Relatório de Comparação de Algoritmos de Ordenação

Grupo: Bernado Jakubiak, Danilo Gabriel e Henrique Grigoli

1. Introdução

Este relatório tem como objetivo analisar o desempenho dos algoritmos de ordenação **Bubble Sort**, **Insertion Sort** e **Quick Sort** ao processarem diferentes conjuntos de dados. Os testes foram realizados com dados em três formas distintas (aleatórios, ordenados crescentemente e ordenados decrescentemente) e em três tamanhos diferentes (100, 1.000 e 10.000 elementos). O tempo de execução foi medido em **nanossegundos** utilizando System.nanoTime().

---

2. Metodologia

A metodologia consistiu nos seguintes passos:

- Implementamos três algoritmos de ordenação.
- Leitura dos conjuntos de dados a partir de arquivos .csv.
- Execução dos algoritmos sobre os conjuntos.
- Medição do tempo de execução de cada algoritmo para cada arquivo.
- Registro e comparação dos tempos obtidos.

---

3. Tabela de Resultados

Conjunto de Dados	Bubble Sort	Insertion Sort	Quick Sort
aleatorio_100.csv	1 ms	42 µs	21 µs
aleatorio_1000.csv	5 ms	2 ms	326 µs
aleatorio_10000.csv	106 ms	36 ms	1 ms
crescente_100.csv	1 µs	900 ns	20 µs
crescente_1000.csv	700 ns	1 µs	422 µs
crescente_10000.csv	4 µs	13 µs	56 ms
decrescente_100.csv	47 µs	3 µs	13 µs
decrescente_1000.csv	1 ms	138 µs	329 µs
decrescente_10000.csv	70 ms	12 ms	31 ms

---

4. Análise dos Resultados

A análise dos tempos de execução permite observar os seguintes pontos:

- **Quick Sort** foi o algoritmo mais eficiente na maioria dos cenários, especialmente com conjuntos grandes (1.000 e 10.000 elementos), independentemente da ordem dos dados.
- **Insertion Sort** apresentou desempenho bastante eficiente em conjuntos pequenos (100 elementos) e especialmente em dados já ordenados, devido à sua complexidade adaptativa.
- **Bubble Sort** foi o algoritmo mais lento em praticamente todos os cenários, demonstrando sua baixa eficiência para conjuntos grandes ou desordenados.
- Em **dados crescentes**, tanto o Insertion quanto o Quick Sort foram muito rápidos, mostrando bom aproveitamento da estrutura já ordenada.
- Em **dados decrescentes**, os algoritmos simples tiveram um impacto negativo maior no tempo, enquanto o Quick Sort ainda se manteve eficiente.
- O **tempo de execução** cresce significativamente com o aumento da quantidade de dados, principalmente nos algoritmos com maior complexidade.

---

5. Conclusão

A comparação mostra que, apesar de sua simplicidade, algoritmos como Bubble Sort e Insertion Sort são viáveis apenas para conjuntos pequenos ou ordenados. O Quick Sort se destacou como o algoritmo mais eficiente, sendo a escolha mais apropriada para conjuntos maiores e/ou desordenados.