Laboratório de Estrutura de Dados

Primeira versão do projeto da disciplina

Comparação entre os algoritmos de ordenação elementar

Lucas de Lucena Siqueira (201080354), Daniel Xavier Brito de Araújo (201080303), Ian Vasconcelos Bernardo (201080044)

1. Introdução

Este relatório corresponde ao relato dos resultados obtidos no projeto da disciplina de LEDA que tem foco no site AirBNB. Responsável por promover acomodações, sendo uma plataforma disruptiva no setor desde o lançamento. É apresentado um dataset que contém informações gerais como nome do dono, vizinhança, latitude, longitude, tipo do lugar, preço, reviews, etc...

O projeto exige transformações de data, filtro por melhores valores tanto acima quanto abaixo da média. Todos gerando arquivos listings.

Também foi proposto o desenvolvimento de ordenações. Todas funcionando para médio, pior e melhor casos. Os arquivos para ordenação são:

- listagens pelo nomes dos lugares (campo name) em ordem alfabética;
- listagens pelos preços (campo price) dos lugares do menor para o maior;
- listagens pelos número de reviews (campo number_of_reviews) dos lugares do menor para o maior;
- listagens pela data do último review (campo last_review) dos lugares do mais recente para o mais antigo.

Cada algoritmo foi testado com a mesma base de dados em iguais condições. Quanto às ferramentas utilizadas, a implementação foi feita a partir da linguagem JAVA, e para monitorar o consumo de memória o VisualVM. Foi possível notar que alguns algoritmos são mais rápidos que outros. Onde até mesmo no melhor caso, o tempo de execução é maior. Outro ponto observado foi que os algoritmos que usam variável auxiliar consomem mais memória.

Link para acessar o desenvolvimento do projeto:
https://github.com/lucaslucenak/airBnb-dataAnalyses-springBoot

2. Descrição geral sobre o método utilizado

Há três classes de gráficos, na primeira comparamos o consumo de memória em todos os algoritmos para os três referenciais usados: Preços, quantidade de reviews e nomes. Após ver o consumo de memória para todos os citados, na segunda classe, analisamos cada algoritmo individualmente, comparando melhor, pior e médio caso. Ou seja, temos o Insertion Sort, por exemplo, sendo usado para os nomes, quantidades de reviews e preços comparando todos os casos. Na terceira classe, comparamos todos os algoritmos tomando como referencial de comparação todos os melhores casos, em seguida, todos os piores e enfim todos os médios.

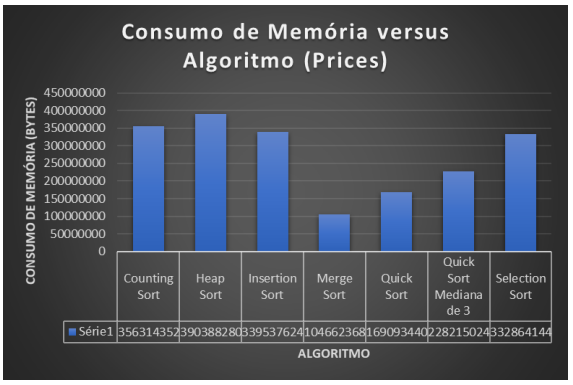
Descrição geral do ambiente de testes

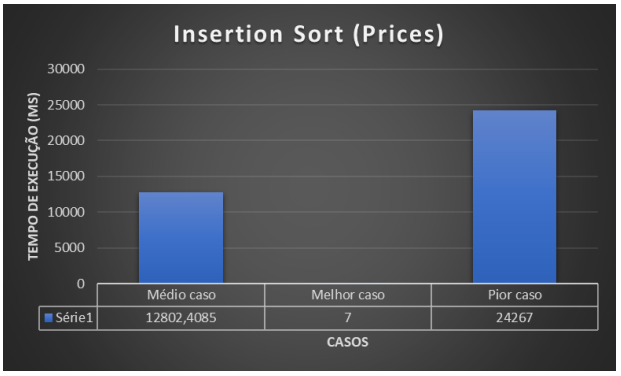
Ambiente de testes consiste em:

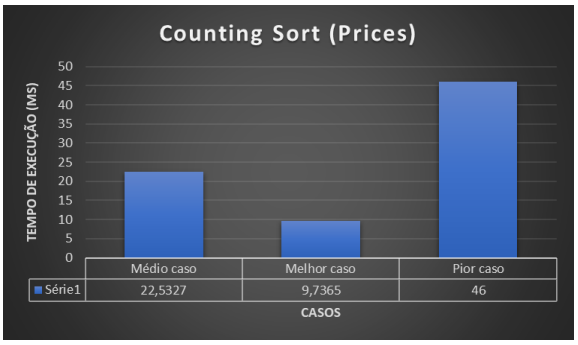
- Processador Ryzen 5 1600x (6 núcleos físicos e 12 threads)
- Placa de vídeo GTX 1050ti 4GB GDDR5
- Memória RAM 16GB DDR4 2800Mhz
- Windows 10

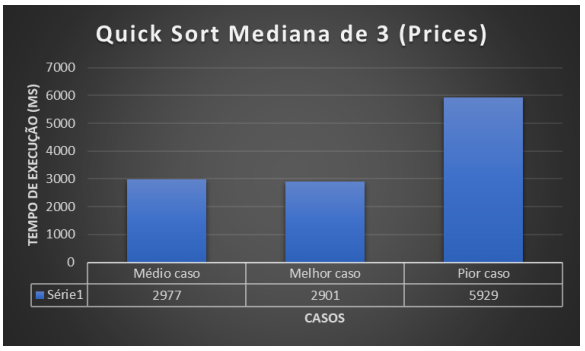
3. Resultados e Análise

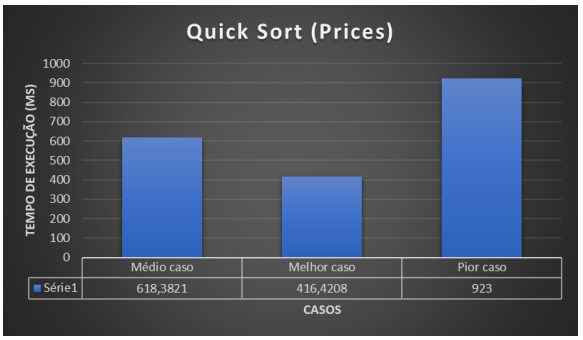
Resultados dos testes usando tabelas e gráficos

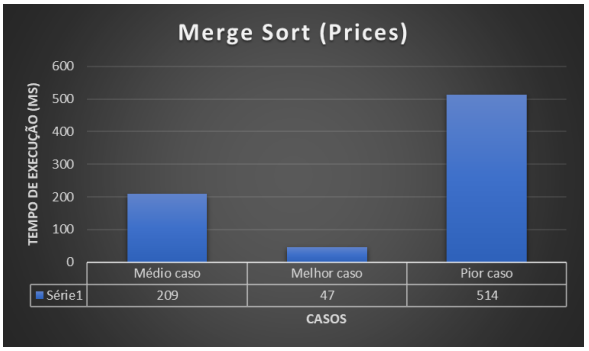


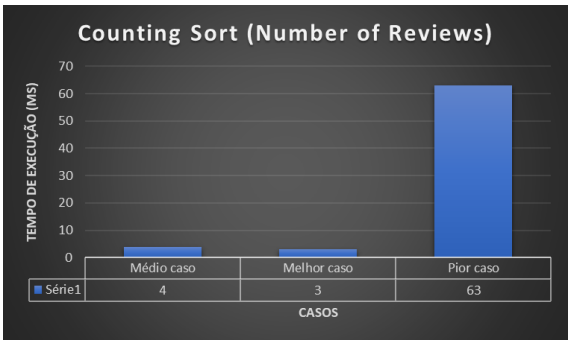


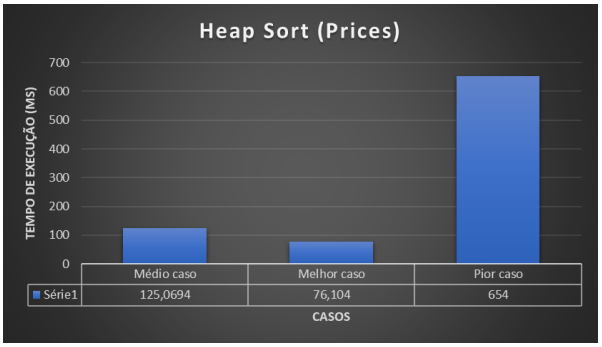




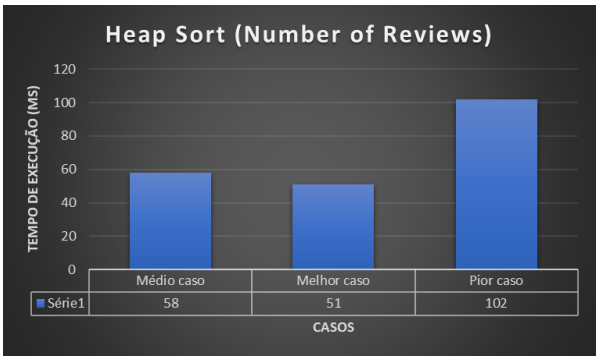


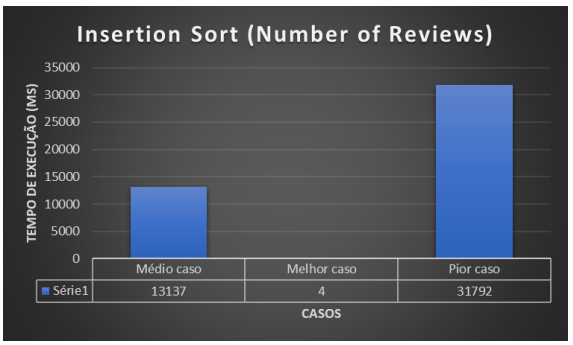


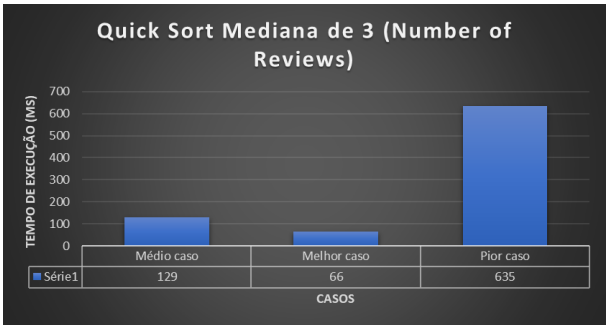

Counting Sort (Number of Reviews)
TEMPO DE EXECUÇÃO (MS)
70
60
50
40
30
20
10
0
Médio caso
Melhor caso
Pior caso
Série1
4
3
63
CASOS

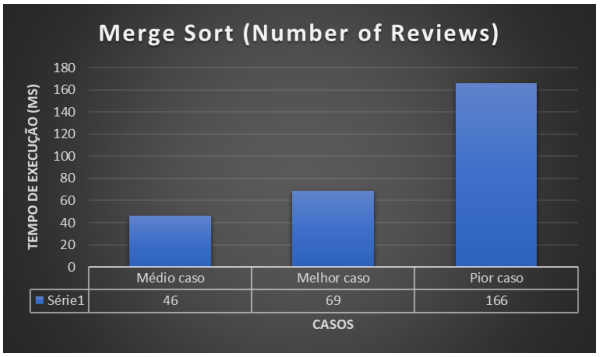

Heap Sort (Prices)
TEMPO DE EXECUÇÃO (MS)
700
600
500
400
300
200
100
0
Médio caso
Melhor caso
Pior caso
Série1
125,0694
76,104
654
CASOS

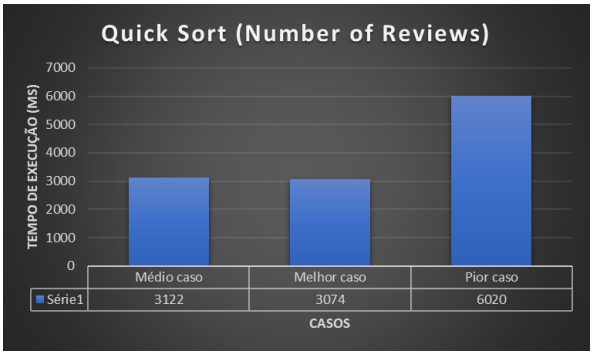

Selection Sort (Prices)
TEMPO DE EXECUÇÃO (MS)
1400
1200
1000
800
600
400
200
0
Médio caso
Melhor caso
Pior caso
Série1
604,1377
645,4517
1301
CASOS

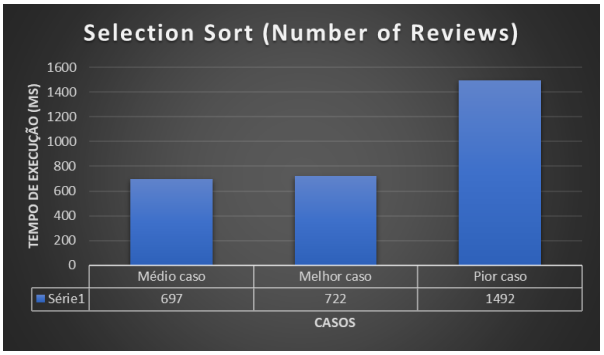

Heap Sort (Number of Reviews)
TEMPO DE EXECUÇÃO (MS)
120
100
80
60
40
20
0
Médio caso
Melhor caso
Pior caso
Série1
58
51
102
CASOS

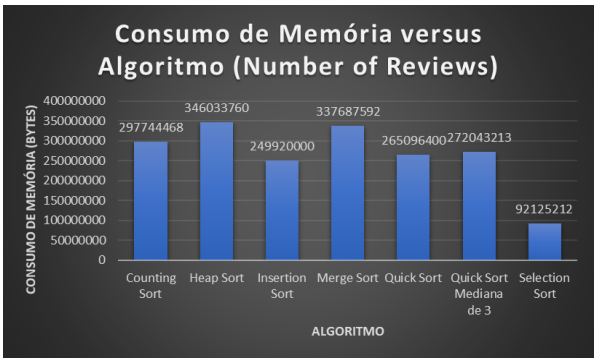

Insertion Sort (Number of Reviews)
TEMPO DE EXECUÇÃO (MS)
35000
30000
25000
20000
15000
10000
5000
0
Médio caso
Melhor caso
Pior caso
Série1
13137
4
31792
CASOS

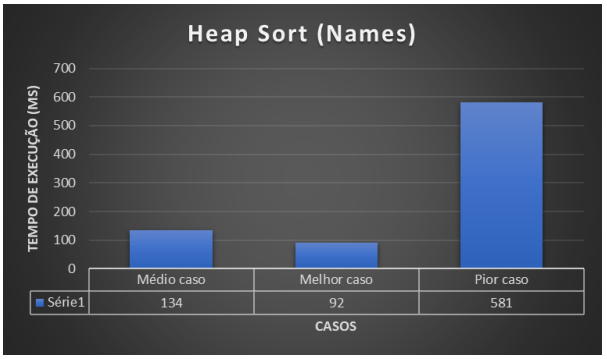

Quick Sort Mediana de 3 (Number of Reviews)
TEMPO DE EXECUÇÃO (MS)
700
600
500
400
300
200
100
0
Médio caso
Melhor caso
Pior caso
Série1
129
66
635
CASOS

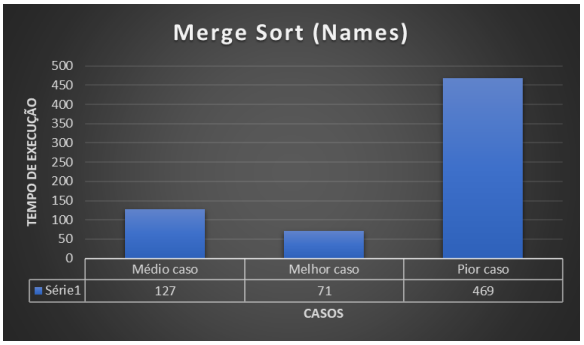

Merge Sort (Number of Reviews)
TEMPO DE EXECUÇÃO (MS)
180
160
140
120
100
80
60
40
20
0
Médio caso
Melhor caso
Pior caso
Série1
46
69
166
CASOS

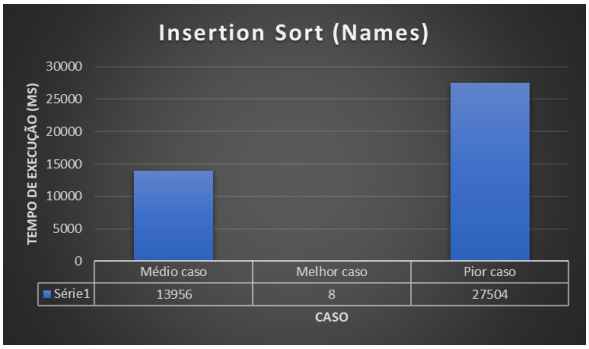

Quick Sort (Number of Reviews)
TEMPO DE EXECUÇÃO (MS)
7000
6000
5000
4000
3000
2000
1000
0
Médio caso
Melhor caso
Pior caso
Série1
3122
3074
6020
CASOS

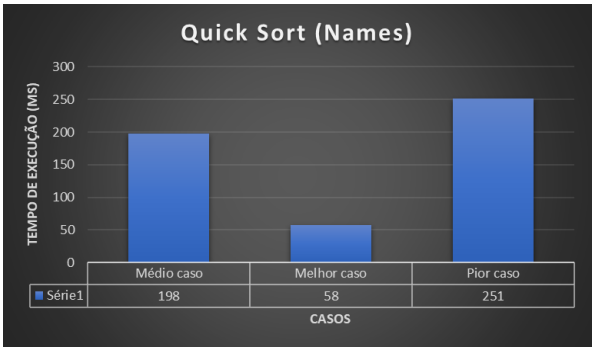

Selection Sort (Number of Reviews)
TEMPO DE EXECUÇÃO (MS)
1600
1400
1200
1000
800
600
400
200
0
Médio caso
Melhor caso
Pior caso
Série1
697
722
1492
CASOS

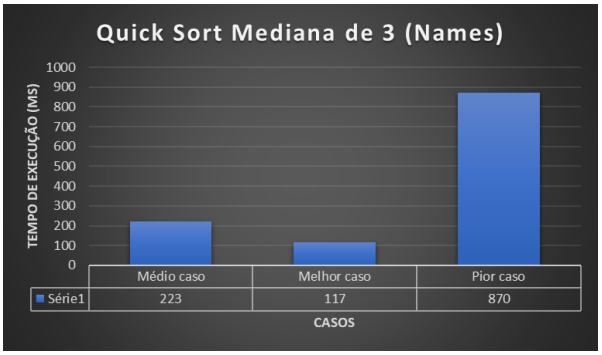

Consumo de Memória versus Algoritmo (Number of Reviews)
CONSUMO DE MEMÓRIA (BYTES)
400000000
350000000
300000000
250000000
200000000
150000000
100000000
50000000
0
297744468
346033760
249920000
337687592
265096400
272043213
92125212
Counting Sort
Heap Sort
Insertion Sort
Merge Sort
Quick Sort
Quick Sort Mediana de 3
Selection Sort
ALGORITMO

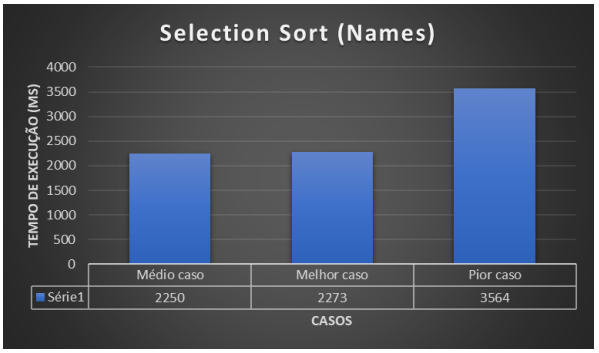

Heap Sort (Names)
TEMPO DE EXECUÇÃO (MS)
700
600
500
400
300
200
100
0
Médio caso
Melhor caso
Pior caso
Série1
134
92
581
CASOS

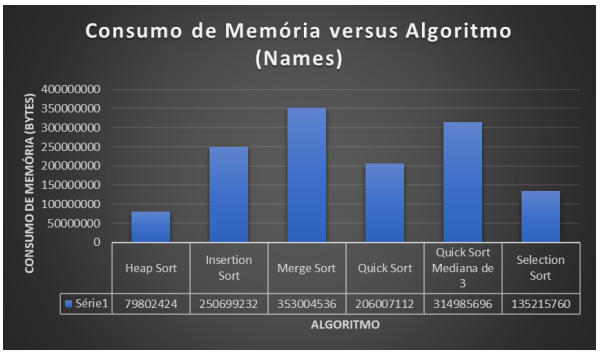

Merge Sort (Names)
TEMPO DE EXECUÇÃO
500
450
400
350
300
250
200
150
100
50
0
Médio caso
Melhor caso
Pior caso
Série1
127
71
469
CASOS

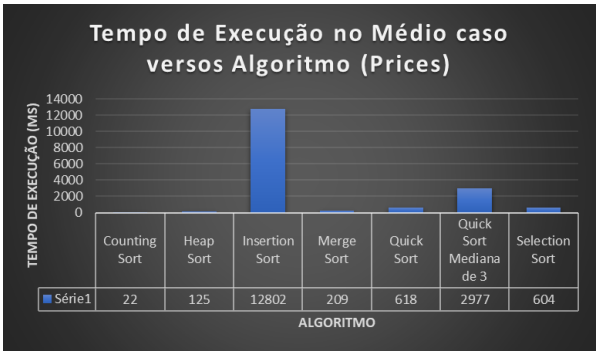

Insertion Sort (Names)
TEMPO DE EXECUÇÃO (MS)
30000
25000
20000
15000
10000
5000
0
Médio caso
Melhor caso
Pior caso
Série1
13956
8
27504
CASO

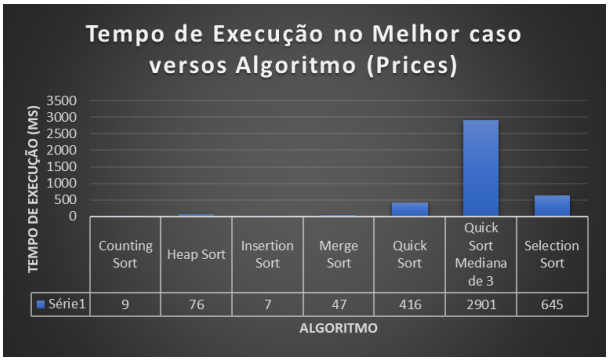

Quick Sort (Names)
TEMPO DE EXECUÇÃO (MS)
300
250
200
150
100
50
0
Médio caso
Melhor caso
Pior caso
Série1
198
58
251
CASOS

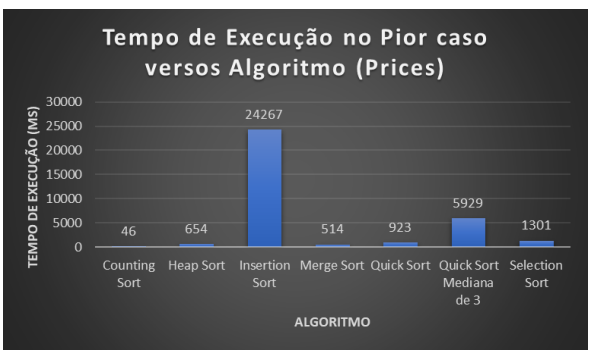

Quick Sort Mediana de 3 (Names)
TEMPO DE EXECUÇÃO (MS)
1000
900
800
700
600
500
400
300
200
100
0
Médio caso
Melhor caso
Pior caso
Série1
223
117
870
CASOS

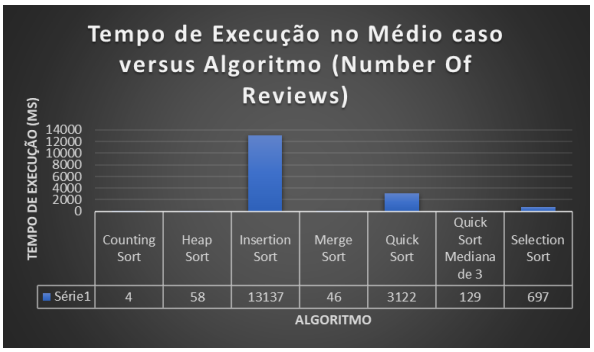

Selection Sort (Names)
TEMPO DE EXECUÇÃO (MS)
4000
3500
3000
2500
2000
1500
1000
500
0
Médio caso
Melhor caso
Pior caso
Série1
2250
2273
3564
CASOS

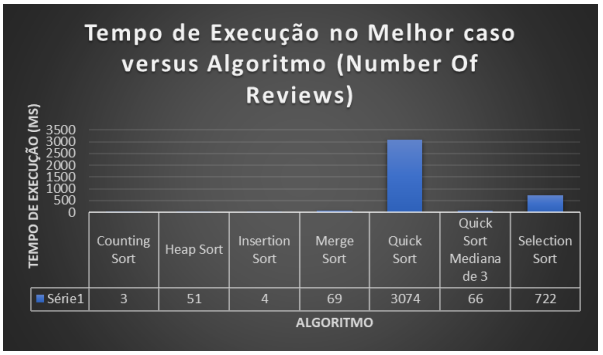

Consumo de Memória versus Algoritmo (Names)
CONSUMO DE MEMÓRIA (BYTES)
400000000
350000000
300000000
250000000
200000000
150000000
100000000
50000000
0
Heap Sort
Insertion Sort
Merge Sort
Quick Sort
Quick Sort Mediana de 3
Selection Sort
Série1
79802424
250699232
353004536
206007112
314985696
135215760
ALGORITMO

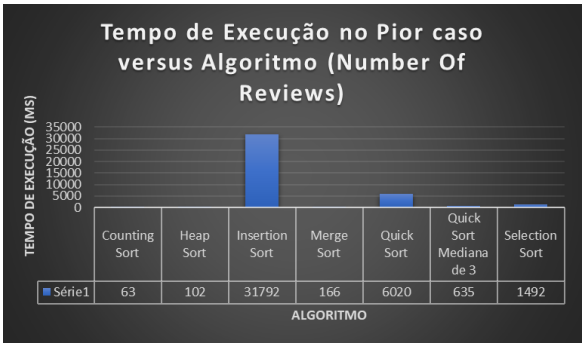

Tempo de Execução no Médio caso versos Algoritmo (Prices)
TEMPO DE EXECUÇÃO (MS)
14000
12000
10000
8000
6000
4000
2000
0
Counting Sort
Heap Sort
Insertion Sort
Merge Sort
Quick Sort
Quick Sort Mediana de 3
Selection Sort
Série1
22
125
12802
209
618
2977
604
ALGORITMO

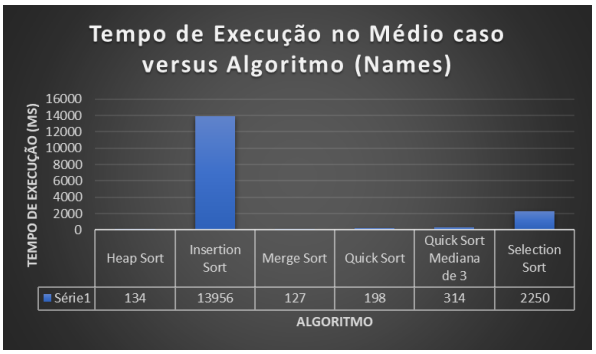

Tempo de Execução no Melhor caso versos Algoritmo (Prices)
TEMPO DE EXECUÇÃO (MS)
3500
3000
2500
2000
1500
1000
500
0
Counting Sort
Heap Sort
Insertion Sort
Merge Sort
Quick Sort
Quick Sort Mediana de 3
Selection Sort
Série1
9
76
7
47
416
2901
645
ALGORITMO

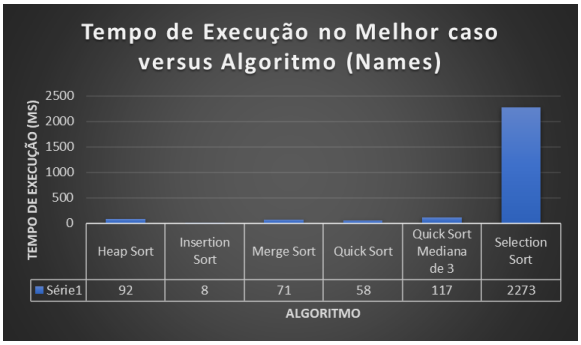

Tempo de Execução no Pior caso versos Algoritmo (Prices)
TEMPO DE EXECUÇÃO (MS)
30000
25000
20000
15000
10000
5000
0
46
654
24267
514
923
5929
1301
Counting Sort
Heap Sort
Insertion Sort
Merge Sort
Quick Sort
Quick Sort Mediana de 3
Selection Sort
ALGORITMO

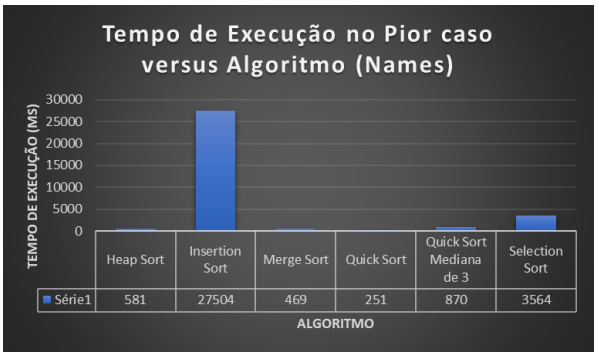












Observando o quesito de consumo de memória, o Mergesort se saiu melhor entre os métodos testados, mostrando ser o mais eficiente, visto que o tempo de execução não fica muito diferente da média geral.

Em geral, analisando os resultados apresentados nas tabelas acima, a maioria dos algoritmos tem o resultado de comparação parecidos. Melhor e médio casos sempre com tempo de execução significativamente menor que o pior caso.