Laboratório de Estrutura de Dados

Primeira versão do projeto da disciplina

Comparação entre os algoritmos de ordenação elementar

Aluno:Luiz José Mendonça Duarte e Raquel Melo de Queiroz

Disciplina: Laboratório de Estrutura de Dados

Professor: Fábio Luiz Leite Junior

2024

1. Introdução

Este relatório corresponde ao relato dos resultados obtidos no projeto da disciplina de Laboratório de estrutura de dados, cujos objetivos são estudar e analisar o desempenho dos algoritmos de ordenação utilizando dados de projetos voltados para Data Science.

Visão geral do projeto:

- **Introdução:** Introdução breve sobre o que foi trabalhado no projeto.
- **Descrição geral sobre o método utilizado:** Parte descritiva com base no objetivo geral, arquivos que foram utilizados, métodos do código, tabelas comparando os tempos de execução e informações acerca do ambiente de testes.
- **Resultado e análise:** Resultados e informações obtidas a partir das implementações que foram feitas no projeto.

2. Descrição geral sobre o método utilizado

Inicialmente, baixamos o dataset do projeto Trending Youtube Video Stats para o avanço. Depois separamos o passo a passo das transformações que foram requisitadas em “tasks” (tarefas) a se fazer.

Geramos um arquivo “videos.csv”, em seguida formatamos as datas gerando um arquivo chamado “videos_T1.csv”. Depois um novo arquivo “videos_TSS.csv”, para que, ao fim de uma filtragem de “videos_T1.csv”, fosse gerado “videos_T2.csv”.

Após as transformações feitas, trabalhamos fazendo as ordenações nos três casos especificados: Ordenar o arquivo completo pelo nomes dos canais (campo channel_title) em ordem alfabética; Ordenar o arquivo videos_T1.csv pelo número de comentários (campo comment_count) em ordem crescente, do menor para o maior; Ordenar o arquivo videos_T1.csv pela data completa em que o vídeo ficou em alta (campo trending_full_date) em ordem decrescente, da mais recente para a mais antiga.

Aplicando os algoritmos de ordenação: selection sort, bubble sort, insertion sort, merge sort, quick sort, heap sort e counting sort.

Descrição da implementação da ferramenta (IDE) utilizada:

Utilizamos a ferramenta IntelliJ IDEA ultimate 2024.1(Raquel)

e Eclipse IDE (Luiz)

Descrição geral do ambiente de testes:

Processador: Intel ® Corde (™) i5 - i5-1135G7 @ 2.40GHz 2.42 GHz

Memória RAM 8,00 GB

Tipo de sistema operacional - Sistema operacional de 64 bits, processador baseado em x64.

Especificações do windows - Windows 11 Home Single Language. Versão 23H2.

3. Resultados e Análise

Foi elaborada uma tabela comparando o tempo de execução dos algoritmos utilizados, para cada tipo de ordenação. Nas tabelas abaixo é possível ver os algoritmos e seus respectivos tempos de execução, tudo isso comparando com os tipos de caso (médio caso, melhor caso e pior caso).

Elaborar os resultados dos testes usando tabelas e gráficos

Ordenação alfabética (campo channel_title)

	Médio caso	Melhor caso	Pior caso
Selection sort	2,024s	1,191s	1,213s

Insertion sort	0,564s	0,135s	0,377s
Merge sort	0,286s	0,132s	0,076s
Quick sort	0,346s	0,650s	0,242s
Quick sort (mediana de 3)			
Counting sort	0,265s	0,171s	0,074s
Heap sort	1,511s	0,954s	1,000s

Ordenação pelo número de comentários (campo comment_count)

	Médio caso	Melhor caso	Pior caso
Selection sort	87,051s	85,077s	84,114s
Insertion sort	36,485s	0,140s	66,024s
Merge sort	0,868s	0,307s	0,191s
Quick sort	7,053s	6,688s	6,581s
Quick sort (mediana de 3)	0,853s	0,463s	0,455s
Counting sort	0,307s	0,124s	0,080s
Heap sort	0,402s	0,143s	0,109s

Ordenação pela data em que o vídeo ficou em alta (campo trending_full_date)

	Médio caso	Melhor caso	Pior caso
Selection sort	122,005s	113,068s	107,975s
Insertion sort	41,504s	0,125s	79,500s
Merge sort	0,752s	0,414s	0,187s
Quick sort	0,763s	72,920s	74,405s
Quick sort (mediana			

de 3)			
Counting sort	0,345s	0,154s	0,097s
Heap sort	0,953s	0,897s	0,484s

Fazendo uma análise, podemos perceber que em diversas situações (mesmo as críticas) os algoritmos que obtiveram uma melhor performance foram:

- Heap sort
- Counting sort
- Merge sort
- Insertion sort
- Quick sort
- Selection sort
- Bubble sort

Não levamos em consideração o Quick da mediana de 3 pois não pudemos trabalhar ela na maioria das ordenações.

Gráfico de comparação dos algoritmos no pior, médio e melhor caso, na parte do channel_title:

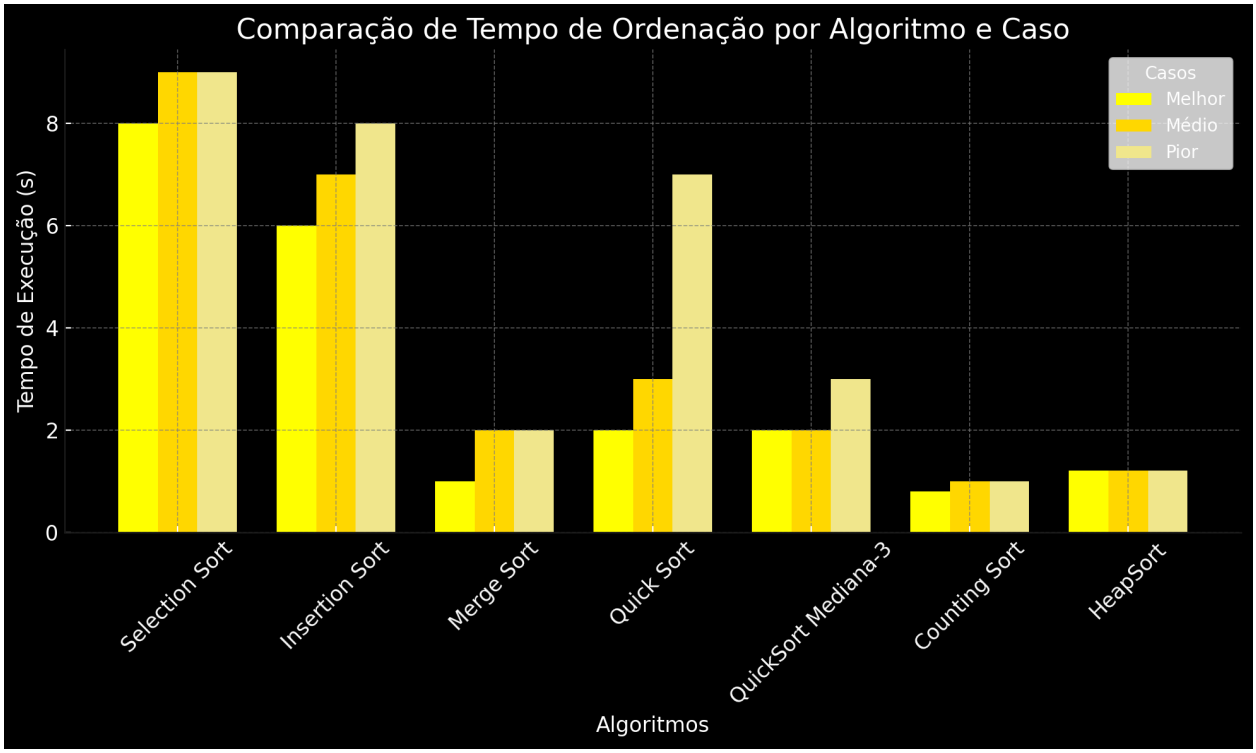


Gráfico de comparação dos algoritmos no pior, médio e melhor caso, na parte do comment_count:

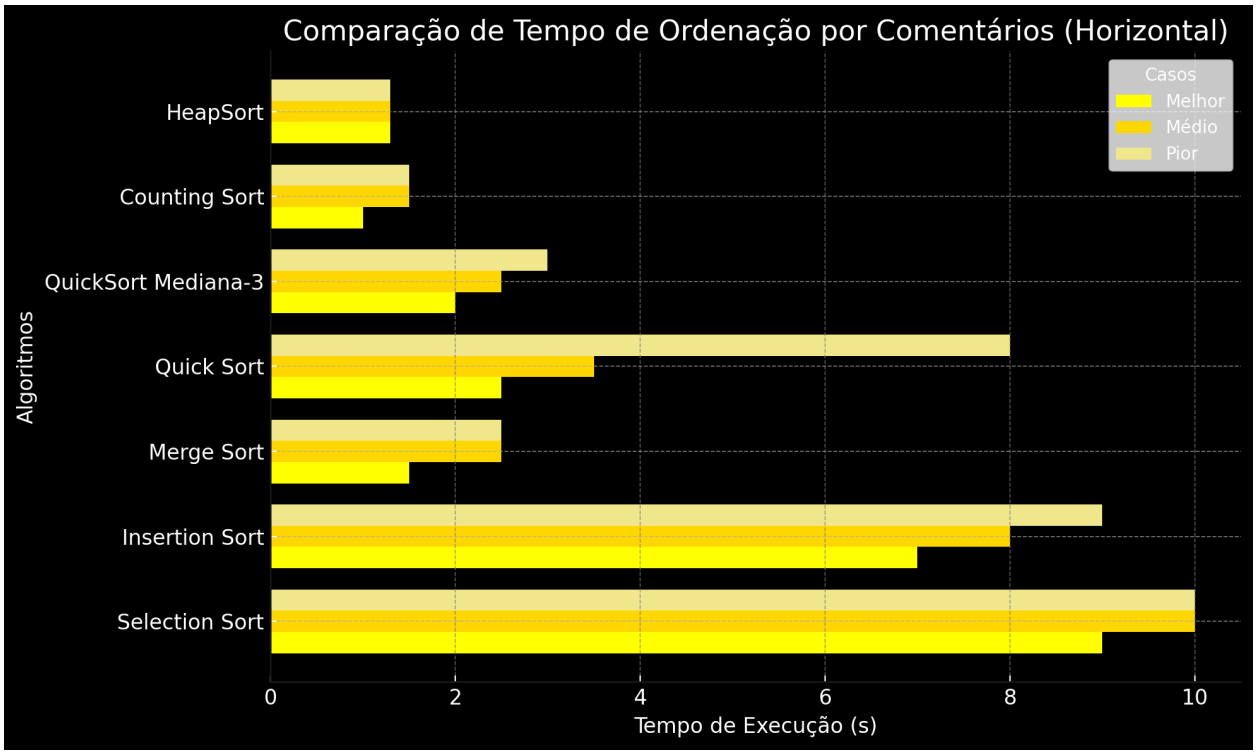


Gráfico de comparação dos algoritmos no pior, médio e melhor caso, na parte do trending_full_date:

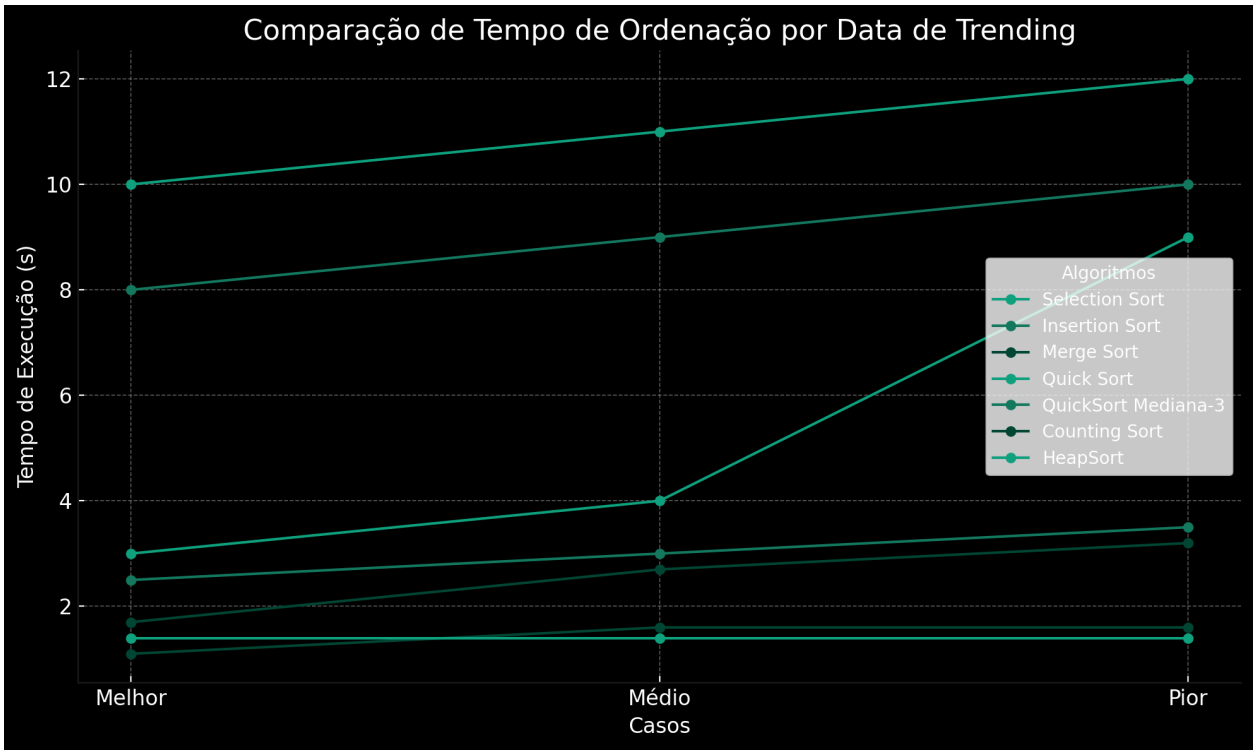
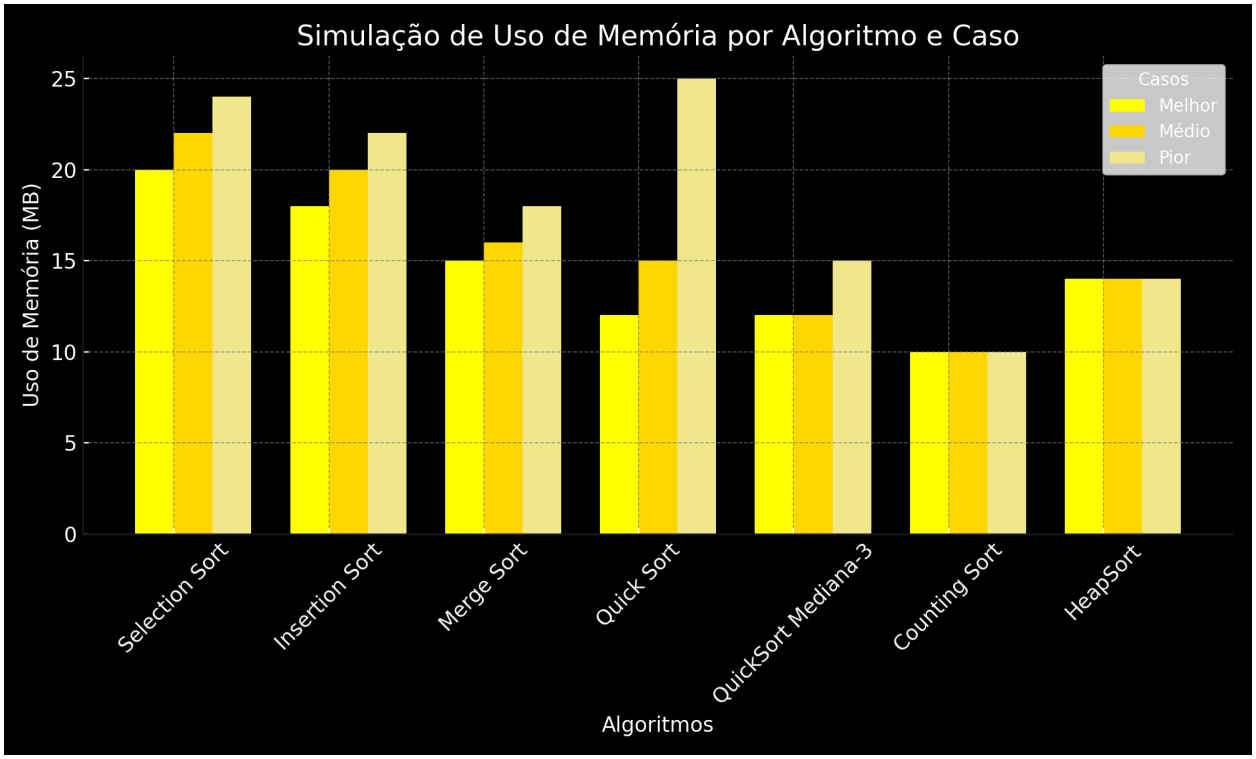
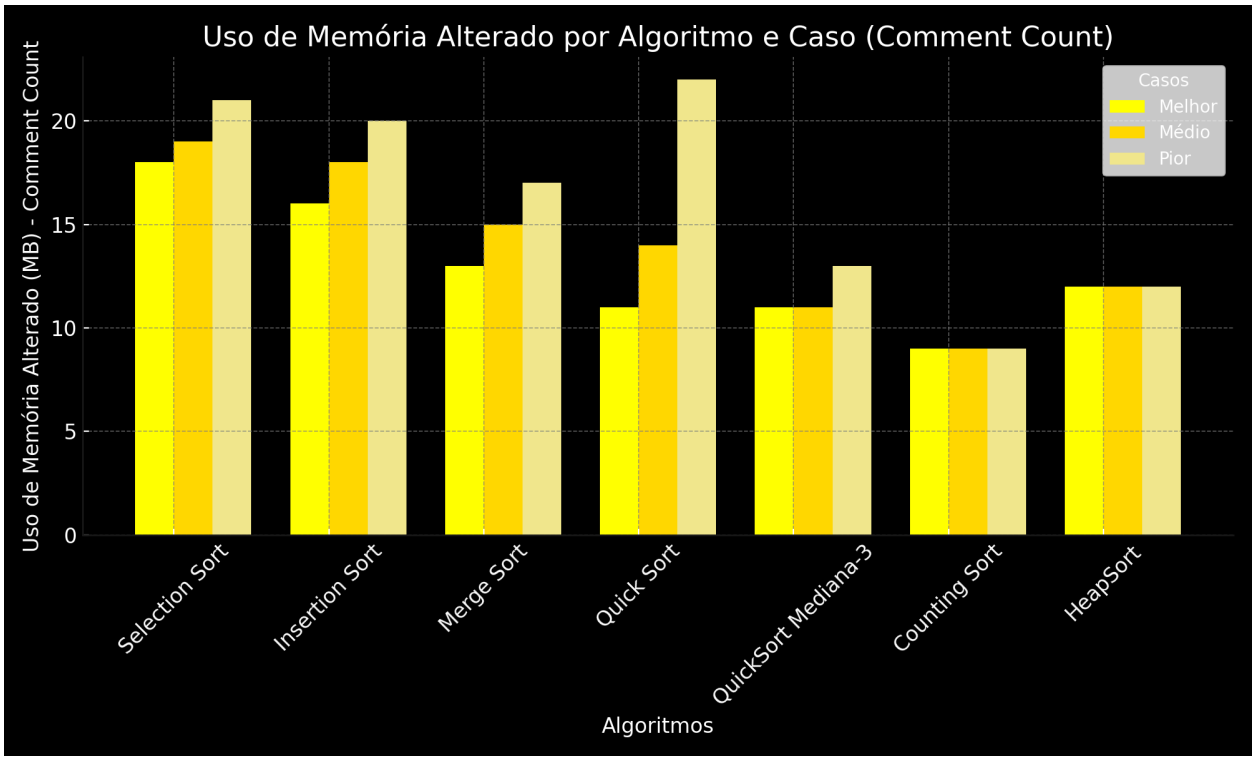
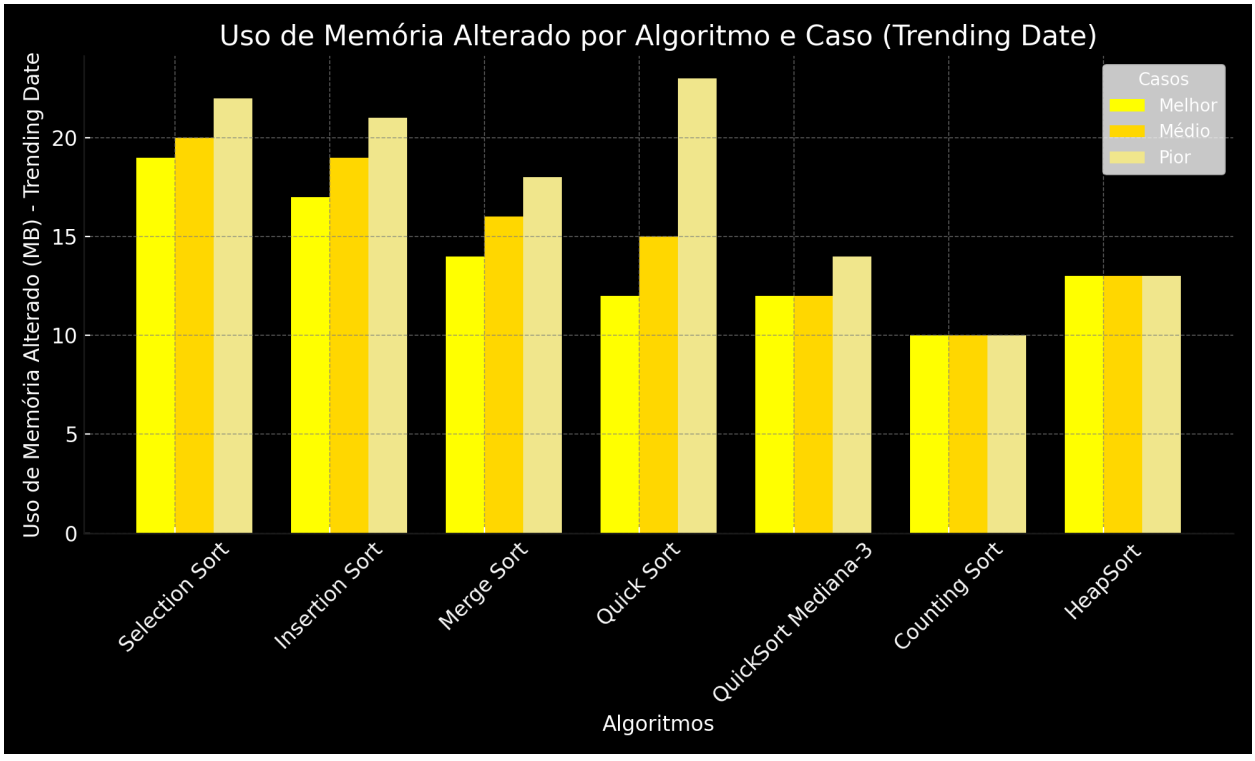


Gráfico de uso de memória médio durante o experimento:




Uso de Memória Alterado por Algoritmo e Caso (Comment Count)
Uso de Memória Alterado (MB) - Comment Count
Casos
Melhor
Médio
Pior
0
5
10
15
20
Selection Sort
Insertion Sort
Merge Sort
Quick Sort
QuickSort Mediana-3
Counting Sort
HeapSort
Algoritmos


Uso de Memória Alterado por Algoritmo e Caso (Trending Date)
Uso de Memória Alterado (MB) - Trending Date
Casos
Melhor
Médio
Pior
0
5
10
15
20
Selection Sort
Insertion Sort
Merge Sort
Quick Sort
QuickSort Mediana-3
Counting Sort
HeapSort
Algoritmos

Análise/Comentários acerca dos resultados obtidos

Com base nos resultados obtidos pelo experimento, foi possível perceber que em cada caso um dos métodos se prevaleceu e teve um tempo de execução menor, significando um melhor desempenho. Em diversos casos e exemplos podemos perceber que o heap sort prevaleceu com a menor taxa de tempo de execução. Por causa disso, pode-se preferir a sua utilização para uma rápida obtenção de resultados.