Instituto Federal Goiano – Campus Urutaí

BRUNO SAMUEL DA SILVA

Comparação e Medidas de Tempo Entre Algoritmos de Ordenação

Estrutura de Dados 2

Urutaí

2022

Rodovia Geraldo Silva Nascimento, Km 2.5, Zona Rural, Urutaí - Goiás CEP: 75790-000 Email: bruno.samuel1@estudante.ifgoiano.edu.br
www.ifgoiano.edu.br/urutai

Tabela representando o tempo de execução (média de 50 aplicações):

	Quantidade de elementos									
	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
InsertionSort	6,17	8,24	13,97	17,32	20,43	23,43	25,27	28,85	31,11	34,62
ShellSort	1	1,39	1,62	1,57	1,54	1,59	1,5	1,5	1,48	1,55
QuickSort	1,34	1	1	1	1	1	1	1	1	1
HeapSort	2,64	1,5	2,08	1,57	1,51	1,5	1,46	1,43	1,44	1,47

Gráfico de linhas aprensentando o resultado:

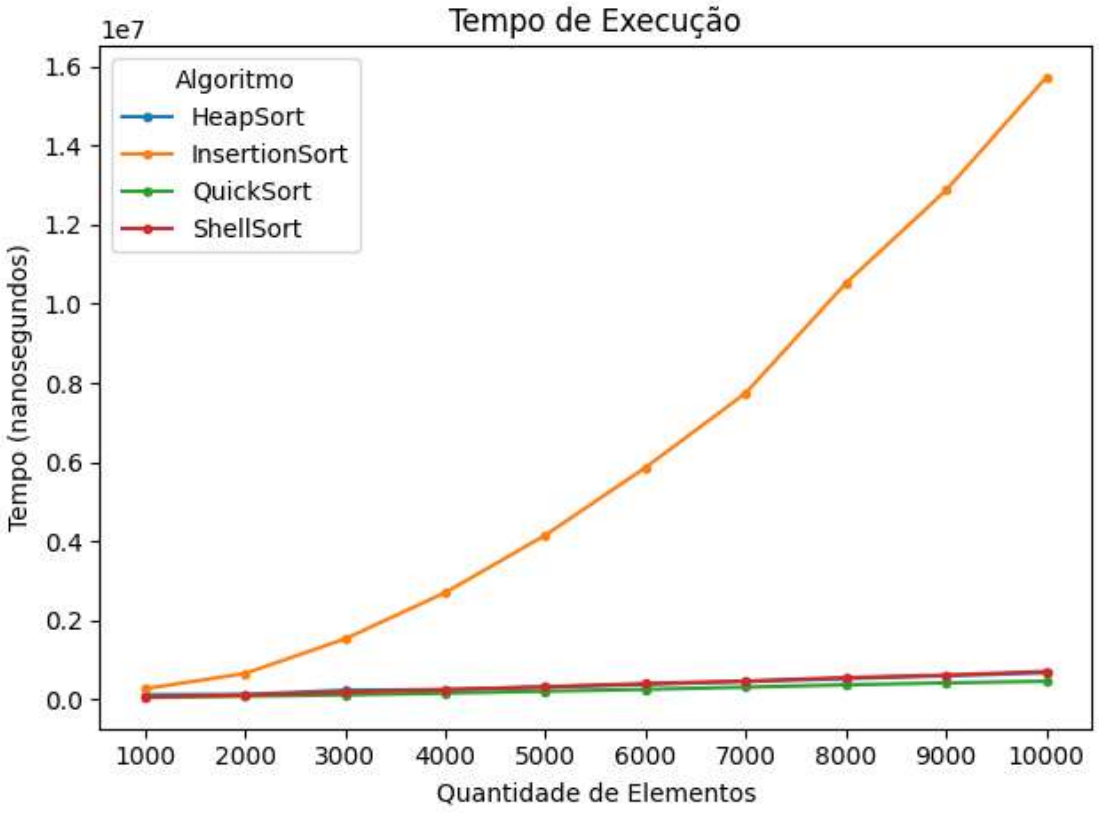
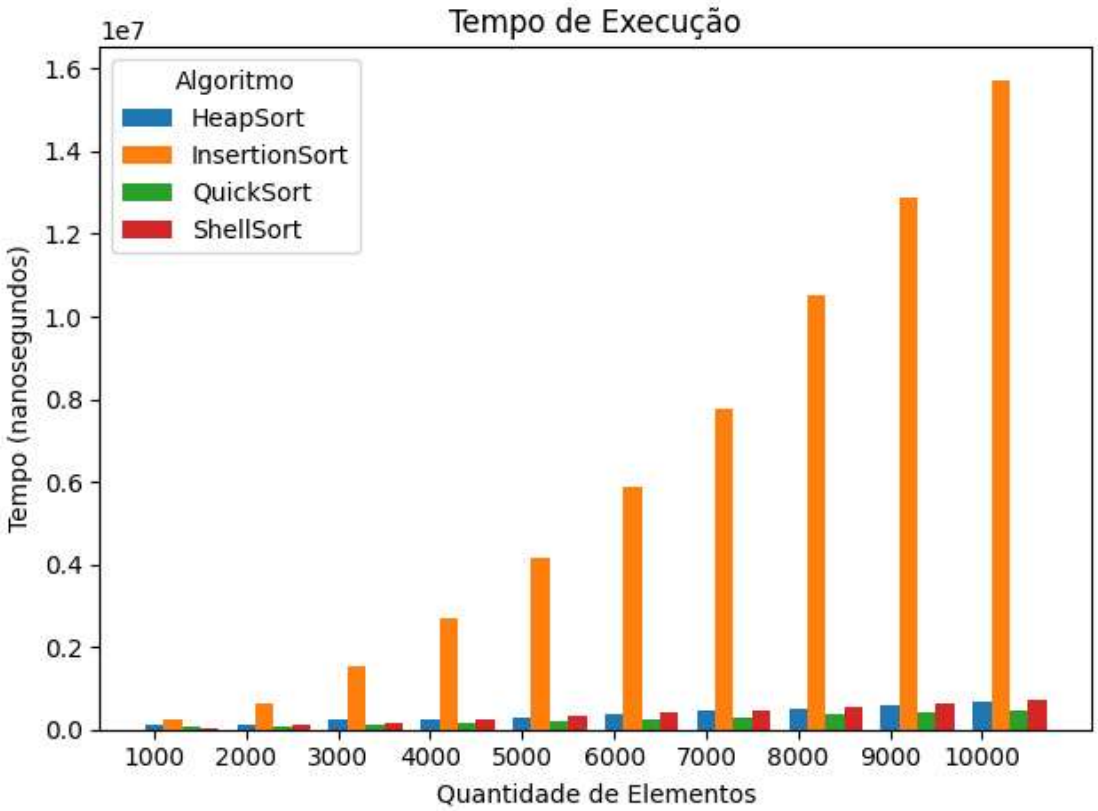


Gráfico de barras:



Notas

As funções de ordenação foram implementados em java e baseados nos algoritmos apresentados em sala de aula. Os gráficos foram gerados utilizando a linguagem python juntamente com as biblioteca numpy e matplotlib.

Os resultados estão dentro do esperado, é possível visualizar que o algoritmo de InsertionSort tem um aumento de tempo exponencial, próximo a n^2, já os demais algoritmo possuem resultados bem semelhantes, mas o algoritmo QuickSort tem vantagem, perdendo para o ShellSort apenas em vetores com mil elementos.

Ambos os algoritmos HeapSort, QuickSort e ShellSort tiveram aumento de tempo aparentemente logarítmico, mas possuem aplicações diferentes, o QuickSort mesmo tendo vantagem não deve ser usados em aplicações de tempo real, pois possui pior caso $O(n^2)$, o HeapSort possui pior caso e caso

médio em $O(\log_2 N)$ e deve ser utilizados em aplicações de tempo real, mesmo tendo tempo médio acima do QuickSort. Até mesmo o InsertionSort pode ser aplicado em situações onde se sabe que os dados estão parcialmente ordenados (os dados estão próximos das suas posições de pós-ordenação), pois consegue terminar de ordenar os dados com poucas trocas.