Relatório de Desempenho dos Algoritmos de Ordenação

1. Introdução

Este relatório analisa e compara o desempenho de cinco algoritmos de ordenação: Bubble Sort, Insertion Sort, Quick Sort, Merge Sort e Heap Sort. Os algoritmos foram avaliados com vetores de tamanhos 10^2, 10^3, 10^4, 10^5 e 10^6 elementos, e os tempos de execução foram medidos em milissegundos.

2. Metodologia

Os algoritmos foram implementados em C e testados em um ambiente controlado. A geração de vetores de entrada foi feita com valores aleatórios e os tempos de execução foram medidos utilizando a função *clock()* da biblioteca padrão de C.

3. Resultados

Os tempos de execução obtidos são apresentados na tabela abaixo:

Tamanho do Vetor	**Bubble Sort**	**Insertion Sort**	**Quick Sort**	**Merge Sort**	**Heap Sort**
100	0.027 ms	0.010 ms	0.006 ms	0.013 ms	0.011 ms
1.000	2.109 ms	0.706 ms	0.075 ms	0.139 ms	0.149 ms
10.000	231.193 ms	70.330 ms	0.897 ms	1.784 ms	2.133 ms
100.000	29.137,449 ms	14.000,358 ms	18.602 ms	18.756 ms	26.220 ms
1.000.000	3.989.793,362 ms	1.462.040,582 ms	1.284,848 ms	603.208 ms	1.058,093 ms

4. Análise

4.1 Algoritmos $O(n^2)$

- **Bubble Sort** e **Insertion Sort**: Ambos os algoritmos exibem um crescimento exponencial no tempo de execução à medida que o tamanho do vetor aumenta. O

Bubble Sort, sendo o menos eficiente, atinge tempos de execução extremamente altos em vetores grandes, como evidenciado pelos mais de 3.989.793 ms (quase 4 milhões de milissegundos) para 10^6 de elementos. O Insertion Sort, embora também lento para vetores grandes, apresenta um desempenho um pouco melhor, com cerca de 1.462.040 ms para o mesmo tamanho de vetor.

4.2 Algoritmos O(n log n)

- **Quick Sort**, **Merge Sort** e **Heap Sort**: Esses algoritmos, com complexidade O(n log n), são muito mais eficientes para grandes vetores. O Quick Sort com um tempo de 1.284 ms para 1.000.000 de elementos Merge Sort (603 ms) e Heap Sort (1.058 ms). Apesar de variações no desempenho, todos os três algoritmos mantêm tempos de execução significativamente menores do que os algoritmos O(n^2).

6. Conclusão

Os resultados confirmam as expectativas teóricas em relação ao comportamento dos algoritmos. Os algoritmos com complexidade O(n^2), como Bubble Sort e Insertion Sort, não são adequados para grandes volumes de dados, uma vez que seus tempos de execução crescem de forma exponencial com o tamanho do vetor. Em contraste, os algoritmos com complexidade O(n log n), como Quick Sort, Merge Sort e Heap Sort, mostraram-se muito mais eficientes e escaláveis, sendo a escolha ideal para vetores maiores.