Universidade Federal do Maranhão
Departamento de Informática
Disciplina: ***Estrutura de Dados II 2023.1***
Prof.: ***João Dallyson Sousa de Almeida***

Atividade Prática 1 (Dupla)
Valor: 50% da 1ª Avaliação

1. DESCRIÇÃO

O objetivo deste trabalho prático é analisar, comparar e desenvolver algoritmos de ordenação para lidar com grandes volumes de dados em um sistema de gerenciamento de estoque de uma empresa. Você deve implementar diferentes algoritmos de ordenação e avaliar o seu desempenho em lidar com grandes volumes de dados, a fim de melhorar a eficiência do sistema e reduzir o tempo de processamento.
Segue lista atividades:

1) Adquirir base de dados: Coleta de dados: Coletar dados de estoque de uma empresa, incluindo informações como códigos de produtos, quantidade em estoque, datas de entrada e saída, entre outros dados relevantes. Exemplos de repositórios: Kaggle, Data.gov e Data World. No Kaggle (https://www.kaggle.com/datasets) você pode buscar por datasets de gerenciamento de estoque usando os seguintes termos: “"stock management" ou "inventory management".
2) Pré-processamento dos dados: Realizar pré-processamento dos dados coletados, como remoção de valores ausentes, normalização, etc.
3) Implemente o algoritmos **SelectSort, InsertSort QuicSort, MergeSort e HeapSort** padrão apresentados em aula.
4) Implemente um algoritmo denominado QM-Sort, combinação do QuickSort + MergeSorte. Sabendo que o Quicksort é muito rápido em média, mas pode ser muito lento no pior caso. Por outro lado, o Mergesort é sempre rápido, mas pode usar mais memória. Uma combinação interessante é usar o Quicksort até um tamanho “T” do subvetor, e depois usar o Mergesort para juntar os subvetores ordenados. Isso aproveita a rapidez do Quicksort na maioria dos casos, mas evita o pior caso.
5) Implemente um algoritmo denominado de SI-Sort, combinação do Selection Sort + Insertion Sort. Para usar esse algoritmo, o usuário deve um %X do tamanho do vetor. Esse percentual é usando para delimitar o percentual de elementos nas extremidades, inicio e fim, que devem ser ordenados, inicialmente, usando o SelectSort visto em sala. Por último, ordene os demais elementos usando o InsertSort.
6) Você deve usar os Padrões de Projeto *Template Method e Strategy*. O padrão Template Method define a estrutura de um algoritmo e permite que as subclasses definam os detalhes específicos de como cada etapa é realizada, tornando o código mais modular e permitindo que diferentes algoritmos sejam criados a partir de uma única classe base. Já o padrão Stratety permite que diferentes algoritmos de ordenação sejam selecionados em tempo de execução. Ele define uma família de algoritmos, encapsula cada um deles e torna-os intercambiáveis. Isso significa que é possível alterar o método de ordenação sem alterar a classe que o utiliza.

Experimentos:

1) Todos o métodos de ordenação deverão ordenar objetos utilizando *Generics* *(https://docs.oracle.com/javase/tutorial/java/generics/types.html).*
2) O usuário dever ter a opção de ordenar em ordem crescente ou decrescente em todos os algoritmos.
3) Você deve realizar teste usando diferentes tipos de dados como critério de ordenação texto e numérico.
4) Compare a execução dos algoritmos implementados com os métodos de ordenação disponíveis no JDK do Java nas apis: java.util.Arrays e java.util.Collection
5) Você deverá enviar, também, um relatório mostrando e discutindo os resultados obtidos. Mostre prints da execução e dos resultados.

2. Análise dos resultados

A análise deve ser feita sobre o número de comparações, atribuições e tempo de execução dos algoritmos.

3. Entrega

- Código fonte do programa em JAVA (bem identado e comentado) utilizando os conceitos de Orientação a Objetos.
- Relatório com os resultados (Gráficos comparando os métodos de ordenação e breve discussão dos resultados).
- Upload no SIGAA.

O Relatório deve apresentar:

1. Testes: apresentação dos testes realizados.
2. Referências: referências utilizadas no desenvolvimento do trabalho.