Estrutura de Dados

Hamilton José Brumatto

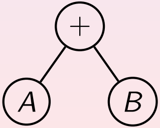
Bacharelado em Ciências da Computação - UESC

7 de fevereiro de 2024

Aplicações de Árvores Binárias

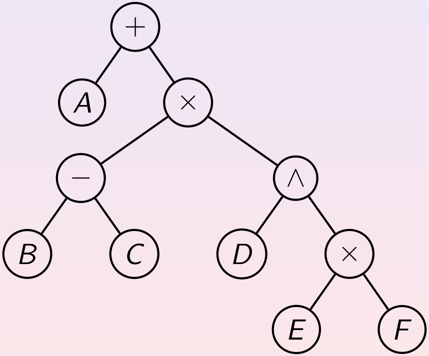
Árvores de Expressão

- Considere a seguinte expressão: $A + B$
- Esta expressão pode ser representada na forma de uma árvore binária:
  - O raiz é a operação.
  - O filho esquerdo é o operando esquerdo.
  - O filho direito é o operando direito.



Árvores de Expressão

- Uma expressão maior: $A + (B - C) \times D \wedge (E \times F)$



Aplicações

- A primeira aplicação direta é que a árvore serve para conversão de formas de notação:
  - Varredura em pré-ordem $\rightarrow$ expressão pré-fixa.
    $+A \times -BC \wedge D \times EF$
  - Varredura em pós-ordem $\rightarrow$ expressão pós-fixa.
    $ABC - DEF \times \wedge \times +$
  - Varredura em ordem simétria (desde que antes de visitar o filho esquerdo imprima-se um "Abre parêntesis", e após visitar o filho direito imprima-se um "Fecha parêntesis" $\rightarrow$ expressão infixa.
    $(A + ((B - C) \times (D \wedge (E \times F))))$

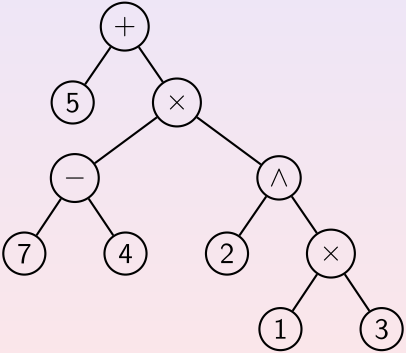
Aplicações: Algoritmo para calcular expressão

- A segunda aplicação é o cálculo do valor da expressão

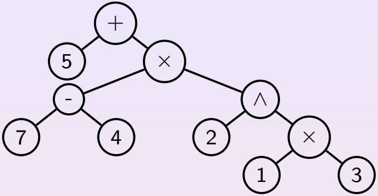
Algoritmo CALCULAR(Arvore a)
 $val \leftarrow 0$
 se $a \neq vazia$ **então**
  se $info(a) = operacao$ **então**
   $op1 = Calcular(filhoEsq(a))$
   $op2 = Calcular(filhoDir(a))$
   $val \leftarrow Operacao(op1, info(a), op2)$
  senão
   $val \leftarrow info(a)$
 retorne val

Cálculo da Expressão

- Vamos pegar por exemplo a árvore:

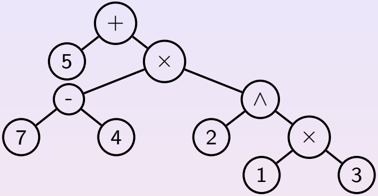


Cálculo da Expressão



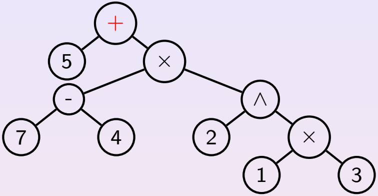
$val \leftarrow 0$ $op1:$ $op2:$ $val:$

Cálculo da Expressão



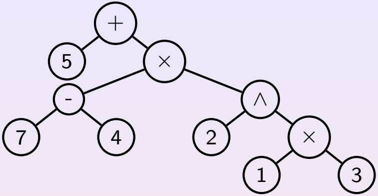
se $a \neq vazia$ **então** $op1:$ $op2:$ $val:0$

Cálculo da Expressão



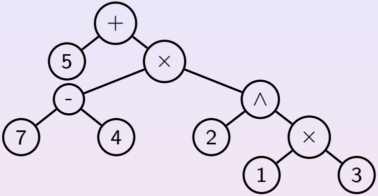
se $info(a) = operacao$ **então** $op1:$ $op2:$ $val: 0$

Cálculo da Expressão



$op1 = \text{Calcular}(\text{filhoEsq}(a))$ $op1 :$ $op2 :$ $val : 0$

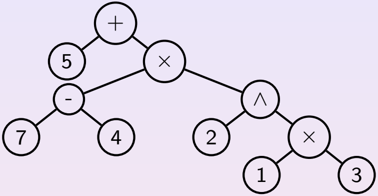
Cálculo da Expressão



$val \leftarrow 0$ $\quad op1: \quad op2: \quad val:$

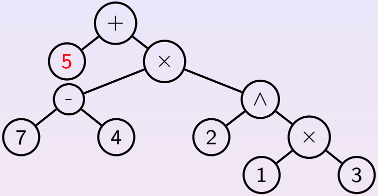
$op1 = \mathrm{Calcular}(\mathrm{filhoEsq}(a))$ $\quad op1: \quad op2: \quad val: 0$

Cálculo da Expressão



se $a \neq vazia$ **então**	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0

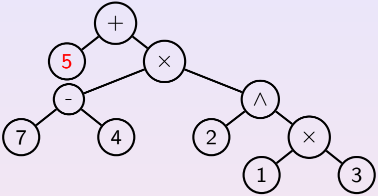
Cálculo da Expressão



se $\mathit{info}(a) = \mathit{operacao}$ **então** $\quad op1: \quad op2: \quad val: 0$

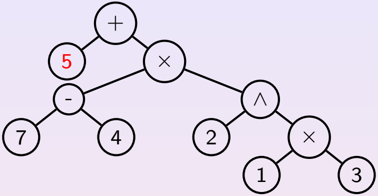
$op1 = \mathrm{Calcular}(\mathrm{filhoEsq}(a)) \quad op1: \quad op2: \quad val: 0$

Cálculo da Expressão



senão	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0

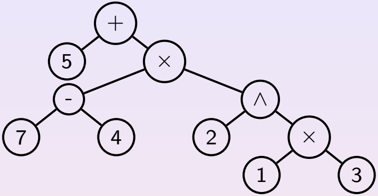
Cálculo da Expressão



$val \leftarrow \text{info}(a)$ — $op1:$ $op2:$ $val: 0$

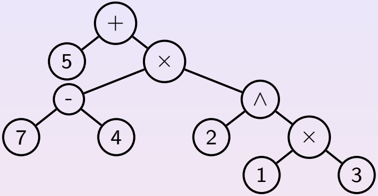
$op1 = \text{Calcular}(\text{filhoEsq}(a))$ — $op1:$ $op2:$ $val: 0$

Cálculo da Expressão



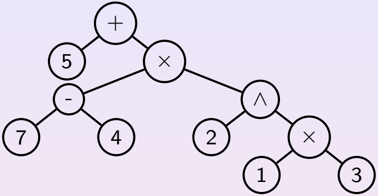
retorne *val*	$op1$:	$op2$:	val : 5
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0

Cálculo da Expressão



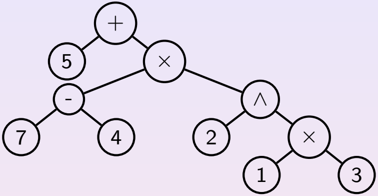
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$ $op1 : 5$ $op2 :$ $val : 0$

Cálculo da Expressão



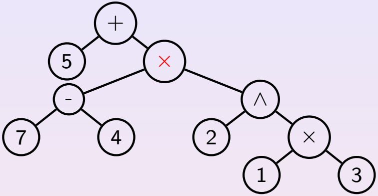
$val \leftarrow 0$	$op1$:	$op2$:	val :
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



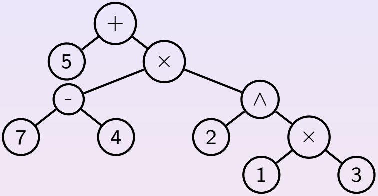
se *a* $\neq$ *vazia* **então**	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



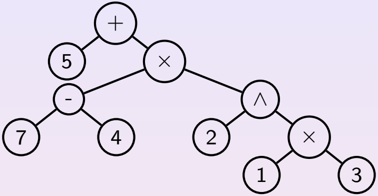
se $info(a) = operacao$ **então**	$op1$:	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



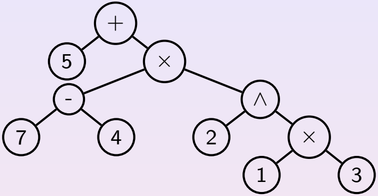
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



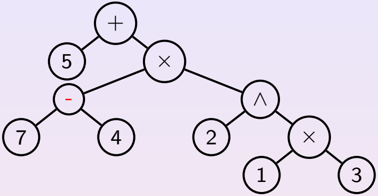
$val \leftarrow 0$	$op1$:	$op2$:	val :
$op1 = \mathrm{Calcular}(\mathrm{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



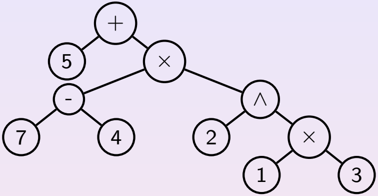
se $a \neq vazia$ **então**	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



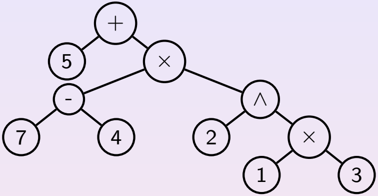
se $info(a) = operacao$ **então**	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



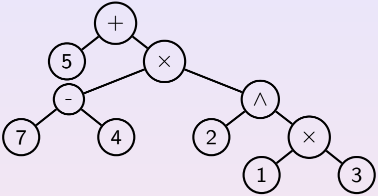
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1:$	$op2:$	$val: 0$
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1:$	$op2:$	$val: 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1: 5$	$op2:$	$val: 0$

Cálculo da Expressão



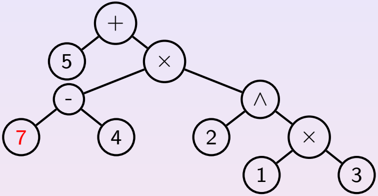
$val \leftarrow 0$	$op1$:	$op2$:	val :
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



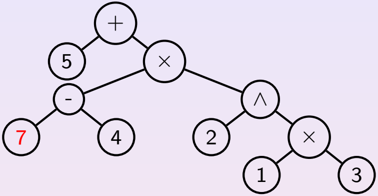
se *a* $\neq$ *vazia* **então**	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



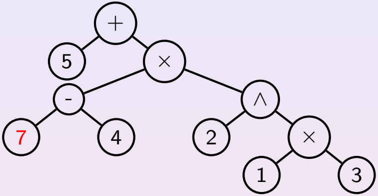
se *info*(a) = *operacao* **então**	$op1$:	$op2$:	val : 0
$op1 = \mathrm{Calcular}(\mathrm{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op1 = \mathrm{Calcular}(\mathrm{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



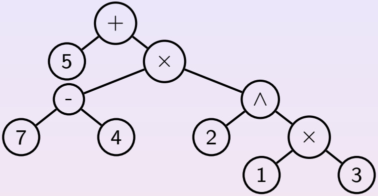
senão	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



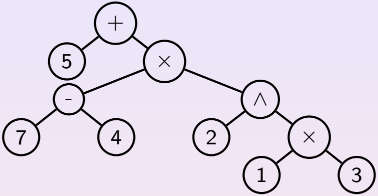
$val \leftarrow \text{info}(a)$	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



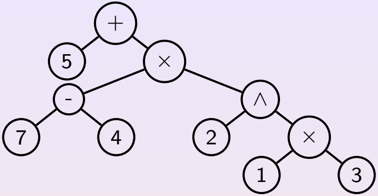
retorne *val*	$op1$:	$op2$:	val : 7
$op1 = \mathrm{Calcular}(\mathrm{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op1 = \mathrm{Calcular}(\mathrm{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



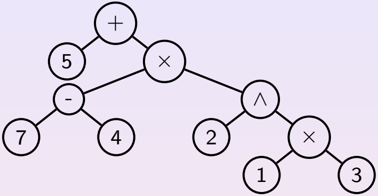
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 7$	$op2 :$	$val : 0$
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1 :$	$op2 :$	$val : 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 5$	$op2 :$	$val : 0$

Cálculo da Expressão



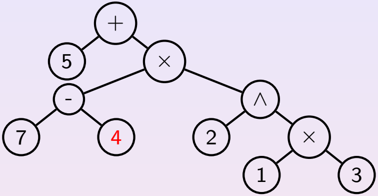
$val \leftarrow 0$	$op1$:	$op2$:	val :
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 7	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



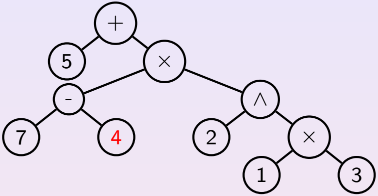
se $a \neq vazia$ **então**	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 7	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



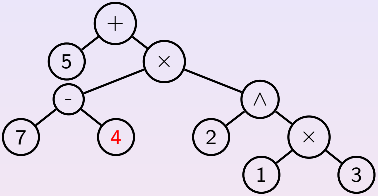
se $info(a) = operacao$ **então**	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 7	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



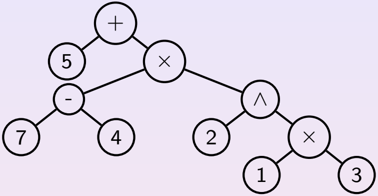
senão	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 7	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



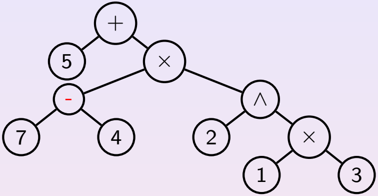
$val \leftarrow \mathrm{info}(a)$	$op1$:	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 7	$op2$:	val : 0
$op1 = \mathrm{Calcular}(\mathrm{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



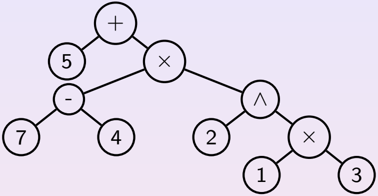
retorne *val*	$op1$:	$op2$:	val : 4
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 7	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



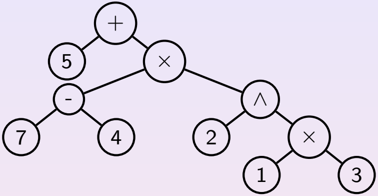
$val \leftarrow \text{Operacao}(op1, \text{info}(a), op2)$	$op1 : 7$	$op2 : 4$	$val : 0$
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1 :$	$op2 :$	$val : 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 5$	$op2 :$	$val : 0$

Cálculo da Expressão



retorne *val*	$op1 : 7$	$op2 : 4$	$val : 3$
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1 :$	$op2 :$	$val : 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 5$	$op2 :$	$val : 0$

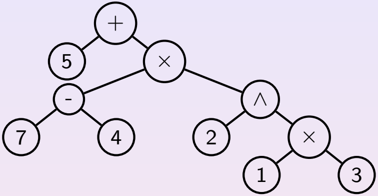
Cálculo da Expressão



$op2 = \text{Calcular}(\text{filhoDir}(a))$ $op1 : 3$ $op2 :$ $val : 0$

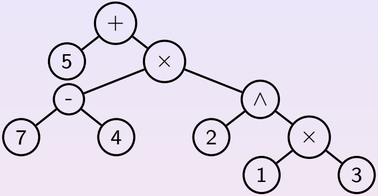
$op2 = \text{Calcular}(\text{filhoDir}(a))$ $op1 : 5$ $op2 :$ $val : 0$

Cálculo da Expressão



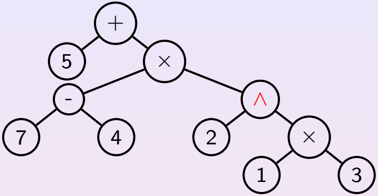
$val \leftarrow 0$	$op1$:	$op2$:	val :
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



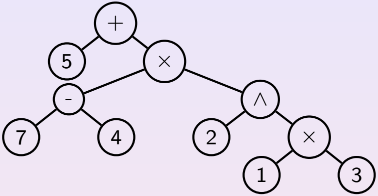
se $a \neq$ *vazia* **então**	$op1$:	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



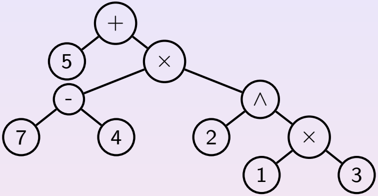
se $info(a) = operacao$ **então**	$op1$:	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



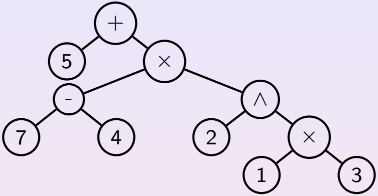
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



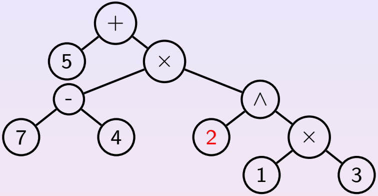
$val \leftarrow 0$	$op1$:	$op2$:	val :
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



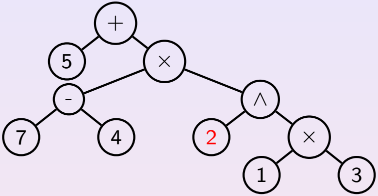
se $a \neq vazia$ **então**	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



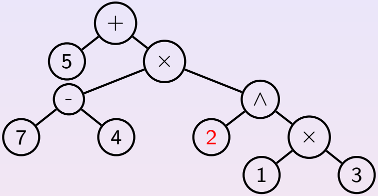
se $info(a) = operacao$ **então**	$op1$:	$op2$:	val : 0
$op1 = \mathrm{Calcular}(\mathrm{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



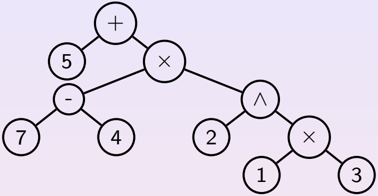
senão	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



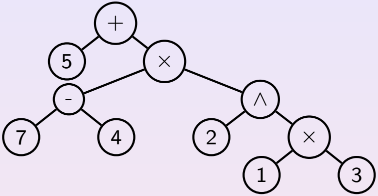
$val \leftarrow \mathrm{info}(a)$	$op1$:	$op2$:	val : 0
$op1 = \mathrm{Calcular}(\mathrm{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



retorne *val*	$op1$:	$op2$:	val : 2
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão

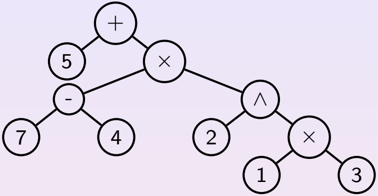


$op2 = \text{Calcular}(\text{filhoDir}(a))$ $op1: 2$ $op2:$ $val: 0$

$op2 = \text{Calcular}(\text{filhoDir}(a))$ $op1: 3$ $op2:$ $val: 0$

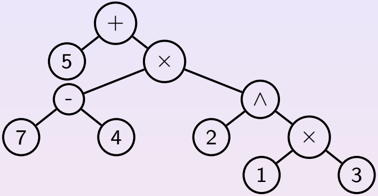
$op2 = \text{Calcular}(\text{filhoDir}(a))$ $op1: 5$ $op2:$ $val: 0$

Cálculo da Expressão



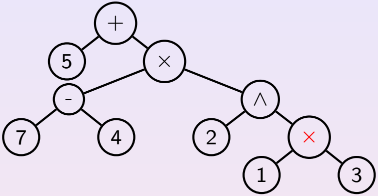
$val \leftarrow 0$	$op1$:	$op2$:	val :
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 2$	$op2$:	$val : 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 3$	$op2$:	$val : 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 5$	$op2$:	$val : 0$

Cálculo da Expressão



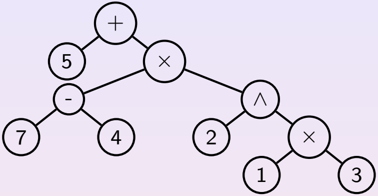
se $a \neq vazia$ **então**	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



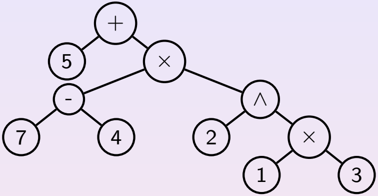
se $info(a) = operacao$ **então**	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



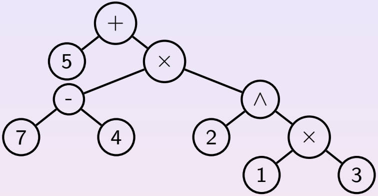
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1:$	$op2:$	$val: 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1: 2$	$op2:$	$val: 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1: 3$	$op2:$	$val: 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1: 5$	$op2:$	$val: 0$

Cálculo da Expressão



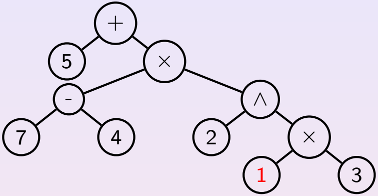
$val \leftarrow 0$	$op1$:	$op2$:	val :
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



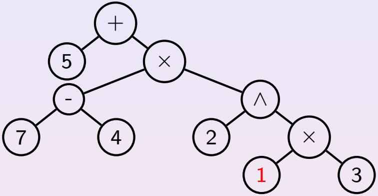
se $a \neq vazia$ **então**	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



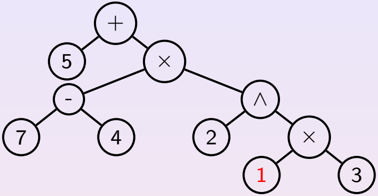
se $info(a) = operacao$ **então**	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



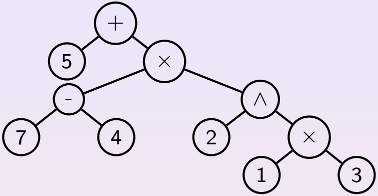
senão	$op1$:	$op2$:	val : 0
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



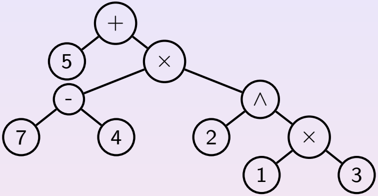
$val \leftarrow \mathrm{info}(a)$	$op1$:	$op2$:	val : 0
$op1 = \mathrm{Calcular}(\mathrm{filhoEsq}(a))$	$op1$:	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



retorne *val*	$op1:$	$op2:$	$val: 1$
$op1 = \text{Calcular}(\text{filhoEsq}(a))$	$op1:$	$op2:$	$val: 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1: 2$	$op2:$	$val: 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1: 3$	$op2:$	$val: 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1: 5$	$op2:$	$val: 0$

Cálculo da Expressão



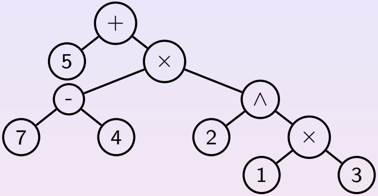
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$ $op1 : 1$ $op2 :$ $val : 0$

$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$ $op1 : 2$ $op2 :$ $val : 0$

$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$ $op1 : 3$ $op2 :$ $val : 0$

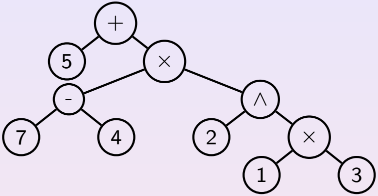
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$ $op1 : 5$ $op2 :$ $val : 0$

Cálculo da Expressão



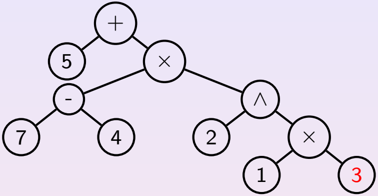
$val \leftarrow 0$	$op1$:	$op2$:	val :
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 1	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



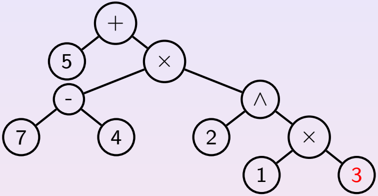
se $a \neq vazia$ **então**	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 1	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



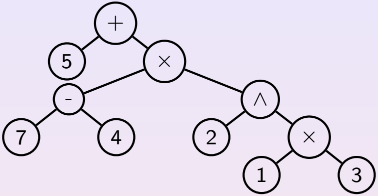
se $info(a) = operacao$ **então**	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 1	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



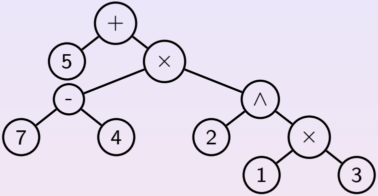
senão	$op1$:	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 1	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



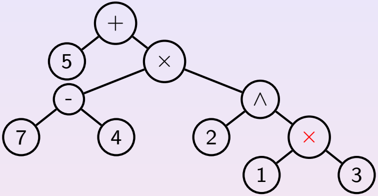
$val \leftarrow \mathrm{info}(a)$	$op1$:	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 1	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



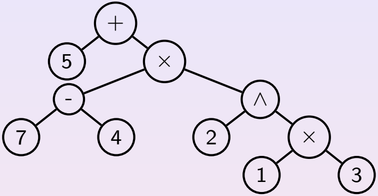
retorne *val*	$op1$:	$op2$:	val : 3
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 1	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 2	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 3	$op2$:	val : 0
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1$: 5	$op2$:	val : 0

Cálculo da Expressão



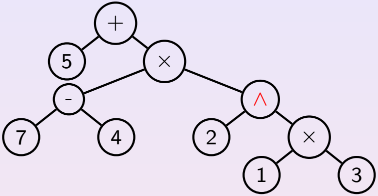
$val \leftarrow \text{Operacao}(op1, \text{info}(a), op2)$	$op1 : 1$	$op2 : 3$	$val : 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 2$	$op2 :$	$val : 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 3$	$op2 :$	$val : 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 5$	$op2 :$	$val : 0$

Cálculo da Expressão



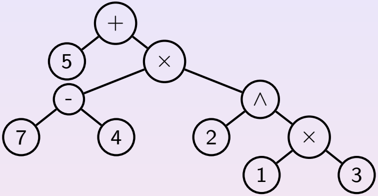
retorne *val*	$op1 : 1$	$op2 : 3$	$val : 3$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 2$	$op2 :$	$val : 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 3$	$op2 :$	$val : 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 5$	$op2 :$	$val : 0$

Cálculo da Expressão



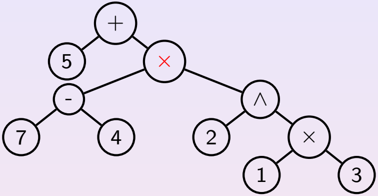
$val \leftarrow \mathrm{Operacao}(op1, \mathrm{info}(a), op2)$	$op1 : 2$	$op2 : 3$	$val : 0$
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1 : 3$	$op2 :$	$val : 0$
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1 : 5$	$op2 :$	$val : 0$

Cálculo da Expressão



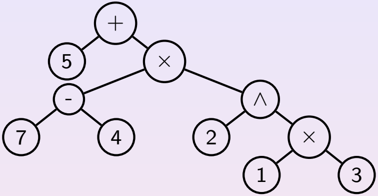
retorne *val*	$op1 : 2$	$op2 : 3$	$val : 8$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 3$	$op2 :$	$val : 0$
$op2 = \text{Calcular}(\text{filhoDir}(a))$	$op1 : 5$	$op2 :$	$val : 0$

Cálculo da Expressão



$val \leftarrow \mathrm{Operacao}(op1, \mathrm{info}(a), op2)$	$op1 : 3$	$op2 : 8$	$val : 0$
$op2 = \mathrm{Calcular}(\mathrm{filhoDir}(a))$	$op1 : 5$	$op2 :$	$val : 0$

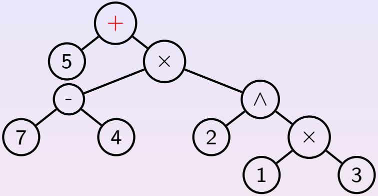
Cálculo da Expressão



retorne *val* | $op1 : 3$ $op2 : 8$ $val : 24$

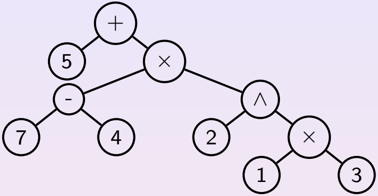
$op2 = \text{Calcular}(\text{filhoDir}(a))$ | $op1 : 5$ $op2 :$ $val : 0$

Cálculo da Expressão



$val \leftarrow \mathrm{Operacao}(op1, \mathrm{info}(a), op2)$ $op1 : 5$ $op2 : 24$ $val : 0$

Cálculo da Expressão



retorne *val*

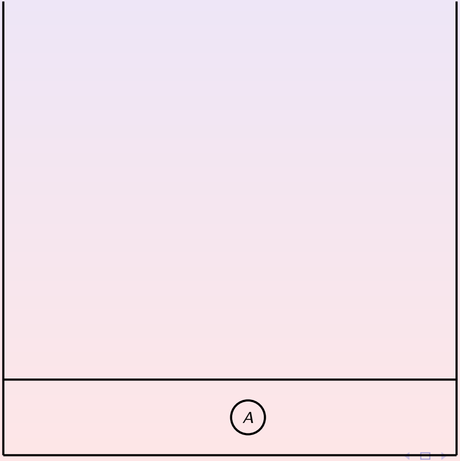
$op1 : 5$ $op2 : 24$ $val : 29$

Construção - Pós Fixa

- A partir de uma expressão pós-fixa $\rightarrow$
- Se o item lido é um operando, crie uma árvore folha com o operando como raiz e empilhe.
- Se o item lido é um operador:
  - crie uma árvore binária com o operador como raiz.
  - retire um operando da pilha e insira-o como filho direito.
  - retire um operando da pilha e insira-o como filho esquerdo.
  - empilhe a árvore resultante.
- No final do processo, resta na pilha a árvore de expressão.

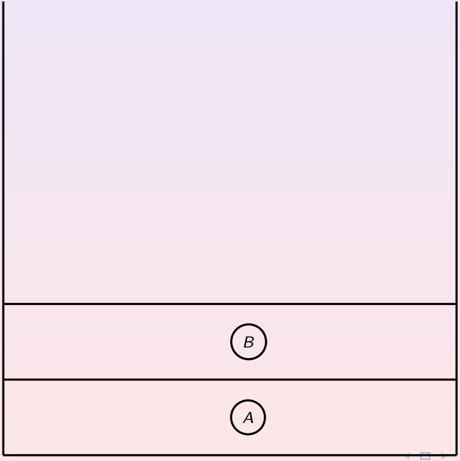
Construção - Pós Fixa

$ABC - DEF \times \wedge \times +$



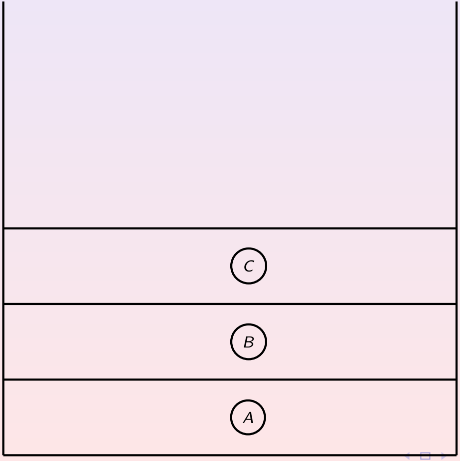
Construção - Pós Fixa

$ABC-DEF\times\wedge\times+$



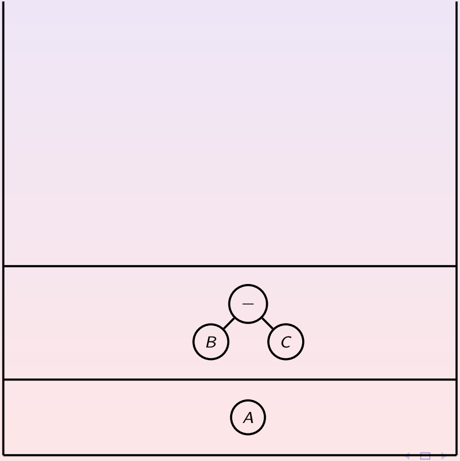
Construção - Pós Fixa

$ABC-DEF\times\wedge\times+$



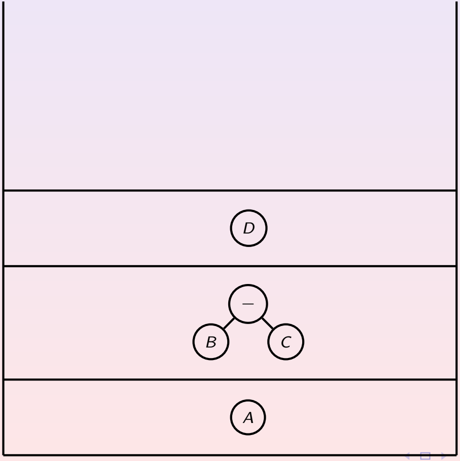
Construção - Pós Fixa

$ABC - DEF \times \wedge \times +$



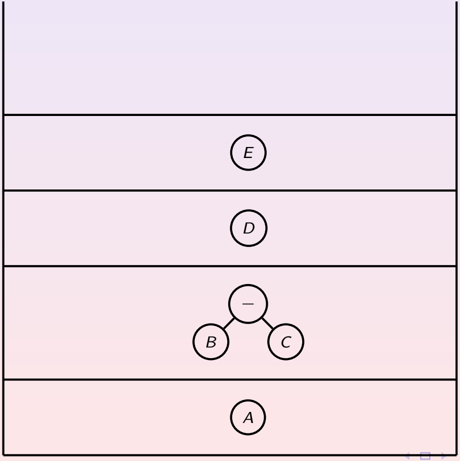
Construção - Pós Fixa

$ABC - DEF \times \wedge \times +$



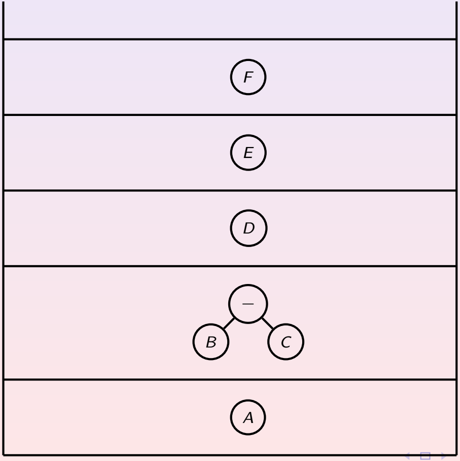
Construção - Pós Fixa

$ABC-DEF\times\wedge\times+$



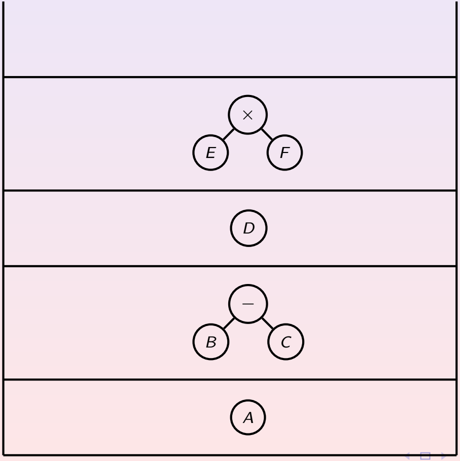
Construção - Pós Fixa

$ABC-DE\textcolor{blue}{F}\times\wedge\times+$



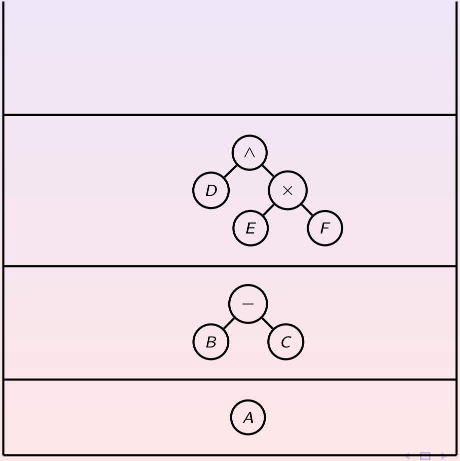
Construção - Pós Fixa

$ABC-DEF{\color{blue}\times}\wedge\times+$



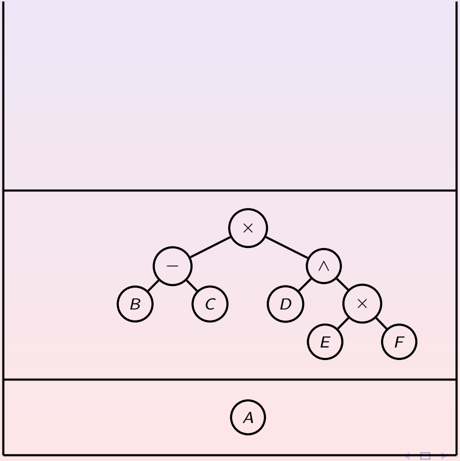
Construção - Pós Fixa

$ABC-DEF\times\wedge\times+$



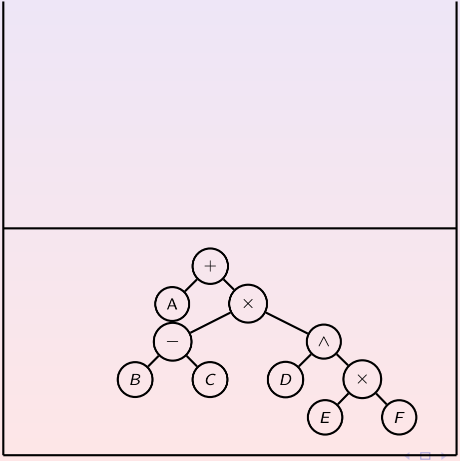
Construção - Pós Fixa

$ABC-DEF\times\wedge\times+$



Construção - Pós Fixa

$ABC-DEF\times\wedge\times+$



Construção - Infixa

- A partir de uma expressão infixa: É necessária uma total parentisação para evitar que a ordem de precedência de operandos seja determinante na ordem das operações:
- A expressão: $A + (B - C) \times D \wedge (E \times F)$ deve ser escrita como:
  $(A + ((B - C) \times (D \wedge (E \times F))))$
- O algoritmo irá ler: $operando1$, $operacao$, $operando2$ e construir uma árvore de expressão com estes três elementos.
- Ao encontrar um abre-parêntesis chama-se recursivamente o algoritmo.
- A última leitura é um fecha-parêntesis, quando retorna a árvore ao chamador.
- Na leitura de um operador, se não houver símbolo, o $operando1$ é a árvore de expressão.

Construção - Infixa

Algoritmo ConstruirInfixa
    $S \leftarrow$ lerSimbolo()
    se $S =$ '(' **então**
        $op1 \leftarrow$ ConstruirInfixa()
    senão
        $op1 \leftarrow$ NovaArvore(S)
    $S \leftarrow$ lerSimbolo()
    se $S = \emptyset$ **então retorne** $op1$
    senão
        $op \leftarrow$ NovaArvore(S)
        $S \leftarrow$ lerSimbolo()
        se $S =$ '(' **então**
            $op2 \leftarrow$ ConstruirInfixa()
        senão
            $op2 \leftarrow$ NovaArvore(S)
        $op \rightarrow$InsereFilhoEsq($op1$)
        $op \rightarrow$InsereFilhoDir($op2$)
        lerSimbolo() **retorne** op

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$S \leftarrow$ lerSimbolo() : $S =$ '(' $op1$: op : $op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S = '(') \rightarrow$ **então** $op1:$ $op:$ $op2:$

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op1 \leftarrow$ ConstruirInfixa() $op1:$ $op:$ $op2:$

Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

$S \leftarrow$ lerSimbolo() : $S = A$	$op1$:	op :	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S = '(') \rightarrow$ **senão**	$op1$:	op :	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op1 \leftarrow$ NovaArvore(S)	$op1$: A	op :	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$S \leftarrow$ lerSimbolo() : $S = +$	$op1$: A	op :	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

se$(S=\emptyset)\rightarrow$ **senão**	$op1$: A	op :	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op \leftarrow$ NovaArvore(S)	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$S \leftarrow$ lerSimbolo() : $S =$'('	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S = '(') \rightarrow$ **então**	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$S \leftarrow$ lerSimbolo() : $S =$'('	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S = '(') \rightarrow$ **então**	$op1:$	$op:$	$op2:$
$op2 \leftarrow$ ConstruirInfixa()	$op1:$ A	$op:$ $+$	$op2:$
$op1 \leftarrow$ ConstruirInfixa()	$op1:$	$op:$	$op2:$

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$S \leftarrow$ lerSimbolo() : $S = B$	$op1$:	op :	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S = '(')$ $\rightarrow$ **senão**	$op1$:	op :	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op1 \leftarrow \text{NovaArvore}(S)$	$op1$: B	op :	$op2$:
$op1 \leftarrow \text{ConstruirInfixa}()$	$op1$:	op :	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow \text{ConstruirInfixa}()$	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$S \leftarrow$ lerSimbolo() : $S = -$	$op1$: B	op :	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S=\emptyset)\rightarrow$ **senão**	$op1$: B	op :	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op \leftarrow$ NovaArvore(S)	$op1$: B	op : $-$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$S \leftarrow$ lerSimbolo() : $S = C$	$op1$: B	op : $-$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S=\text{'('})\rightarrow$ **senão**	$op1$: B	op : $-$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

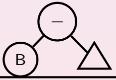
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op2 \leftarrow \text{NovaArvore}(S)$	$op1$: B	op : $-$	$op2$: C
$op1 \leftarrow \text{ConstruirInfixa}()$	$op1$:	op :	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow \text{ConstruirInfixa}()$	$op1$:	op :	$op2$:

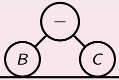
Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

$op \rightarrow$InsereFilhoEsq$(op1)$	$op1$: B	op : 	$op2$: C
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

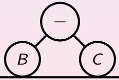
Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

$op \rightarrow$InsereFilhoDir($op2$)	$op1$: B	op : 	$op2$: C
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

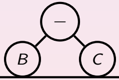
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

lerSimbolo()	$op1$: B	op : 	$op2$: C
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

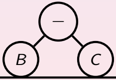
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

retorne op	$op1$: B	op : 	$op2$: C
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

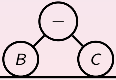
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op1 \leftarrow$ ConstruirInfixa()	$op1$: 	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

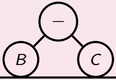
Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

$S \leftarrow$ lerSimbolo(): $S = \times$	$op1$: 	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

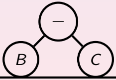
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S=\emptyset)\rightarrow$ **senão**	$op1$: 	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

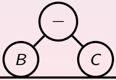
Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

$op \leftarrow$ NovaArvore(S)	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

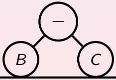
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$S \leftarrow$ lerSimbolo() : $S =$'('	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

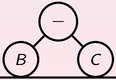
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S = \text{'('}) \rightarrow$ **então**	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

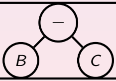
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

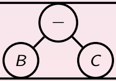
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$S \leftarrow$ lerSimbolo() : $S = D$	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

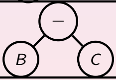
Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

se$(S = '(') \rightarrow$ **senão**	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

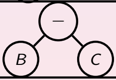
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op1 \leftarrow$ NovaArvore(S)	$op1$: D	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

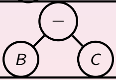
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$S \leftarrow$ lerSimbolo() : $S = \wedge$	$op1$: D	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

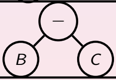
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S=\emptyset)\rightarrow$ **senão**	$op1$: D	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

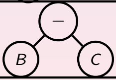
$(A+((B-C)\times(D\wedge(E\times F))))$

$op \leftarrow \text{NovaArvore}(S)$	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow \text{ConstruirInfixa}()$	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$S \leftarrow$ lerSimbolo() : $S =$'(' $op1$: D op : $\wedge$ $op2$:

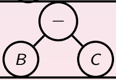
$op2 \leftarrow$ ConstruirInfixa() $op1$:  op : $\times$ $op2$:

$op2 \leftarrow$ ConstruirInfixa() $op1$: A op : $+$ $op2$:

$op1 \leftarrow$ ConstruirInfixa() $op1$: op : $op2$:

Construção - Infixa

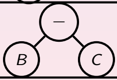
$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S =$ '(' $)\rightarrow$ **então**	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op2 \leftarrow$ ConstruirInfixa() $op1 :$ D $op :$ $\wedge$ $op2 :$

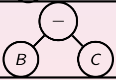
$op2 \leftarrow$ ConstruirInfixa() $op1 :$  $op :$ $\times$ $op2 :$

$op2 \leftarrow$ ConstruirInfixa() $op1 :$ A $op :$ $+$ $op2 :$

$op1 \leftarrow$ ConstruirInfixa() $op1 :$ $op :$ $op2 :$

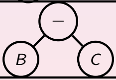
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$S \leftarrow$ lerSimbolo() : $S = E$	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

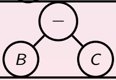
Construção - Infixa

$$(A+((B-C)\times(D\wedge(\color{blue}{E}\color{black}\times F))))$$

se$(S = \text{'('}) \rightarrow$ **senão**	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op1 \leftarrow \text{NovaArvore}(S)$	$op1$: E	op :	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow \text{ConstruirInfixa}()$	$op1$:	op :	$op2$:

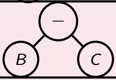
Construção - Infixa

$$(A+((B-C)\times(D\wedge(E{\color{blue}\times}F))))$$

$S \leftarrow$ lerSimbolo() : $S = \times$	$op1$: E	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: B $-$ C	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

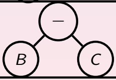
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S=\emptyset)\rightarrow$ **senão**	$op1$: E	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op \leftarrow \text{NovaArvore}(S)$	$op1$: E	op : $\times$	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow \text{ConstruirInfixa}()$	$op1$:	op :	$op2$:

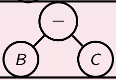
Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

$S \leftarrow$ lerSimbolo() : $S = F$	$op1$: E	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: $B - C$	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

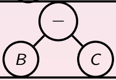
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

se$(S = '(')\rightarrow$ **senão**	$op1$: E	op : $\times$	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow \text{ConstruirInfixa}()$	$op1$:	op :	$op2$:

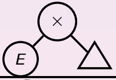
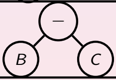
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op2 \leftarrow \text{NovaArvore}(S)$	$op1$: E	op : $\times$	$op2$: F
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow \text{ConstruirInfixa}()$	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow \text{ConstruirInfixa}()$	$op1$:	op :	$op2$:

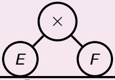
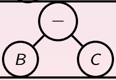
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

op $\rightarrow$InsereFilhoEsq($op1$)	$op1$: E	op : 	$op2$: F
$op2 \leftarrow$ ConstruirInfixa()	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

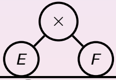
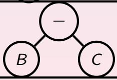
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

op →InsereFilhoDir($op2$)	$op1$: E	op : 	$op2$: F
$op2 \leftarrow$ ConstruirInfixa()	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

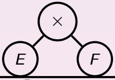
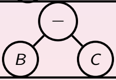
Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

lerSimbolo()	$op1$: E	op : 	$op2$: F
$op2 \leftarrow$ ConstruirInfixa()	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

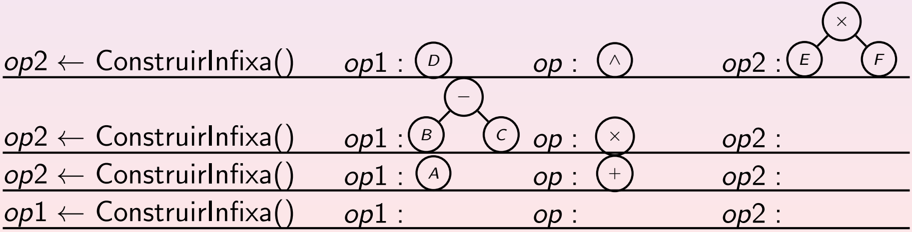
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

retorne *op*	$op1$: E	op : 	$op2$: F
$op2 \leftarrow$ ConstruirInfixa()	$op1$: D	op : $\wedge$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

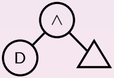
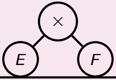
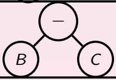
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$



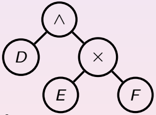
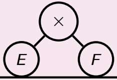
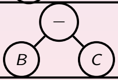
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op \rightarrow$InsereFilhoEsq($op1$)	$op1$: D	op : 	$op2$: 
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

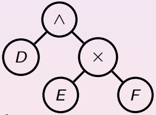
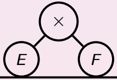
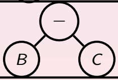
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

op →InsereFilhoDir($op2$)	$op1$: D	op : 	$op2$: 
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

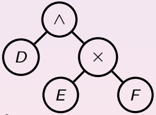
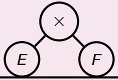
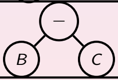
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

lerSimbolo()	$op1$: D	op : 	$op2$: 
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

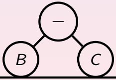
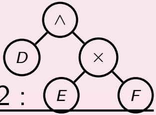
Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

retorne *op*	$op1$: D	op : 	$op2$: 
$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

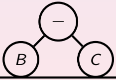
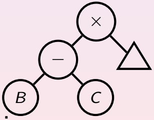
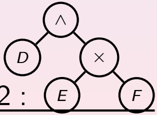
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op2 \leftarrow$ ConstruirInfixa()	$op1$: 	op : $\times$	$op2$: 
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

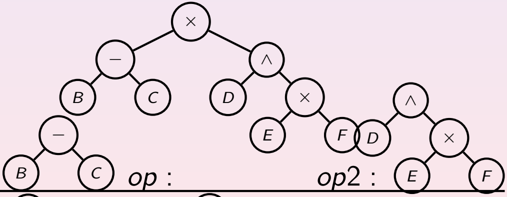
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

op →InsereFilhoEsq($op1$)	$op1$: 	op : 	$op2$: 
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

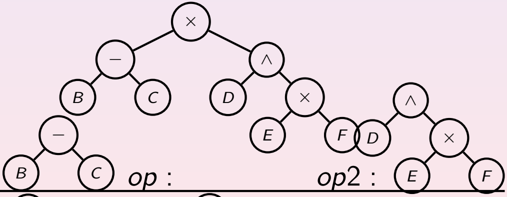
$(A+((B-C)\times(D\wedge(E\times F))))$



op $\rightarrow$InsereFilhoDir($op2$)	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

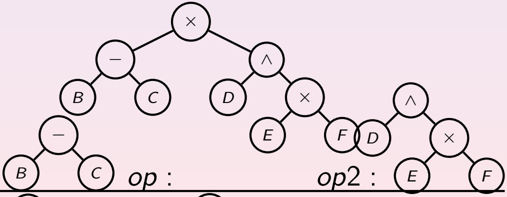
$(A+((B-C)\times(D\wedge(E\times F))))$



lerSimbolo()	$op1$:	op :	$op2$:
$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	$op2$:
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

Construção - Infixa

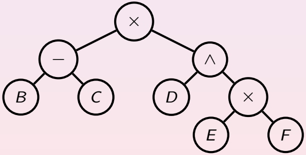
$(A+((B-C)\times(D\wedge(E\times F))))$



retorne *op*	*op*1 :	*op* :	*op*2 :
*op*2 ← ConstruirInfixa()	*op*1 : A	*op* : +	*op*2 :
*op*1 ← ConstruirInfixa()	*op*1 :	*op* :	*op*2 :

Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

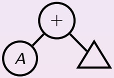
$op2$: 

$op2 \leftarrow$ ConstruirInfixa()	$op1$: A	op : $+$	
$op1 \leftarrow$ ConstruirInfixa()	$op1$:	op :	$op2$:

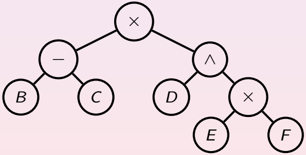
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

op :



$op2$:



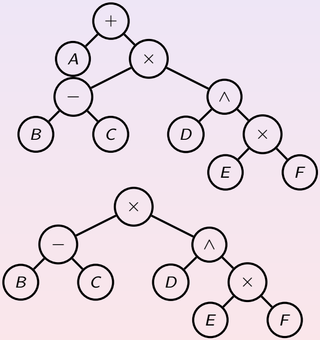
$op \rightarrow$InsereFilhoEsq$(op1)$	$op1$: 
$op1 \leftarrow$ ConstruirInfixa()	$op1$: op : $op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

op :

$op2$:



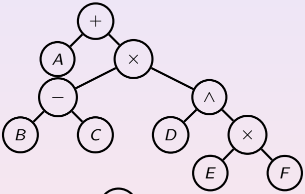
$op \rightarrow$InsereFilhoDir($op2$) $op1$: A

$op1 \leftarrow$ ConstruirInfixa() $op1$: op : $op2$:

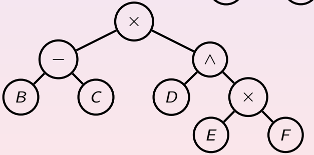
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

op :



*op*2 :

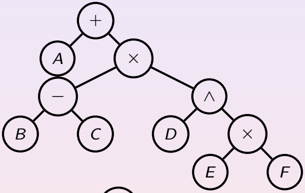


lerSimbolo()	*op*1 : 		
*op*1 ← ConstruirInfixa()	*op*1 :	*op* :	*op*2 :

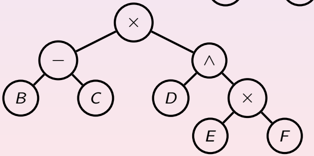
Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

op :



$op2$:



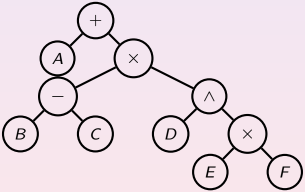
retorne op $\quad$ $op1$: A

$op1 \leftarrow \text{ConstruirInfixa}()$ $\quad$ $op1$: $\quad$ op : $\quad$ $op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op1$:

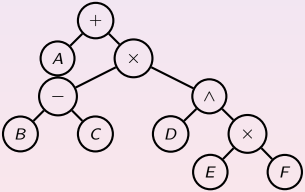


$op1 \leftarrow$ ConstruirInfixa() op : $op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op1$:

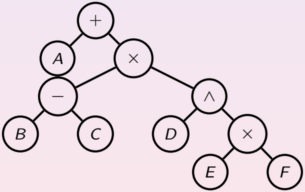


$S \leftarrow$ lerSimbolo() op : $op2$:

Construção - Infixa

$$(A+((B-C)\times(D\wedge(E\times F))))$$

$op1$:

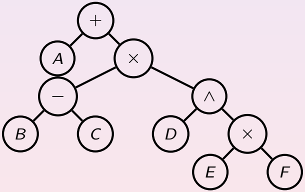


se$(S=\emptyset)\rightarrow$ **então** op : $op2$:

Construção - Infixa

$(A+((B-C)\times(D\wedge(E\times F))))$

$op1$:



retorne $op1$ op : $op2$:

Códigos de Huffman

- Representa uma técnica de compressão de dados que pode atingir valores entre 20 e 90%.
- É aplicado em arquivos de símbolos (texto) onde existe um distribuição diferenciada na freqüência com que cada símbolo aparece.
- Codifica-se os símbolos com tamanhos distintos de bits. Símbolos mais frequentes recebem menos bits, símbolos menos frequentes recebem mais bits.
- Na média o número de bits do arquivo será menor.

Exemplo do código de Huffman

- Considere o alfabeto $C = \{a, b, c, d, e, f\}$.
- Um dado arquivo possui a freqüência indicada na tabela abaixo para os caracteres do alfabeto.
- Também na tabela estão indicadas duas possíveis codificações para cada objeto, uma de tamanho fixo e outra de tamanho variável.

	a	b	c	d	e	f
Freqüência (milhares)	45	13	12	16	9	5
Código: *Tamanho Fixo*	000	001	010	011	100	101
Código: *Tamanho Variável*	0	101	100	111	1101	1100

- Qual o tamanho do arquivo para cada uma das codificações?

Calculando o custo para cada tipo de codificação

- Codificação com códigos de tamanho fixo:

$$\text{Total de bits} = 3 \times 100.000 = 300.000\,bits$$

- Codificação com códigos de tamanho variável:

$$\underbrace{1 \times 45}_{a} + \underbrace{3 \times 13}_{b} + \underbrace{3 \times 12}_{c} + \underbrace{3 \times 16}_{d} + \underbrace{4 \times 9}_{e} + \underbrace{4 \times 5}_{f} = 224.000\,bits$$

- Há um ganho de aproximadamente 25% se utilizarmos a codificação de tamanho variável.

O método

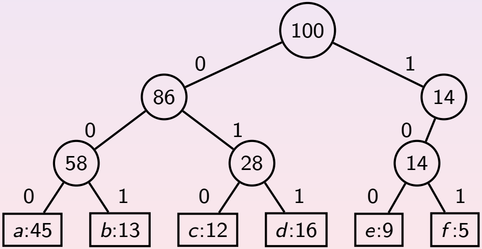
- A solução implica no uso de uma "codificação livre de prefixo".
- Em uma codificação livre de prefixo, para quaisquer símbolos distintos i e j codificados, a codificação de i não é prefixo da codificação de j.
- No exemplo anterior, usando a codificação variável para a palavra "abc" obtemos: `0101100`.
  - O único caracter começado com 0 e que portanto utiliza somente um bit é o 'a';
  - A seqüência 101 define o caracter 'b' e não há qualquer outro caracter que inicie com o código 101;
  - O restante 100 representa o caracter 'c'.

Representando o código

- Precisamos identificar uma estrutura que associe um código ao caracter, de forma que na decodificação encontremos facilmente o símbolo utilizando o código fornecido.
- Uma solução é utilizar uma árvore binária:
  - Um filho esquerdo está associado a um bit 0.
  - Um filho direito a um bit 1.
  - Nas folhas se encontram os símbolos.
  - O código lido 0 ou 1 faz com que na navegação na árvore chegue a um símbolo.
  - Ao achar um símbolo o próximo código é aplicado a partir do raiz.

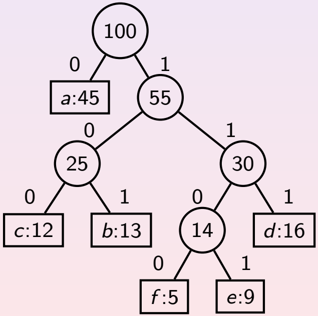
Código de tamanho fixo na forma de árvore

	a	b	c	d	e	f
Freqüência (milhares)	45	13	12	16	9	5
Código	000	001	010	011	100	101



Código de tamanho variável na forma de árvore

	a	b	c	d	e	f
Freqüência (milhares)	45	13	12	16	9	5
Código	0	101	100	111	1101	1100



Propriedades da árvore de código

- Cada código é livre de prefixo: Só há um único caminho para chegar a uma folha, que não passa por outra folha, assim o código de um símbolo não é um prefixo de outro símbolo.
- Uma codificação ótima deve ser representado por uma árvore binária cheia, cada vértice interno tem dois filhos. Seja uma codificação com um vértice interno que só tenha um filho:
  - Se o filho for uma folha. Podíamos colocar esta folha no lugar do vértice e economizaríamos um bit para o código deste símbolo.
  - Se o filho for outro vértice. A partir deste vértice buscamos uma folha, colocamos esta folha como segundo filho do vértice. Economizaríamos no mínimo um bit.
- Buscamos uma árvore binária cheia com $|C|$ folhas (o tamanho do alfabeto) e $|C - 1|$ vértices internos.

Entendendo a proposta de solução

- Começar com $|C|$ árvores folhas isoladas e realizar seqüencialmente $|C - 1|$ operações de agregação, agregando duas árvores a um novo vértice raiz comum. O raiz passa a ter como "peso" a soma dos custos de cada árvore agregada.
- A escolha do par de árvores que serão agregadas dependerá do custo de cada árvore. As duas árvores de menor custo serão escolhidas.
- O raiz de uma árvore carrega como informação o custo da árvore.

O algoritmo de Huffman

Entrada: Conjunto de caracteres de C e a freqüências f de cada caracter

Saída: Raiz da árvore binária representando codificação ótima livre de prefixo

Algoritmo HUFFMAN(C)

$n \leftarrow |C|$

$Q \leftarrow C$ $\triangleright$ Q é fila de árvores ordenadas por custo

para $i \leftarrow 1$ **até** $n - 1$ **faça**

$z \leftarrow$ **novo** *Arvore*

$z.esq \leftarrow Extrai\_Minimo(Q)$

$z.dir \leftarrow Extrai\_Minimo(Q)$

$z.info = z.esq.info + z.dir.info$

$Insere(Q, z)$

retorne $Extrai\_Minimo(Q)$

Árvore Binária de Busca

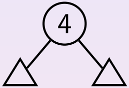
- Árvore Binária de Busca utiliza uma classificação de ordem ao inserir um elemento.
- Dado um raiz da árvore ou subárvore:
  - Todos os descendentes do lado esquerdo tem valores menor ou igual ao valor da raiz.
  - Todos descendentes do lado direito tem valores maior que o da raiz.
- As operações sobre uma árvore binária de busca são:
  - Inserir um elemento.
  - Remover um elemento.
  - Buscar um elemento.
  - Listar os elementos em ordem.
- O nome advém da busca, que é semelhante a uma busca binária em uma sequência ordenada.

ABB - Operações: Inserção

- A inserção de um elemento é simples, se dá a partir do raiz da árvore.
  - Se o elemento for menor ou igual, tenta-se inserir no filho esquerdo.
  - Se o elemento for maior, tenta-se inserir no filho direito.
- O elemento é inserido quando atinge uma árvore vazia.

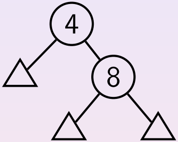
Inserindo Elementos:

4 8 5 0 2 8 1 4 5 9 6 3



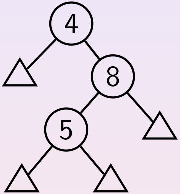
Inserindo Elementos:

4 8 5 0 2 8 1 4 5 9 6 3



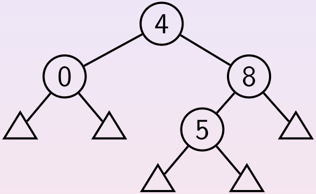
Inserindo Elementos:

4 8 5 0 2 8 1 4 5 9 6 3



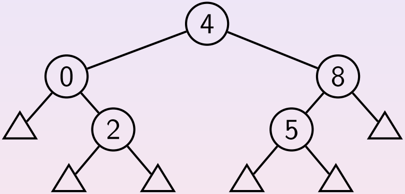
Inserindo Elementos:

4 8 5 0 2 8 1 4 5 9 6 3



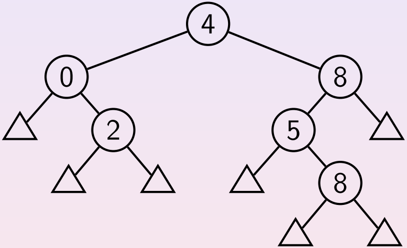
Inserindo Elementos:

4 8 5 0 2 8 1 4 5 9 6 3



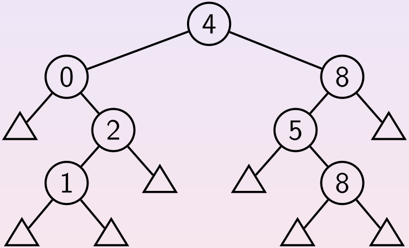
Inserindo Elementos:

4 8 5 0 2 8 1 4 5 9 6 3



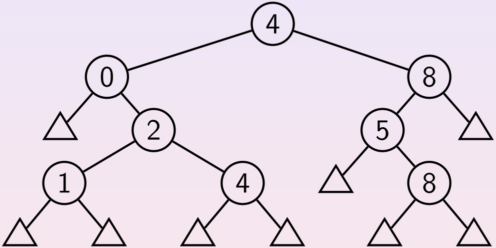
Inserindo Elementos:

4 8 5 0 2 8 1 4 5 9 6 3



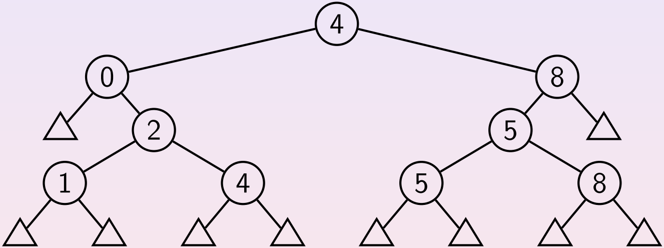
Inserindo Elementos:

4 8 5 0 2 8 1 4 5 9 6 3



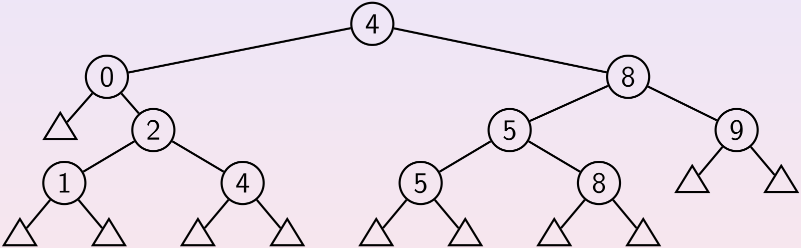
Inserindo Elementos:

4 8 5 0 2 8 1 4 5 9 6 3



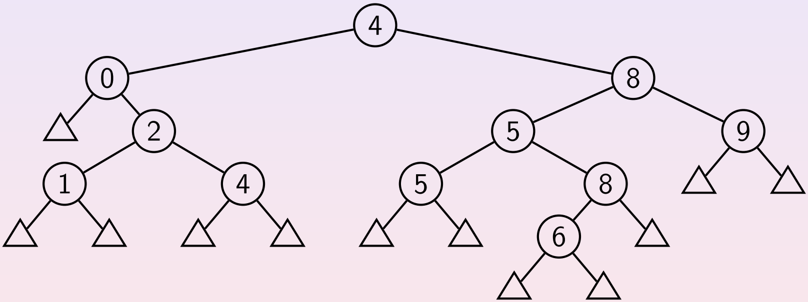
Inserindo Elementos:

4 8 5 0 2 8 1 4 5 9 6 3



Inserindo Elementos:

4 8 5 0 2 8 1 4 5 9 6 3



Inserindo Elementos:

4 8 5 0 2 8 1 4 5 9 6 3

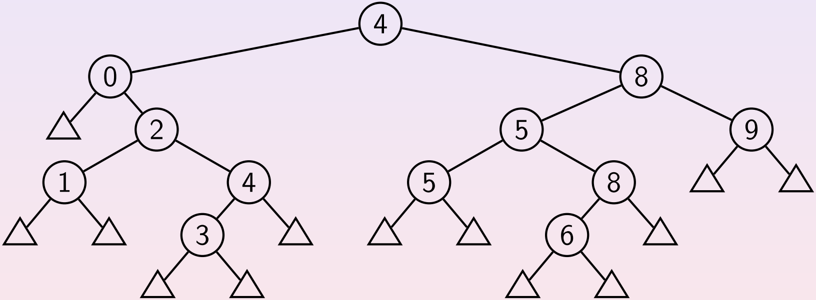


ABB - Operações: Inserir Elementos - Algoritmo

Algoritmo $\textsc{insValor}(raiz, v)$

$p = \texttt{criarArvore}(v)$

se $raiz = vazia$ **então**

$raiz = p$

senão

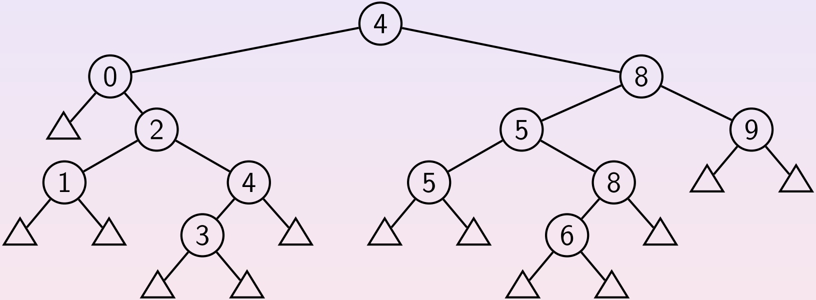
se $v \leqslant \texttt{info}(raiz)$ **então**

$\texttt{insValor}(\texttt{filhoEsq}(raiz))$

senão

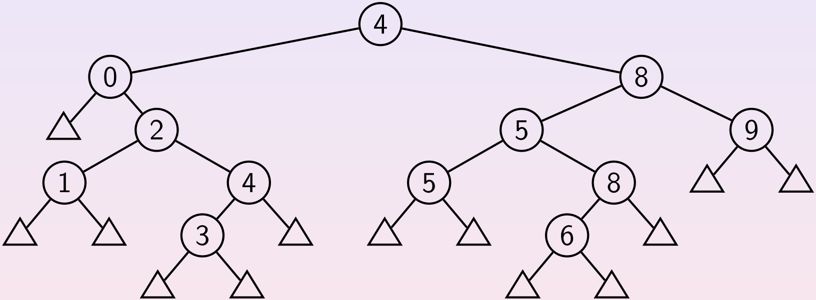
$\texttt{insValor}(\texttt{filhoDir}(raiz))$

ABB - Operações: Listar os elementos em ordem



Como ficaria uma busca em ordem nesta árvore?

ABB - Operações: Listar os elementos em ordem



Como ficaria uma busca em ordem nesta árvore: 0 1 2 3 4 4 5 5 6 8 8 9

ABB - Operações: Busca de elementos

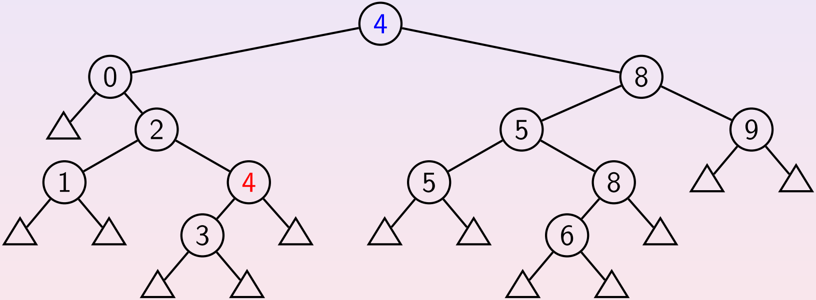
- A busca de um valor é exatamente igual à busca binária em uma lista ordenada, onde o meio é o raiz:
    - Se o $valor == info$: retorna - encontrou (ou a árvore)
    - Se o $valor < info$: busca recursivamente no filho esquerdo.
    - Senão: busca recursivamente no filho direito.
- A base é a folha. Se atingir a folha: retorna - não encontrou (ou a árvore (que é NULL))

ABB - Operações: Remoção de um elemento

- Se o item que queremos remover é uma folha, é simples, basta remover a folha.
- Se o item só possui um filho (o outro é uma árvore vazia), então promove o filho para esta posição e remove o item.
- Caso contrário, precisamos fazer uma troca.
- São duas opções de escolha:
  - Descendente "mais direito" do filho esquerdo: `predecessor`
  - Descendente "mais esquerdo" do filho direito: `sucessor`
- Se o descendente for uma folha, promove ele para ocupar a posição do elemento que irá remover e remove o elemento.
- Se o descendente não for uma folha, possui um filho que não é o "mais". Promove o filho para a posição deste descendente. Promove o descendente para a posição do elemento que será removido e remove o elemento.

ABB - Operações: Remoção - Exemplo

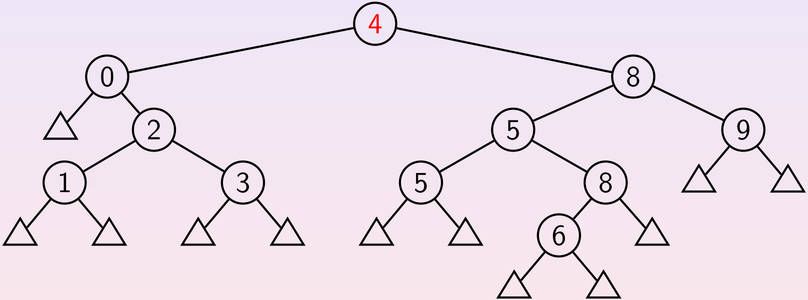
- Elemento a remover: 4 (Descendente "mais à direita" do filho esquerdo: 4)



- 4 não é folha: Promove o filho (3) à sua posição, promove ele na posição de quem será removido:

ABB - Operações: Remoção - Exemplo

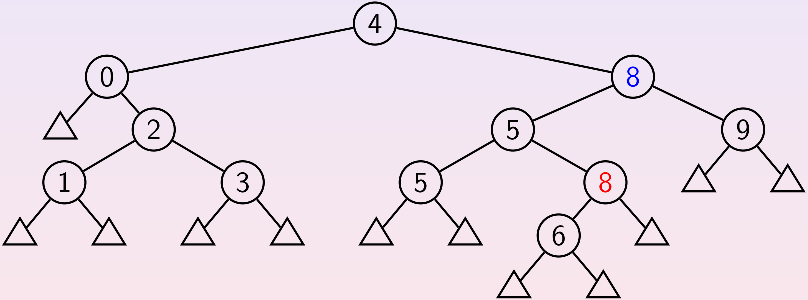
- Elemento 4 removido:



- 4 foi promovido, bem como 3

ABB - Operações: Remoção - Exemplo

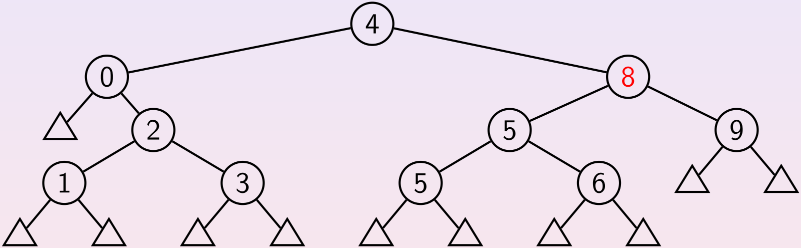
- Elemento a remover: 8 (Descendente "mais à direita" do filho esquerdo: 8)



- 8 não é folha: Promove o filho (6) à sua posição, promove ele na posição de quem será removido:

ABB - Operações: Remoção - Exemplo

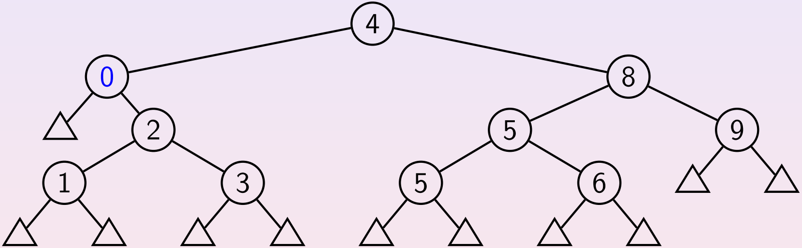
- Elemento 8 removido:



- 8 foi promovido, bem como 6

ABB - Operações: Remoção - Exemplo

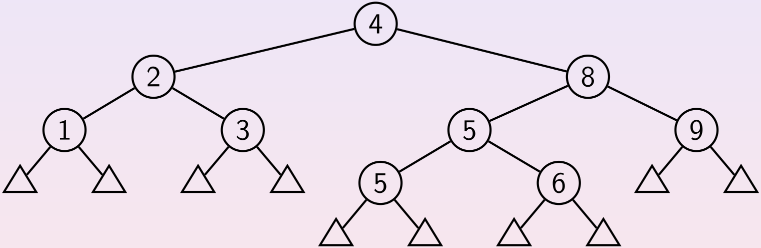
- Elemento a remover: 0: Só tem um descendente



- Promove o filho (2) à sua posição:

ABB - Operações: Remoção - Exemplo

- Elemento 0 removido:



- 2 foi promovido

ABB - Operações: Remoção - Exemplo

- Elemento a remover: 9: É folha, basta remover

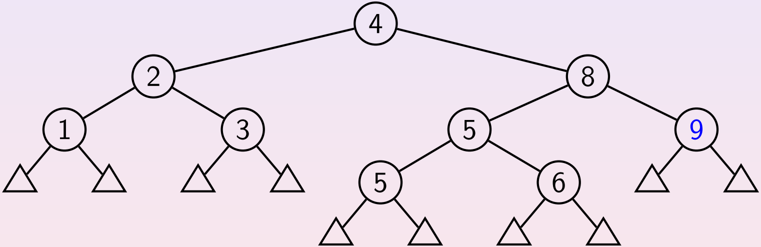


ABB - Operações: Remoção - Exemplo

- Elemento 9 removido:

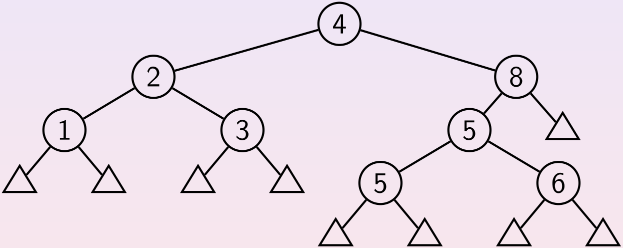


ABB: Operações Remover Elemento - Algoritmo

Algoritmo $\textsc{Remover}(raiz, valor)$
  $fe = \texttt{filhoEsq}(raiz)$
  $fd = \texttt{filhoDir}(raiz)$
  $p = raiz$
  se $\texttt{ehVazia}(fe)$ **e** $\texttt{ehVazia}(fd)$ **então**
    $raiz = \texttt{vazia}$
  senão
    se $\texttt{ehVazia}(fe)$ **então**
      $raiz = fd$
    se $\texttt{ehVazia}(fd)$ **então**
      $raiz = fe$
  se (**não** $\texttt{ehVazia}(fe)$) **e** (**não** $\texttt{ehVazia}(fd)$) **então**
    enquanto não $\texttt{ehVazia}(\texttt{filhoDir}(fe))$ **faça** $fe = \texttt{filhoDir}(fe)$
    $\texttt{setInfo}(raiz, \texttt{info}(fe))$
    $p = fe$
    $fe = \texttt{filhoEsq}(fe)$
  $\texttt{free}(p)$
  retorne

Ordenação por Inserção em árvore Binária

- São n elementos inseridos.
- Cada elemento inserido percorre a altura da árvore.
- No pior caso a altura da árvore é n (sequência já ordenada) $\rightarrow O(n^2)$
- No melhor caso a árvore binária é completa, a altura é $\log n \rightarrow O(n \log n)$
- No caso médio a altura da árvore é proporcional a $\log n \rightarrow O(n \log n)$