RELATÓRIO DE IMPLEMENTAÇÃO E ANÁLISE DE ALGORITMOS DE ORDENAÇÃO

Aluno: Guilherme Morais Ribeiro
Disciplina: Estrutura de Dados
Data: 25/02/2026

1. Introdução O objetivo deste trabalho foi implementar e comparar o desempenho de seis algoritmos de ordenação clássicos: Bubble Sort, Selection Sort, Insertion Sort, Shell Sort, Merge Sort e Quick Sort. A implementação foi realizada na linguagem C, utilizando um único arquivo fonte (sorting_algorithms.cpp) onde cada algoritmo foi isolado em sua própria função. Para garantir a justiça nos testes de desempenho, foi implementada uma rotina que gera um array de números aleatórios e fornece uma cópia idêntica deste mesmo array para cada algoritmo processar.

2. Metodologia Para a medição de desempenho, utilizou-se a biblioteca <time.h> e a função clock(). O fluxo de execução do programa segue os seguintes passos:

1. O usuário define o tamanho do vetor (ex: 20.000 elementos).
2. O programa gera números aleatórios de 0 a 9999.
3. O programa executa cada algoritmo sequencialmente, sempre restaurando o vetor para seu estado original (desordenado) antes de cada teste.

3. Resultados Obtidos Os testes foram realizados com um vetor de **20.000 elementos** inteiros gerados aleatoriamente. Abaixo estão os tempos de execução registrados:

Algoritmo	Tempo de Execução (segundos)	Complexidade Teórica (Médio)
Bubble Sort	0.880000 s	$O(n^2)$
Selection Sort	0.415000 s	$O(n^2)$
Insertion Sort	0.211000 s	$O(n^2)$
Shell Sort	0.003000 s	$O(n \log n)$ ou $O(n^{1.25})$

Merge Sort	0.007000 s	$O(n \log n)$
Quick Sort	0.002000 s	$O(n \log n)$

4. Análise Comparativa e Conclusão

Com base nos dados obtidos, foi possível observar uma clara distinção entre dois grupos de algoritmos:

1. **Algoritmos Quadráticos ($O(n^2)$):** O Bubble Sort, Selection Sort e Insertion Sort apresentaram tempos significativamente maiores. O Bubble Sort, em particular, tendeu a ser o mais lento devido à quantidade excessiva de trocas.
2. **Algoritmos Logarítmicos ($O(n \log n)$):** O Merge Sort e o Quick Sort foram ordens de magnitude mais rápidos. Para o vetor de 20.000 elementos, a execução foi quase instantânea (frequentemente abaixo de 0.01 segundos), enquanto os algoritmos quadráticos levaram vários segundos.
3. **O Caso do Shell Sort:** O Shell Sort demonstrou ser uma excelente otimização do Insertion Sort, alcançando tempos muito próximos aos algoritmos mais avançados (Quick/Merge) para este volume de dados.

Conclui-se que, para grandes volumes de dados, a escolha de algoritmos eficientes como o Quick Sort ou Merge Sort é indispensável, sendo o Bubble Sort adequado apenas para fins didáticos ou conjuntos de dados extremamente pequenos.