

# **Primeira versão do projeto da disciplina**

## **Comparação entre os algoritmos de ordenação elementar**

---

**UNIVERSIDADE ESTADUAL DA PARAÍBA**

**CENTRO DE CIÊNCIAS E TECNOLOGIA**

**CIÊNCIA DA COMPUTAÇÃO**

**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, avaliações, dentre outros atributos.

Foram realizadas algumas transformações e tratamento dos dados fornecidos no dataset, como a formatação da data para dd/MM/yyyy, e filtragem dos apartamentos que estavam acima ou abaixo do preço médio, sendo gerado para cada um desses tratamentos um arquivo .csv diferente (todos esses arquivos podem ser encontrados no diretório "airBnb-dataAnalyses\src\main\resources\csvFiles")

Também foi proposto o desenvolvimento de ordenações a partir do uso de alguns algoritmos de ordenação, sendo eles o Counting Sort, Heap Sort, Insertion Sort, Merge Sort, Quick Sort, Quick Sort Mediana de 3 e por fim o Selection Sort. Todas as ordenações tomaram como parâmetro para a execução no médio, melhor e pior caso os seguintes campos fornecidos no dataset:

- Listagens pelo nome dos lugares (campo name) em ordem alfabética;
- Listagens pelos preços (campo price) dos lugares do menor para o maior;
- Listagens pelo 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 e também nas mesmas condições de ambiente. Quanto às ferramentas utilizadas, a implementação foi feita a partir da linguagem JAVA e a fim de monitorar o consumo de memória foi utilizado o VisualVM.

Após a execução do projeto foi possível observar as diferenças presentes em cada um dos algoritmos de ordenação utilizados, tanto no consumo de memória como principalmente no tempo de execução. Todas as análises serão apresentadas no decorrer do relatório.

## 2. Descrição geral sobre o método utilizado

Para a realização dos procedimentos de teste, o projeto foi organizado em alguns passos bem definidos que são personificados para cada uma das ordenações. O primeiro passo foi utilizar o arquivo gerado anteriormente com o tratamento das datas ("listings\_review\_date.csv") para a instanciação dos dos apartamentos apresentados. Após os objetos terem sido instanciados o seguimento lógico utilizado foi referente à indicação dos processos de ordenação com a utilização de cada um dos algoritmos propostos e também dos parâmetros de ordenação definidos.

Para o processo de ordenação, os algoritmos foram escritos no package "ordenationAlgorithms". Com os algoritmos já definidos, foram criados métodos específicos para cada uma das ordenações, presentes no package "ordenationMethods", sendo que cada método tem a mesma lógica de funcionamento procedural, que seria definida nos seguintes passos:

- Ordenar os dados com o médio caso capturando o tempo de execução do procedimento e criando o arquivo .csv referente a essa ordenação.
- Ordenar os dados com o melhor caso capturando o tempo de execução do procedimento e criando o arquivo .csv referente a essa ordenação.
- Ordenar os dados com o pior caso capturando o tempo de execução do procedimento e criando o arquivo .csv referente a essa ordenação.

Vale ressaltar que todos os arquivos gerados se encontram no diretório "airBnb-dataAnalyses\src\main\resources\ordenatedCsvFiles".

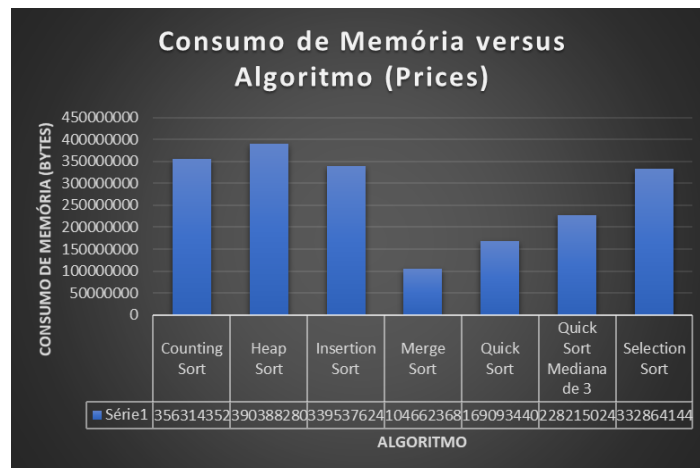
### Descrição do ambiente de testes

- Processador AMD Ryzen 5 1600x (6 núcleos e 12 threads)
- Placa de vídeo NVIDIA GTX 1050TI 4GB GDDR5
- Memória RAM 16GB DDR4 2800MHz (2x8)
- Sistema Operacional Windows 10 Pro

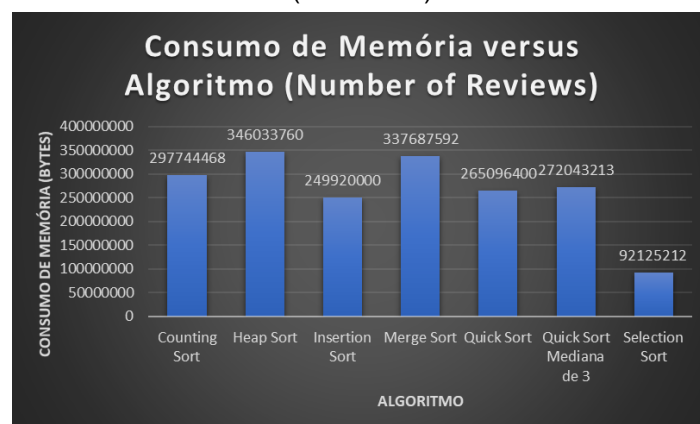
### 3. Resultados e Análise

#### Resultados dos testes usando gráficos

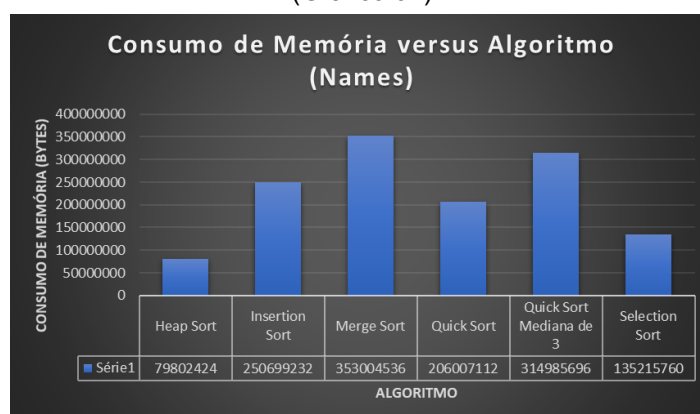
As primeiras análises são referentes ao consumo de memória de cada algoritmo tendo como base cada um dos parâmetros de ordenação. Segue abaixo os gráficos obtidos com o auxílio do VisualVM para monitorar o consumo de memória dos algoritmos.



(Gráfico 01)



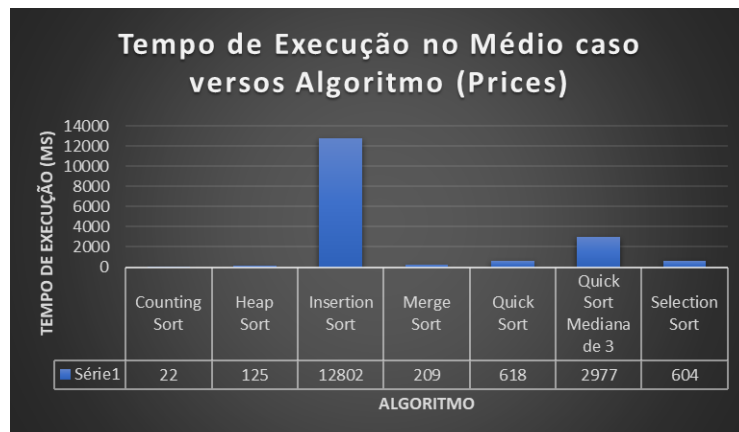
(Gráfico 02)



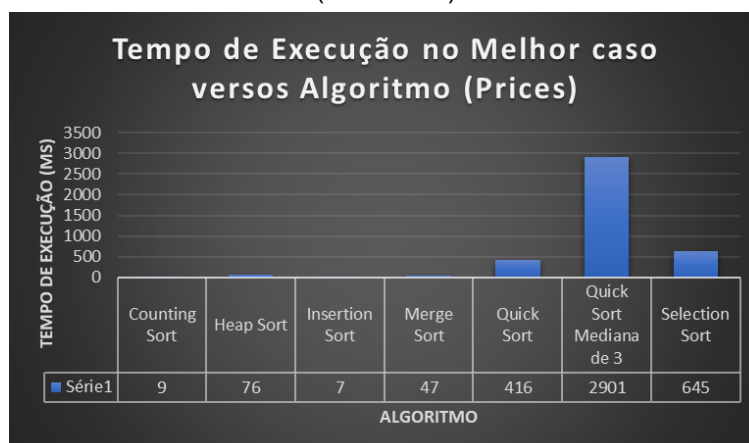
(Gráfico 03)

Já as análises a seguir se referem ao desempenho dos algoritmos de ordenação tendo em vista os tempos de execução de cada um para cada um dos parâmetros definidos previamente.

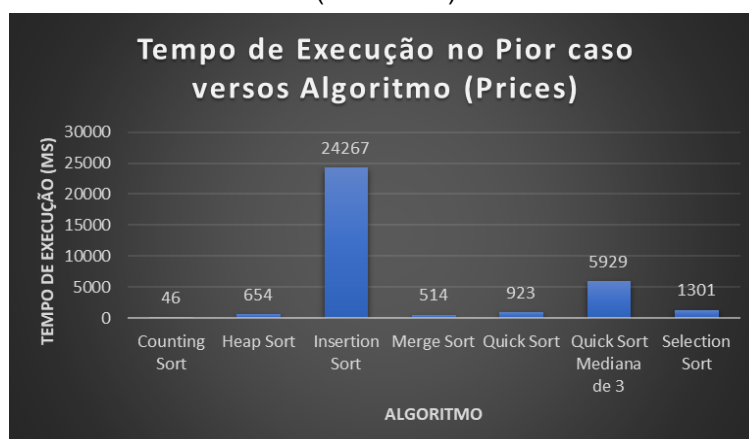
- **Parâmetro de ordenação: Prices**



(Gráfico 04)

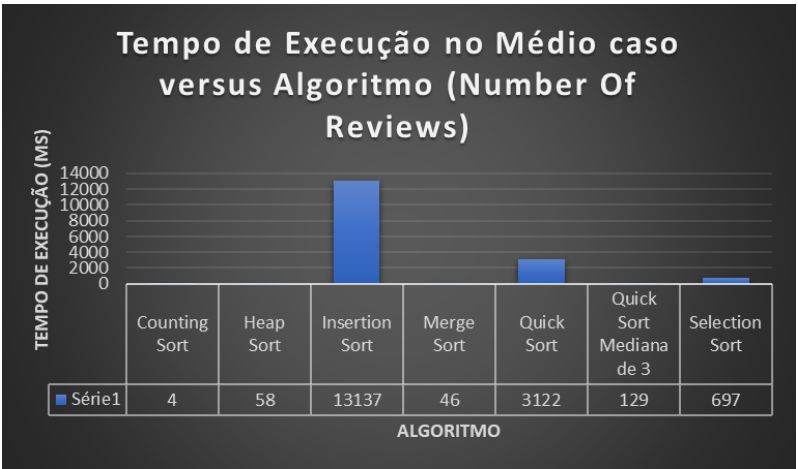


(Gráfico 05)

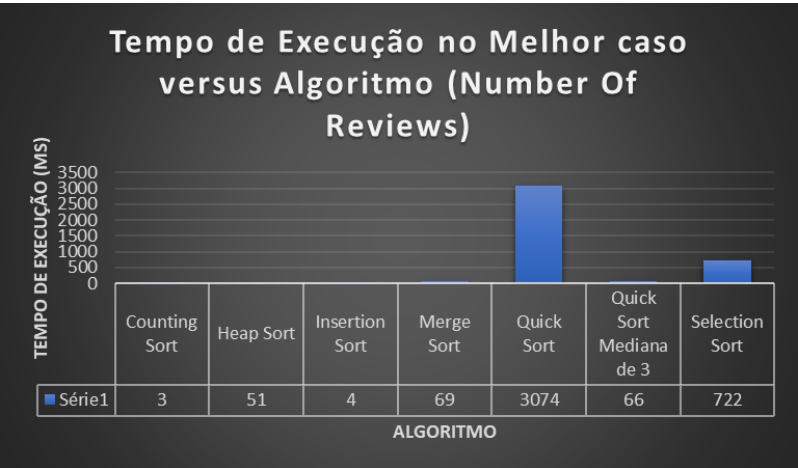


(Gráfico 06)

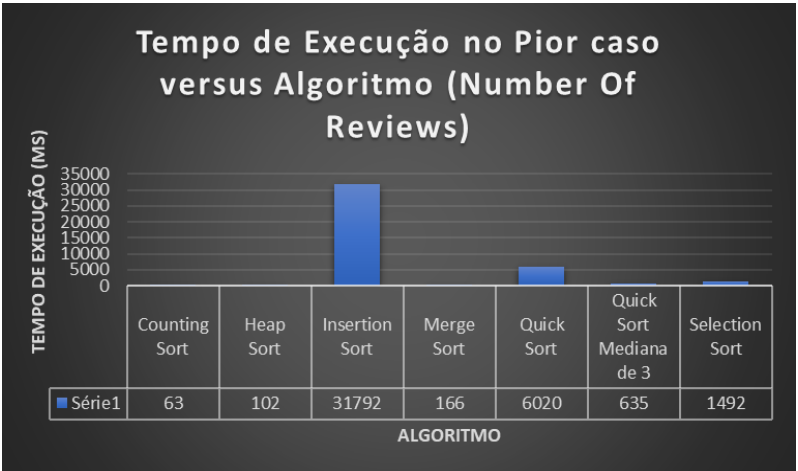
- **Parâmetro de ordenação: Number of Reviews**



(Gráfico 07)

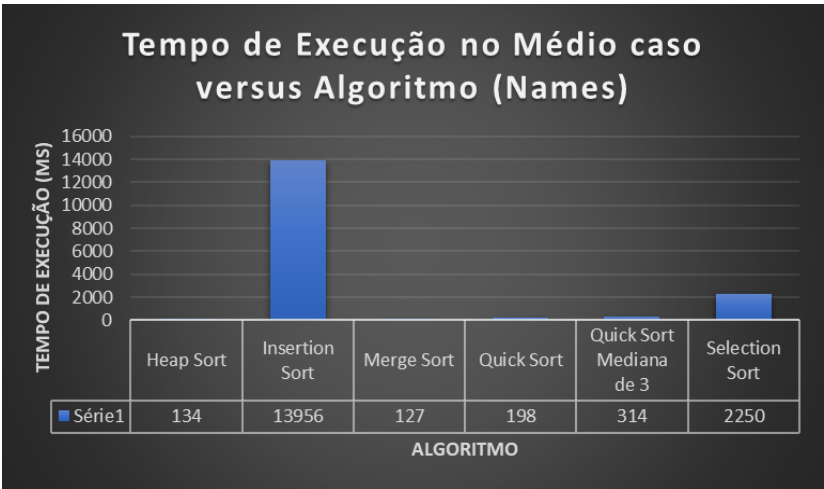


(Gráfico 08)

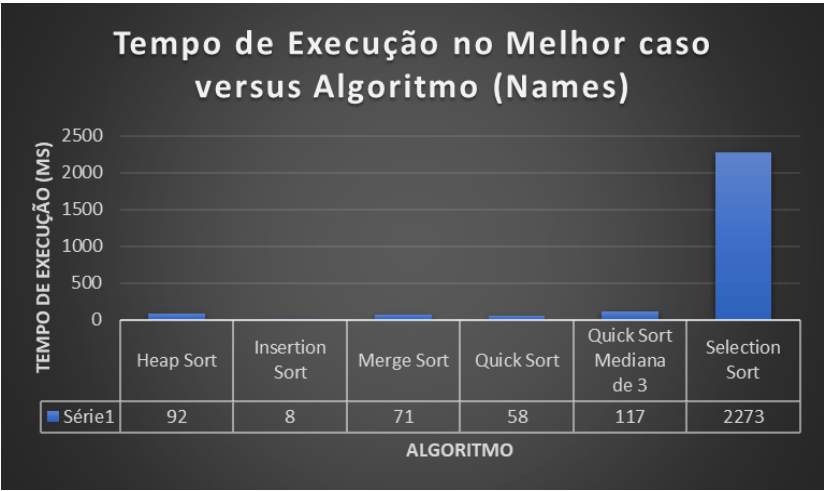


(Gráfico 09)

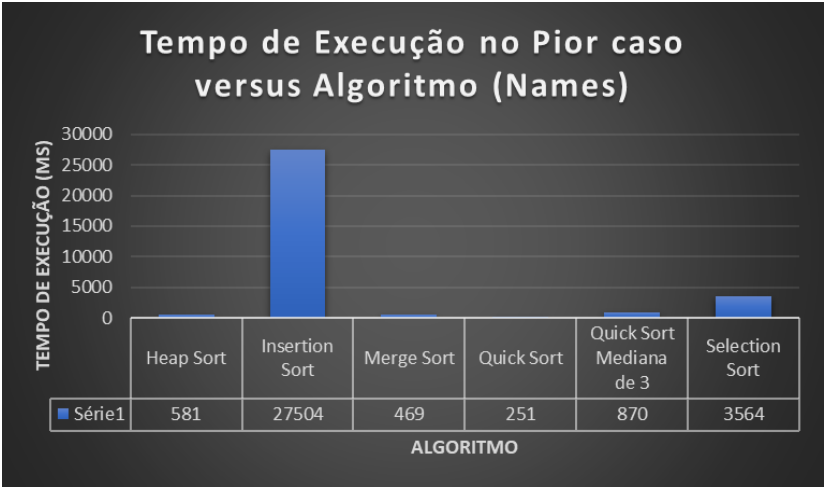
- **Parâmetro de ordenação: Names**



(Gráfico 10)



(Gráfico 11)



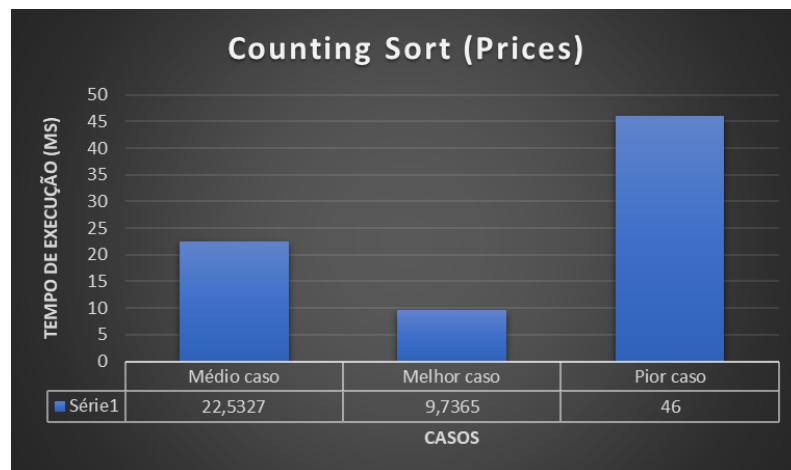
(Gráfico 12)

Por fim, podemos analisar os tempos de execução de cada algoritmo individualmente, levando em consideração cada um dos parâmetros pré definidos.

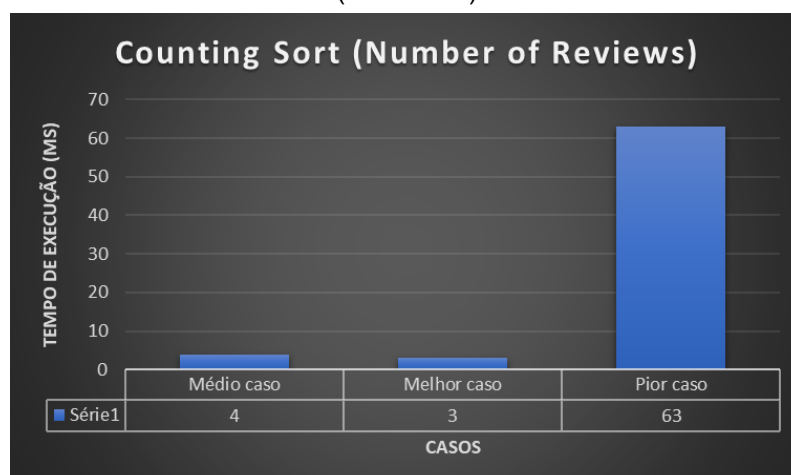
- **Counting Sort**

É um algoritmo de ordenação com complexidade “O” para números pequenos, em que suas chaves podem assumir valores entre 0 e “M-1”. Caso tenha “k0” chaves de valor 0, ocupa as primeiras posições “k0” do vetor final: 0 a “k0-1”. A ideia básica de classificar por contagem é para cada entrada x, o número de elementos menor que i, que seria cada inteiro do vetor. Tais informações são utilizadas para colocar o elemento x na posição correta no array de saída.

Para o caso do Counting Sort, não é possível fazer a ordenação com parâmetros que sejam do tipo String, logo não foi possível fazer a ordenação com os parâmetros “Names” e “Last Review”.



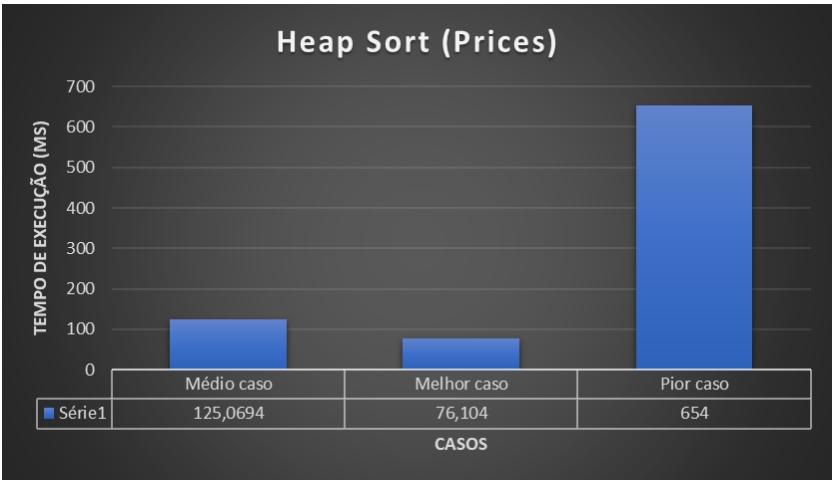
(Gráfico 13)



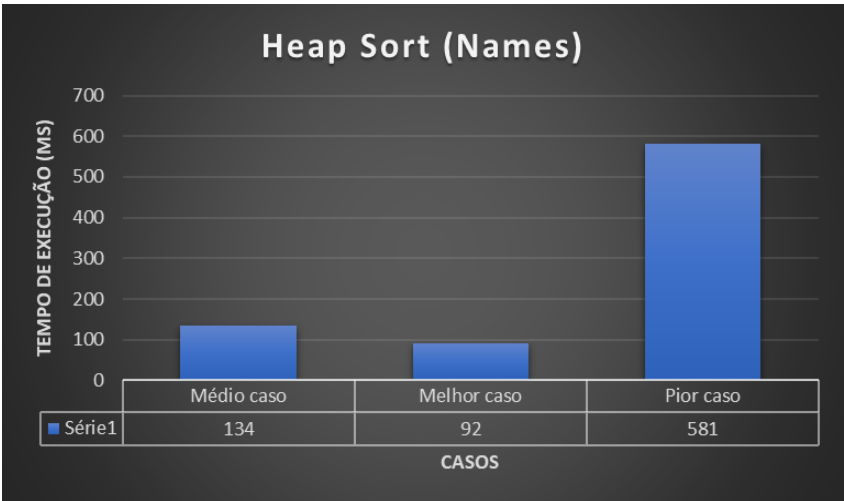
(Gráfico 14)



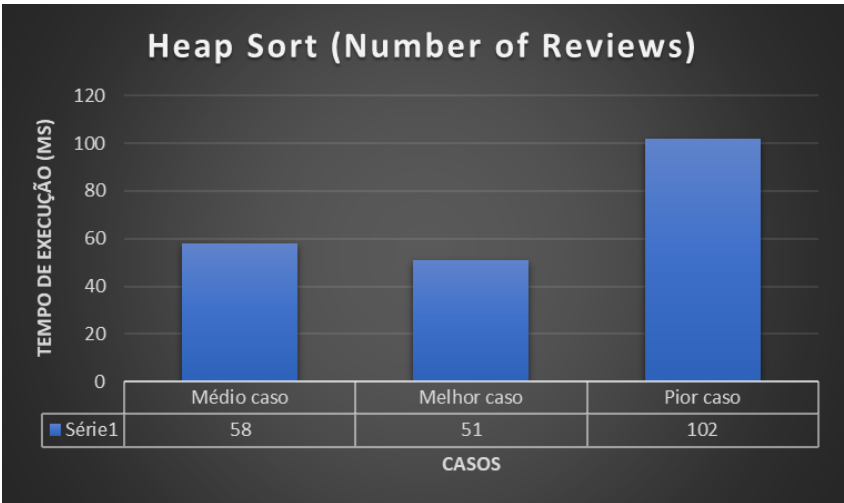
- **Heap Sort**



(Gráfico 15)

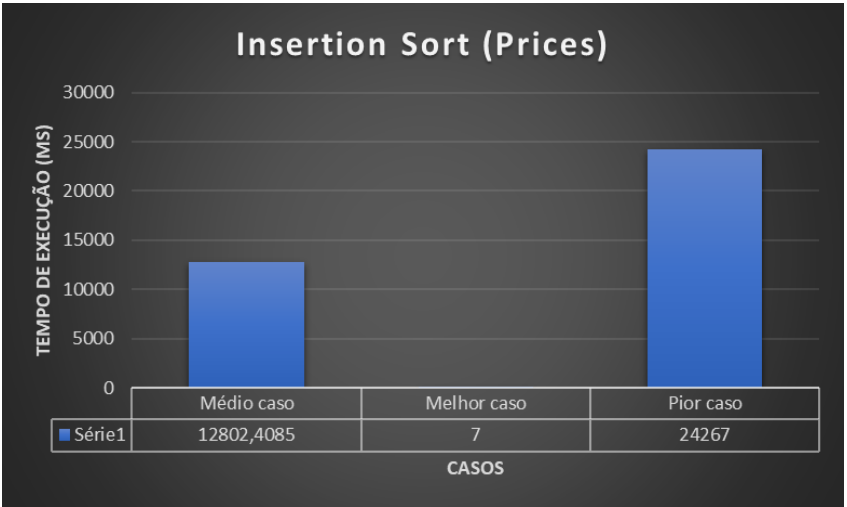


(Gráfico 16)

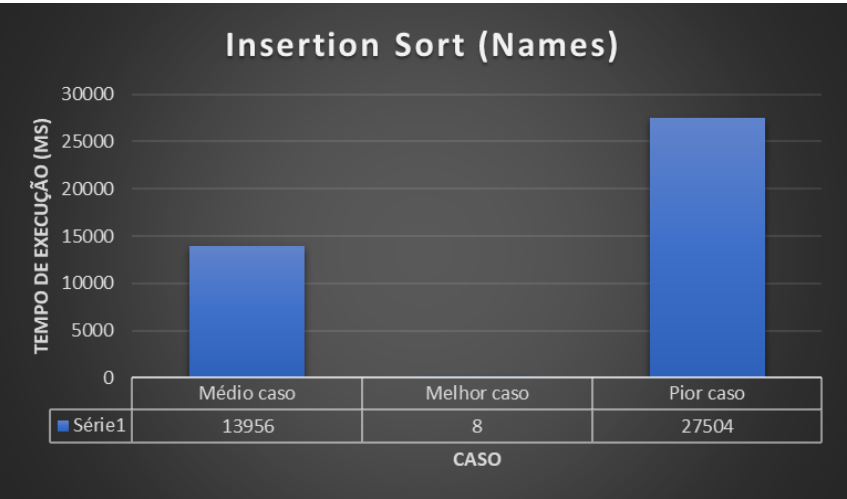


(Gráfico 17)

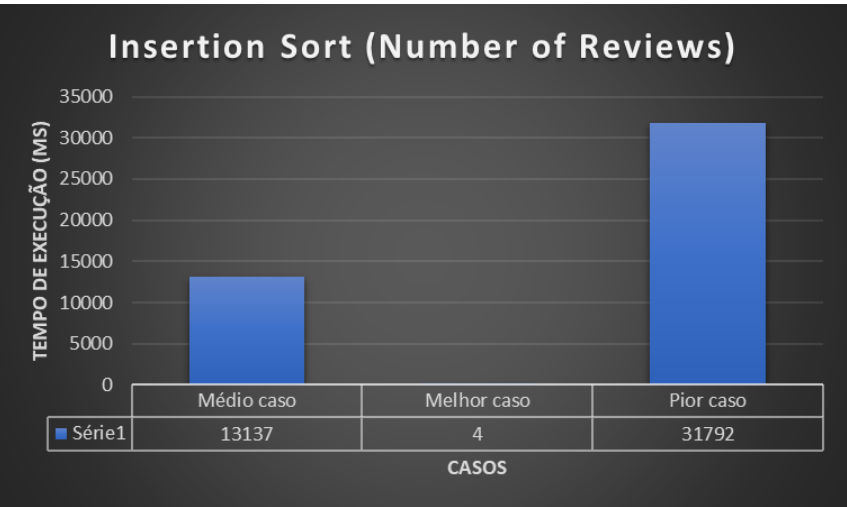
- Insertion Sort



(Gráfico 18)

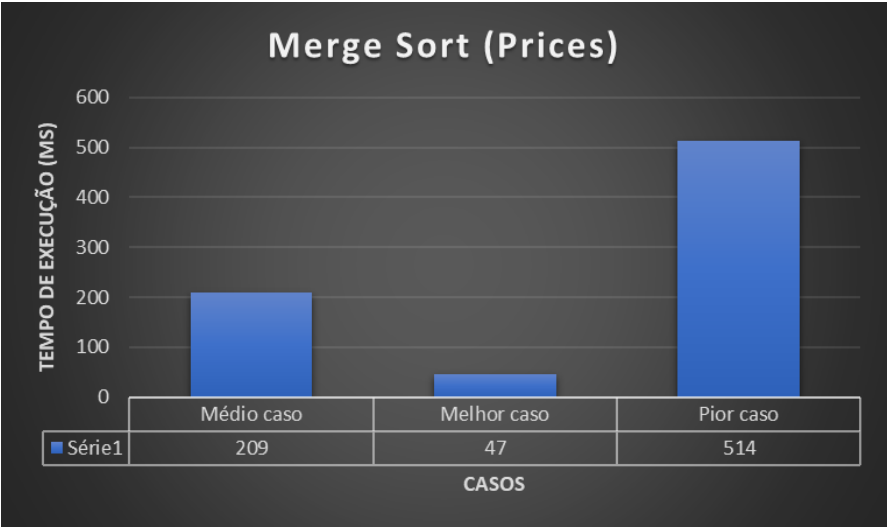


(Gráfico 19)

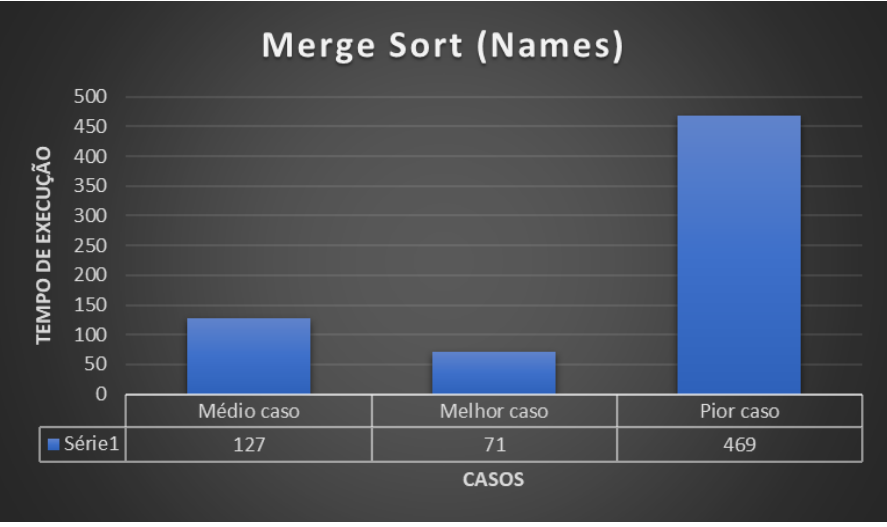


(Gráfico 20)

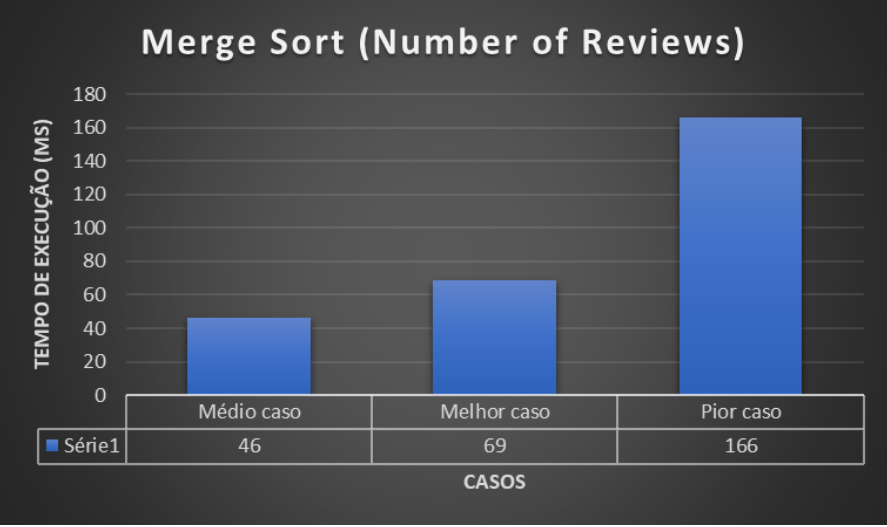
- Merge Sort



(Gráfico 21)

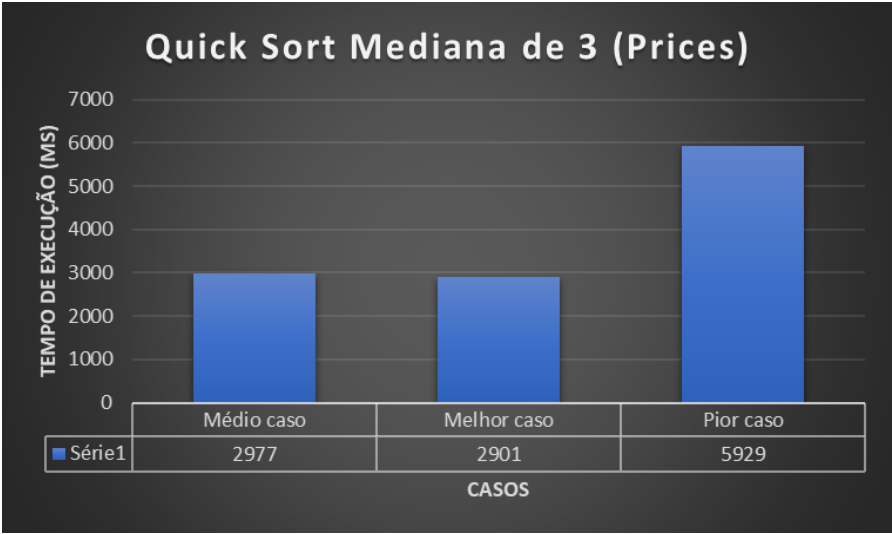


(Gráfico 22)

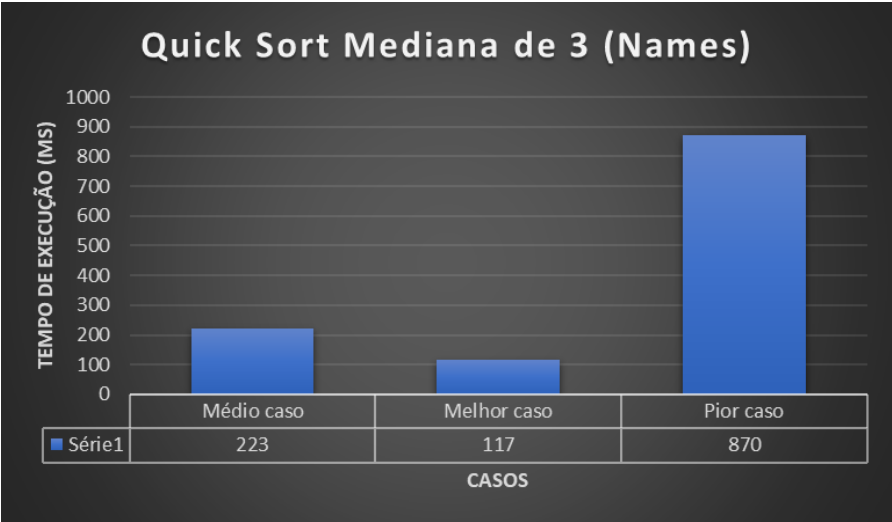


(Gráfico 23)

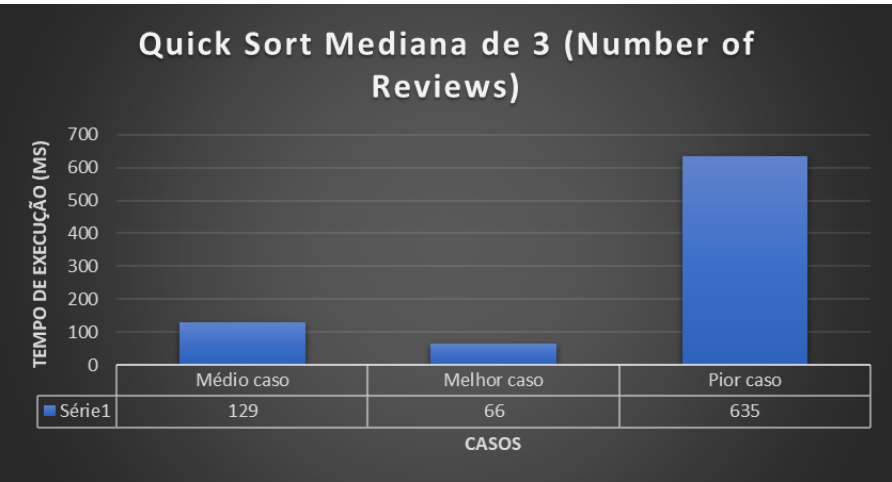
- Quick Sort Mediana de 3



(Gráfico 24)

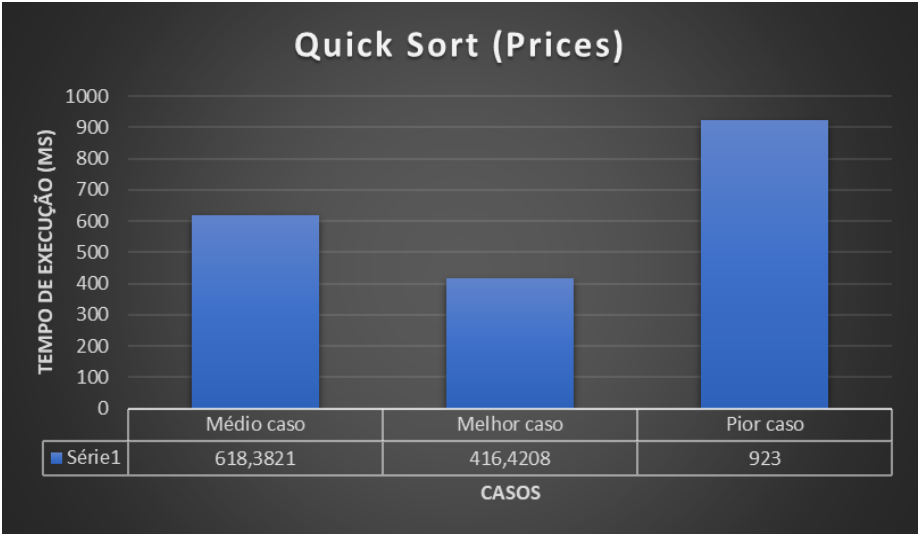


(Gráfico 25)

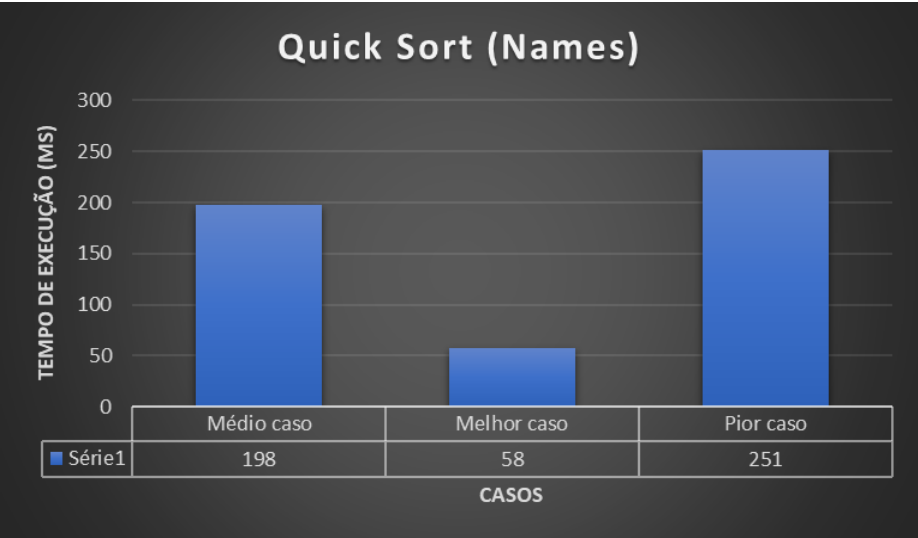


(Gráfico 26)

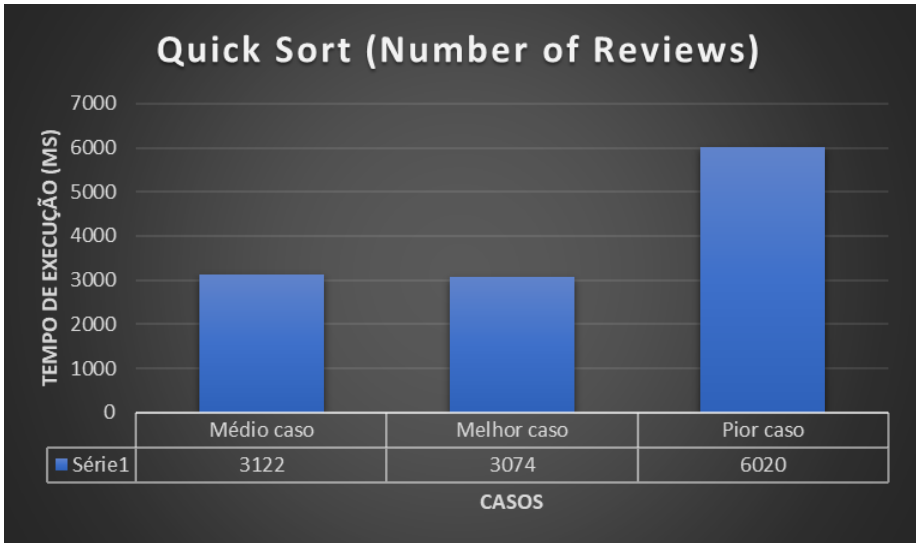
- Quick Sort



(Gráfico 27)



(Gráfico 28)

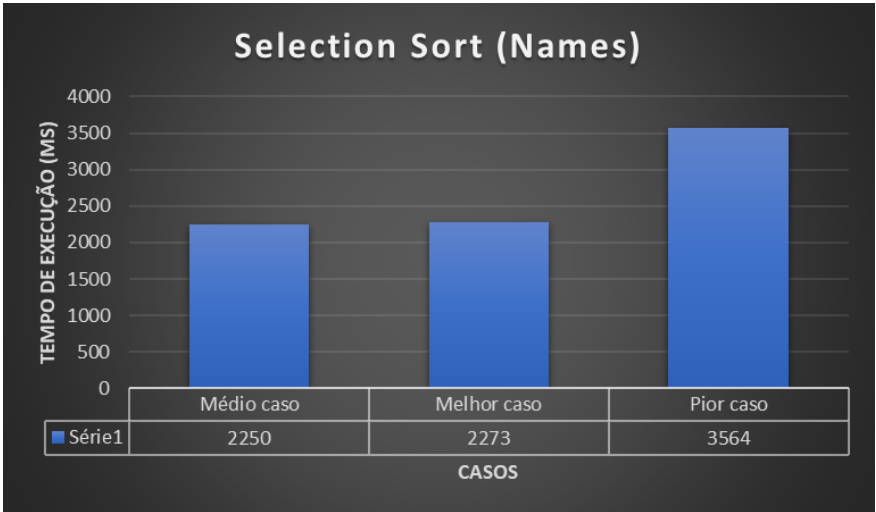


(Gráfico 29)

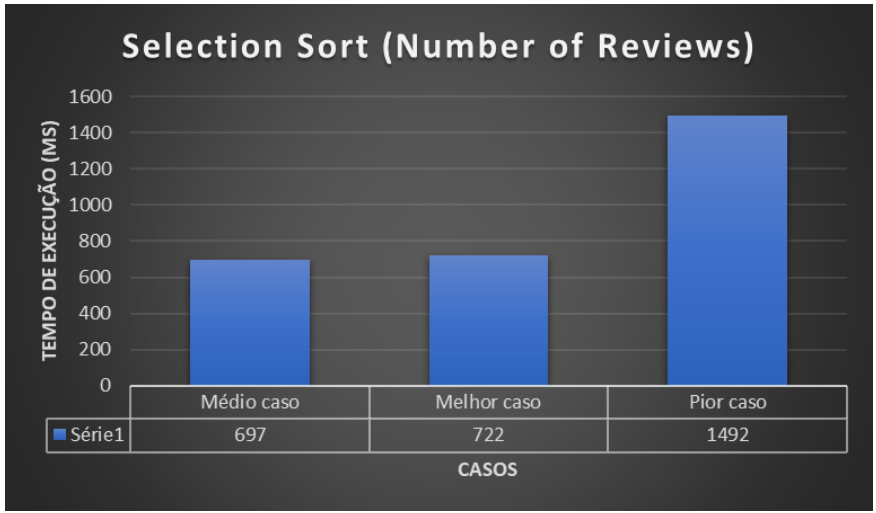
- Selection Sort



(Gráfico 30)



(Gráfico 31)



(Gráfico 32)

Após a análise gráfica apresentada é possível realizar algumas conclusões, dentre elas a mais assertiva se refere ao benefício do uso do Merge Sort, que se mostrou mais eficiente no quesito referente ao tempo de execução e o consumo de memória se mostrou dentro da média dos demais.