Análise de Desempenho de Algoritmos de Ordenação

Vitor Totaro Fialho

5 de maio de 2025

Resumo

Este trabalho tem como objetivo comparar o desempenho de quatro algoritmos de ordenação: Selection Sort, Insertion Sort, Bubble Sort e Quicksort. A análise foi realizada em função do tempo de execução, número de comparações e movimentações. Os resultados foram coletados em diferentes tamanhos de vetor, e as conclusões visam entender a eficiência dos algoritmos em diferentes cenários.

1 Introdução

A ordenação é um problema fundamental em ciência da computação, com aplicações em diversas áreas, como busca, análise de dados e otimização. Diversos algoritmos de ordenação possuem características distintas em termos de tempo de execução e uso de memória. Este trabalho compara quatro algoritmos clássicos de ordenação: Selection Sort, Insertion Sort, Bubble Sort e Quicksort.

2 Metodologia

Para a realização do experimento, foram gerados vetores de inteiros com diferentes tamanhos, variando de 100 a 10.000 elementos. Para cada tamanho de vetor, o tempo de execução, número de comparações e movimentações foram registrados. O código foi implementado em linguagem `Python` e os dados foram armazenados em um arquivo CSV. A análise foi realizada utilizando gráficos para cada métrica de desempenho.

3 Resultados

Os resultados obtidos para os três parâmetros de desempenho (tempo, comparações e movimentações) são apresentados nas figuras a seguir.

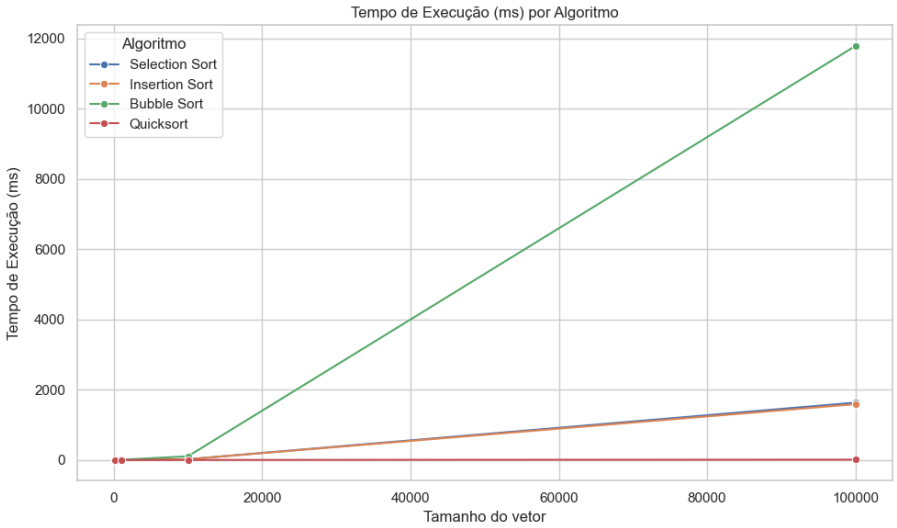

Figura 1: Tempo de Execução (ms) por Algoritmo

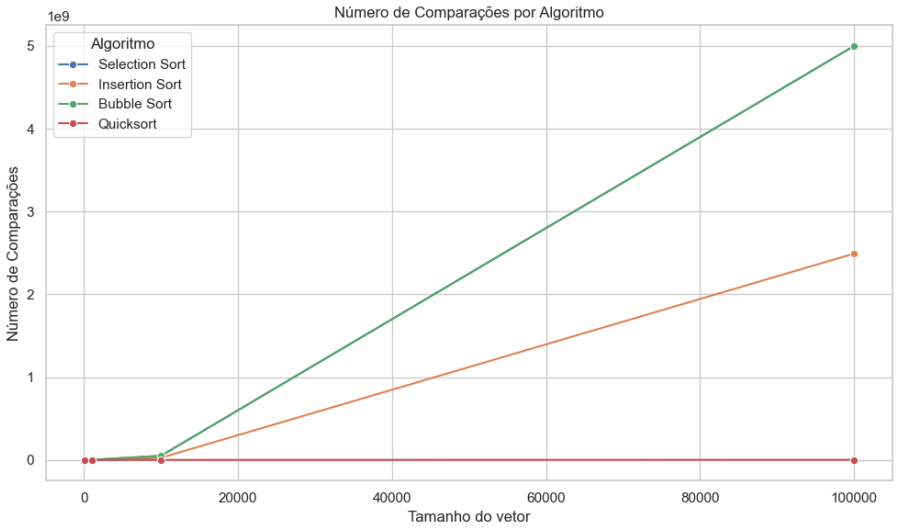

Figura 2: Número de Comparações por Algoritmo

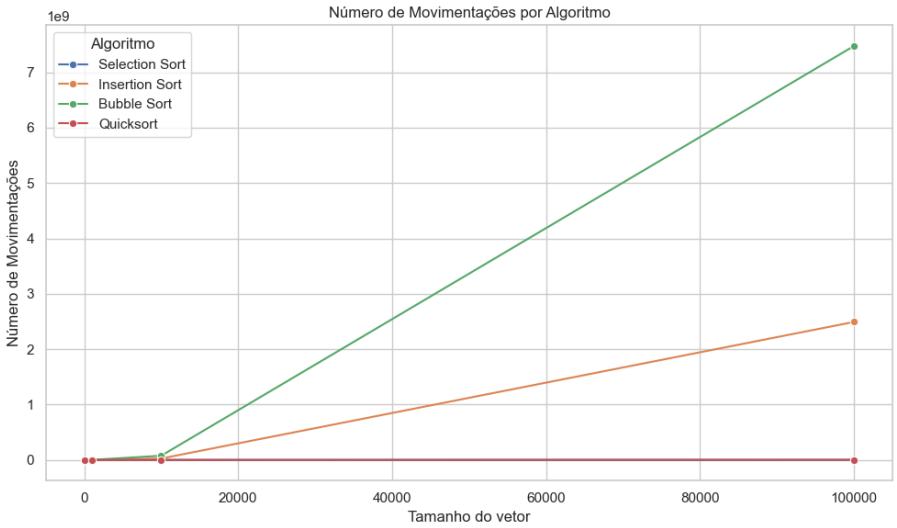

Figura 3: Número de Movimentações por Algoritmo

4 Discussão

Com base nos gráficos apresentados, podemos observar que o algoritmo Quicksort apresenta o melhor desempenho em termos de tempo de execução, especialmente para vetores de tamanho maior. Em contrapartida, os algoritmos Selection Sort e Bubble Sort apresentam uma escalabilidade pior, com tempos de execução significativamente maiores à medida que o tamanho do vetor aumenta.

Em relação ao número de comparações e movimentações, o Quicksort também se destaca, embora seu número de movimentações seja um pouco mais elevado do que o dos outros algoritmos. Isso é esperado, pois o Quicksort realiza mais trocas durante sua execução devido à sua estratégia de particionamento.

5 Conclusão

Este trabalho demonstrou a eficiência do Quicksort em comparação com os algoritmos Selection Sort, Insertion Sort e Bubble Sort, especialmente em termos de tempo de execução. A análise dos gráficos proporcionou uma visão clara sobre o impacto do tamanho do vetor nos desempenhos dos algoritmos, confirmando que algoritmos com complexidade mais baixa, como o Quicksort, tendem a escalar melhor para grandes volumes de dados.

6 Referências

- Knuth, D. E. (1998). The Art of Computer Programming, Volume 3: Sorting and Searching. Addison-Wesley.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.