Universidade de São Paulo - Instituto de Ciências Matemáticas e de Computação
SCC0220 - Laboratório de Introdução à Ciência da Computação II
Aluno: Gabriel Toschi de Oliveira (9763039)

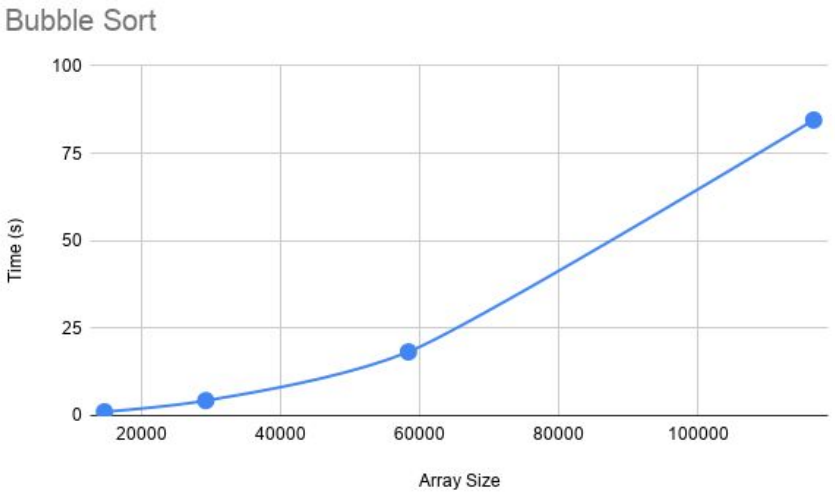
Atividade 4 Ordenação de Palavras

Introdução

Esta atividade consiste em exercitar os processos de medição de tempo de execução de algoritmos de ordenação. De acordo com a especificação, foram usados os algoritmos bubble sort, insertion sort e merge sort. Como as ordens de grandeza dos tempos médios são bem diferentes, não foi possível construir um gráfico único com os dois dados.

Algoritmo 1: bubble sort

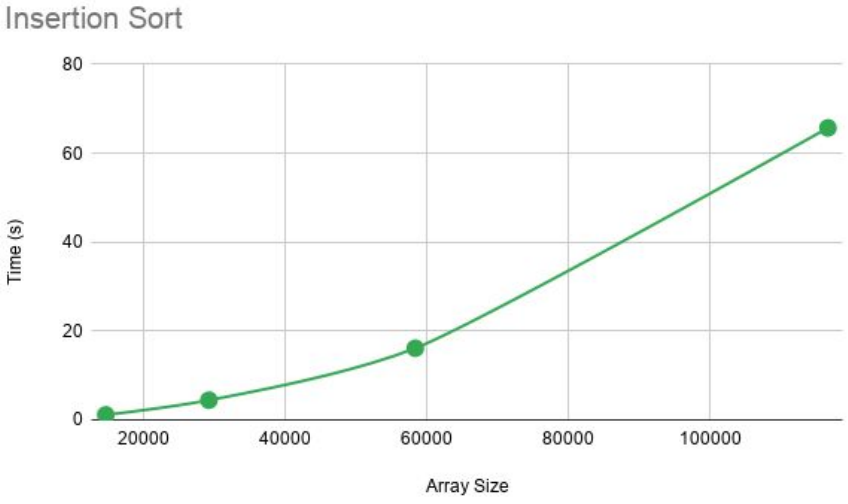
Entrada	Tempo (s)
14580	1,098543
29159	4,293295
58318	18,221255
116636	84,509604



Como já apresentado em atividades anteriores da disciplina, o **bubble sort** é conhecido por ter um **desempenho quadrático** em pior caso, **O(n²)**. Este fato é facilmente observado pelos resultados empíricos, como pode ser visto no gráfico acima.

Algoritmo 2: insertion sort

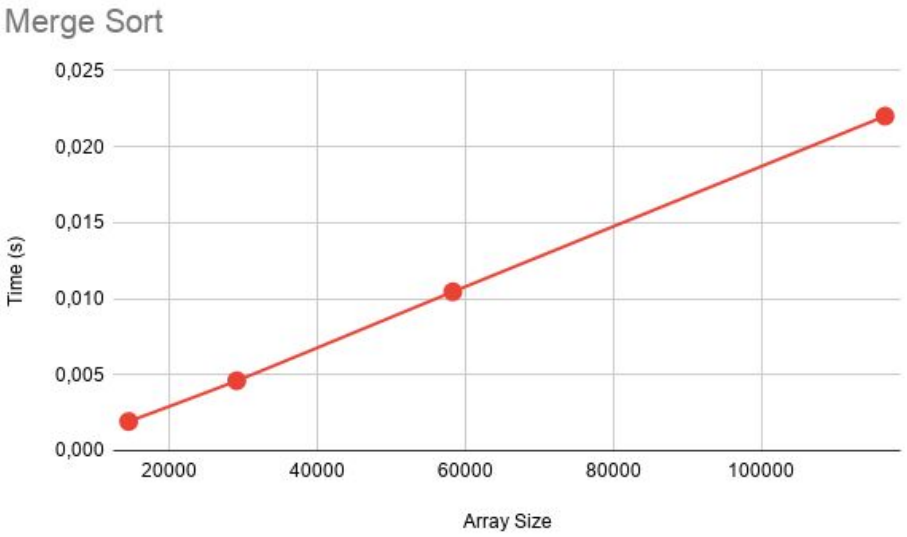
Entrada	Tempo (s)
14580	1,143626
29159	4,445481
58318	16,13293
116636	65,706715



É possível observar o **desempenho quadrático, O(n²), para o insertion sort** nos dados empíricos calculados durante a atividade. No pior caso, o algoritmo assume um comportamento de “bolha” semelhante ao bubble sort, levando um elemento à extremidade do array a cada iteração.

Algoritmo 3: merge sort

Entrada	Tempo (s)
14580	0,001921
29159	0,004603
58318	0,010454
116636	0,022012



Os dados empíricos do merge sort **podem apresentar um comportamento linear a uma primeira vista**, mas isto pode se dar pela quantidade pequena de entradas para equivaler-se ao seu comportamento teórico linear-logarítmico, **O(n log n)**.

Conclusão

Tanto a teoria quanto os resultados práticos mostram que **o merge sort é o algoritmo de ordenação mais eficiente entre os três analisados**, com tempos de execução muito menores — o que pode ser visto também pelo seu comportamento teórico. Entretanto, é importante apontar que o merge sort utiliza de vetores auxiliares durante o processo de merge, aumentando seu gasto de memória, o que não é feito pelos demais algoritmos. Quanto ao bubble sort e o insertion sort, por mais que ambos, no pior caso, tenham a mesma complexidade, o segundo provavelmente apresenta constantes menores que o primeiro.