Relatório

O presente relatório refere-se aos algoritmos de busca e ordenação, foram realizados testes para verificar o número de comparações e o tempo de execução, desta forma conseguimos identificar o mais eficiente dado o problema que cada um soluciona.

Os testes a seguir foram realizados 10 vezes o resultado de número de comparações e tempo de execução é a média dessas 10 execuções. Porém o tempo de execução médio pode ser menor ou maior dependo do computador que o executa.

Algoritmos de Busca

Foi implementado algoritmos de busca conhecidos como *Busca Sequencial* e *Busca Binária,* onde dado um vetor ordenado de tamanho **N** (Número natural) e um elemento **X (**Número natural), com **0 <= X <= N**, o algoritmo encontra **X** no vetor e retorna sua posição, e como este processo demanda tempo e comparações entre elementos do vetor, este critério foi analisado.

Busca Sequencial

Este algoritmo no pior caso para dado um vetor de tamanho **N,** realiza **N** comparações para encontrar o elemento **X**.

A seguir a tabela com os resultados dos testes:

Tamanho do Vetor	Número de Comparações Média	Tempo de execução Médio(s)
100	37	0.000004
1000	530	0.000027
10000	5793	0.000228
100000	48506	0.001565

Busca Binária

Este algoritmo no pior caso para dado um vetor de tamanho **N,** realiza $N.\log 2\ N + 1$ comparações para encontrar o elemento **X**.

A seguir a tabela com os resultados dos testes:

Tamanho do Vetor	Número de Comparações Média	Tempo de execução Médio(s)
100	5	0.000002
1000	9	0.000001
10000	12	0.000002
100000	16	0.000002

Algoritmos de Ordenação

Foi implementado os algoritmos de ordenação SelectionSort, InsertionSort, MergeSort e QuickSort, onde dado um vetor de tamanho **N** (Número natural), o algoritmo ordena o vetor de forma crescente de forma que **I** sendo o índice de um elemento no vetor, e vetor[**I**] sendo elemento de forma que vetor[**I**] <= vetor [**I+1**] para todo **I**∈ vetor. Da mesma forma este processo demanda tempo e comparações entre elementos do vetor.

SelectionSort

Este algoritmo para dado um vetor de tamanho **N,** sempre realiza **N * (N – 1) / 2** comparações para ordenar o vetor.

A seguir a tabela com os resultados dos testes:

Tamanho do Vetor	Número de Comparações Média	Tempo de execução Médio(s)
100	4950	0.000033
1000	499500	0.002364
10000	49995000	0.2165521

InsertionSort

Este algoritmo no pior caso para dado um vetor de tamanho **N,** realiza $\frac{N^2-N}{2}$ comparações para ordenar o vetor.

A seguir a tabela com os resultados dos testes:

Tamanho do Vetor	Número de Comparações Média	Tempo de execução Médio(s)
100	2378	0.000068
1000	241298	0.005322
10000	24200445	0.5044788

MergeSort

Este algoritmo no pior caso para dado um vetor de tamanho **N,** realiza $\sim N \times \log 2\ N$ comparações para ordenar o vetor.

A seguir a tabela com os resultados dos testes:

Tamanho do Vetor	Número de Comparações Média	Tempo de execução Médio(s)
100	356	0.000021
1000	5044	0.000257
10000	69008	0.003522
100000	853904	0.084521

QuickSort

Este algoritmo no pior caso para dado um vetor de tamanho **N,** realiza $\frac{N^2+N-2}{2}$ comparações para ordenar o vetor.

A seguir a tabela com os resultados dos testes:

Tamanho do Vetor	Número de Comparações Média	Tempo de execução Médio(s)
100	346	0.000018
1000	5497	0.000200
10000	79978	0.002651
100000	1032764	0.032360