Trabalho - Análise de Desempenho de Estruturas de Dados em Java

Objetivo:

Comparar o desempenho de diferentes estruturas de dados (vetores, árvores binárias e árvores AVL) em operações de inserção e busca, utilizando diversas ordens de inserção e tamanhos de conjuntos de dados.

Descrição Geral:

Você deverá implementar:

1. **Estruturas de Dados:**
   - Vetor
   - Árvore Binária
   - Árvore AVL (Árvore Binária de Busca Balanceada)
2. **Tamanhos dos Conjuntos de Dados:**
   - 100 elementos
   - 1.000 elementos
   - 10.000 elementos
3. **Ordens de Inserção dos Dados:**
   - Ordenada (em ordem crescente)
   - Inversamente Ordenada (em ordem decrescente)
   - Aleatória
4. **Operações a serem Realizadas:**
   - **Inserção** dos dados em cada estrutura, considerando as diferentes ordens de inserção.
   - **Busca** de elementos em cada estrutura:
     - Primeiro elemento
     - Último elemento
     - Elemento do meio
     - Valores aleatórios
     - Valor inexistente na estrutura
5. **Métodos de Busca em Vetores:**
   - Busca Sequencial
   - Busca Binária (apenas em vetores ordenados)
6. **Algoritmos de Ordenação para Vetores:**
   - Implementar pelo menos duas técnicas de ordenação:

    - Uma básica (exemplo: Bubble Sort)
    - Uma aprimorada (exemplo: QuickSort ou MergeSort)
  - Medir o tempo de execução de cada algoritmo.
7. **Implementação:**
  - Todos os métodos e estruturas de dados devem ser implementados pelo aluno, sem utilizar bibliotecas ou funções prontas para essas finalidades.

Instruções Detalhadas:

A. Implementação das Estruturas e Algoritmos:

1. **Estruturas de Dados:**
  - **Vetor:**
    - Implementar operações de inserção e métodos de busca (sequencial e binária).
  - **Árvore Binária:**
    - Implementar operações de inserção e busca.
  - **Árvore AVL:**
    - Implementar operações de inserção e busca, garantindo o balanceamento após cada inserção.
2. **Algoritmos de Ordenação:**
  - Escolher e implementar dois algoritmos:
    - **Básico:** Por exemplo, Bubble Sort.
    - **Aprimorado:** Por exemplo, QuickSort ou MergeSort.
  - Aplicar os algoritmos aos vetores para ordená-los e permitir a busca binária.

B. Realização dos Testes:

1. **Preparação dos Conjuntos de Dados:**
  - Gerar conjuntos de dados com os tamanhos especificados (100, 1.000 e 10.000 elementos).
  - Assegurar que os dados sejam números ou outro tipo de dados comparáveis.
2. **Ordens de Inserção:**
  - **Ordenada:** Inserir os dados em ordem crescente.
  - **Inversamente Ordenada:** Inserir os dados em ordem decrescente.
  - **Aleatória:** Inserir os dados em ordem aleatória (utilizar um gerador de números aleatórios).
3. **Execução dos Testes:**
  - **Inserção:**
    - Medir o tempo de inserção dos dados em cada estrutura, para cada ordem de inserção.
  - **Busca:**
    - Realizar buscas pelos seguintes elementos e medir o tempo de cada busca:
      - Primeiro elemento

        - Último elemento
        - Elemento do meio
        - Três valores aleatórios presentes no conjunto
        - Um valor inexistente
    - **Repetições:**
        - Executar cada teste pelo menos **5 vezes** e calcular a média dos tempos para obter resultados mais confiáveis.
4. **Ordenação dos Vetores:**
    - Aplicar os dois algoritmos de ordenação implementados aos vetores.
    - Medir o tempo que cada algoritmo leva para ordenar os vetores de diferentes tamanhos.

C. Coleta e Análise dos Dados:

1. **Registro dos Tempos:**
    - Anotar todos os tempos medidos em tabelas organizadas por estrutura, tamanho do conjunto e ordem de inserção.
2. **Cálculo das Médias:**
    - Calcular a média dos tempos das 5 execuções para cada teste.
3. **Comparação dos Resultados:**
    - Comparar os tempos de inserção e busca entre as diferentes estruturas de dados.
    - Comparar o desempenho dos algoritmos de ordenação.
4. **Visualização:**
    - Criar gráficos para ilustrar as diferenças de desempenho.

D. Elaboração do Relatório:

1. **Metodologia:**
    - Descrever como os testes foram conduzidos.
    - Explicar a geração dos conjuntos de dados e as ordens de inserção.
    - Detalhar como as medições de tempo foram realizadas.
2. **Resultados:**
    - Apresentar as tabelas com os tempos medidos.
    - Incluir gráficos para facilitar a interpretação dos dados.
3. **Análise:**
    - Discutir os resultados obtidos.
    - Explicar por que certas estruturas ou algoritmos tiveram melhor desempenho.
    - Relacionar os resultados com a complexidade teórica das operações.
4. **Conclusão:**
    - Resumir as principais observações.

- Destacar aprendizados e possíveis melhorias.

5. **Referências:**
   - Listar quaisquer fontes consultadas.

E. Submissão do Código Fonte:

1. **Repositório Git:**
   - Subir todo o código em um repositório Git.
   - Incluir um **README** com:
     - Instruções de compilação e execução.
     - Descrição dos arquivos e estruturas do projeto.
     - Qualquer dependência necessária.
2. **Documentação do Código:**
   - Utilizar nomes de variáveis e funções que reflitam sua funcionalidade.

Considerações Finais:

- **Precisão nas Medidas:**
  - Certificar-se de que as medições sejam consistentes.
- **Validação:**
  - Antes de realizar os testes principais, verificar se as implementações estão corretas com conjuntos de dados menores.
- **Organização:**
  - Manter uma estrutura de pastas organizada no repositório.
  - Separar o código por classes.