Análise de Algoritmos de Ordenação e Busca

Gabriel Simon Batista Ribeiro
Departamento de Informática
Universidade Federal do Paraná – UFPR
Curitiba, Brasil
gabriel.simon@ufpr.br
&
Bruno Fuchs Santos da Silva
Departamento de Informática
Universidade Federal do Paraná – UFPR
Curitiba, Brasil
bruno.fuchs@ufpr.br

***Resumo*—Tem como objetivo apresentar de forma sucinta a análise básica de principais algoritmos de Busca e Ordenação, a lógica de sua origem, suas etapas de desenvolvimento, custos de operações e suas peculiaridades.**

***Index Terms*—Comparação, pior custo, análise, ordenação, busca, índice, recursão.**

I. Introdução

Analisar é uma técnica indispensável para um cientista da computação competende, neste trabalho usada como ferramenta de estudo e argumentação, utilizada em seis algoritmos distintos, separando dois para Buscas de índices dentro de vetores: *"Busca Sequencial"* e *"Busca Binária"*; quatro para Ordenações de elementos de vetores: *"Insertion Sort", "Selection Sort", "Merge Sort"* e *"Quick Sort"*

É de importância fundamental conhecer outros conceitos e estabelecer diferentes métodos de analises, de forma lógica, para que possamos concluir as especificidades dos diferentes casos que o algoritmo possa apresentar, destacando seus piores e melhores casos, maiores e menores custos para sua execução e conclusão de sua tarefa de forma efetiva e correta.

II. Método de Análise

Os métodos levados como base de análise são, de forma simples, o custo de comparações e de tempo, inerentemente ligados ao tamanho do vetor, de forma que conseguimos esboçar seu comportamento. Abstraindo informações para a definição do contexto que melhor adequa cada algoritmo.

Todos os algoritmos objetos de estudo foram submetidos aos seguintes atributos de avaliação: quinze testes de execução com diferentes tamanhos de vetores em potência de um até cinco, em base dez. O ambiente de teste foi caracterizado por uso de uma máquina com o sistema *"Linux"* intalada, mais especificamente a distribuição chamada *"PopOs"*. Os tempos de execução de cada algoritmo foram velados em conta como análise, mas com seu foco maior voltado para o número de comparações realizadas por cada, o atributo mais relevante na análise em questão.

III. Busca

A busca de um elemento dentro de uma lista de diversos outros é um problema extremamente comum e diário no contexto da computação, ainda que atualmente existam diversas ferramentas que abstraem e resolvam os possíveis obstáculos encontrados no caminho entender como tais ferramentas e algoritmos atuais chegaram a ser o que são é essêncial para a formação crítica e lógica de um cientista da computação. Dentro de tais algoritmos de busca, um dos primeiros e mais simples é chamado de *"Busca sequêncial"* ou *"Busca ingênua"*, intuitivamente consiste em percorrer de forma sequêncial índice por índice de um vetor, parando quando encontra o elemento procurado especificado, sendo caracterizado com altos números de comparações quando em seu pior caso.

Outra forma de busca, melhor otimizada é chamada *"Busca Binária"*, uma solução mais sofisticada, contendo uma lógica concisa porém efetiva, seu desempenho é altíssimo, a diferença é ainda mais expressiva quando comparada a fim com o algoritmo de busca anterior, é indeficiente, como pode-se observar na Tabela I.

Tabela I
Relação Tamanho do Vetor/Número de Comparações.

	Vetor/Nro Comparações			
Tamanho Vetor	10^1	10^2	10^3	10
Busca Sequêncial	3.4	6.72	9.97	13.36
hline Busca Binária	0.3	0.07	0.009	0.00085

IV. Ordenação

A ordenação caracteriza-se como um problema pilar da computação, existem diversos sistemas de algoritmos largamente usados em diversas ferramentas e *softwares*, contudo para este trabalho foram selecionados os chamados: *"Insertion Sort", "Selection Sort", "Merge Sort", "Quick Sort"* por serem os mais comuns da área. Conforme o gráfico abaixo é observado o número de comparação, que no *"Selection Sort"* acaba por defasado, visto o alto número de comparações na

medida que o vetor cresce de tamanho, isso por sua vez se dá em vista do seu pior caso, que é o pior de todos dentro dos quatro. É nítido a relevância e quantidade de comparações do *"Merge Sort"* e do *"Quick Sort"* em que ambos são considerados os melhores no âmbito analisado, mas no caso do *"merge"* em específico surge um pequeno problema de alocação de memória, pois este analisado tem como atributo a criação de diversos sub-vetores auxiliares para seus cálculos, quando aplicado em um array muito grande certamente criará problemas de memória.

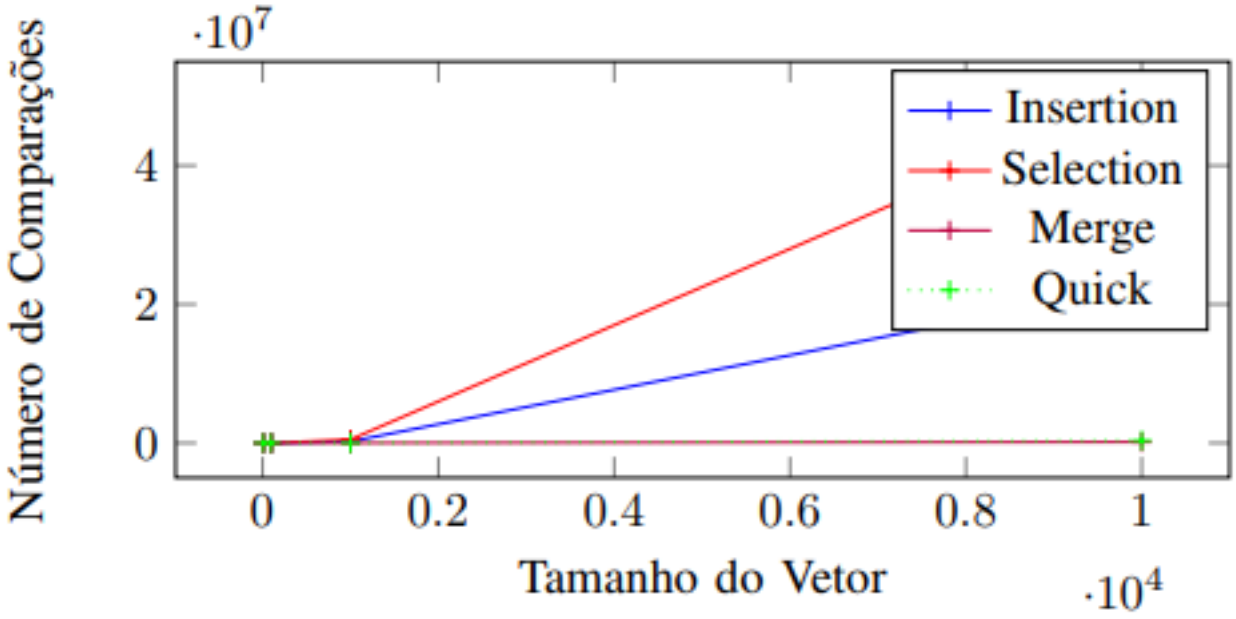

Figura 1. Gráfico com a relação Tamanho Vetor / Número de Comparação.

V. Conclusão

Contudo, obtivemos dados que constataram uma maior utilização do algortimo de *"Busca Binária"*, faz de fato sentido, pois é até então logicamente o melhor algoritmo de busca, com menor custo de uso. Enquanto que na ordenação em sí, *"Quick Sort"* é o mais popular, até mesmo em bibliotecas padrões de varias linguagens, também o mais utilizado pela comunidade em um âmbito geral.