UNIVERSIDADE ESTADUAL DO MATO GROSSO DO SUL
UNIDADE UNIVERSITÁRIA DE DOURADOS

VICTOR RECH VENDRUSCOLO

ANÁLISE COMPARATIVA DE ALGORITMOS DE ORDENAÇÃO

Dourados, MS
2025

VICTOR RECH VENDRUSCOLO

ANÁLISE COMPARATIVA DE ALGORITMOS DE ORDENAÇÃO

Análise Comparativa de Algoritmos de Ordenação: um estudo sobre escalabilidade e casos de uso.

Professor: Fabrício Sérgio de Paula

Dourados, MS

2025

VICTOR RECH VENDRUSCOLO

1 INTRODUÇÃO

Este relatório apresenta uma análise comparativa de performance entre algoritmos de ordenação clássicos, incluindo variações de Bubble Sort, Insertion Sort, Heapsort, Mergesort e Quicksort. O objetivo é avaliar empiricamente a eficiência desses algoritmos sob diferentes condições de entrada — vetores crescentes, decrescentes e aleatórios — e com variados tamanhos de dados, a fim de validar as análises de complexidade teóricas e fornecer uma base para a escolha do algoritmo mais adequado a um determinado contexto de aplicação.

2 METODOLOGIA

Para a realização desta análise, foram implementados oito algoritmos em C. Os testes foram executados em um ambiente controlado para garantir a consistência dos resultados. O tempo de execução de cada algoritmo foi medido em segundos para ordenar vetores de inteiros com tamanhos que variam de 10 a 500.000 elementos. Foram considerados três tipos de cenários para a entrada de dados:

- **Vetor Crescente**: Elementos já ordenados, representando o melhor caso para algoritmos como o Insertion Sort.
- **Vetor Decrescente**: Elementos em ordem inversa, frequentemente o pior caso para algoritmos como o Quicksort com pivô fixo.
- **Vetor Aleatório**: Elementos em ordem aleatória, simulando o caso médio – e mais comum – em aplicações reais.

2.1. Ambiente de Teste

As medições de tempo foram realizadas em um único sistema, cujas especificações, extraídas no momento dos testes, são detalhadas abaixo:

Sistema Operacional: Ubuntu 22.04.2 LTS (Kernel 6.2.0-35-generic, x86_64)
Processador: 4 núcleos, frequência entre 800 MHz e 3900 MHz
Memória RAM: 8 GB
Compilador: GCC (g++) versão 11.4.0
Data dos Testes: 03 de Outubro de 2025

2.3. Algoritmos Avaliados:

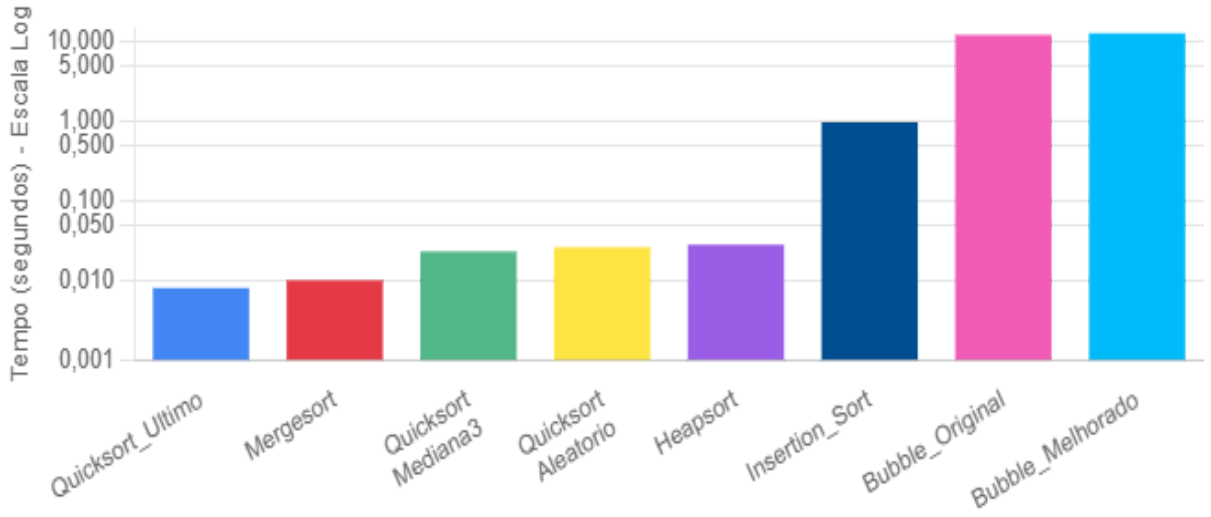
1. **Bubble Original:** Implementação padrão do Bubble Sort.
2. **Bubble Melhorado:** Versão com flag de controle para interrupção antecipada.
3. **Insertion Sort:** Implementação padrão do Insertion Sort.
4. **Mergesort:** Implementação recursiva do Mergesort.
5. **Quicksort Último:** Quicksort utilizando o último elemento como pivô.
6. **Quicksort Aleatório:** Quicksort utilizando um elemento aleatório como pivô.
7. **Quicksort Mediana 3:** Quicksort utilizando a mediana de três elementos como pivô.
8. **Heapsort:** Implementação padrão do Heapsort.

Para cada combinação de algoritmo, tamanho e tipo de entrada, o tempo de execução foi cronometrado. Os resultados foram compilados em um arquivo CSV para posterior análise e visualização

3 RESULTADOS

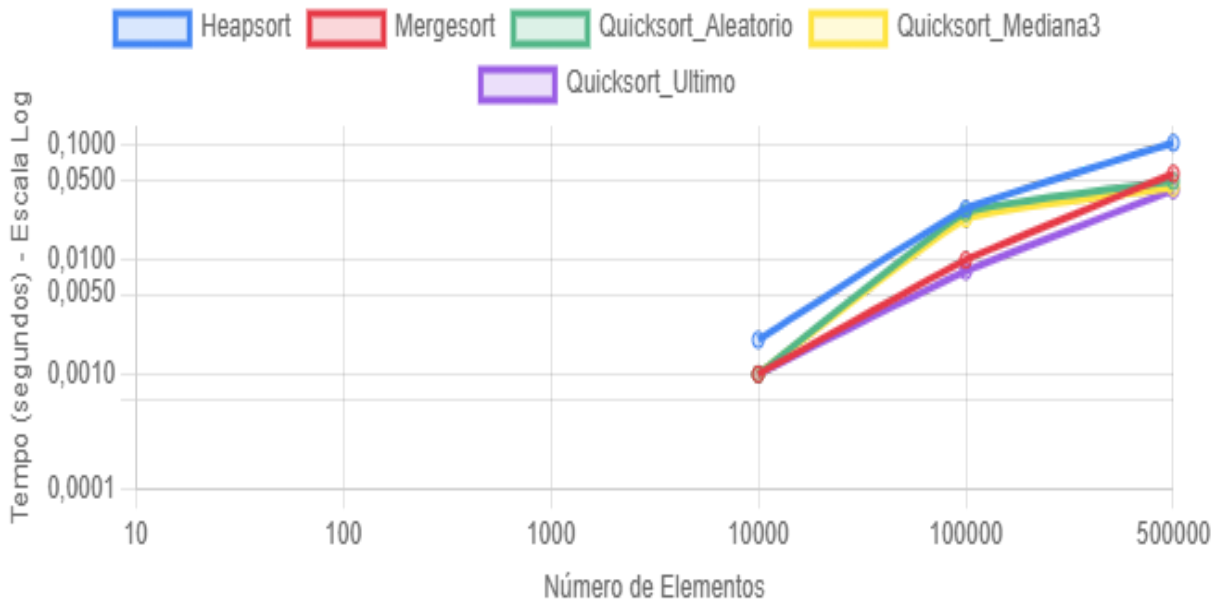
3.1 Visão Geral da Performance (100.000 elementos aleatórios)

A primeira análise compara o tempo de execução para ordenar um vetor aleatório de 100.000 elementos. O gráfico de barras, com escala logarítmica, evidencia a disparidade de eficiência. Algoritmos de complexidade quadrática, como o Bubble Sort, são ordens de magnitude mais lentos que os algoritmos de complexidade O(n log n).



3.2 Impacto da Escalabilidade (Dados Aleatórios)

O gráfico de linhas a seguir ilustra a escalabilidade dos algoritmos mais eficientes. À medida que o número de elementos aumenta, a superioridade dos algoritmos O(n log n) se torna ainda mais clara. *O Bubble Sort e o Insertion Sort foram omitidos deste gráfico, pois seus tempos de execução para entradas maiores tornaram a visualização dos outros algoritmos impraticável.*

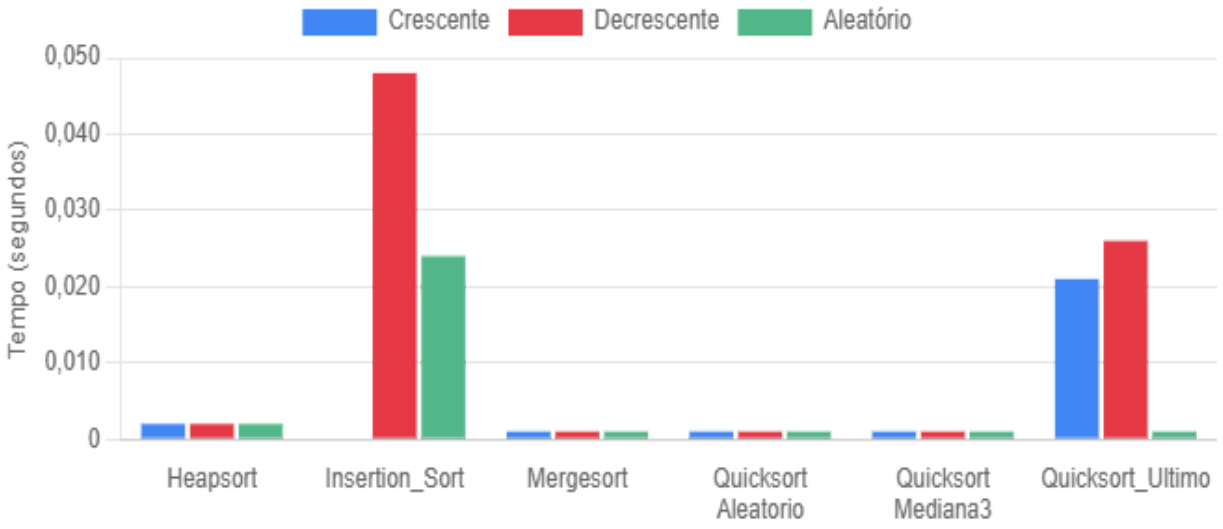


3.3 Tabela de Tempos para Vetor Aleatório (em segundos)

Algoritmo	1.000	10.000	100.000	500.000
Bubble Original	0.001	0.105	12.145	—
Bubble Melhorado	0.002	0.113	12.716	—
Heapsort	0.000	0.002	0.0280	0.106
Insertion Sort	0.000	0.024	0.968	—
Mergesort	0.000	0.001	0.010	0.057
Quicksort Aleatório	0.000	0.001	0.026	0.049
Quicksort Mediana 3	0.000	0.001	0.023	0.043
Quicksort Último	0.000	0.001	0.008	0.041

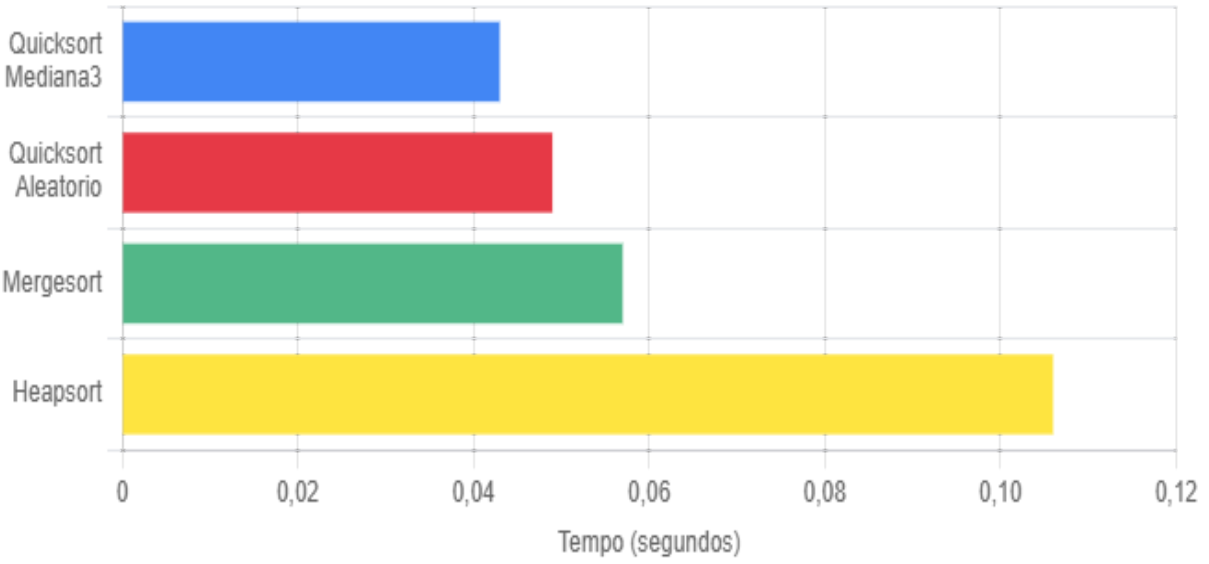
3.4 Análise de Casos: Melhor, Médio e Pior (10.000 elementos)

Este gráfico de barras agrupadas compara o desempenho para vetores ordenados, inversamente ordenados e aleatórios. É notável o comportamento do Insertion Sort, extremamente rápido para dados já ordenados (melhor caso), e a degradação do Quicksort com pivô no último elemento em cenários não aleatórios.



3.5 Desempenho em Grande Escala (500.000 elementos aleatórios)

Isolando a performance para o maior volume de dados testado, vemos que o Quicksort com mediana de 3 e o Mergesort apresentam os melhores tempos. O Heapsort, embora muito eficiente, mostra-se ligeiramente mais lento nesse cenário.



4 CONCLUSÃO

Os resultados confirmam de forma clara a teoria da complexidade de algoritmos. Algoritmos com complexidade de tempo $O(n^2)$, como Bubble Sort e Insertion Sort, são impraticáveis para conjuntos de dados de tamanho moderado a grande. Em contrapartida, algoritmos O(nlogn) como Mergesort, Heapsort e Quicksort demonstram escalabilidade e eficiência.

A análise também ressalta que a escolha do algoritmo "ótimo" depende criticamente da natureza dos dados. O Insertion Sort, por exemplo, pode ser a melhor escolha para listas pequenas ou quase ordenadas. O Quicksort, embora extremamente rápido em média, requer uma estratégia de pivoteamento cuidadosa para evitar seu pior caso. O Mergesort se destaca por sua performance estável e previsível em todos os cenários, sendo uma escolha robusta e segura. O Heapsort também se mostra uma alternativa consistente.

A análise reforça a importância de não apenas conhecer a complexidade teórica de um algoritmo, mas de entender como ele interage com diferentes tipos de dados para escolher a ferramenta mais adequada para cada problema de ordenação.

5 REFERÊNCIAS

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. Introduction to Algorithms. 3. ed. Cambridge: MIT Press, 2009. 1292 p.

TENENBAUM, A., M.; LANGSAM, Y.; AUGENSTEIN, M. J. Estruturas de Dados usando C. Pearson Makron Books, 2010.

PAULA, Fabrício Sérgio de. Algoritmos e Estruturas de Dados II: material didático. Campo Grande: Universidade Estadual de Mato Grosso do Sul, 2025. 120 p. (Notas de aula).