**Raphael Wilson Paulon da Silva.**

**202203192116.**

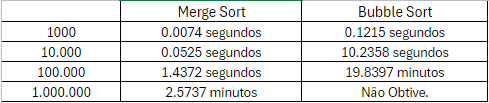
**Introdução**

Os algoritmos de ordenação são ferramentas que permitem ordenar grandes quantidades de dados de maneira mais fácil e precisa. Os algoritmos escolhidos para o desenvolvimento deste trabalho foram o Merge Sort e o Bubble Sort, a complexidade desses algoritmos são O(n log n) referente ao Merge Sort e O(n^2) sendo referente ao Bubble Sort, a seguir irei mostrar gráficos que comparam o tempo de execução entre esses dois algoritmos com 1.000,10.000,100.000 e 1.000.000 números aleatórios, onde a linguagem de programação escolhida foi a linguagem Python ( link Github: https://github.com/RaphaelZ9/Trabalho-Algoritmos-e-Complexidade).

**Resultados**

No link do Github está presente todos os códigos com as funções para os dois algoritmos citados acima, o objetivo é criar um vetor que gera elementos aleatoriamente, para cada algoritmo de ordenação citado (Merge e Bubble). Observação: Ao tentar executar o vetor de 1 milhão junto em apenas um código não obtive sucesso, então realizei a separação em duas partes para os dois algoritmos, entre tanto mesmo assim somente o código do algoritmo Merge sort me disponibilizou resultados.

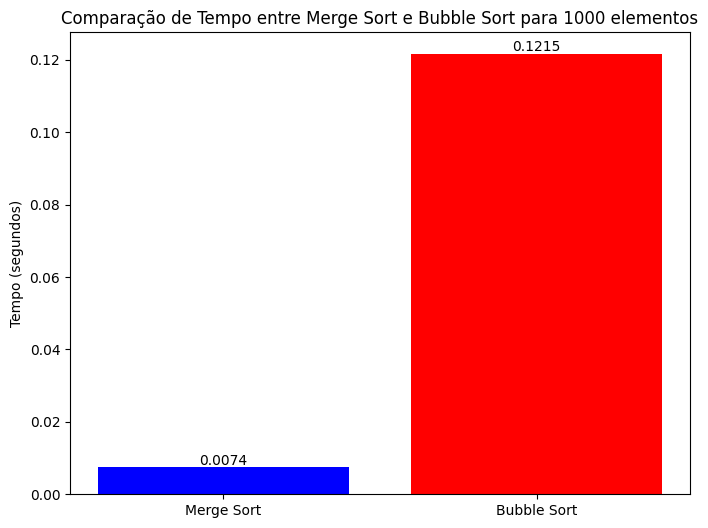
* As configurações da máquina realizadora destes testes foi: processador Ryzen 5 3400g, 500GB de ssd, 3 TB de HD, 16GB de memória ram, placa de vídeo GTX 1650 4GB de ram, o Google Colab foi utilizado para a efetuação dos códigos.
* Tabela de Tempo de execução para os algoritmos Merge e Bubble:



Explicação da Tabela: Efetuando uma análise rápida pode se perceber, que o Bubble sort tem um tempo de execução elevado devido a sua complexidade o tornando ineficaz para certas situações, onde vetores muito grandes se tornam impossível de ser executado em algumas máquinas. Diferentemente do Merge que sua complexidade, é mais eficaz sendo possível obter o valor de 1.000.00, sendo um tempo de execução moderado.

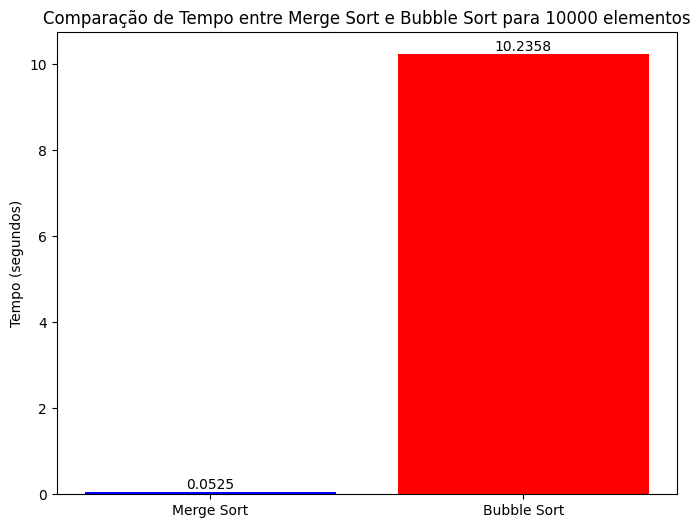
* Gráficos comparando execução:

1° Gráfico:

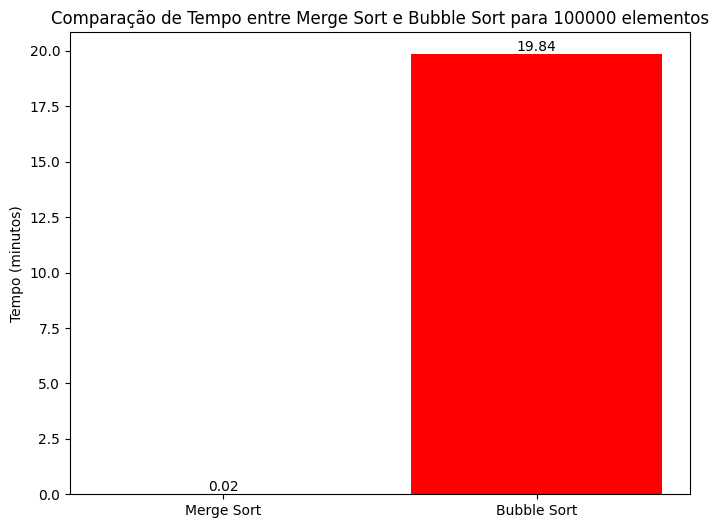


De acordo com gráfico acima conseguimos comparar o tempo de execução dos dois algoritmos, com 1.000 elementos, assim conseguimos ver a diferença de execução dos dois algoritmos sendo o Azul o Merge com o tempo de 0.0074 segundos e o Bubble em vermelho tendo o tempo de 0.1215.

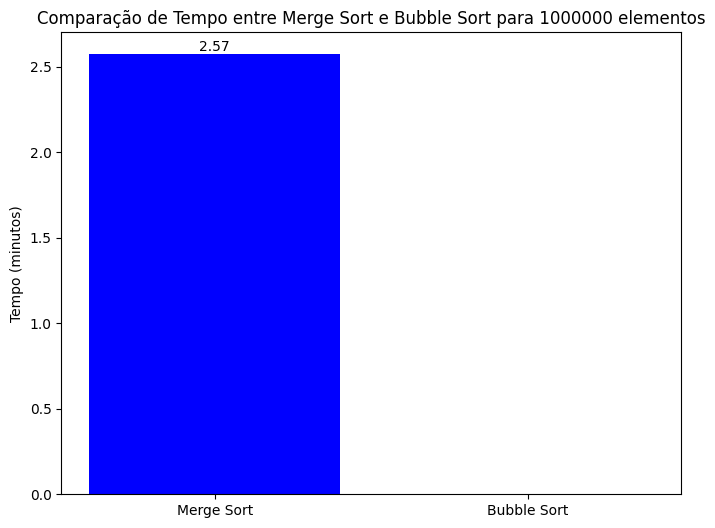
2° Gráfico:



De acordo com este gráfico acima conseguimos comparar 10.000 elementos, o Bubble aumenta seu tempo de execução significativamente causando uma demora maior, o Merge diferentemente do Bubble demora um pouco a mais apenas, logo conseguimos tirar a conclusão de que em mais um caso o Merge consegue ser mais eficiente do que o Bubble nessa situação gerada de 10.000 elementos.

3°Gráfico:  


Conforme este gráfico conseguimos mostrar uma diferença entre os dois algoritmos muito grande, mostrando que o Merge consegue executar os 100.000 elementos em um tempo muito curto, já o Bubble com um tempo muito superior em comparação com o Merge levando cerca de 19 para obter resultados, logo mostrando que o Merge com sua complexidade de tempo de O(n log n), é muito mais eficiente que a complexidade do Bubble.

4°Gráfico:  
  


Conforme o gráfico acima podemos ver que apenas o Merge foi capaz de executar o código com eficiência, onde foi preciso quase 3 minutos para obter a ordenação de 1.000.000 de elementos, já o Bubble com sua dificuldade com grandes conjuntos de dados, precisaria de dias ou semanas para o Bubble obter resultado assim o tornando inviável.

**Conclusão**

Com todo desenvolvimento do projeto, tivemos a oportunidade de perceber que se você precisar executar um algoritmo de ordenação muito grande o Bubble não é o indicado pois demoraria muito para obter o resultado solicitado, mas se você quiser executar um vetor que não seja tão grande como o de 1.000.000, tanto o Merge Sort quanto o Bubble Sort podem ser utilizados entre tanto, o Merge traz um resultado em um período muito menor.

OBS: Vale ressaltar que todos os testes foram realizados em uma máquina pessoal com uma certa configuração, onde caso seu hardware for diferente sendo superior ou inferior, o resultado da execução do código, pode ser mais rápido ou mais demorada do que as mostradas neste projeto.