Relatório técnico

- **Introdução** -
Este relatório apresenta um estudo comparativo de desempenho entre três estruturas de dados — Vetor, Árvore Binária de Busca (ABB) e Árvore AVL — além de dois algoritmos de ordenação: Bubble Sort e Quick Sort.

Os testes foram realizados com conjuntos de tamanhos 100, 1.000 e 10.000, em três ordens diferentes: ordenado, inverso e aleatório.

O objetivo é verificar, na prática, como o comportamento de cada estrutura e algoritmo se relaciona com a complexidade teórica estudada em Estrutura de Dados.

- **Metodologia** -
1. Geração dos conjuntos de dados

Foram gerados três conjuntos para cada tamanho:

Ordenado: 1, 2, 3, ..., n
Inversamente ordenado: n, n-1, ..., 1
Aleatório: valores embaralhados

Os tamanhos utilizados foram:
100 elementos
1.000 elementos
10.000 elementos

2. Estruturas e algoritmos implementados

Vetor: inserção sequencial, busca sequencial e busca binária
ABB: inserção e busca
AVL: inserção com balanceamento automático e busca
Bubble Sort: algoritmo simples
QuickSort: algoritmo avançado

3. Medição de desempenho

Para cada estrutura e cenário foram medidos:
Inserção
Tempo total para inserir todos os elementos
Média de 5 execuções
Busca

Foram realizadas buscas para:
Primeiro elemento
Último elemento
Elemento do meio
Três elementos aleatórios existentes
Um elemento inexistente
Média de 5 execuções
Ordenação (vetor apenas)
Tempo para ordenar usando Bubble Sort
Tempo para ordenar usando QuickSort
Média de 5 execuções

4. Ferramentas utilizadas

Java 17
IDE jGRASP
Medição com System.nanoTime()
Resultados impressos no terminal

- **Resultados** -
1. Inserção (ms)Tabela 1 — Tempo médio de inserção

Tamanho/ordem	Vetor	ABB	AVL
100 ordenado	5,6493	11,2278	9,4405
100 inverso	0,0034	0,1379	0,2015
100 aleatório	0,0048	0,0291	0,0391

1.000 ordenado	0,0410	4,1620	0,1097
1.000 inverso	0,0328	3,5321	0,0872
10.000 ordenado	0,2593	387,3354	0,8926
10.000 inverso	0,0632	343,3890	0,7410
10.000 aleatório	0,1114	2,2160	2,0011

2 Busca (ms)Tabela 2 — Tempo médio de busca

Tamanho/ordem	Vetor Seq	Vetor Bin	ABB	AVL
100 ordenado	0,0044	0,0016	0,0130	0,0021
100 inverso	0,0138	—	0,0066	0,0057
100 aleatório	0,0065	—	0,0012	0,0055
1.000 ordenado	0,0411	0,0024	0,0524	0,0010
1.000 inverso	0,0108	—	0,0190	0,0015
1.000 aleatório	0,0086	—	0,0025	0,0011
10.000 ordenado	0,2053	0,0037	0,2027	0,0036
10.000 inverso	0,0110	—	0,1890	0,0016
10.000 aleatório	0,0072	—	0,0034	0,0038

3 Ordenação (ms)Tabela 3 — Tempo médio dos algoritmos de ordenação

- Análise dos Resultados -

Tamanho/ordem	Bubble	Quick
100 ordenado	4,3427	3,7850
100 inverso	0,2719	0,0441
100 aleatório	0,1693	0,0077
1.000 ordenado	0,0204	0,6120
1.000 inverso	1,3896	0,2935
1000 aleatório	1,8116	0,0369

10.000 ordenado	0,0039	38,5546
10.000 inverso	69,7587	26,4987
10.000 aleatório	86,1853	0,5041

1. Inserção Vetor

Tem desempenho extremamente estável e rápido para todos os tamanhos.

Mesmo com 10.000 elementos, mantém tempos abaixo de 0,3 ms.

O caso dos 100 ordenados, com 5,6 ms, foi um pico isolado de overhead do Java.

ABB

O desempenho piora drasticamente quando os dados vêm ordenados ou inversos, confirmando a degeneração da ABB para uma lista.
Exemplo claro:

10.000 ordenado → 387 ms
10.000 inverso → 343 ms

Em dados aleatórios, o desempenho melhora muito (apenas 2,2 ms).

AVL

Mantém desempenho estável e previsível.
Mesmo em dados ordenados (pior caso teórico), continua eficiente:

10.000 ordenado → 0,89 ms

Como esperado, é um pouco mais lenta que ABB em dados aleatórios pela necessidade de rebalanceamento.

2. BuscaVetor

Sequencial cresce proporcional ao tamanho → coerente com O(n).
Binária tem tempos muito baixos, provando o O(log n).

ABB
Sofre muito em ordenado/inverso:

10.000 ordenado → 0,2027 ms, muito mais que AVL.

Excelente em aleatório (0,0034 ms), mostrando árvore equilibrada por sorte.

AVL

Melhor desempenho geral.
Sempre entre 0,001 e 0,005 ms.
Confirma o comportamento garantido O(log n).

3. OrdenaçãoBubble Sort

Muito lento em vetores grandes:
10.000 inverso → 69 ms
10.000 aleatório → 86 ms

Confirma claramente a complexidade $O(n^2)$.
QuickSort
Extremamente rápido nos casos aleatórios:
10.000 aleatório → apenas 0,50 ms

Entretanto, apresentou um comportamento inesperado em 10.000 ordenado (38 ms), indicando que sua implementação caiu em caso ruim (pior caso pivot).

- Conclusão -

Com base nos testes e análises, concluímos que:

Vetor é excelente para inserções e aceitável para buscas, principalmente a binária.

ABB só funciona bem quando os dados não estão ordenados; caso contrário, degrada totalmente.

AVL é a estrutura mais estável e eficiente, o que confirma sua vantagem teórica.

Bubble Sort é inviável para grandes volumes.

QuickSort é disparado o mais eficiente, exceto em casos ordenados dependendo da escolha do pivô.

Os resultados práticos confirmam a teoria estudada em Estruturas de Dados.