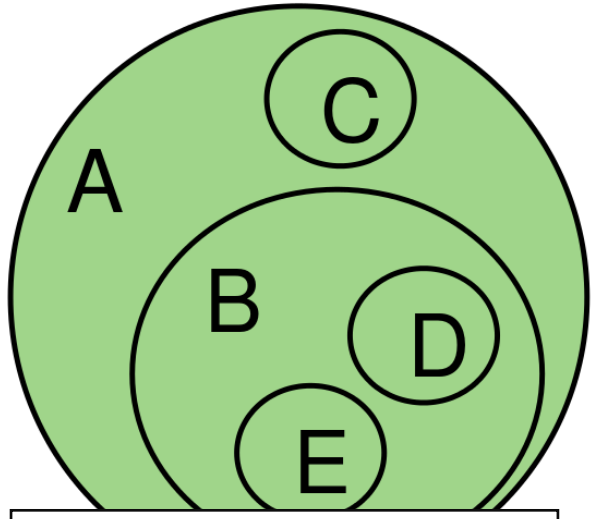


Diagrama de Conjuntos

1A; 1.1B; 1.1.1D; 1.1.2E; 1.2C

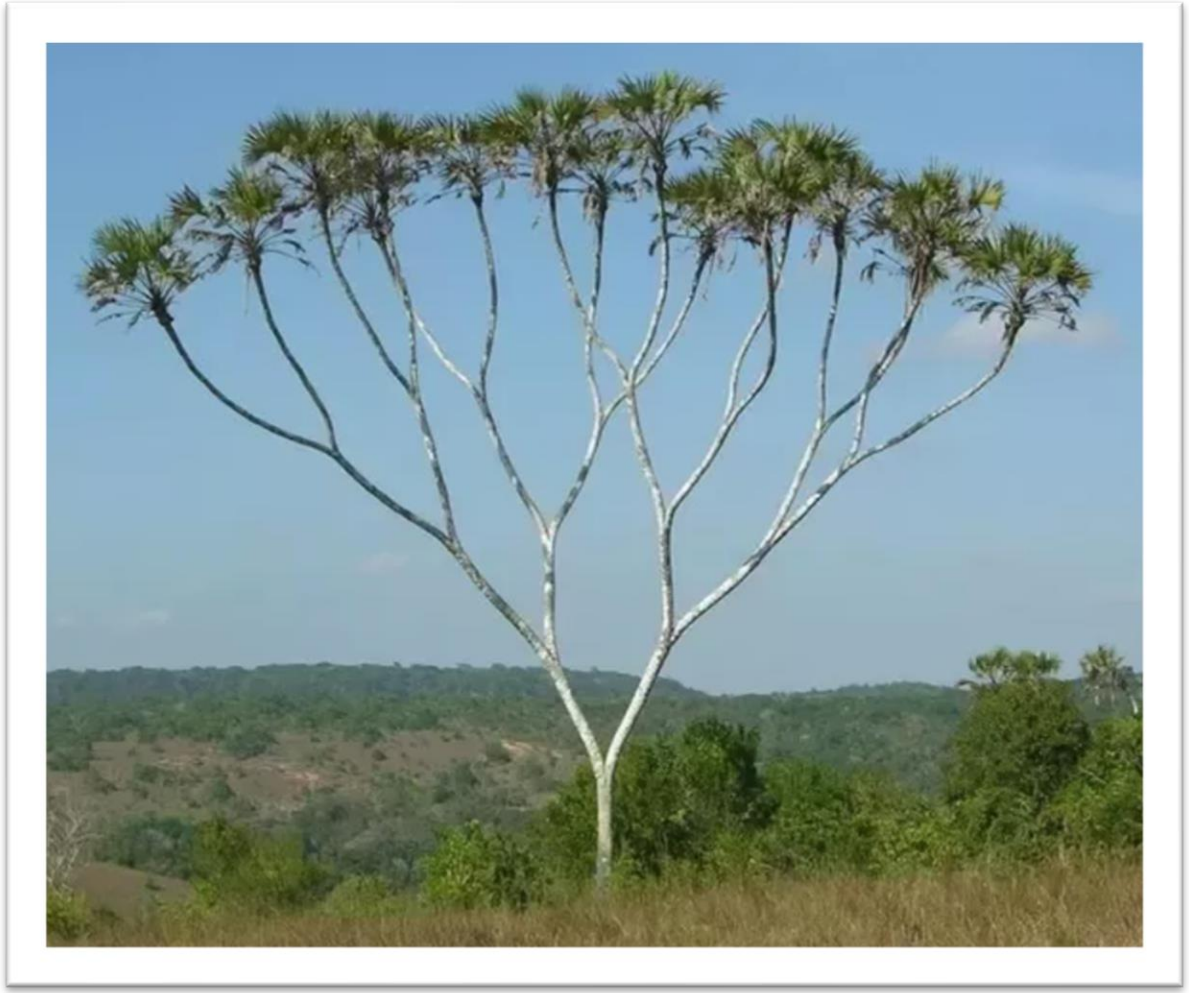
Lista Enumerada

(A(B(D)(E))(C))

Aninhamento

Árvore

Árvore



Árvore

Árvore

ÁRVORE AVL

ÁRVORE BINÁRIA

ÁRVORE HUFFMAN

ÁRVORE B

ÁRVORE B*

ÁRVORE B+

ÁRVORE 2-3

Árvore: Definição

Árvore: um conjunto finito e não-vazio de elementos, com um elemento inicial que é particionado em $m \geq 0$ subconjuntos disjuntos, cada um dos quais sendo uma árvore em si mesmo.

Árvore Binária: um conjunto finito e não-vazio de elementos, com um elemento inicial que é particionado em $0 \leq m \leq 2$ subconjuntos disjuntos, cada um dos quais sendo uma árvore em si mesmo.

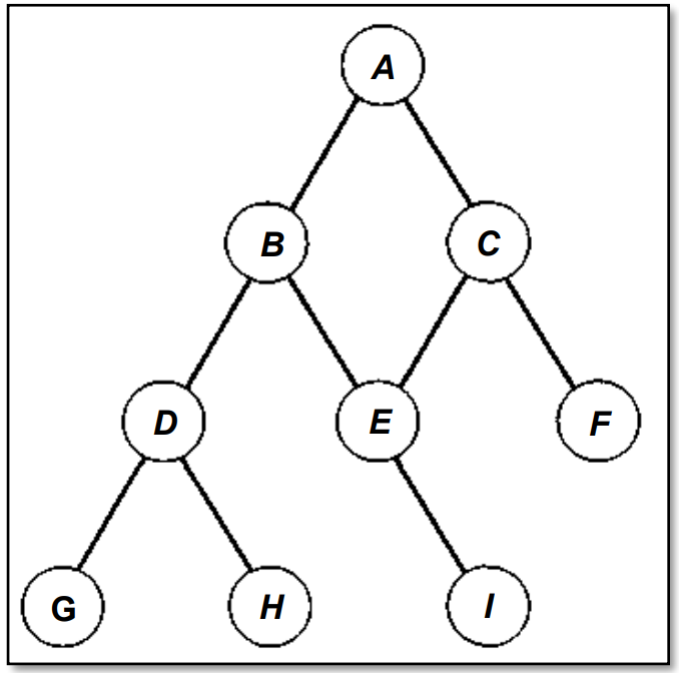
Árvore: Definição

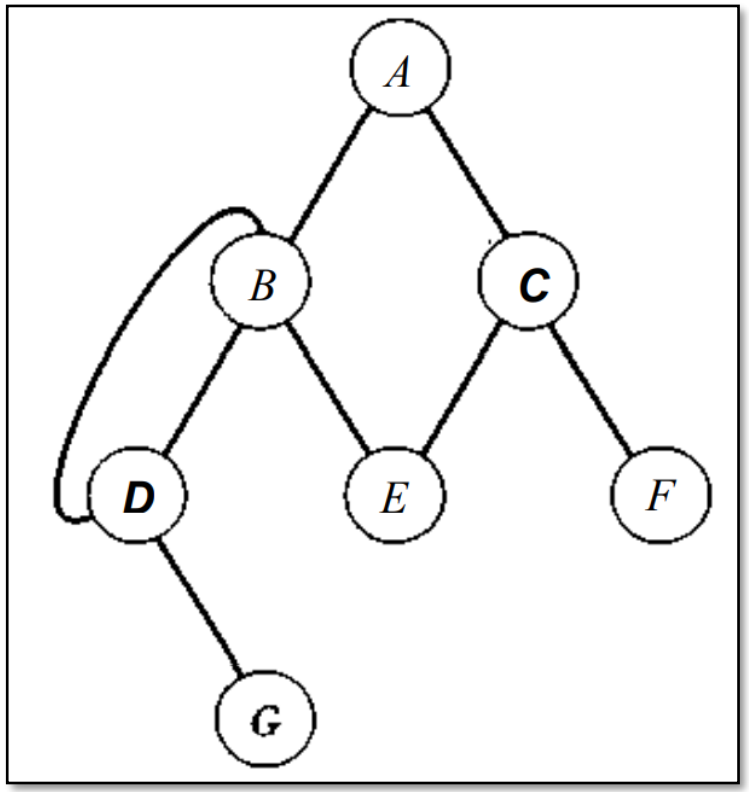
Árvore: um conjunto finito e não-vazio de elementos, com um elemento inicial que é particionado em $m \geq 0$ subconjuntos disjuntos, cada um dos quais sendo uma árvore em si mesmo.

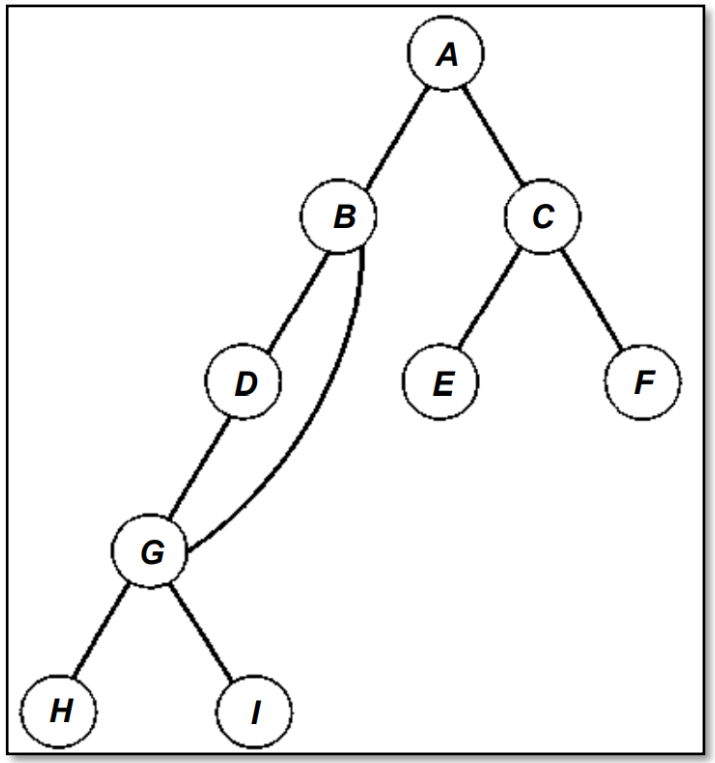
estritamente

Árvore Binária: um conjunto finito e não-vazio de elementos, com um elemento inicial que é particionado em $0 = m = 2$ subconjuntos disjuntos, cada um dos quais sendo uma árvore em si mesmo.

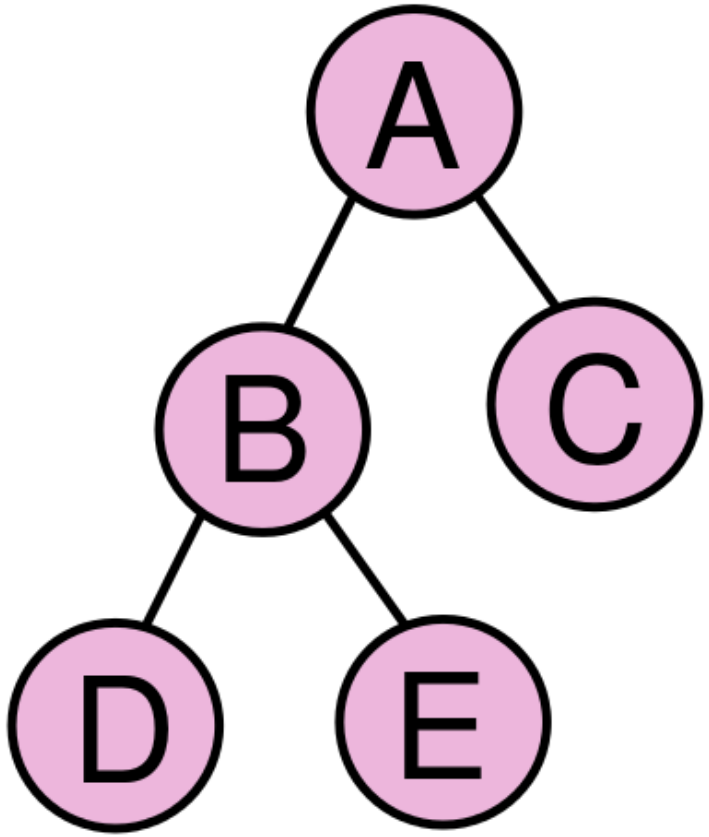
Exercícios



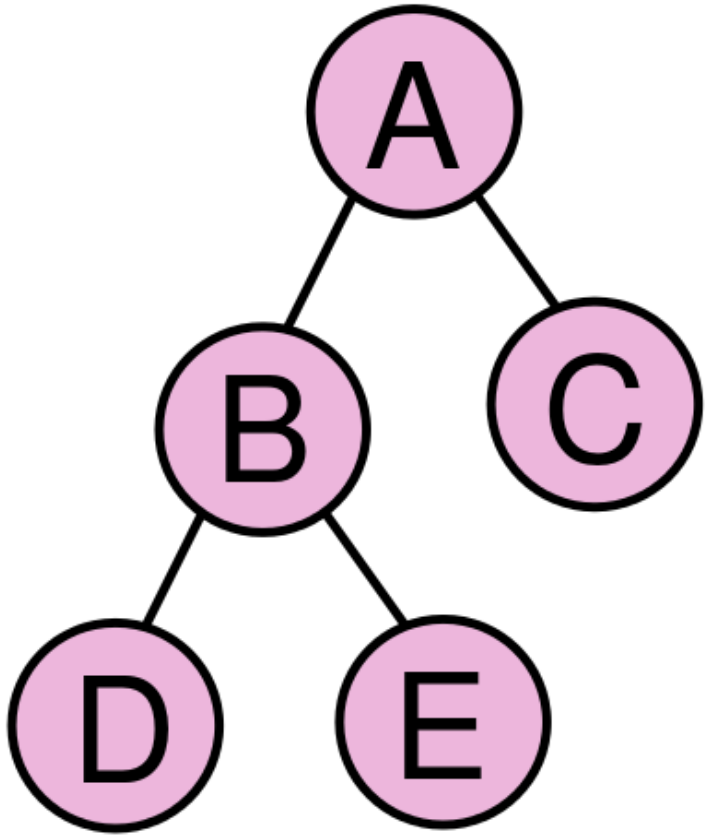




Árvore: Nomenclatura



Árvore: Nomenclatura



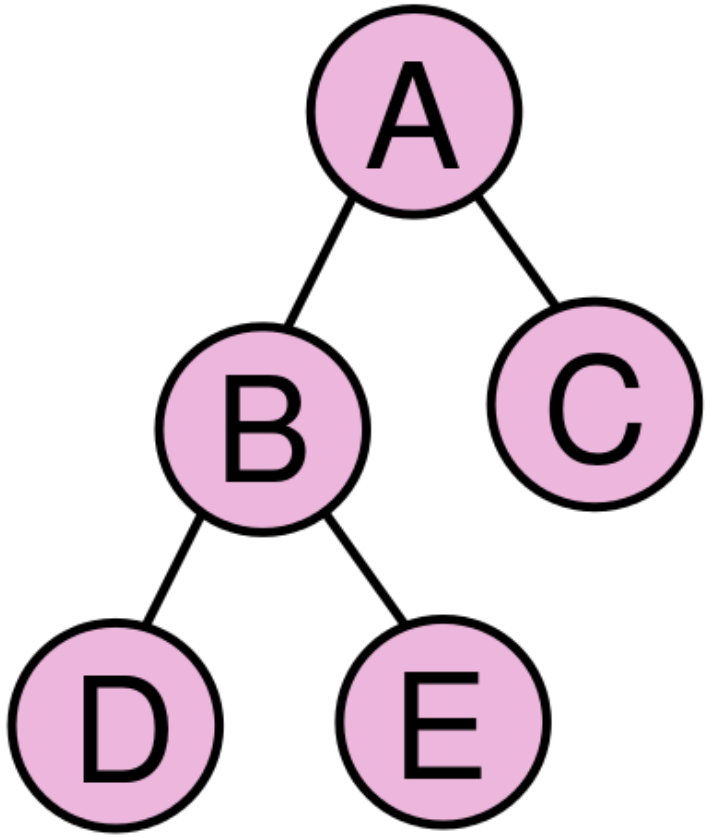
Nó: A B C D E

Raiz: A

Folha: D E C

Nós internos: A B

Árvore: Nomenclatura



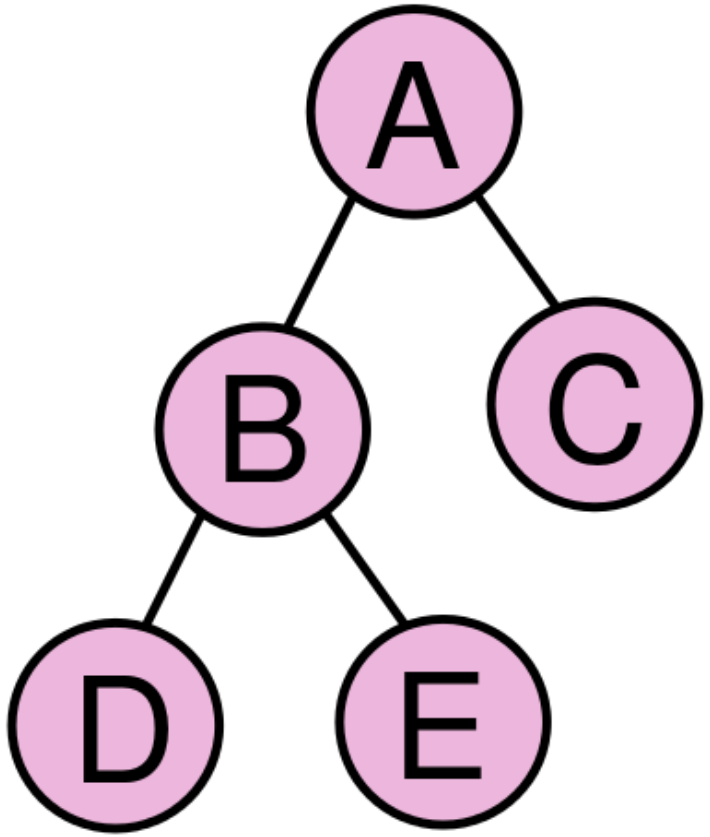
Altura:
Maior distância entre um nó raiz e um nó folha

Grau:
Representa o número de subárvores de um nó

Aridade:
Representa o grau máximo de uma árvore

Nível (nó):
Números de nós até o nó raiz

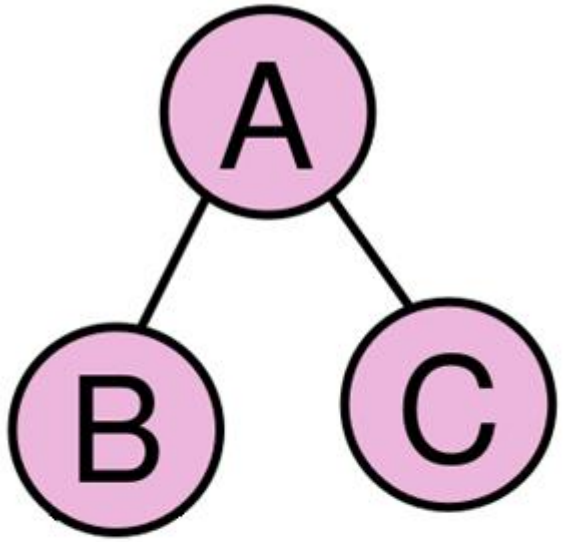
Árvore: Nomenclatura



Árvore Estritamente Binária:

Cada nó tem 2 filhos ou nenhum

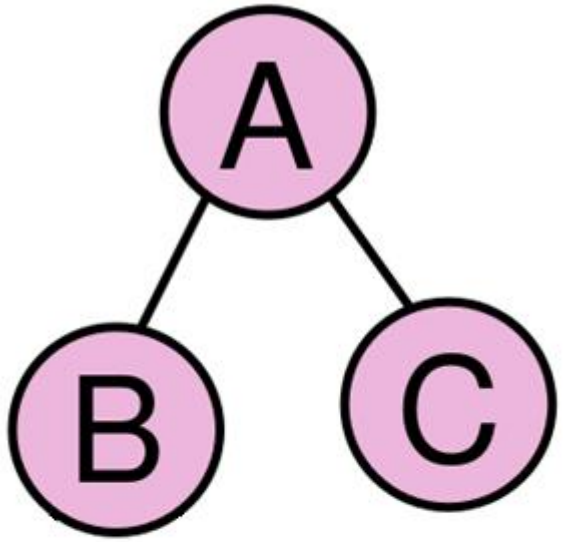
Árvore: Nomenclatura



Árvore Completa (Cheia):

É uma árvore estritamente binária onde todas as folhas estão no mesmo nível

Árvore: Nomenclatura



Árvore Completa (Cheia):

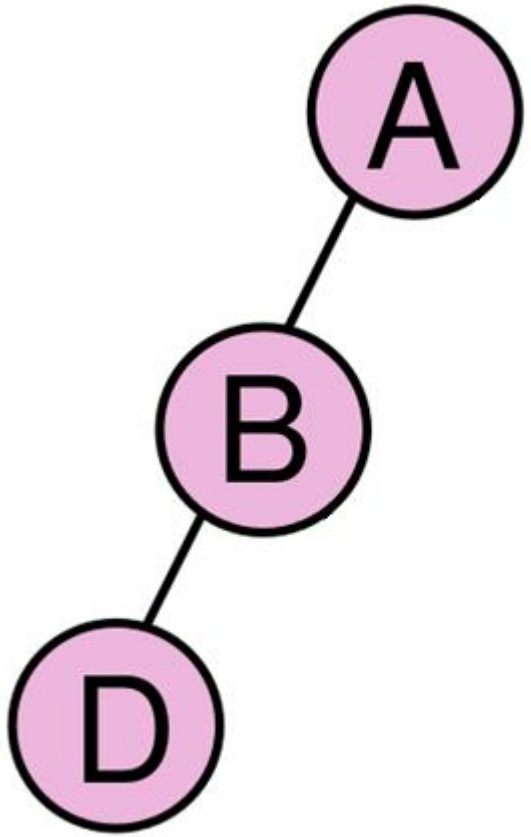
É uma árvore estritamente binária onde todas as folhas estão no mesmo nível

Árvore Balanceada:

É uma árvore onde cada nó possui a diferença de altura de suas subárvores de no máximo **1**

Fator de Balanceamento	=	AlturaEsquerda – AlturaDireita

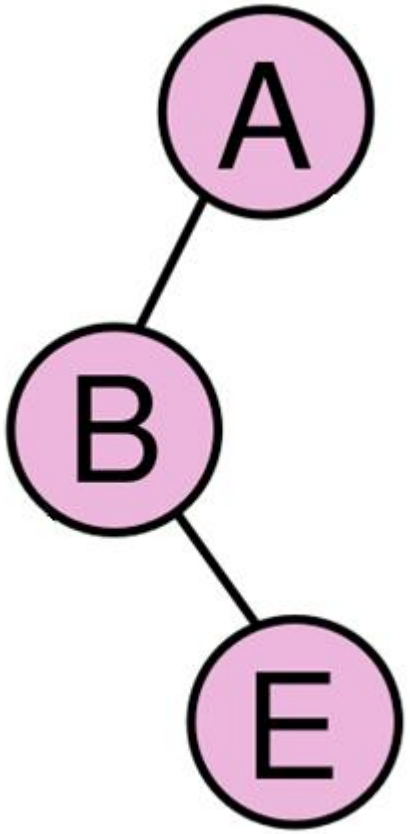
Árvore: Nomenclatura



Árvore Degenerada

É uma árvore onde todos os nós tem no máximo **1** filho

Árvore: Nomenclatura



Árvore Degenerada

É uma árvore onde todos os nós tem no máximo **1** filho

Árvore Binária Zigue-Zague

É uma árvore onde todos os nós tem no máximo **1** filho

Aplicações

Uma árvore binária é uma estrutura de dados útil quando precisam ser tomadas decisões bidirecionais em cada ponto de um processo.

Útil para:

- HeapSort
- Busca/Pesquisa
- Sintaxe de Expressões

Sintaxe de Expressões

Notação Polonesa
(Prefixa)

Notação Polonesa Inversa
(Posfixa)

Sintaxe de Expressões

Notação Polonesa
(Prefixa)

+ A B

+5 / *2 6 3

Notação Polonesa Inversa
(Posfixa)

A B +

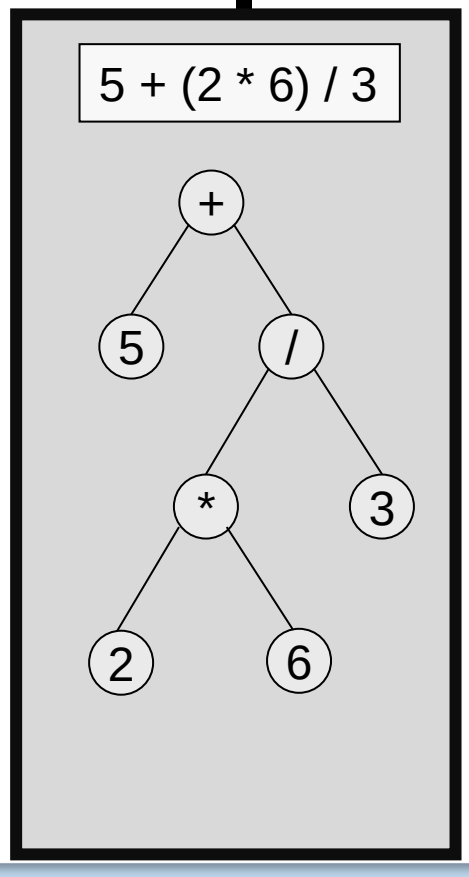
5 2 6* 3/+

Sintaxe de Expressões

Notação Polonesa
(Prefixa)

+ A B

+5 / *2 6 3



Notação Polonesa Inversa
(Posfixa)

A B +

5 2 6* 3/+

Busca (Algoritmos de Travessia)

Pré-Ordem

1. Visitar nó.
2. Percorrer em pré-ordem o ramo formado pelas subárvores da primeira árvore, se houver alguma.
3. Percorrer em pré-ordem o ramo formado pelas árvores restantes, se houver alguma.

Em Ordem

1. Percorrer em ordem o ramo formado pelas subárvores da primeira árvore, se houver alguma.
2. Visitar nó.
3. Percorrer em ordem o ramo formado pelas árvores restantes, se houver alguma.

Pós-Ordem

1. Percorrer em pós-ordem o ramo formado pelas subárvores da primeira árvore, se houver alguma.
2. Percorrer em pós-ordem o ramo formado pelas subárvores restantes, se houver alguma.
3. Visitar nó.

Busca (Algoritmos de Travessia)

Pré-Ordem

1. Lê nó.
2. Esquerda.
3. Direita.

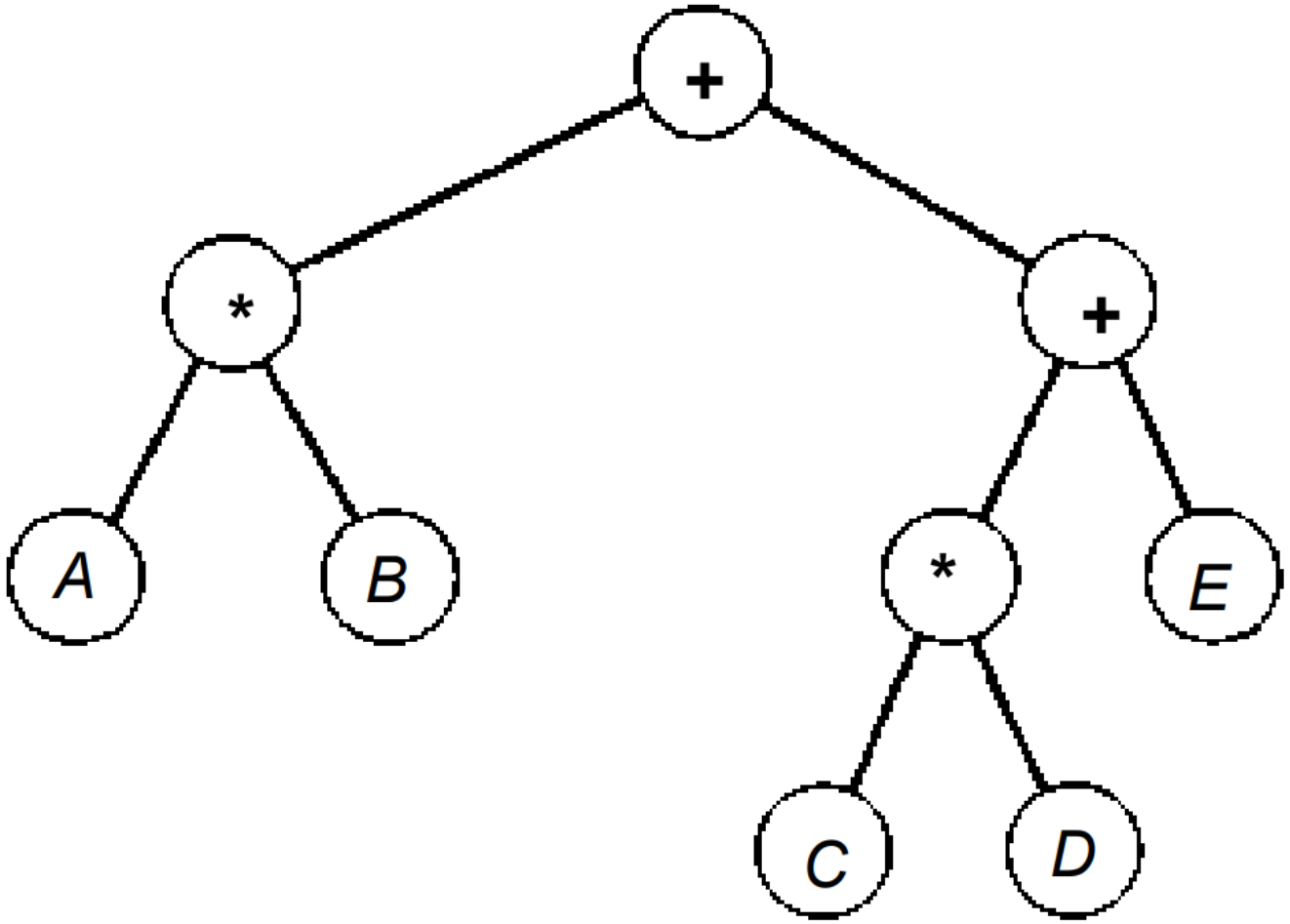
Em Ordem

1. Esquerda.
2. Lê nó.
3. Direita.

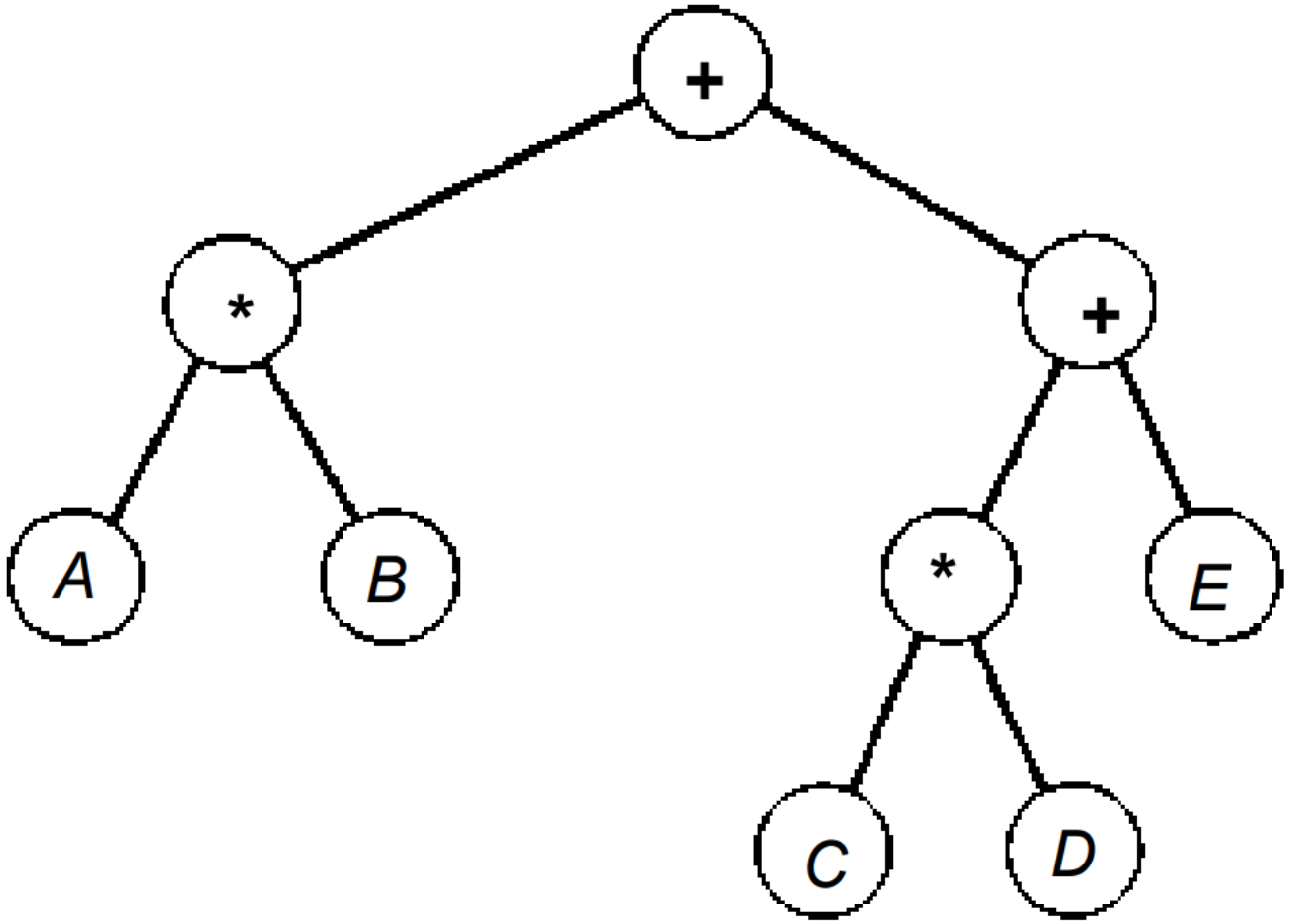
Pós-Ordem

1. Esquerda.
2. Direita.
3. Lê nó.

Exercício



Exercício

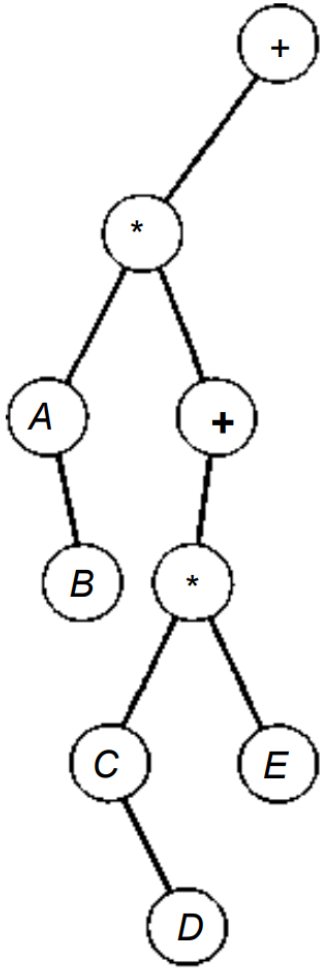


Pré-ordem: +*AB + * CDE

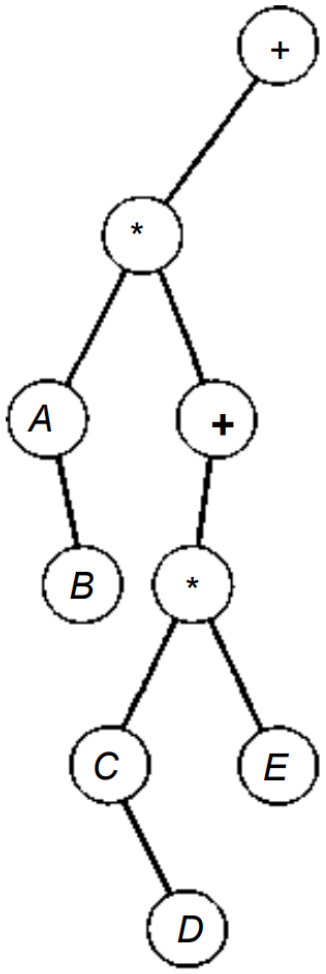
Em-ordem: A*B + C*D + E

Pós-ordem: AB*CD * E + +

Exercício



Exercício



Pré-ordem: +* AB + * CDE

Em-ordem: AB * CD * E + +

Pós-ordem: BADCE * + * +

Referência

ZIVIANI, Nivio. **Projeto de Algoritmos**: com implementações em Pascal e C. 3. ed. rev. e ampl. São Paulo: Pioneira Thomson Learning, 2005

TENENBAUM, A. M. **Estruturas de dados usando C**. São Paulo: Makron Books: 1995. ISBN 85-346-0348-0

Extra – Implementação em C

```
struct no
{
    char info;
    struct node *left;
    struct node *right;
};
typedef struct no node;
```

```
node* maketree(int x)
{
  node *p;
  p = getnode();
  p->info = x;
  p->left = NULL;
  p->right = NULL;
  return (p);
}
```

```
setleft (node *p, int x)
{
  if (p == null) printf ("inserção vazia\n");
  else if (p->left != NULL)
    printf ("inserção incorreta\n");
  else
    p->left = maketree(x);
}
```

```
setright (node *p, int x)
{
  if (p == null) printf ("inserção vazia\n");
  else if (p->right != NULL)
    printf ("inserção incorreta\n");
  else
    p->right = maketree(x);
}
```