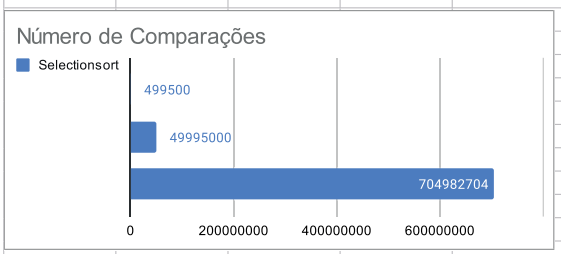
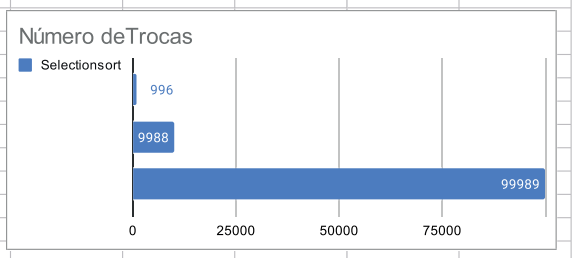
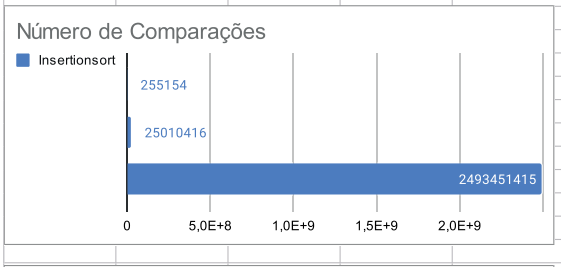
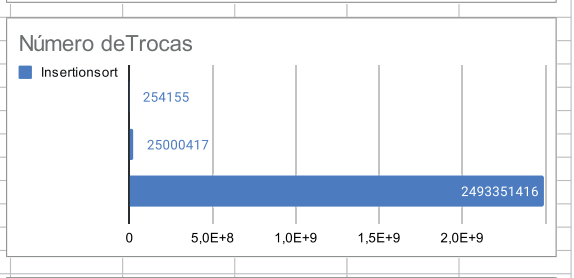
Número de Comparações			
Método	1000	10000	100000
Selectionsort	499500	49995000	704982704
Insertionsort	255154	25010416	2493451415
Bubblesort	999000	99990000	1409965408
Shellsort	13846	246515	3792451
Heapsort	25307	353079	4259112
Quicksort	17161	229366	2985290
Mergesort	15362	204892	2545557

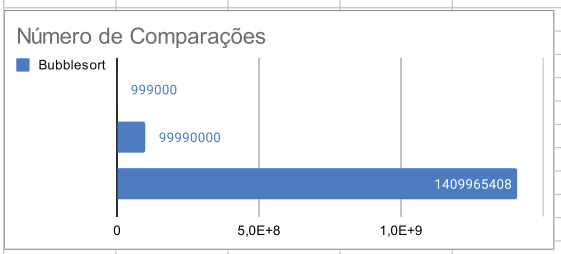
Número de Trocas			
Método	1000	10000	100000
Selectionsort	996	9988	99989
Insertionsort	254155	25000417	2493351416
Bubblesort	229860	22558745	2041961563
Shellsort	8389	171272	398148
Heapsort	9072	124252	2825305
Quicksort	2388	31515	390904
Mergesort	19952	267232	3337856

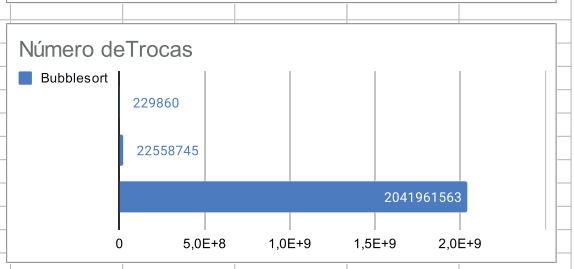


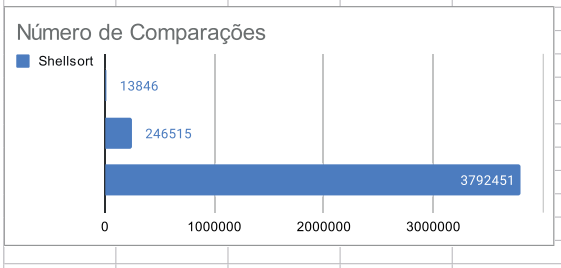


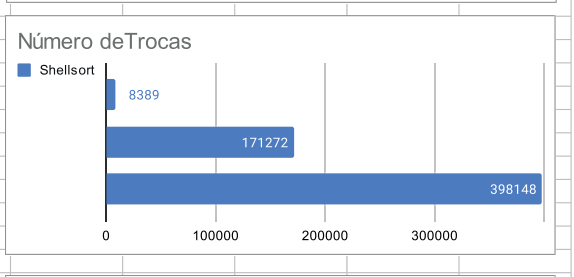


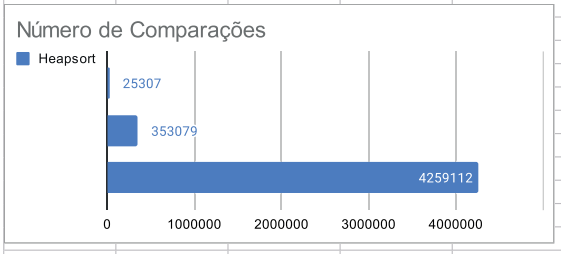


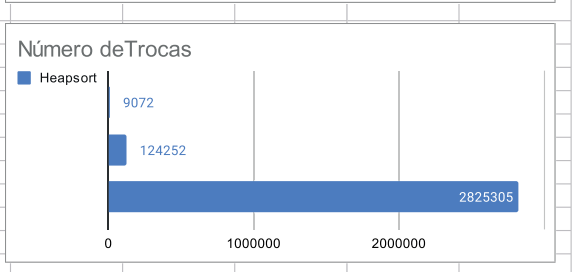


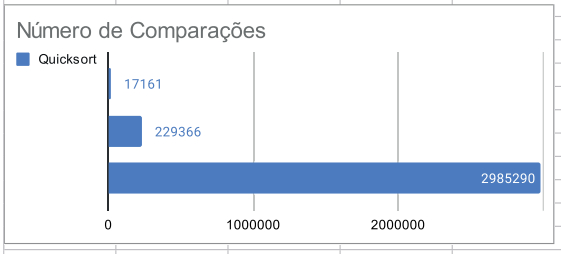


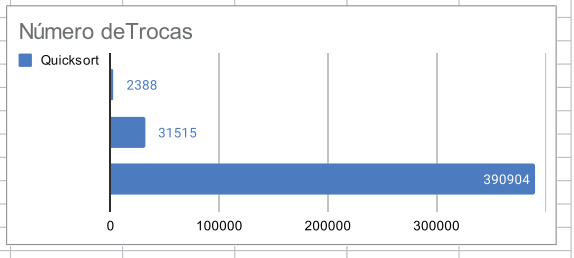


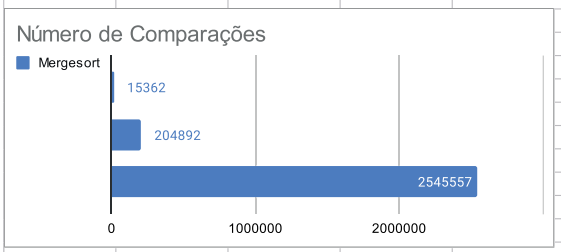


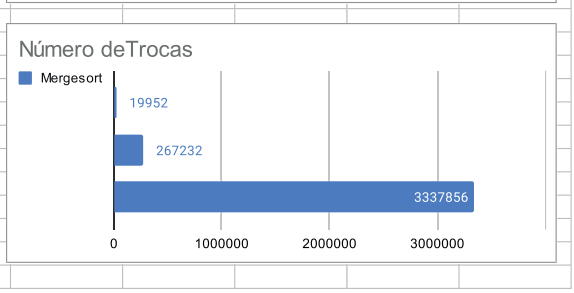












Durante o último exercício, vimos que o algoritmo Insertion Sort foi o mais eficiente em termos de comparações e trocas, especialmente em conjuntos de dados maiores. O Selection Sort também foi decente, mas um pouco mais lento que o Insertion Sort em conjuntos grandes. Por outro lado, o Bubble Sort foi o pior dos três, sendo significativamente mais lento e menos eficiente.

Agora, considerando os novos métodos como Shell Sort, Heapsort, Quicksort e Mergesort, pudemos ampliar nossa compreensão sobre diferentes formas de ordenação. O Shell Sort mostrou ser mais rápido que o Bubble Sort em conjuntos maiores. Heapsort, Quicksort e Mergesort têm vantagens únicas, como eficiência média, ordenação in-place e estabilidade.

1.000 elementos: O Insertion Sort pode ser o melhor devido à sua simplicidade e eficiência em conjuntos pequenos.

10.000 elementos: O Insertion Sort ainda pode ser uma escolha decente, mas métodos como Quicksort e Mergesort podem começar a se destacar.

100.000 elementos: Métodos mais eficientes como Quicksort e Mergesort são preferíveis, devido ao aumento no tamanho do conjunto de dados.

Ao comparar todos os métodos, agora podemos fazer escolhas mais informadas sobre qual usar, dependendo do tamanho e das características do conjunto de dados.

Durante o desenvolvimento do trabalho, enfrentamos dificuldades em entender e implementar os códigos dos métodos de ordenação. Todos os algoritmos foram um pouco complicados de implementar, exceto o Shell Sort, que se mostrou mais acessível. Além disso, percebemos que muitos desses métodos chamavam uns aos outros de maneira intricada, o que aumentou a complexidade do desenvolvimento. Uma outra dificuldade encontrada foi em relação à inserção de instruções de impressão para acompanhar o número de trocas e comparações dentro dos métodos de ordenação. Foi possível realizar essa inserção apenas no Shell Sort, devido à sua estrutura mais simples e direta.