Equipe: Davi Sousa Soares
Victor Kauan da Silva Miranda
João Pedro Saleh de Sousa

Professor: Raimundo Santos Moura

SETEMBRO
2024

Conteúdo

1 Introdução

Esse trabalho será destinado a observar a execução de determinados algoritmos de ordenação em suas tarefas, sendo eles o Heapsort, o Quicksort, o Mergesort,o Shellsort,o Bubblesort,o Insertionsort e o Selectionsort.

1.1 Objetivos

O objetivo deste trabalho é medir o tempo gasto pelos algoritmos citados executando em arquivos de 50.000, 100.000, 250.000, 500.000 e 1.000.000 elementos.
Os algoritmos são divididos em 2 grupos, o grupo 1 realiza em média O(n log(n)) operações, sendo n o número de elementos no conjunto. Fazem parte desse grupo o Heapsort, o Quicksort,o Mergesort e o Shellsort.
Já o grupo 2 realiza em média $O(n^2)$ operações. Fazem parte desse grupo o Bubblesort,o Insertionsort, e o Selectionsort.

2 Funcionalidades

O código em si é separado em 4 módulos para que seja facilitado entendimento e diminuir a quantidade de linhas do mesmo.
O Algoritmos.py contém todos os algoritmos de ordenação em forma de funções, além de uma função decorator que calcula quanto tempo leva do início ao fim da função.
O Menu contém as operações que podem ser realizadas para medir os tempo ,realiza_operação calcula o tempo gasto por uma função, checa _se_ta_ ordenando transfere a lista ordenada para um arquivo para que seja feita a checagem de que o algoritmos está funcionando adequadamente e o calcula_media que calcula a média de tempos de um algoritmo em determinado arquivo
O sort Quadráticos cria o gráfico dos algoritmos quadráticos

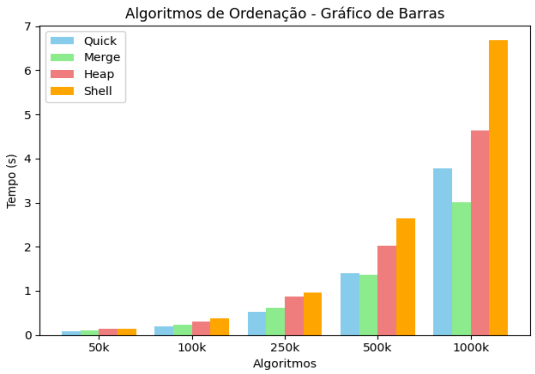
O sort Logaritmos cria o gráfico dos algoritmos não quadráticos

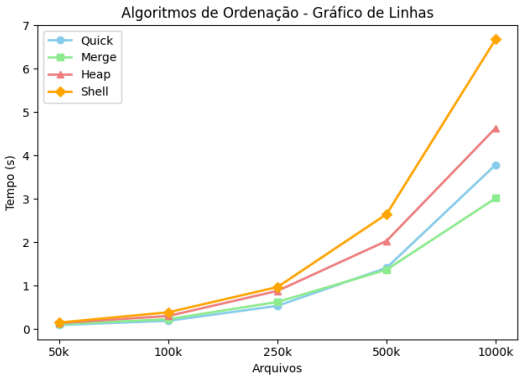
2.1 Especificações

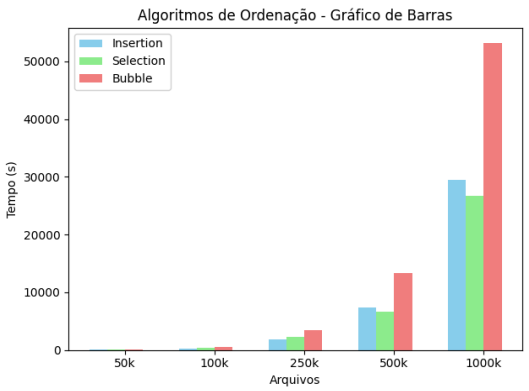
Origem de cada algoritmos:

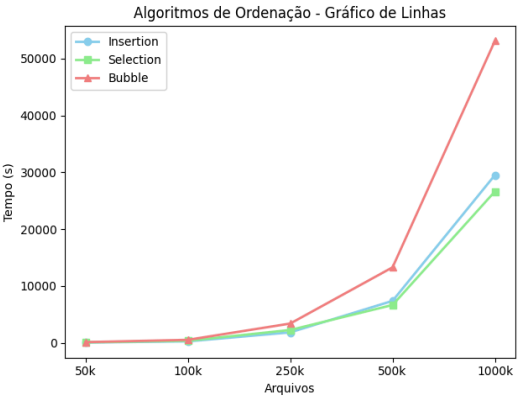
Heapsort: https://www.geeksforgeeks.org/heap-sort/
Quicksort: https://www.geeksforgeeks.org/quick-sort-algorithm/
Mergesort:Victor Kauan
Shellsort: https://pt.wikipedia.org/wiki/Shell_sort modificado por João Pedro Saleh
Insertionsort: https://www.geeksforgeeks.org/insertion-sort-algorithm/ modificado por Davi Sousa
Selectionsort: https://www.geeksforgeeks.org/selection-sort-algorithm-2/
Bubblesort:https://www.geeksforgeeks.org/bubble-sort-algorithm/

2.2 Gráficos









2.3 Conclusões

- Mesmo aumentando significativamente o número de elementos, os algoritmos do grupo 1 não tiveram um aumento de tempo considerável , sendo possível de realizar a média de 5 execuções de cada um em pouco menos de 1 minuto
- A única inconsistência na comparação dos resultados do grupo 1 é a partir dos 500.000 ,onde o quick se torna mais lento que o merge, mas fora isso a classificação dos tempos permanece igual

- O shell é consideravelmente mais lento que seus colegas de grupo, ainda assim percorre 1.000.000 elementos mais rápido que qualquer um do grupo dois percorre 50.000 elementos
- O grupo 2 se provou difícil de comparar desde o primeiro arquivo de elementos já que os 3 juntos levam cerca de 3 minutos
- A execução das médias do grupo 2 levou cerca de 4 dias para construir o gráfico, com a exclusão do arquivo de 1.000.000 elementos,que foi feita usando uma estimativa baseada nos resultados dos exemplos anteriores
- O tempo nos algoritmos quadráticos pode ser lido como 0, 2.7, 5.5, 8.3,11.11 e 13.8 horas respectivamente
- O bubble é o algoritmo mais ineficaz
- O quick e o merge variam entre os mais eficazes