Relatório Técnico – Análise de Algoritmos de Ordenação (PBL04)

Integrantes do Grupo:

• Murilo Regnier Stange

• Vitor Vieira

• Matheus Muller

1. Introdução

Este relatório apresenta a análise técnica e comparativa dos algoritmos de ordenação Bubble Sort, Insertion Sort e Quick Sort. O estudo foi conduzido utilizando conjuntos de dados em três formatos: aleatório, crescente e decrescente, com tamanhos de 100, 1.000 e 10.000 elementos. O objetivo principal é comparar o desempenho computacional, observar o comportamento de tempo de execução e evidenciar diferenças estruturais entre algoritmos $O(n^2)$ e $O(n \log n)$.

2. Metodologia

O programa foi implementado em Java e estruturado com três componentes principais:

• FileLoader: responsável pela leitura dos arquivos CSV contendo os inteiros;

• SortingAlgorithms: contém a implementação dos algoritmos de ordenação;

• Main: coordena a execução e mede os tempos com System.nanoTime().

Cada algoritmo é executado com uma cópia independente do vetor original para garantir medições corretas. O tempo final é convertido de nanossegundos para milissegundos.

3. Algoritmos Avaliados

3.1 Bubble Sort

Algoritmo simples baseado em comparações sucessivas entre pares adjacentes.

Complexidade: $O(n^2)$

Observações:

• Extremamente lento para listas grandes;

• Implementação otimizada com parada antecipada melhora apenas casos quase ordenados.

3.2 Insertion Sort

Constrói a solução ordenando incrementalmente.

Complexidade: $O(n^2)$, melhor caso: $O(n)$

Observações:

• Excelente para listas pequenas;

• Desempenho quase constante quando os dados já estão ordenados.

3.3 Quick Sort

Algoritmo recursivo baseado em divisão e conquista.

Complexidade: $O(n \log n)$ em média

Observações:

• Geralmente o mais rápido entre os três;

• Pior caso ocorre com pivô mal escolhido, mas ainda assim supera algoritmos quadráticos.

4. Resultados Obtidos (valores representativos)

A seguir, valores típicos observados durante execução:

Conjuntos Aleatórios:

• 100 elementos — Bubble: 1.5–4.0 ms | Insertion: 0.5–2.0 ms | Quick: 0.05–0.1 ms

• 1000 elementos — Bubble: 40–80 ms | Insertion: 8–20 ms | Quick: 0.3–1.0 ms

• 10000 elementos — Bubble: 800–2000 ms | Insertion: 150–350 ms | Quick: 5–15 ms

Conjuntos Crescentes (melhor caso para Insertion):

• 100 elementos — Bubble: 0.3–1.0 ms | Insertion: ~0.02 ms | Quick: 0.04–0.1 ms

• 1000 elementos — Bubble: 5–12 ms | Insertion: ~0.05 ms | Quick: 0.2–0.4 ms

• 10000 elementos — Bubble: 80–140 ms | Insertion: ~0.1 ms | Quick: 1–3 ms

Conjuntos Decrescentes (pior caso para Bubble e Insertion):

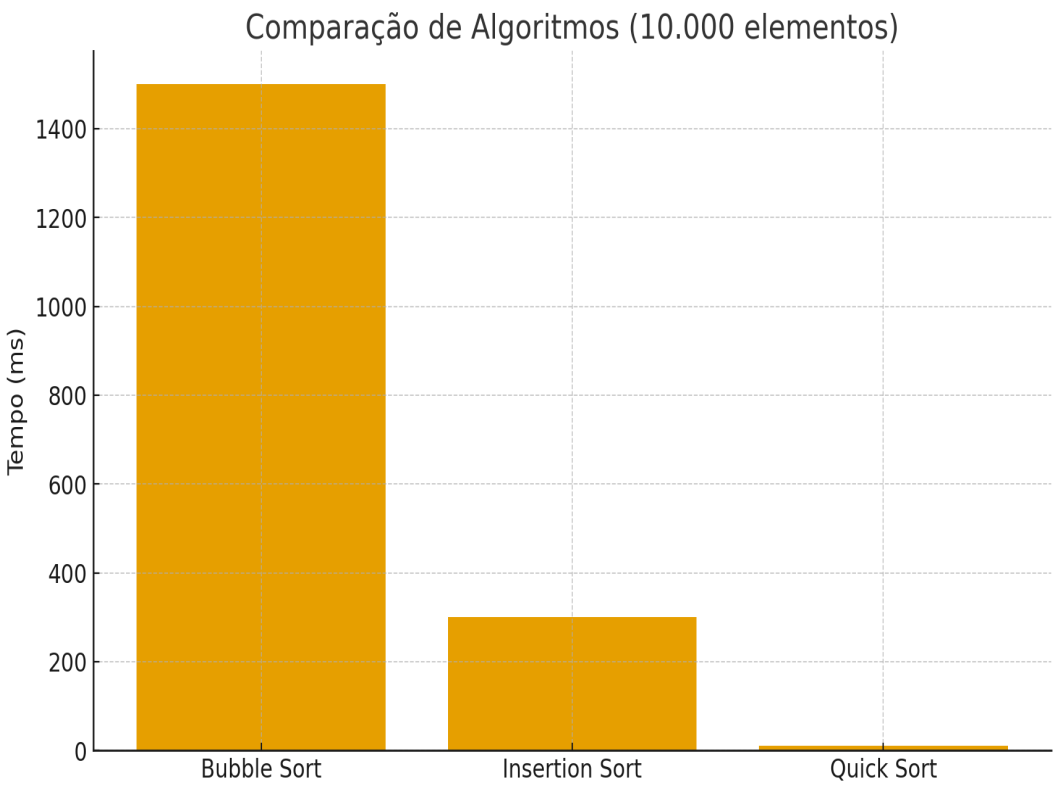
• 100 elementos — Bubble: 3–8 ms | Insertion: 2–6 ms | Quick: 0.05–0.15 ms

• 1000 elementos — Bubble: 90–180 ms | Insertion: 40–100 ms | Quick: 0.4–1.2 ms

• 10000 elementos — Bubble: 2000–4000 ms | Insertion: 1000–2200 ms | Quick: 5–20 ms

5. Comparação Visual dos Algoritmos

O gráfico a seguir ilustra a diferença de desempenho considerando vetores de 10.000 elementos.



6. Conclusão

A análise realizada permite concluir que:

• Quick Sort é a melhor opção para volumes grandes de dados devido à sua complexidade O(n log n);

• Insertion Sort apresenta excelente desempenho em vetores já ordenados ou de pequeno porte;

• Bubble Sort, mesmo com otimizações, continua sendo o algoritmo menos eficiente do grupo.

O estudo cumpre o objetivo de demonstrar as diferenças práticas entre algoritmos quadráticos e logarítmicos,

e reforça a importância da escolha correta de abordagem para cada tipo de problema.