Relátorio Trabalho 1

Ricardo Quer do Nascimento Filho
GRR 20224827
Universidade Federal do Paraná – UFPR
Curitiba, Brasil
ricardo.filho@ufpr.br

***Resumo*—Este trabalho tem como afinidade mostrar algoritmos de ordenação e busca, produzidos em sua forma interativa e recursiva. Além da modalidade de mostrar eficiência e desempenho por meio de tempos e números totais de comparações de cada algoritmo.**

***Index Terms*—Algoritmos, Recursão, Busca Ingênua/Sequencial, Busca Binária, Insertion Sort, Selection Sort, Merge Sort, Análise de Desempenho, Número de Comparações**

I. Introdução

Este relátorio tem como intuito destacar a análise sobe os algoritmos de ordenação e buscas interativos: Insertion, Selection, Busca Sequencial e Busca Binária. Juntamente algoritmos recursivos: Insertion, Selection, Merge, Busca Sequencial e Busca Binária.

A análise desses algoritmos sera apresentado em forma de tabelas e gráficos para fins de comparação entre algoritmos, para essas comparações será levado em conta o tamanho do vetor *N*, seu tempo de execução e número totais de comparação dentro dos algoritmos.

II. Condições de Teste

Os testes foram realizados em uma máquina com sistema operacional Windows, em primeira mão foi aloca o vetor na memória com tamanho *N*, digitado pelo usuário, os valores do vetor são carregados por uma função:

preencherVetorAleatorio()

Ela calcula um valor baseado numa taxa de variação entre 1 e 100.000 utilizando uma semente de geração aleátoria *rand()*.

Vale ressalaar que toda memória alocada foi liberada utilizando a função *free(vetor)* , já que temos algoritmos como Merge Sort é necessário a utilização de um vetor auxiliar, foram feitos diversos testes e otimizações no código para não ter alocação de memória desnecessaria, além de verificações no código com o objetivo de averiguar se não escrevia em posições de memória que não poderiam ser acessadas.

III. Explorando os Algoritmos de Ordenação

1) **Insertion Sort** ou ordenação por inserção é uma forma de ordenação que percorre um vetor de elementos da esquerda para a direita e à medida que avança vai ordenando os elementos à esquerda. O funcionamento do algoritmo é bem simples: consiste em cada passo a partir do segundo elemento selecionar o próximo item da sequência e colocá-lo no local apropriado de acordo com o critério de ordenação.

2) **Selection Sort** ou ordenação por seleção consiste em selecionar o menor item e colocar na primeira posição, selecionar o segundo menor item e colocar na segunda posição, segue estes passos até que reste um único elemento.Suas principais vantagens estão na fácil implementação do algoritmo,

3) **Merge Sort** é um exemplo de algoritmo de ordenação que faz uso da estratégia "dividir para conquistar" para resolver problemas, sendo um algoritmo muito eficiente. Esse algoritmo divide o problema em pedaços menores, resolve cada pedaço e depois junta (merge) os resultados

IV. Trabalhando com Buscas

1) **A Busca Sequencial/Ingênua** é uma busca que dizemos que eles possui uma relação linear ou sequencial. Cada item de dados é armazenado relativamente aos outros itens, essas posições são valores de índice dos itens individuais. A maneira como são armazenados torna possível realizar busca em sequência, esse processo dá origem a busca sequencial. Fácil de implementar, mas ineficiente em grandes conjuntos de dados devido ao tempo de sua execução linear.

2) **A Busca Binária** consiste em dividir o vetor e analisar se o dado requerido está antes ou depois da metade encontrada na divisão - se estiver antes, o programa descarta tudo que estiver depois do 'meio'; caso contrário, o programa descarta tudo que estiver antes. Isto é feito várias vezes até que o item desejado seja encontrado.

V. Testes Avaliados

Os testes foram feitos em vetores de 1.000, 10.000 e 100.000 posições devido ao processamento da maquina não ser tão potente, os testes de avaliação foram realizados em uma máquina com processador *AMD Ryzen 5 3400G with Radeon Vega Graphics 3.70 GHz*, *8,00 GB de RAM* e um sistema operacional*Windows 11 Pro, versão 23H2*. A análise de eficiência dos algoritmos estarão logo abaixo: .

VI. Algoritmos de Ordenação Interativa

Nesta seção, discutiremos dois algoritmos de ordenação: Insertion Sort e Selection Sort.

Tabela I
COMPARAÇÃO DE COMPARAÇÕES ENTRE INSERTION SORT E SELECTION SORT EM DIFERENTES TAMANHOS DE VETOR

Tamanho do Vetor	Comparações(Insertion)	Comparações(Selection)
1000	244 445	499 500
10 000	24 916 112	49 995 000
100 000	2 502 649 136	4 999 950 000

Tabela II
COMPARAÇÃO DE COMPARAÇÕES ENTRE INSERTION SORT E SELECTION SORT EM DIFERENTES TAMANHOS DE VETOR

Tamanho do Vetor	Tempo(Insertion)	Tempo(Selection)
1000	244 445	499 500
10 000	24 916 112	49 995 000
100 000	2 502 649 136	4 999 950 000

VII. ALGORITMOS DE ORDENAÇÃO RECURSIVA

são três insertion Sort, Selection Sort e Merge Sort. Tabela de tempos e número de comparações visando os tamanhos do vetor 1.000, 10.000 e 100.000

Tabela III
TEMPOS DE ALGORITMOS DE ORDENAÇÃO EM DIFERENTES TAMANHOS DE VETOR

Tempo (Insertion Sort)	Tempo (Selection Sort)	Tempo (Merge Sort)
0.000 144	0.000 123	0.000 158
0.001 728	0.001 578	0.002 084
0.016 912	0.015 243	0.018 367

Tabela IV
NÚMEROS DE COMPARAÇÕES DE ALGORITMOS DE ORDENAÇÃO EM DIFERENTES TAMANHOS DE VETOR

Comparações(Insertion)	Comparações(Selection)	Comparações(Merge)
10	999	9976
13	9999	133 616
17	99 999	1 668 928

O principal motivo do Insertion Sort Recursivo ser tão eficiente é causando pela adição da busca binária no código, o causando essa disparidade de comparações. O Merge Sort possui naturalmente um dos menores tempos e comparações, se compararmos com o Insertion com busca binária, é clara a alta eficiência do Merge Sort.

VIII. BUSCAS

Conforme a figura abaixo,mostrá a quantidade de comparações propondo um vetor com 1.000 posições, em suas versões interativas e recursivas. Figura 1

Figura 1

IX. TABELA DE CUSTOS

É notorio qur varios algoritmos tenham suas finalidades em ser mais rápido, ter menos custo em relação as comparções, aqui está uma tabela que mostre os devidos

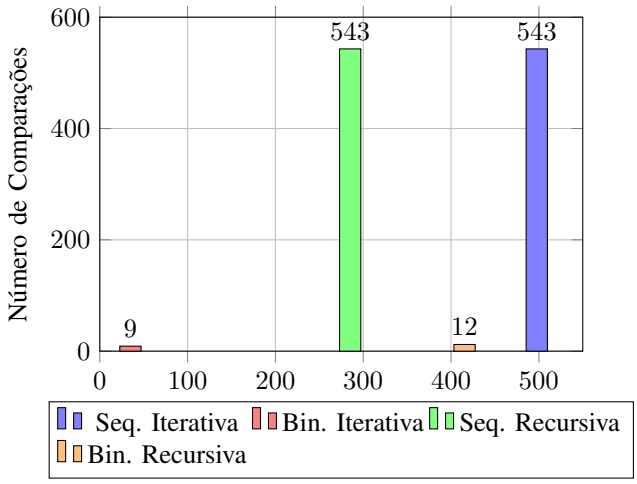

Figura 1. Comparação de Buscas Sequencial e Binária (Iterativas e Recursivas) em um Vetor

custos aproximados dos algoritmos no melhor e pior caso (nota: BI e BB no insertion sort recursivo é a abreviação de Busca Ingênua e Busca Binária) Figura 2

Tabela V
CUSTOS DOS ALGORITMOS NO MELHOR E PIOR CASO

Algoritmo	Pior Caso	Melhor Caso
Busca Ingênua	$C(n)$	$C(1)$
Busca Binária	$C(n \log n)$	$C(1)$
Insertion Sort	$C(n^2/2)$	$C(n)$
Insertion Sort (Recursivo, BI)	$C(n^2/2)$	$C(n)$
Insertion Sort (Recursivo, BB)	$C(n \log n)$	$C(n \log n)$
Selection Sort	$C(n^2/2)$	$C(n^2/2)$
Selection Sort (Recursivo)	$C(n^2/2)$	$C(n^2/2)$
Merge Sort	$C(n \log n)$	$C(n \log n)$

Figura 2

X. CONCLUSÃO

Os algoritmos de ordenação e buscas neste estudo revelam nuances interessantes em desempenho e eficiência. Por exemplo, a ordenação por inserção e a ordenação por seleção destacam-se pela sua simplicidade de implementação, mas a sua complexidade quadrática limita a sua eficácia em conjuntos de dados muito grandes. A classificação por mesclagem, por outro lado, é muito eficiente, possui complexidade logarítmica e é adequada para grandes conjuntos de dados. Depois de analisar a busca sequencial e a busca binária, descobrimos que a busca binária é mais eficiente do que a busca sequencial. A comparação das versões iterativas e recursivas do algoritmo revela compromissos interessantes entre complexidade e desempenho.A comparação das versões interativas e recursivas do algoritmo revela compromissos interessantes entre complexidade e desempenho. Os métodos recursivos fornecem implementação elegante e clara, mas geralmente são mais caros em termos de espaço ou tempo de execução.