Laboratório de Estrutura de Dados

Primeira versão do projeto da disciplina

Comparação entre os algoritmos de ordenação elementar

João Vitor Barbosa da Silva e Klayton Marcos Veloso da Silva Júnior,

1. Introdução

Este relatório corresponde ao relato dos resultados obtidos no projeto da disciplina de LEDA e EDA que tem como objetivo ordenar dados emitidos no site do brasil.io relacionados ao COVID-19.

O sistema tem como objetivo fazer três ordenações por algoritmo e gerar suas métricas, essas ordenações são: ordenar pelo número de casos, ordenar por cidade (ordem alfabética) e ordenar pelo número de óbitos.

Ao final de cada execução do algoritmo escolhido, temos como resultado 6 arquivos .csv, 3 com as métricas para cada tipo de ordenação e os outros 3 são o documento principal ordenados pelo seu respectivo parâmetro. Vale ressaltar que no código do sistema já tem o arquivo .csv do brasil.io completo e filtrado para pegar apenas os últimos casos e os casos não repetidos.

Nos próximos tópicos, serão abordados uma descrição dos métodos utilizados para a construção e análise do sistema, os testes que foram feitos, a descrição dos ambientes utilizados para desenvolver e testar o sistema e por fim no último tópico os resultados dos testes com os gráficos e as análises necessárias.

2. Descrição geral sobre o método utilizado

O sistema foi testado em duas máquinas diferentes com o mesmo código, presente no github do projeto (link). Após rodar todos os algoritmos nas duas máquinas, foram pegos os arquivos de métrica gerados pelo sistema e criados os gráficos por meio do Pandas utilizando Python. Os gráficos estão na próxima sessão, seguidos da análise de desempenho geral.

A implementação de todo sistema foi feito com Java, os sistema funciona lendo o arquivo csv, em seguida roda o algoritmo selecionado pelo usuário, que usará a leitura do csv para ordená-lo, criando outro arquivo csv com a ordenação e com as métricas. O mesmo csv será ordenado 3 vezes, mas por critérios diferentes, primeiramente será ordenado pelo

número de casos, depois pelo número de óbito e por fim ordenar as cidades por ordem alfabética. Para cada ordenação dessa serão criadas as suas respectivas métricas, métricas essas que foram utilizadas para a construção dos gráficos da sessão 3. Os algoritmos abordados no sistema são: Selection Sort, Insertion Sort, Merge Sort, Quick Sort, Quick Sort com Mediana de 3, Heap Sort e Counting Sort.

Descrição geral do ambiente de testes

Máquina 1:

Processador: 2,3 GHz Intel Core i3-7020U

Memória: 4 GB 2667 MHz DDR4

Sistema Operacional: Linux Ubuntu

Máquina 2:

Processador: 2,3 GHz Intel Core i5 Dual-Core

Memória: 8 GB 1333 MHz DDR3

Sistema Operacional: macOS Catalina

3. Resultados e Análise

Após os testes realizados podemos concluir que a máquina e o algoritmo interferem bastante na execução do código, em seguida temos os gráficos primeiramente da máquina 1 e em seguida da máquina 2. Os gráficos serão divididos pelo tipo de algoritmo e por cada ordenação, são elas: o número de casos, número de óbitos e ordenação das cidades por ordem alfabética.

3.1 Gráficos da Máquina 1

Selection Sort:

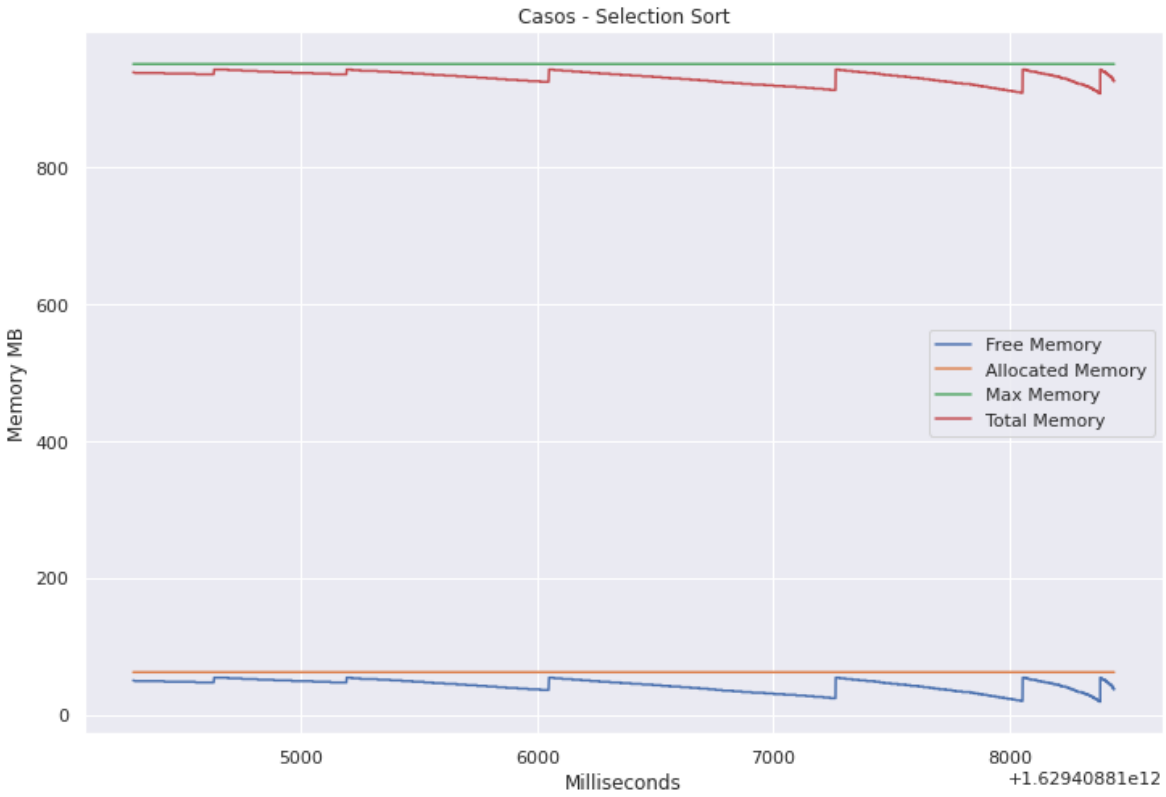

Gráfico 1: Casos Selection Sort Máquina 1.

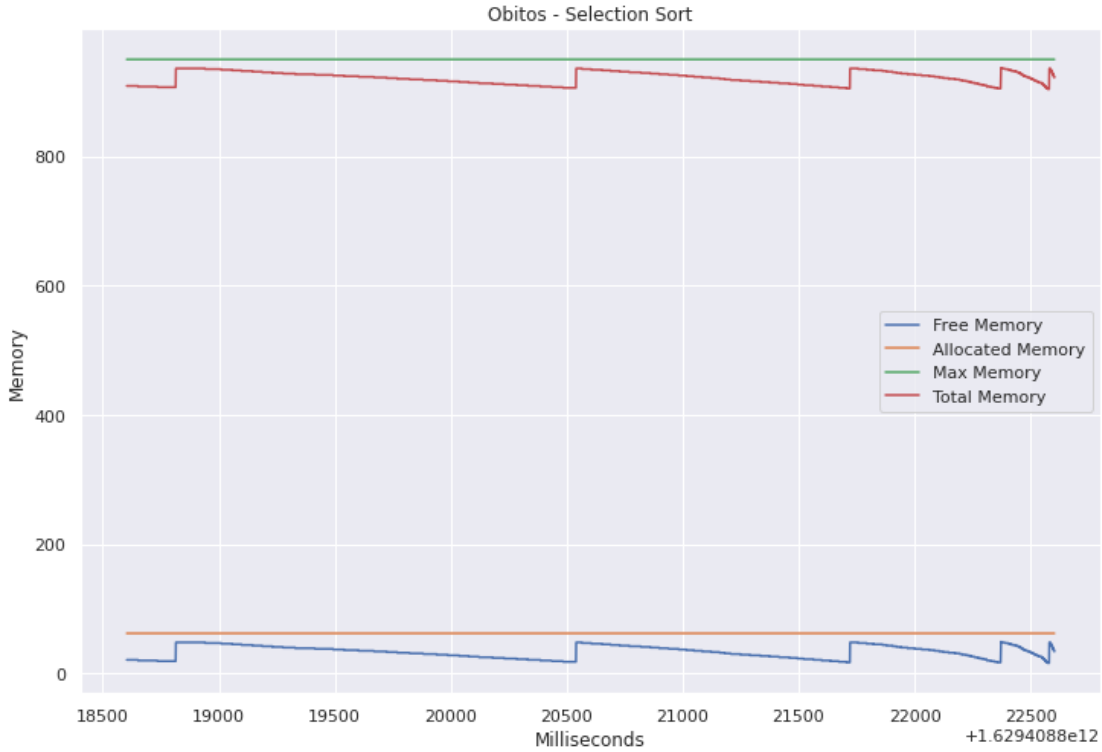

Gráfico 2: Óbitos Selection Sort Máquina 1.

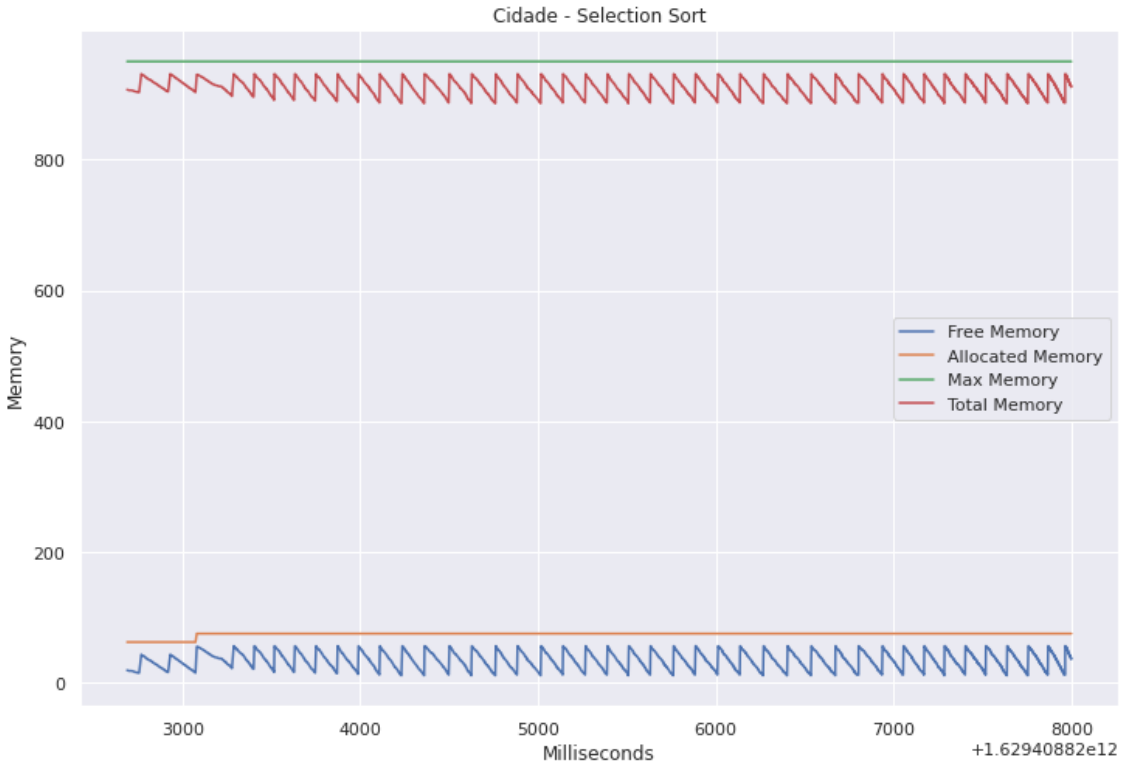

Gráfico 3: Cidade Selection Sort Máquina 1.

Insertion Sort:

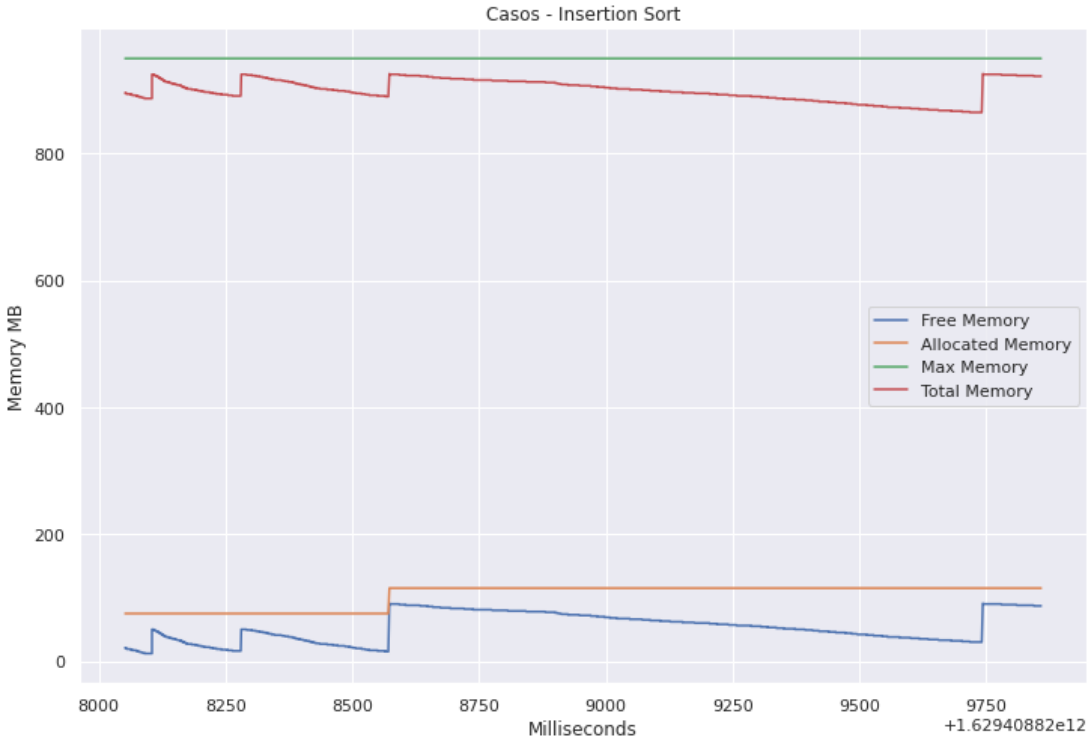

Gráfico 4: Casos Insertion Sort Máquina 1.

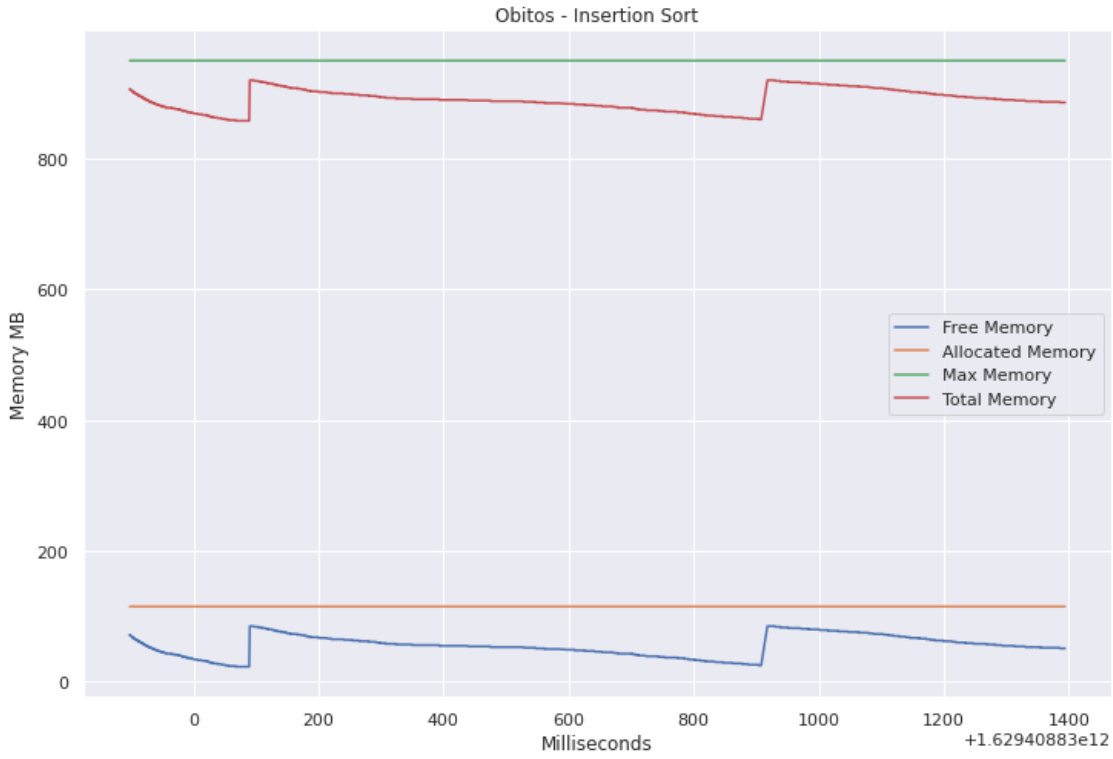

Gráfico 5: Óbitos Insertion Sort Máquina 1.

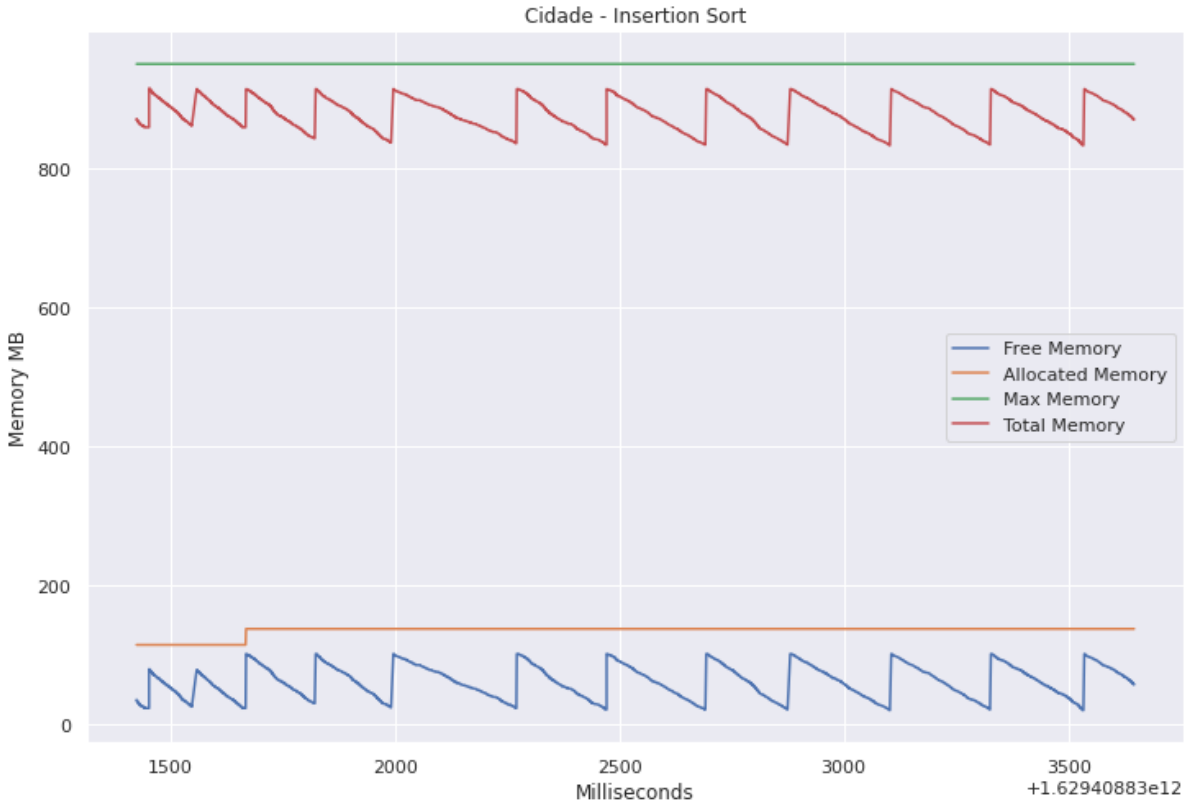

Gráfico 6: Cidade Insertion Sort Máquina 1.

Merge Sort:

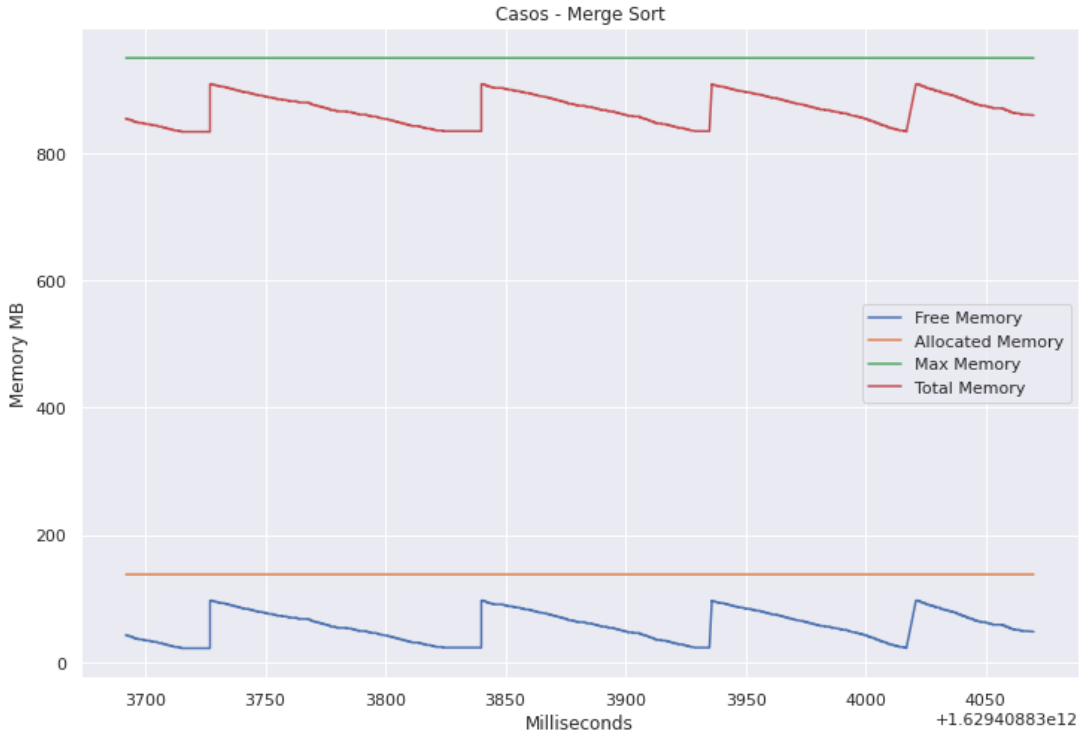

Gráfico 7: Casos Merge Sort Máquina 1.

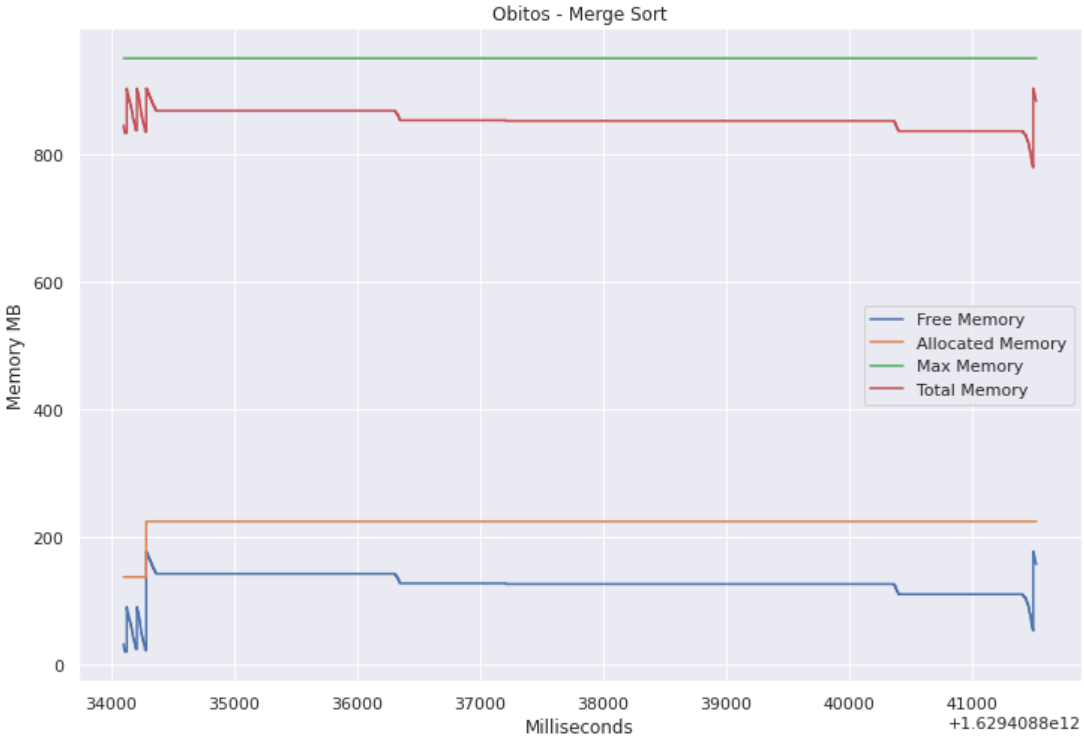

Gráfico 8: Óbitos Merge Sort Máquina 1.

Gráfico 9: Cidade Merge Sort Máquina 1.

Quick Sort:

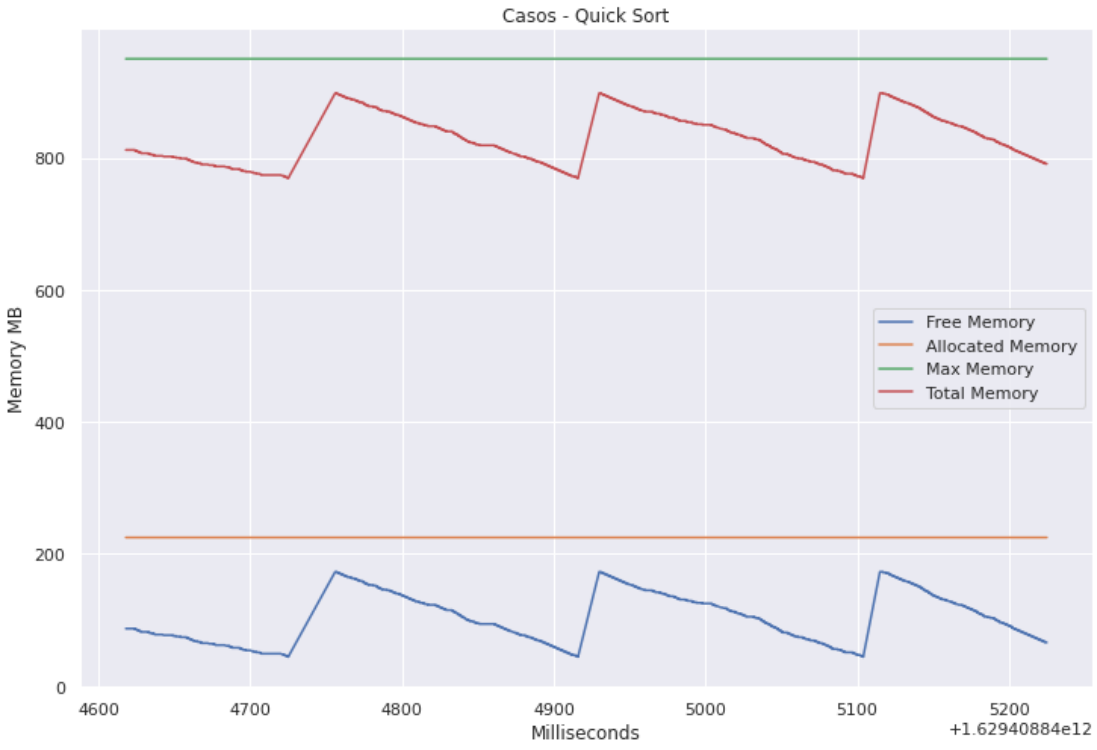

Gráfico 10: Casos Quick Sort Máquina 1.

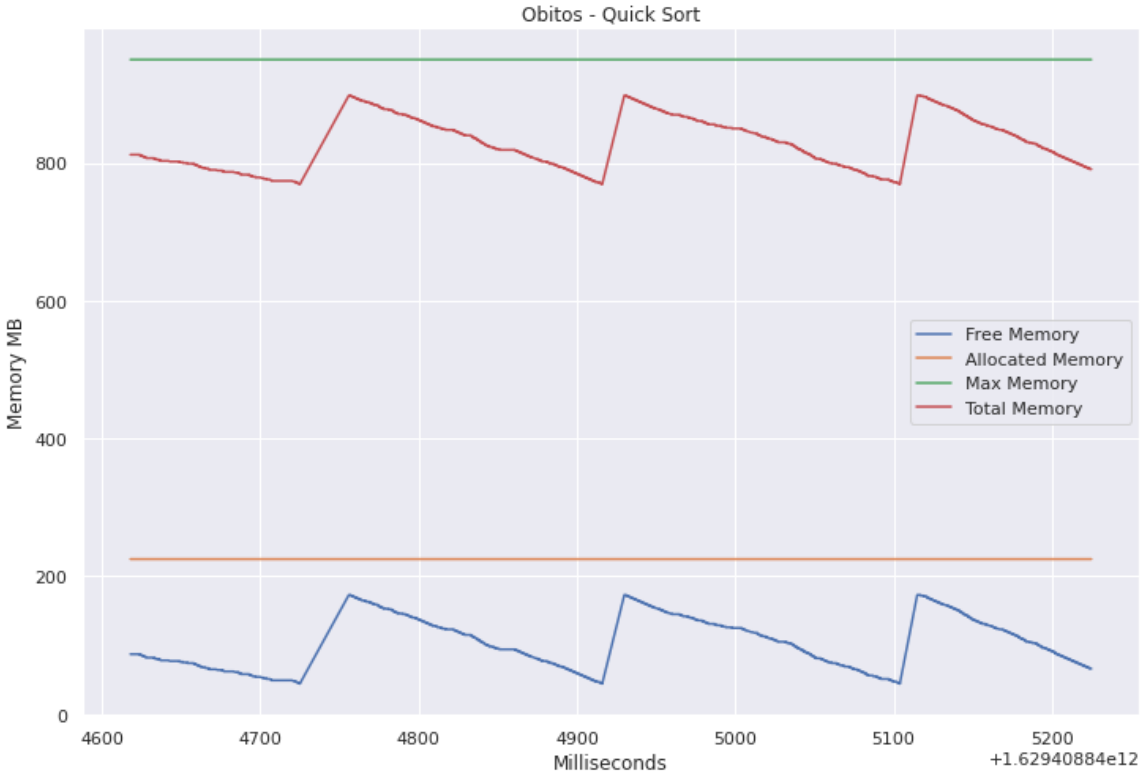

Gráfico 11: Óbitos Quick Sort Máquina 1.

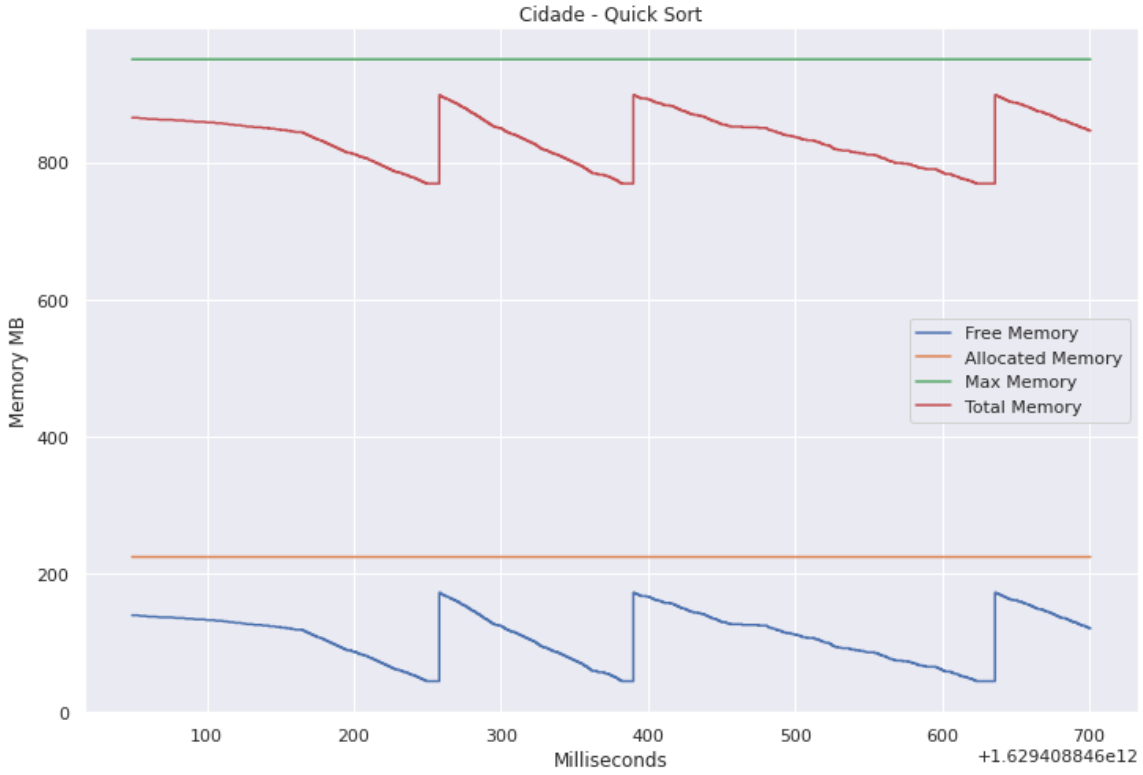

Gráfico 12: Cidade Quick Sort Máquina 1.

Quick Sort com Mediana de 3:

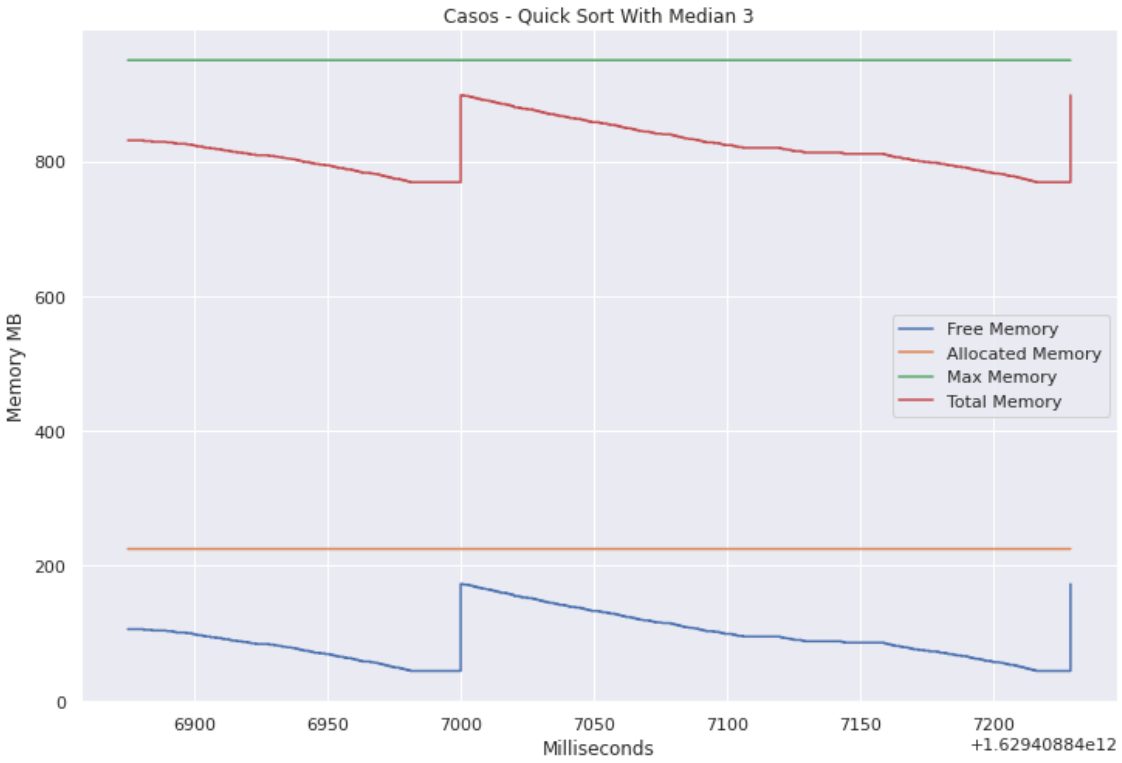

Gráfico 13: Casos Quick Sort Com Mediana de 3 Máquina 1.

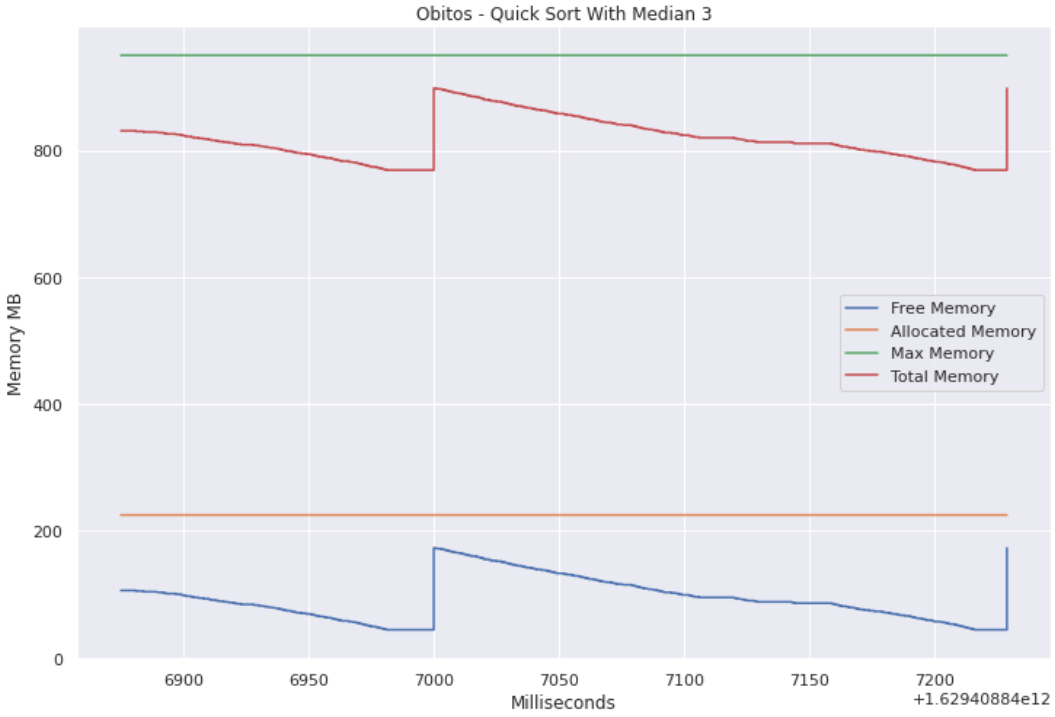

Gráfico 14: Casos Quick Sort com Mediana de 3 Máquina 1.

Gráfico 15: Cidade Quick Sort com Mediana de 3 Máquina 1.

Heap Sort:

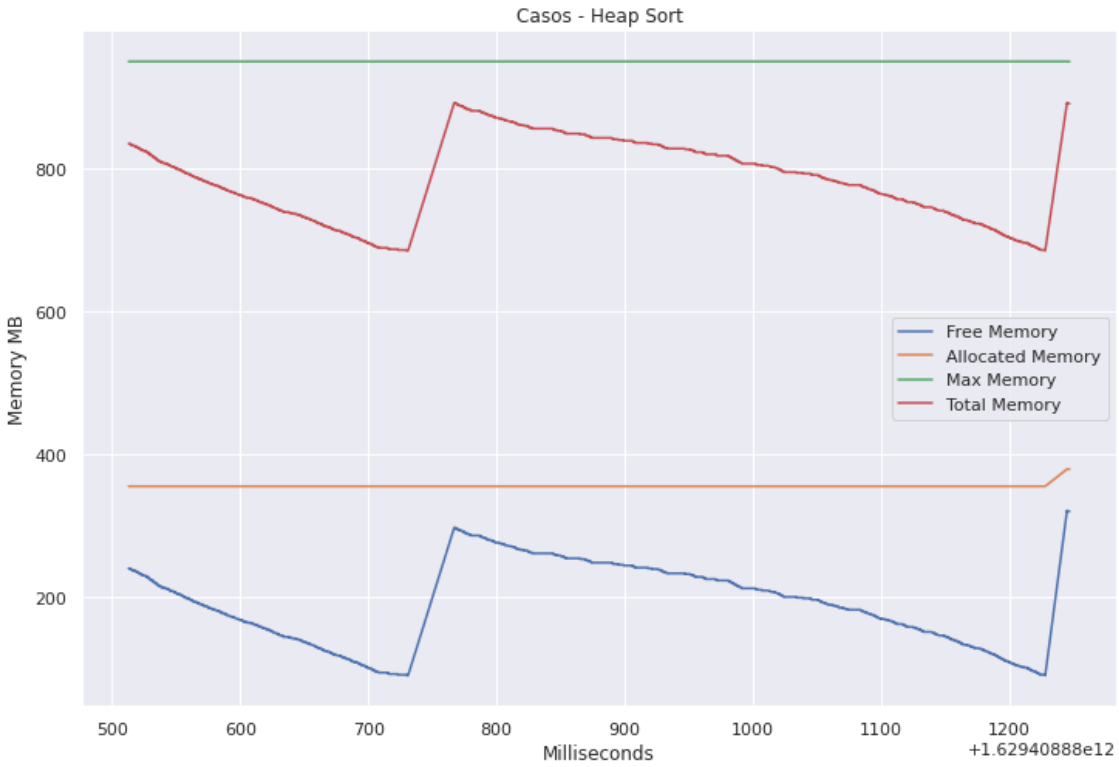

Gráfico 16: Casos Heap Sort Máquina 1.

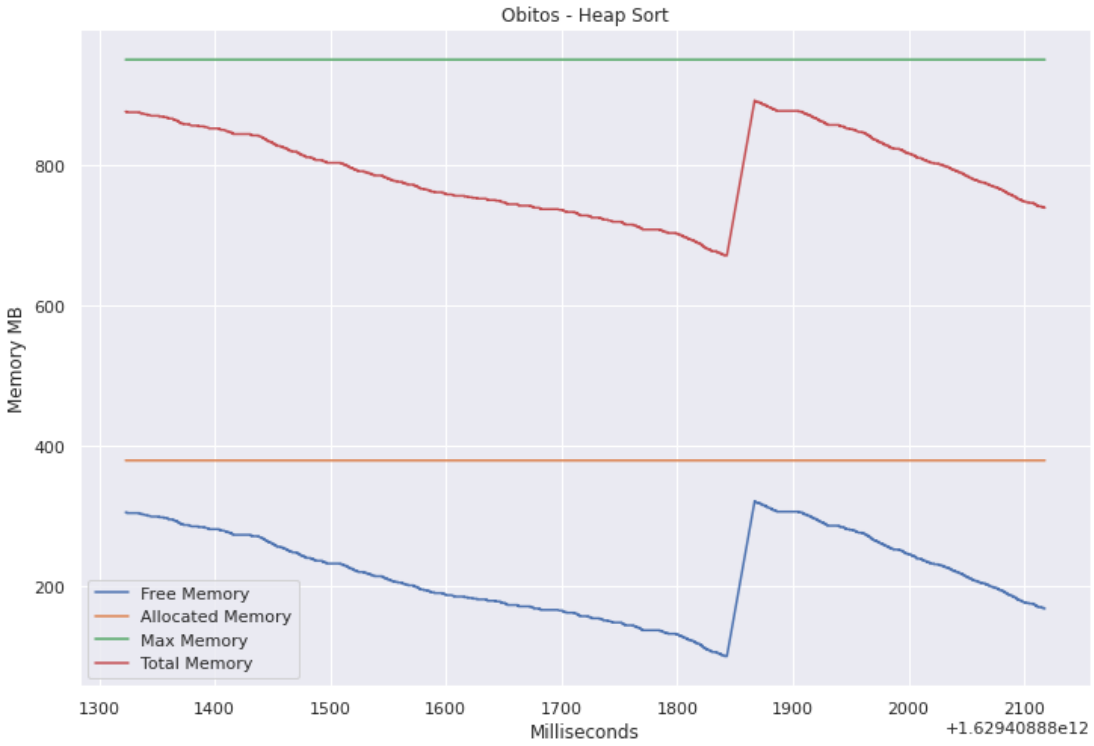

Gráfico 17: Óbitos Heap Sort Máquina 1.

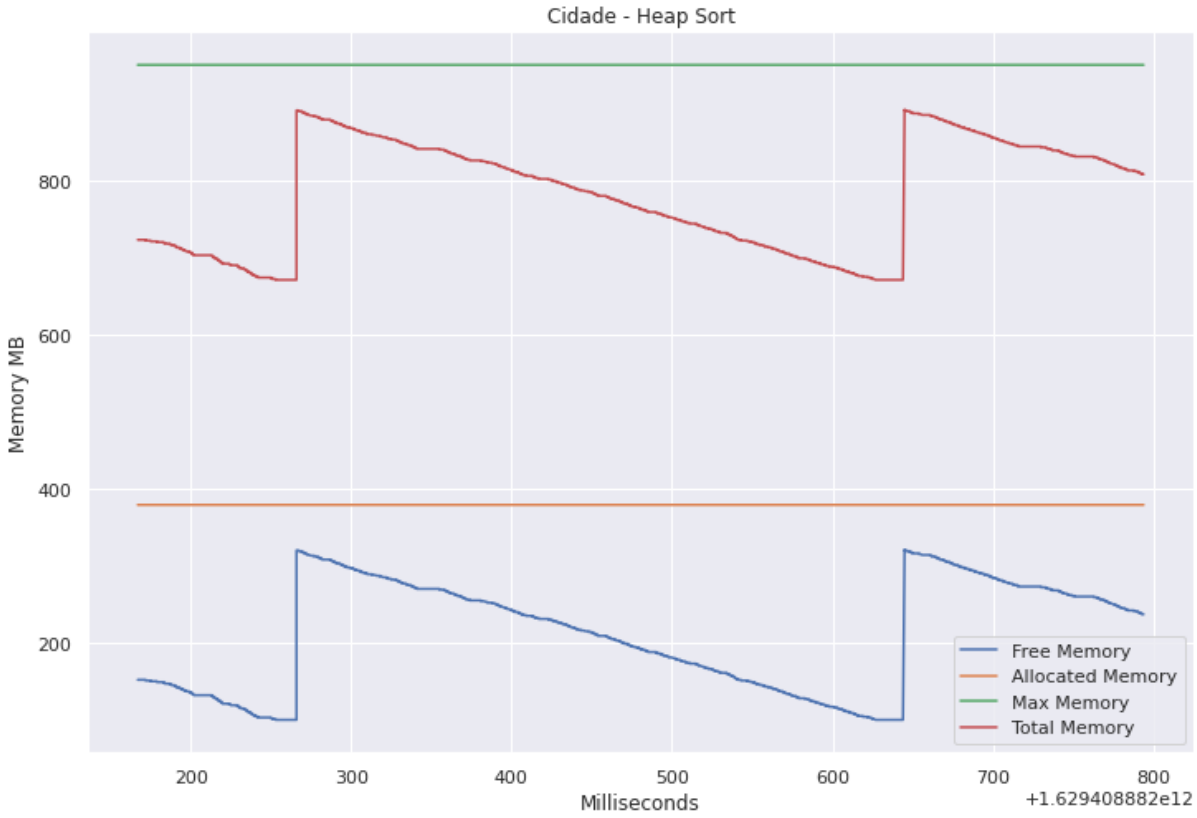

Gráfico 18: Cidade Heap Sort Máquina 1.

Counting Sort:

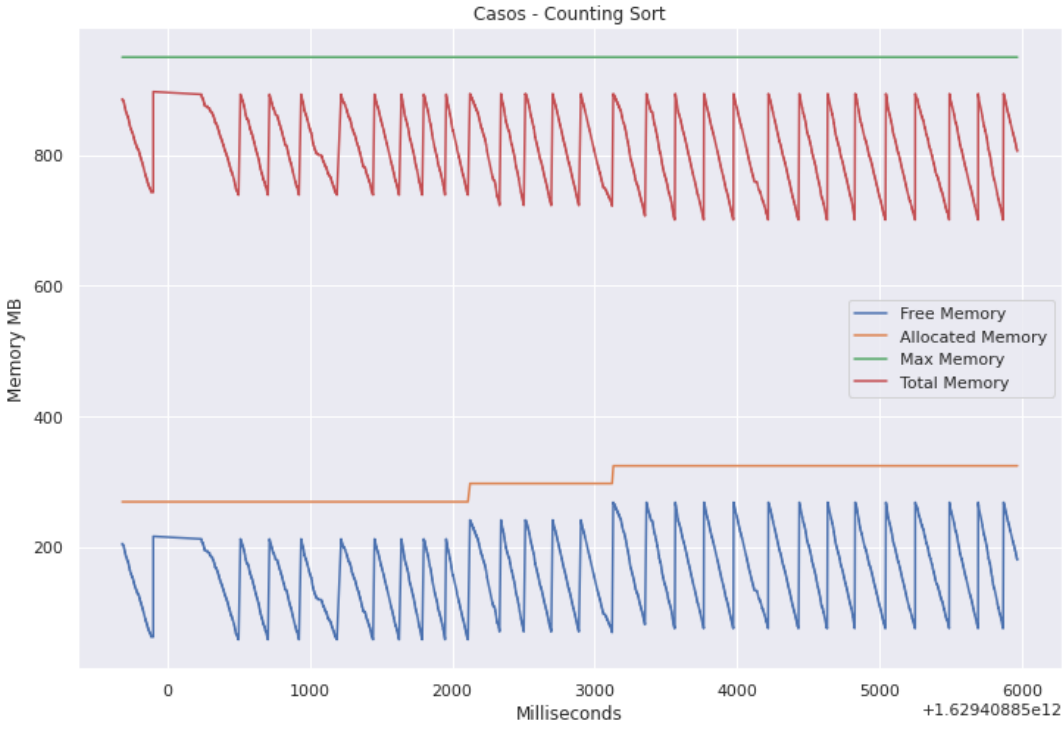

Gráfico 19: Casos Counting Sort Máquina 1.

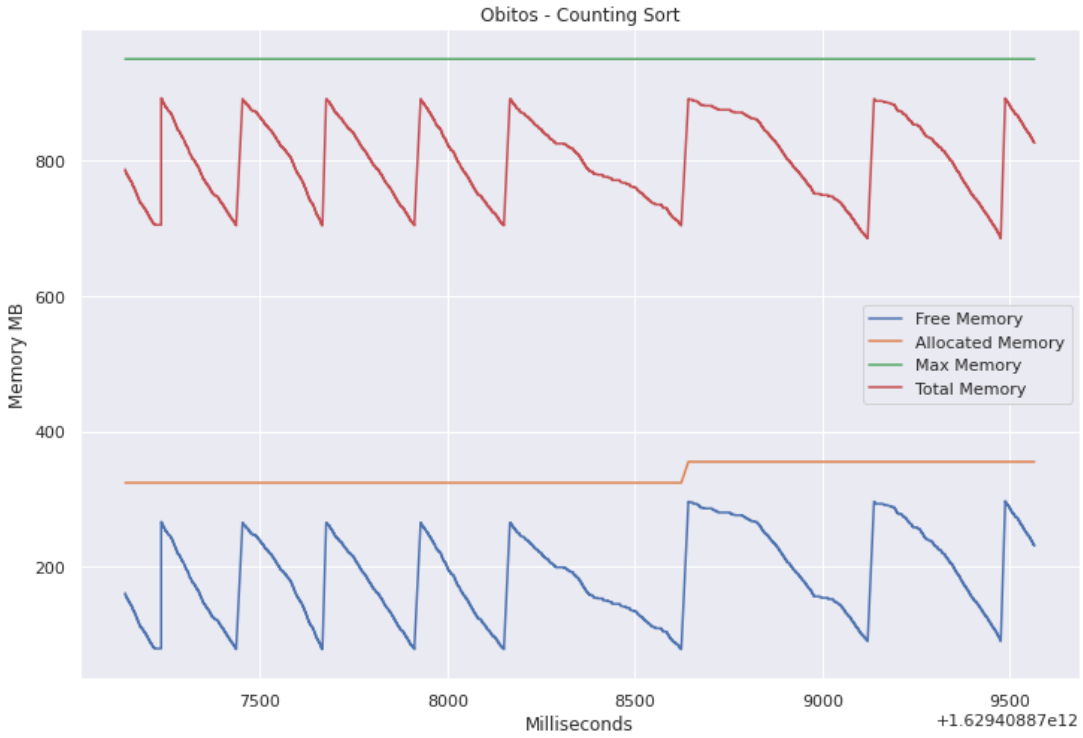

Gráfico 20: Óbitos Counting Sort Máquina 1.

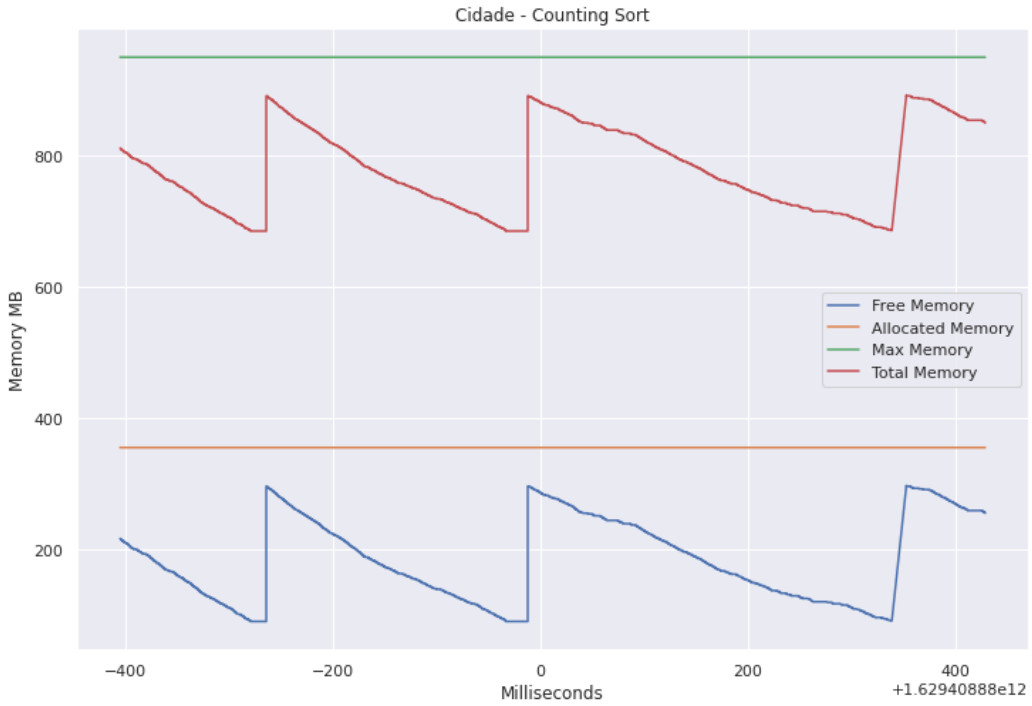

Gráfico 21: Cidade Counting Sort Máquina 1.

3.2 Gráficos da Máquina 2

Selection Sort:

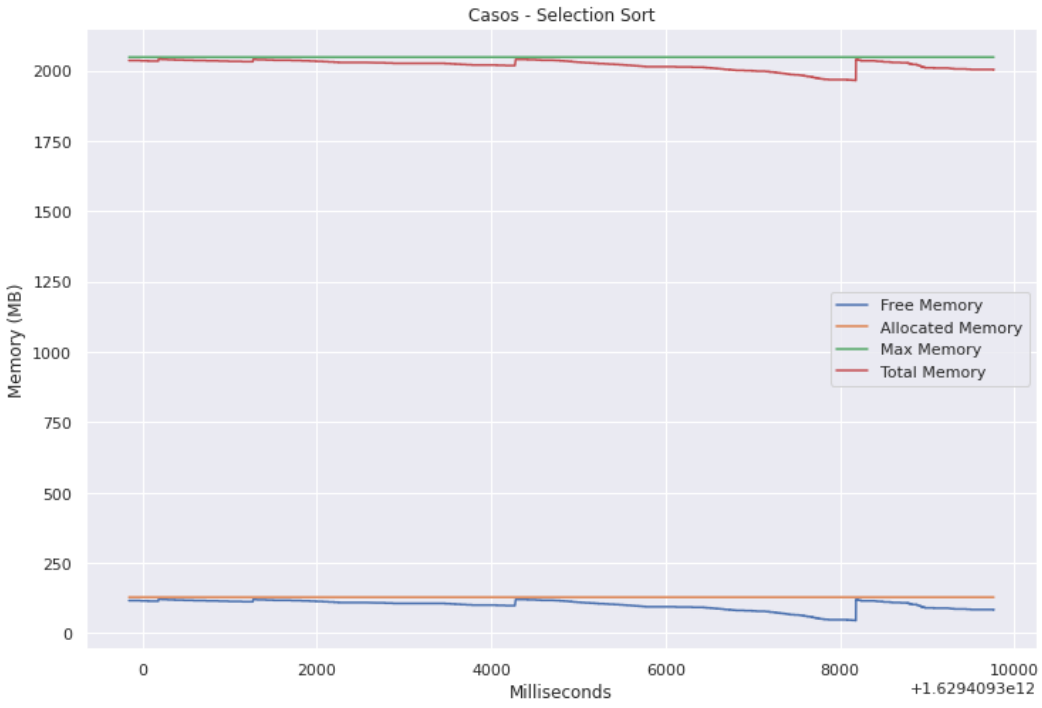

Gráfico 22: Casos Selection Sort Máquina 2.

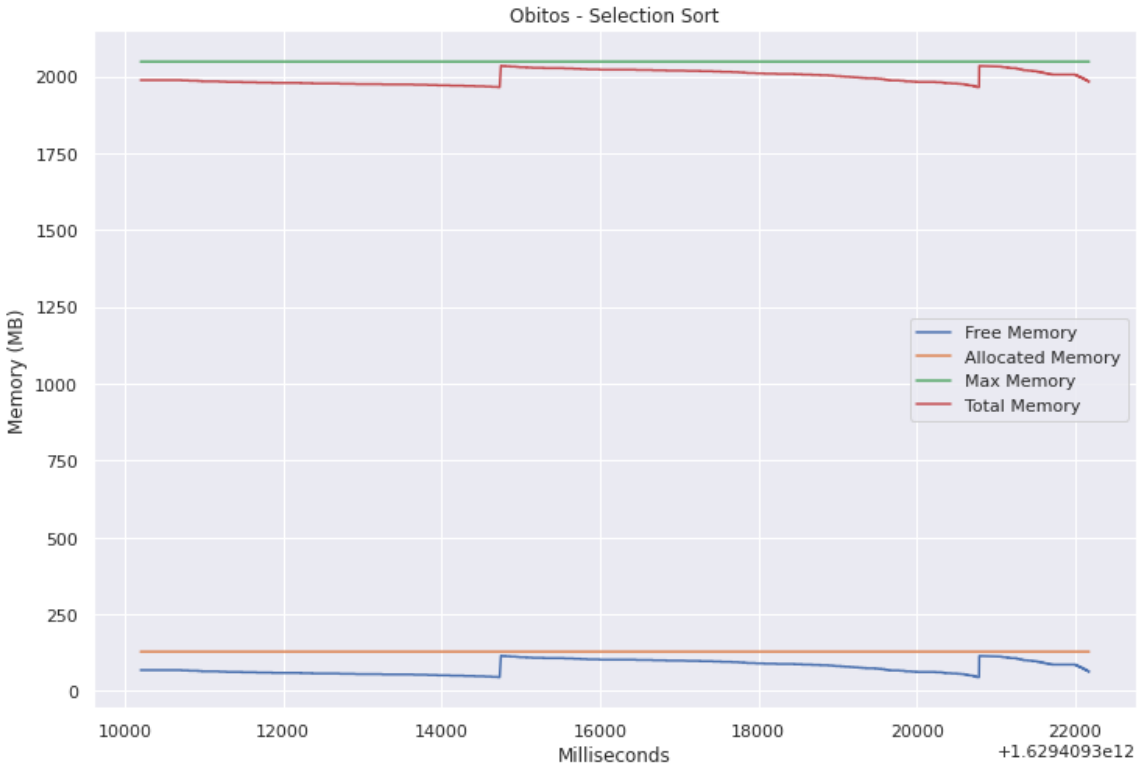

Gráfico 23: Óbitos Selection Sort Máquina 2.

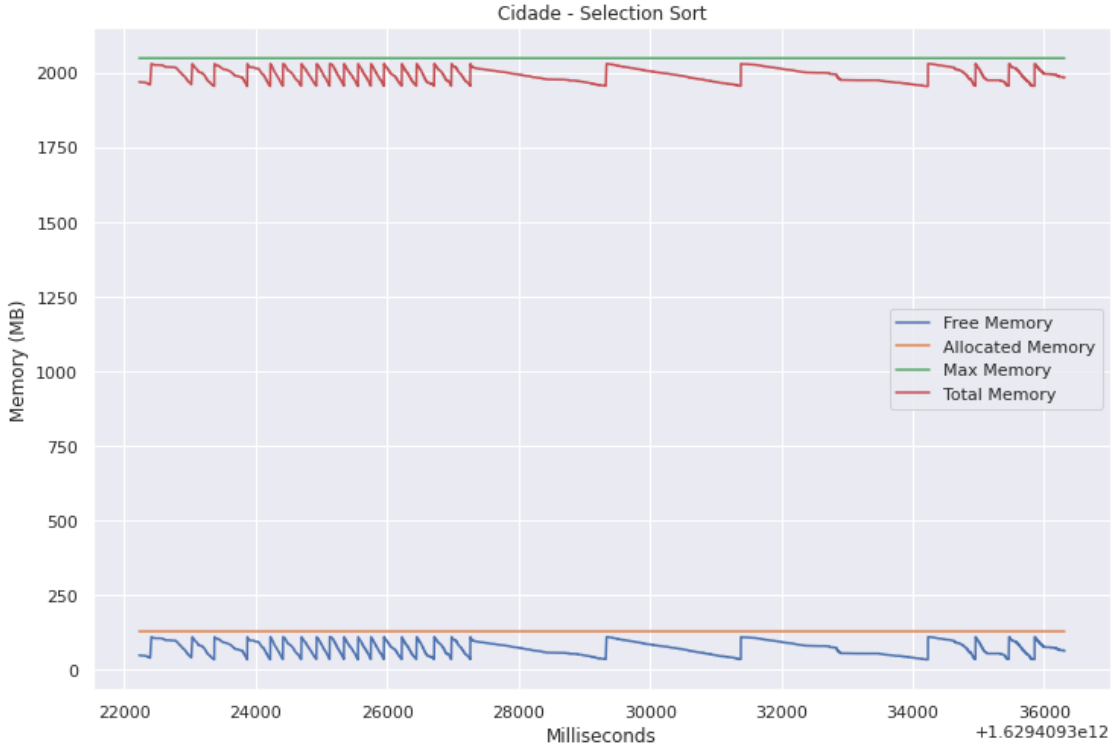

Gráfico 24: Cidade Selection Sort Máquina 2.

Insertion Sort:

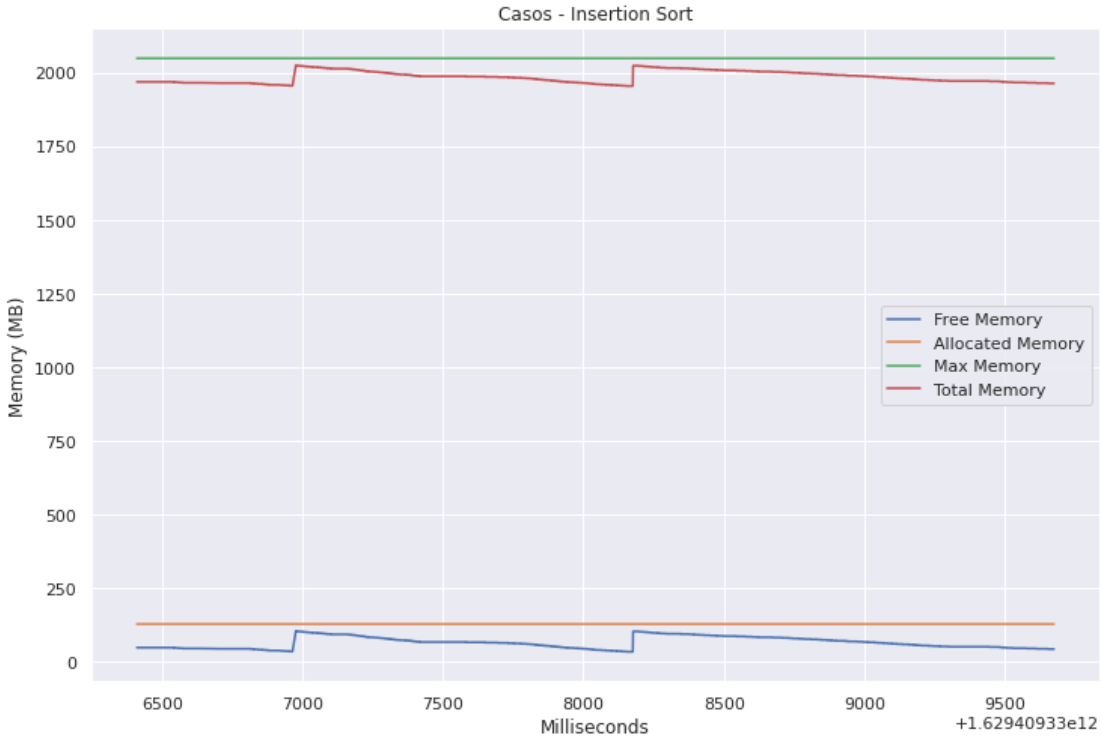

Gráfico 25: Casos Insertion Sort Máquina 2.

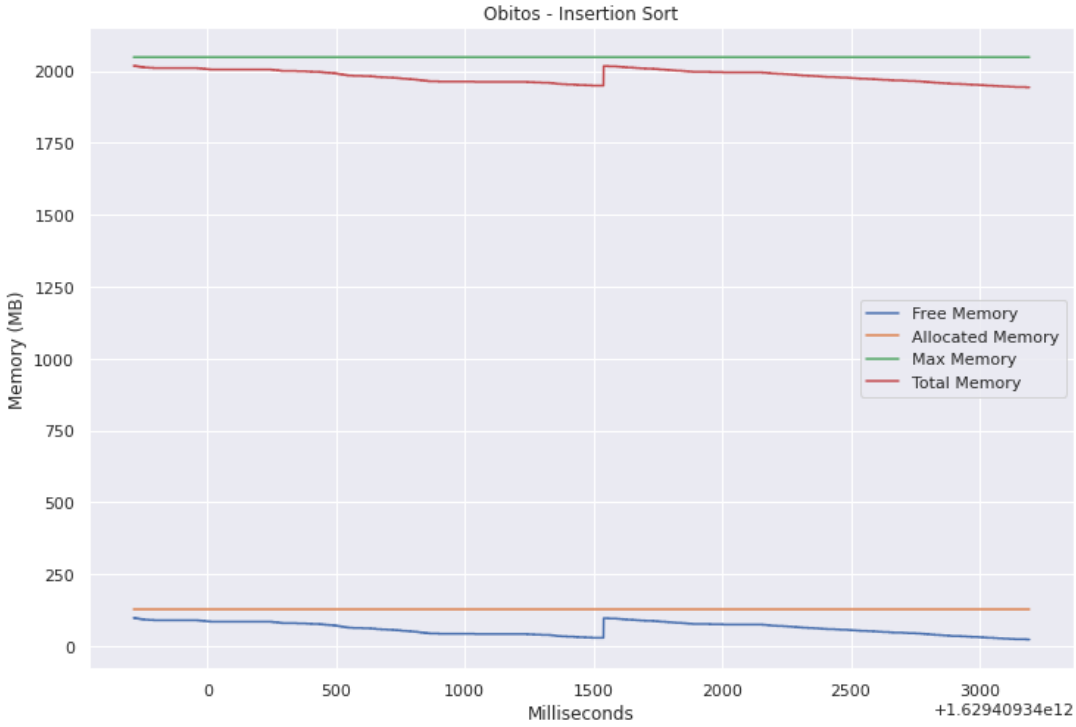

Gráfico 26: Óbitos Insertion Sort Máquina 2.

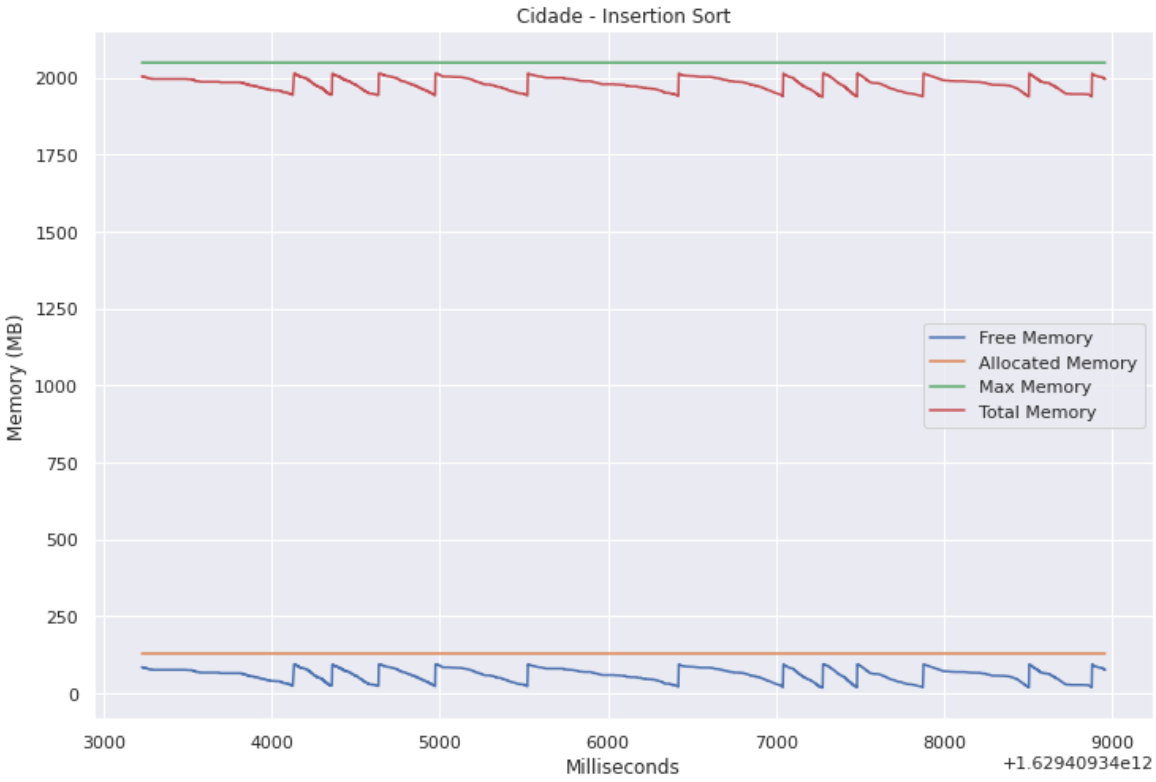

Gráfico 27: Cidade Insertion Sort Máquina 2.

Merge Sort:

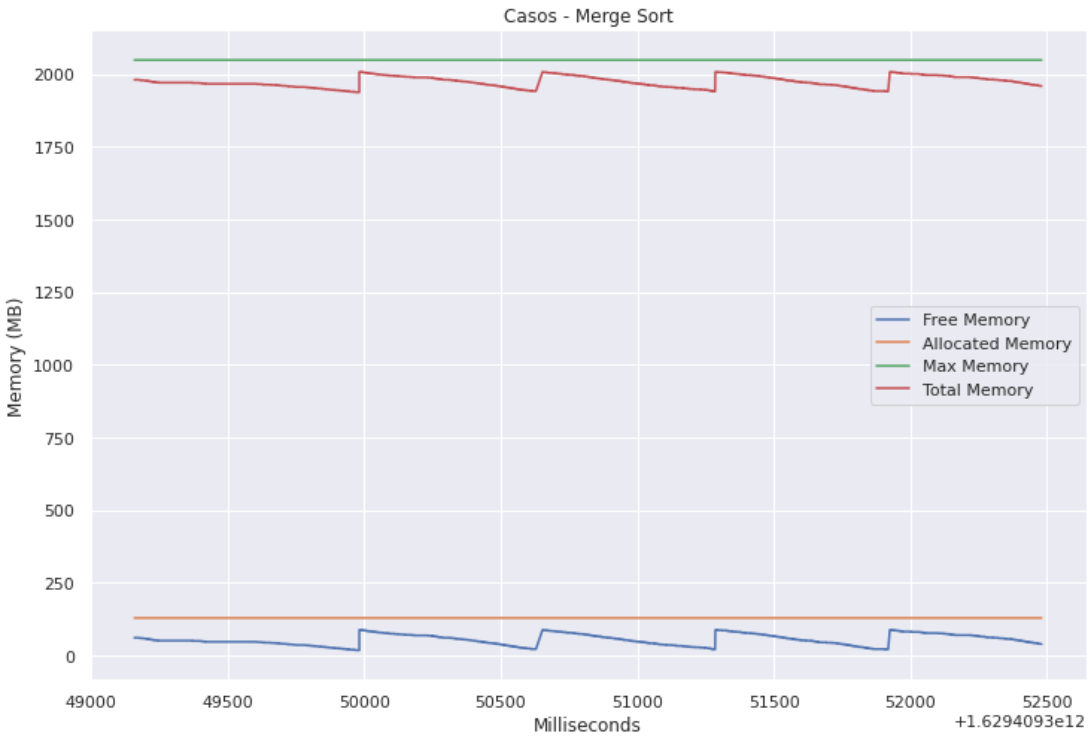

Gráfico 28: Casos Merge Sort Máquina 2.

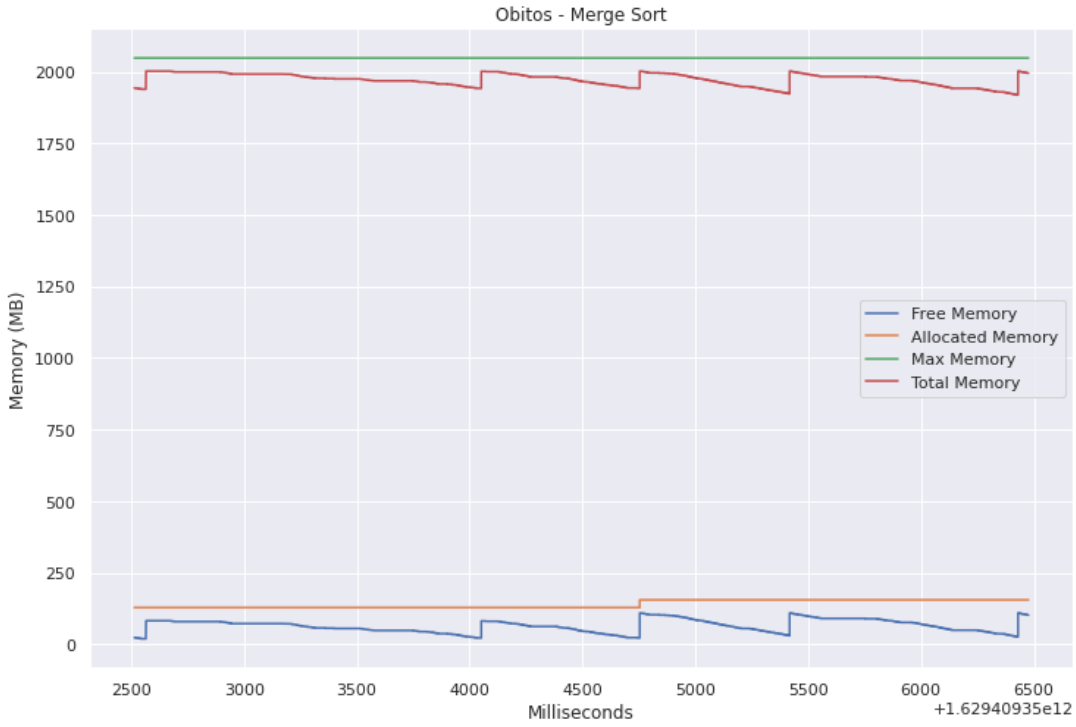

Gráfico 29: Óbitos Merge Sort Máquina 2.

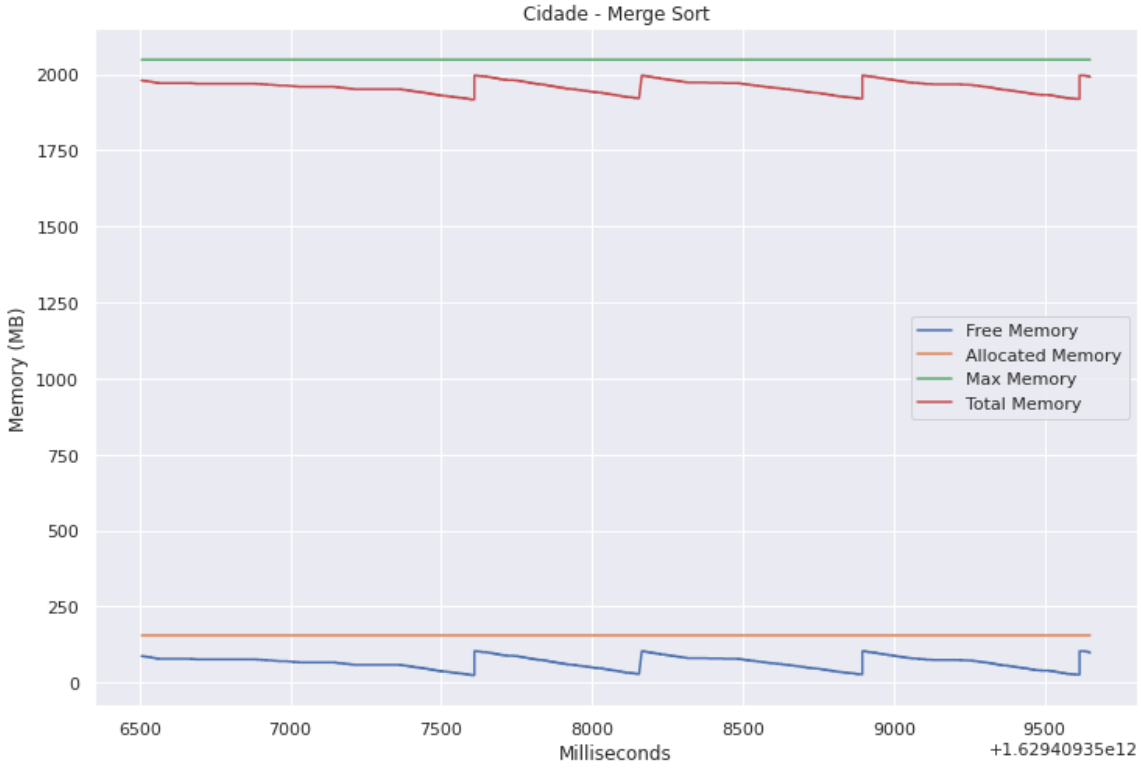

Gráfico 30: Cidade Merge Sort.

Quick Sort:

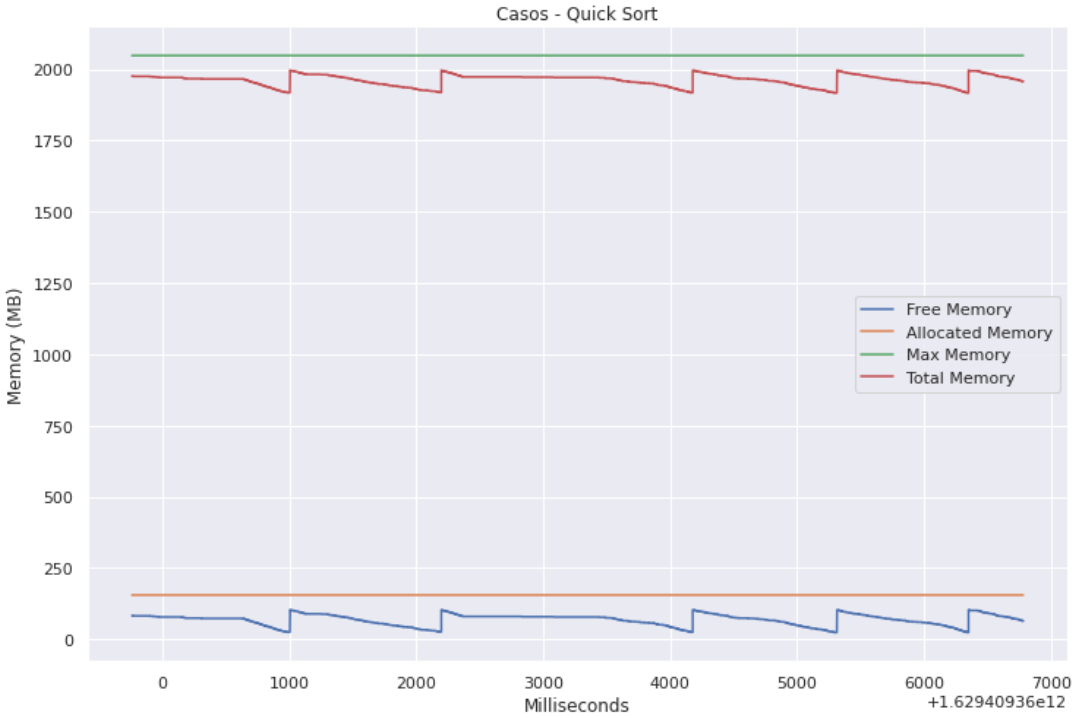

Gráfico 31: Casos Quick Sort Máquina 2.

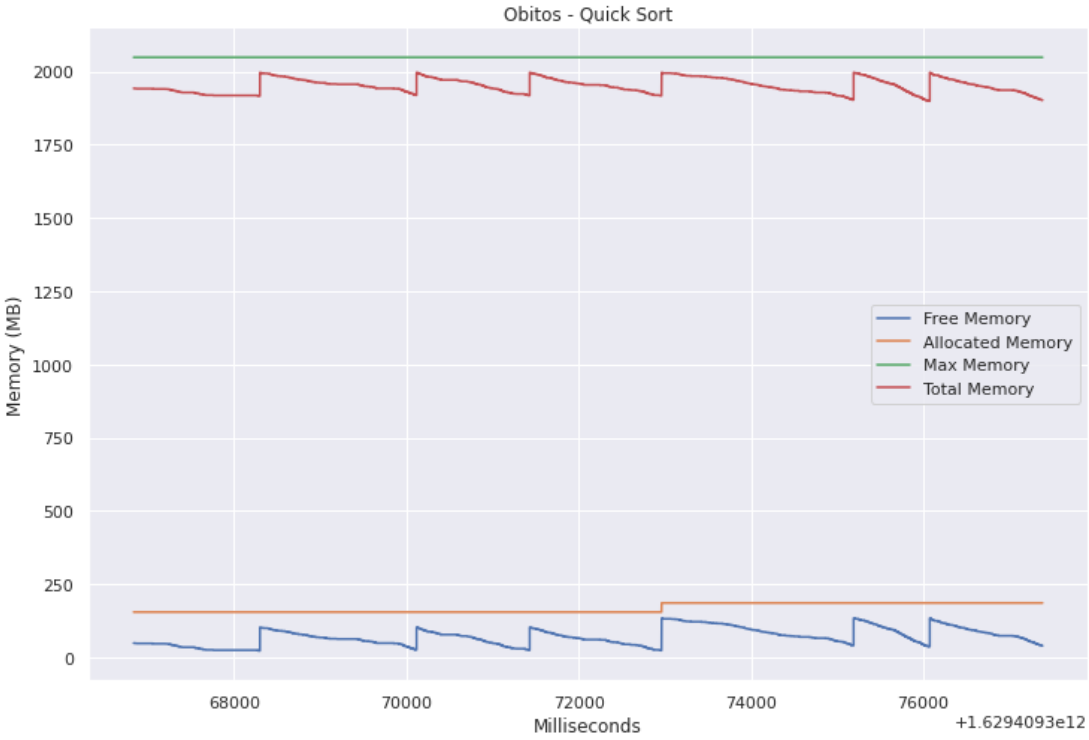

Gráfico 32: Óbitos Quick Sort Máquina 2.

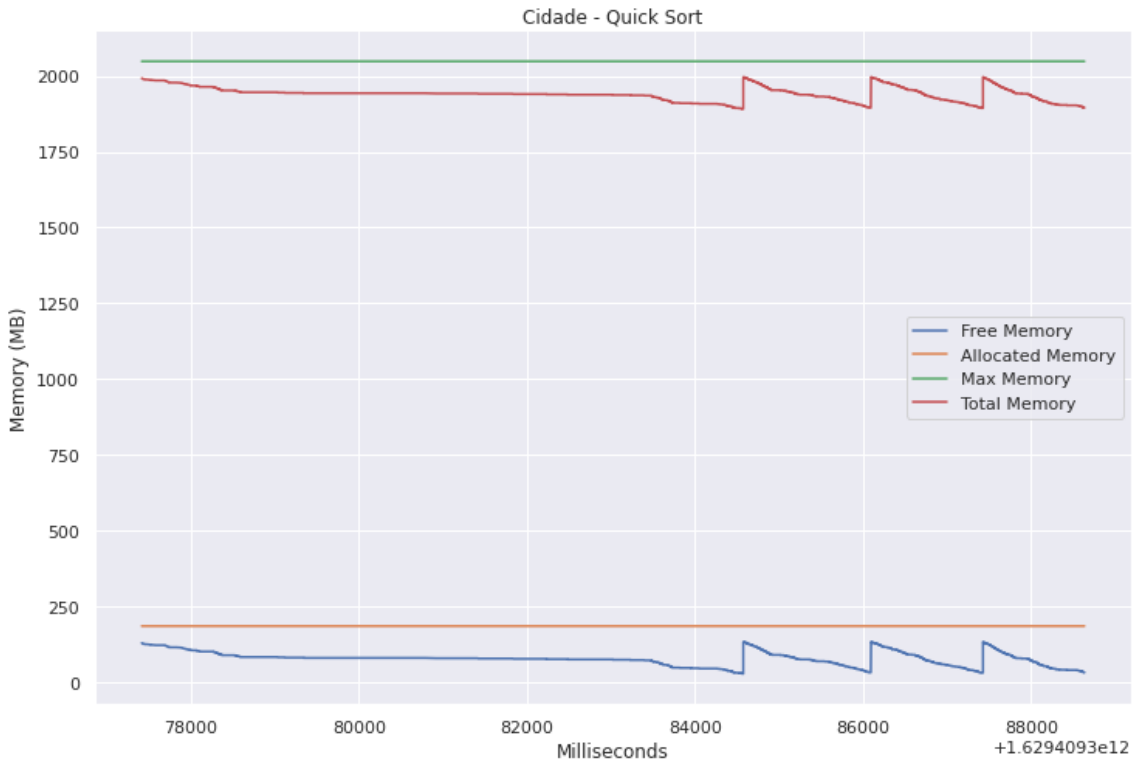

Gráfico 33: Cidade Quick Sort Máquina 2.

Quick Sort com Mediana de 3:

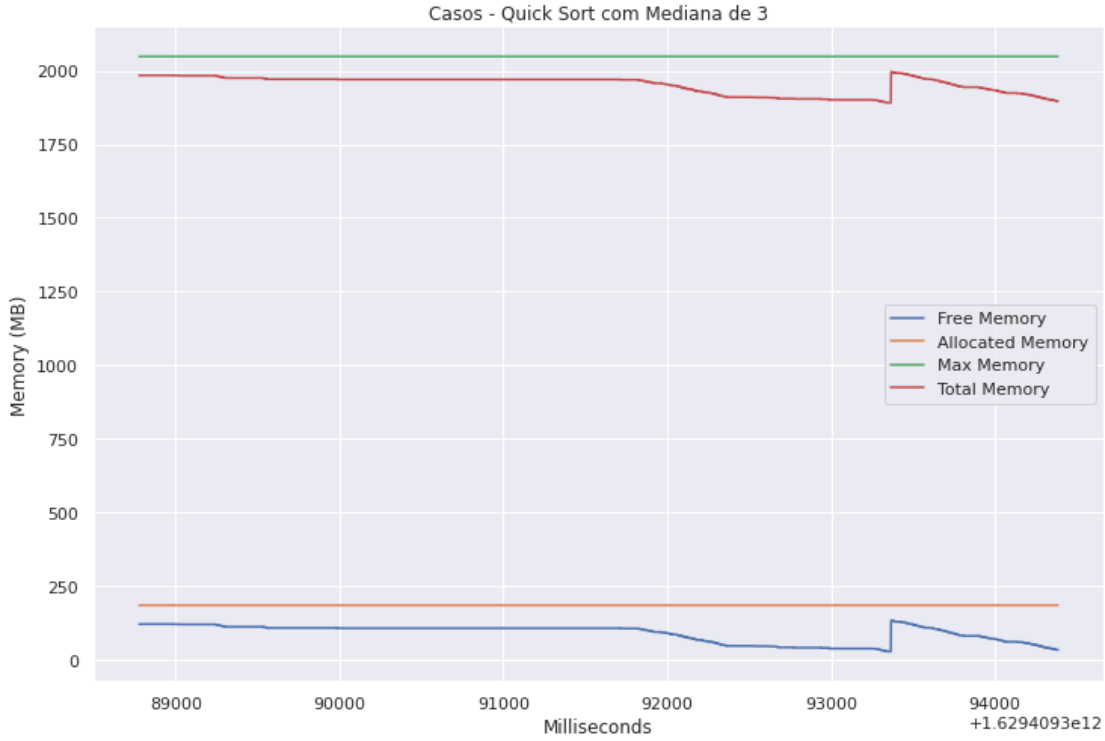

Gráfico 34: Casos Quick Sort Com Mediana de 3 Máquina 2.

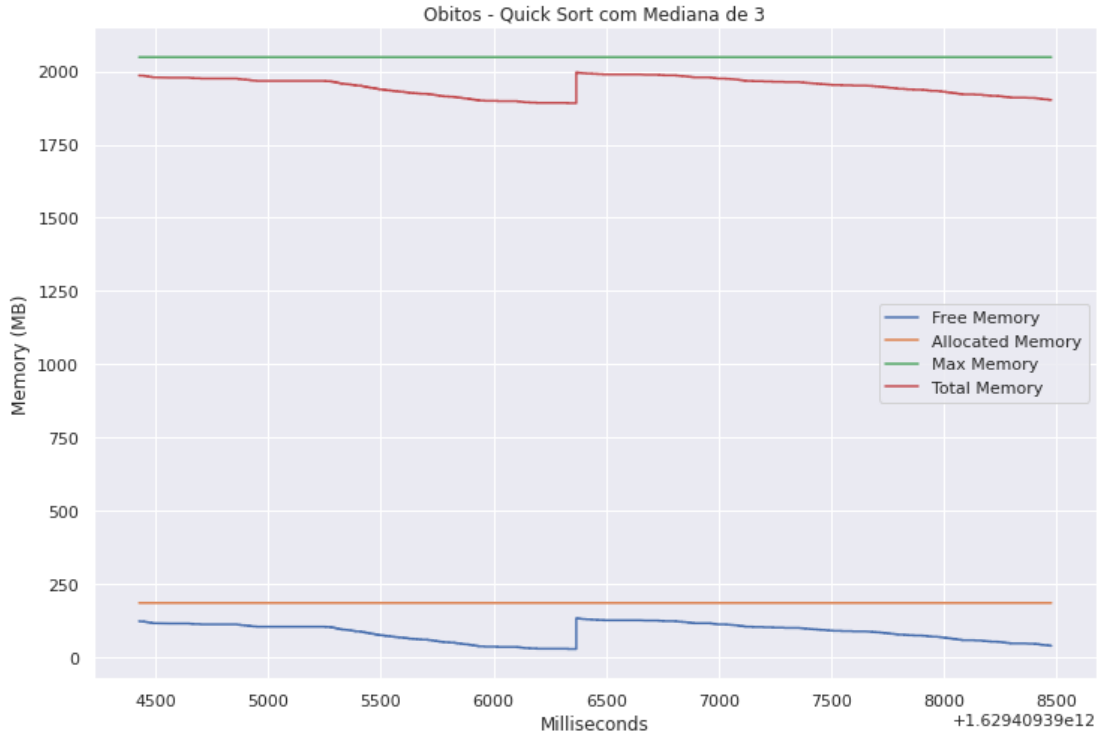

Gráfico 35: Casos Quick Sort com Mediana de 3 Máquina 2.

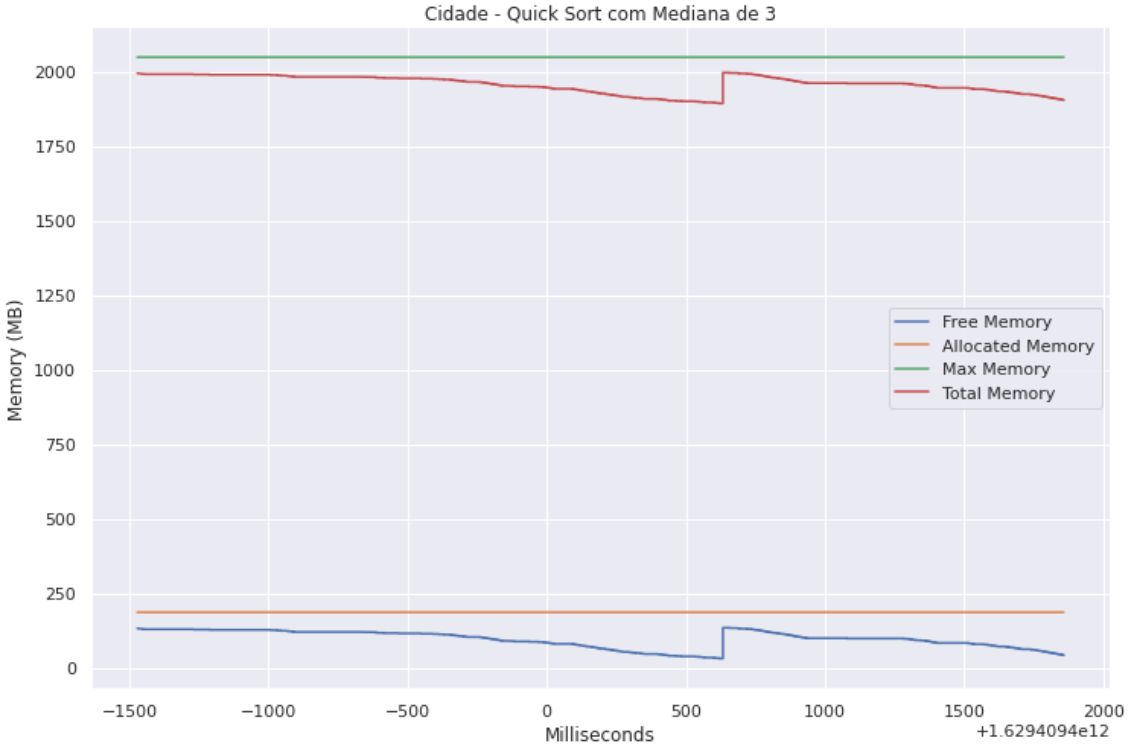

Gráfico 36: Cidade Quick Sort com Mediana de 3 Máquina 2.

Heap Sort:

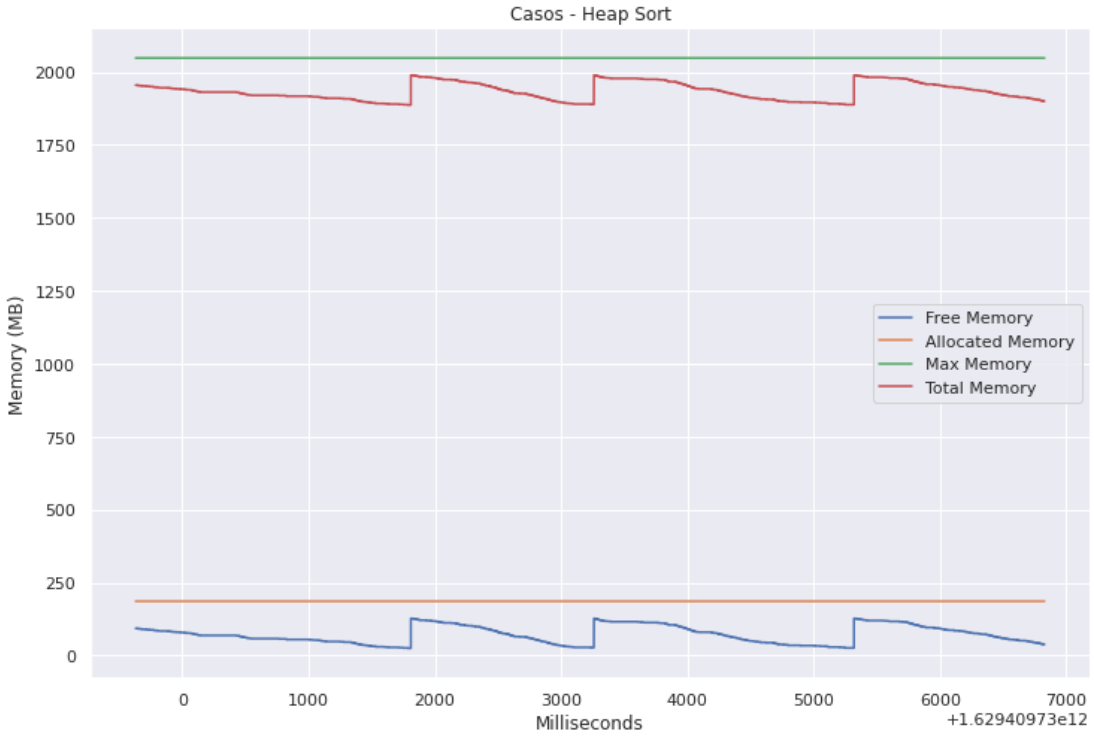

Gráfico 37: Casos Heap Sort Máquina 2.

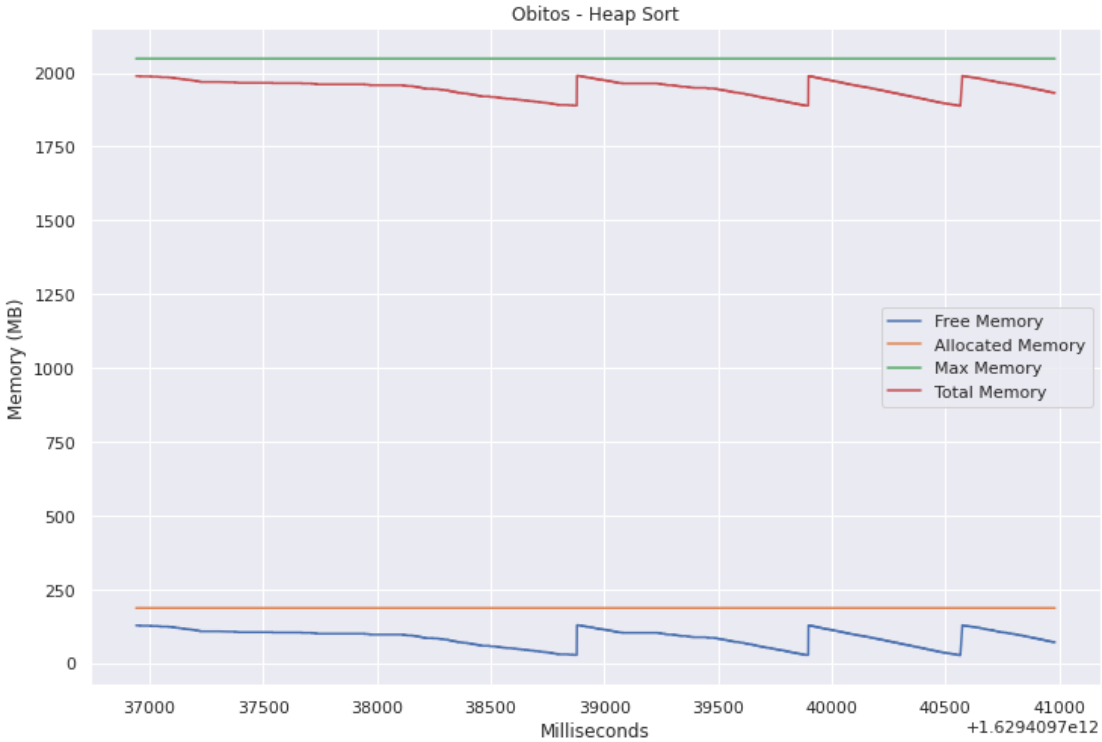

Gráfico 38: Óbitos Heap Sort Máquina 2.

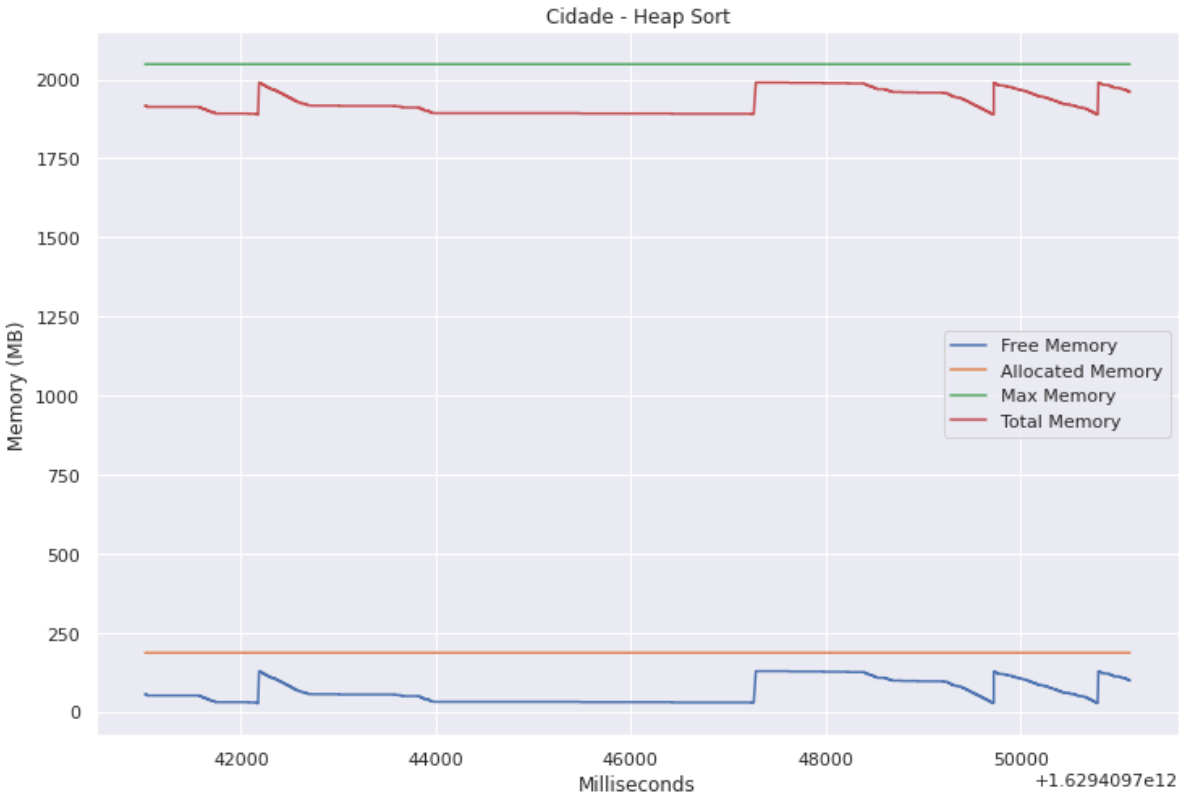

Gráfico 39: Cidade Heap Sort Máquina 2.

Counting Sort:

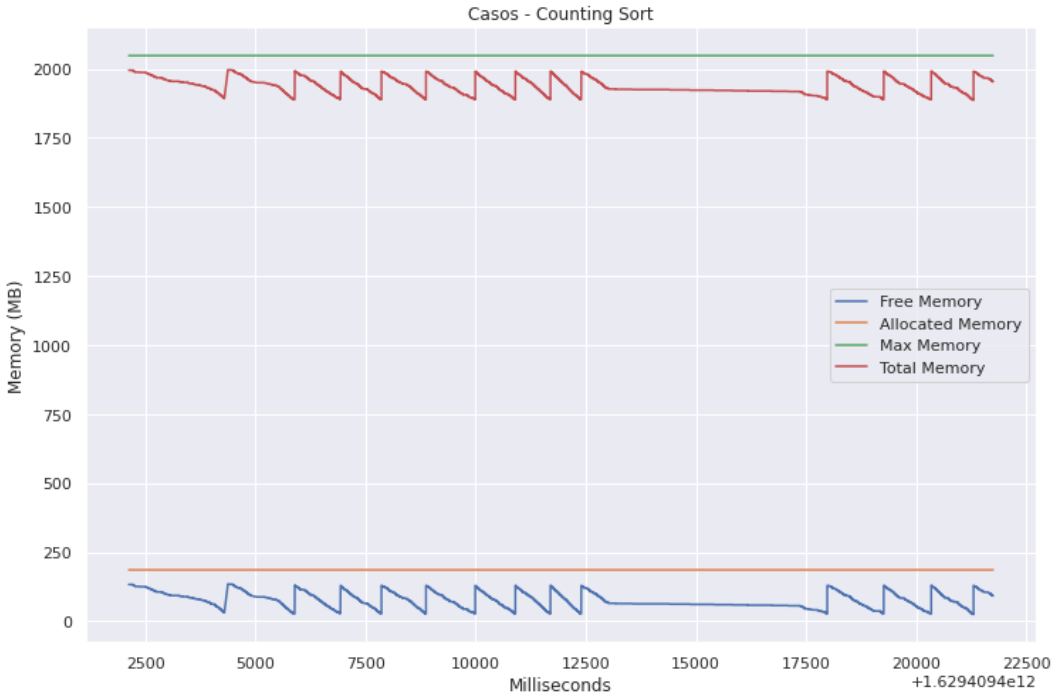

Gráfico 40: Casos Counting Sort Máquina 2.

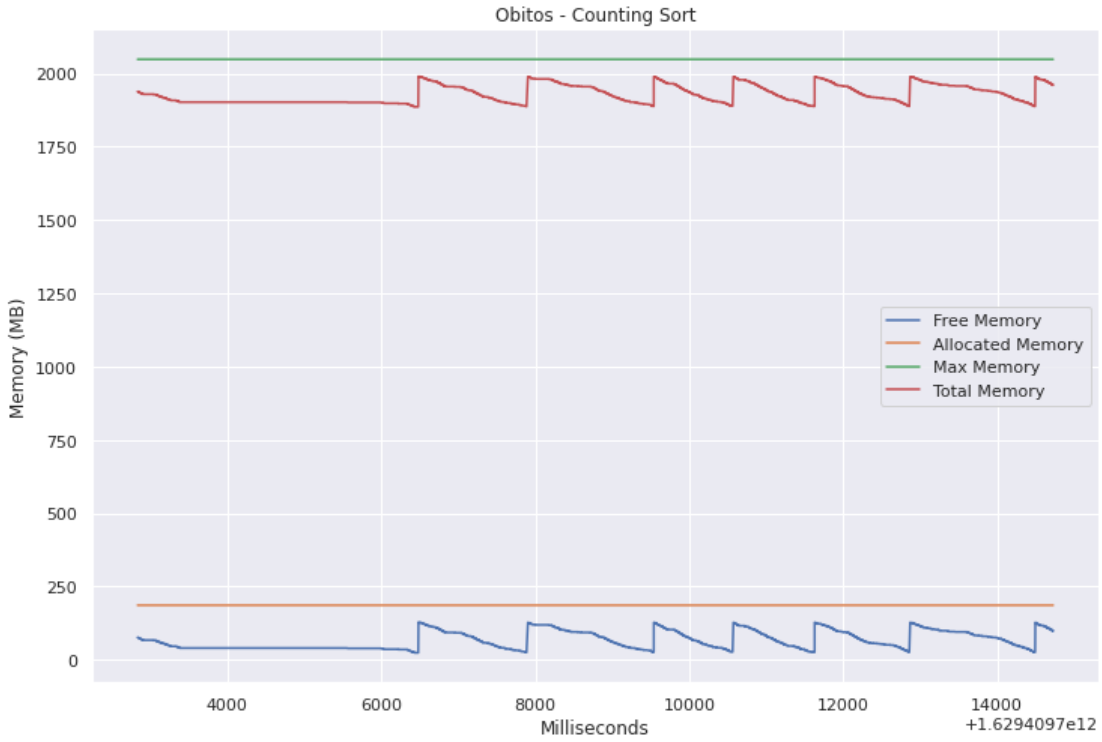

Gráfico 41: Óbitos Counting Sort Máquina 2.

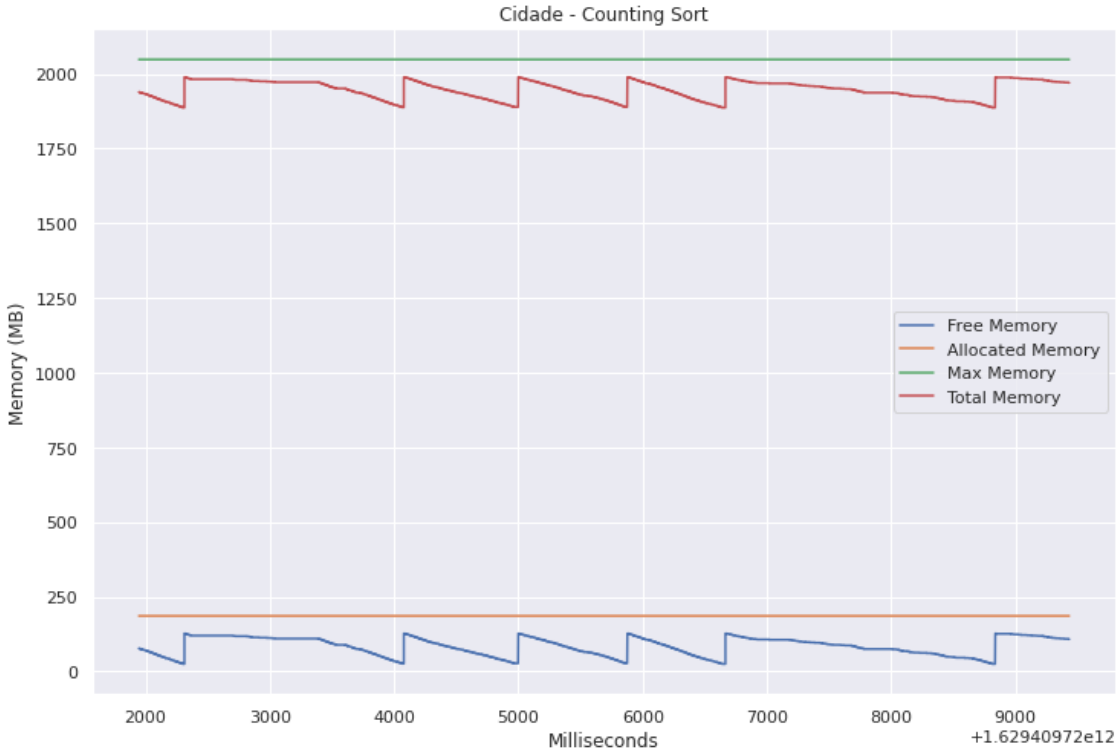

Gráfico 42: Cidade Counting Sort Máquina 2.

Resultados dos testes usando gráficos

Por fim, após a análise de todos os gráficos gerados nas métricas do sistema, podemos comparar os resultados da execução entre as duas máquinas e dos diferentes algoritmos de ordenação. Após a análise dos gráficos pode-se perceber que na máquina 1 o algoritmo mais rápido foi o Heap Sort e na máquina 2 o algoritmo mais rápido foi Insertion Sort, no entanto em casos gerais o algoritmo mais eficientes serão o Heap Sort e o Counting Sort, pois nas duas máquinas são o que mantém a média do menor consumo de memória. Vale salientar que os dados podem ter divergência devido à máquina e o número de programas rodando simultaneamente ao programa.

Em uma análise geral, concluímos que o Heap Sort é o algoritmo mais eficiente, apesar de na segunda máquina ter demorado um pouco mais, o consumo de memória foi reduzido. Já na primeira máquina realmente o Heap Sort foi o mais eficiente, tanto no consumo de memória como na velocidade de processamento.