Comparação de Algoritmos de Ordenação

Descrição

Nesta análise, implementamos e comparamos diferentes algoritmos de ordenação, incluindo **Merge Sort**, **Quick Sort** e **Insertion Sort**. O objetivo principal foi avaliar o desempenho de cada algoritmo ao processar arrays de diferentes tamanhos e em várias condições: **ordenados**, **semi-ordenados**, **desordenados** e **totalmente desordenados**.

Objetivo

Compreender as vantagens e desvantagens de cada algoritmo é crucial para escolher a melhor abordagem em cenários específicos. Para isso, usamos um conjunto de dados com diferentes níveis de ordenação e tamanhos, a fim de observar como cada algoritmo se comporta sob essas variações.

Metodologia

- Implementação dos algoritmos de **Merge Sort**, **Quick Sort** e **Insertion Sort**.
- Testes realizados com arrays de diferentes tamanhos: 100, 1000, 10.000, até 5.000.000 elementos.
- Variações de ordenação dos arrays:
    - **Ordenado**: o array já está ordenado.
    - **Semi-ordenado**: o array está parcialmente ordenado.
    - **Desordenado**: elementos em uma ordem aparentemente aleatória.
    - **Totalmente desordenado**: o array está na pior ordem possível para o algoritmo.

Resultados

Tabela de Tempos de Execução

obs: A partir de um array com 100.000 valores, o uso do Insertion Sort se tornou inviável.

Tamanho do Array	Algoritmo	Ordenado	Semi-ordenado	Desordenado	Totalmente Desordenado
100	**Merge Sort**	0.000000s	0.000000s	0.000000s	0.000000s
100	**Quick Sort**	0.000000s	0.000000s	0.000000s	0.000000s
100	**Insertion Sort**	0.000000s	0.000000s	0.000000s	0.000000s
1.000	**Merge Sort**	0.014359s	0.019289s	0.007516s	0.015646s
1.000	**Quick Sort**	0.000000s	0.000000s	0.000000s	0.015630s
1.000	**Insertion Sort**	0.000000s	0.011628s	0.067492s	0.078534s

10.000	**Merge Sort**	0.109997s	.0.109777s	0.094247s	0.141794s
10.000	**Quick Sort**	0.031259s	0.046872s	0.047371s	0.046882s
10.000	**Insertion Sort**	0.000000s	1.053753s	14.300924s	16.681853s
25.000	**Merge Sort**	0.293000s	0.287077s	0.292934s	0.339815s
25.000	**Quick Sort**	0.116757s	0.163526s	0.136547s	0.155408s
25.000	**Insertion Sort**	0.005911s	10.038889s	93.793305s	100.854766
50.000	**Merge Sort**	0.574965s	0.788563s	0.782218s	0.709652s
50.000	**Quick Sort**	0.298443s	0.421078s	0.391752s	0.318711s
50.000	**Insertion Sort**	0.011056s	57.623216s	428.313520s	381.549561s
100.000	**Merge Sort**	1.234819s	1.507258s	1.515478s	1.417176s
100.000	**Quick Sort**	0.672137s	1.321523s	1.122411s	1.109537s
300.000	**Merge Sort**	4.419832s	5.161769s	8.159733s	8.945805s
300.000	**Quick Sort**	2.982266s	5.194449s	3.960138s	3.546137s
700.000	**Merge Sort**	5.962725s	15.237419s	17.554029s	17.150410s
700.000	**Quick Sort**	7.535729s	10.728704s	8.561315s	8.420370s
1.000.000	**Merge Sort**	14.734413s	26.715120s	25.421363s	24.001251s
1.000.000	**Quick Sort**	10.738105s	14.514845s	15.429889s	12.899822s
5.000.000	**Merge Sort**	104.034867s	128.226027s	129.396120s	131.498782s
5.000.000	**Quick Sort**	76.578766s	110.016435s	81.536446s	84.027759s

Discussão dos Resultados

- **Merge Sort**: Apresentou um comportamento consistente em todos os tamanhos de dados, mantendo tempos de execução razoáveis mesmo em arrays maiores.
- **Quick Sort**: Foi o mais rápido em arrays menores e ordenados, mas seu desempenho varia mais significativamente conforme o tamanho e a condição dos dados. Mesmo assim acaba tendo um desempenho superior ao do Merge Sort.

- **Insertion Sort**: Apresenta boa performance em arrays muito pequenos ou quase ordenados, mas se torna inviável para tamanhos maiores e arrays desordenados, como observado a partir de 50.000 elementos.

Conclusão

Cada algoritmo apresenta vantagens em cenários específicos. O **Insertion Sort** é eficaz apenas para pequenas quantidades de dados ou arrays quase ordenados, mas se torna inviável para grandes volumes ou dados desordenados. O **Merge Sort** mantém um desempenho estável, sendo uma boa opção para dados de tamanhos variados, enquanto o **Quick Sort** se destaca como o algoritmo mais rápido na maioria dos testes, especialmente para arrays menores e com condições mais favoráveis.