Entendendo Árvores Binárias

Serão apresentadas a seguir as operações com árvores binárias: inserir elemento na árvore binária, exibir a árvore binária, remover elemento da árvore binária e implementando operações de árvore binária em uma linguagem de programação

ABB

Uma árvore binária pode estar vazia ou pode ser composta de três partes: o nó raiz, a subárvore à direita do nó raiz e a subárvore à esquerda do nó raiz que também são árvores binárias.

Uma subárvore binária também pode estar vazia. Cada elemento da árvore binária é o que chamamos de nó e cada ligação de um nó para outro é o que chamamos de arco. Na seção seguinte vamos exemplificar uma árvore binária.

A árvore binária de busca também é conhecida como árvore binária de pesquisa.

A árvore binária de busca é uma estrutura de dados que armazena informações de forma organizada.

À esquerda de um nó de uma árvore binária de busca, as informações possuem valores menores do que a do nó.

À direita de um nó de uma árvore binária de busca, as informações possuem valores maiores ou iguais do que a do nó.

O objetivo principal de uma árvore binária de busca é estruturar suas informações de forma que seja possível uma busca binária.

A árvore binária de busca é também conhecida como ABB.

As operações das árvores binárias de busca são as inserções, remoções e buscas.

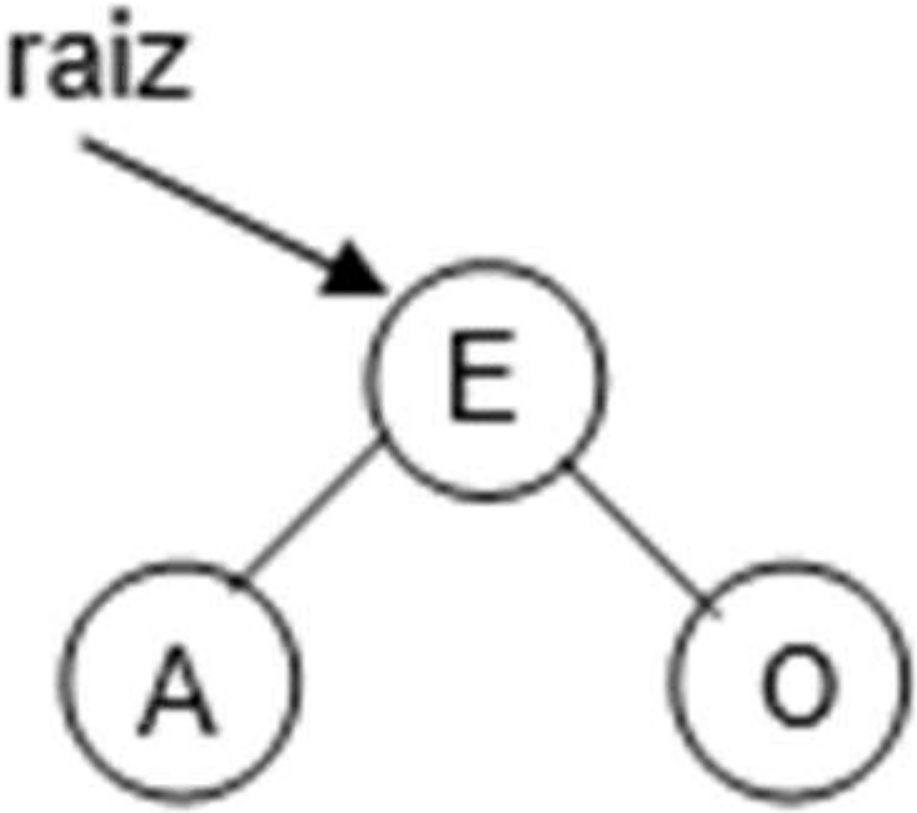
INSERIR NA ABB

Inserir é um módulo procedimento da operação inserir que recebe como parâmetro um BIntNo árvore e um novoNo a ser inserido.

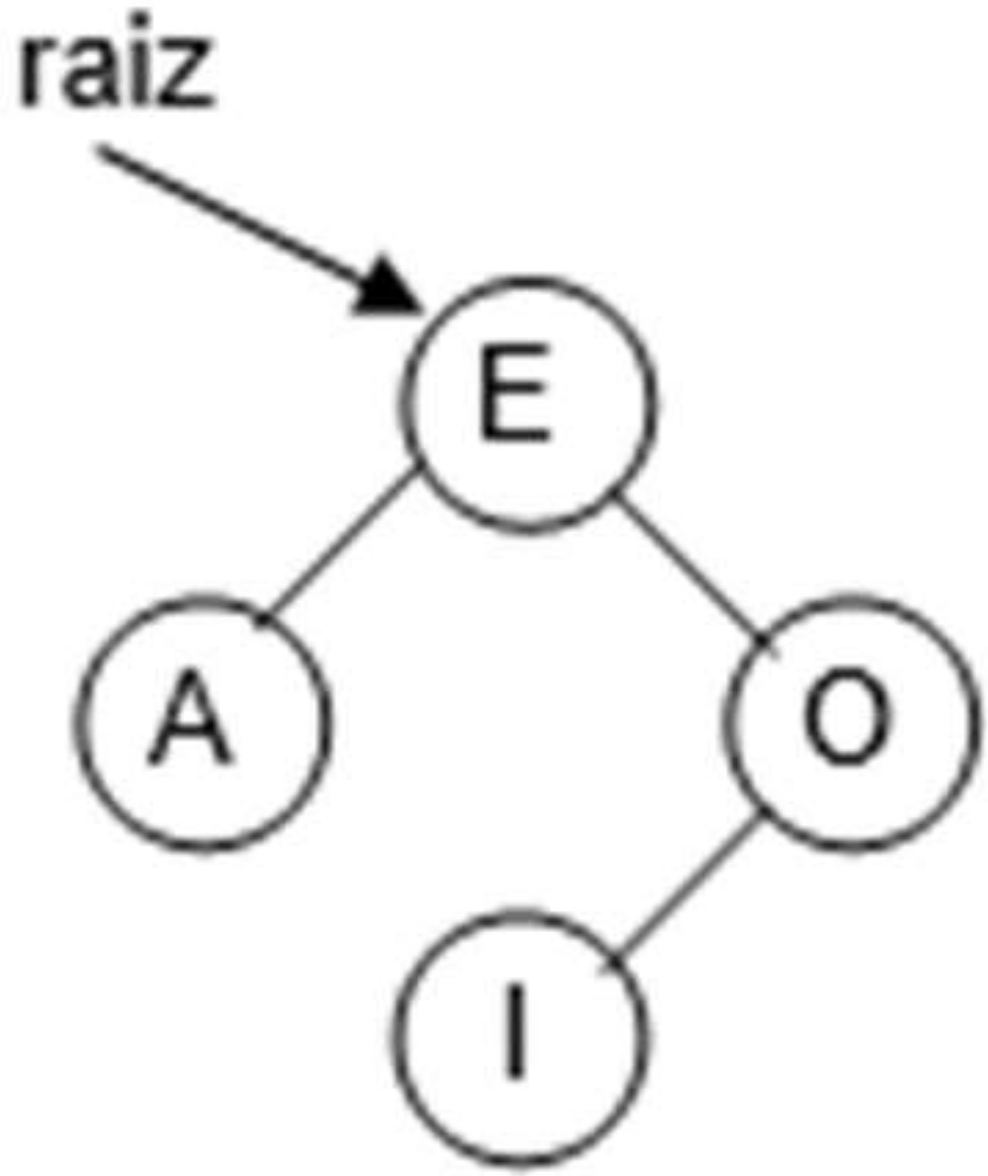
```
BIntNo inserir (arvore BIntNo, novoNo numérico_inteiro)
início_módulo
 se (arvore = nulo)
   então
    retornar novo BIntNo (novoNo);
   senão
    se (novoNo < arvore.valor)
      então
       arvore.esq ← inserir (arvore.esq, novoNo);
      senão
      arvore.dir ← inserir (arvore.dir, novoNo);
    fimse;
```

```
  fimse;

  retornar arvore;

fim_módulo;



inserirNo (novoValor numérico_inteiro)

início_módulo

  Raiz ← inserir(Raiz, novoValor);

fim_módulo;
```

Observando a figura seguinte, a inserção aconteceu primeiro da letra E, depois da letra A, à esquerda de E, pois A é menor do que E e, por fim, a inserção da letra O, à direita de E, pois a letra O é maior do que E.



Por último, realizamos a inserção da letra I na árvore ABB, a letra I, para ser inserido na árvore, percorreu por E, indo à direita de E, pois, I é maior do que E e, depois percorreu pela letra O, indo à esquerda da letra O, pois I é menor do que O. Observe a figura seguinte.



REMOVER DA ÁRVORE BINÁRIA

Remover é uma operação da árvore binária. É um módulo procedimento que recebe um nó e remove esse nó.

No (item numérico_inteiro)

início_módulo

tempNo, pai, filho, temp BIntNo;

```
tempNo ← Raiz;

pai ← null;

filho ← Raiz;



enquanto (tempNo <> nulo e tempNo.valor <> item) faça

 pai ← tempNo;

 se (item < tempNo.valor)

  então

   tempNo ← tempNo.esq;

  senão

   tempNo ← tempNo.dir;

 fimse;

 se (tempNo = nulo)

  então

   escrever(“item não localizado!”);

 fimse;

 se (pai = nulo)

  então
```

```
se (tempNo.dir = nulo)
 então
  Raiz ← tempNo.esq;
 senão
  se (tempNo.esq = nulo)
   então
    Raiz ← tempNo.dir;
   senão
    para temp ← tempNo e filho ← tempNo.esq até filho.dir <> null
                                                passo temp ← filho e filho ← filho.dir
faça
    fimpara;
    se (filho <> tempNo.esq)
     então
      temp.dir ← filho.esq;
      filho.esq ← Raiz.esq;
    fimse;
    filho.dir ← Raiz.dir;
    Raiz ← filho;
```

```
      fimse;
   fimse;
senão
  se (tempNo.dir = nulo)
   então
    se (pai.esq = tempNo)
      então
       pai.esq ← tempNo.esq;
      senão
       pai.dir ← tempNo.esq;
    fimse;
   senão
    se (tempNo = nulo)
      então
       se (pai.esq = tempNo)
         então
          pai.esq ← tempNo.dir;
         senão
          pai.dir ← tempNo.dir;
```

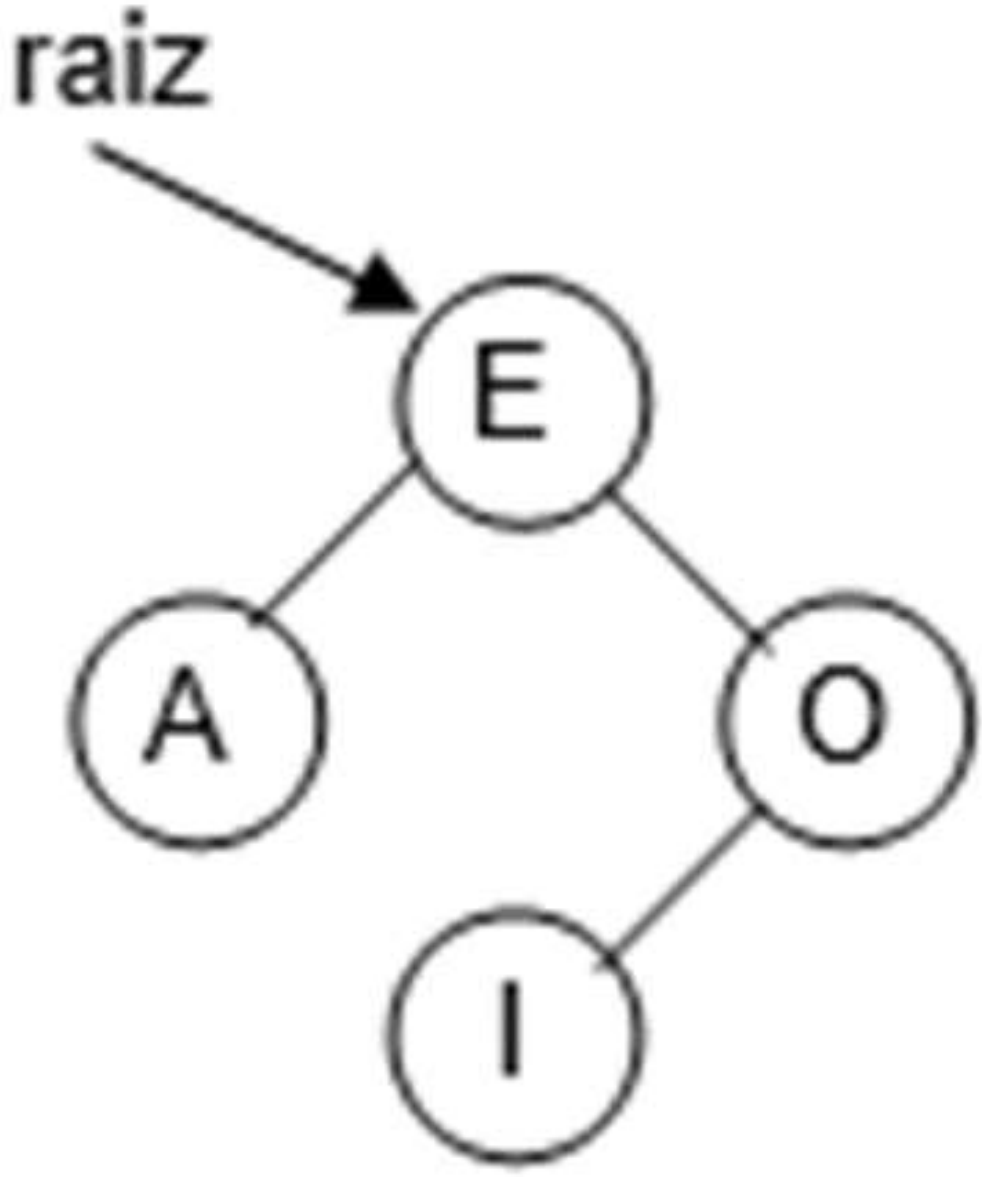
```
      fimse;
    senão
      para temp ← tempNo e filho ← tempNo.esq até filho.dir <> nulo
                                        passo temp ← filho e filho ← filho.dir
      fimpara;
      se (filho <> tempNo.esq)
        então
          temp.dir ← filho.esq;
          filho.esq ← tempNo.esq;
      fimse;
      filho.dir ← tempNo.dir;
      se (pai.esq = tempNo)
        então
          pai.esq ← filho;
        senão
          pai.dir ← filho;
      fimse;
  fimse;
fimse;
```

fimse;

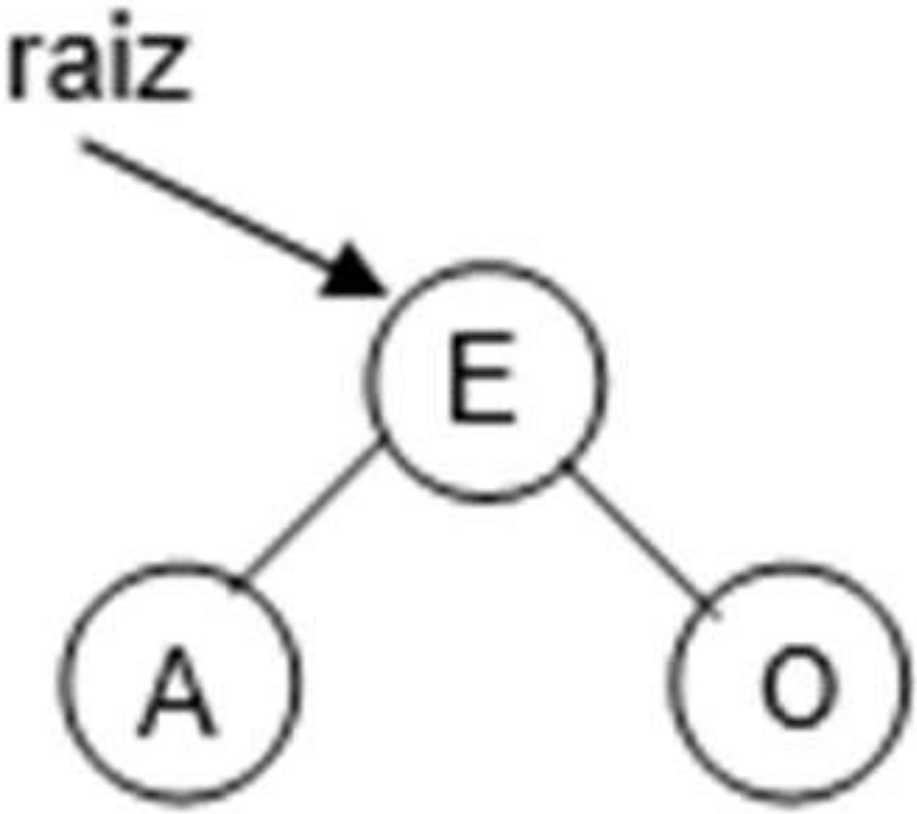
fimenquanto;

fimmódulo;

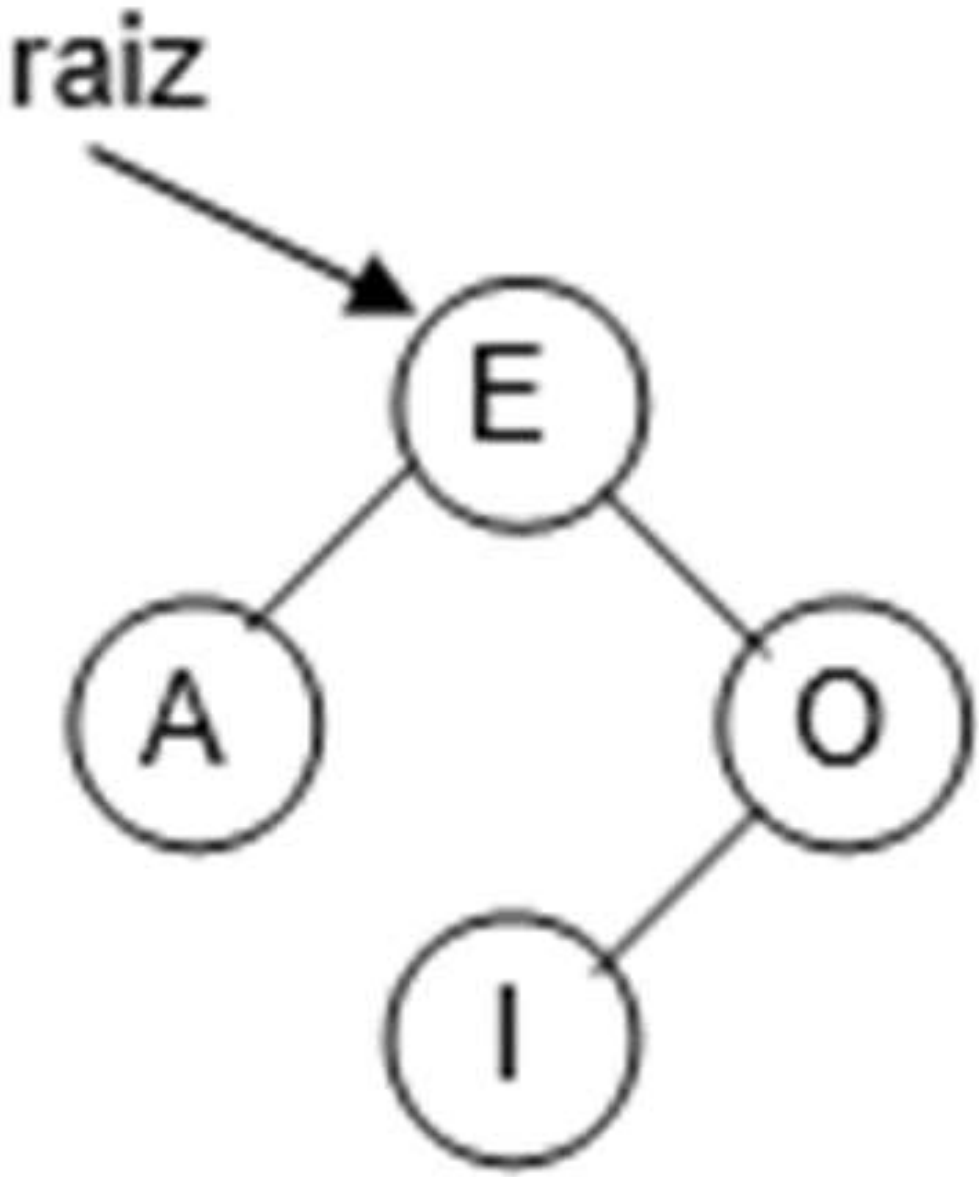
Observada a imagem seguinte, vamos remover a letra I da árvore ABB. Como a letra I está na folha, basta realizar a remoção.



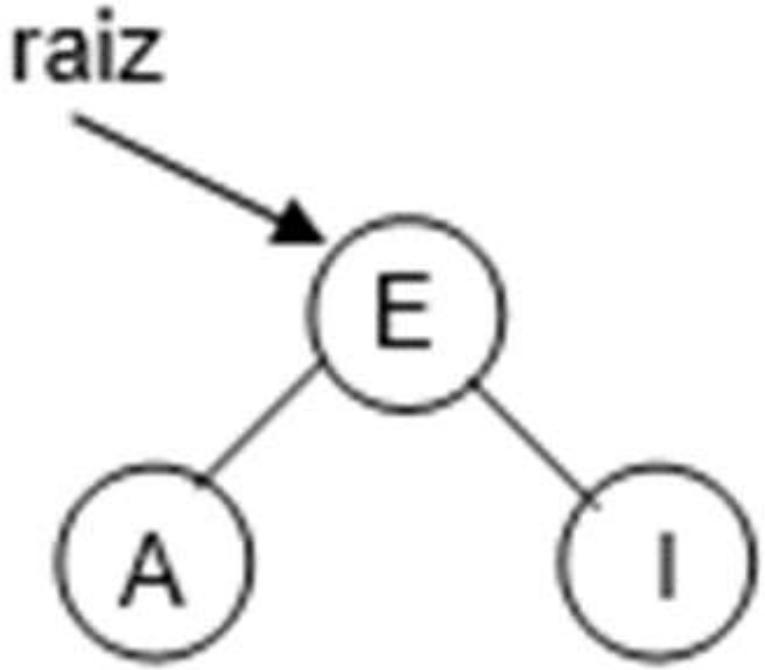
Segue a imagem da árvore ABB com o nó da letra I removida.



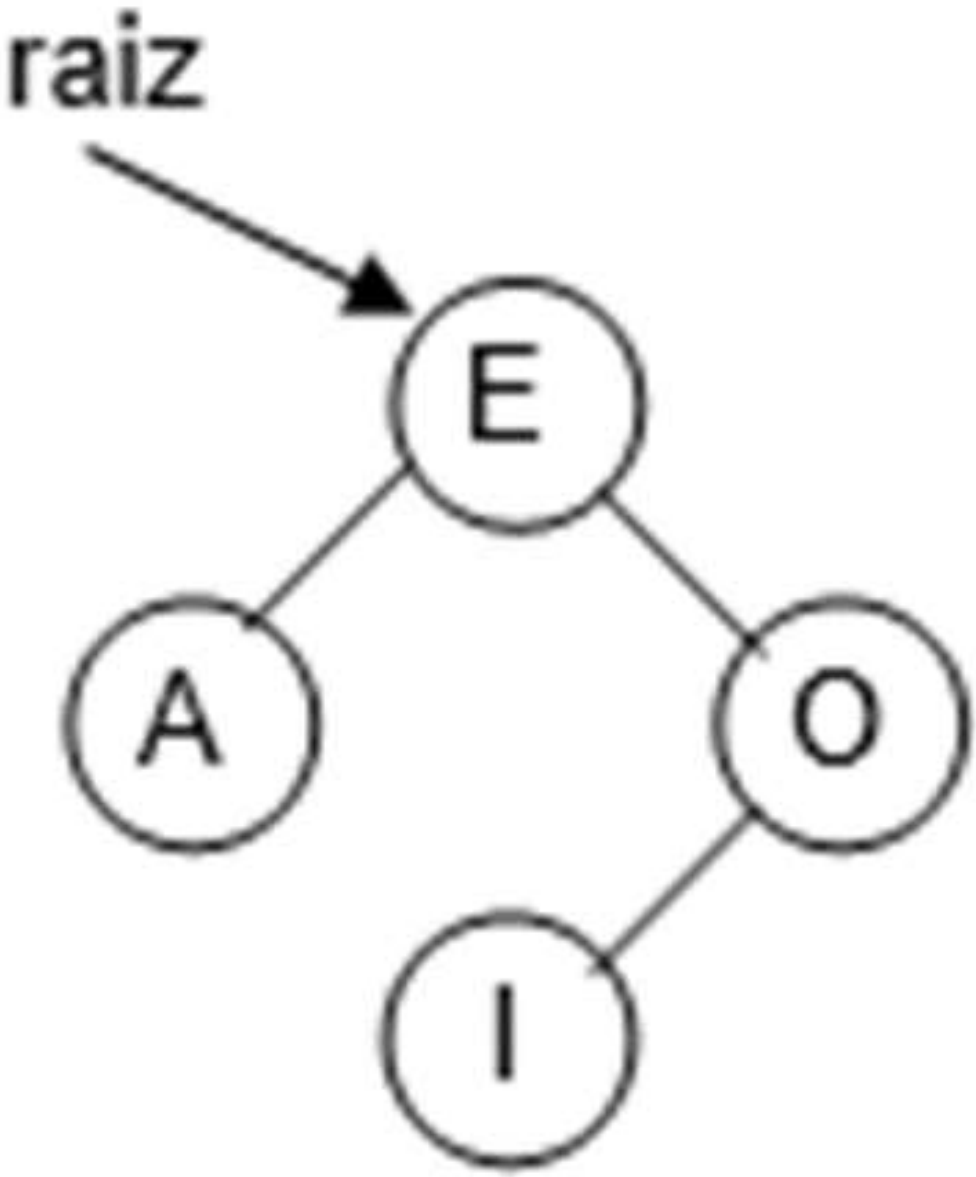
Observada a imagem seguinte, vamos remover a letra O da árvore ABB. Como a letra O não está na folha, para realizar a remoção, verificamos que o nó da letra O possui um filho, trocamos os elementos dos dois nós, O com I, e removemos o nó folha com a letra O.



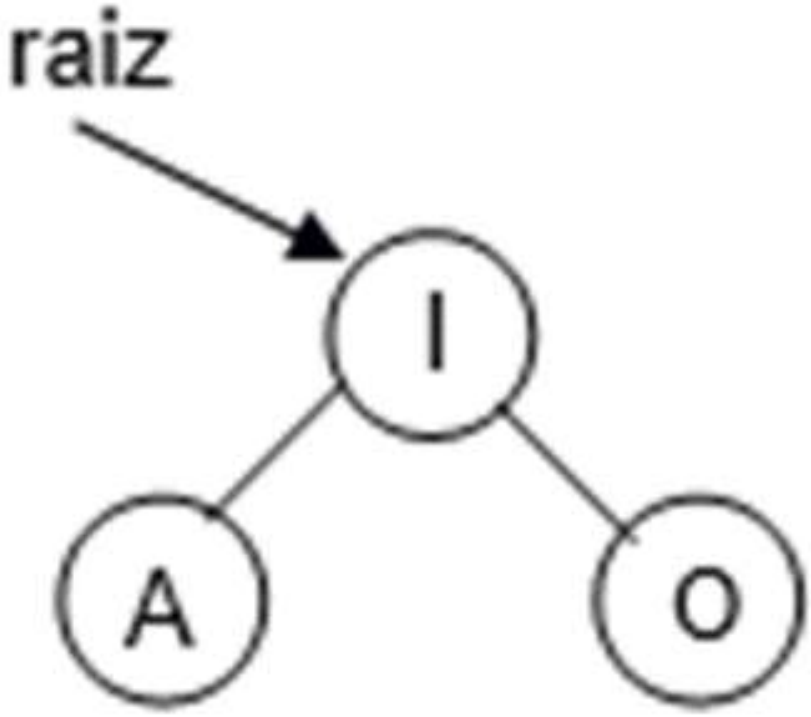
Observe a imagem da árvore ABB com o nó da letra O removida.



Observada a imagem seguinte, vamos remover a letra E da árvore ABB. Como a letra E não está na folha, para realizar a remoção, verificamos que o nó da letra E possui dois filhos, buscamos o menor elemento da subárvore à direita, que será uma folha, neste caso, será a letra I e trocamos os elementos dos dois nós, E com I, e removemos o nó folha com a letra E.



Observe a imagem da árvore ABB com o nó da letra E removida.



EXIBIR ÁRVORE BINÁRIA

ExibirEsq é uma operação de árvore binária. É um módulo procedimento que recebe um nó e mostra o elemento que está à esquerda desse nó.

```
exibirEsquerdo (arv BIntNo)
início_módulo
  se (arv <> nulo)
    então
      exibirEsquerdo (arv.esq);
      escrever(arv.valor);
```

```
  fimse;
fim_módulo;
```

```
exibirNoEsq()
início_módulo
  exibirEsquerdo(Raiz);
fim_módulo;
```

ExibirDir é uma operação da árvore binária. É um módulo procedimento que recebe um nó e mostra o elemento que está à direita desse nó.

```
exibirDireito (arv BIntNo)
início_módulo
  se (arv <> nulo)
    então
      exibirDireito(arv.dir);
      escrever (arv.valor);
  fimse;
fim_módulo;
```

exibirNoDir()

início_módulo

  exibirDireito(Raiz);

fim_módulo;

ExibirRaiz é uma operação da árvore binária. É um módulo procedimento que mostra o elemento que está no nó raiz.

exibirRaiz()

início_módulo

  escrever("raiz " , Raiz.valor);

fim_módulo;

ABB NO JAVA

Para exemplificar as operações de árvore binária na linguagem de programação Java, vamos desenvolver um algoritmo que recebe do usuário cinco números

inteiros.

Esses números serão inseridos como elementos nos nós de uma árvore binária, de acordo com a regra (recursiva) de árvore binária:

Regra recursiva: “Em uma árvore binária, todo elemento à esquerda é menor que a raiz, todo elemento à direita é maior ou igual à raiz”.

E, ao final do desenvolvimento do algoritmo, mostrar todos os elementos que foram inseridos na árvore binária.

```
class teste{

  public static void main (String args []) {

    ArvoreBinaria arv = new ArvoreBinaria();


          arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog(“Digite um número inteiro”)));

          arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog(“Digite um número inteiro”)));

          arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog(“Digite um número inteiro”)));
```

```
            arv.inserirNo(Integer.parseInt(  JOptionPane.showInputDialog("Digite  um
número inteiro")));

            arv.inserirNo(Integer.parseInt(  JOptionPane.showInputDialog("Digite  um
número inteiro")));

     arv.exibirNo();

     System.exit(0);

   }

}
```

O programa Java escrito no NetBeans.

```
import javax.swing.*;

class BIntNo
 {
    int valor;
    BIntNo esq, dir;

    BIntNo(int novoValor)
      {
        valor = novoValor;
      }
 }
```

```
class ArvoreBinaria
{
  private BIntNo Raiz;

  private BIntNo inserir (BIntNo arvore, int novoNo)
   {...18 lines }

  public void inserirNo (int novoValor)
   {...3 lines }

  private void exibirEsquerdo (BIntNo arv)
   {...7 lines }

  private void exibirDireito (BIntNo arv)
   {...7 lines }

  public void exibirRaiz()
   {...3 lines }

  public void exibirNoEsq()
   {...3 lines }

  public void exibirNoDir()
   {...3 lines }

  public void exibirNo()
   {...5 lines }

  public void excluirNo (int item)
   {...102 lines }
}
```

```
class teste
{
  public static void main (String args [])
   {
     ArvoreBinaria arv = new ArvoreBinaria();

     arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog
                                     ("Digite um número inteiro")));
     arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog
                                     ("Digite um número inteiro")));
     arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog
                                     ("Digite um número inteiro")));
     arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog
                                     ("Digite um número inteiro")));
     arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog
                                     ("Digite um número inteiro")));
     arv.exibirNo();
     System.exit(0);
   }
}
```

Atividade extra

Indicação de leitura:

Você pode utilizar o livro Estrutura de Dados: algoritmos, análise da complexidade e implementações em Java e C/C++, da Ana Fernanda Gomes Ascencio e Graziela Santos de Araújo, no capítulo 7 sobre a estrutura de dados do tipo Árvore Binária.

Referência Bibliográfica

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.;STEIN, C. **Algoritmos Teoria e Prática**. Editora Cammpus. 3a Edição. 2012.

GOODRICH, M. T.; TAMASSIA, R. **Estruturas de Dados & Algoritmos em Java**. Editora Grupo A: Bookman, 5a Edição. 2013.

Ir para exercício