Nome: Ruan Pablo Vitkoski de Souza
22/09/2023
GEX609 - PESQUISA E ORDENAÇÃO DE DADOS - T01 (2023.2)

Nome da Tarefa:
Trabalho Prático 2 (TP2)
Descrição:

Trabalho TP2- POD

Objetivo: comparar os métodos RadixSort e HeapSort com o desempenho dos algoritmos do trabalho TP1 em relação ao tempo de execução e ao número de trocas. **Gerar um relatório dos resultados.**

A comparação dos métodos irá resultar em **dois** gráficos variando o tamanho da **entrada** entre **100, 1000, 5000,10.000,50.000, 100.000**. O primeiro gráfico irá reportar o número de trocas realizadas por cada algoritmo em cada **entrada**. Já o segundo, irá reportar o tempo gasto por cada método para o ordenamento. É importante que o tempo seja a média de 3 execuções.

Deverá ser gerado um relatório explicando o comportamento das curvas. Por exemplo, o InsertionSort teve um número maior de comparações devido ao alto número de trocas conforme linha X do algoritmo.

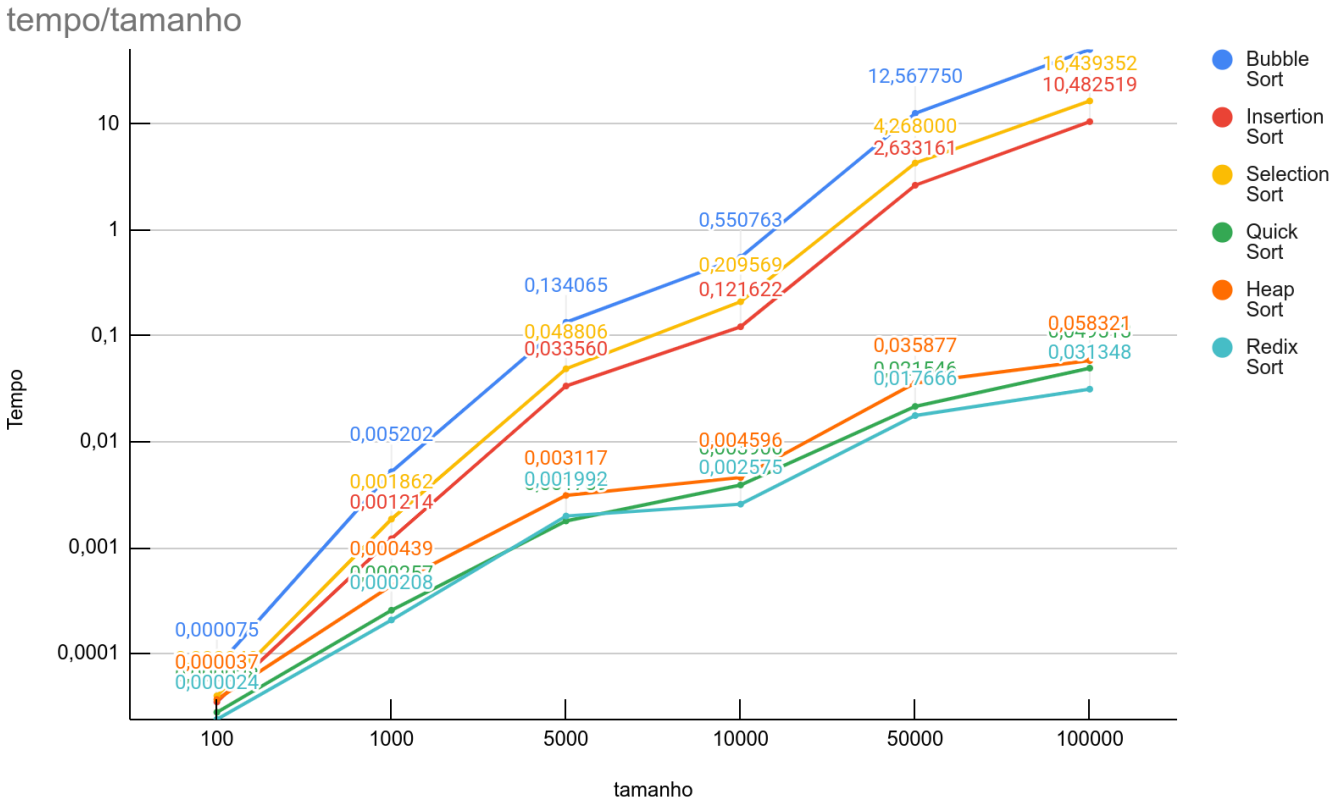
O trabalho poderá ser feito em duplas ou individual e será apresentado individualmente.

Programa:

https://github.com/Ruan2PVS9/uffs_cc_repository/blob/main/POD/Trabalho_1/trabalho_p1_ruan.c

Coleta de dados:
Com a coleta dos dados das execuções foi pego a seguintes médias:

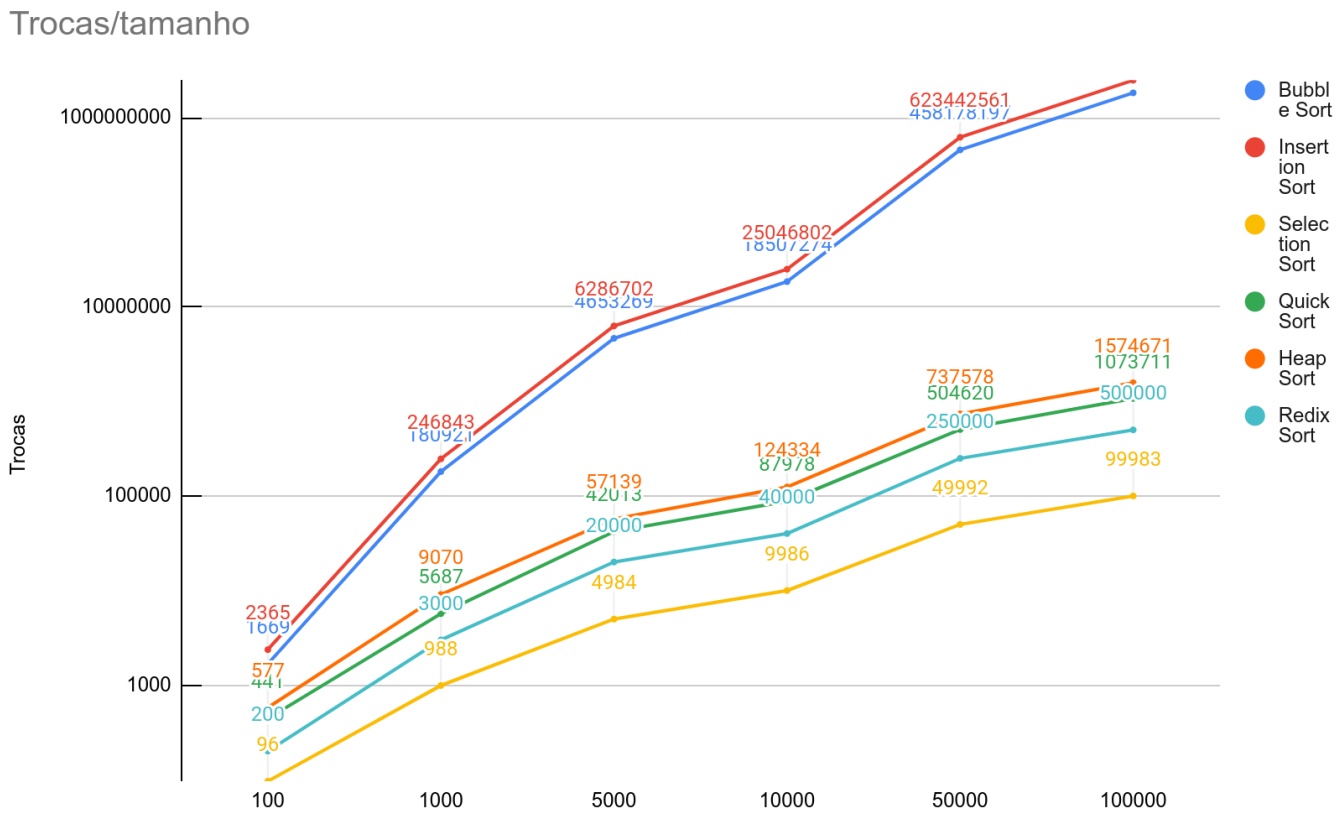
tempo/tamanho						
	Bubble Sort	Insertion Sort	Selection Sort	Quick Sort	Heap Sort	Redix Sort
100	0,000075	0,000035	0,000040	0,000028	0,000037	0,000024
1000	0,005202	0,001214	0,001862	0,000257	0,000439	0,000208
5000	0,134065	0,03356	0,048806	0,001789	0,003117	0,001992
10000	0,550763	0,121622	0,209569	0,003906	0,004596	0,002575
50000	12,567750	2,633161	4,268000	0,021546	0,035877	0,017666
100000	50,276997	10,482519	16,439352	0,049513	0,058321	0,031348



Na consideração de tempo por tamanho da entrada de elementos se percebe o que os novos métodos HeapSort e RadixSort são tão efetivos quanto o quickSort, sendo esse ficando entre eles, se saído ligeiramente melhor que o Heap, é ligeiramente pior que o Radix.

ja na quantidade de trocas:

Trocas/tamanho						
	Bubble Sort	Insertion Sort	Selection Sort	Quick Sort	Heap Sort	Redix Sort
100	1669	2365	96	441	577	200
1000	180921	246843	988	5687	9070	3000
5000	4653269	6286702	4984	42013	57139	20000
10000	18507274	25046802	9986	87978	124334	40000
50000	458178197	623442561	49992	504620	737578	250000
100000	1839029164	2501720796	99983	1073711	1574671	500000



Se repete o fato do QuickSort ficar entre o heapSort e o Radix Sort, mas ainda sim o que possui menos trocas se mantém o método SelectionSort.