Relatório – Pesquisa Prática: Algoritmos de Ordenação Recursivos e Iterativos

1. Introdução

Esta pesquisa tem como objetivo analisar e comparar o desempenho prático de dois algoritmos de ordenação — um recursivo (Merge Sort) e um iterativo (Selection Sort). A proposta é observar como o tamanho dos dados e o uso da recursividade influenciam no tempo de execução, a partir de testes simples feitos pelo próprio aluno. Essa análise é fundamental para compreender o impacto das abordagens iterativa e recursiva no processamento de grandes volumes de informações, contribuindo para a escolha de algoritmos mais eficientes em situações reais.

2. Descrição dos Algoritmos Estudados

Merge Sort (recursivo):

O Merge Sort segue o método "dividir e conquistar". Ele divide o vetor em duas metades de forma recursiva até restar apenas um elemento em cada parte. Em seguida, combina (merge) essas partes de maneira ordenada, comparando os menores elementos e formando um vetor final crescente. Esse processo se repete até que todo o vetor esteja ordenado. O algoritmo possui complexidade O(n log n), utiliza memória auxiliar e é bastante eficiente para grandes volumes de dados.

Selection Sort (iterativo):

O Selection Sort percorre o vetor em busca do menor elemento e o coloca na posição correta a cada iteração. Em cada passada, a parte já ordenada cresce, enquanto o restante do vetor ainda é verificado. O processo é repetido até que todo o vetor esteja ordenado. Sua complexidade é $O(n^2)$, sendo simples de implementar, mas pouco eficiente para vetores extensos, pois exige muitas comparações e trocas.

3. Resultados dos Testes Realizados

Os experimentos foram realizados em um ambiente Linux com compilador GCC, utilizando vetores de 10, 20, 100 e 500 elementos inteiros gerados aleatoriamente. Foram executados diversos testes para cada tamanho, a fim de obter médias consistentes e minimizar variações de execução.
Os tempos observados variaram da seguinte forma:

Tamanho do Vetor	Tempo Mínimo (Merge Sort)	Tempo Máximo (Merge Sort)	Tempo Mínimo (Selection Sort)	Tempo Máximo (Selection Sort)
20 elementos	0.002 ms	0.002 ms	0.0001 ms	0.001 ms
10 elementos	0.001 ms	0.002 ms	0.000 ms	0.001 ms
100 elementos	0.005 ms	0.019 ms	0.006 ms	0.010 ms
500 elementos	0.028 ms	0.035 ms	0.091 ms	0.107 ms

Em todas as execuções, o Merge Sort apresentou menores tempos médios e maior estabilidade, confirmando sua eficiência superior em vetores maiores. Já o Selection Sort manteve desempenho próximo apenas em vetores pequenos, mas exibiu crescimento quadrático do tempo de execução à medida que o tamanho do vetor aumentou, conforme esperado pela sua complexidade $O(n^2)$.

4. Análise Comparativa dos Resultados

Na prática, os tempos de execução obtidos foram muito próximos entre o Merge Sort e o Selection Sort, variando entre 0.001 e 0.002 milissegundos para vetores de até 20 elementos. Essa pequena diferença ocorre porque os vetores utilizados são pequenos, e o custo adicional da recursividade do Merge Sort se torna irrelevante.

Entretanto, teoricamente, o Merge Sort possui desempenho mais eficiente para grandes volumes de dados devido à sua complexidade $O(n \log n)$, enquanto o Selection Sort cresce de forma $O(n^2)$. Assim, embora nos testes os resultados tenham sido parecidos, em vetores muito maiores o Merge Sort tende a ser significativamente mais rápido e estável.

5. Conclusões

Com base na pesquisa e nos testes realizados, o algoritmo **Merge Sort** demonstrou desempenho superior e maior eficiência na ordenação. O uso da **recursividade** se mostrou vantajoso, pois permite que o processamento seja distribuído em etapas menores e mais rápidas. O **Selection Sort**, embora simples e fácil de entender, não é adequado para grandes volumes de dados devido ao seu elevado número de comparações.
Portanto, o Merge Sort é o algoritmo mais indicado quando se busca eficiência e escalabilidade.

6. Referências

- CORMEN, T. H. et al. *Algoritmos: Teoria e Prática.* 3. ed. Rio de Janeiro: Elsevier, 2012.
- SEDGEWICK, R.; WAYNE, K. *Algorithms.* 4th ed. Addison-Wesley, 2011.
- STROUD, K. A. *Data Structures and Algorithms in C.* MIT Press, 2020.
- GeeksforGeeks. *Merge Sort and Selection Sort Algorithms.* Disponível em: https://www.geeksforgeeks.org