Laboratório de Estrutura de Dados

Primeira versão do projeto da disciplina

Comparação entre os algoritmos de ordenação elementar

Alunos:

Rafaela Candido Carneiro Fernandes, 192080393

Renan Rey Costa Rodrigues, 201080419

1. Introdução

Este relatório corresponde ao relato dos resultados obtidos no projeto da disciplina de EDA e LEDA que tem como objetivo fazer uma análise comparativa dos algoritmos de ordenação aplicado aos dados do covid-19. Os algoritmos de ordenação utilizados no projeto são: O Insertion Sort, Selection Sort, Merge Sort, Quick Sort, Quick Sort com Mediana de 3, counting, e Heap Sort.

Foi utilizado o site brasil.io para reorganizar os dados dos casos registrados de Covid-19 no Brasil.

Para resolver o projeto, deve-se ordenar um arquivo.csv disponibilizado pelo site brasil.io. Nesse arquivo contém o registro histórico de ocorrência de Covid-19 para todas as cidades e estados do Brasil. O programa tem como objetivo tratar apenas os dados mais atuais, os valores antigos não interessam. Ao final de cada processo de ordenação, o programa deve gerar um arquivo formatado.

Os resultados obtidos no final do projeto foram significativos, por meio de testes pode-se notar as diferenças de tempo de exercução e cpu de cada algoritmos. Alguns algoritmos por conseguir tratar uma quantidades maior de dados, conseguiu ser mais eficientes que outros em alguns testes.

2. Descrição geral sobre o método utilizado

Os testes do programa foram feitos utilizando as ferramentas do canva, para fazer os gráficos e o Lucidchart para mostrar o cunsumo de tempo e o VisualVM para mostrar a memória usada durante a execução do algoritmo.

Para obter os resultados foram realizados vários testes, para cada algoritmo, analisando o melhor caso, médio caso e pior caso de cada um, analisando o tempo de exercução para cada analise.

Os testes foram feitos no próprio programa e priorizam o tempo que cada algoritmo demora para cumprir seu objetivo, mas pode-se observar também o uso da CPU, que são úteis para medir e eficiência da performance de cada um dos algoritmos de ordenação.

Descrição geral do ambiente de testes

Os testes foram realizados em dois Notebook, cuja expecificações são:

PC 1 (Renan):

- Intel Core I7 7th Gen
- Windows 10
- 8GB Ram

PC 2 (Rafaela):

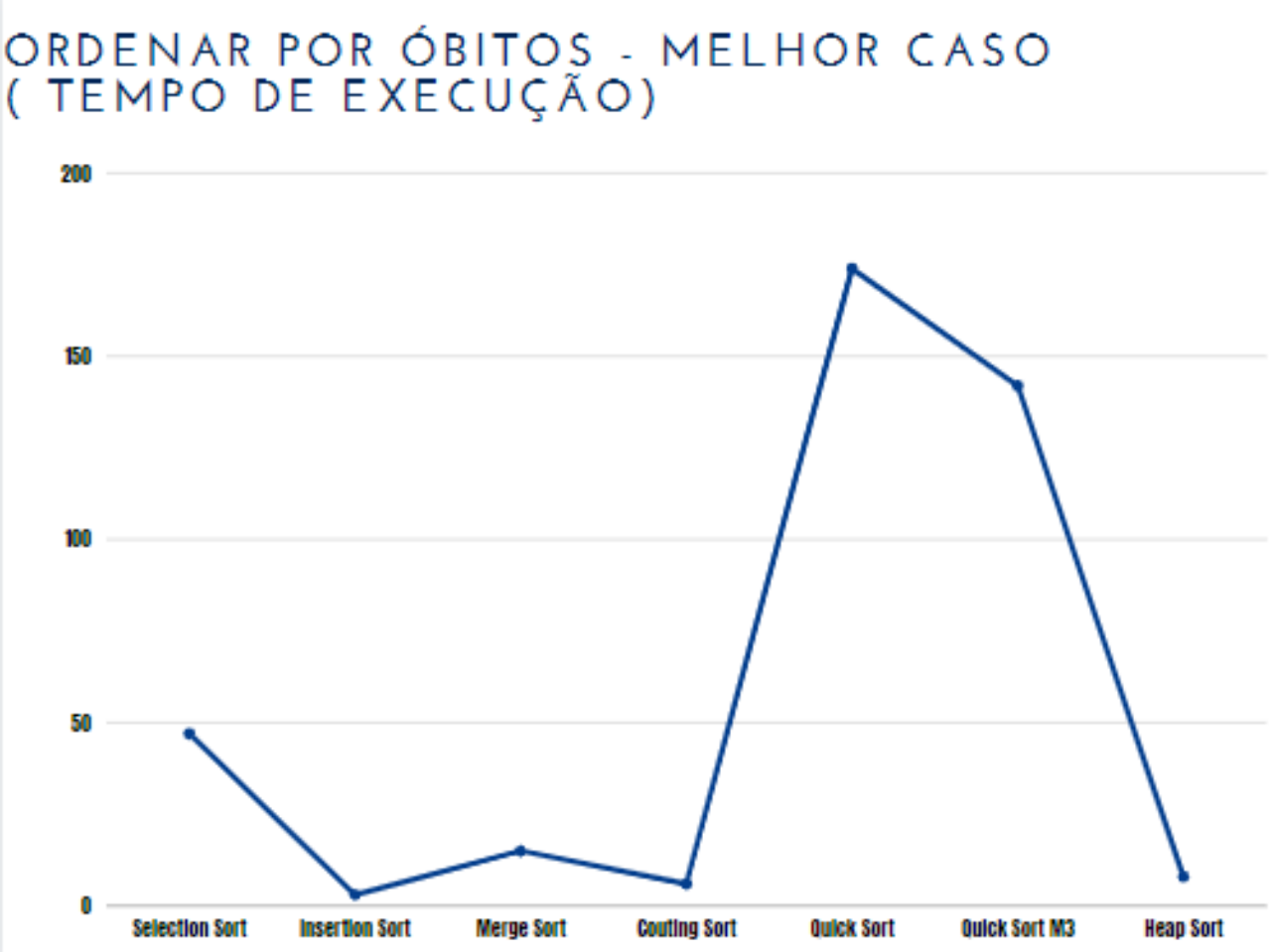
- Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
- Sistema operacional de 64 bits, processador baseado em x64
- Windows 10 Home Single Language
- 8,00 GB (utilizável: 7,86 GB)

3. Resultados e Análise

Resultados Utilizando gráficos mostrando o tempo de execução - Plataforma Canva

- Ordenação por óbitos:

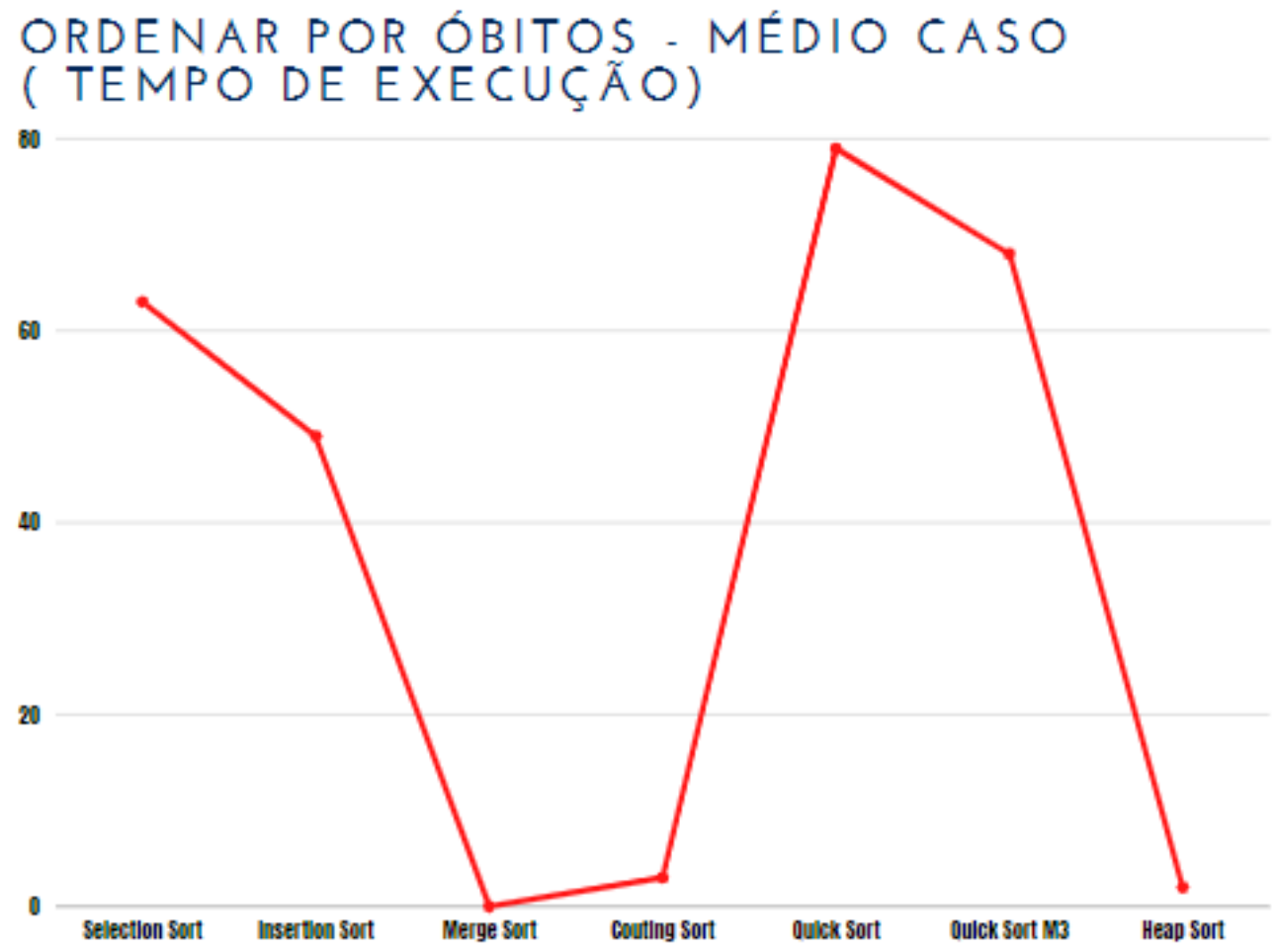
Melhor caso:



- Selection Sort : 47 ms
- Insertion Sort : 3 ms

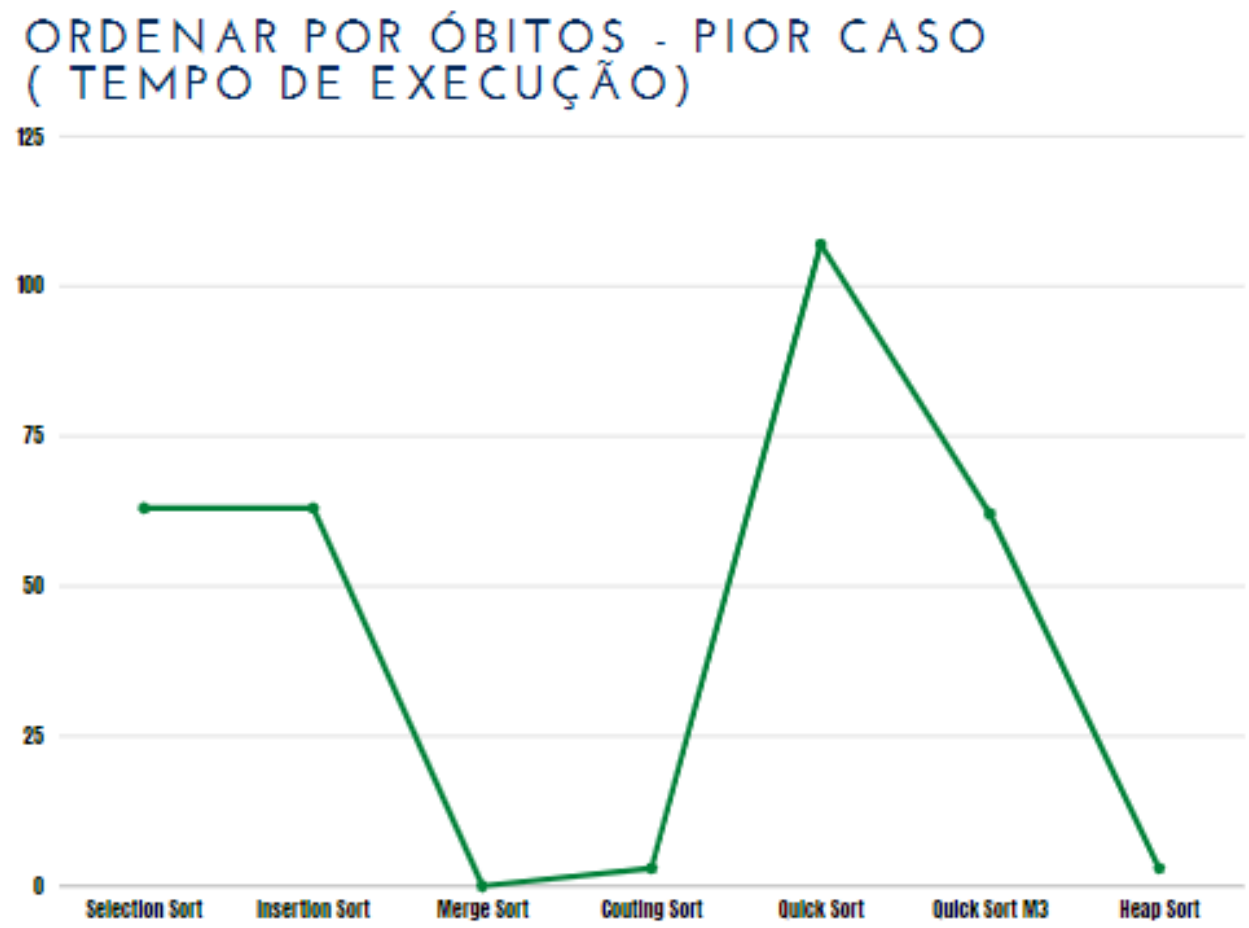
- Merge Sort : 15 ms
- Couting Sort : 6 ms
- Quick Sort : 174 ms
- Quick sort M3 : 142 ms
- Heap Sort: 8 ms

Médio Caso:



- Selection Sort : 63 ms
- Insertion Sort : 49 ms
- Merge Sort : -
- Couting Sort : 3 ms
- Quick Sort : 79 ms
- Quick sort M3 : 68 ms
- Heap Sort: 2 ms

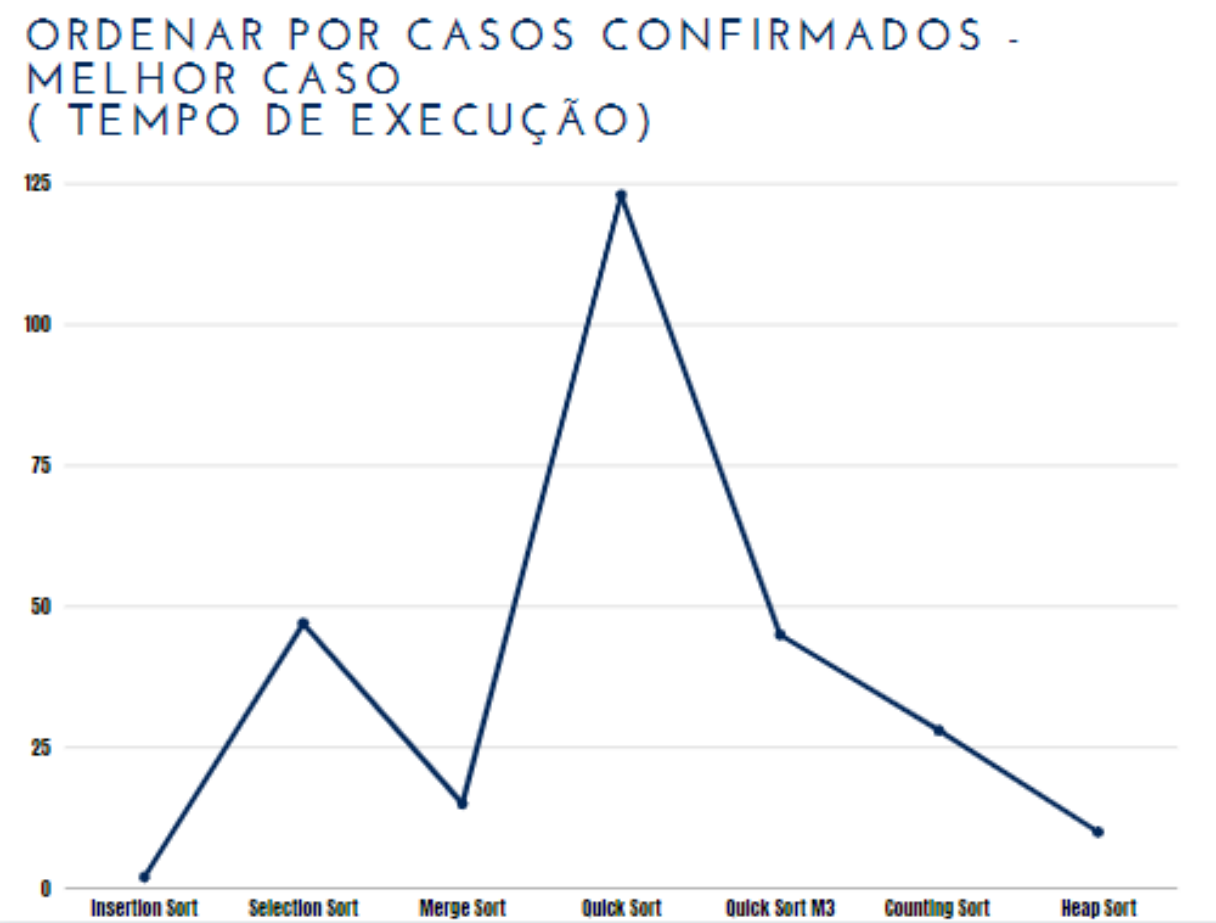
Pior Caso:



- Selection Sort : 63 ms
- Insertion Sort : 63 ms
- Merge Sort : -
- Couting Sort : 3 ms
- Quick Sort : 107 ms
- Quick sort M3 : 62 ms
- Heap Sort: 3 ms

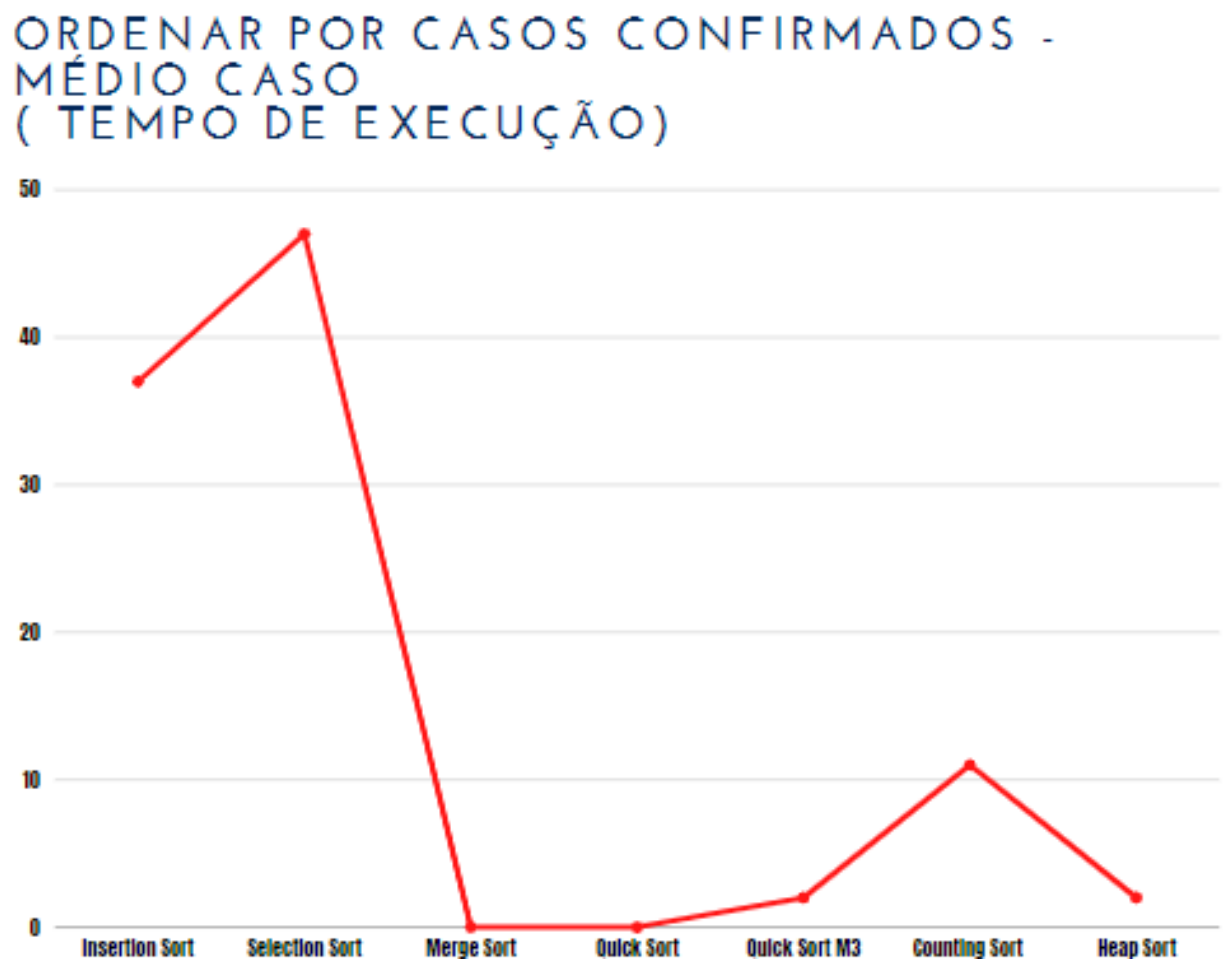
- Ordenação por casos confirmados

Melhor Caso:



- Selection Sort : 47 ms
- Insertion Sort : 2 ms
- Merge Sort : 15 ms
- Couting Sort : 28 ms
- Quick Sort : 123 ms
- Quick sort M3 : 45 ms
- Heap Sort: 10 ms

Médio Caso:



- Selection Sort : 47 ms
- Insertion Sort : 37 ms
- Merge Sort : -

- Couting Sort : 11 ms
- Quick Sort : -
- Quick sort M3 : 2 ms
- Heap Sort: 2 ms

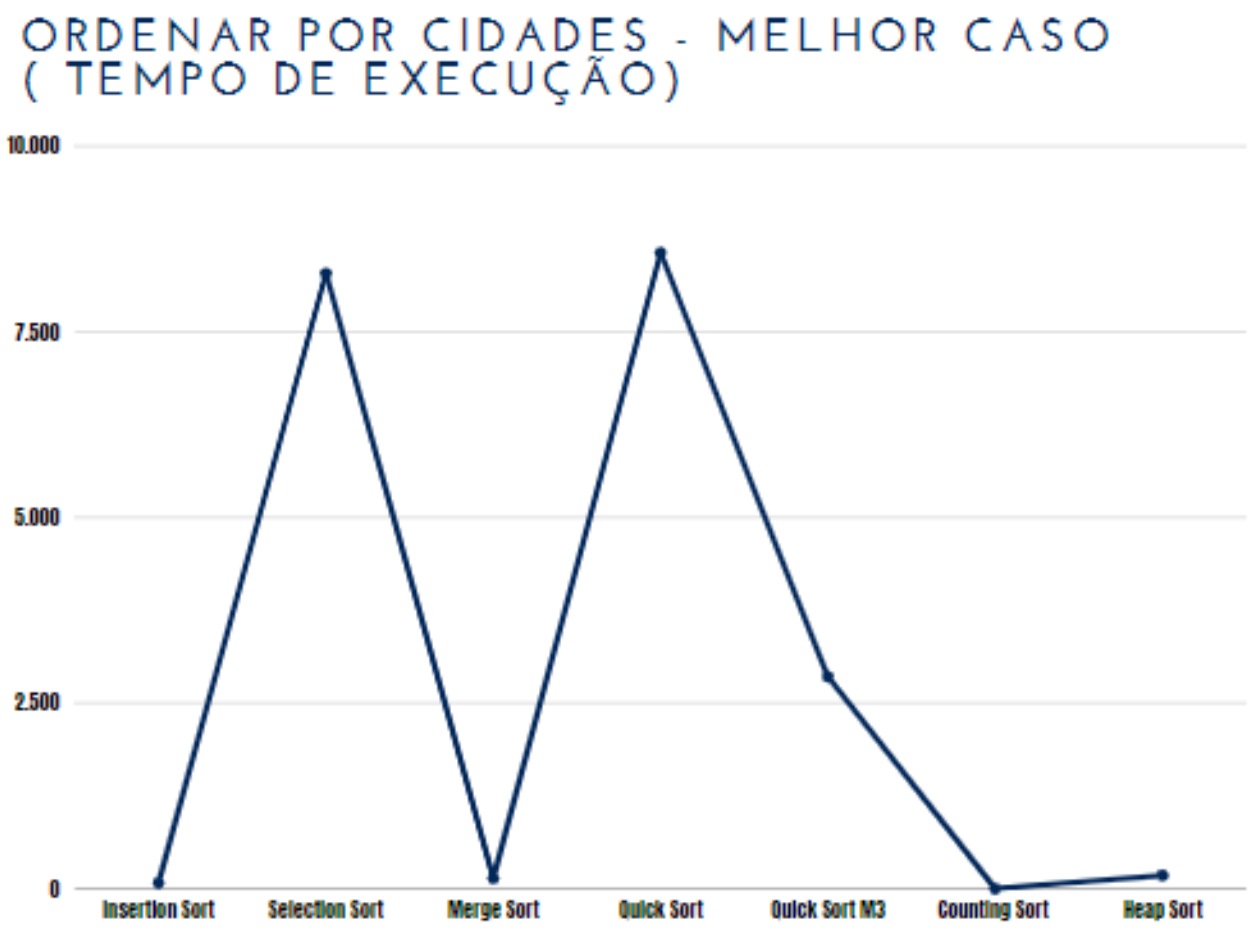
Pior Caso:

- Selection Sort : 47 ms
- Insertion Sort : 94 ms
- Merge Sort : -
- Couting Sort : 21 ms
- Quick Sort : 47 ms
- Quick sort M3 : -

- Heap Sort: 2 ms

- Ordenação por cidades

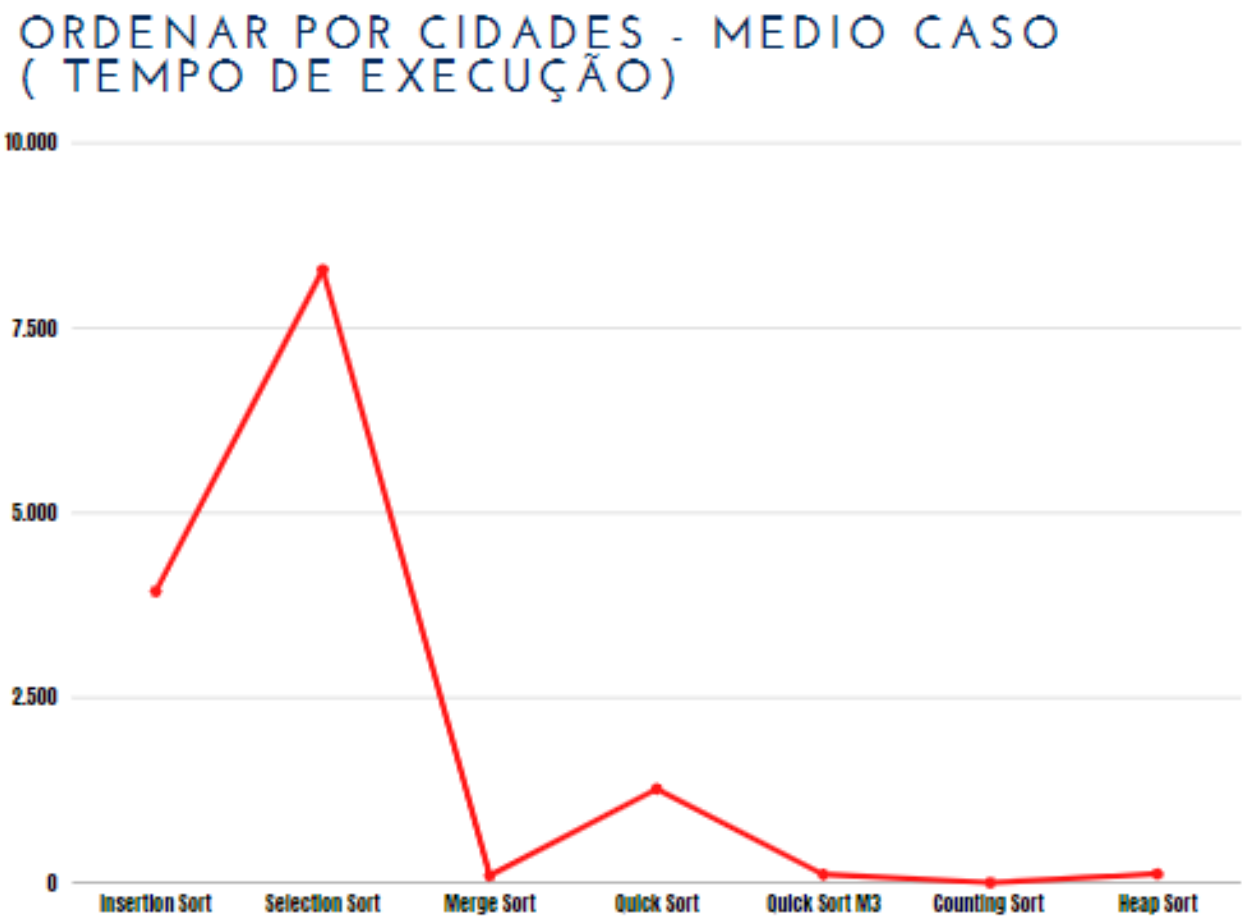
Melhor Caso:



- Selection Sort : 8284 ms
- Insertion Sort : 78 ms
- Merge Sort : 141 ms
- Couting Sort : -
- Quick Sort : 8566 ms

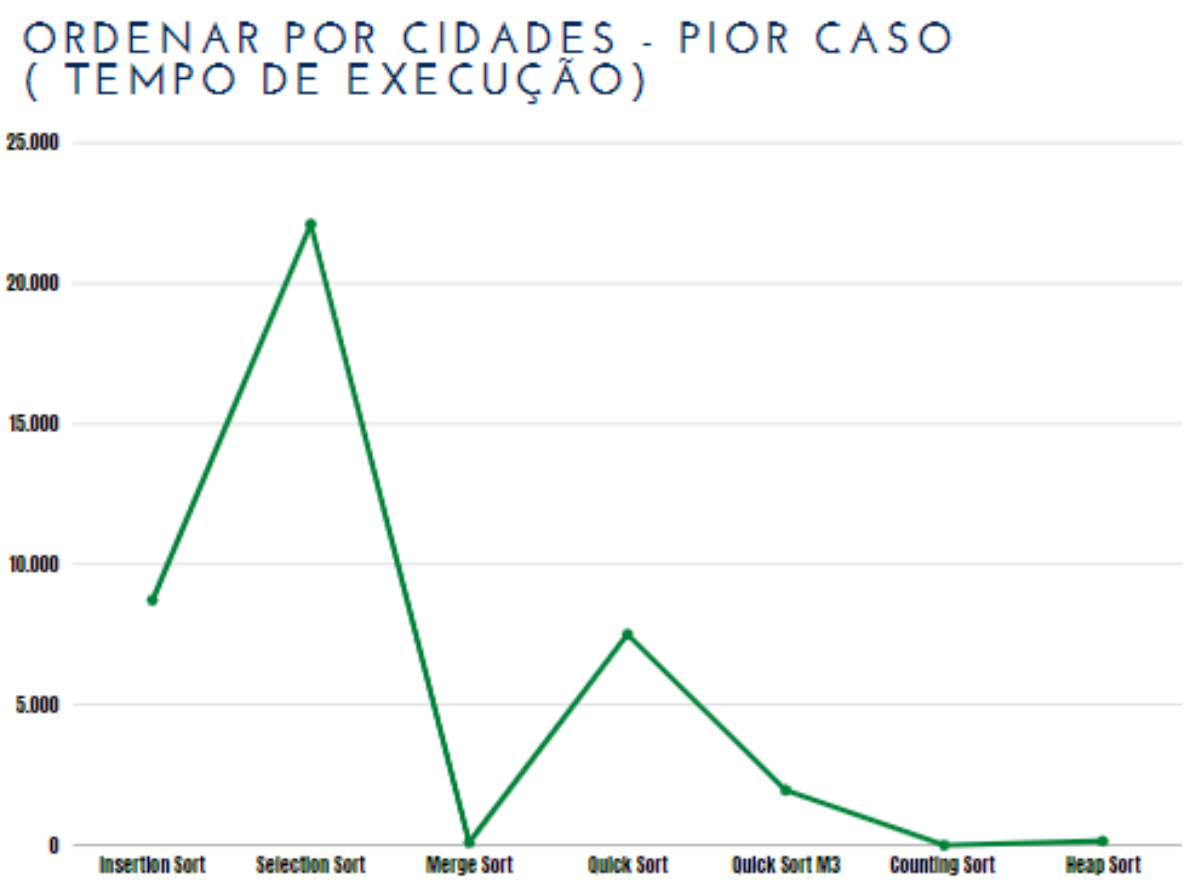
- Quick sort M3 : 2854 ms
- Heap Sort: 180 ms

Médio Caso:



- Selection Sort : 8285 ms
- Insertion Sort : 3940 ms
- Merge Sort : 93 ms
- Couting Sort : -
- Quick Sort : 1266 ms
- Quick sort M3 : 109 ms
- Heap Sort: 118 ms

Pior Caso:

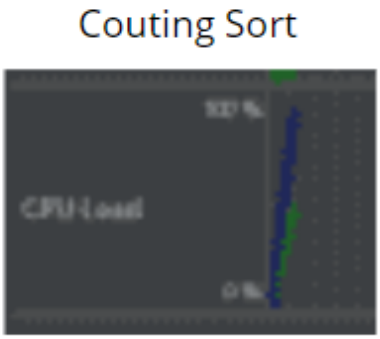


- Selection Sort : 22106 ms
- Insertion Sort : 8724 ms
- Merge Sort : 78 ms
- Couting Sort : -
- Quick Sort : 7504 ms
- Quick sort M3 : 1949 ms
- Heap Sort: 145 ms

Uso de CPU


Selection Sort
Insertion Sort
Quick Sort


Heap Sort
CPU Load


Couting Sort
CPU Load


Quick Sort M3
CPU Load

Nos gráficos pode ser observado que em alguns algoritmos a exercução em relação ao tempo e a cpu é mais rápida, ou seja, mais eficiente.

Quais os algoritmos mais eficientes e por qual motivo?

	Ordenação por óbitos			Ordenação por casos confimados			Ordenação por cidades		
Tipo de teste	Melhor caso	Médio caso	Pior caso	Melhor caso	Médio caso	Pior caso	Melhor caso	Médio caso	Pior caso
Selection Sort	47 ms	63 ms	63 ms	47 ms	47 ms	47 ms	8284 ms	8285 ms	8724 ms
Insertion Sort	3 ms	49 ms	63 ms	2 ms	37 ms	94 ms	78 ms	3940 ms	22106 ms
Merge Sort	15 ms	-	-	15 ms	-	-	141 ms	93 ms	78 ms
Quick Sort	174 ms	79 ms	107 ms	123 ms	-	47 ms	8566 ms	1266 ms	7504 ms
Quick Sort com Mediana de 3	142 ms	68 ms	62 ms	45 ms	2 ms	-	2854 ms	109 ms	1949 ms
counting	6 ms	3 ms	3 ms	28 ms	11 ms	21 ms	-	-	-
Heap Sort	8 ms	2 ms	3 ms	10 ms	2 ms	2 ms	180 ms	118 ms	145 ms

De acordo com a tabela acima, é notório observa que quando se trata de grandes quantidades de dados alguns mostraram ser mais rápidas em exercução que outros, como também comparar as diferenças entre o melhor, médio e pior caso.

Como vimos em sala de aula, alguns algoritmos como o quick sort mediana de 3, por ter um pivot mais lógico, ele se torna mais eficiente. O heap Sort por ser melhor em trabalhar com grande quantidades de dados se mostrou mais rapido em todos os casos de ordenação, e testes. O insertion, selection e merge sort obtiveram resultados esses resultados, por se tratar de trabalhar com uma quantidades de dados, fazendo assim mais intereações pela lógica dele.