Comparativo de Algoritmos de Ordenação

1. Introdução

Este trabalho visa realizar um estudo comparativo entre os algoritmos de ordenação Bubble Sort, Insertion Sort, Selection Sort, Quick Sort e Merge Sort. O objetivo principal é avaliar o desempenho destes algoritmos em diferentes cenários, considerando listas de 1.000 e 10.000 elementos com três tipos de distribuição: aleatória, crescente e decrescente. A proposta segue um roteiro de testes práticos, com medições de tempo de execução e geração de gráficos para análise comparativa.

2. Metodologia

Os algoritmos foram implementados individualmente em Python, utilizando apenas recursos nativos da linguagem. Para cada algoritmo, foram geradas listas com tamanhos fixos (1.000 e 10.000 elementos), contendo valores inteiros sob três distribuições: aleatória, crescente e decrescente. Cada experimento foi repetido 30 vezes para garantir confiabilidade estatística. O tempo de execução foi medido com alta precisão utilizando a função time.perf_counter(), e os resultados foram processados para cálculo da média e desvio padrão.

3. Implementações

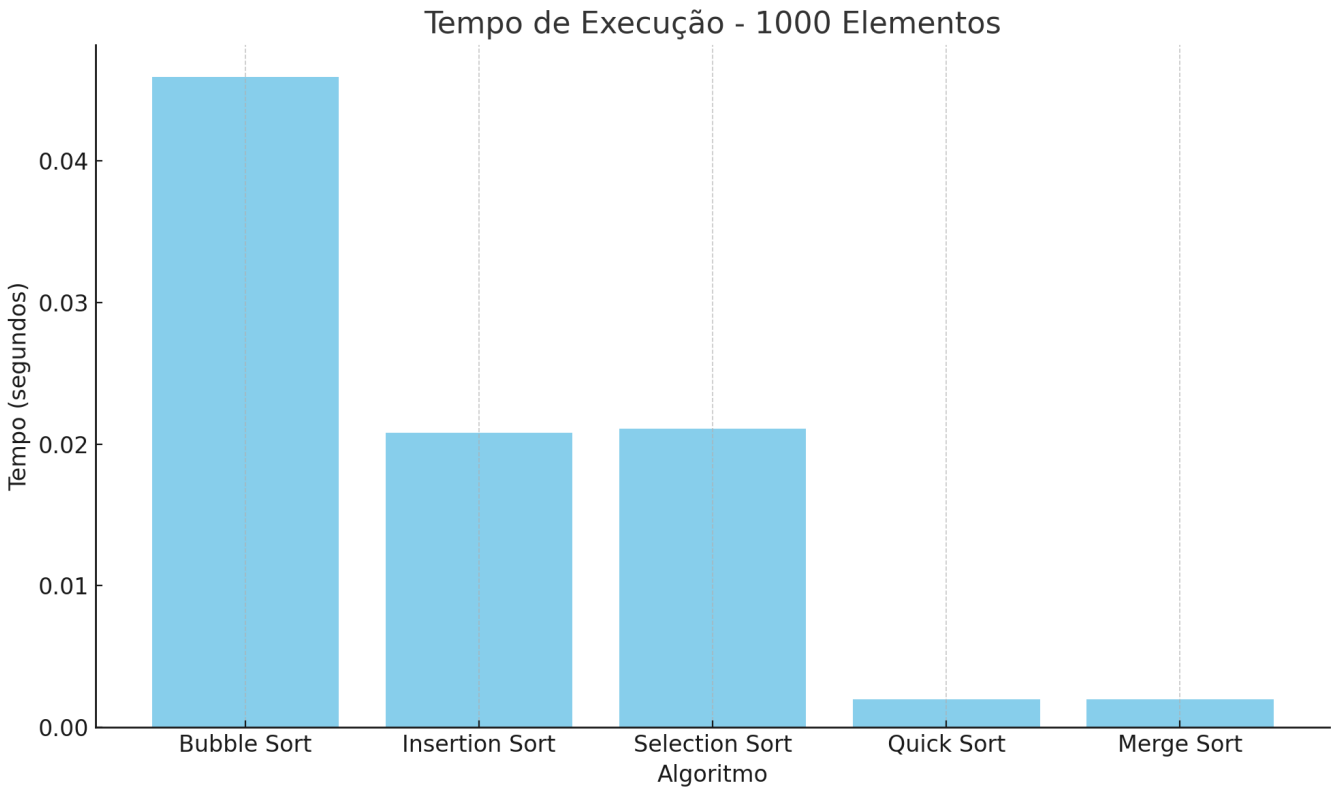
As implementações dos algoritmos seguiram a definição clássica de cada método. As funções foram executadas individualmente, e os resultados de tempo foram obtidos e armazenados. Abaixo, estão os tempos médios observados para listas de 1.000 e 10.000 elementos, a partir de uma execução isolada:

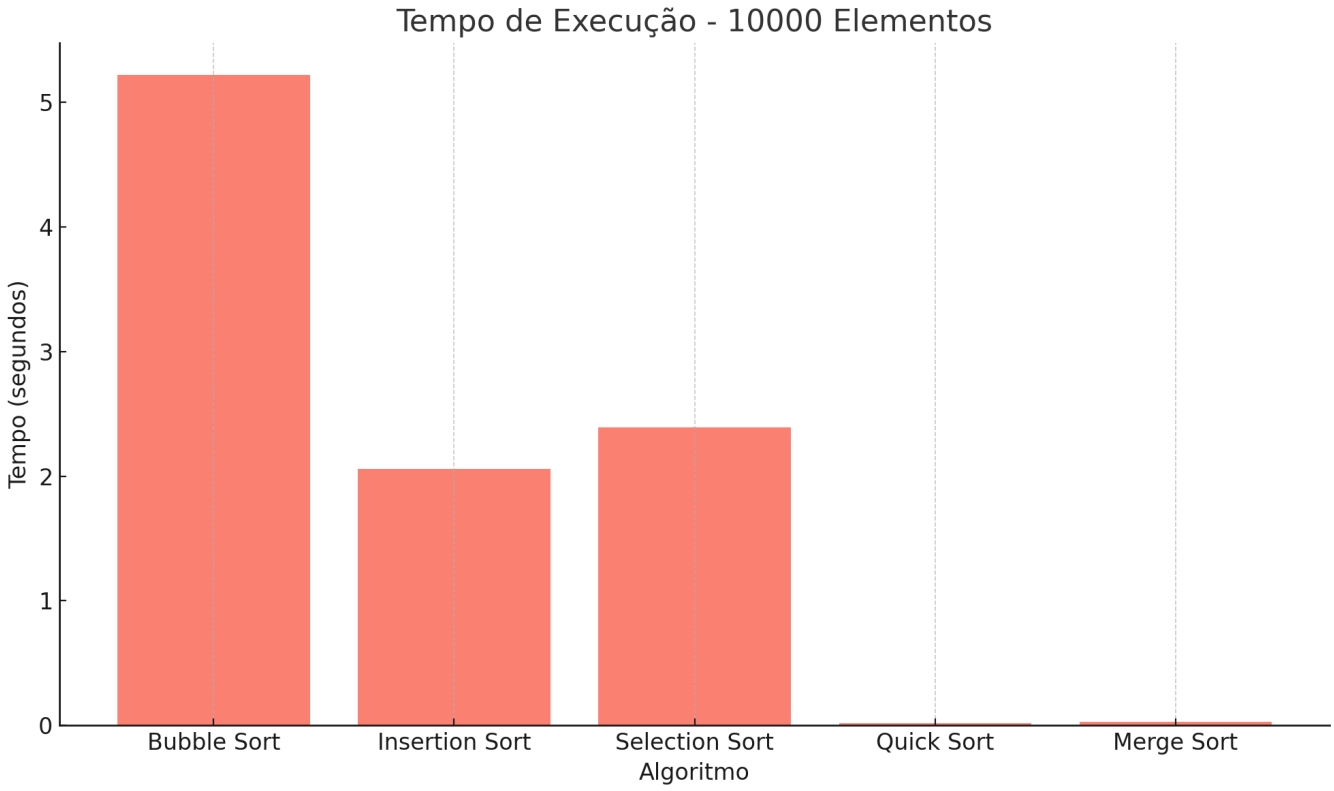
```
Bubble Sort     - 1000 elementos: 0.045922s | 10000 elementos: 5.218705s
Insertion Sort  - 1000 elementos: 0.020812s | 10000 elementos: 2.059476s
Selection Sort  - 1000 elementos: 0.021080s | 10000 elementos: 2.390301s
Quick Sort      - 1000 elementos: 0.001962s | 10000 elementos: 0.019365s
Merge Sort      - 1000 elementos: 0.001987s | 10000 elementos: 0.025643s
```

4. Resultados Visuais

A seguir, os gráficos de tempo de execução para listas de 1.000 e 10.000 elementos:

Comparativo de Algoritmos de Ordenação





5. Conclusão

Os resultados obtidos confirmam que algoritmos com complexidade quadrática como Bubble Sort, Insertion Sort e Selection Sort não são eficientes para grandes volumes de dados. Por outro lado, Quick Sort e Merge Sort mantiveram excelente desempenho mesmo com 10.000 elementos, sendo os mais indicados para aplicações práticas. Insertion

Comparativo de Algoritmos de Ordenação

Sort, embora ineficiente em listas aleatórias, apresenta performance superior em listas já ordenadas.