1o trabalho de Estrutura de Dados II
Ordenação Parcial em Memória Principal

Este trabalho trata de ***Ordenação Parcial em Memória Principal*** e deverá ser realizado em dupla. A avaliação ocorrerá em três etapas sequenciais:

1) Entrega do código fonte conforme a especificação, compilando e gerando o resultado esperado para os casos de teste fornecidos;
    a) O código deverá aplicar boas práticas de programação, como comentários, modularização e reuso de código. Por exemplo, o mesmo código poderia ser facilmente modificado para ordenar outros tipos de dados, ou outros métodos poderiam ser facilmente acrescentados.
2) Relatório comparando e explicando os resultados obtidos; deverá apresentar gráficos avaliando o aumento da complexidade de cada algoritmo para entradas de um mesmo tipo (aleatório, quase ordenado, invertido).
3) Entrevista com os componentes do grupo sobre o trabalho executado.

A falha em uma etapa implica que as seguintes são automaticamente zeradas. Falha em responder as questões da entrevista, demonstrando que o entrevistado não compreende o próprio trabalho, implica em perda de pontos para a dupla. Portanto, se o seu companheiro do trabalho não estiver interessado na compreensão do mesmo, comunique ao professor para entregar o trabalho individualmente.

Atenção: plágio não será tolerado! Os grupos envolvidos receberão nota zero. Se algum colega pedir ajuda, não forneça o seu código como exemplo, mas trabalhem juntos sobre o código dele/dela.

Bônus: a (correta) inclusão/análise de outros métodos além dos exigidos permitirá à dupla recuperar pontos (eventualmente) perdidos na avaliação.

Entrega: o aluno deverá realizar via Google Classroom o upload de dois arquivos: (i) um arquivo .zip contendo os arquivos fontes e makefile, e (ii) um arquivo .pdf para o relatório.

Especificação:

Desenvolva um programa em C capaz de ler arquivos de números inteiros e ordená-los de forma a obter os T maiores elementos usando algum dos seguintes algoritmos: ordenação por seleção, ordenação por inserção, shellsort, quicksort e heapsort. Escreva um relatório comparando o desempenho teórico e prático de cada um dos algoritmos para diferentes tamanhos e tipos de entrada (aleatório, quase ordenado, invertido). A comparação prática deve envolver o número de comparações, trocas e tempo de CPU.

O programa deverá considerar os seguintes parâmetros:

(a) Executa todos os métodos de ordenação;
(s) Executa método de ordenação por seleção
(i) Executa método de ordenação por inserção
(e) Executa método de ordenação por shellsort

(q) Executa método de ordenação por quicksort
(h) Executa método de ordenação por heapsort
(1) imprime em tela os T maiores elementos:
Exemplo: Top 5 para entrada [100,300,400,200,500,50,1,10]
500
400
300
200
100

(2) imprime estatísticas:
Tempo de CPU: 0.0001 segundos.
Comparações: 300
Trocas: 200

(3) imprime dados/estatísticas separados por tab:

[algoritmo	**arquivo**	**tam.**	**T(top)**	**comp.**	**trocas**	**tempo(s)]**
seleção	./in/5.txt	5	5	x	y	0.00001

Execução: ./a.out [-123iseqha] top_T caminho_arquivo

Arquivos de entrada:
A primeira linha contém a quantidade de items. Cada linha seguinte contém um item ordenável.

Casos de Teste:
(1) ./a.out -1s 5 ./in/5.txt
800000
700000
200000
150000
150000

(2) ./a.out -1s 5 ./in/20.txt
484802
420713
396295
392019
383972

(3) ./a.out -1s 5 ./in/100.txt
699707
698297
688079
688043
681744