Estudo Sistemático de Algoritmos de Busca e Ordenação em Diferentes Contextos

Felipe Duarte Silva
GRR20231957
Universidade Federal do Paraná – UFPR
felipeduarte@ufpr.br

***Resumo*—Este relatório aborda e compara diferentes algoritmos de ordenação, iterativos e recursivos, considerando sua eficiência, número de comparações e tempo de execução para diferentes tamanhos de entradas.**

***Index Terms*—Algoritmos, Merge Sort, Insertion Sort, Selection Sort, Busca Binária, Busca Sequencial, Análise de eficiência e Comparação de desempenho.**

I. Introdução

Este trabalho avalia o desempenho de algoritmos de ordenação e busca implementados em C, abordando Insertion Sort, Selection Sort (ambos em formas iterativa e recursiva), Merge Sort (recursivo) e as buscas Sequencial e Binária. Utilizamos vetores em ordem não crescente, com tamanhos variados para explorar as diferenças de eficiência de cada algoritmo. Resultados e implementações específicas são discutidos adiante.

II. Resultados dos Testes

As avaliações dos desempenhos dos algoritmos foram realizadas usando vetores configurados em ordem não crescente testados em uma máquina com processador 11th Gen Intel® Core™ i7-1165G7 com velocidade base de 2.80GHz e um total de 15GiB de RAM. Sistema Operacional: Ubuntu 20.04.6. Os resultados são apresentados abaixo graficamente ilustrados com números de comparações aproximados.

III. Algoritmos de Ordenação

A. Comparações por tamanho

Algoritmos recursivos: Segue na figura 1 comparação entre Insertion Sort e Selection Sort recursivos.

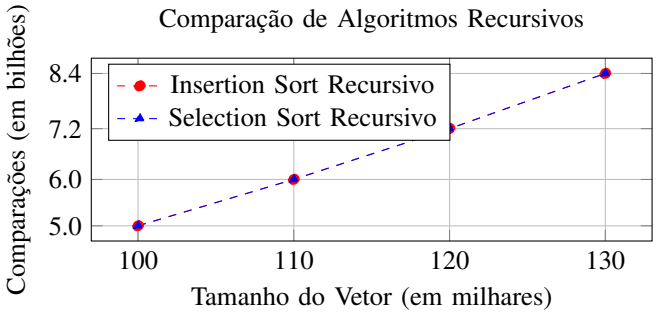

Figura 1: Comparação do número de comparações entre Insertion Sort e Selection Sort recursivos

Algoritmos iterativos: Segue representado na figura 2 o gráfico que compara o Insertion Sort e Selection Sort quanto ao número de comparações pelo tamanho do vetor ordenado em ordem não crescente.

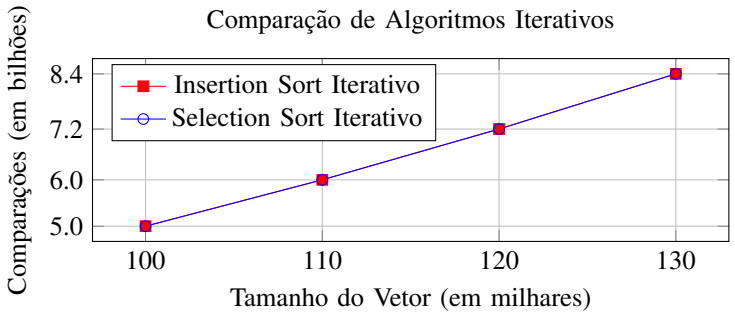

Figura 2: Comparação do número de comparações entre Insertion Sort e Selection Sort iterativos

As Figuras 1 e 2 mostram a quantidade de comparações realizadas pelos algoritmos Insertion Sort e Selection Sort, em versões recursivas e iterativas, conforme o tamanho do vetor aumenta. Os dados foram ordenados de forma não crescente para representar o pior cenário para esses métodos. Ambos os algoritmos mostraram um aumento quadrático no número de comparações, confirmando a complexidade teórica de $O(n^2)$ para o pior caso. Além disso, as versões recursiva e iterativa de cada algoritmo mantiveram desempenhos comparáveis, evidenciando a consistência dos métodos mesmo em condições desfavoráveis.

Merge Sort Recursivo: A figura 3 apresenta o número de comparações do Merge Sort pelo tamanho do vetor

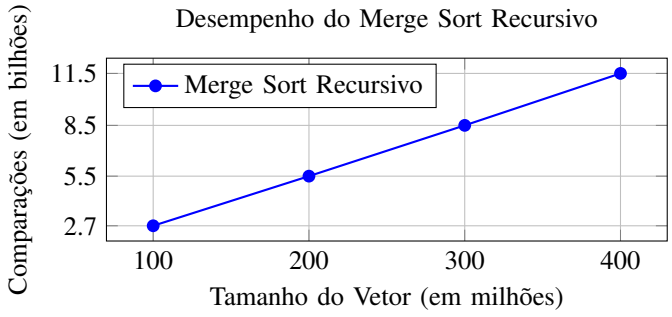

Figura 3: Comparação do número de comparações no Merge Sort Recursivo para diferentes tamanhos de vetor

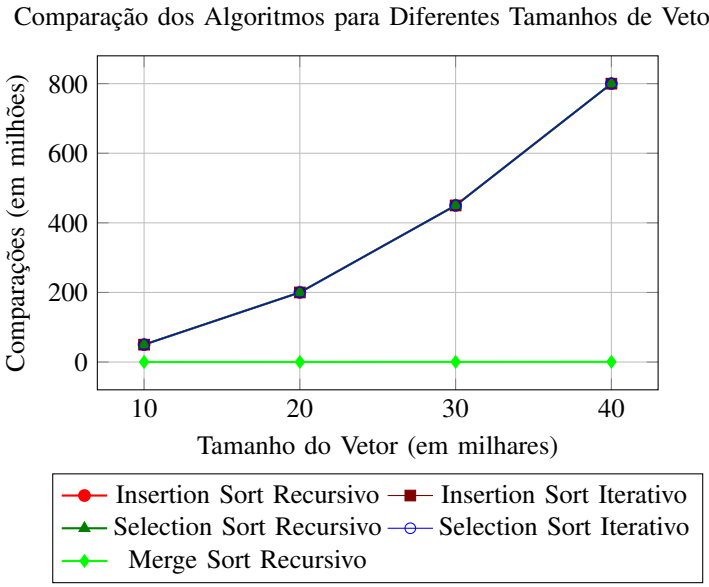

Figura 4: Comparação entre quantidade de comparações

Na Figura 4, é comparado o número de comparações necessárias para cada algoritmo de ordenação. O gráfico destaca claramente a eficiência do Merge Sort, cuja complexidade $O(n \log n)$ é substancialmente mais eficiente que a $O(n^2)$ dos algoritmos Insertion e Selection Sort, tanto em suas versões recursivas quanto iterativas. Os dados confirmam as discussões teóricas em sala de aula, demonstrando a superioridade do Merge Sort em minimizar comparações. Isso enfatiza a importância de selecionar o algoritmo de ordenação mais adequado às necessidades específicas de cada aplicação.

B. Tempo de Execução por tamanho

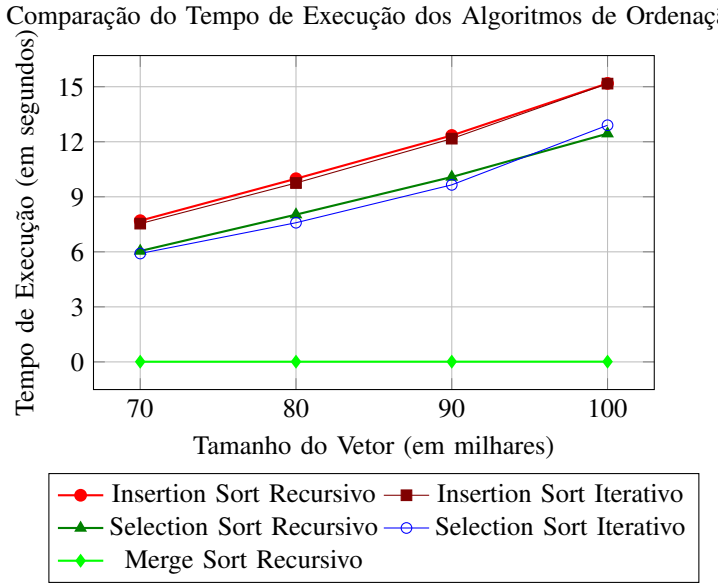

Figura 5: Comparação do tempo de execução entre diferentes algoritmos de ordenação para vetores de 70.000 a 100.000 posições

A Figura 5 ilustra os tempos de execução dos algoritmos de ordenação, com destaque para as versões recursivas e iterativas processando vetores em ordem decrescente. Os resultados destacam a eficiência do Merge Sort, cujo tempo de execução é consistentemente menor que o dos algoritmos Insertion Sort e Selection Sort em todos os tamanhos testados. Esta eficiência, já observada na Figura 4, deve-se à sua complexidade $O(n \log n)$, muito vantajosa em relação à $O(n^2)$ dos outros algoritmos.

IV. Algoritmos de Busca

Comparação entre Algoritmos de Busca: Testes foram realizados com algoritmos de busca em suas formas iterativas e recursivas, utilizando vetores em ordem crescente e buscando elementos inexistentes para destacar o pior caso. Esse método enfatiza as diferenças de eficiência e eficácia entre os algoritmos representadas nas tabelas I e II abaixo.

Busca Sequencial

Tabela I: Comparação dos tempos de execução e número de comparações para o algoritmo de busca sequencial.

Tamanho do Vetor	Método	Tempo (s)	Comparações
10^3	Recursivo	0.0000	10^3
10^3	Iterativo	0.000008	1682
10^4	Recursivo	0.0001	10^4
10^4	Iterativo	0.000061	10686
10^5	Recursivo	0.0011	10^5
10^5	Iterativo	0.000539	100689

Busca Binária

Tabela II: Comparação dos tempos de execução e número de comparações para o algoritmo de busca binária.

Tamanho do Vetor	Método	Tempo (s)	Comparações
10^8	Recursivo	0.000001	27
10^8	Iterativo	0.000002	699
10^9	Recursivo	0.000002	30
10^9	Iterativo	0.000002	702
1.41×10^9	Recursivo	0.000003	31
1.41×10^9	Iterativo	0.000004	703

Os resultados apresentados nas tabelas acima evidenciam a superioridade da busca binária recursiva em termos de eficiência quando comparada com a busca sequencial e ainda sim, com a busca binária iterativa, especialmente em contextos onde os vetores estão previamente ordenados em ordem crescente. A busca binária demonstra um desempenho significativamente melhor devido à sua estratégia de divisão e conquista, que minimiza o número de comparações necessárias para localizar um elemento, mesmo no pior caso.

V. Conclusão

Este estudo confirmou a eficácia do Merge Sort sobre os algoritmos de complexidade $O(n^2)$ e validou a eficiência da busca binária implementada recursivamente sobre às implementações de busca sequencial e sobre a busca binária implementada iterativamente em vetores ordenados.