Árvore binária

Estruturas de Dados

Bruno Prado

Departamento de Computação / UFS

Introdução

- O que é uma árvore (em computação)?
  - Representação natural de informações hierárquicas
    - Árvore genealógica
    - Organograma
    - Estruturas de diretórios
    - ...
  - Eficiência e simplicidade

Introdução

- Tipos de árvores
  - Árvore livre ou não enraizada (*unrooted*) ou grafo
  - Árvore enraizada (*rooted*)
  - Árvore k-ária

Introdução

- Grafo
  - Conjunto de vértices V e arestas E
  - Os vértices são conectados pelas arestas
  - As arestas podem conter pesos associados

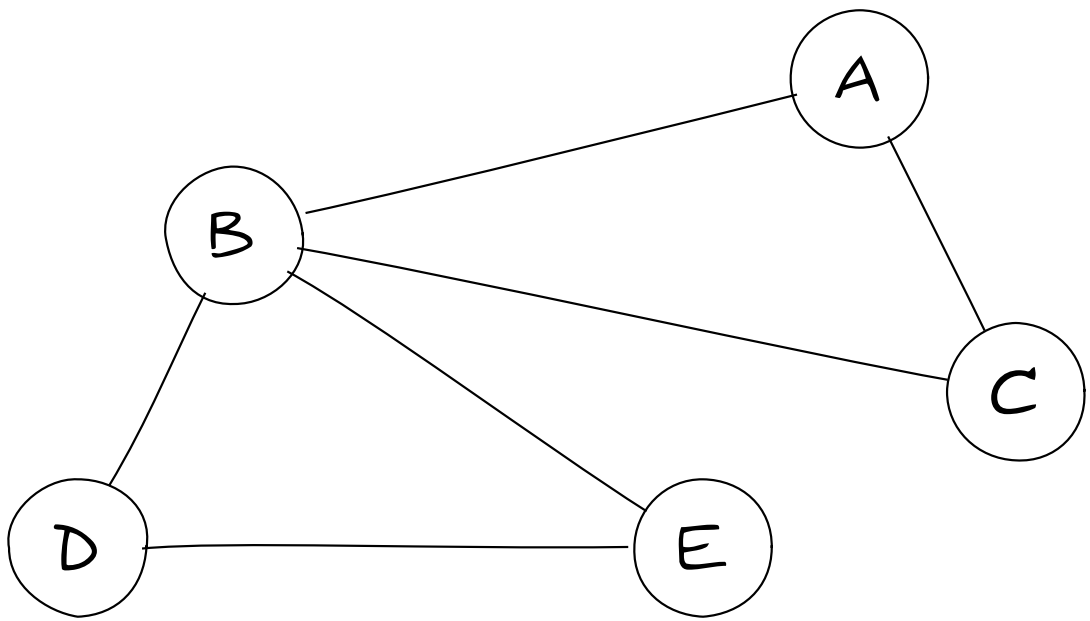
$$G = (V, E)$$

$$V = \{v_1, v_2, \ldots, v_n\}$$

$$E = \{e_1, e_2, \ldots, e_m\}$$

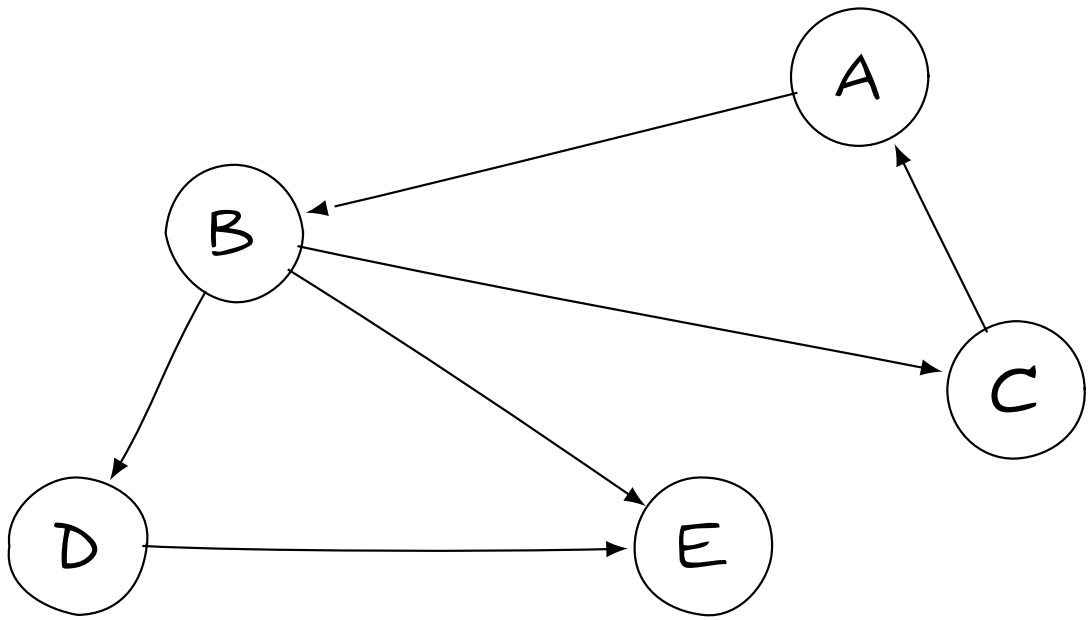
Introdução

- Grafo não direcionado
  - $V = \{A, B, C, D, E\}$
  - $E = \{AB, AC, BA, BC, BD, BE, CA, CB, DB, DE, EB, ED\}$



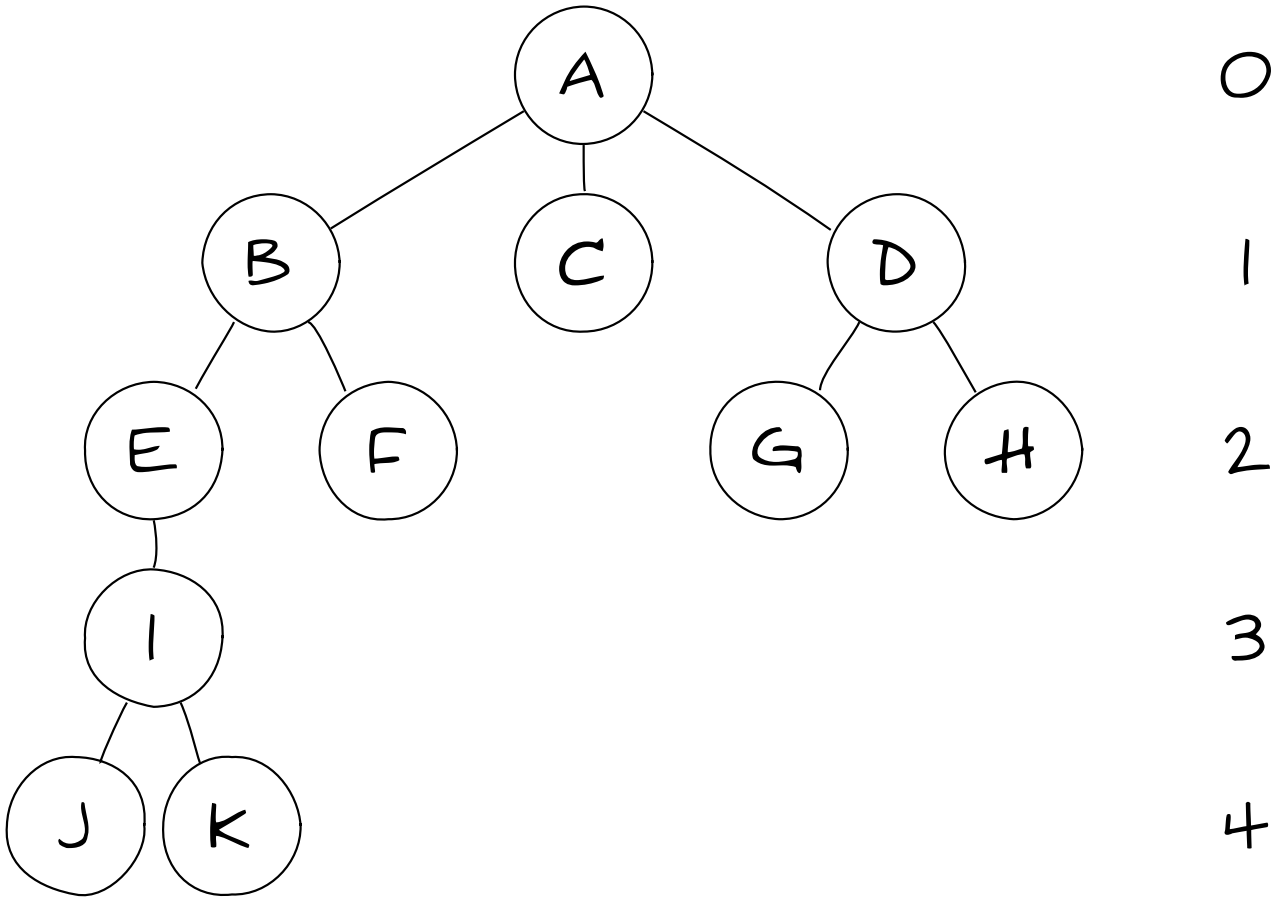
Introdução

- Grafo direcionado (dígrafo)
  - $V = \{A, B, C, D, E\}$
  - $E = \{AB, BD, BC, BE, CA, DE\}$



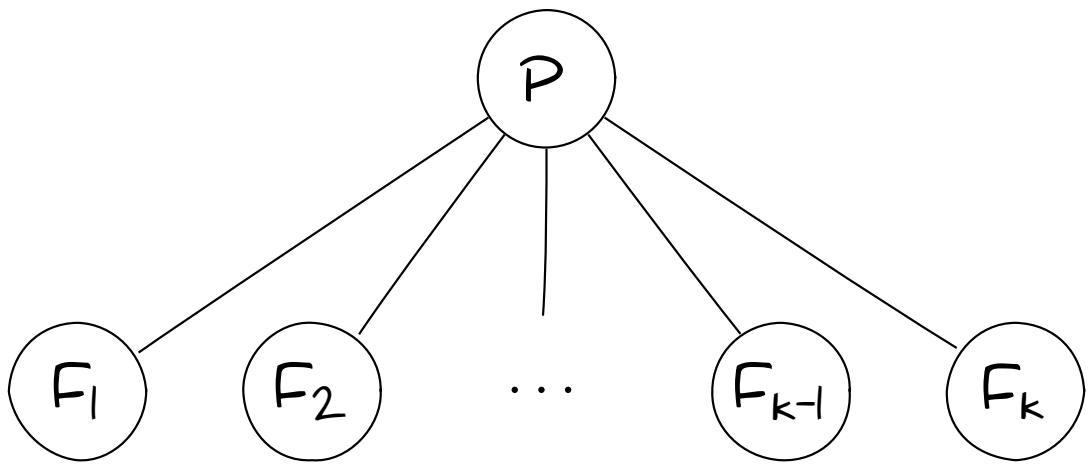
Introdução

- Árvore enraizada
  - Conjunto de nós ou de vértices, com um nó especial que é denominado de raiz
  - Possuem níveis e as partições geram subárvores



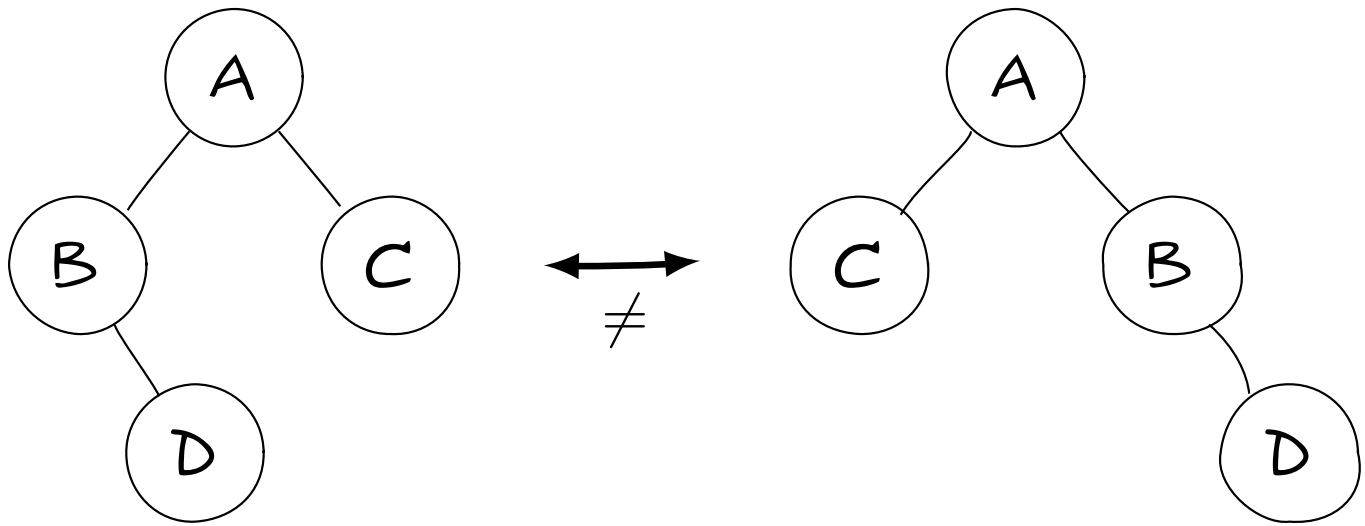
Introdução

- Árvore k-ária
  - Cada nó ou vértice tem no máximo k subárvores
  - O número de subárvores define o grau do vértice
  - Conceito de pai P e filho F_i



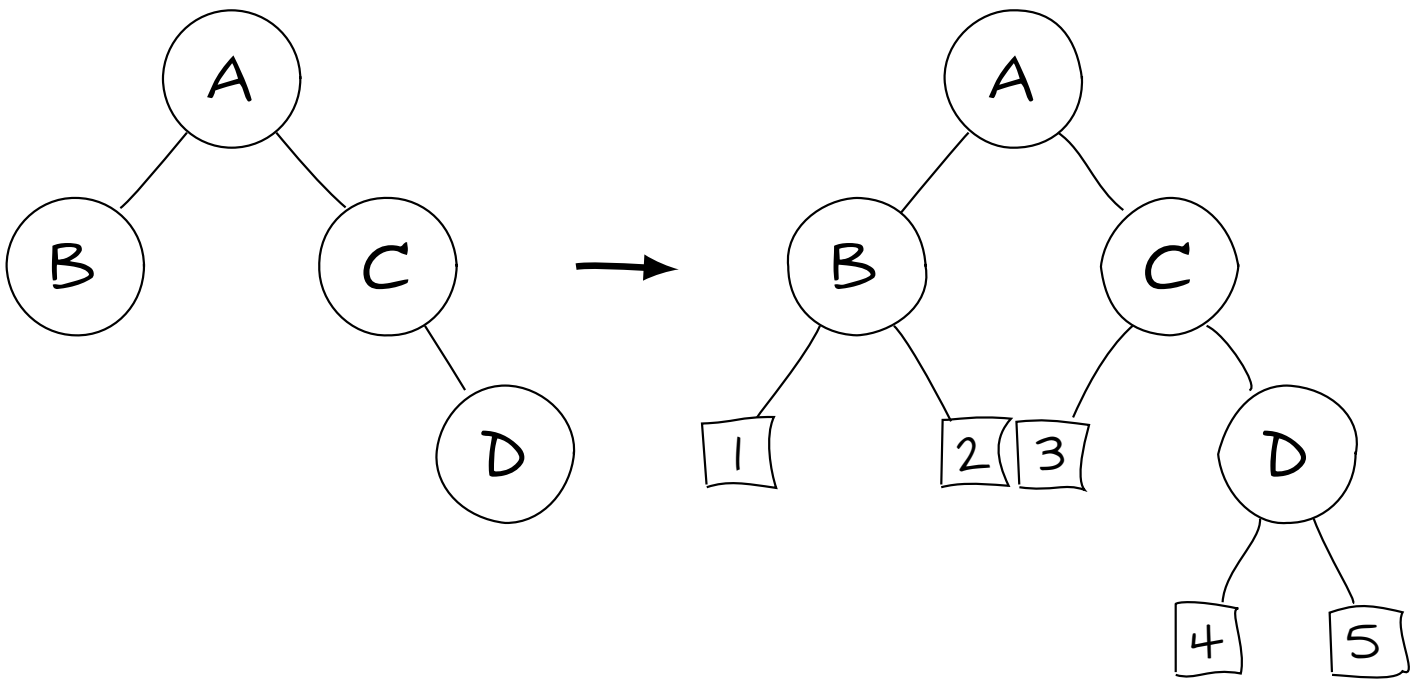
Árvore binária

- O que é uma árvore binária?
  - É uma árvore enraizada cujos nós pai referenciam até 2 nós filho ou 2 subárvores
  - Assimetria dos nós e dos ramos



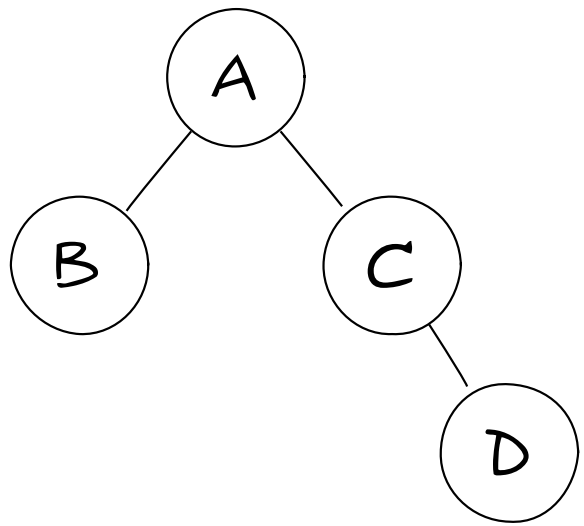
Árvore binária

- Propriedades
  - Uma árvore binária com n nós internos (redondos) possui $n + 1$ nós externos (quadrados)
    - Cada nó interno possui exatamente dois filhos, utilizando um total de $2n$ ramos
    - Existem $n - 1$ ramos para os n nós internos provenientes dos nós pais, excluindo o nó raiz que não possui ramo
    - Restam $n + 1$ ramos para cada nó externo



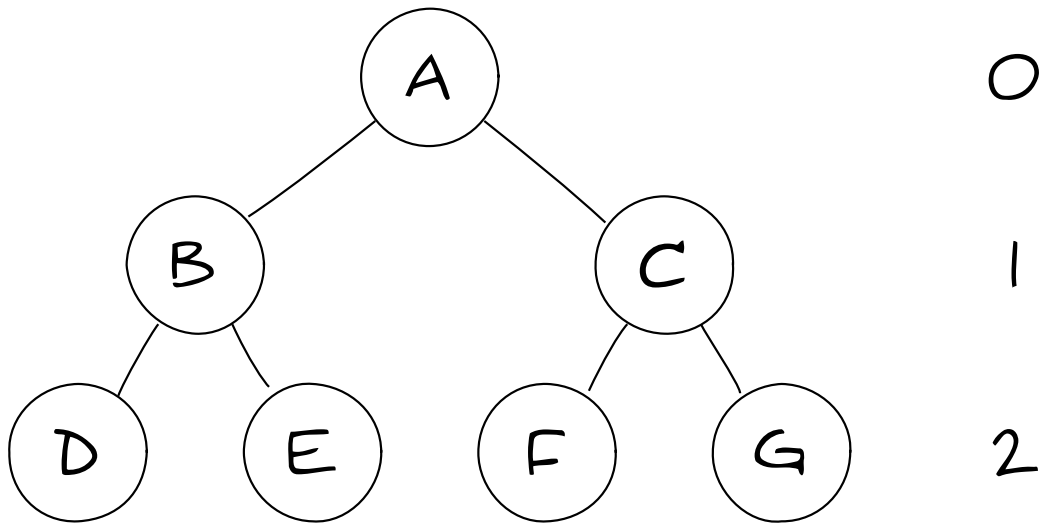
Árvore binária

- Propriedades
  - Qualquer árvore binária não vazia com n_0 nós folha e n_2 nós com grau 2, temos que $n_0 = n_2 + 1$
    - Total de nós é $n = n_0 + n_1 + n_2$
    - A árvore possui $n - 1$ ramos, todos partindo de nós que possuem grau 1 ou 2
    - Implicando em $n - 1 = n_1 + 2n_2 \rightarrow n = n_1 + 2n_2 + 1$



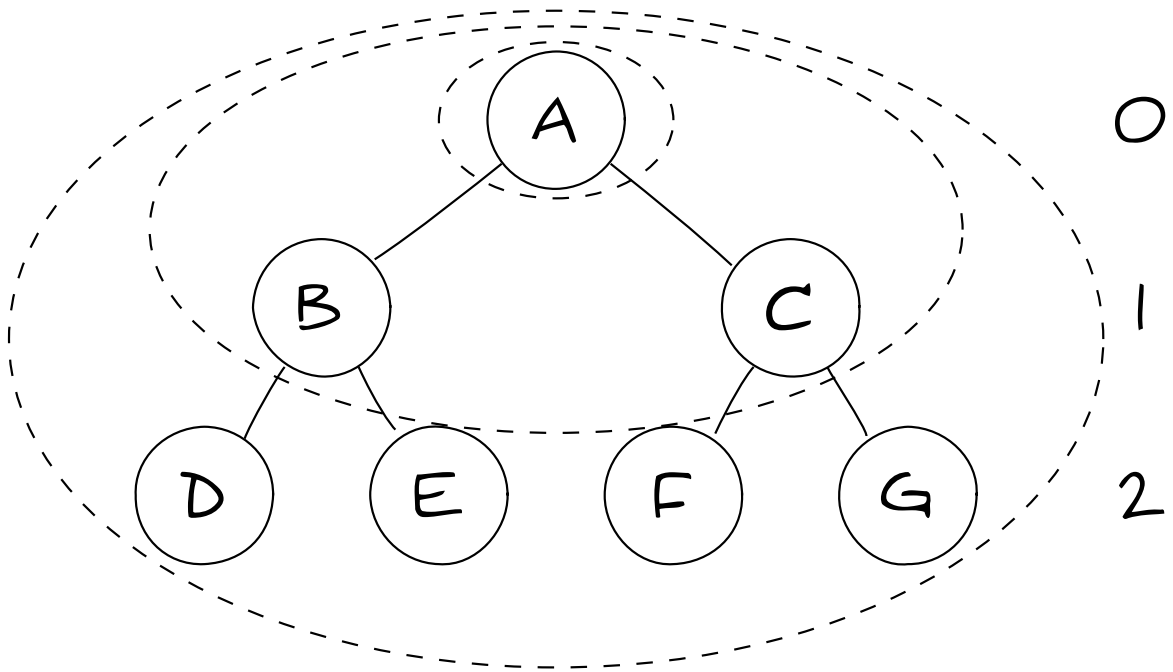
Árvore binária

- Propriedades
  - O número máximo de nós em um nível i é 2^i para $i \geq 0$
    - Progressão geométrica
    - Cada passo duplica o número máximo de nós



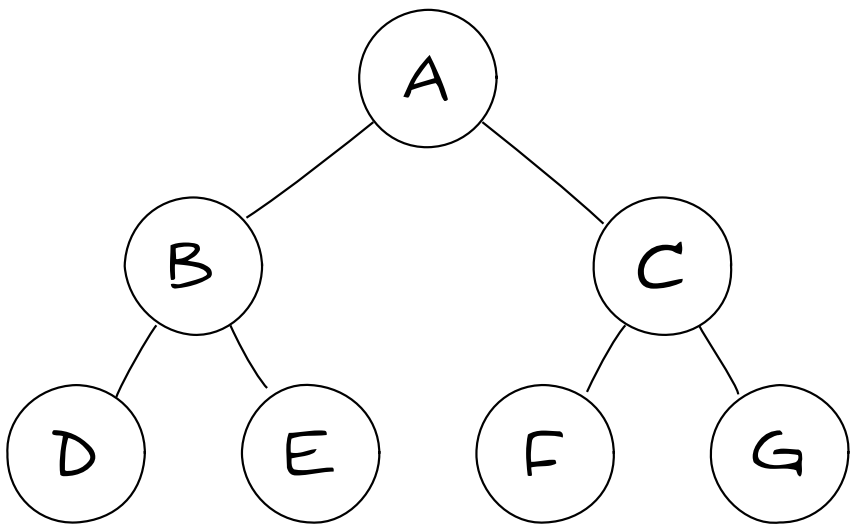
Árvore binária

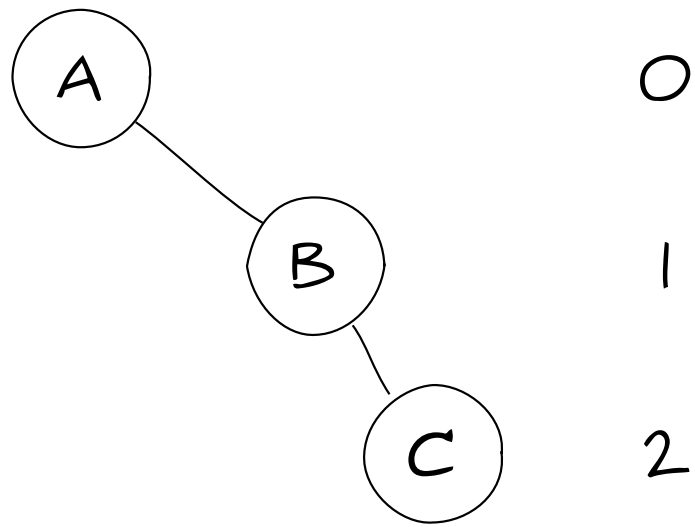
- Propriedades
  - Uma árvore de altura h possui até $2^{h+1} - 1$ nós
    - Cada nível possui no máximo 2^i nós
    - Temos que $0 \leq i \leq h$
    - Total de nós é $\sum_{i=0}^{h} 2^i$ e utilizando o somatório de progressão geométrica $\sum_{k=1}^{n} ar^{k-1} = \frac{a(1-r^n)}{1-r}$, é obtido $\sum_{k=1}^{h+1} 2^{k-1} = \frac{1(1-2^{h+1})}{1-2} = 2^{h+1} - 1$



Árvore binária

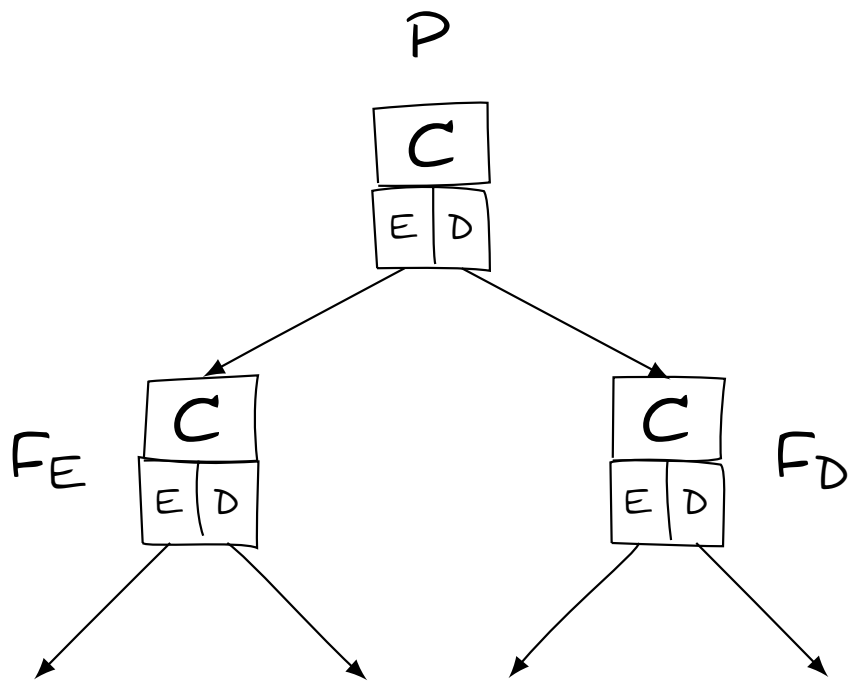
- Propriedades
  - A altura h de uma árvore binária com n nós internos é pelo menos $\log_2(n+1) - 1$ e no máximo $n - 1$
    - No caso inferior, com níveis com 2^i nós e total máximo de nós igual a $n \leq 2^{h+1} - 1$ e $h \geq \log_2(n+1) - 1$
    - No caso superior, o número de nós externos é $n + 1$ e que temos $n - 1$ ramos que é igual a altura h





Árvore binária

- Definição da estrutura



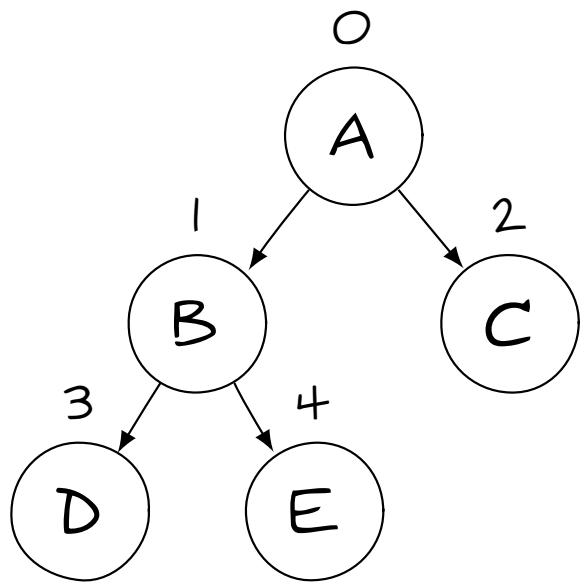
Árvore binária

- Implementação em C
  - Estrutura e ponteiros

```
1  // Padrão de tipos por tamanho
2  #include <stdint.h>
3  // Estrutura de nó
4  typedef struct no {
5      // Chave do nó
6      uint32_t C;
7      // Filho da direita
8      struct no* D;
9      // Filho da esquerda
10     struct no* E;
11 } no;
```

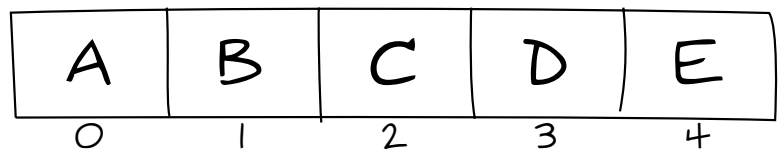
Árvore binária

- Implementação em C
  - Vetores e índices



Árvore binária

- Implementação em C
  - Vetores e índices
    - Pai com índice i
    - Filho da esquerda é $2i + 1$
    - Filha da direita é $2i + 2$



Árvore binária

- Implementação em C
  - Vetores e índices

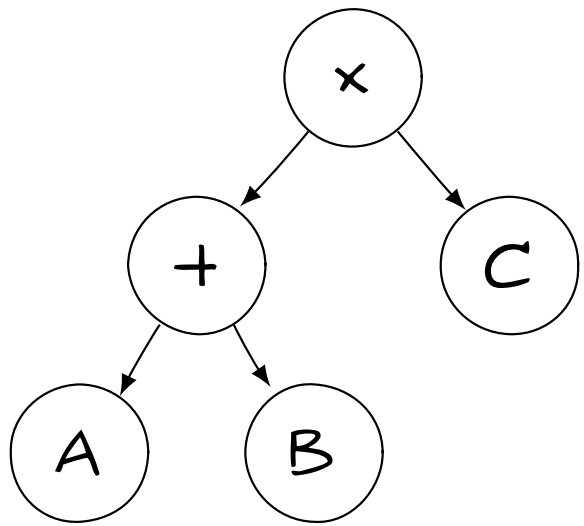
```
1  // Padrão de tipos por tamanho
2  #include <stdint.h>
3  // Função direita
4  uint32_t direita(uint32_t i) {
5      return 2 * i + 2;
6  }
7  // Função esquerda
8  uint32_t esquerda(uint32_t i) {
9      return 2 * i + 1;
10 }
```

Árvore binária

- Percursos na árvore: definem como os nós da árvore serão visitados
  - Em ordem (*EPD*)
  - Pré-ordem (*PED*)
  - Pós-ordem (*EDP*)

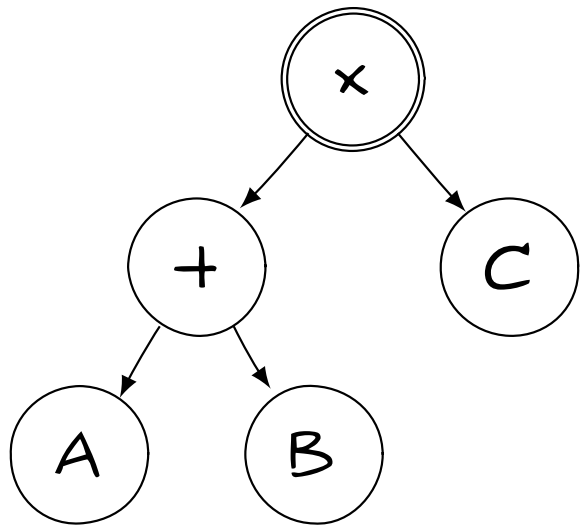
Árvore binária

- Percurso em ordem
  - Expressão $(A + B) \times C$
  - Regra *EPD*



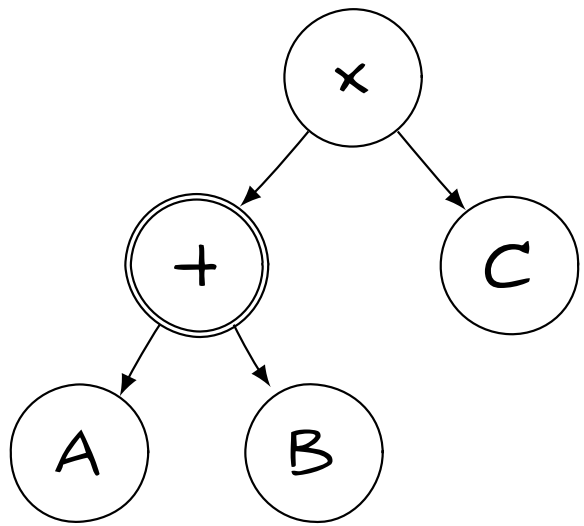
Árvore binária

- Percurso em ordem
  - Expressão $(A + B) \times C$
  - Regra *EPD*



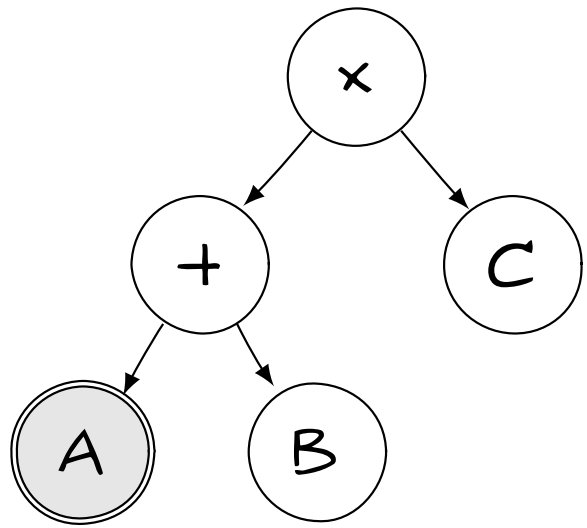
Árvore binária

- Percurso em ordem
  - Expressão $(A + B) \times C$
  - Regra *EPD*



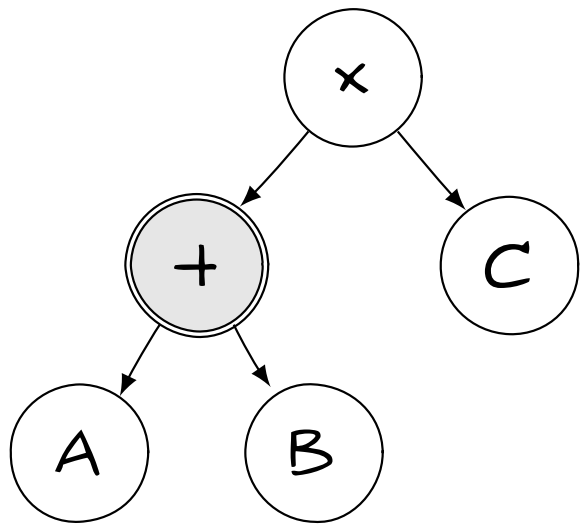
Árvore binária

- Percurso em ordem
  - Expressão $(A + B) \times C$
  - Regra *EPD*



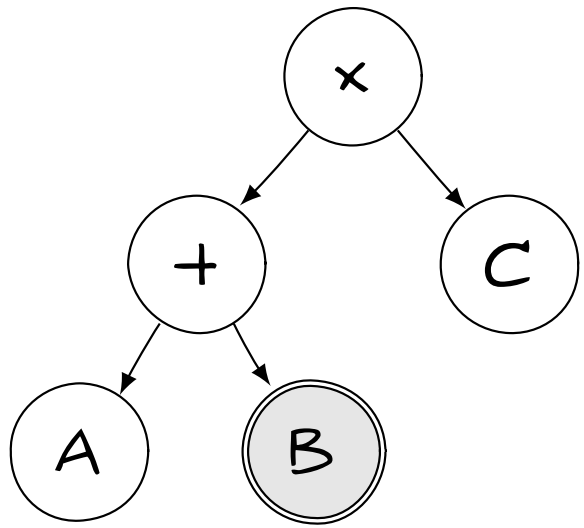
Árvore binária

- Percurso em ordem
  - Expressão $(A + B) \times C$
  - Regra *EPD*



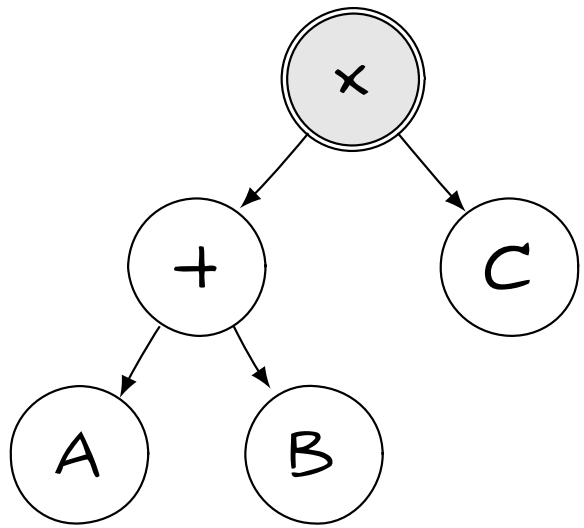
Árvore binária

- Percurso em ordem
  - Expressão $(A + B) \times C$
  - Regra *EPD*



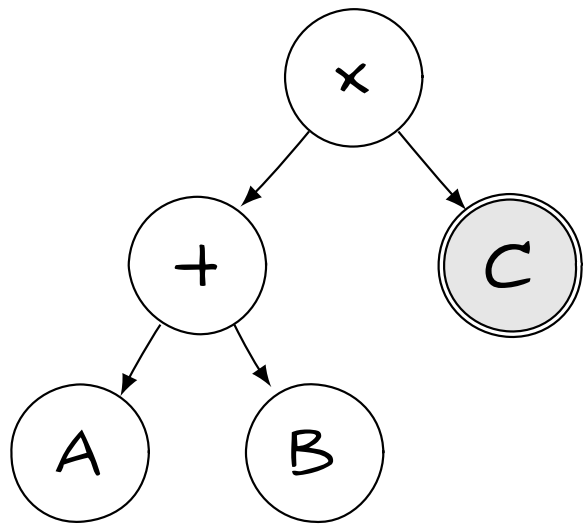
Árvore binária

- Percurso em ordem
  - Expressão $(A + B) \times C$
  - Regra *EPD*



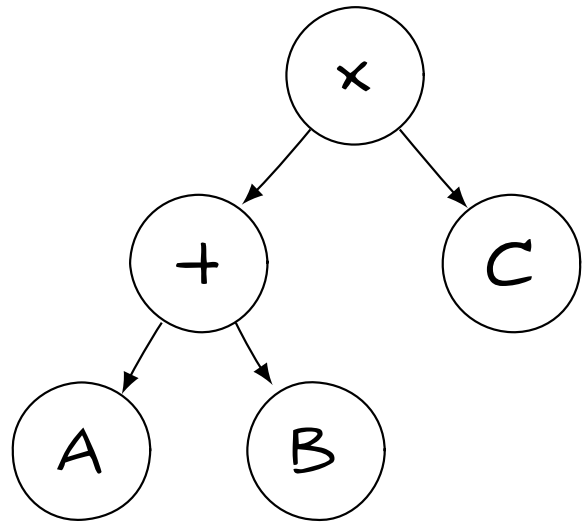
Árvore binária

- Percurso em ordem
  - Expressão $(A + B) \times C$
  - Regra *EPD*



Árvore binária

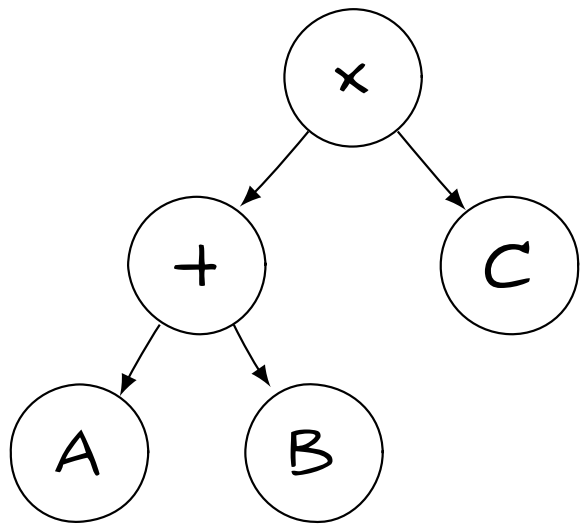
- Percurso em ordem
  - Expressão $(A + B) \times C$
  - Regra *EPD*



O percurso realizado foi $A + B \times C$

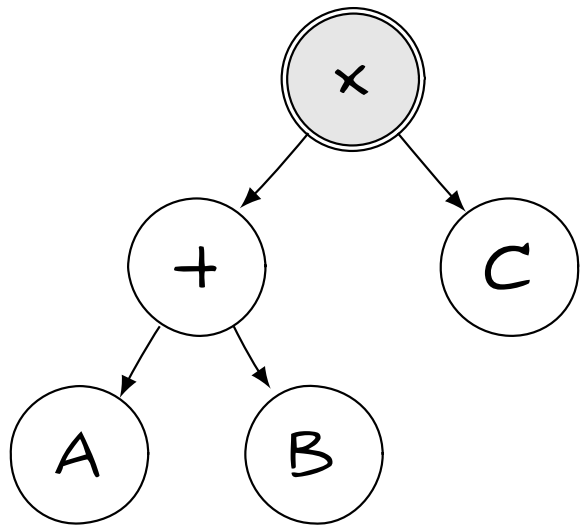
Árvore binária

- Percurso pré-ordem
  - Expressão $(A + B) \times C$
  - Regra *PED*



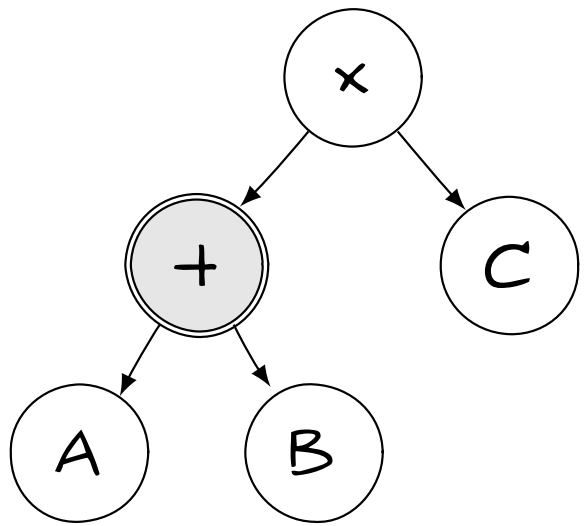
Árvore binária

- Percurso pré-ordem
  - Expressão $(A + B) \times C$
  - Regra *PED*



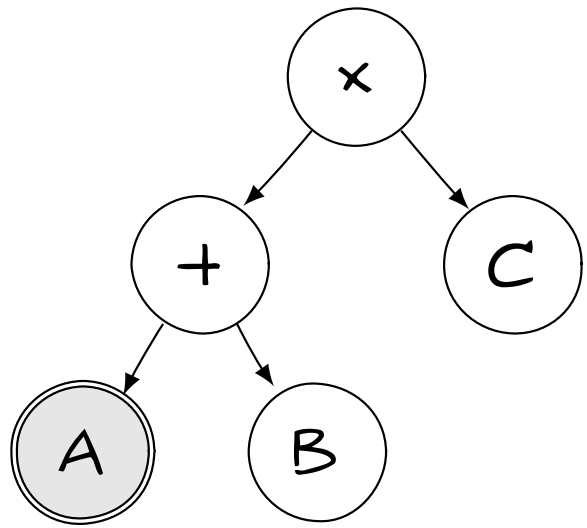
Árvore binária

- Percurso pré-ordem
  - Expressão $(A + B) \times C$
  - Regra *PED*



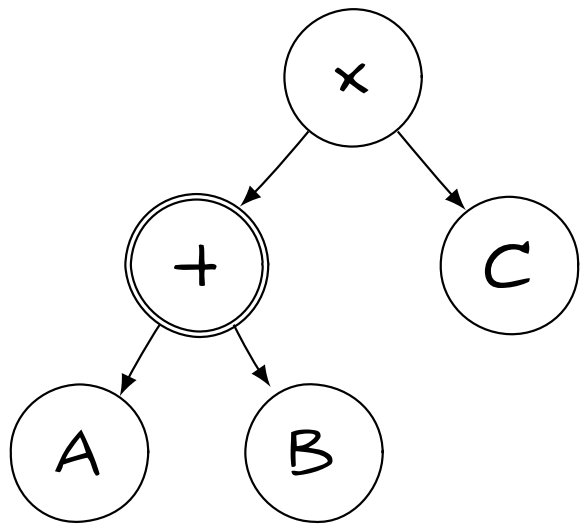
Árvore binária

- Percurso pré-ordem
  - Expressão $(A + B) \times C$
  - Regra *PED*



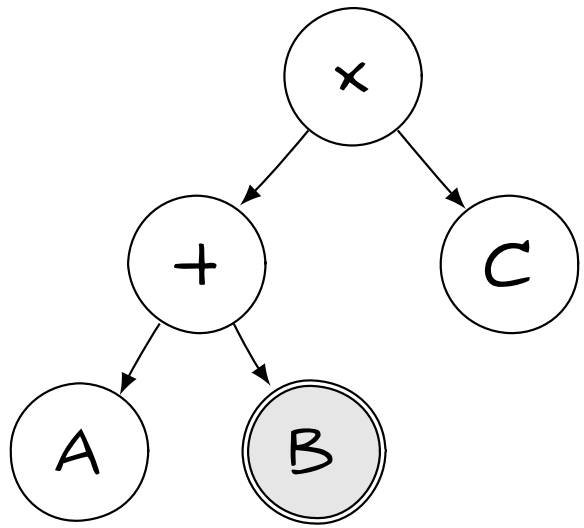
Árvore binária

- Percurso pré-ordem
  - Expressão $(A + B) \times C$
  - Regra *PED*



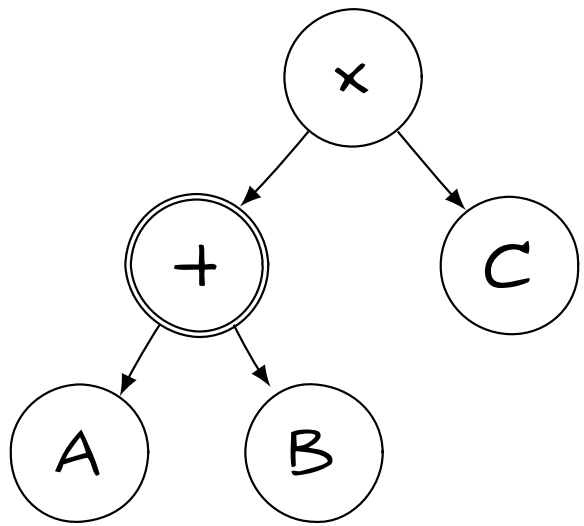
Árvore binária

- Percurso pré-ordem
  - Expressão $(A + B) \times C$
  - Regra *PED*



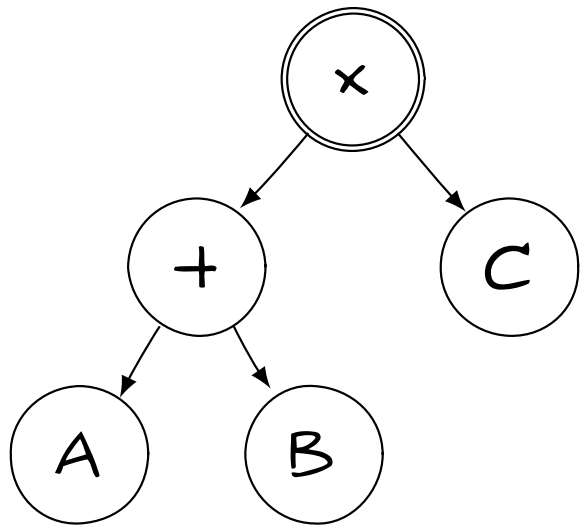
Árvore binária

- Percurso pré-ordem
  - Expressão $(A + B) \times C$
  - Regra *PED*



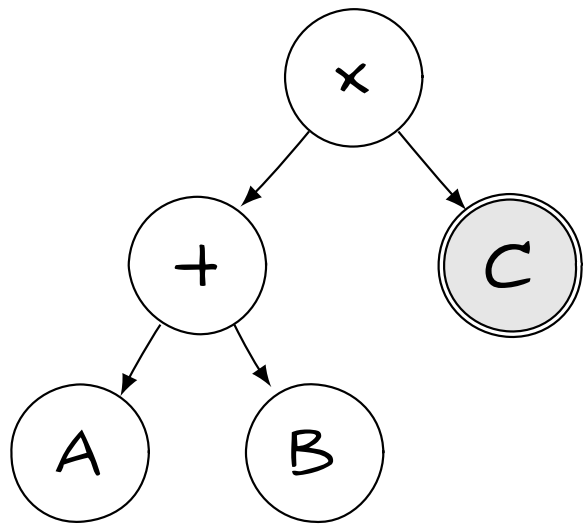
Árvore binária

- Percurso pré-ordem
  - Expressão $(A + B) \times C$
  - Regra *PED*



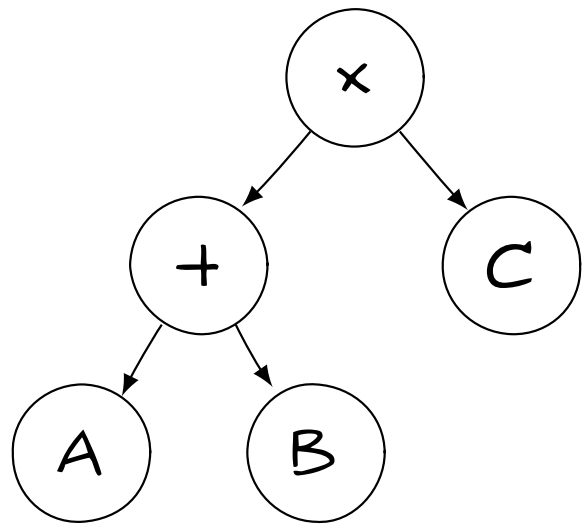
Árvore binária

- Percurso pré-ordem
  - Expressão $(A + B) \times C$
  - Regra *PED*



Árvore binária

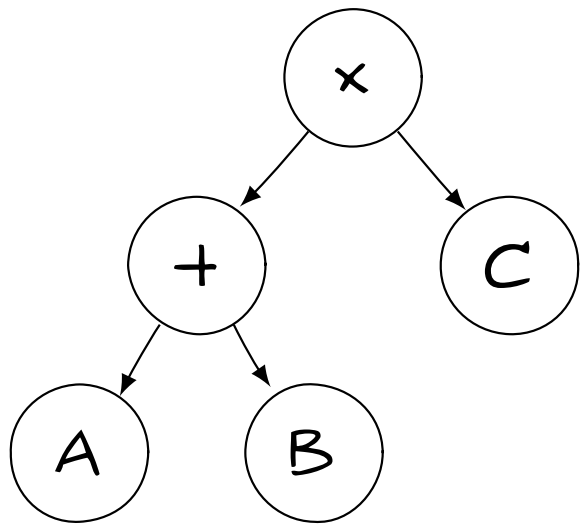
- Percurso pré-ordem
  - Expressão $(A + B) \times C$
  - Regra *PED*



O percurso realizado foi $\times + ABC$

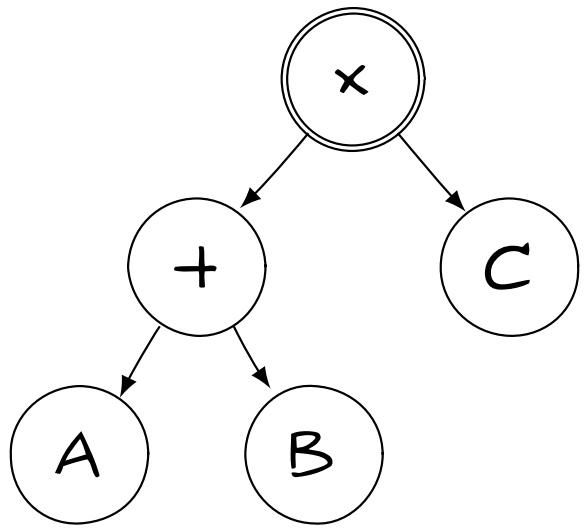
Árvore binária

- Percurso pós-ordem
  - Expressão $(A + B) \times C$
  - Regra *EDP*



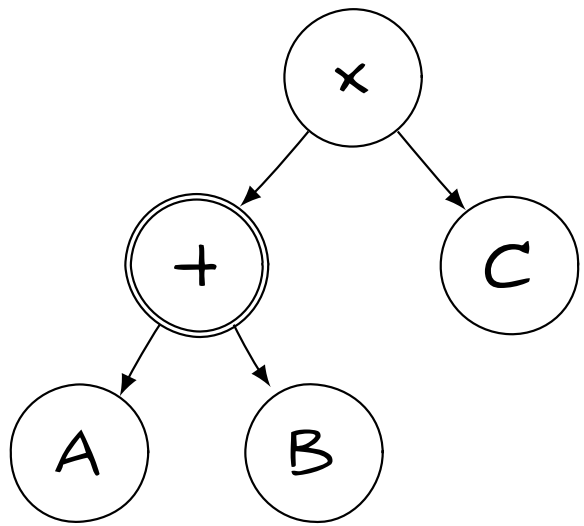
Árvore binária

- Percurso pós-ordem
  - Expressão $(A + B) \times C$
  - Regra *EDP*



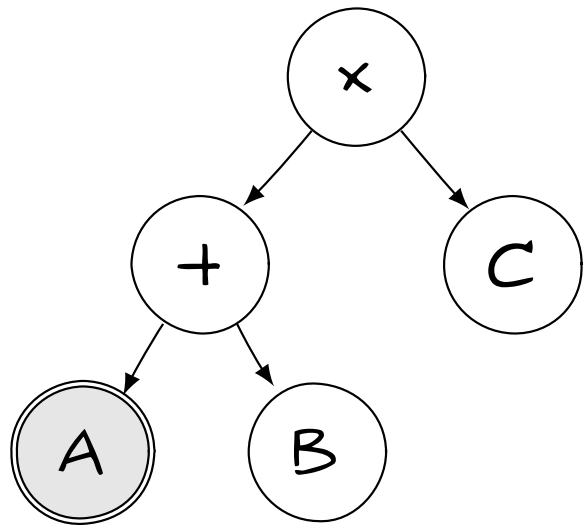
Árvore binária

- Percurso pós-ordem
  - Expressão $(A + B) \times C$
  - Regra *EDP*



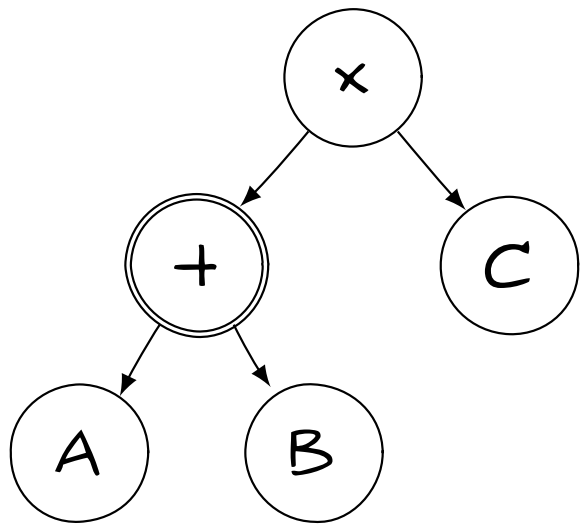
Árvore binária

- Percurso pós-ordem
  - Expressão $(A + B) \times C$
  - Regra *EDP*



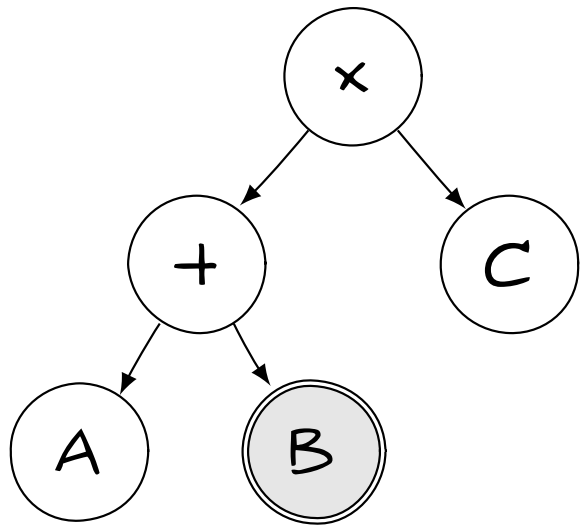
Árvore binária

- Percurso pós-ordem
  - Expressão $(A + B) \times C$
  - Regra *EDP*



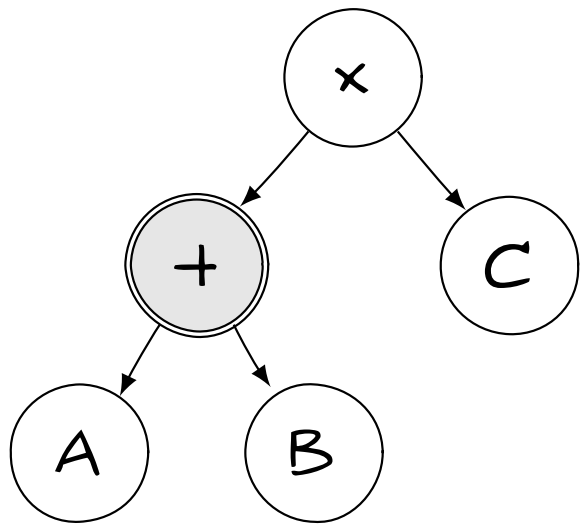
Árvore binária

- Percurso pós-ordem
  - Expressão $(A + B) \times C$
  - Regra *EDP*



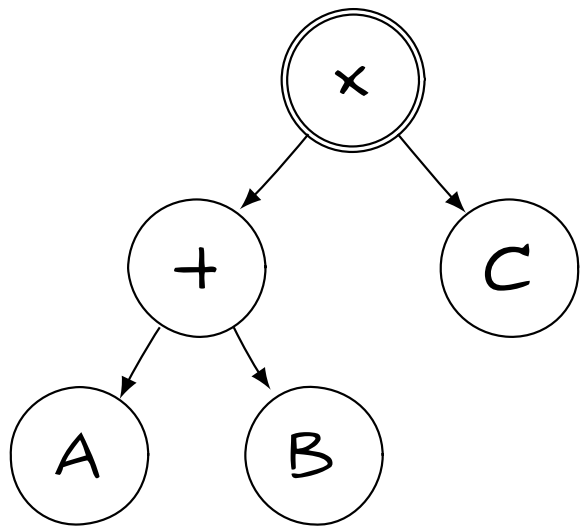
Árvore binária

- Percurso pós-ordem
  - Expressão $(A + B) \times C$
  - Regra *EDP*



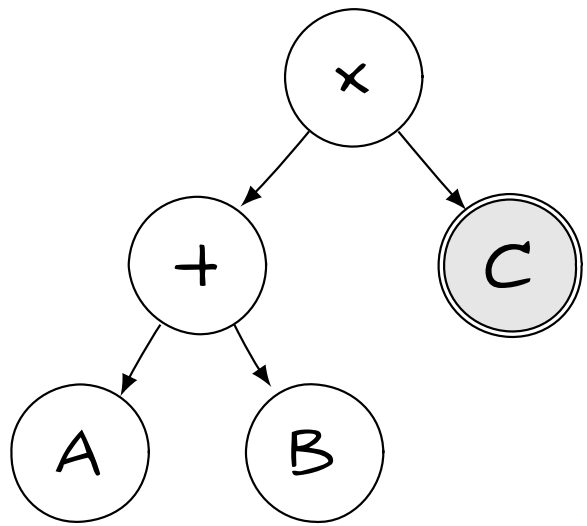
Árvore binária

- Percurso pós-ordem
  - Expressão $(A + B) \times C$
  - Regra *EDP*



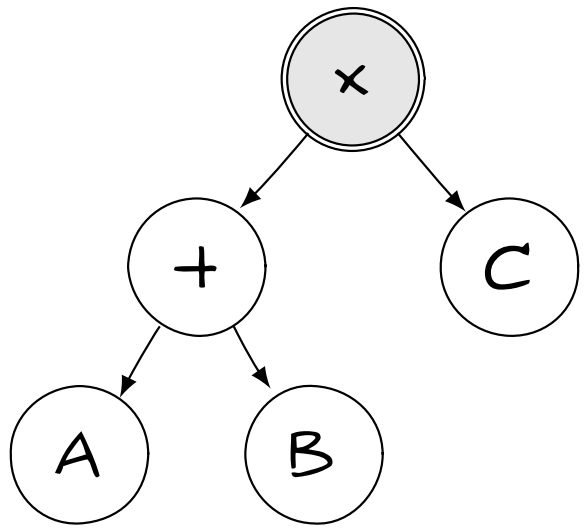
Árvore binária

- Percurso pós-ordem
  - Expressão $(A + B) \times C$
  - Regra *EDP*



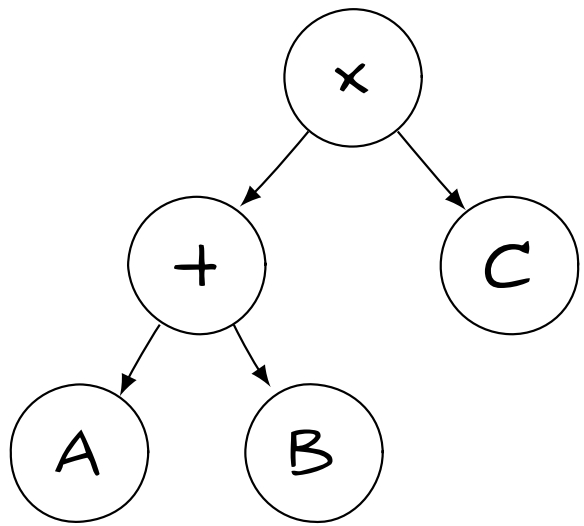
Árvore binária

- Percurso pós-ordem
  - Expressão $(A + B) \times C$
  - Regra *EDP*



Árvore binária

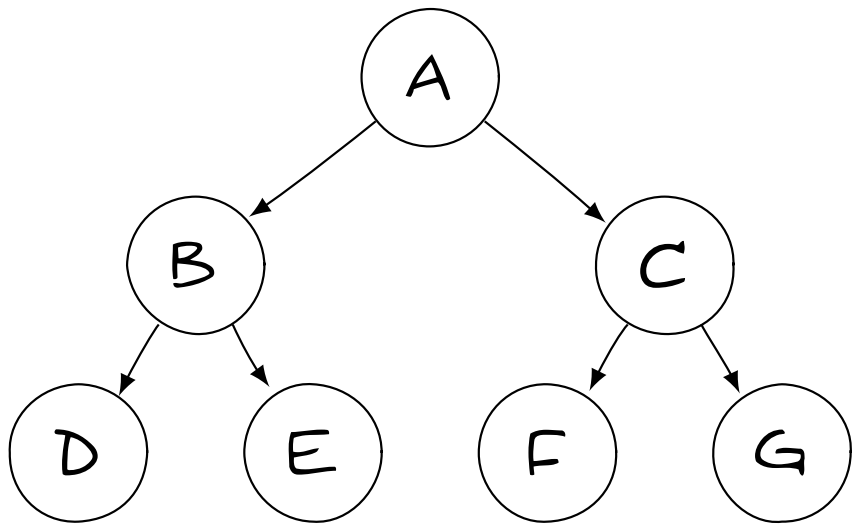
- Percurso pós-ordem
  - Expressão $(A + B) \times C$
  - Regra *EDP*



O percurso realizado foi $AB + C\times$

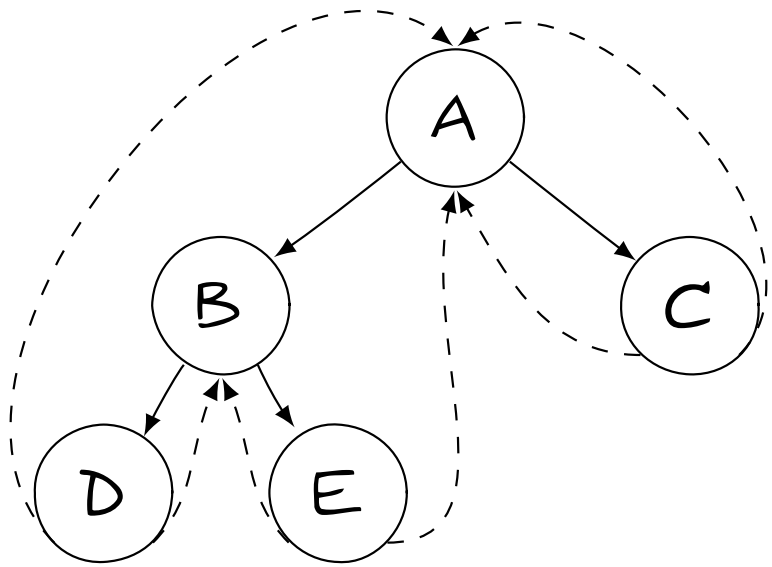
Exemplo

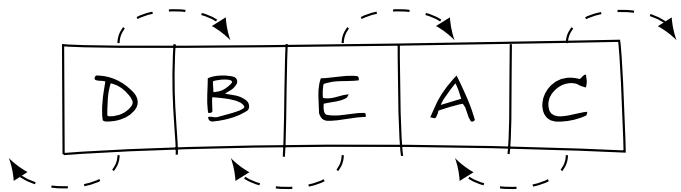
- Realize o percurso em ordem, pré-ordem e pós-ordem na árvore binária



Árvore binária

- Árvore binária costurada
  - Os ponteiros direito e esquerdo do nó folha referenciam seu sucessor e predecessor no percurso em ordem, respectivamente





Exemplo

- Defina os ponteiros de costura da árvore binária