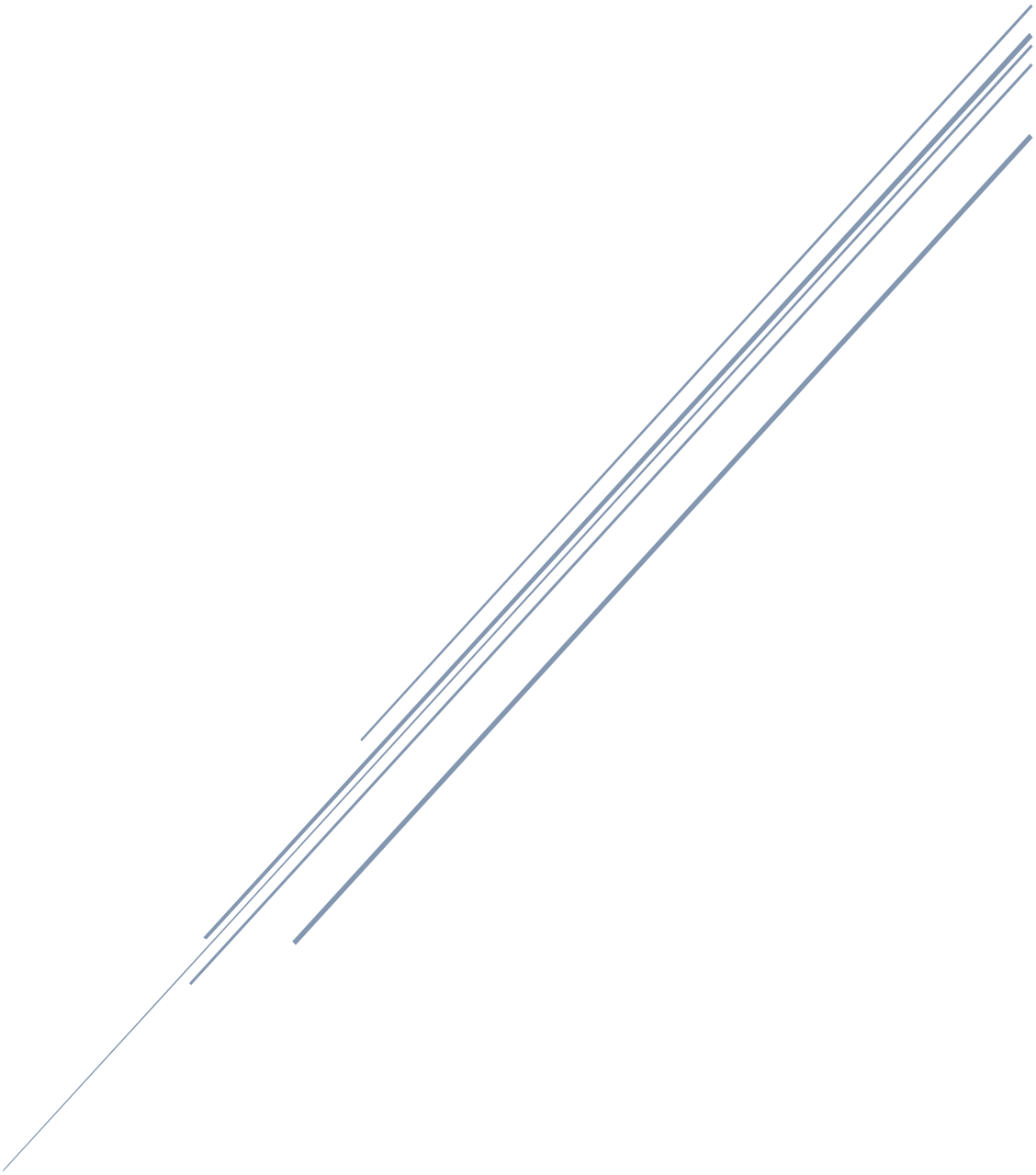
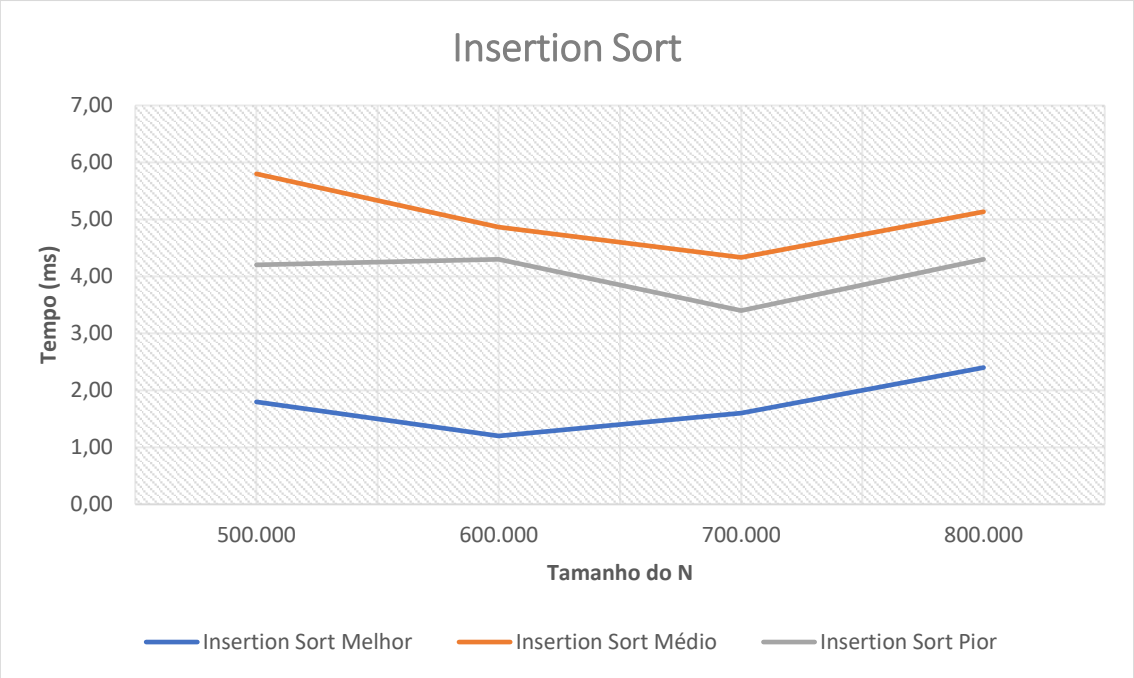
ANÁLISE DE ALGORITMOS

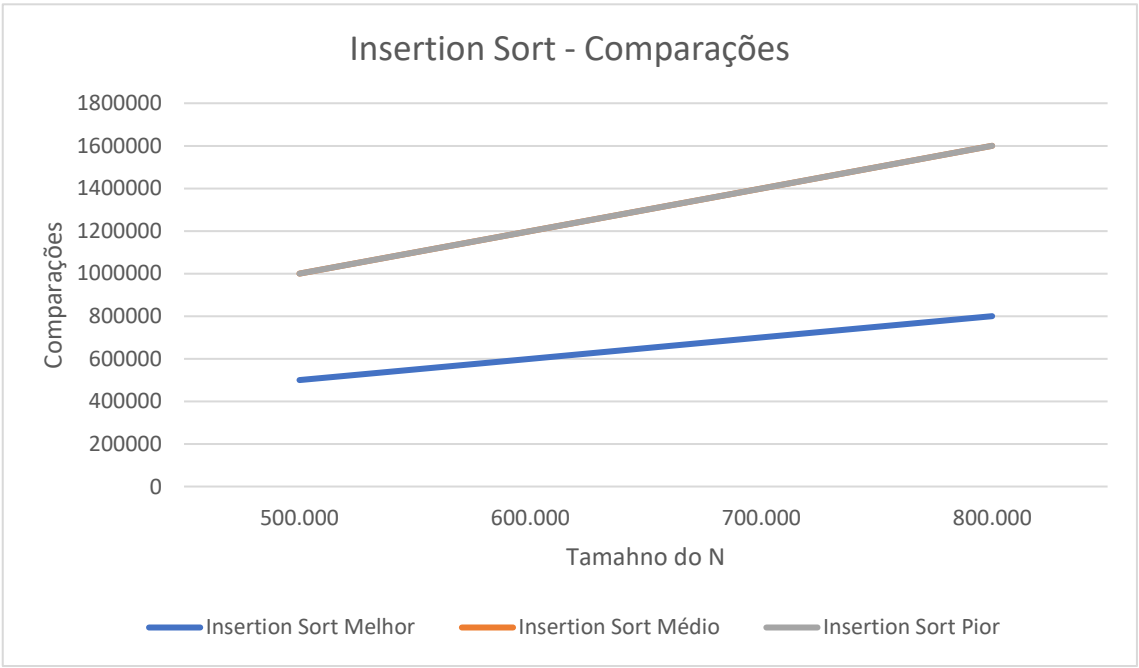
Brian Bruno Emanuel Brandão Silva

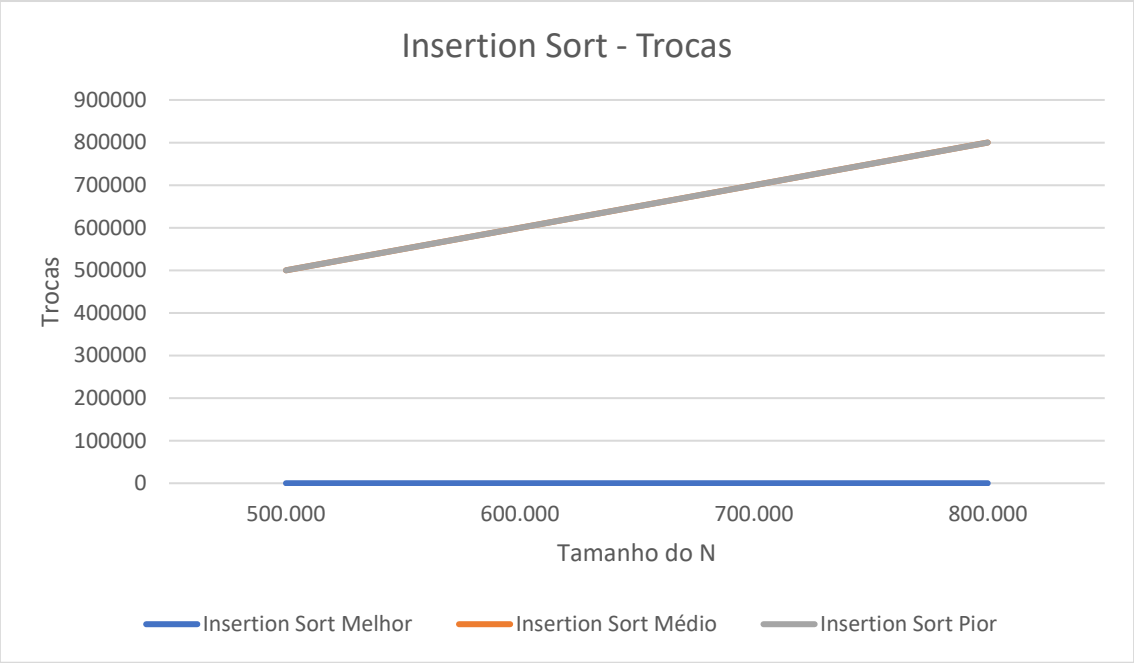


PUC MINAS
Engenharia de Software

Insertion Sort

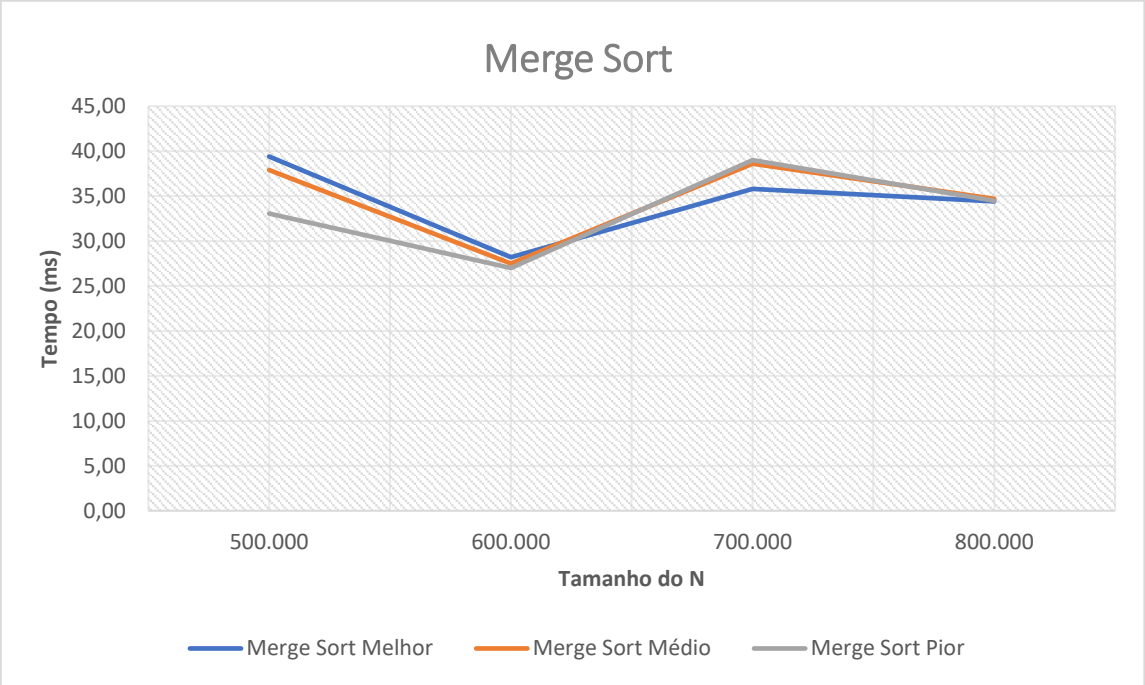


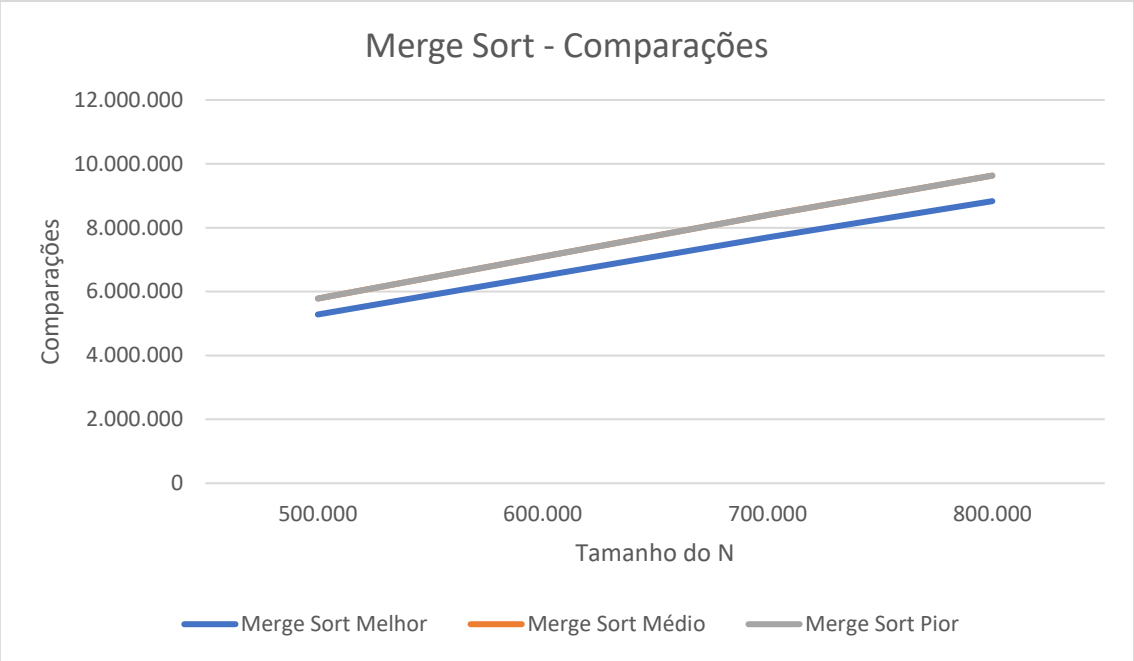


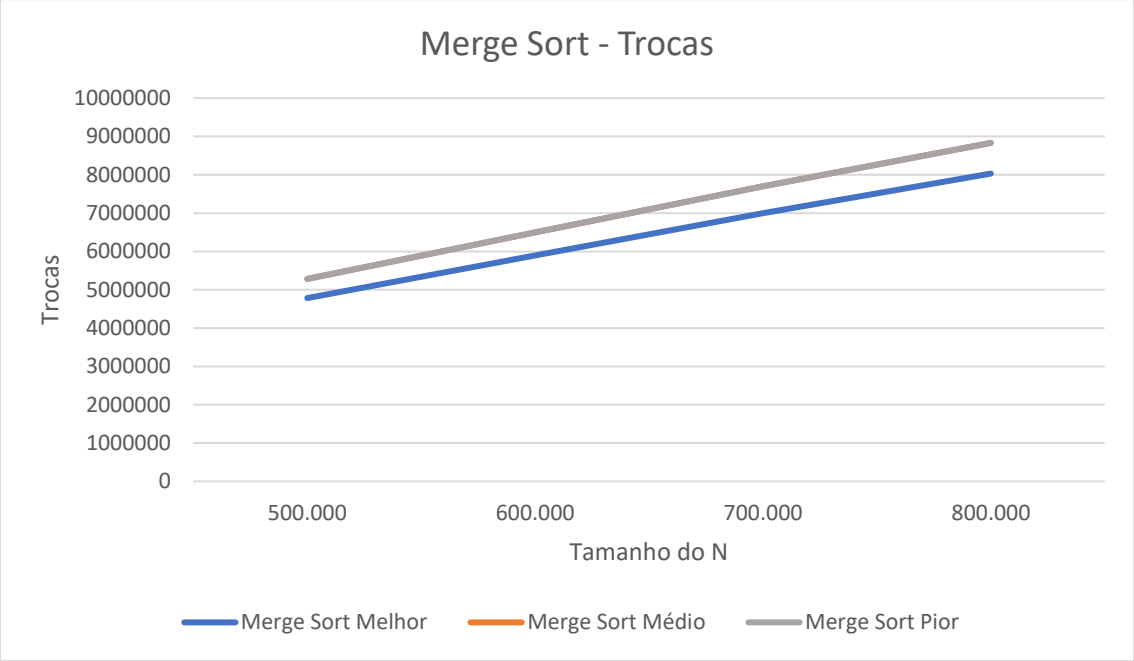


Nesse algoritmo aconteceu algo interessante: o tempo médio no caso médio foi superior ao tempo no pior caso. Com isso, pode se concluir que esse algoritmo não é tão eficiente assim para quando estiver em “Melhor Caso”. No caso das comparações, no caso médio e no pior caso tiveram a mesma quantidade.

Merge Sort

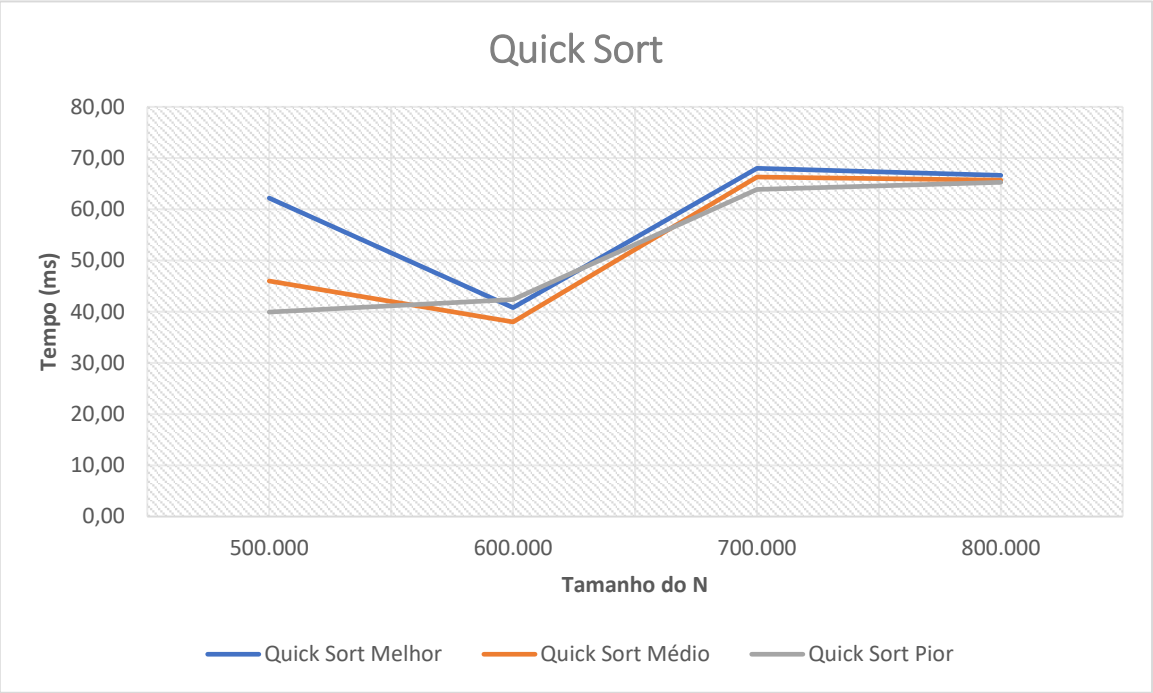


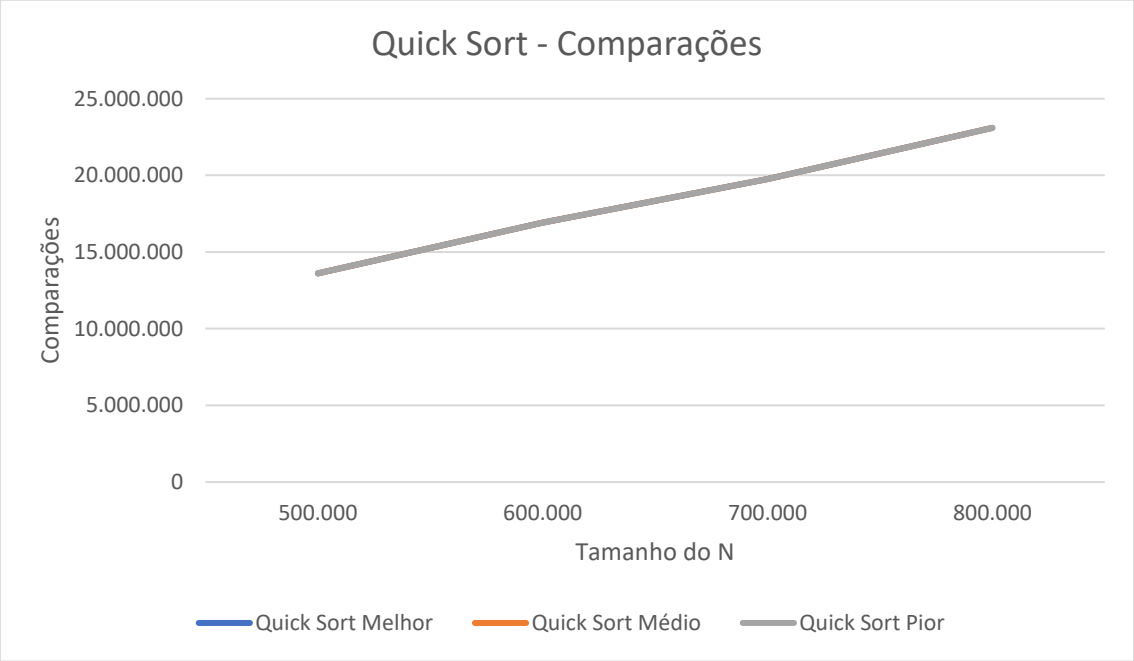


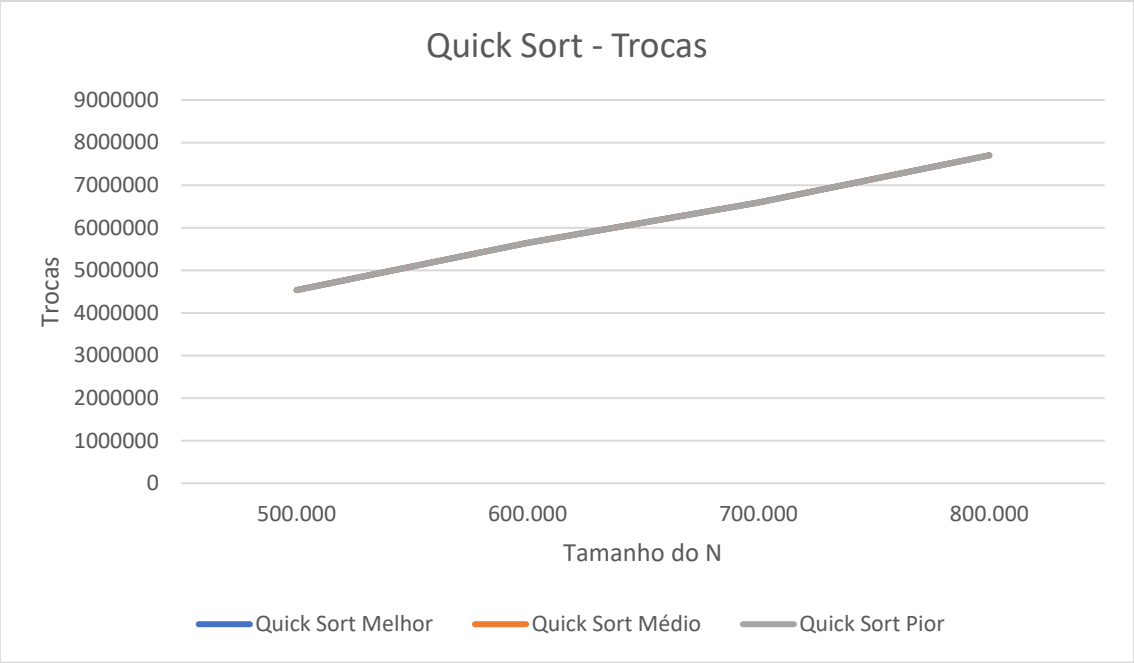


Nesse caso, o N aumentou, mas o tempo não aumentou tanto assim. Entretanto, os tempos foram parecidos. Entretanto, o tempo foi o igual, mesmo com um N cerca de 50% maior. No caso das comparações, no merge sort, o caso médio se equiparou com o pior caso.

Quick Sort

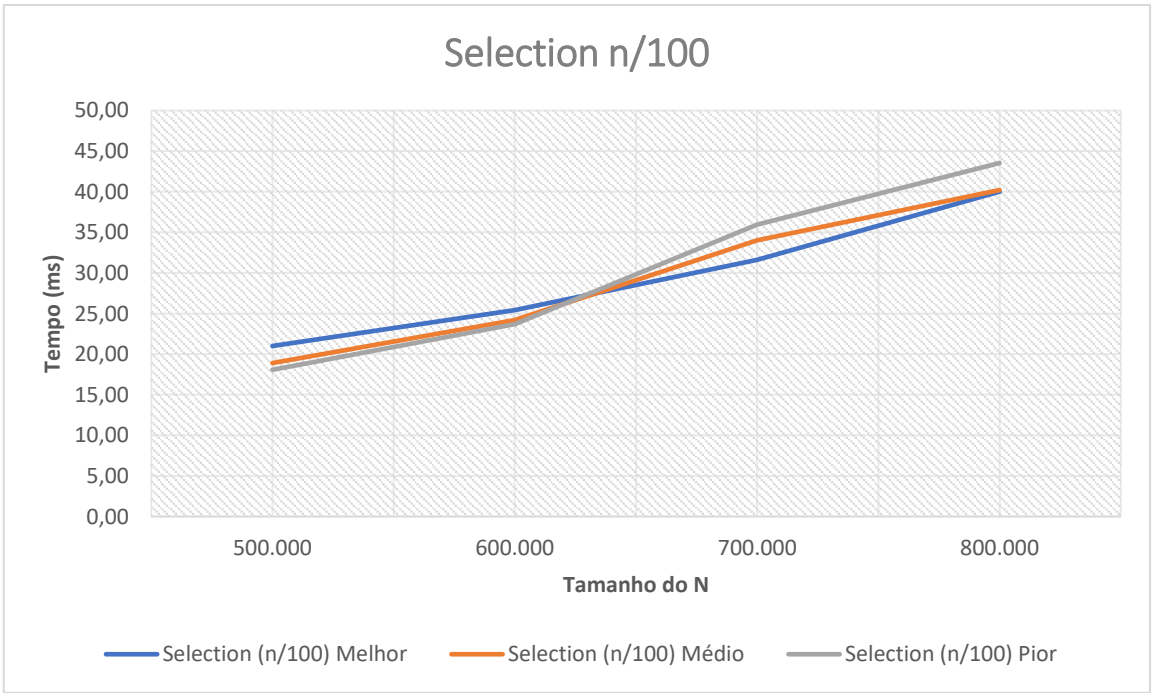

Quick Sort
80,00
70,00
60,00
50,00
40,00
30,00
20,00
10,00
0,00
Tempo (ms)
500.000
600.000
700.000
800.000
Tamanho do N
Quick Sort Melhor
Quick Sort Médio
Quick Sort Pior

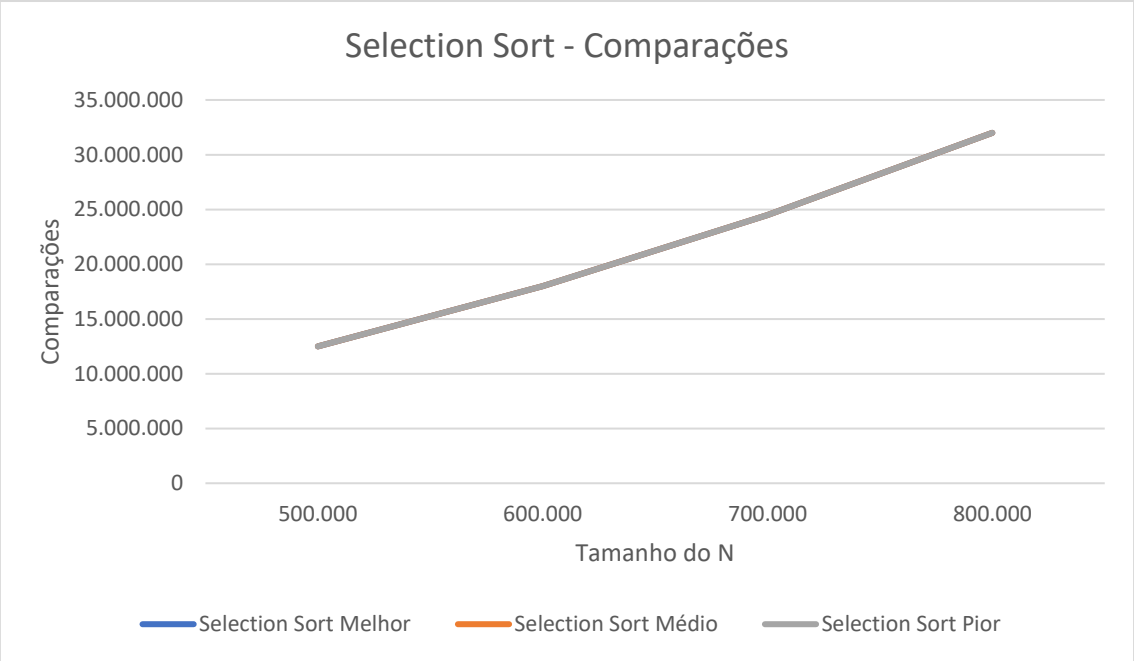

Quick Sort - Comparações
25.000.000
20.000.000
15.000.000
10.000.000
5.000.000
0
Comparações
500.000
600.000
700.000
800.000
Tamanho do N
Quick Sort Melhor
Quick Sort Médio
Quick Sort Pior

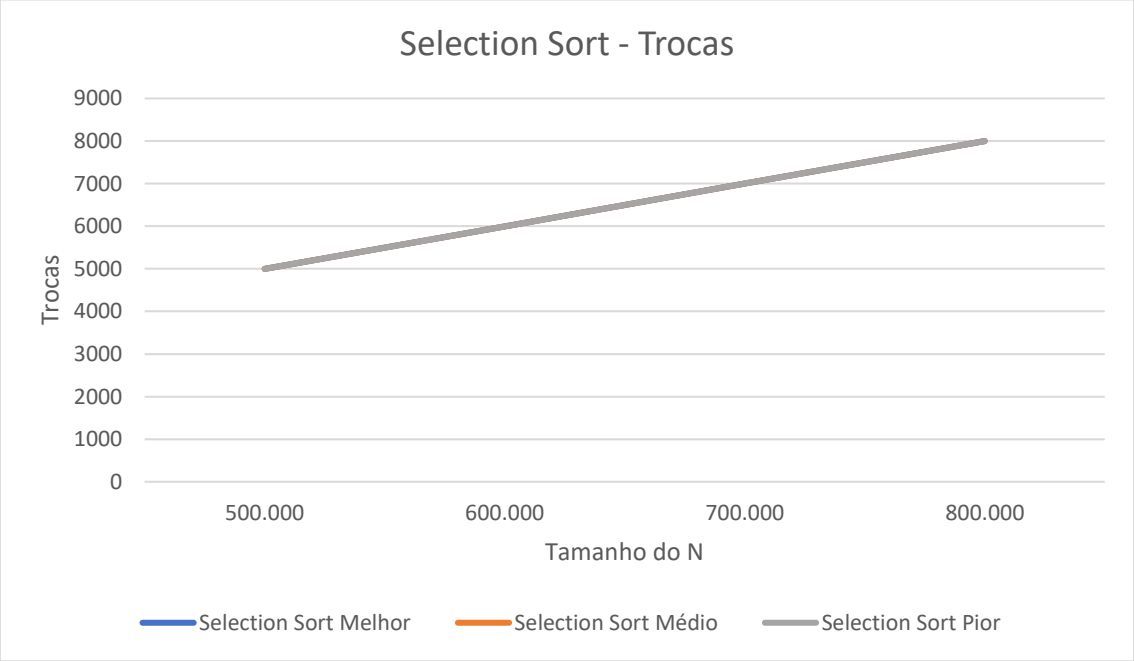


Nesse caso, o pior caso está mais rápido do que o melhor caso para N = 500.000. Isso é um pouco estranho, pois esperava-se que o pior caso fosse mais lento. Nos gráficos de comparações e trocas é possível perceber que os valores estão muito próximos (nem sempre coincidem).

Selection Sort

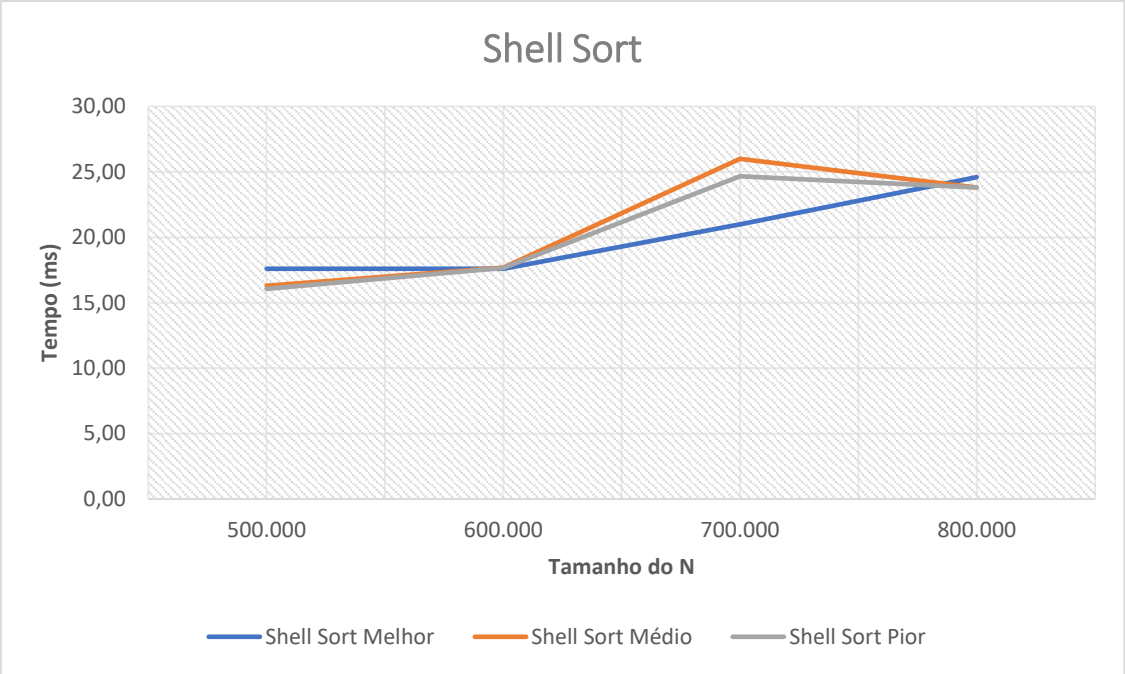


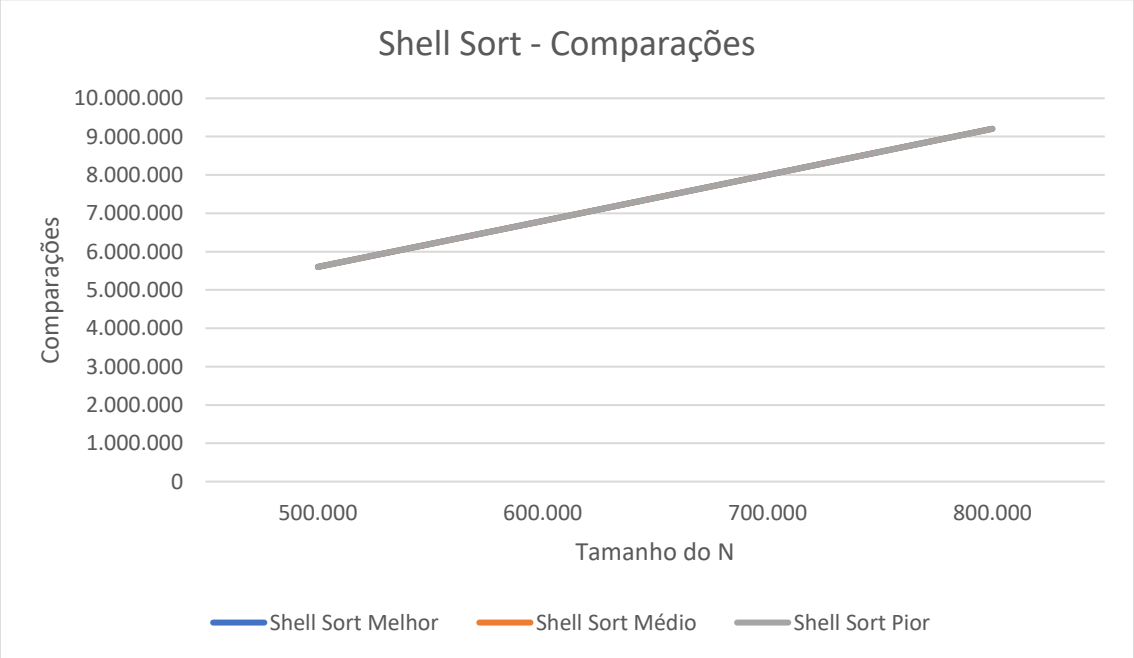


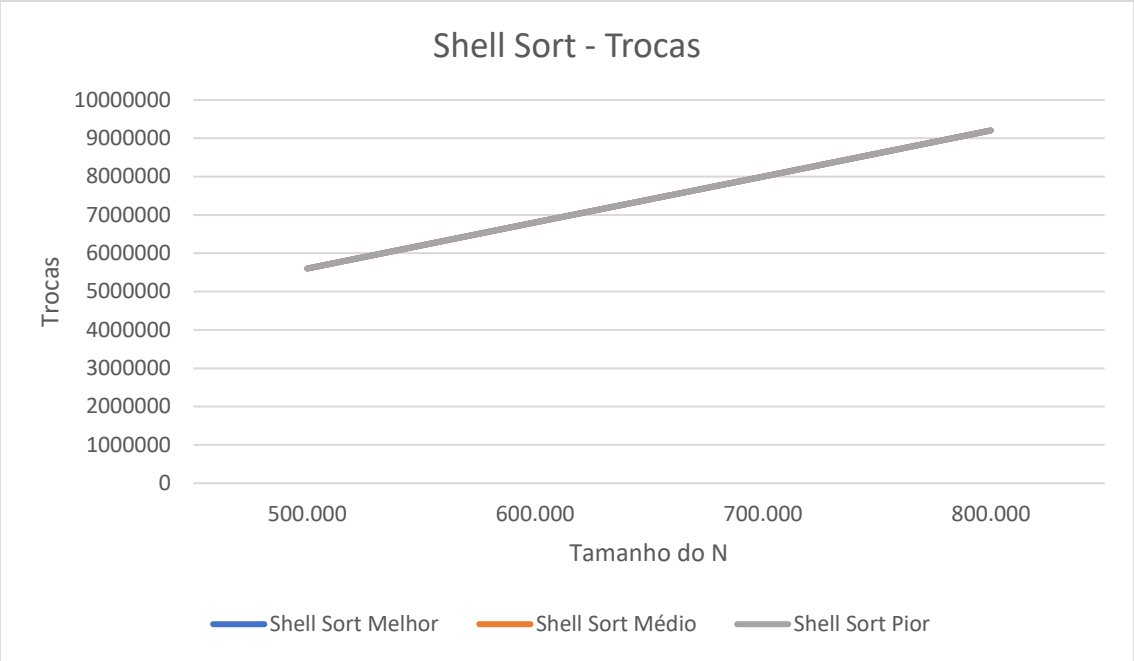


Nesse caso, o valor de n é n/100. Isso foi necessário pois esse algoritmo é extremamente lento. Nesse caso, para um N inferior a 700.000, o pior caso foi o mais rápido. Após 700.00 foi normalizado. Para o valor de trocas e comparações, o valor foi o mesmo em todos os casos.

Shell Sort

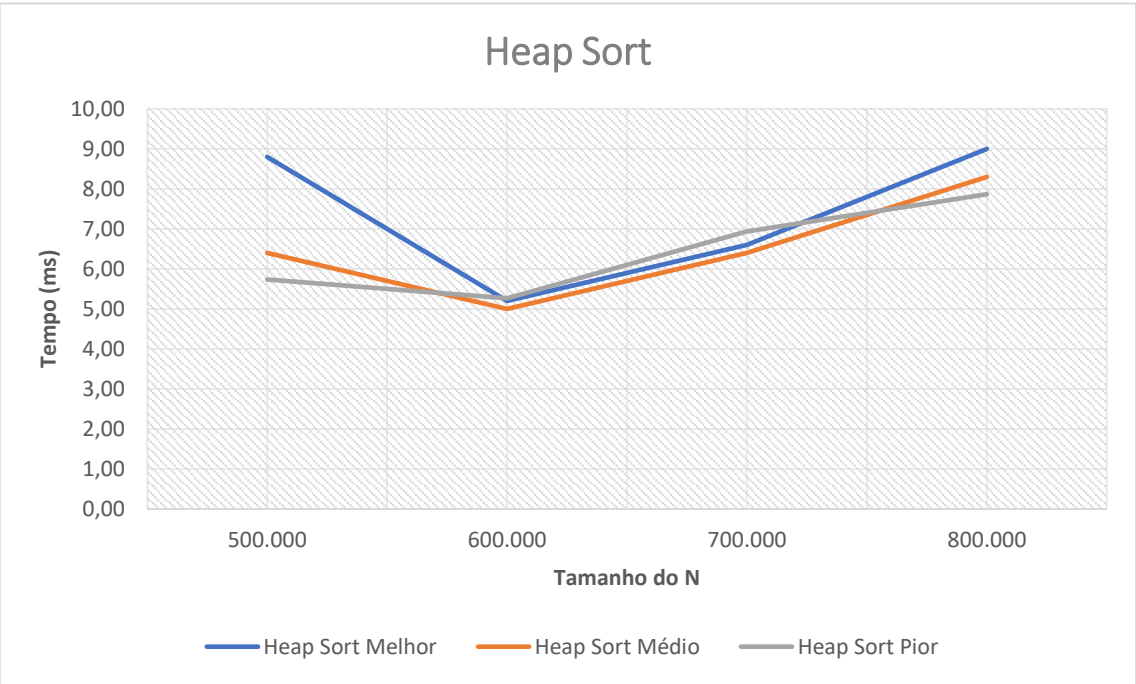

Shell Sort
30,00
25,00
20,00
15,00
10,00
5,00
0,00
Tempo (ms)
500.000
600.000
700.000
800.000
Tamanho do N
Shell Sort Melhor
Shell Sort Médio
Shell Sort Pior

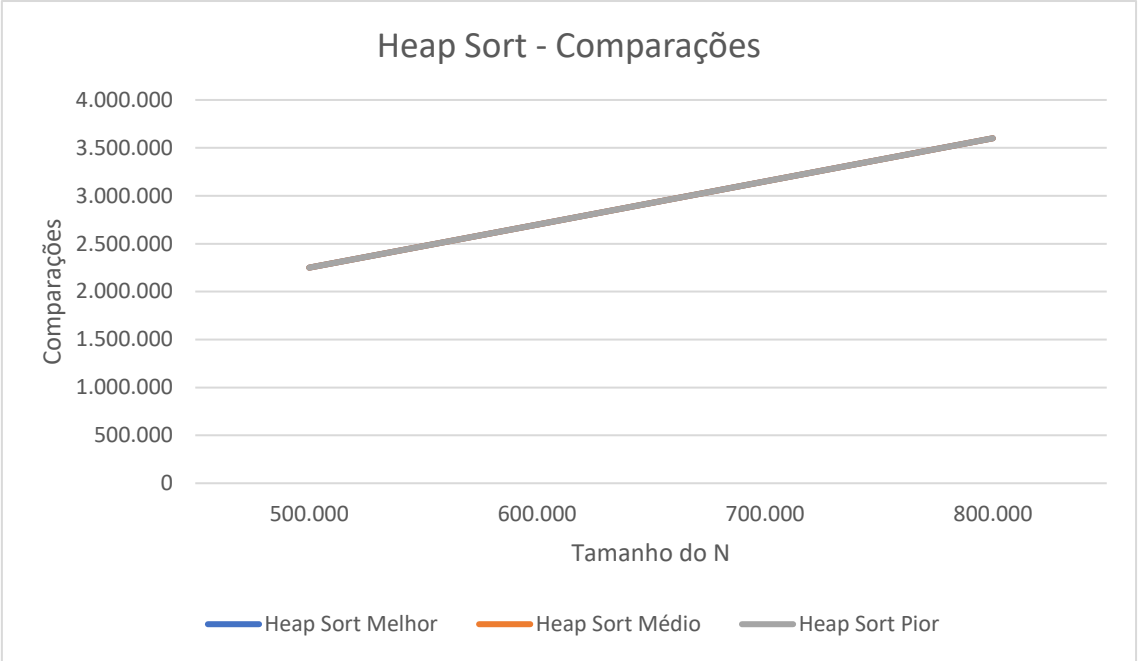

Shell Sort - Comparações
10.000.000
9.000.000
8.000.000
7.000.000
6.000.000
5.000.000
4.000.000
3.000.000
2.000.000
1.000.000
0
Comparações
500.000
600.000
700.000
800.000
Tamanho do N
Shell Sort Melhor
Shell Sort Médio
Shell Sort Pior

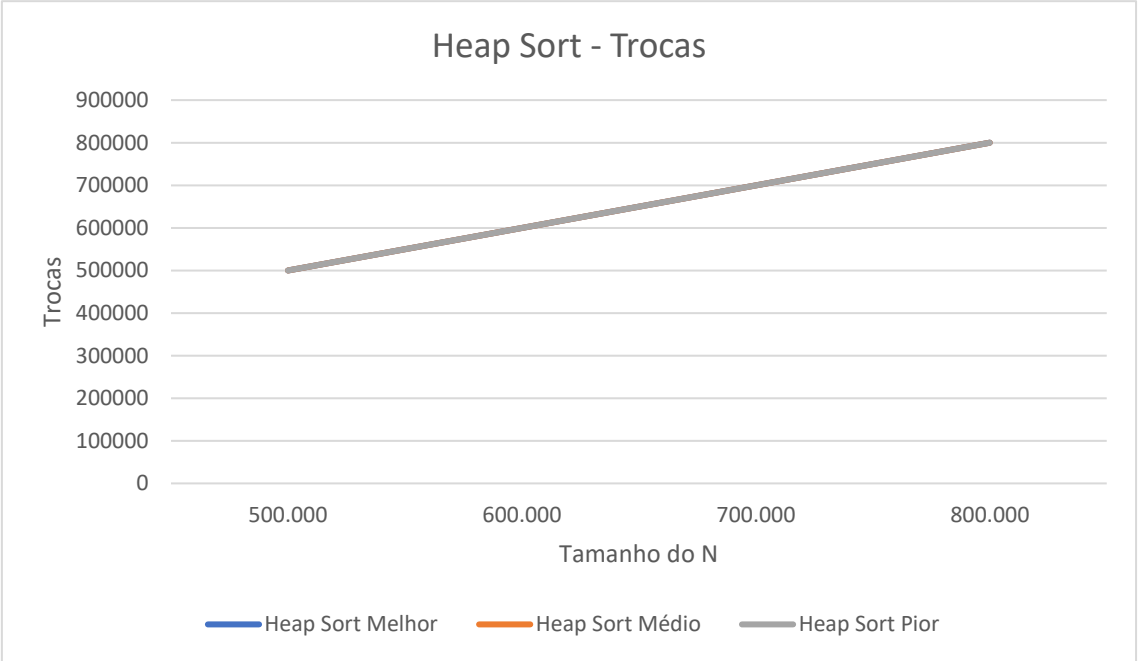


Nesse caso, os tempos não foram tão variantes. Com N = 700.000, o tempo do caso médio foi mais elevado do que os outros casos. Os casos de trocas e comparações foram bastante parecidos em todos os casos.

Heap Sort

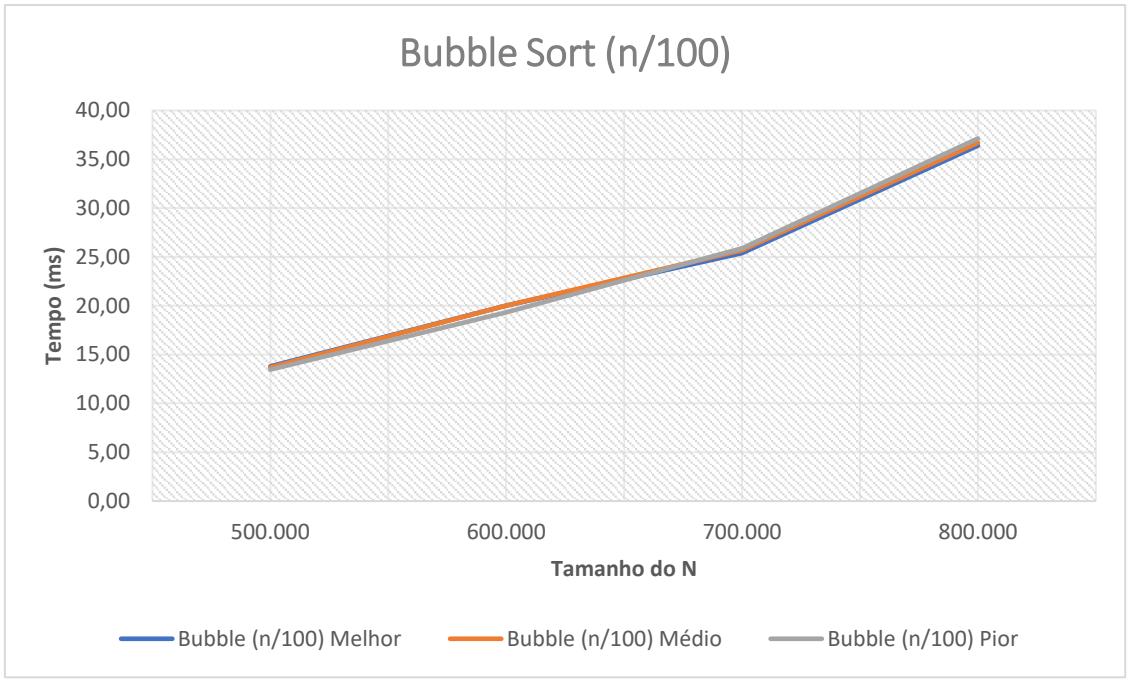


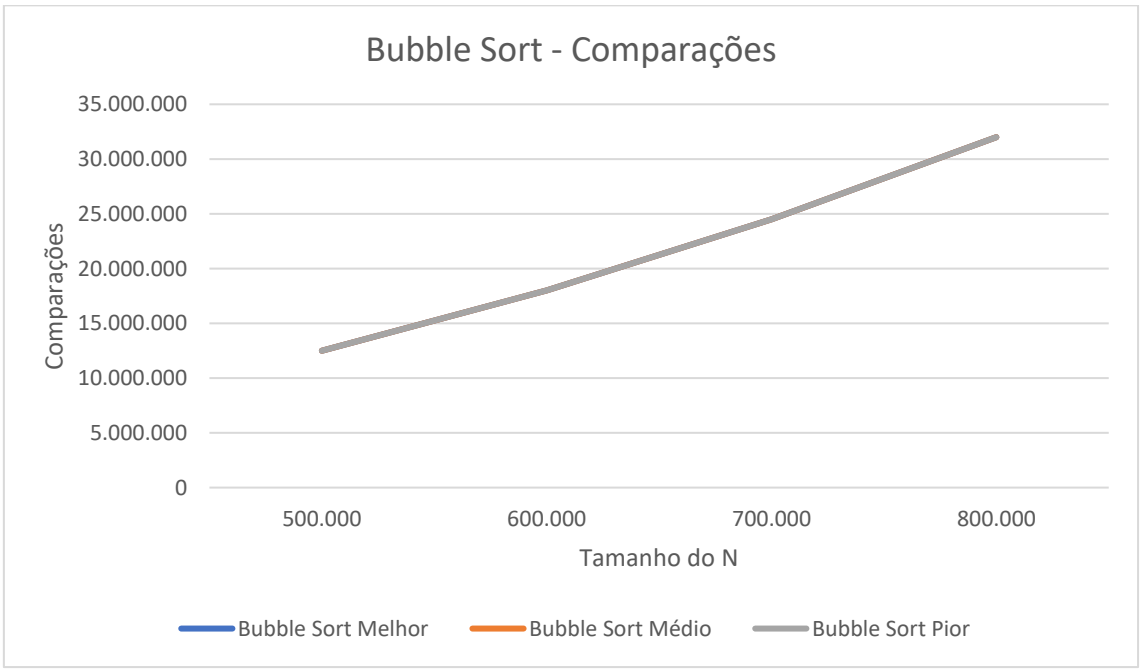


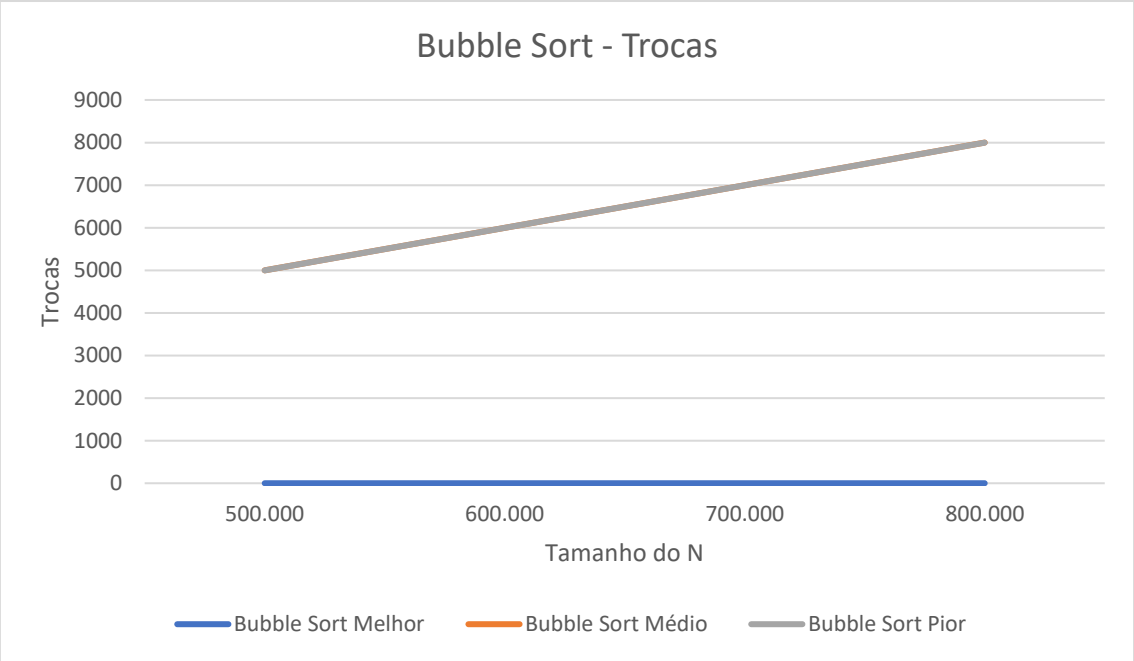


No Heap Sort, os tempos diminuíram de 500.000 para 600.000. Entretanto, após esse N os valores do tempo aumentaram. Para N = 800.000, o pior caso foi o mais rápido.

Bubble Sort

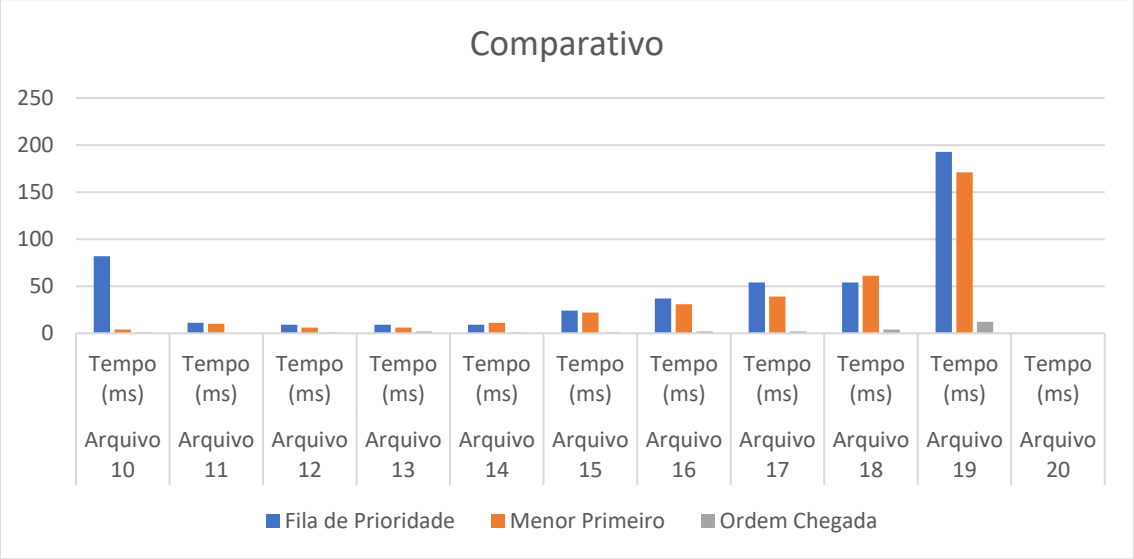

Bubble Sort (n/100)
40,00
35,00
30,00
25,00
20,00
15,00
10,00
5,00
0,00
Tempo (ms)
500.000
600.000
700.000
800.000
Tamanho do N
Bubble (n/100) Melhor
Bubble (n/100) Médio
Bubble (n/100) Pior


Bubble Sort - Comparações
35.000.000
30.000.000
25.000.000
20.000.000
15.000.000
10.000.000
5.000.000
0
Comparações
500.000
600.000
700.000
800.000
Tamanho do N
Bubble Sort Melhor
Bubble Sort Médio
Bubble Sort Pior

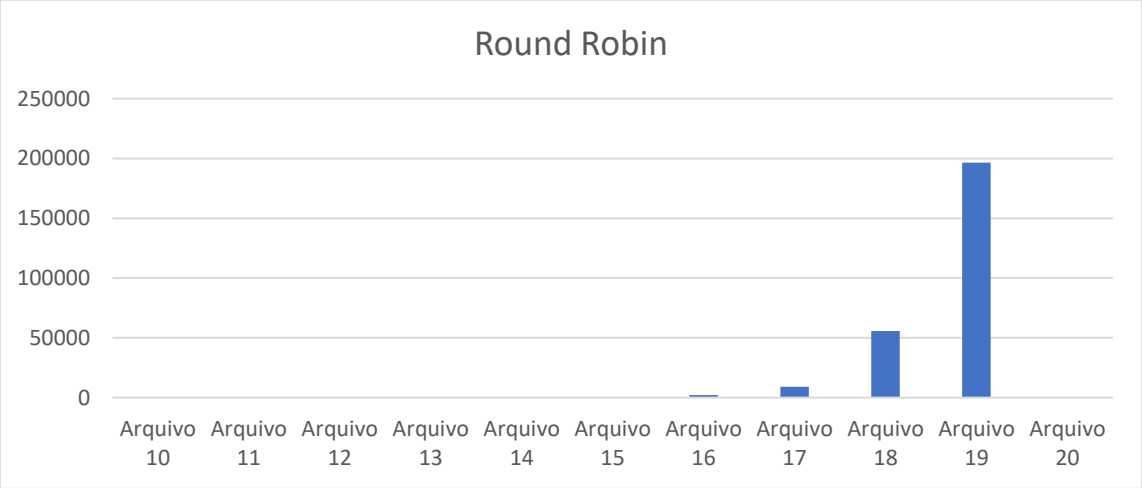


No Bubble Sort foi necessário dividir o N por 100 para que fosse possível realizar a contabilização do tempo. Neste algoritmo, o tempo foi muito parecido em todos os N registrados. Aumentaram de acordo com o tamanho do N, mas não de forma quadrática, como é previsto na teoria. Nesse caso, no melhor caso, apresentou um número de trocas igual a zero. Entretanto, o número de comparações permanece muito alto em comparação com outros gráficos.

Parte II



Nesse comparativo, sem o Round Robin, é possível perceber uma pequena anomalia no início (Arquivo 10) Fila de Prioridade, que foi corrigida logo em seguida. Os tempos foram aumentando gradativamente de acordo com o tamanho do arquivo. O tempo do Ordem de Chegada permaneceu o mínimo, pois nesse método não há cálculos ou contas apuradas para constatar qual será o próximo a ser executado. Quem chegou primeiro, vai primeiro. Nos outros, em que existem regras para a execução, quanto maior o tamanho do arquivo, maior o tempo de processamento.



No Round Robin, que teve que ficar separado pois os tempos eram muito superiores, é observado comportamento semelhante ao comportamento anterior. Quanto maior o arquivo, maior o tempo de processamento. Entretanto, nesse método, como constatado na aula de Sistemas Operacionais, o tempo médio de Resposta e de Retorno são menores. Porém, devido a troca de contexto, o tempo de execução se torna maior.

Devido ao fato de o arquivo ser demasiadamente grande, não foi possível executar o arquivo 20 em tempo hábil. Portanto, em ambos os casos ele está zerado.

O código do projeto que foi realizado pode ser encontrado em:

https://github.com/brianbruno/puc-lab03-atv1