Universidade Presbiteriana Mackenzie

ESTRUTURA DE DADOS II

Árvore AVL – parte 1/2
Atividade (máx. três alunos)

Objetivo

Implementar cálculo do fator de balanceamento e operações de rotação para iniciar a implementação de árvore AVL em Java.

Instruções

- A atividade deve ser resolvida usando a linguagem Java.
- A solução não deve usar as estruturas de dados oferecidas pela linguagem Java (projetos que usarem tais estruturas serão desconsiderados – zero).
- Inclua a identificação do grupo (nome completo e RA de cada integrante) no início de cada arquivo de código, como comentário.
- Inclua todas as referências (livros, artigos, sites, vídeos, entre outros) consultadas para solucionar a atividade, como comentário no arquivo .java que contém a main().

Enunciado

1. Para a primeira parte da implementação da árvore AVL, você deve usar uma versão *modificada* da classe Java que representa um nó de uma árvore binária, criada na atividade *Lab1a - Árvore Binária*. Essa nova versão do nó deve:

- Armazenar um número inteiro como dado (data), ao invés de uma String.
- Ter um novo atributo balanceFactor que armazena o fator de balanceamento de um nó.
- Ter um novo método público getBalanceFactor() que retorna o fator de balanceamento do nó.
- Ter um novo método privado updateBalanceFactor() que atualiza o fator de balanceamento do nó sempre que necessário.

2. Crie uma classe Java que define um novo tipo de dado usado para representar uma árvore AVL (ex. AVL). Essa classe deve ser, obrigatoriamente, uma subclasse (especialização) da BST que você criou na atividade *Lab1b - Árvore Binária de Busca (BST)*.

A classe da árvore AVL não possui novos atributos, apenas novos métodos *privados*, conforme a tabela a seguir.

OPERAÇÃO	DESCRIÇÃO
Construtor(es)	Construtor(es) da classe.
rotateLeft(root)	Aplica a rotação para esquerda (rotação LL) na subárvore cuja raiz é o nó indicado no parâmetro root. Retorna a nova raiz da subárvore após a rotação ser aplicada.

rotateRight(root)	Aplica a rotação para direita (rotação RR) na subárvore cuja raiz é o nó indicado no parâmetro root. Retorna a nova raiz da subárvore após a rotação ser aplicada.
rotateLeftRight(root)	Aplica a rotação esquerda-direita (rotação LR) na subárvore cuja raiz é o nó indicado no parâmetro root. Retorna a nova raiz da subárvore após a rotação ser aplicada.
rotateRightLeft(root)	Aplica a rotação direita-esquerda (rotação RL) na subárvore cuja raiz é o nó indicado no parâmetro root. Retorna a nova raiz da subárvore após a rotação ser aplicada.

Atenção! Caso julgue necessário, sua classe da árvore AVL pode ter outros métodos auxiliares para implementar cada operação indicada, inclusive métodos públicos para testes (ver item 3 a seguir).

3. Para testar a sua implementação parcial da árvore AVL, construa as árvores indicadas a seguir.
a) Inserir nós com chaves 1, 2 e 3 (nesta sequência).
b) Inserir nós com chaves 3, 2 e 1 (nesta sequência).
c) Inserir nós com chaves 3, 1 e 2 (nesta sequência).
d) Inserir nós com chaves 1, 3 e 2 (nesta sequência).

Para cada árvore criada, você deve:
- Exibir os dados atualizados de todos os nós (pelo menos quem é o nó pai, o nó filho esquerdo, o nó filho direito e o fator de balanceamento do nó);
- Aplicar a rotação correta para balancear a árvore (nessa primeira parte da implementação da árvore AVL, basta chamar manualmente o método que realiza a rotação);
- Por fim, exibir os dados atualizados de todos os nós.

Observação: Os métodos de rotação indicados no item 2 são privados, já que apenas a classe da AVL é quem deve executar as rotações. No entanto, como as rotações serão feitas manualmente nesta atividade, você pode criar métodos públicos de testes que realizam as rotações.

******* Sem Entrega *******