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 (nomes completo e TIA 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 *Lab1c - 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 *Lab1d - 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.

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). 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.

b) Inserir nós com chaves 3, 2 e 1 (nesta sequência). 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.

c) Inserir nós com chaves 3, 1 e 2 (nesta sequência). 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.

d) Inserir nós com chaves 1, 3 e 2 (nesta sequência). 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.

Entrega

Compacte o código-fonte (somente arquivos *.java) no **formato zip**.

Atenção: O arquivo zip não deve conter arquivos intermediários e/ou pastas geradas pelo compilador/IDE (ex. arquivos *.class, etc.).

Prazo de entrega: via link do Moodle até 15/10/2023 23:59.

Universidade Presbiteriana Mackenzie

ESTRUTURA DE DADOS II

Critérios de avaliação

A nota da atividade é calculada de acordo com os critérios da tabela a seguir.

ITEM AVALIADO	PONTUAÇÃO MÁXIMA
1. Implementação básica da classe que representa um nó usado pela árvore AVL (atributos, construtores, *getters*/*setters*).	0,5
1. Implementação da operação `updateBalanceFactor()`.	0,5
2. Implementação básica da classe que representa uma árvore AVL (subclasse da árvore BST e construtores).	1,0
2. Implementação da operação `rotateLeft()`.	1,25
2. Implementação da operação `rotateRight()`.	1,25
2. Implementação da operação `rotateLeftRight()`.	1,25
2. Implementação da operação `rotateRightLeft()`.	1,25
3. Teste A (1, 2, 3).	0,5
3. Teste B (3, 2, 1).	0,5
3. Teste C (3, 1, 2).	0,5
3. Teste D (1, 3, 2).	0,5
Funcionamento geral do programa, de acordo com o enunciado.	1,0

Tabela 1 - Critérios de avaliação.

A tabela a seguir contém critérios de avaliação que podem **reduzir** a nota final da atividade.

ITEM INDESEJÁVEL	REDUÇÃO DE NOTA
O projeto é cópia de outro projeto.	Projeto é zerado
O projeto usa estruturas de dados oferecida pela linguagem Java.	Projeto é zerado
Há erros de compilação e/ou o programa trava durante a execução[1].	-1,0
Não há identificação do grupo. Não há indicação de referências. Arquivos enviados em formatos incorretos. Arquivos e/ou pastas intermediárias que são criadas no processo de compilação ou pela IDE foram enviadas junto com o código-fonte.	-1,0

Tabela 2 - Critérios de avaliação (redução de nota).

O código-fonte será compilado com o compilador javac (17.0.8) na plataforma Windows da seguinte forma:

```
> javac *.java
```

O código compilado será executado com java (17.0.8) na plataforma Windows da seguinte forma (`<Classe>` deve ser substituído pelo nome da classe que contém o método `public static void main(String[] args)`):

```
> java <Classe>
```

[1] Sobre erros de compilação: considere apenas erros. Não há problema se o projeto tiver *warnings* (embora *warnings* podem avisar sobre possíveis travamentos em tempo de execução, como loop infinito, divisão por zero, etc.).