Relatório Avaliação RA2

Classe Main:

A classe `Main` serve como o ponto de entrada do programa e coordena as operações relacionadas à inserção, busca e exclusão de elementos nas árvores AVL e BST. O programa permite ao usuário escolher a quantidade de elementos a serem inseridos nas árvores, insere esses elementos e fornece funcionalidades para busca e exclusão de elementos. Além disso, o programa mede e compara o tempo necessário para a inserção de elementos em ambas as árvores.

Métodos da Classe Main

1. `main`: O método `main` é o ponto de entrada do programa. Ele cria instâncias de `AVLTree`, `BinarySearchTree`, `Scanner` e `Random`. O programa é estruturado em um loop que exibe um menu principal para o usuário escolher a quantidade de elementos a serem inseridos nas árvores. O loop continua até que o usuário escolha a opção para sair.
2. `insertRandomElements`: Este método insere a quantidade especificada de elementos gerados aleatoriamente nas árvores. Os valores são gerados aleatoriamente e inseridos na árvore AVL.
3. `showMainMenu`: Este método exibe um menu secundário que oferece opções para busca e exclusão de elementos nas árvores AVL e BST. O usuário pode também retornar ao menu anterior.

Funcionalidades do Programa

- **Inserção de Elementos**: O programa permite ao usuário escolher quantos elementos deseja inserir nas árvores (100, 500, 1000, 10000 ou 20000). Esses valores são gerados aleatoriamente e inseridos nas duas árvores.
- **Tempo de Inserção Comparativo**: Após a inserção dos elementos, o programa mede o tempo gasto para inseri-los em ambas as árvores AVL e BST. Esses tempos são exibidos ao usuário, permitindo a comparação do desempenho de ambas as árvores.
- **Busca de Elementos**: O programa oferece a funcionalidade de buscar elementos em ambas as árvores AVL e BST. O usuário fornece um valor a ser pesquisado, e o programa verifica se o elemento está presente em cada árvore.

- **Exclusão de Elementos**: O programa permite ao usuário excluir elementos das árvores AVL e BST. O usuário fornece o valor a ser excluído, e o programa realiza a operação de exclusão em ambas as árvores.
- **Comparação de Desempenho**: O programa fornece ao usuário informações sobre o tempo necessário para inserção de elementos nas duas árvores, permitindo uma comparação do desempenho de cada estrutura de dados para diferentes quantidades de elementos.

Usabilidade do Programa

O programa é projetado de forma amigável para o usuário, fornecendo menus claros e opções para interagir com as árvores AVL e BST. Ele fornece informações úteis para comparar o desempenho das duas estruturas de dados, com foco na inserção de elementos.

Em geral, a classe `Main` coordena a execução do programa e fornece uma interface amigável para comparar o desempenho de árvores AVL e BST na inserção de elementos. Ela é uma parte importante deste sistema de teste e comparação de estruturas de dados.

Classe AVLNode:

A classe `AVLNode` representa um nó em uma Árvore AVL. Cada nó possui as seguintes propriedades:

- `int data`: Armazena o valor dos dados no nó.
- `int height`: Armazena a altura do nó na árvore.
- `AVLNode left`: Referência para o nó filho à esquerda.
- `AVLNode right`: Referência para o nó filho à direita.

O construtor `AVLNode(int data)` é responsável por inicializar um nó com os valores iniciais de dados e altura.

Classe Node:

A classe `Node` representa um nó em uma Árvore Binária de Busca (BST). Cada nó possui as seguintes propriedades:

- `int data`: Armazena o valor dos dados no nó.
- `Node left`: Referência para o nó filho à esquerda.
- `Node right`: Referência para o nó filho à direita.

O construtor `Node(int data)` é responsável por inicializar um nó com os valores iniciais de dados e referências nulas para os filhos.

Classe AVLTree:

A classe `AVLTree` é responsável por representar uma Árvore AVL. Ela possui as seguintes propriedades e métodos:

- Propriedades:
  - `AVLNode root`: Representa a raiz da árvore AVL.
- Métodos Públicos:
  - `void insert(int data)`: Insere um valor na árvore AVL.
  - `boolean search(int data)`: Realiza uma busca na árvore AVL para encontrar um valor específico.
  - `void delete(int data)`: Remove um valor da árvore AVL.
  - `void inOrderTraversal()`: Percorre a árvore em ordem (inOrder) e imprime os valores.
  - `void preOrderTraversal()`: Percorre a árvore em pré-ordem (preOrder) e imprime os valores.
  - `void postOrderTraversal()`: Percorre a árvore em pós-ordem (postOrder) e imprime os valores.
- Métodos Privados:
  - `AVLNode insertRec(AVLNode root, int data)`: Realiza a inserção recursiva de um valor na árvore AVL.
  - `AVLNode searchRec(AVLNode root, int data)`: Realiza a busca recursiva na árvore AVL.
  - `AVLNode deleteRec(AVLNode root, int data)`: Realiza a exclusão recursiva de um valor da árvore AVL.
  - `int height(AVLNode node)`: Retorna a altura de um nó na árvore.
  - `int getBalance(AVLNode node)`: Calcula o fator de balanceamento de um nó.
  - `AVLNode rightRotate(AVLNode y)`: Realiza uma rotação para a direita em um nó.
  - `AVLNode leftRotate(AVLNode x)`: Realiza uma rotação para a esquerda em um nó.

- `AVLNode minValueNode(AVLNode node)`: Encontra o nó com o menor valor na árvore.

Classe BinarySearchTree:

A classe `BinarySearchTree` representa uma Árvore Binária de Busca (BST). Ela possui propriedades e métodos semelhantes à classe `AVLTree`, mas não realiza o balanceamento automático. As principais diferenças estão nos métodos `insert` e `delete`, onde não há reequilíbrio.