Relatório de Análise de Desempenho de Algoritmos de Ordenação – Grupo 01

1. Metodologia:

Foram testados três algoritmos de ordenação clássicos:

- **Bubble Sort** (ordenação por trocas)
- **Insertion Sort** (ordenação por inserção)
- **Quick Sort** (ordenação por divisão e conquista)

1.1 Cenários de teste:

- **Dados aleatórios**
- **Dados ordenados crescentemente**
- **Dados ordenados decrescentemente**

- Tamanhos dos conjuntos: 100, 1.000 e 10.000 elementos.

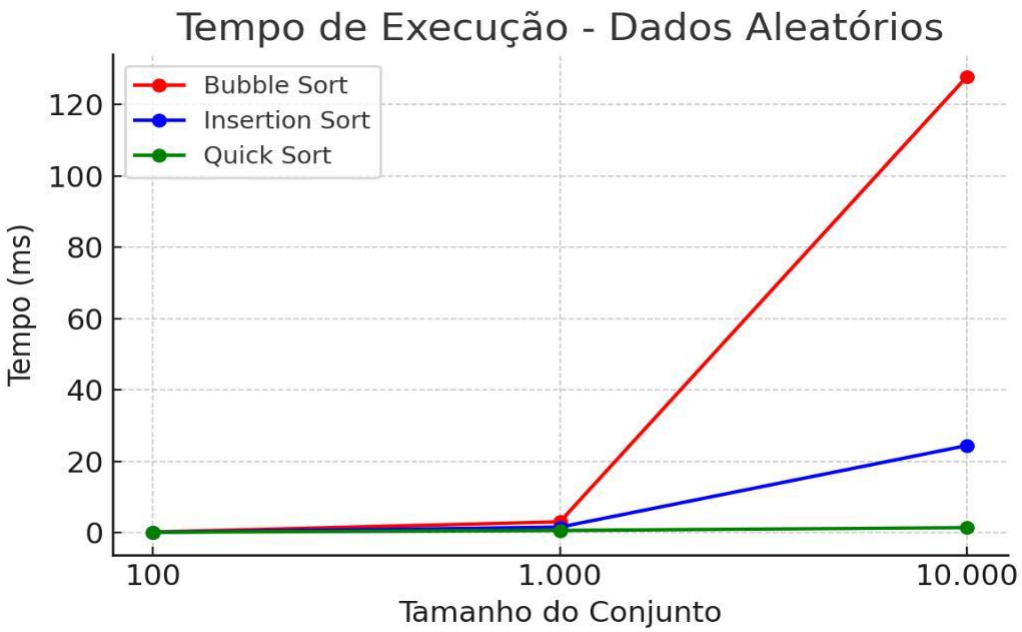
- Unidade de medida: Tempo de execução em milissegundos (ms).

2. Tabela e Gráfico:

Tabela 1: Tempos de Execução por Algoritmo

Cenario	Tamanho	BubbleSort	Insertion Sort	Quick Sort
Aleatório	100	0.115	0.040	0.066
	1.000	2.999	1.518	0.587
	10.000	127.756	24.374	1.357
Crescente	100	0.001	0.019	0.010
	1.000	0.001	0.040	0.052
	10.000	0.003	0.055	0.335
Decrescente	100	0.025	0.020	0.004
	1.000	1.593	0.158	0.035
	10.000	55.453	14.563	0.363

Gráfico: Tempo de Execução em Dados Aleatórios



3. Análise Comparativa

3.1. Desempenho Geral

- **Quick Sort** destacou-se como o mais eficiente em todos os cenários, especialmente em grandes volumes (1.357 ms para 10.000 elementos aleatórios).
- **Insertion Sort** apresentou desempenho intermediário, sendo 5x mais rápido que Bubble Sort em dados aleatórios grandes.
- **Bubble Sort** mostrou-se ineficiente para conjuntos grandes, mas excelente em dados já ordenados (0.003 ms para 10.000 elementos crescentes).

3.2. Comportamento por Cenário

A) Dados Aleatórios

- **Quick Sort** mantém tempo quase constante independentemente do tamanho (+0.066 ms → +1.357 ms).
- **Insertion Sort** tem crescimento linear, enquanto **Bubble Sort** tem crescimento quadrático.

B) Dados Crescentes

- **Bubble Sort** é instantâneo (0.001 ms para 1.000 elementos) devido à flag de otimização.
- **Quick Sort** tem leve overhead devido à recursão.

C) Dados Decrescentes

- **Bubble Sort** sofre com muitas trocas (55.453 ms para 10.000 elementos).
- **Insertion Sort** tem desempenho 4x melhor que Bubble Sort nesse cenário.

4. Conclusões

1. **Quick Sort** é ideal para aplicações com grandes volumes de dados e distribuição imprevisível.
2. **Insertion Sort** é viável para pequenos conjuntos ou dados parcialmente ordenados.
3. **Bubble Sort** é recomendado para listas quase ordenadas.