RELATÓRIO DE ANÁLISE DE DESEMPENHO DE ESTRUTURAS DE DADOS

1. METODOLOGIA

Este trabalho visa comparar o desempenho de três estruturas de dados fundamentais: Vetor, Árvore Binária de Busca (ABB) e Árvore AVL. Adicionalmente, analisa-se a eficiência de algoritmos de busca e ordenação implementados manualmente em Java.

1.1 Geração de Dados e Implementação

Os testes foram realizados utilizando conjuntos de dados de 100, 1.000 e 10.000 números inteiros. Para garantir a validade estatística e cobrir diferentes cenários de complexidade, utilizaram-se três ordens de geração:

Ordenada: Inserção sequencial crescente (0, 1, 2...).

Inversa: Inserção sequencial decrescente.

Aleatória: Números gerados pseudo-aleatoriamente através da classe java.util.Random.

1.2 Métricas e Ambiente de Teste

A medição de tempo foi realizada utilizando a função System.nanoTime(), que oferece precisão em nanossegundos. A metodologia seguiu os seguintes critérios:

Repetições: Cada cenário foi executado 5 vezes, sendo o resultado final a média aritmética dessas execuções.

Busca: O teste de busca incluiu uma bateria mista de alvos (primeiro, último, meio, aleatórios e inexistente) para simular um uso real.

Unidade de Medida: Os resultados são apresentados em nanossegundos (ns) para manter a precisão dos dados coletados.

2. RESULTADOS

I. Tamanho 100:

- Inserção Média (ns):

ORDEM	VETOR	ABB	AVL
ORDENADA	4.280	162.900	149.660
INVERSA	2.980	60.200	25.340
ALEATÓRIA	2.880	8.420	16.740

- Busca Média (ns):

ORDEM	VETOR	ABB	AVL
ORDENADA	1.097	1.914	411
INVERSA	794	285	400
ALEATÓRIA	862	120	108

- Ordenação média (ns):

ORDEM	BUBBLE	QUICK
ORDENADA	118.580	110960
INVERSA	222.000	15020
ALEATÓRIA	104.980	6020

II. Tamanho 1000:

- Inserção Média (ns):

ORDEM	VETOR	ABB	AVL
ORDENADA	26.180	1.718.120	156.620
INVERSA	26.160	1.332.380	133.520
ALEATÓRIA	22.920	80.340	139.200

- Busca Média (ns):

ORDEM	VETOR	ABB	AVL
ORDENADA	3.240	2.165	245
INVERSA	1.360	1.965	285
ALEATÓRIA	2.348	280	160

- Ordenação média (ns):

ORDEM	BUBBLE	QUICK
ORDENADA	534.440	806.100
INVERSA	248.960	430.600

ALEATÓRIA	468.580	37.580

III. Tamanho 10.000:

- Inserção Média (ns):

ORDEM	VETOR	ABB	AVL
ORDENADA	74.140	197.713.560	697.660
INVERSA	63.780	131.987.740	1.174.260
ALEATÓRIA	50.760	854.980	1.296.420

- Busca Média (ns):

ORDEM	VETOR	ABB	AVL
ORDENADA	1.511	8.771	811
INVERSA	1.597	14.474	960
ALEATÓRIA	1.065	637	428

- Ordenação média (ns):

ORDEM	BUBBLE	QUICK
ORDENADA	9.449.900	30.471.620
INVERSA	27.107.460	24.493.500
ALEATÓRIA	33.418.420	446.640

3. ANÁLISE DOS RESULTADOS

3.1.Inserção: O Impacto do Balanceamento

Os resultados de inserção evidenciam claramente a limitação da Árvore Binária de Busca (ABB) simples.

ABB (Pior Caso): No teste com 10.000 elementos ordenados, a ABB registou um tempo extremamente elevado de ~197 milhões de ns (vs. ~0,7 milhões da AVL). Isto ocorre porque, ao inserir dados ordenados, a ABB degenera numa lista encadeada, elevando a complexidade de inserção para O(n).

AVL (Eficiência): A AVL manteve tempos estáveis em todas as ordens. Embora seja ligeiramente mais lenta que a ABB em dados aleatórios (devido ao custo de cálculo de altura e rotações), ela compensa drasticamente nos cenários ordenados, mantendo a complexidade O(log n).

Vetor: A inserção no vetor foi consistentemente a mais rápida, pois trata-se apenas de adicionar um valor ao final do array (O(1)), sem a sobrecarga de alocação de nós ou ponteiros.

3.2.Busca: Eficiência Logarítmica

Na busca, a AVL demonstrou superioridade clara.

Com 10.000 elementos aleatórios, a AVL (428 ns) foi cerca de 2,5 vezes mais rápida que a Busca Sequencial no Vetor (1.065 ns).

A busca na ABB sofreu novamente no cenário ordenado (8.771 ns), comportando-se pior que o vetor, pois precisou percorrer a árvore desbalanceada linearmente.

3.3.Ordenação: O Caso Curioso do QuickSort

A comparação entre Bubble Sort e Quick Sort revelou comportamentos teóricos importantes:

Cenário Aleatório: O Quick Sort brilhou, sendo ~74 vezes mais rápido que o Bubble Sort para 10.000 elementos (446 mil ns vs 33 milhões ns). Isto confirma a eficiência média de O(n log n) do Quick Sort contra $O(n^2)$ do Bubble Sort.

Cenário Ordenado (Pior Caso): Observou-se que o Quick Sort foi mais lento que o Bubble Sort nos dados ordenados de tamanho 10.000 (~30ms vs ~9ms).

Explicação: A implementação utilizada do Quick Sort escolhe o último elemento como pivô. Em vetores já ordenados, isso gera o pior caso do algoritmo ($O(n^2)$), causando recursão profunda. O Bubble Sort, apesar de também ser $O(n^2)$, realiza apenas comparações sem trocas neste cenário, o que, devido à simplicidade das operações de baixo nível e ausência de overhead de chamadas recursivas, acabou por ser mais rápido nesta execução específica.

4. CONCLUSÃO

Este experimento comprovou que não existe uma estrutura de dados universalmente perfeita; a escolha depende do contexto dos dados.

Vetor: É imbatível para inserção e eficiente para buscas em volumes pequenos, mas a busca sequencial e a ordenação (Bubble Sort) tornam-se proibitivas em grandes volumes.

ABB: É simples e rápida para dados aleatórios, mas perigosa para dados reais que possam vir parcialmente ordenados, degradando o seu desempenho drasticamente.

AVL: Apresentou-se como a estrutura mais robusta. O custo extra de processamento nas inserções (rotações) é insignificante comparado com o ganho de performance na busca e a garantia de que o sistema nunca enfrentará o "pior caso" degenerado, independentemente da ordem de entrada dos dados