ATIVIDADE DE PARTICIPAÇÃO 01 - RELATÓRIO TÉCNICO

Grupo:

Delphino Luciani de Paula Araujo Filho - 20219019597

Samia Ribeiro Gonçalves - 20219022082

**1.1. Introdução:**

O presente relatório visa detalhar a Atividade de Participação 05 do grupo, que compõe a segunda nota da disciplina de Estrutura de Dados do Departamento de Computação - CCN da Universidade Federal do Piauí, período 2022.2, ministrada pelo professor Dr. Raimundo Moura.

Escrever uma aplicação para criar uma lista encadeada que armazenar nomes de pessoas, tendo as seguintes opções: inserir um nome no início da lista, inserir um nome no final da lista, buscar um determinado nome na lista, excluir um determinado nome, se ele estiver na lista, por fim deve ter a opção de mostrar todos os nomes da lista.

**O vídeo com a execução da aplicação encontra-se no link:**

[**https://youtu.be/jBxLHu3twOM**](https://youtu.be/jBxLHu3twOM)

**1.2. Preparação do grupo**

Todos os integrantes do grupo se reuniram em uma chamada online onde todos contribuíram para a criação do trabalho, dividindo em parte de código e simulação/ elaboração do relatório, criação de gráficos e estudo dos resultados

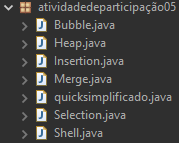
**1.3. - Material utilizado**

Para gerar os códigos com máxima eficiência e agilidade foi utilizado uma IA chamada ChatGpt além de utilizar o aúxilio do livro para dúvidas e segurança para realização, contando com uma leve adaptação manual para adicionar o tempo de execução no programa, para criação dos gráficos e tabelas foi utilizado o programa Excel e o Eclipse foi utilizado como principal plataforma de desenvolvimento e simulação em um Desktop com processador Ryzen 5 3600(6 núcleos 12 threads 3.6 Ghz)

**1.4. Estrutura do Programa**

Para realização da atividade foram geradas classes com os códigos de cada algoritmo de ordenação

**Classes:**



Teresina 01 de Fevereiro de 2023

**2.1 Analise dos Resultados Gráficos(10k)**

Para a simulação das ordenações dos Algoritmos com o arquivo de 10 mil palavras foram feitas 10 simulações individuais para cada algoritmo, destacando o tempo de execução médio, maior e menor obtidos na execução.

OBS: tempos de execução em milisegundos

**-Insertion Sort:**

**-Selection Sort:**

**-Bubble Sort:**

**-Shell Sort:**

**-Heap Sort:**

**-Merge Sort:**

**-Quick Sort:**

**2.2 Analise dos Resultados Gráficos(100k)**

Para a simulação das ordenações dos Algoritmos com o arquivo de 100 mil palavras foram feitas 10 simulações individuais para cada algoritmo, destacando o tempo de execução médio, maior e menor obtidos na execução.

OBS: tempos de execução em milisegundos

**-Insertion Sort:**

**-Selection Sort:**

**-Bubble Sort:**

**-Shell Sort:**

**-Heap Sort:**

**-Merge Sort:**

**-Quick Sort:**

**2.3 Analise dos Resultados Gráficos(**500k**)**

OBS: tempos de execução em milisegundos

Para a simulação das ordenações dos Algoritmos com o arquivo de 500 mil palavras foram feitas 3 simulações individuais para cada algoritmo (redução necessária considerando o alto tempo de execução obtido) destacando o tempo de execução médio, maior e menor obtidos na execução.

**-Insertion Sort:**

**-Selection Sort:**

**-Bubble Sort:**

**-Shell Sort:**

**-Heap Sort:**

**-Merge Sort:**

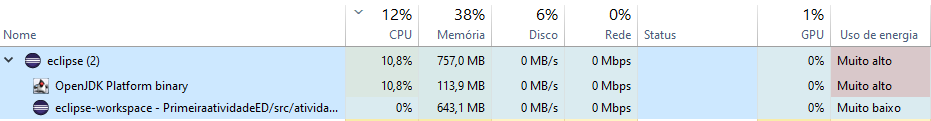
**-Quick Sort:**

**3.1 Comparação de eficiência (Insertion x Selection x Bubble)**

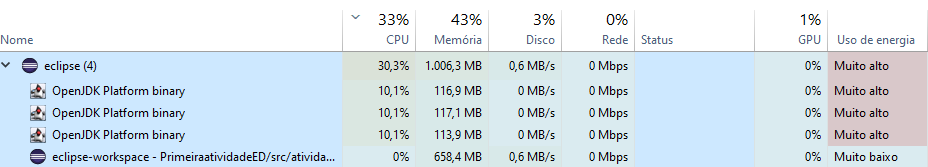
**3.2 Comparação de eficiência (Shell x Heap x Merge x Quick)**

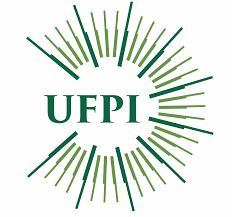
**4.1 Conclusão**

Contudo apresentado no relatório é possível inferir que o algoritmo de Insetion Sort foi o mais rápido dentre sua classe em ambos os tamanhos de arquivos, já o Quick Sort foi o ganhador geral nos arquivos de 10k e 100k porem perdeu por pouco para o merge sort no arquivo de 500 mil palavras.

Porem fica uma dúvida a respeito da execução pois mesmo com uma larga margem de processamento a linguagem java não utiliza 100% do processamento do computador por conta de sua proteção Multi thread e possivelmente só utilizar um único núcleo de processamento 

Uma possível solução para a lentidão encontrada quando se roda unicamente os algoritmos insertion, selection e bubble é rodar os 3 ao mesmo tempo, visto que aparentemente é a única forma de escalabilizar a utilização do processamento, ocasionando na piora dos resultados, visto que no menor tempo encontrado em todas foi com apenas uma das execuções atuando



Outra possível solução seria mudar a linguagem para C ou C++ visto que não há a proteção multi thread e a execução é liberada a utilizar a completa potencia de processamento da máquina.