Algoritmos e Estruturas de Dados II
Roteiro de Laboratório – Árvore AVL

Objetivo

Implementar uma Árvore AVL a partir de uma Árvore Binária de Pesquisa (ABP).

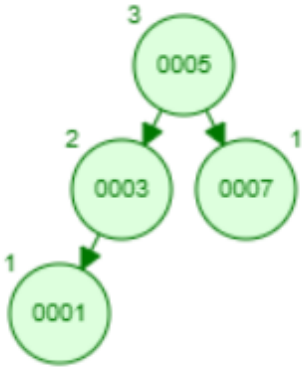
Pré-requisitos

- Código funcional de uma Árvore Binária de Pesquisa (classes No e ArvoreBinaria).
- Compreensão dos conceitos de balanceamento em árvores binárias a partir de rotações simples e duplas

Atividades

1. Adicionar atributo nível

   Alterar a classe No para incluir um novo atributo chamado "nível" do tipo inteiro. Este atributo irá armazenar a quantidade de níveis que existem abaixo do nó (incluindo o próprio nó). Veja o exemplo a seguir que apresenta os valores de níveis para os nós em uma AVL:



2. Criar método para atualizar nível

   Criar um método na classe No "void setNivel()" que atualiza o nível do nó, seguindo a definição fornecida para o atributo. Dica: pense na regra que define o nível de um nó como uma equação.

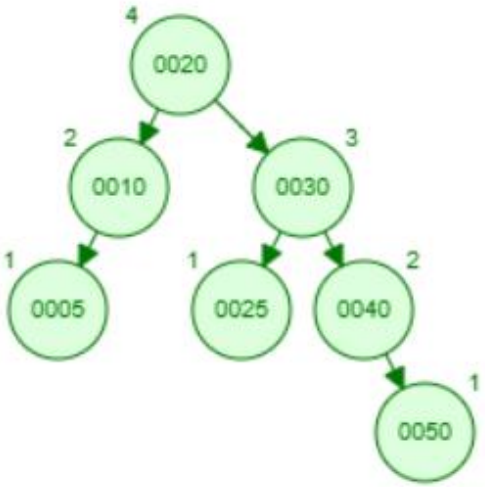
3. Atualizar os níveis dos nós na inserção

   Alterar o método de inserção recursivo da classe ArvoreBinaria para que, antes de retornar o ponteiro para o nó, o nível deste nó seja atualizado.

4. Validar o cálculo dos níveis

   Criar um programa para validar que os níveis dos nós estão sendo calculados com sucesso. Para isso, seu programa deve:

   a. Criar uma árvore vazia.
   b. Inserir os elementos na árvore conforme exemplo a seguir.
   c. Caminhar na árvore e, em cada visita, imprimir o elemento visita juntamente com seu nível.



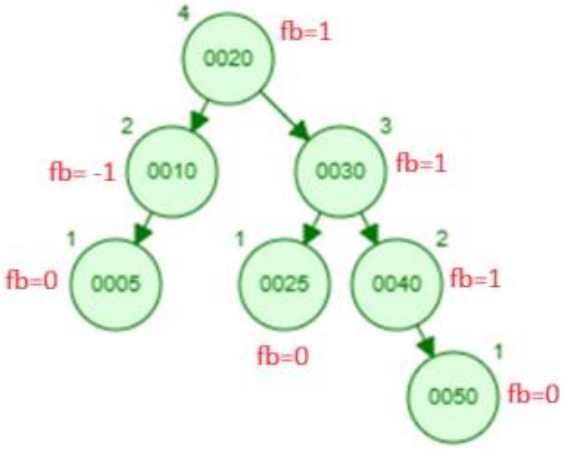
Para esta árvore, espera-se que seu programa imprima no caminhamento central:

5(1) 10(2) 20(4) 25(1) 30(3) 40(2) 50(1)

5. Criar método para obter fator de balanceamento
   Criar um método na classe No “int getFatorBalanceamento()” que calcula e retorna o fator de balanceamento do nó. O fator de balanceamento do nó é calculado com base nos níveis dos filhos, seguindo a regra:

   fator de balanceamento = nível(filho à direita) - nível(filho à esquerda)

   Veja o exemplo a seguir onde são apresentados, para cada nó da AVL, o nível e o fator de balanceamento.



6. Validar o cálculo dos níveis
   Alterar seu programa para que, em cada visita, também seja impresso o fator de balanceamento do nó. Para essa árvore, espera-se que seu programa imprima no caminhamento central:

   5(1)(fb=0) 10(2)(fb=-1) 20(4)(fb=1) 25(1)(fb=0) 30(3)(fb=1) 40(2)(fb=1) 50(1)(fb=0)

7. Criar os métodos para os casos de balanceamento
   Criar, na classe ArvoreBinaria, os métodos para os casos de balanceamento:

a. Rotação simples à esquerda
b. Rotação simples à direita
c. Rotação dupla direita-esquerda
d. Rotação dupla esquerda-direita

Cada método deve receber um ponteiro para o nó a ser rotacionado, aplicar a rotação e retornar um ponteiro para o nó que é a nova raiz do conjunto após a rotação. Lembre-se de atualizar o nível dos nós que sofreram alteração de nível no processo de rotação.

8. Criar o método para balancear um nó
   Criar um método na classe ArvoreBinaria "No balancear(No i)" que recebe um ponteiro para um nó e aplica as regras de balanceamento neste nó. O método deve retornar um ponteiro para o nó que é a nova raiz do conjunto após o balanceamento.

9. Integrar o balanceamento à inserção
   Alterar o método de inserção recursivo da classe ArvoreBinaria para que, antes de retornar o ponteiro para o nó, aplique o balanceamento neste nó e retorne o ponteiro para o nó que é raiz resultante do balanceamento.

10. Validar a AVL
    Alterar seu programa para que insira elementos na AVL e verifique se os balanceamentos estão sendo aplicados corretamente. Use o exemplo a seguir para verificação.

    Ordem de inserção: 4, 35, 10, 13, 3, 30, 15, 12, 7, 40 e 20