**Integrantes: Diego Henrique Zanella, Gabriel Hortmann de Campos Bueno, Jonathan Domingos Rodrigues**

**Descrição dos códigos**

**Árvore Binária**

* Classe ArvoreBinaria:
  + Esta classe representa uma árvore de busca binária.
  + Ela possui um construtor que inicializa a raiz da árvore como nula.
  + Inclui métodos para inserir (inserir), remover (remover) e buscar (buscar) elementos na árvore.
  + Também contém métodos auxiliares privados para ajudar nas operações de inserção (inserir), remoção (remover) e encontrar o valor mínimo (valorMinimo) na árvore.
* Classe Node:
  + Esta classe representa um nó na árvore binária.
  + Cada nó contém um valor inteiro (valor) e referências para seus nós filhos esquerdo e direito (esquerda e direita).
* Classes de Teste de Desempenho:
  + Existem três classes de teste de desempenho: InserirPerformanceTeste, BuscaPerformanceTeste e RemocaoPerformanceTeste.
  + Cada classe realiza testes de desempenho para uma operação específica na árvore (inserção, busca ou remoção).
  + Elas criam uma árvore binária com um número especificado de elementos aleatórios, medem o tempo necessário para realizar a operação em cada elemento e, em seguida, exibem os resultados.
* Classe ArvoreMain:
  + Esta é a classe principal que executa os testes de desempenho para inserção, busca e remoção.
  + Ela chama os métodos de teste das respectivas classes de teste de desempenho.

**Árvore AVL**

* Classe ArvoreAVL
  + Representa uma Árvore AVL (Adelson-Velsky e Landis), uma árvore de busca binária autoequilibrada.
  + Inclui métodos para inserção, remoção e busca de elementos na árvore.
  + Contém métodos auxiliares para calcular a altura e o fator de equilíbrio dos nós.
  + Implementa rotações à direita e à esquerda para manter o equilíbrio.
* Classe Node
  + Representa um nó na Árvore AVL.
  + Cada nó possui um valor inteiro, uma altura e referências para seus nós filhos à esquerda e à direita.
* Classes de Teste de Desempenho
  + Existem três classes de teste de desempenho: InserirPerformanceTesteAVL, BuscaPerformanceTesteAVL e RemocaoPerformanceTesteAVL.
  + Essas classes medem o tempo necessário para operações de inserção, busca e remoção na Árvore AVL.
  + Criam uma Árvore AVL com elementos aleatórios, realizam as operações e registram o tempo decorrido.
* Classe ArvoreAVLMain
  + Classe principal que coordena a execução dos testes de desempenho.
  + Chama os métodos de teste nas classes de teste de desempenho para inserção, busca e remoção na Árvore AVL.

**Recomendações**

**Árvore Binária**

* É uma estrutura de dados mais leve, então é melhor para quem tenha restrições de espaço ou custo operacional;
* É mais simples de implementar e tem um desempenho aceitável em cenários onde o balanceamento não é crítico.

**Árvore AVL**

* A árvore AVL garante uma altura balanceada, o que mantém um desempenho elevado para operações de busca, mostrando uma maior diferença com uma grande quantidade de dados;
* Ela é automaticamente balanceada durante a inserção, garantindo tempos de execução previsíveis;
* É recomendada para aplicações de tempo real por apresentar por ter um tempo de execução previsível, garantindo um tempo consistente em todas as operações.

**Tempo de execução**

**Inserção:**

* **Árvore Binária**
  + Tempo médio: 1 ms (para 100 elementos)
  + Tempo médio: 2 ms (para 500 elementos)
  + Tempo médio: 1 ms (para 1.000 elementos)
  + Tempo médio: 7 ms (para 10.000 elementos)
  + Tempo médio: 12 ms (para 20.000 elementos)
* **Árvore AVL**
  + Tempo médio: 3 ms (para 100 elementos)
  + Tempo médio: 1 ms (para 500 elementos)
  + Tempo médio: 2 ms (para 1.000 elementos)
  + Tempo médio: 10 ms (para 10.000 elementos)
  + Tempo médio: 15 ms (para 20.000 elementos)

A árvore binária tem um desempenho melhor na inserção, com a diferença ficando mais perceptível à medida que a quantidade de dados aumenta;

A árvore AVL é mais lenta por realizar as funções de balanceamento da árvore, o que pode ser crucial em cenários em que a busca é uma operação mais frequente.

**Busca:**

* **Árvore Binária**
  + Tempo médio: 0 ms (para 100 elementos)
  + Tempo médio: 1 ms (para 500 elementos)
  + Tempo médio: 1 ms (para 1.000 elementos)
  + Tempo médio: 3 ms (para 10.000 elementos)
  + Tempo médio: 8 ms (para 20.000 elementos)
* **Árvore AVL**
  + Tempo médio: 0 ms (para 100 elementos)
  + Tempo médio: 1 ms (para 500 elementos)
  + Tempo médio: 1 ms (para 1.000 elementos)
  + Tempo médio: 3 ms (para 10.000 elementos)
  + Tempo médio: 8 ms (para 20.000 elementos)

Os tempos médios de busca são bastante semelhantes entre a árvore binária e a árvore AVL. Isso ocorre porque a estrutura da árvore não afeta significativamente o desempenho da operação de busca. Porém, a árvore AVL mantém o balanceamento da árvore, o que pode resultar em um melhor desempenho se houver uma maior quantidade de dados.

**Remoção**

* **Árvore Binária**
  + Tempo médio: 1 ms (para 100 elementos)
  + Tempo médio: 1 ms (para 500 elementos)
  + Tempo médio: 1 ms (para 1.000 elementos)
  + Tempo médio: 4 ms (para 10.000 elementos)
  + Tempo médio: 10 ms (para 20.000 elementos)
* **Árvore AVL**
  + Tempo médio: 1 ms (para 100 elementos)
  + Tempo médio: 2 ms (para 500 elementos)
  + Tempo médio: 1 ms (para 1.000 elementos)
  + Tempo médio: 7 ms (para 10.000 elementos)
  + Tempo médio: 14 ms (para 20.000 elementos)

Assim como na inserção, a árvore binária tem um desempenho melhor na remoção, exatamente pelo mesmo motivo, já que a árvore AVL mantém a árvore balanceada mesmo após a remoção de dados.