Análise de Desempenho de Algoritmos de Ordenações

Henrique Paes de Carvalho

Professor: Max do Val Machado

1

Resumo. *Este relatório apresenta uma avaliação de desempenho dos algoritmos Selection Sort, Insertion Sort, Bubble Sort e Quicksort. A avaliação é realizada com base em três critérios: duração da execução, quantidade de comparações e número de trocas, levando em consideração vetores de diferentes dimensões.*

1. Metodologia

Os algoritmos foram implementados em Java com contadores para:

- Comparações: cada verificação condicional entre dois elementos.
- Movimentações: operações de troca ou cópia de elementos.
- Tempo de execução: em milissegundos.

Os testes foram realizados em vetores com tamanhos 100, 1.000, 10.000 e 100.000, com números aleatórios.

2. Resultados

Gráficos:

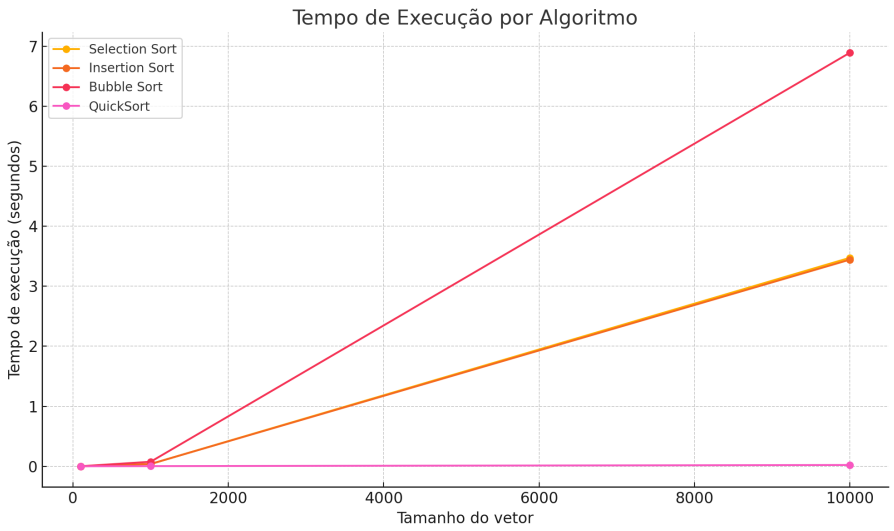

Figure 1. Tempo de execução dos algoritmos.

3. Tempo de Execução

QuickSortteve o melhor desempenho em todos os tamanhos de entrada, com tempos de execução significativamente menores, especialmente à medida que o tamanho do vetor aumenta.

Bubble Sort, Insertion Sort e Selection Sort apresentaram tempos de execução muito maiores, com crescimento quase quadrático, o que é esperado pois possuem complexidade de tempo média e pior caso em $O(n^2)$.

Bubble Sort foi o mais lento, sendo claramente ineficiente para vetores maiores.

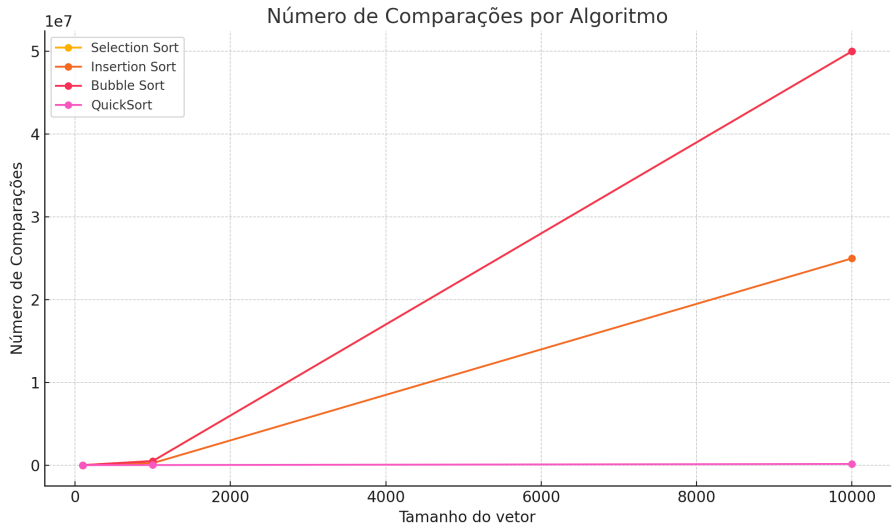

Figure 2. Número de comparações realizadas pelos algoritmos.

4. Número de Comparações

Selection Sort teve um número fixo de comparações próximo de n(n1)/2, independentemente da ordem dos dados. Isso o torna previsível, mas não eficiente.

Bubble Sort e Insertion Sort variam mais dependendo da entrada, mas no caso de dados aleatórios, também realizam muitas comparações.

QuickSort realizou menos comparações em geral, confirmando seu bom desempenho, com complexidade média O(n log n).

5. Número de Movimentações

Insertion Sort e Bubble Sort realizam muitas movimentações, pois constantemente trocam elementos para posicioná-los corretamente.

Selection Sort, apesar de muitas comparações, faz menos trocas porque sempre escolhe o menor valor e o posiciona corretamente de uma só vez.

QuickSort, embora use recursão e partições, mostrou eficiência geral também nesse aspecto.

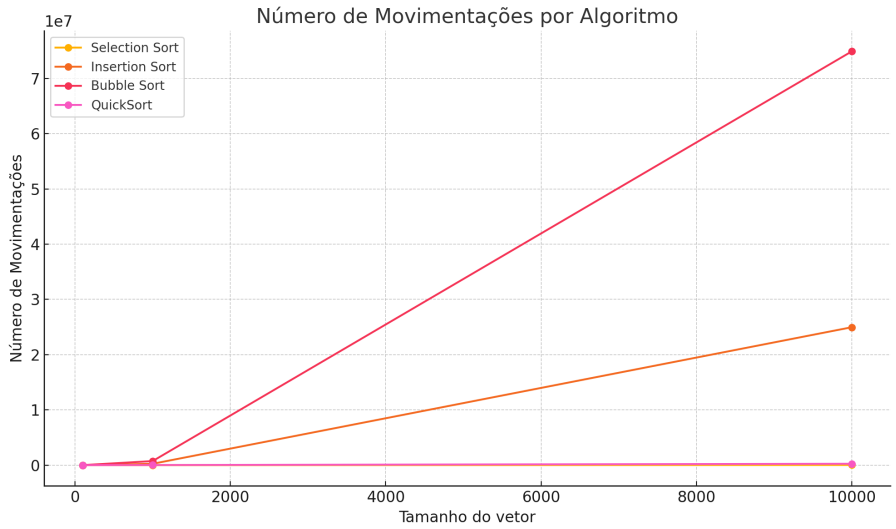

Figure 3. Número de movimentações realizadas pelos algoritmos.

6. Discussão

Os resultados confirmam o comportamento esperado dos algoritmos:

Bubble Sort, Selection Sort e Insertion Sort apresentaram crescimento quadrático ($O(n^2)$) tanto em tempo quanto em comparações, tornando-se ineficientes para entradas grandes. **Quicksort** demonstrou desempenho significativamente superior, com tempo de execução e número de comparações proporcional a $O(n \log n)$.

Em vetores pequenos, o Insertion Sort pode ser eficiente, especialmente se os dados estiverem quase ordenados.

O Bubble Sort foi consistentemente o mais lento, devido ao alto número de comparações e movimentações.

7. Conclusão

A análise mostrou que, para grandes volumes de dados e quando a eficiência é crucial, QuickSort é, de longe, a melhor escolha. Para dados pequenos ou listas parcialmente ordenadas, Insertion Sort ou Selection Sort podem ser mais eficientes em termos de simplicidade e custo computacional.Bubble Sort deve ser evitado, exceto em cenários de aprendizado ou listas muito pequenas, devido à sua ineficiência.