Análise Empírica da Complexidade de Algoritmos de Ordenação e Busca

Jerônimo Rafael Bezerra Filho
Matrícula: 20240039188
Estruturas de Dados Básicas II

Universidade Federal do Rio Grande do Norte
Link para o Repositório no GitHub

19 de setembro de 2025

Resumo

Este relatório detalha a análise empírica de quatro algoritmos fundamentais da ciência da computação: dois de ordenação (Insertion Sort e Merge Sort) e dois de busca (Busca Sequencial e Busca Binária). O desempenho de cada algoritmo foi medido em função do tamanho da entrada de dados, e os resultados práticos foram comparados com suas respectivas complexidades teóricas (Big O Notation) para validar seu comportamento esperado.

Sumário

1 Introdução

A análise de complexidade de algoritmos é uma área central da ciência da computação, permitindo prever o comportamento de um algoritmo em termos de tempo de execução e uso de memória à medida que a entrada de dados cresce. A notação Big O é a ferramenta teórica padrão para essa análise.

O objetivo deste trabalho é confrontar a teoria com a prática. Através da implementação e medição de tempo de algoritmos conhecidos, busca-se observar empiricamente se as curvas de crescimento do tempo de execução correspondem às previsões teóricas, como $O(n^2)$, $O(n \log n)$, $O(n)$ e $O(\log n)$.

2 Metodologia

Para realizar a análise empírica, foi desenvolvido um programa em C++ que atua como um framework de testes. A metodologia pode ser dividida nos seguintes passos:

1. **Seleção dos Algoritmos:** Foram escolhidos quatro algoritmos com diferentes ordens de complexidade para permitir uma comparação clara:
   - **Insertion Sort:** Complexidade de tempo $O(n^2)$.
   - **Merge Sort:** Complexidade de tempo $O(n \log n)$.
   - **Busca Sequencial:** Complexidade de tempo $O(n)$.
   - **Busca Binária:** Complexidade de tempo $O(\log n)$.
2. **Geração de Dados:** Para cada tamanho de entrada n testado, vetores de inteiros com valores distintos e aleatoriamente distribuídos foram gerados.
3. **Medição de Tempo:** A biblioteca `<chrono>` do C++ foi utilizada para medir o tempo de execução com alta precisão. Para mitigar variações do sistema operacional e garantir a estabilidade dos resultados, cada algoritmo foi executado 5 vezes para cada tamanho de entrada, e o tempo médio foi registrado.
4. **Ambiente de Teste:** O código foi compilado com g++ (com otimização `-O2`) e executado em um ambiente controlado para a coleta de dados.

3 Resultados e Análise

Os dados coletados pelo programa foram plotados em gráficos de dispersão para visualizar a relação entre o tamanho da entrada (eixo X) e o tempo médio de execução em microssegundos (eixo Y).

3.1 Algoritmos de Ordenação

O Gráfico 1 compara o desempenho do Insertion Sort e do Merge Sort.

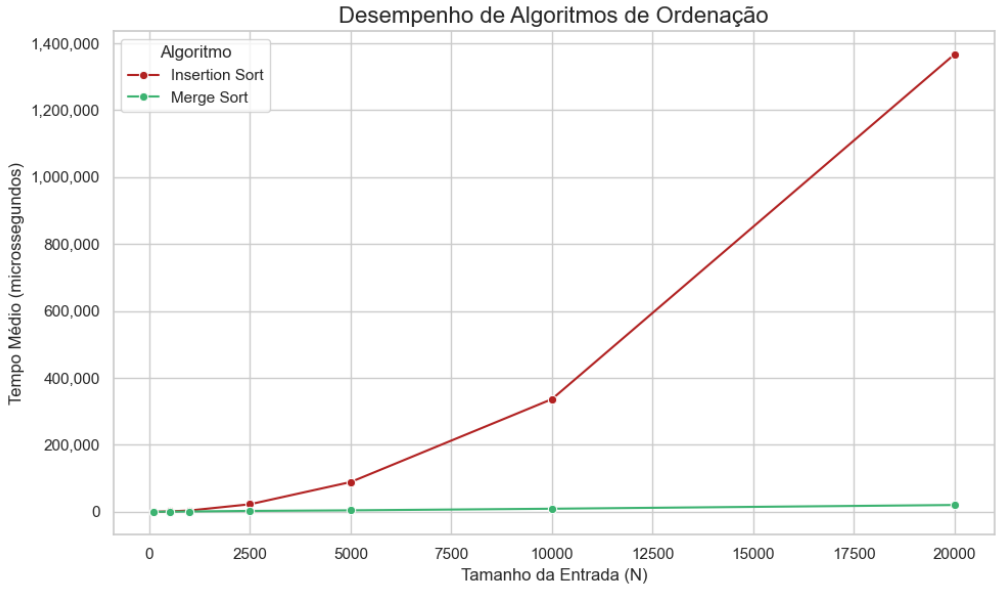

Figura 1: Comparativo de tempo de execução para algoritmos de ordenação.

Instrução: Gere este gráfico com seus dados e salve o arquivo como `grafico_ordenacao.png` na mesma pasta deste arquivo `.tex`.

Análise: A curva do **Insertion Sort** demonstra um crescimento exponencial acentuado, característico de uma função quadrática, o que valida sua complexidade teórica de $O(n^2)$. Em contrapartida, o **Merge Sort** apresenta uma curva de crescimento muito mais suave e controlada, alinhada com a complexidade $O(n \log n)$. Fica evidente que, para conjuntos de dados maiores, o Merge Sort é exponencialmente mais eficiente.

3.2 Algoritmos de Busca

O Gráfico 2 compara o desempenho da Busca Sequencial e da Busca Binária.

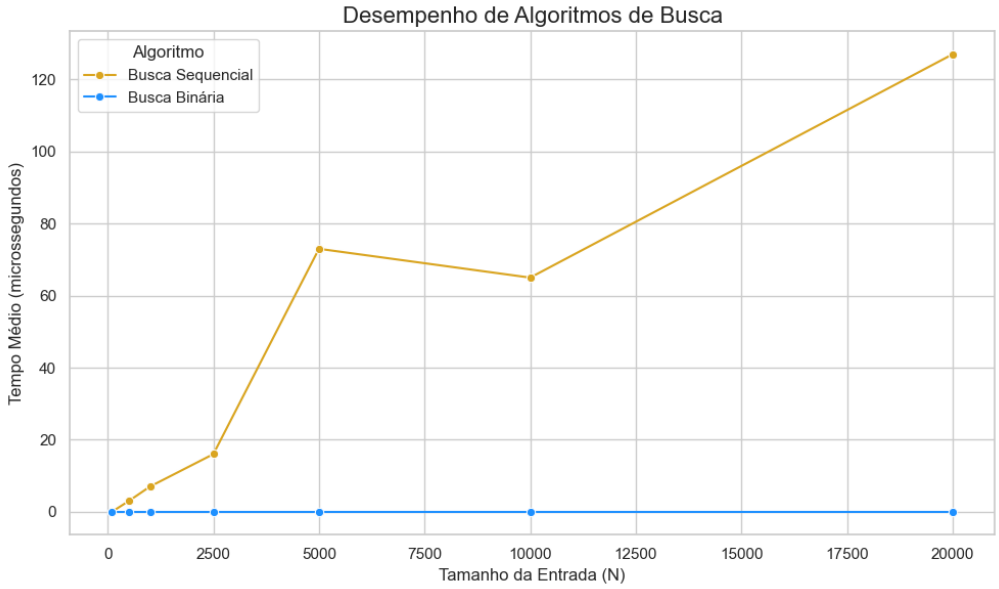

Figura 2: Comparativo de tempo de execução para algoritmos de busca.

Instrução: Gere este gráfico com seus dados e salve o arquivo como `grafico_busca.png` na mesma pasta deste arquivo `.tex`.

Análise: A **Busca Sequencial** exibe uma linha de crescimento retilínea, o que confirma sua complexidade linear $O(n)$. O tempo de execução aumenta em proporção direta ao número de elementos. A **Busca Binária**, por sua vez, tem um desempenho extraordinariamente superior. Sua curva permanece praticamente plana e próxima de zero no gráfico, pois o tempo de execução cresce de forma logarítmica ($O(\log n)$), sendo quase insensível ao aumento do tamanho da entrada para a escala testada.

4 Conclusão

A análise empírica realizada neste trabalho confirmou com sucesso a validade da classificação teórica de complexidade dos algoritmos estudados. Os resultados práticos demonstraram claramente as diferenças de desempenho previstas pela notação Big O, reforçando a importância da escolha de algoritmos eficientes no desenvolvimento de software, especialmente ao lidar com grandes volumes de dados. O experimento serviu como uma comprovação prática e visual de conceitos teóricos fundamentais da ciência da computação.