UNIVERSIDADE FEDERAL DO TOCANTINS
CAMPUS DE
PALMAS COORDENAÇÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO
109 Norte, Avenida NS 15, ALCNO 14, Bloco III | 77402-970 | Palmas/TO
(63) 3229-4527 | http://palmas.uft.edu.br/grad/ccomp |
comppalmas@mail.uft.edu.br

Disciplina: Projeto e Análise de Algoritmos
Professor: Warley Gramacho
Alunos: Gabriel Tavares, Izabela Caldeira, Lindomar Rodrigues, Rafael Silva

Relatório - Algoritmos de Ordenação

INTRODUÇÃO

O presente trabalho se trata de uma análise em relação ao tempo de execução de alguns algoritmos de ordenação, mais concretamente realizar uma análise gráfica em relação ao tempo de execução além da quantidade de comparações e trocas de posições em relação a quantidade de elementos a serem ordenados durante determinada entrada, sendo essas entradas ordenadas em ordem crescente, decrescente e aleatória.

São objetivos deste trabalho demonstrar o tempo de execução, quantidade de comparações e quantidade de trocas de elementos que são realizadas para as entradas de variados tamanhos e encontrar qual dos algoritmos realiza o processo de ordenação das entradas com menor custo, ou seja, menor tempo.

A metodologia utilizada foi a implementação dos algoritmos utilizando a linguagem de programação python em sua versão 3, implementando a lógica de capturar a quantidade de processos realizados e tempo adicionando os mesmo em listas e utilizados para plotar os gráficos utilizando a biblioteca matplotlib do python para realizar uma análise visual.

Todos os algoritmos foram executados em um notebook pessoal, com processador i5 de 7° geração, com 12Gb de memória ram ddr4, e SSD como disco principal.

DESENVOLVIMENTO

Algoritmos (**Bubble Sort, Insertion Sort, Selection Sort e Merge Sort**)

Ao implementar os algoritmos e realizar os testes obtivemos os gráficos de tempo de execução, quantidade de comparações e trocas.

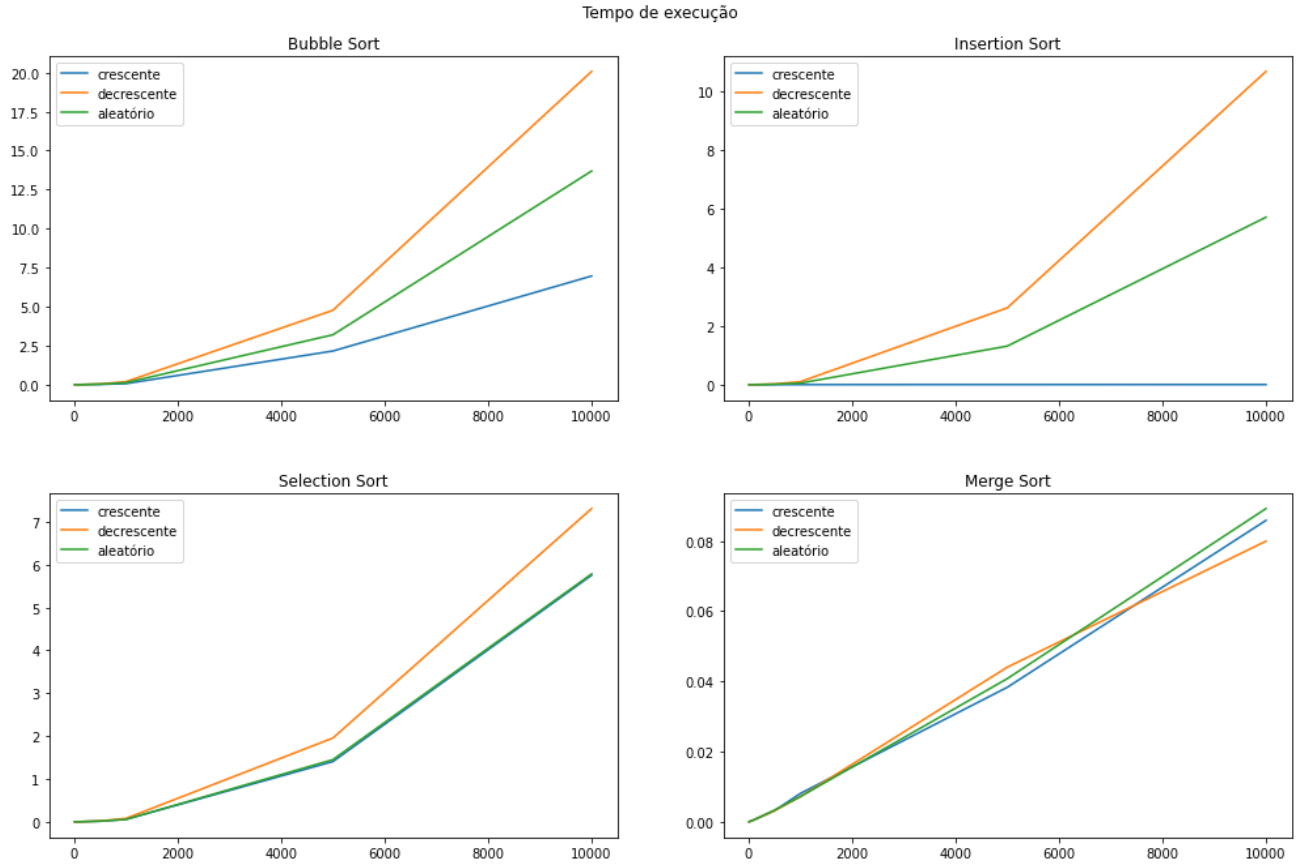

Figura 1. Gráfico de tempo de execução

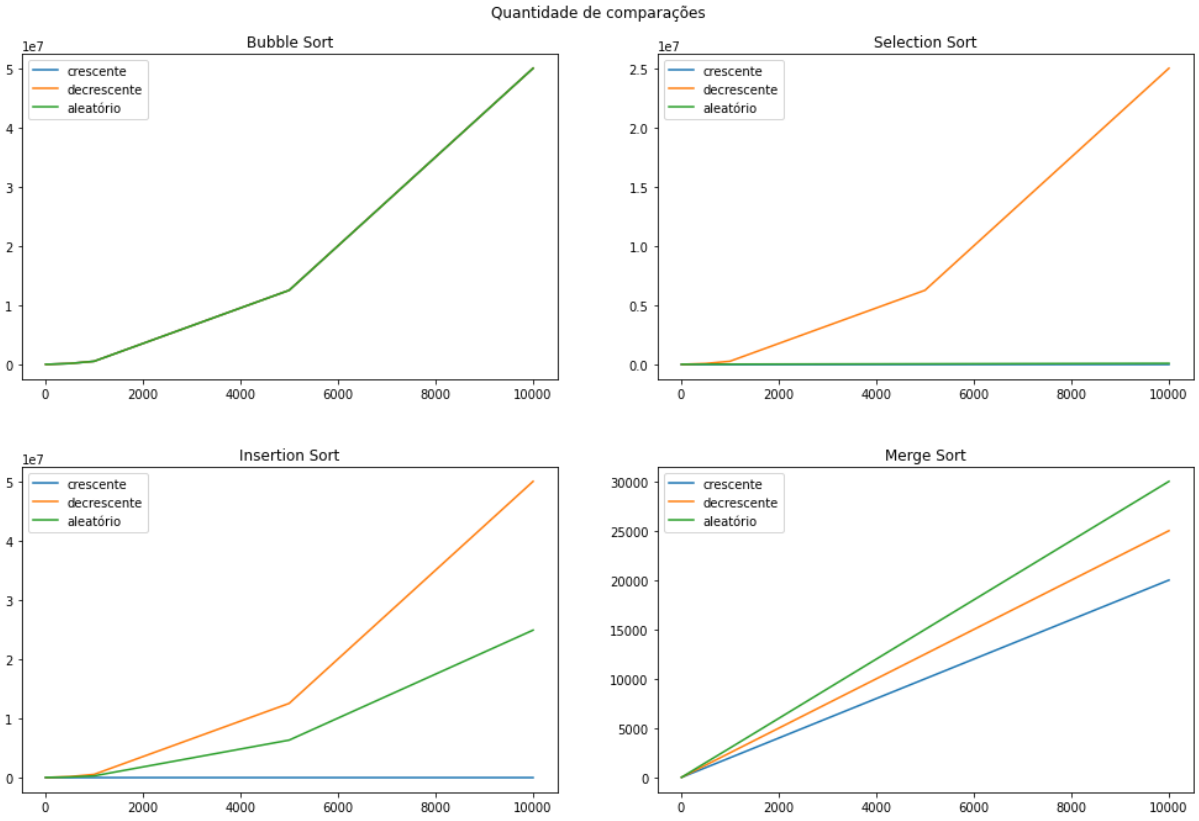

Figura 2. Gráfico de quantidade de comparações

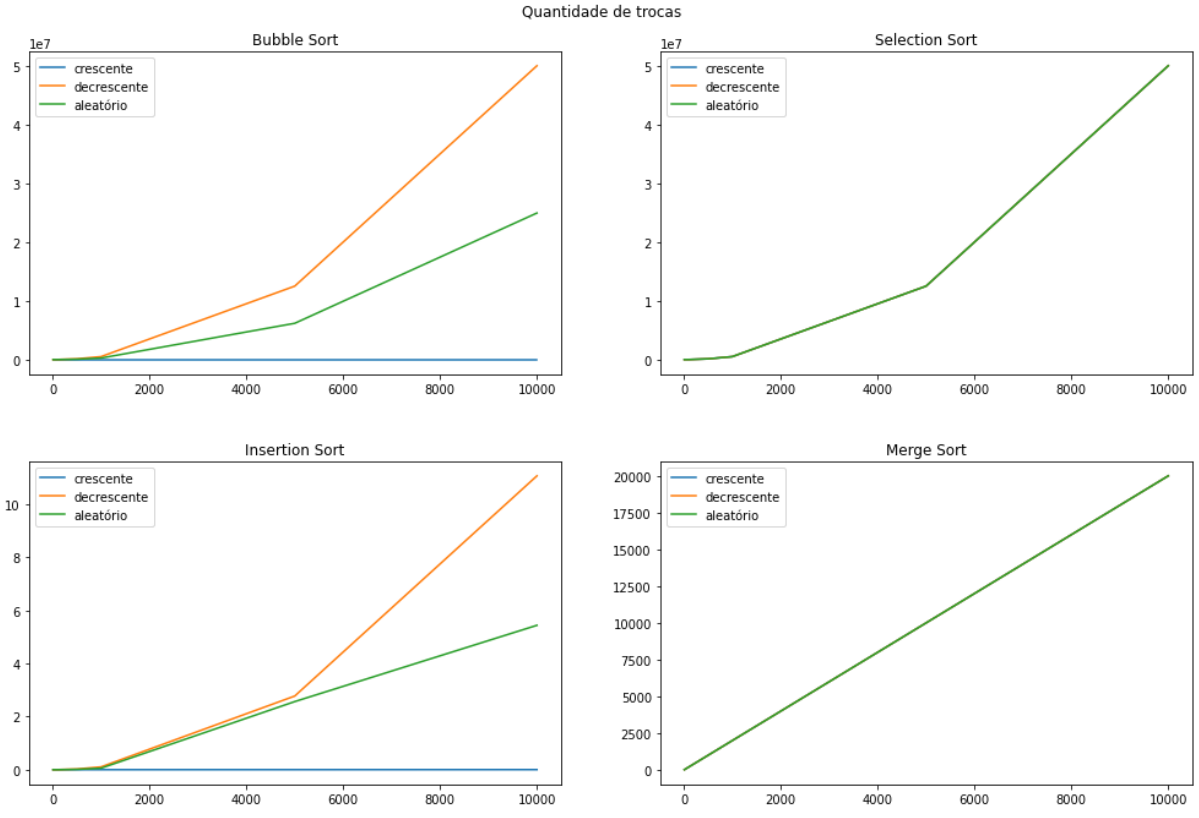

Figura 3. Gráfico de quantidade de trocas

Algoritmos (**Counting Sort, Quick Sort, Radix Sort e Heap Sort**)

Figura 4. Tempo de execução Counting Sort

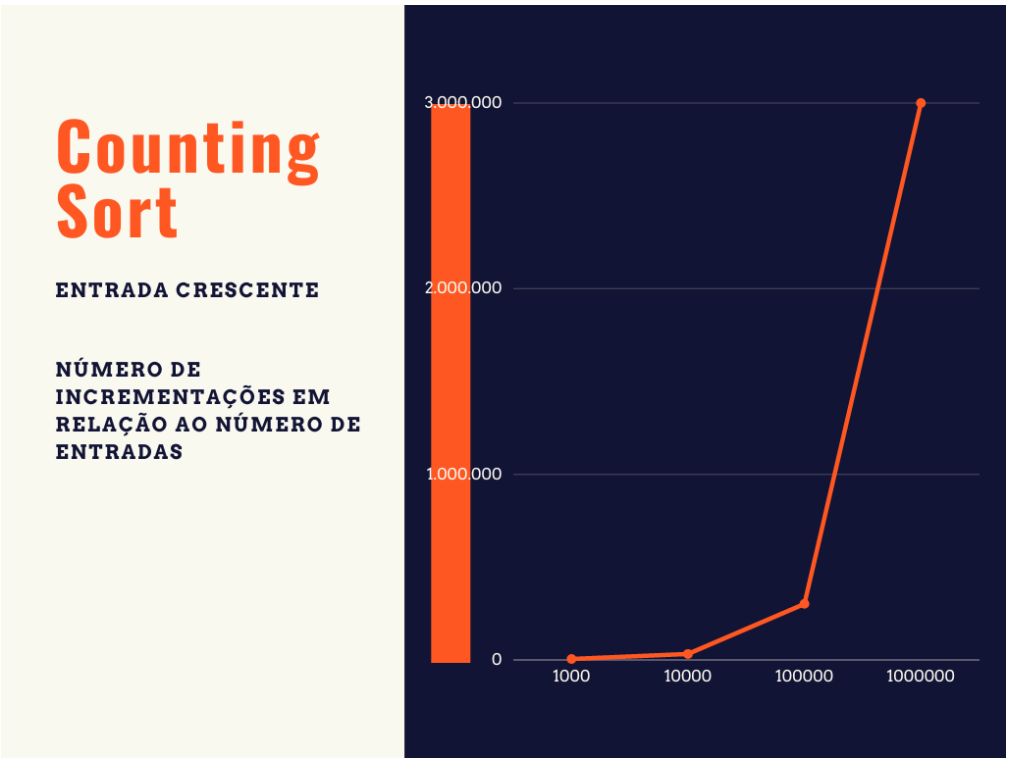

Figura 5. Quantidade de incrementações Counting Sort

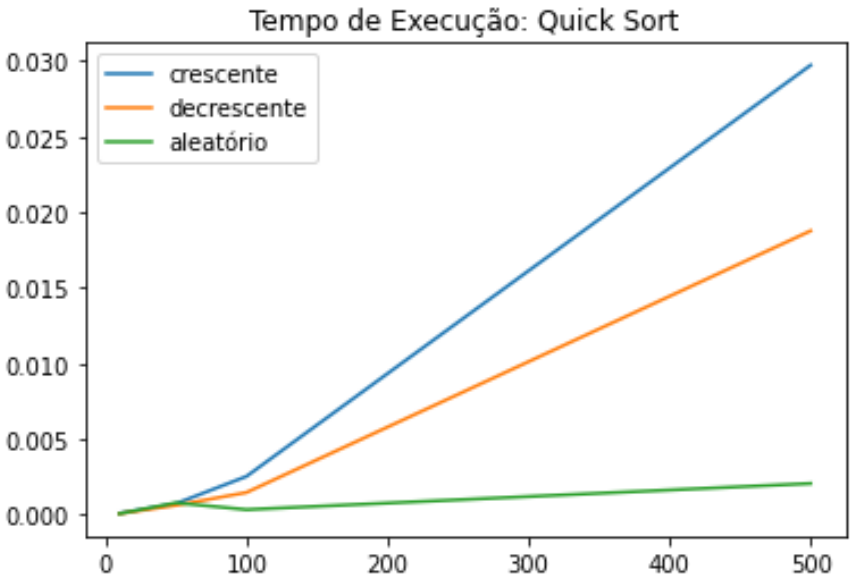

Figura 6. Tempo de execução Quick Sort

Figura 7. Tempo de execução Radix Sort

Figura 8. Tempo de execução Heap Sort

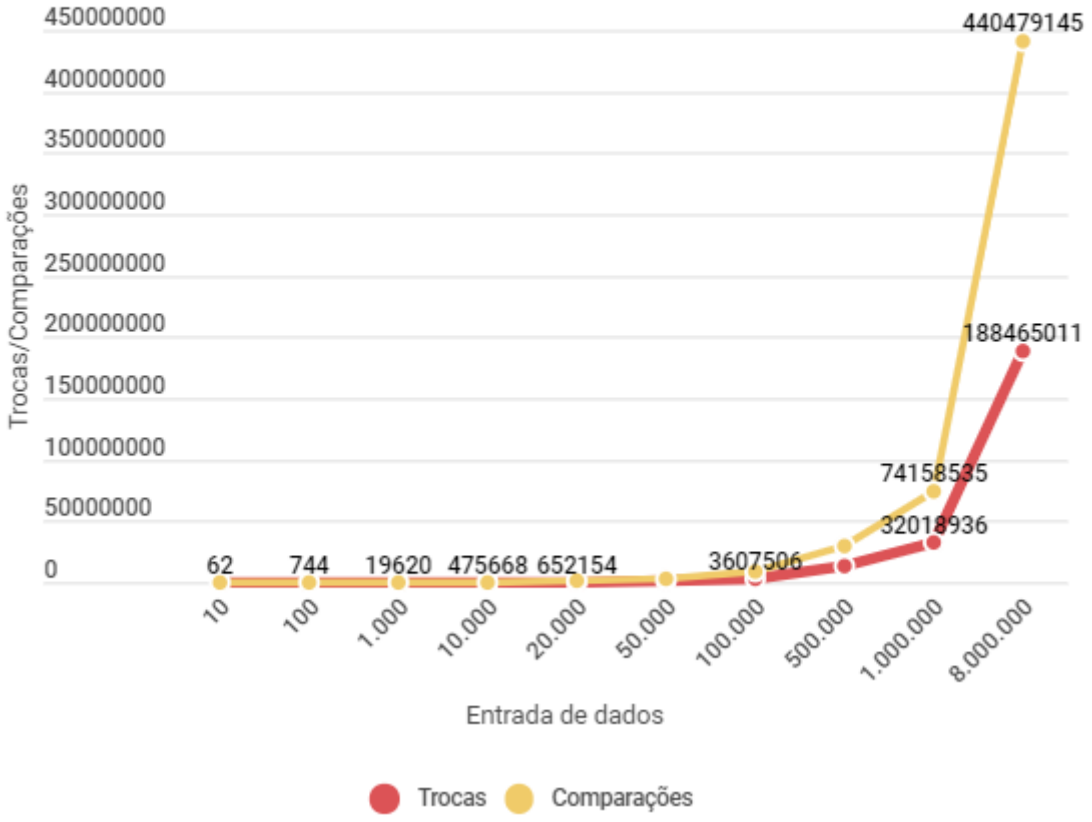

Figura 9. Quantidade de trocas e comparações Heap Sort

CONCLUSÃO

Neste trabalho abordamos sobre a eficiência na execução de oito algoritmos de ordenação por meio da análise gráfica de tempo de execução, quantidade de comparações e trocas realizadas por cada algoritmo.

Por meio dessa análise foi possível concluir que os algoritmos Bubble Sort, Insertion Sort e Selection Sort não são tão eficazes para ordenação de arrays muito grandes, em especial o Bubble Sort que ao executar a ordenação em arrays de tamanho superior a 10 mil elementos tinha um custo de tempo muito alto até mesmo em comparação ao Selection e o Insertion que também tiveram grande custo.

Partindo para os algoritmos mais eficientes do teste (Merge Sort, Counting Sort, Quick Sort, Radix Sort e Heap Sort) aqui todos conseguiram ordenar arrays de tamanho até 10 milhões com muita facilidade e em poucas frações de segundos, e por meio dos gráficos e testes realizados no terminal concluímos que todos esses cinco tiveram ótimo desempenho porém o Merge Sort foi o algoritmo com menor custo de tempo, ordenando arrays de 8 milhões de elementos em apenas 17 segundos e arrays de até 20 mil elementos em menos de 1 segundo.

REFERÊNCIAS

GRAMAXO, Warley. Algoritmos de Ordenação. Ava UFT. 2022.

MAGNUM, Lucas. Algoritmos de Ordenação. Pythonclub. 2018. Disponível em <http://pythonclub.com.br/algoritmos-ordenacao.html>. Acesso em: 24/10/2022.

Merge Sort. Algoritmos em python. Disponível em <https://algoritmosempython.com.br/cursos/algoritmos-python/pesquisa-ordenacao/mergesort/>. Acesso em: 24/10/2022.