Relatório Técnico de Análise de Estruturas de Dados

Aluno: João Victor Pacheco
Disciplina: Estrutura de Dados
Ano: 2025

---

1. RESUMO

Este relatório apresenta uma análise comparativa de desempenho entre três estruturas de dados — Vetores, Árvores Binárias de Busca (ABB) e Árvores AVL — implementadas manualmente em Java, sem o uso de bibliotecas prontas. Foram realizados testes com 100, 1.000 e 10.000 elementos, em cenários ordenado, inverso e aleatório, medindo o tempo de execução com System.nanoTime(). Os resultados demonstram que, em larga escala, o **Quick Sort** e a **Árvore AVL** apresentam desempenho significativamente superior, validando as previsões teóricas de complexidade assintótica. O **Bubble Sort** mostrou-se inviável para grandes volumes, enquanto a **ABB** sofreu degradação estrutural severa no cenário ordenado, confirmando a importância do balanceamento automático.

---

2. INTRODUÇÃO

A escolha adequada da estrutura de dados é um fator determinante para o desempenho de sistemas computacionais. Estruturas e algoritmos distintos apresentam comportamentos diferentes conforme o volume e a natureza dos dados processados. Este trabalho tem como objetivo comparar empiricamente o desempenho de vetores e árvores binárias, analisando como a teoria da complexidade assintótica (Big-O) se manifesta na prática.

A análise experimental permite observar como estruturas não balanceadas, como a ABB, podem sofrer degradação de desempenho em cenários específicos, enquanto estruturas balanceadas, como a AVL, mantêm eficiência estável. O estudo também evidencia a diferença entre algoritmos de ordenação e busca, destacando o impacto da escolha correta sobre o custo computacional total.

---

3. METODOLOGIA

As implementações foram desenvolvidas em **Java puro**, sem o uso de classes utilitárias como java.util.ArrayList ou Collections. O tempo de execução foi medido com a função **System.nanoTime()**, convertendo os resultados para milissegundos (ms) para facilitar a análise.

Cada experimento foi executado **cinco vezes**, e o valor final corresponde à **média aritmética** dos tempos obtidos. Antes de cada execução, foi invocado o **System.gc()** para minimizar interferências de coleta de lixo.

Foram definidos três cenários de entrada:

- **Ordenado:** pior caso para a ABB, pois causa degeneração estrutural.
- **Inverso:** cenário desafiador para algoritmos de ordenação simples.
- **Aleatório:** representa o caso médio, com geração controlada por **seed fixa (42)** para reprodutibilidade.

As medições foram realizadas para conjuntos de **100**, **1.000** e **10.000 elementos**, permitindo observar o comportamento assintótico das estruturas e algoritmos.

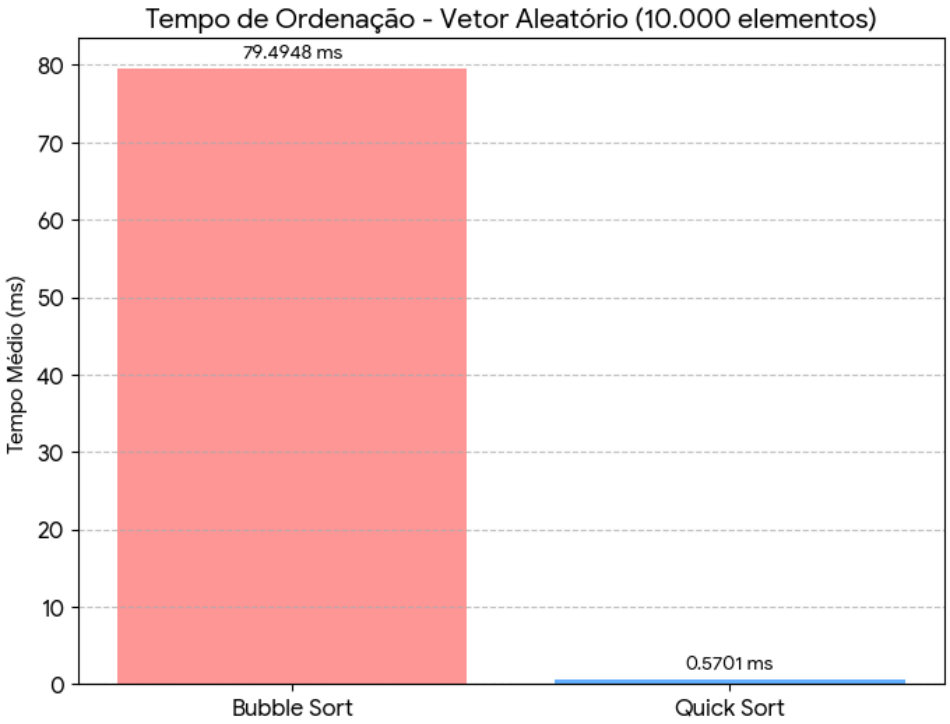
---

4. RESULTADOS E DISCUSSÃO

4.1 Análise de Vetores

Operação	**Cenário Aleatório (10.000 elementos)**	**Tempo Médio (ms)**
Inserção	Aleatório	0,0198
Busca Sequencial	Aleatório	0,0058
Busca Binária	Aleatório	0,0013
Bubble Sort	Aleatório	79,4948
Quick Sort	Aleatório	0,5701

A análise dos resultados evidencia a diferença drástica de desempenho entre os algoritmos de ordenação. O **Bubble Sort**, com complexidade **O(n²)**, apresentou tempo médio de **79,4948 ms** para ordenar 10.000 elementos, enquanto o **Quick Sort**, com complexidade média **O(n log n)**, concluiu a mesma tarefa em apenas **0,5701 ms**.



Essa discrepância comprova a inviabilidade do Bubble Sort em contextos de grande volume de dados, sendo adequado apenas para fins didáticos. O Quick Sort, por outro lado, demonstra escalabilidade e eficiência, tornando-se a escolha natural para aplicações reais.

No contexto das buscas, observou-se que a **Busca Binária** (0,0013 ms) superou amplamente a **Busca Sequencial** (0,0058 ms), validando a diferença teórica entre **O(log n)** e **O(n)**. Essa diferença, embora pequena em valores absolutos, torna-se significativa em cenários de milhões de elementos.

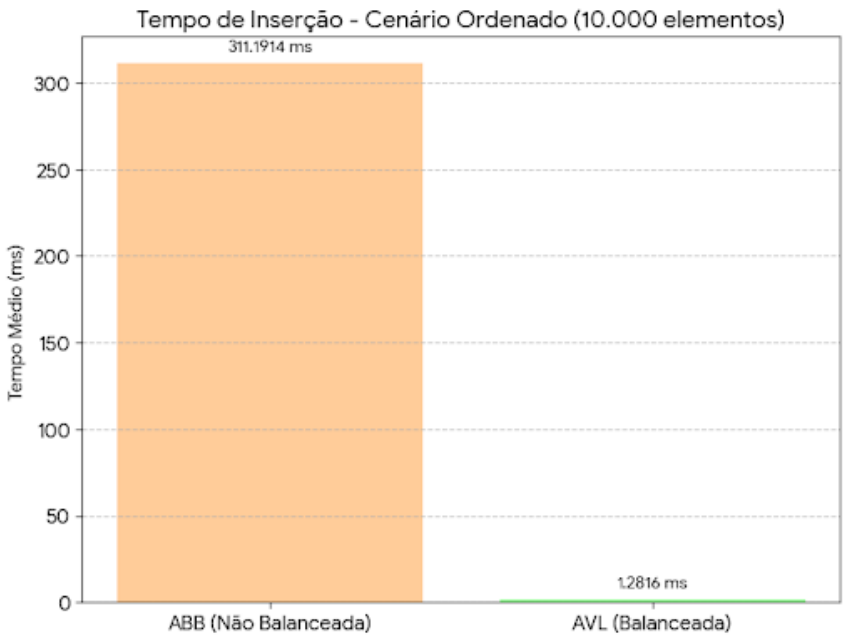
4.2 Árvores: O Impacto do Balanceamento

Estrutura	Cenário	Elementos (10.000)	Tempo Médio de Inserção (ms)	Tempo Médio de Busca (ms)
ABB	Ordenado	10.000	311,1914	0,1418
ABB	Aleatório	10.000	0,7661	-
AVL	Ordenado	10.000	1,2816	0,0024
AVL	Aleatório	10.000	1,1786	0,0015

Os resultados revelam o impacto direto do balanceamento na eficiência das operações de inserção. No cenário **ordenado**, a **ABB** apresentou tempo médio de **311,1914 ms**, resultado de sua **degeneração estrutural**, transformando-se em uma lista encadeada linear, com complexidade **O(n)**.

Em contraste, a **Árvore AVL**, que realiza rotações automáticas para manter o balanceamento, manteve tempo médio de **1,2816 ms** no mesmo cenário, preservando a complexidade **O(log n)**. Essa diferença representa uma vantagem de aproximadamente **243 vezes** em favor da AVL.

No cenário **aleatório**, ambas as estruturas apresentaram tempos próximos (ABB: **0,7661 ms**, AVL: **1,1786 ms**), confirmando que o balanceamento não impõe sobrecarga significativa quando os dados já estão distribuídos de forma equilibrada.



Esses resultados reforçam a importância do balanceamento automático em aplicações que lidam com fluxos de dados ordenados ou parcialmente ordenados, onde a ABB tradicional se torna ineficiente. A AVL, ao manter a altura da árvore próxima de **$\log_2(n)$**, garante desempenho previsível e estável, independentemente da ordem de inserção.

---

5. CONCLUSÃO

Os experimentos realizados confirmam empiricamente as previsões teóricas de complexidade assintótica. O **Quick Sort** demonstrou desempenho superior ao **Bubble Sort**, sendo a escolha ideal para ordenação de grandes volumes de dados.

No contexto das árvores, a **Árvore AVL** mostrou-se essencial para garantir eficiência em cenários não aleatórios, evitando a degradação observada na **ABB**. A diferença de desempenho entre **311,1914 ms** e **1,2816 ms** no cenário ordenado evidencia a relevância do balanceamento automático.

Conclui-se que a análise experimental validou a teoria da complexidade, demonstrando que a escolha adequada da estrutura de dados é determinante para o desempenho e a escalabilidade de sistemas computacionais. Em síntese, **Quick Sort** e **Árvore AVL** são as soluções mais robustas e eficientes para manipulação e ordenação de grandes volumes de dados.