Nome: Luís Felipe Yoshio Sato, Thiago Vinícius Pereira Borges.

- Código:

Classe Node:

Esta classe representa um nó individual da árvore. Cada nó possui:

Uma informação (ou valor) armazenada como um número inteiro.
Um ponteiro para o nó filho à direita.
Um ponteiro para o nó filho à esquerda.

A classe possui métodos getter e setter para cada um desses atributos.

Classe Da Árvore e Árvore Balanceada:

Esta é a classe principal que representa a árvore balanceada. Ela possui:

Um atributo raizInicial que é a raiz da árvore.
Métodos para inserir, buscar e remover nós.
Métodos para calcular o tempo dessas operações.

Métodos auxiliares para realizar rotações e balanceamento dos nós.

Métodos principais:

inserir(int valor): Adiciona um novo nó com o valor especificado à árvore. Após a inserção, a árvore é balanceada.

buscar(int elemento): Busca um nó com o valor especificado na árvore. Retorna o nó se encontrado, caso contrário, retorna nulo.

removerN(Node raiz, int valor): Remove um nó com o valor especificado da árvore e, em seguida, balanceia a árvore.

encontrarMenor(Node raiz): Encontra o menor valor na subárvore especificada. Métodos de balanceamento:

altura(Node no): Retorna a altura de um nó específico.

FB(Node no): Calcula o Fator de Balanceamento (FB) de um nó, que é a diferença entre as alturas das subárvores esquerda e direita.

rotacaoDireita(Node raiz) e rotacaoEsquerda(Node raiz): São métodos que realizam rotações simples nos nós para ajudar no balanceamento da árvore.

balancear(Node raiz): Balanceia a árvore após inserções ou remoções.

Métodos de tempo:

tempoInserir(int[] valores), tempoBuscar(int valor), e tempoRemover(int valor): Calculam o tempo que leva para inserir, buscar e remover um nó, respectivamente.

Análise

Percebemos no comparativo entre elas, que a busca de um valor nas árvores tende a ser mais rápido na árvore AVL, no qual ela já foi previamente balanceada. Em contrapartida, devido ao rebalanceamento, os métodos de inserir e remover são mais demorados, custando um tempo considerável no desempenho.

Como ambas tem seus pontos positivos e negativos, a escolha de uso de cada uma dela depende da necessidade do usuário. No caso da operação de busca ser de maior relevância, então se torna mais coerente o uso da árvore AVL, caso o desempenho da inserção e remoção forem mais frequentes que a de busca, então a árvore binária seria mais adequada.

```
Arvore Binaria
Tempo para inserir 100 números: 0.375085 ms
Tempo para buscar o número 12522: 0.0038 ms
Tempo para remover o número 7797: 0.003918 ms

Arvore AVL
Tempo para inserir 100 números: 0.713124 ms
Tempo para buscar o número 12522: 0.002571 ms
Tempo para remover o número 7797: 0.032687 ms

Arvore Binaria
Tempo para inserir 500 números: 0.987453 ms
Tempo para buscar o número 3875: 0.002788 ms
Tempo para remover o número 1892: 0.00666 ms

Arvore AVL
Tempo para inserir 500 números: 7.369216 ms
Tempo para buscar o número 3875: 0.004069 ms
Tempo para remover o número 1892: 0.026894 ms

Arvore Binaria
Tempo para inserir 1000 números: 2.759564 ms
Tempo para buscar o número 19785: 0.00296 ms
Tempo para remover o número 4320: 0.003114 ms

Arvore AVL
Tempo para inserir 1000 números: 5.129971 ms
Tempo para buscar o número 19785: 0.003018 ms
Tempo para remover o número 4320: 0.022669 ms

Arvore Binaria
Tempo para inserir 10000 números: 4.68374 ms
Tempo para buscar o número 15358: 0.005244 ms
Tempo para remover o número 6429: 0.004969 ms

Arvore AVL
Tempo para inserir 10000 números: 402.25714 ms
Tempo para buscar o número 15358: 0.007027 ms
Tempo para remover o número 6429: 0.4201 ms

Arvore Binaria
Tempo para inserir 20000 números: 2.867661 ms
Tempo para buscar o número 2138: 0.005205 ms
Tempo para remover o número 16613: 0.003465 ms

Arvore AVL
Tempo para inserir 20000 números: 1075.649969 ms
Tempo para buscar o número 2138: 0.004506 ms
Tempo para remover o número 16613: 0.703218 ms
```