Universidade Federal do Maranhão
Departamento de Informática
Disciplina: ***Estrutura de Dados II 2024.1***
Prof.: ***João Dallyson Sousa de Almeida***

Atividade Prática 1 (DUPLA)
Valor: 40% da 1ª Avaliação

Problema 1:

Implemente uma versão do mergesort. Adicione um corte para vetores pequenos, teste se o vetor já está em ordem e evite a cópia trocando argumentos no código recursivo.

- Use o InsertSort para subvetores pequenos. No caso da ordenação, sabemos que a ordenação por inserção é simples e, portanto, provavelmente mais rápida do que a ordenação com o MergeSort para sub-vetores pequenos. Mudar para a ordenação com o InsertSort para sub-vetores pequenos (comprimento 15 ou menos, por exemplo) melhorará o tempo de execução de uma implementação típica de mergesort em 10 a 15%.
- Teste se o vetor já está ordenador. É possível reduzir o tempo de execução para ser linear para vetores que já estão ordenados, adicionando um teste para ignorar a chamada para *merge()* se *vetor[meio]* for menor ou igual a *vetor[meio+1].* Com essa alteração, ainda fazemos todas as chamadas recursivas, mas o tempo de execução para qualquer sub-vetor ordenado é linear

Problema 2:

Implemente uma versão modificada do algoritmo SelectSort. O passo de seleção do menor ou maior item deve ser modificado para selecionar o maior e menor na mesma iteração (verifique algoritmo minmax3 na nota de aula 2, apresentado em aula). Seguindo essa ideia, o algoritmo já organizará o menor e o maior elemento a cada iteração.

Problema 3:

Quicksort + BubbleSort: você deverá implementar a variação "mediana de três", em que o pivô é escolhido usando a mediana entre a chave mais à esquerda, a chave mais à direita, e a chave central. O Quicksort deve ser executado recursivamente (como padrão) e, quando você obtiver uma partição de tamanho menor ou igual a L, essa partição deverá ser imediatamente ordenada usando o algoritmo do Bubble Sort. Determine empiricamente o valor de L para o qual o quicksort é executado mais rapidamente em seu ambiente de computação para ordenar vetores aleatórios de N, para N = 10^2,10^3, 10^4, 10^5 e 10^6

Problema 4:

Implemente um programa em java que implemente uma nova versão do HeapSort, denominada DoubleHeapSort. Nessa versão, você deverá construir dois Heaps, um mínimo e outro máximo, para coordenar a inserção dos elementos alternados preenchendo um vetor auxiliar ordenado, alternando a inserção no lado esquerdo e direito. Verifique, também, a possibilidade de utilizar Threads e avalie a diferença de desempenho.

Experimentos:

1) Todos o métodos deverão implementar os objetos utilizando *Generics (https://docs.oracle.com/javase/tutorial/java/generics/types.html).*
2) Você deverá enviar, também, um relatório mostrando e discutindo os resultados obtidos. Mostre prints da execução e dos resultados.

3. Entrega

- Código fonte do programa em JAVA (bem identado e comentado) utilizando os conceitos de Orientação a Objetos. Relatório com os resultados (Upload no SIGAA)