Relatório do Trabalho AV

Algoritmos e Complexidade – 2024.1

Paulo Vitor Silva Quintanilha

202202780952

Introdução

Algoritmos de ordenação são métodos que permitem que os programadores possam reduzir a complexidade de um problema, eles são utilizados em diversas áreas da computação, desde processamento de dados até otimização de algoritmos, os dois algoritmos que eu irei tratar nesse artigo é o Bubble sort que é um é um algoritmo de ordenação simples, e a sua complexidade de tempo é de O(n^2) e o Merge sort é um algoritmo de ordenação que usa a técnica de divisão e conquista e sua complexidade de tempo é de O(n log n),abaixo 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, a linguagem de programação escolhida foi a linguagem Python, caso queira acompanhar segue o link do repositório: Link do repositório

Resultados

O link do repositório possui todos os códigos com as funções para os dois algoritmos escrito acima neste artigo, o alvo é criar um vetor que gera elementos aleatoriamente de 0 a 99, para cada algoritmo de ordenação, restaurar o vetor ao seu estado original antes de realizar a ordenação e isso garante que cada algoritmo possa operar sobre a mesma coletânea de dados, assim podendo ter uma comparação justa para o tempo de execução para os dois algoritmos. Observação: Para executar o vetor de 1 milhão ao tentar executar junto não obtive êxitos, então ao separar em duas partes para os dois algoritmos, obtive êxito só no código onde estava inserindo o algoritmo do Merge sort.

- Configuração usada: Ryzen 5 5600G, 16GB de memória ram, 256GB de ssd, placa de vídeo GTX 1650 com 4GB de ram e usei o Google Collab para executar os códigos.
- Tabela de Tempo de execução para os dois algoritmos:

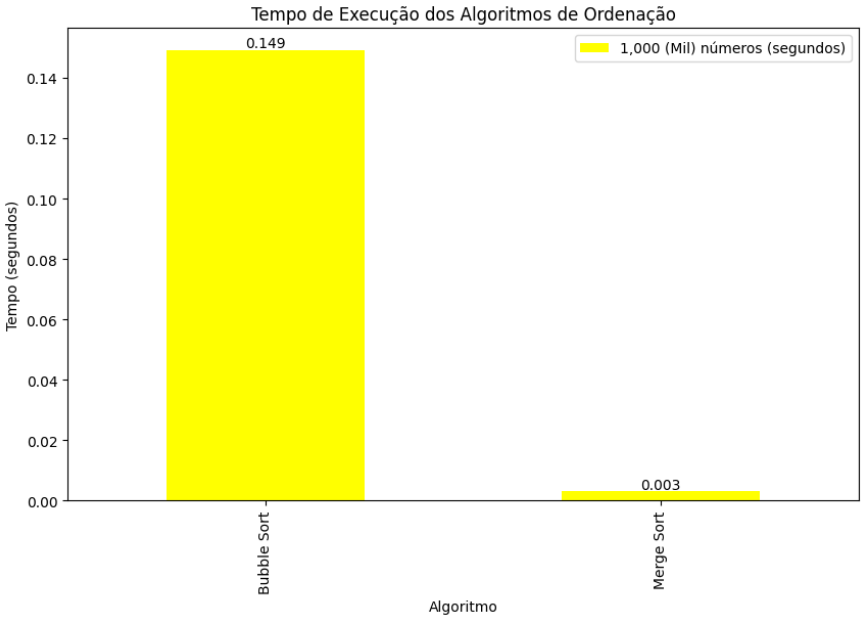
Algoritmo	1.000 (Mil) números (segundos)	10.000 (Dez mil) números (segundos)	100.000 (Cem mil) números (segundos)	1.000.000 (Um milhão) números (segundos)
Bubble Sort	0.149	18.618	1773.691	Inconclusível
Merge Sort	0.003	0.039	0.484	6.319

Explicação da Tabela: Analisando essa tabela podemos ver que o Bubble sort ele é muito ineficaz, o tempo de execução cresce quadraticamente devido a sua complexidade que é

O(n^2), por isso que com vetores como o de 1.000.000 elementos torna o impraticável. Já com o Merge sort com sua complexidade de O(n log n), ele é muito mais eficaz, pois o tempo de execução cresce de forma mais controlada, por isso que com um vetor de 1 milhão de elementos, o tempo de execução dele é ponderado.

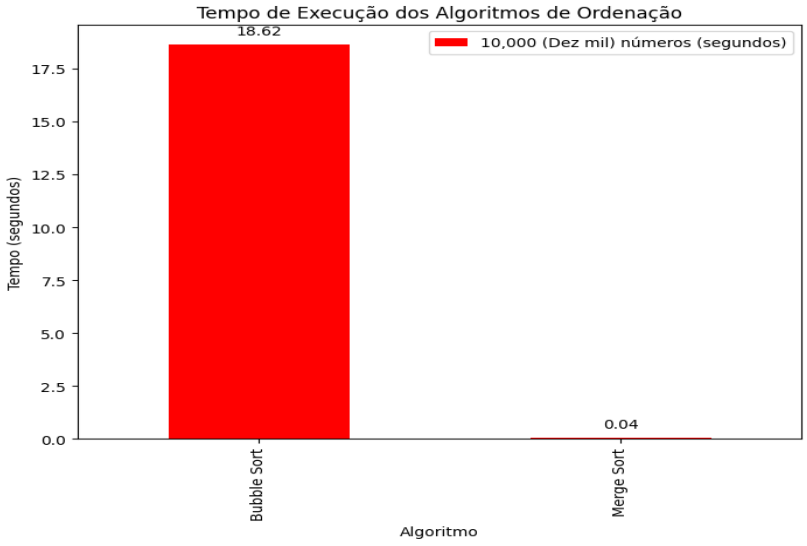
- Gráficos para comparar o tempo de execução entre os dois algoritmos:

Gráfico 1:



Nesse gráfico nos permite ver que o tempo de execução dos dois são eficientes para esse conjunto de dados de 1.000 elementos, mas já podemos ver a diferença de execução dos dois o Bubble que leva 0.149 segundos e o Merge que leva 0.003 segundos.

Gráfico 2:



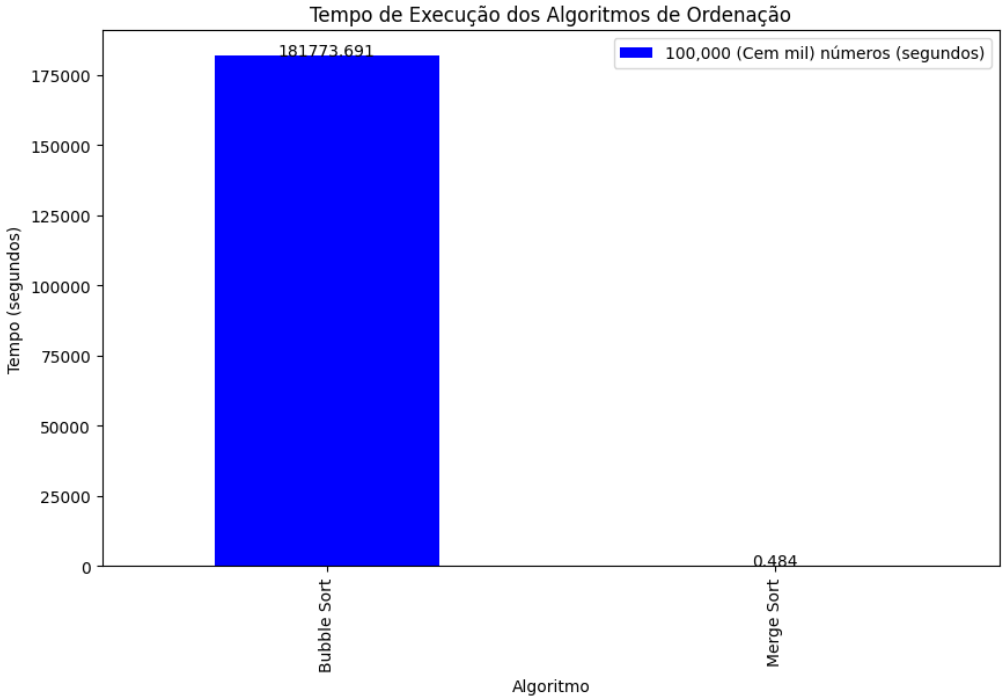
Estácio

Relatório do Trabalho AV

Algoritmos e Complexidade – 2024.1

Nesse gráfico comparado ao outro já podemos ver que com 10 mil elementos o Bubble ele aumenta significativamente levando aproximadamente 18.62 segundos e o Merge levando apenas 0.04 segundos, assim podendo concluir que para 10 mil elementos o Merge é bem mais eficiente.

Gráfico 3:



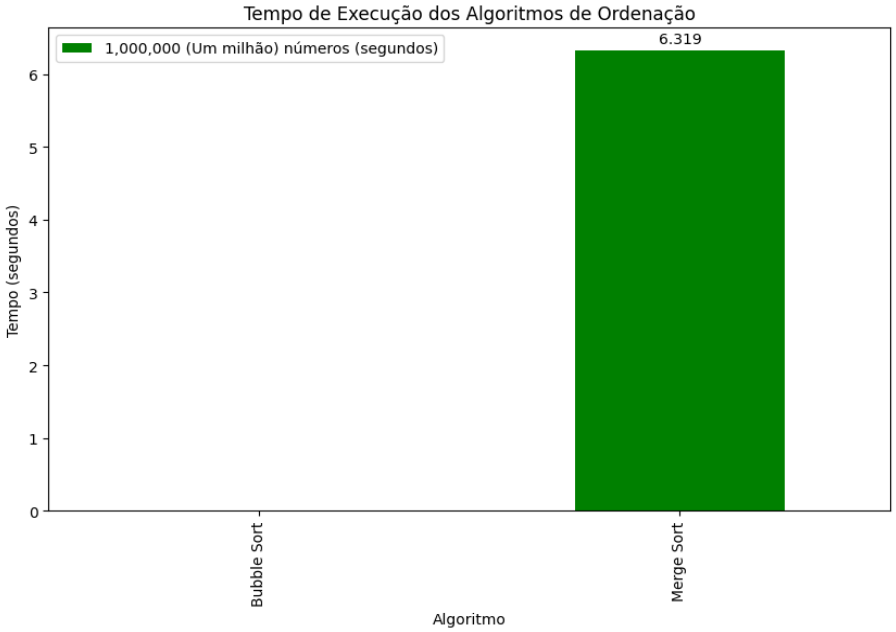
Neste gráfico mostra ainda mais a diferença entre os dois algoritmos, enquanto o Buble mostra a sua ineficiência em conjunto de dados maiores pois o tempo de execução cresce de forma quadrática, assim levando cerca de 181773.691 segundos, já por outro lado o Merge leva um tempo de execução estável mesmo para um grande conjunto de dados, levando apenas 0.484 segundos e isso se deve à sua eficiência e escalabilidade, uma vez que possui uma complexidade de tempo de O(n log n), que é muito melhor do que a complexidade quadrática do Bubble.

Gráfico 4:

Estácio

Relatório do Trabalho AV

Algoritmos e Complexidade – 2024.1



Nesse gráfico aqui podemos ver que o único capaz de rodar o código com eficiência foi o Merge, que precisou apenas de menos de 7 segundos para ordenar 1 milhão de elementos de forma aleatória, já o Bubble com sua ineficiência para grandes conjuntos de dados como o de 1 milhão de elementos, para rodar o Bubble seria necessário dias ou até semanas, assim levando a ser inconclusivo.

Conclusão

Então podemos perceber que através desses gráficos e da tabela, que se você quiser executar um algoritmo de ordenação em pequenos vetores os dois serão eficientes, mas se você quiser executar em grandes vetores, o ideal vai ser o Merge Sort.

Lembrando que esses algoritmos foi rodado no meu computador e dependendo do seu hardware ou onde você executar o código, pode ser mais rápido ou mais demorado.