Mensuração de Ordenação

Trabalho de Estruturas de Dados

Alunos: Diego dos Santos Fernandes e Ueslei Albuquerque Garcia

Introdução

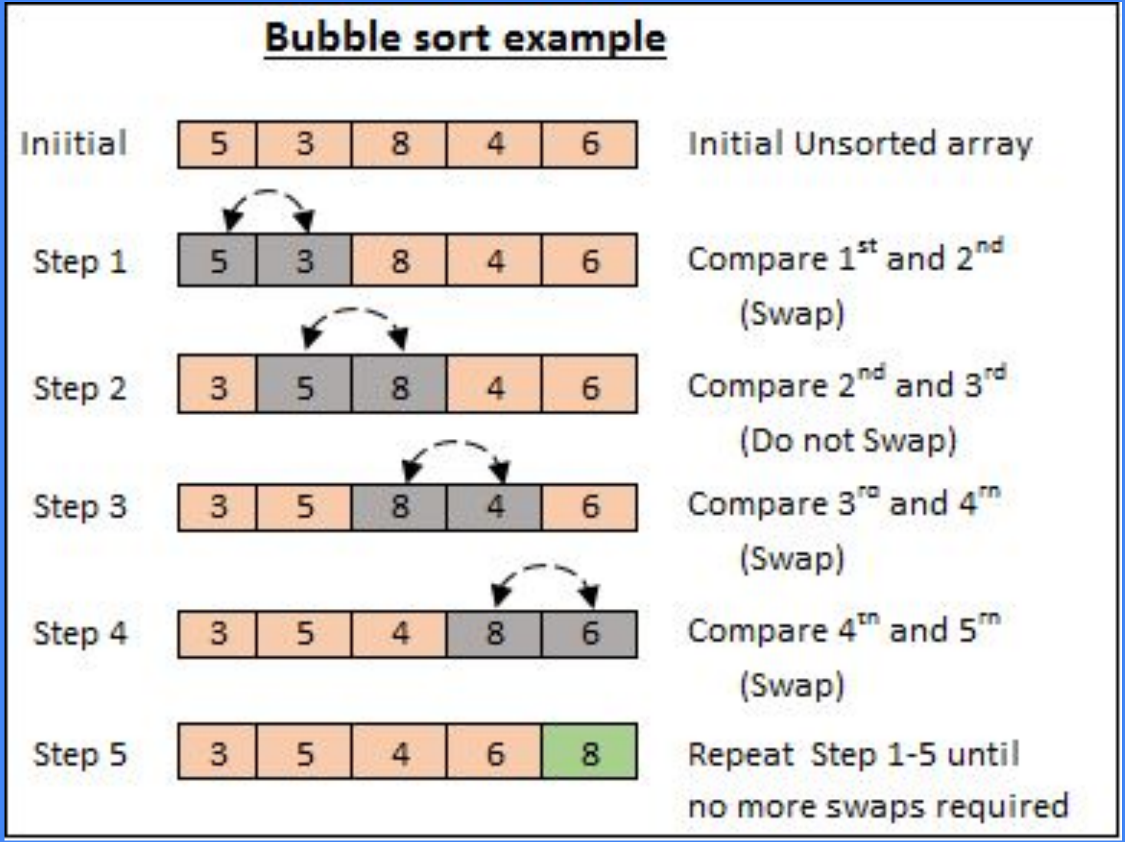
Porque é a ordenação é importante?

- Redução de complexidade de problemas em seu código

Quais suas vantagens ou desvantagens?

- Quanta memória tem disponível para uso?
- A coleção deve crescer?
- Qual o tamanho da coleção a ser ordenada?
- Quais os requisitos de sistema e limitação antes de decidir qual algoritmo utilizar?
- Tem que ser utilizado em curto, médio ou longo tempo

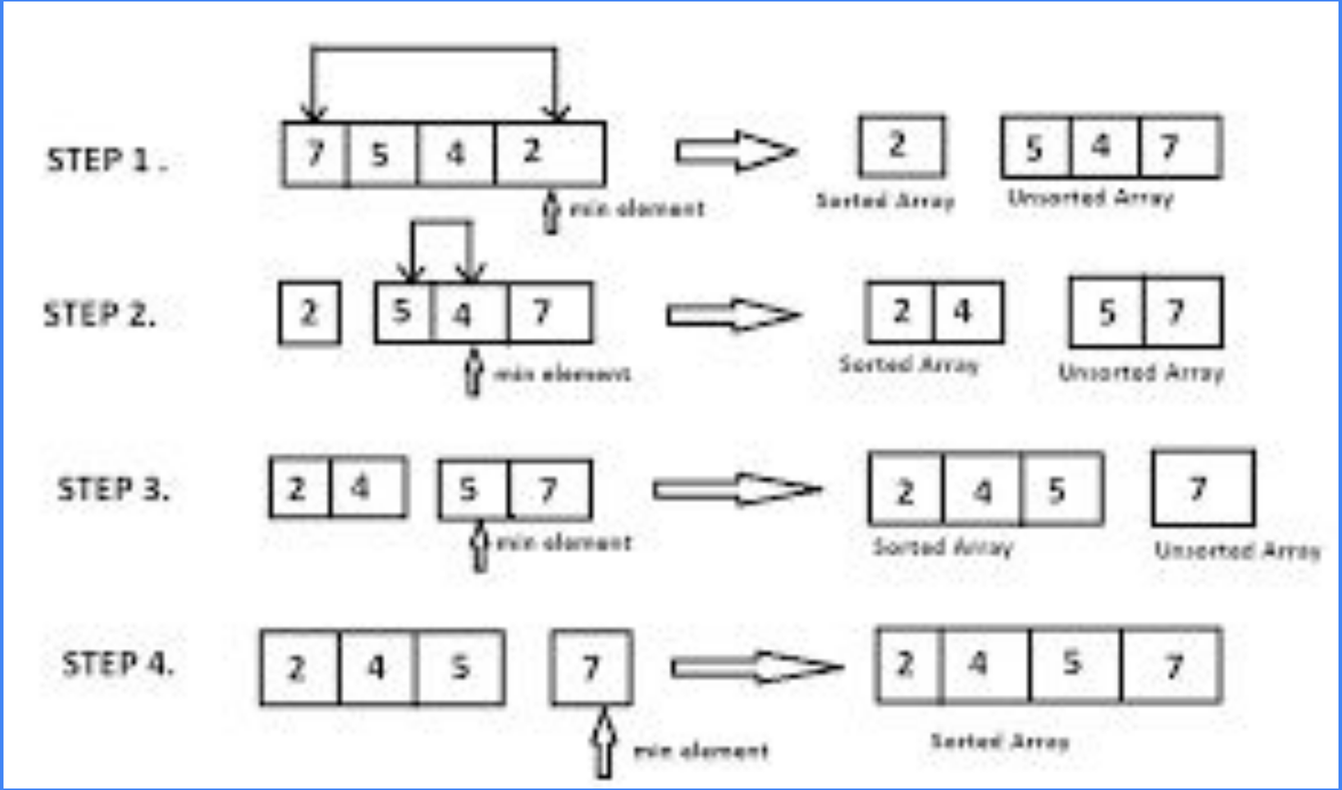
Ordenação do método em bolha



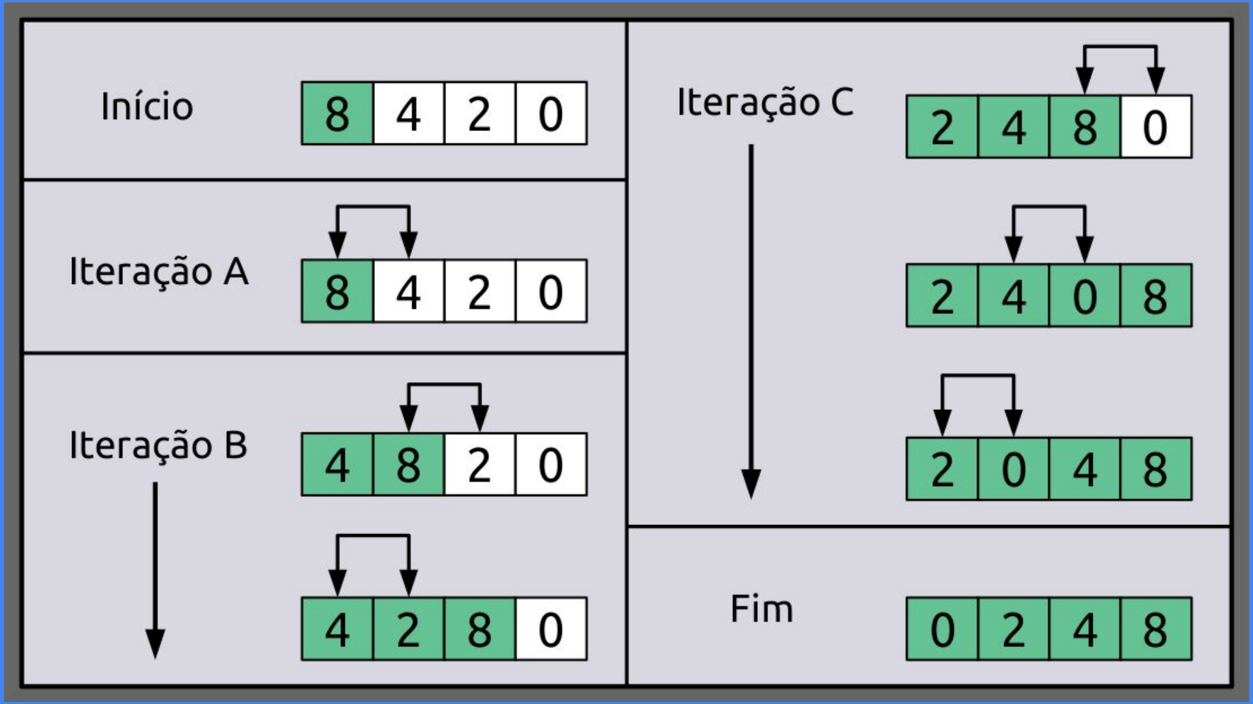
Bubble sort example

Iniitial	5	3	8	4	6	Initial Unsorted array
Step 1	5	3	8	4	6	Compare 1st and 2nd (Swap)
Step 2	3	5	8	4	6	Compare 2nd and 3rd (Do not Swap)
Step 3	3	5	8	4	6	Compare 3rd and 4th (Swap)
Step 4	3	5	4	8	6	Compare 4th and 5th (Swap)
Step 5	3	5	4	6	8	Repeat Step 1-5 until no more swaps required

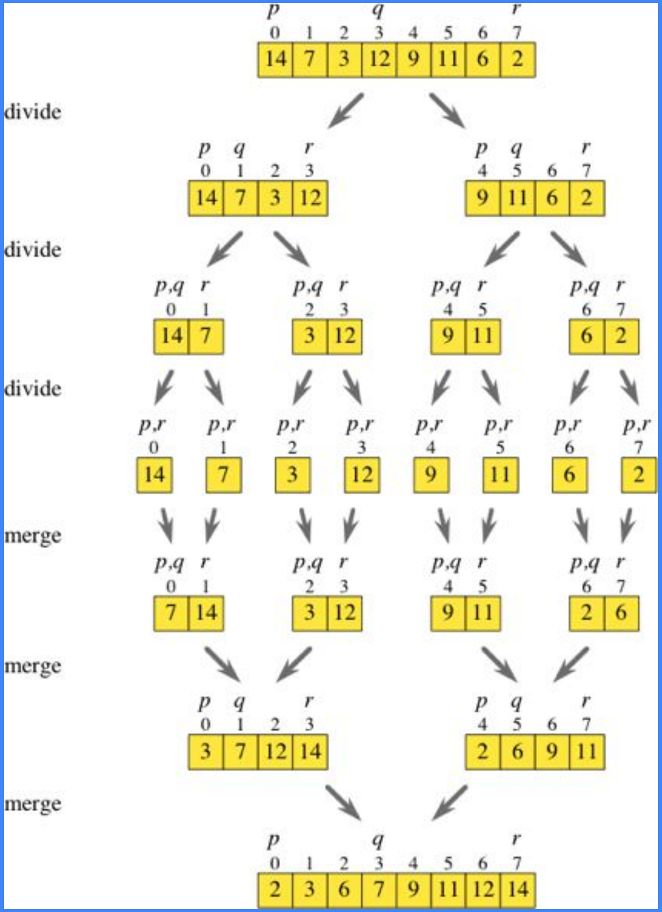
Ordenação do método de seleção



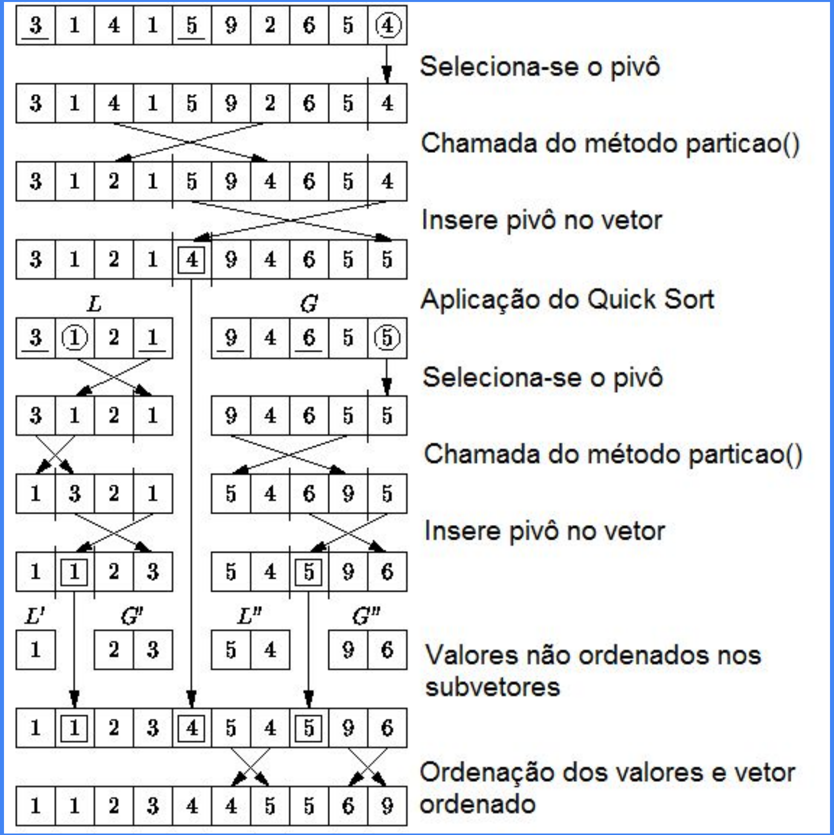
Ordenação do método de inserção



Ordenação do método de intercalação



Ordenação do método rápido



Seleciona-se o pivô

Chamada do método particao()

Insere pivô no vetor

Aplicação do Quick Sort

Seleciona-se o pivô

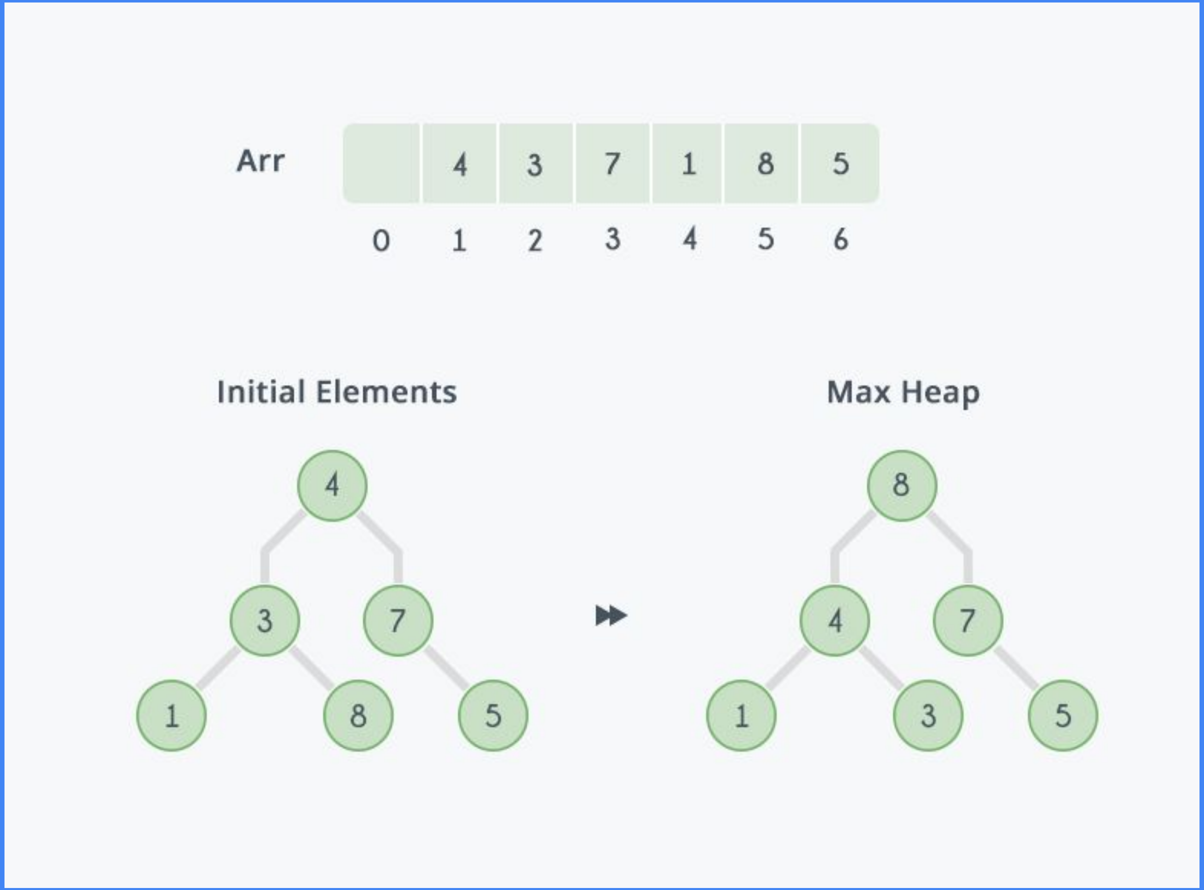
Chamada do método particao()

Insere pivô no vetor

Valores não ordenados nos subvetores

Ordenação dos valores e vetor ordenado

Ordenação do método de pilha

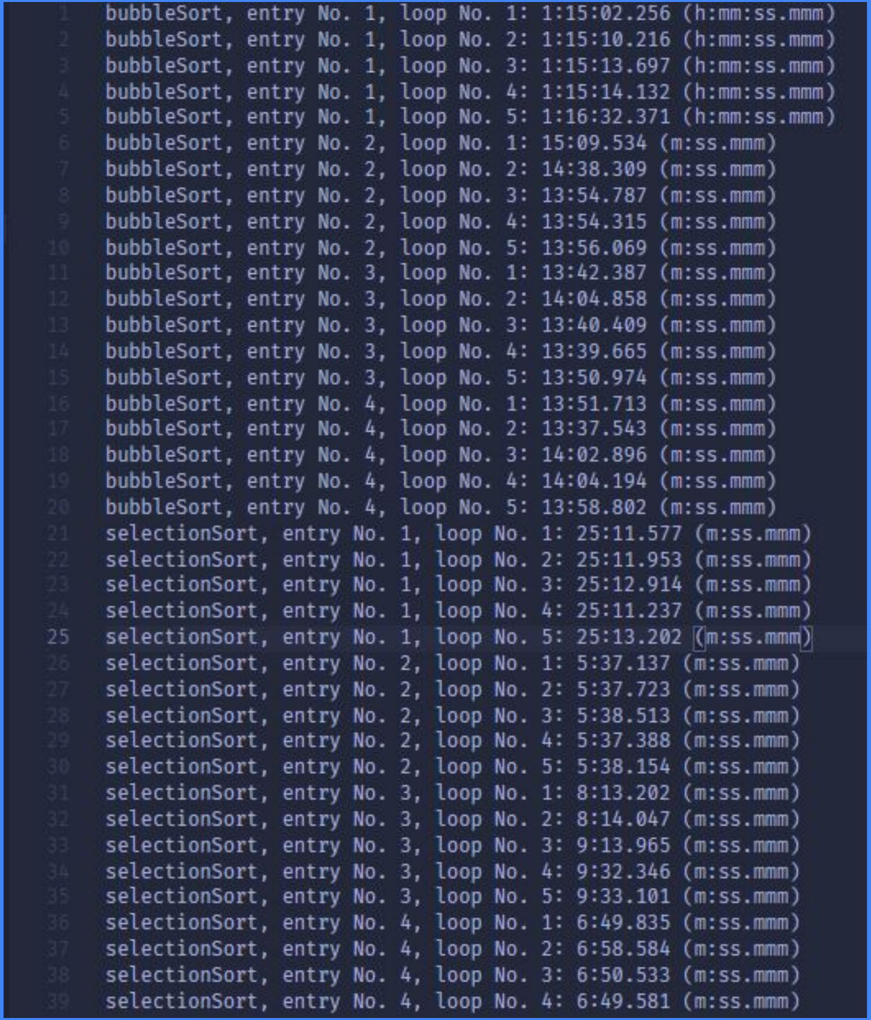


Casos de teste (tempo de execução e seu gráfico de média aritmética)

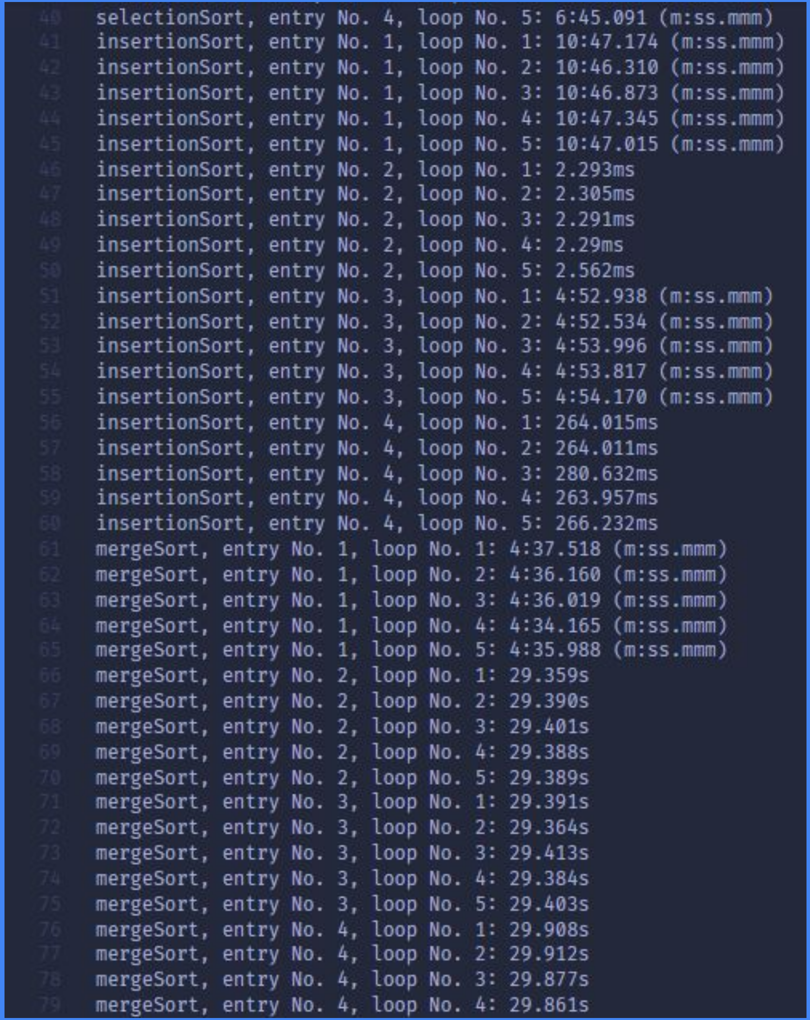
- Caso de teste 1: Você deve gerar um arquivo de teste que contenha 1 milhão de números de 1 a 99999. Os valores devem estar arranjados de maneira aleatória.
- Caso de teste 2: Você deve gerar um arquivo de teste que contenha 750 mil números com valores entre 1 e 99999. Os valores devem estar arranjados em ordem crescente.
- Caso de teste 3: Você deve gerar um arquivo de teste que contenha 750 mil números com valores entre 1 e 99999. Os valores devem estar arranjados em ordem decrescente.
- Caso de teste 4: Você deve gerar um arquivo de teste que contenha 500 mil números com valores entre 1 e 99999. Os valores devem estar arranjados parcialmente em ordem decrescente, ou seja, entre 250 e 1000 números deverão estar fora de ordem.

Configurações de hardware

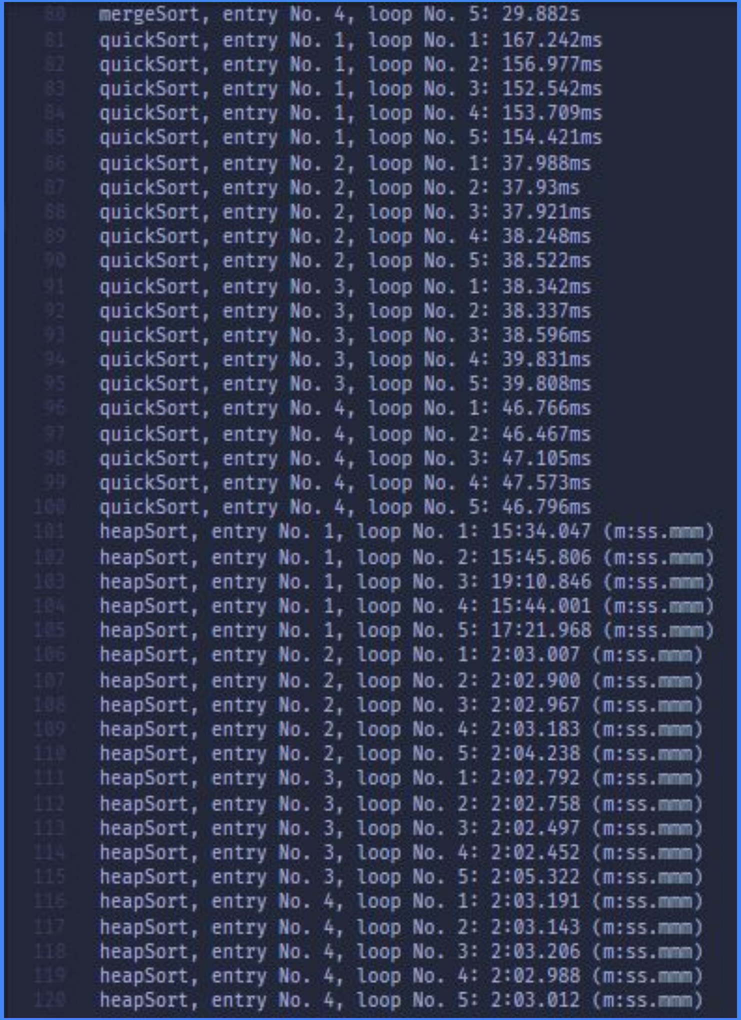
- Cpu: Intel Pentium N5000 1.20 ghz
- Núcleos: 4
- Threads: 4
- Placa mãe: Dell inspiron 15
- Memória RAM: 4 GB DDR4 1300 MHz frequência
- Placa de Vídeo(Integrada): Intel UHD Graphics 605



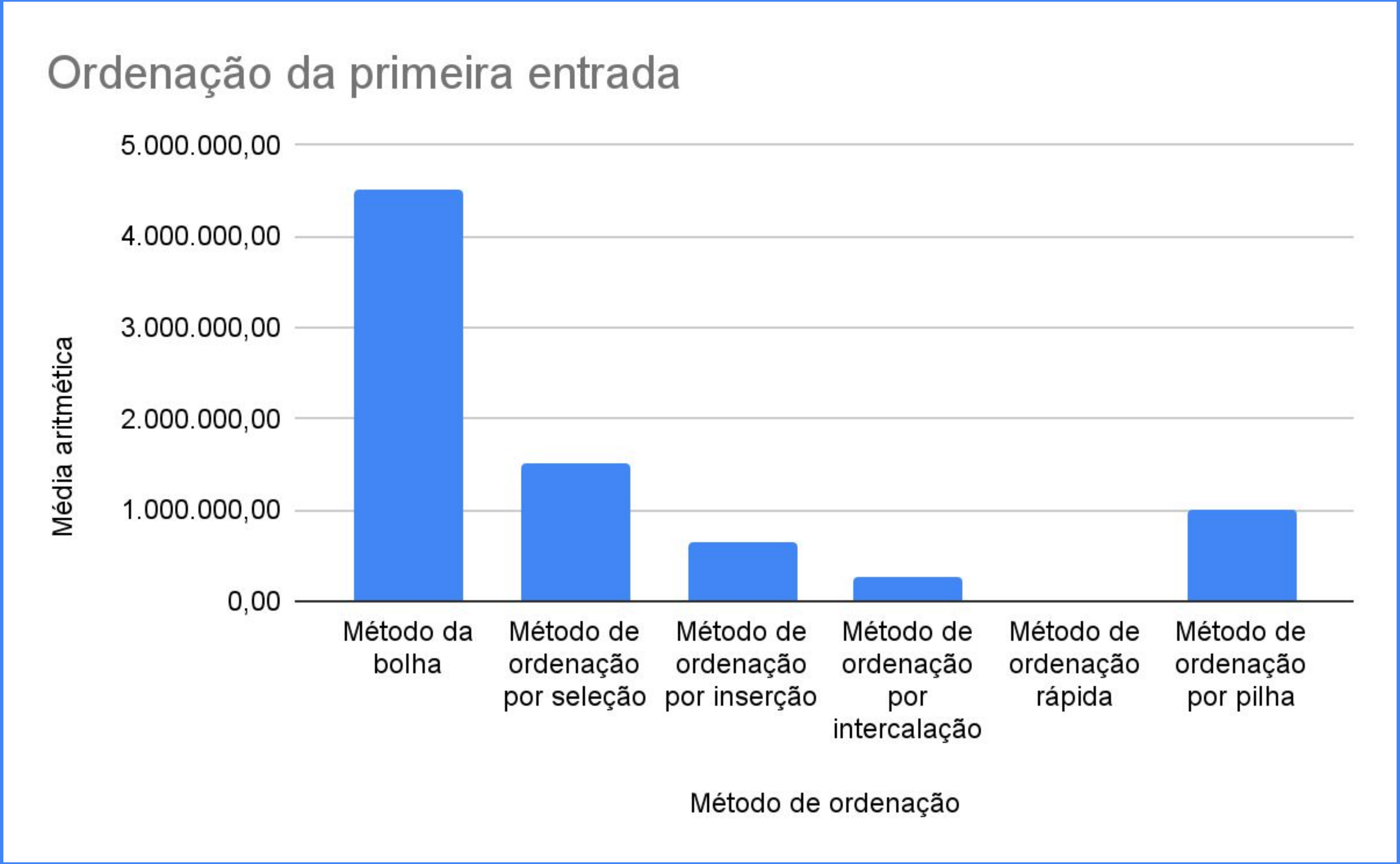
```
bubbleSort, entry No. 1, loop No. 1: 1:15:02.256 (h:mm:ss.mmm)
bubbleSort, entry No. 1, loop No. 2: 1:15:10.216 (h:mm:ss.mmm)
bubbleSort, entry No. 1, loop No. 3: 1:15:13.697 (h:mm:ss.mmm)
bubbleSort, entry No. 1, loop No. 4: 1:15:14.132 (h:mm:ss.mmm)
bubbleSort, entry No. 1, loop No. 5: 1:16:32.371 (h:mm:ss.mmm)
bubbleSort, entry No. 2, loop No. 1: 15:09.534 (m:ss.mmm)
bubbleSort, entry No. 2, loop No. 2: 14:38.309 (m:ss.mmm)
bubbleSort, entry No. 2, loop No. 3: 13:54.787 (m:ss.mmm)
bubbleSort, entry No. 2, loop No. 4: 13:54.315 (m:ss.mmm)
bubbleSort, entry No. 2, loop No. 5: 13:56.069 (m:ss.mmm)
bubbleSort, entry No. 3, loop No. 1: 13:42.387 (m:ss.mmm)
bubbleSort, entry No. 3, loop No. 2: 14:04.858 (m:ss.mmm)
bubbleSort, entry No. 3, loop No. 3: 13:40.409 (m:ss.mmm)
bubbleSort, entry No. 3, loop No. 4: 13:39.665 (m:ss.mmm)
bubbleSort, entry No. 3, loop No. 5: 13:50.974 (m:ss.mmm)
bubbleSort, entry No. 4, loop No. 1: 13:51.713 (m:ss.mmm)
bubbleSort, entry No. 4, loop No. 2: 13:37.543 (m:ss.mmm)
bubbleSort, entry No. 4, loop No. 3: 14:02.896 (m:ss.mmm)
bubbleSort, entry No. 4, loop No. 4: 14:04.194 (m:ss.mmm)
bubbleSort, entry No. 4, loop No. 5: 13:58.802 (m:ss.mmm)
selectionSort, entry No. 1, loop No. 1: 25:11.577 (m:ss.mmm)
selectionSort, entry No. 1, loop No. 2: 25:11.953 (m:ss.mmm)
selectionSort, entry No. 1, loop No. 3: 25:12.914 (m:ss.mmm)
selectionSort, entry No. 1, loop No. 4: 25:11.237 (m:ss.mmm)
selectionSort, entry No. 1, loop No. 5: 25:13.202 (m:ss.mmm)
selectionSort, entry No. 2, loop No. 1: 5:37.137 (m:ss.mmm)
selectionSort, entry No. 2, loop No. 2: 5:37.723 (m:ss.mmm)
selectionSort, entry No. 2, loop No. 3: 5:38.513 (m:ss.mmm)
selectionSort, entry No. 2, loop No. 4: 5:37.388 (m:ss.mmm)
selectionSort, entry No. 2, loop No. 5: 5:38.154 (m:ss.mmm)
selectionSort, entry No. 3, loop No. 1: 8:13.202 (m:ss.mmm)
selectionSort, entry No. 3, loop No. 2: 8:14.047 (m:ss.mmm)
selectionSort, entry No. 3, loop No. 3: 9:13.965 (m:ss.mmm)
selectionSort, entry No. 3, loop No. 4: 9:32.346 (m:ss.mmm)
selectionSort, entry No. 3, loop No. 5: 9:33.101 (m:ss.mmm)
selectionSort, entry No. 4, loop No. 1: 6:49.835 (m:ss.mmm)
selectionSort, entry No. 4, loop No. 2: 6:58.584 (m:ss.mmm)
selectionSort, entry No. 4, loop No. 3: 6:50.533 (m:ss.mmm)
selectionSort, entry No. 4, loop No. 4: 6:49.581 (m:ss.mmm)
```



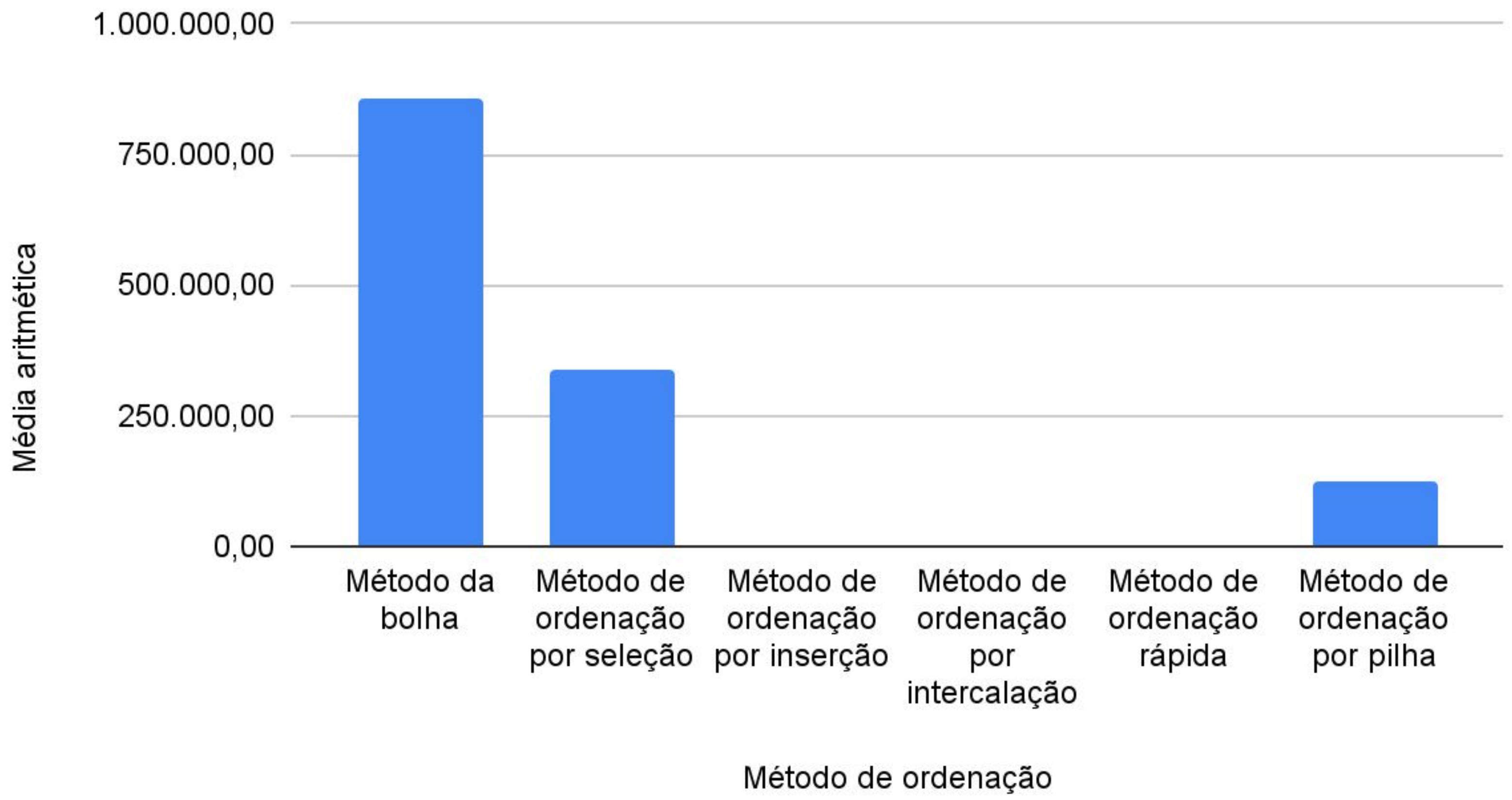
```
selectionSort, entry No. 4, loop No. 5: 6:45.091 (m:ss.mmm)
insertionSort, entry No. 1, loop No. 1: 10:47.174 (m:ss.mmm)
insertionSort, entry No. 1, loop No. 2: 10:46.310 (m:ss.mmm)
insertionSort, entry No. 1, loop No. 3: 10:46.873 (m:ss.mmm)
insertionSort, entry No. 1, loop No. 4: 10:47.345 (m:ss.mmm)
insertionSort, entry No. 1, loop No. 5: 10:47.015 (m:ss.mmm)
insertionSort, entry No. 2, loop No. 1: 2.293ms
insertionSort, entry No. 2, loop No. 2: 2.305ms
insertionSort, entry No. 2, loop No. 3: 2.291ms
insertionSort, entry No. 2, loop No. 4: 2.29ms
insertionSort, entry No. 2, loop No. 5: 2.562ms
insertionSort, entry No. 3, loop No. 1: 4:52.938 (m:ss.mmm)
insertionSort, entry No. 3, loop No. 2: 4:52.534 (m:ss.mmm)
insertionSort, entry No. 3, loop No. 3: 4:53.996 (m:ss.mmm)
insertionSort, entry No. 3, loop No. 4: 4:53.817 (m:ss.mmm)
insertionSort, entry No. 3, loop No. 5: 4:54.170 (m:ss.mmm)
insertionSort, entry No. 4, loop No. 1: 264.015ms
insertionSort, entry No. 4, loop No. 2: 264.011ms
insertionSort, entry No. 4, loop No. 3: 280.632ms
insertionSort, entry No. 4, loop No. 4: 263.957ms
insertionSort, entry No. 4, loop No. 5: 266.232ms
mergeSort, entry No. 1, loop No. 1: 4:37.518 (m:ss.mmm)
mergeSort, entry No. 1, loop No. 2: 4:36.160 (m:ss.mmm)
mergeSort, entry No. 1, loop No. 3: 4:36.019 (m:ss.mmm)
mergeSort, entry No. 1, loop No. 4: 4:34.165 (m:ss.mmm)
mergeSort, entry No. 1, loop No. 5: 4:35.988 (m:ss.mmm)
mergeSort, entry No. 2, loop No. 1: 29.359s
mergeSort, entry No. 2, loop No. 2: 29.390s
mergeSort, entry No. 2, loop No. 3: 29.401s
mergeSort, entry No. 2, loop No. 4: 29.388s
mergeSort, entry No. 2, loop No. 5: 29.389s
mergeSort, entry No. 3, loop No. 1: 29.391s
mergeSort, entry No. 3, loop No. 2: 29.364s
mergeSort, entry No. 3, loop No. 3: 29.413s
mergeSort, entry No. 3, loop No. 4: 29.384s
mergeSort, entry No. 3, loop No. 5: 29.403s
mergeSort, entry No. 4, loop No. 1: 29.908s
mergeSort, entry No. 4, loop No. 2: 29.912s
mergeSort, entry No. 4, loop No. 3: 29.877s
mergeSort, entry No. 4, loop No. 4: 29.861s
```

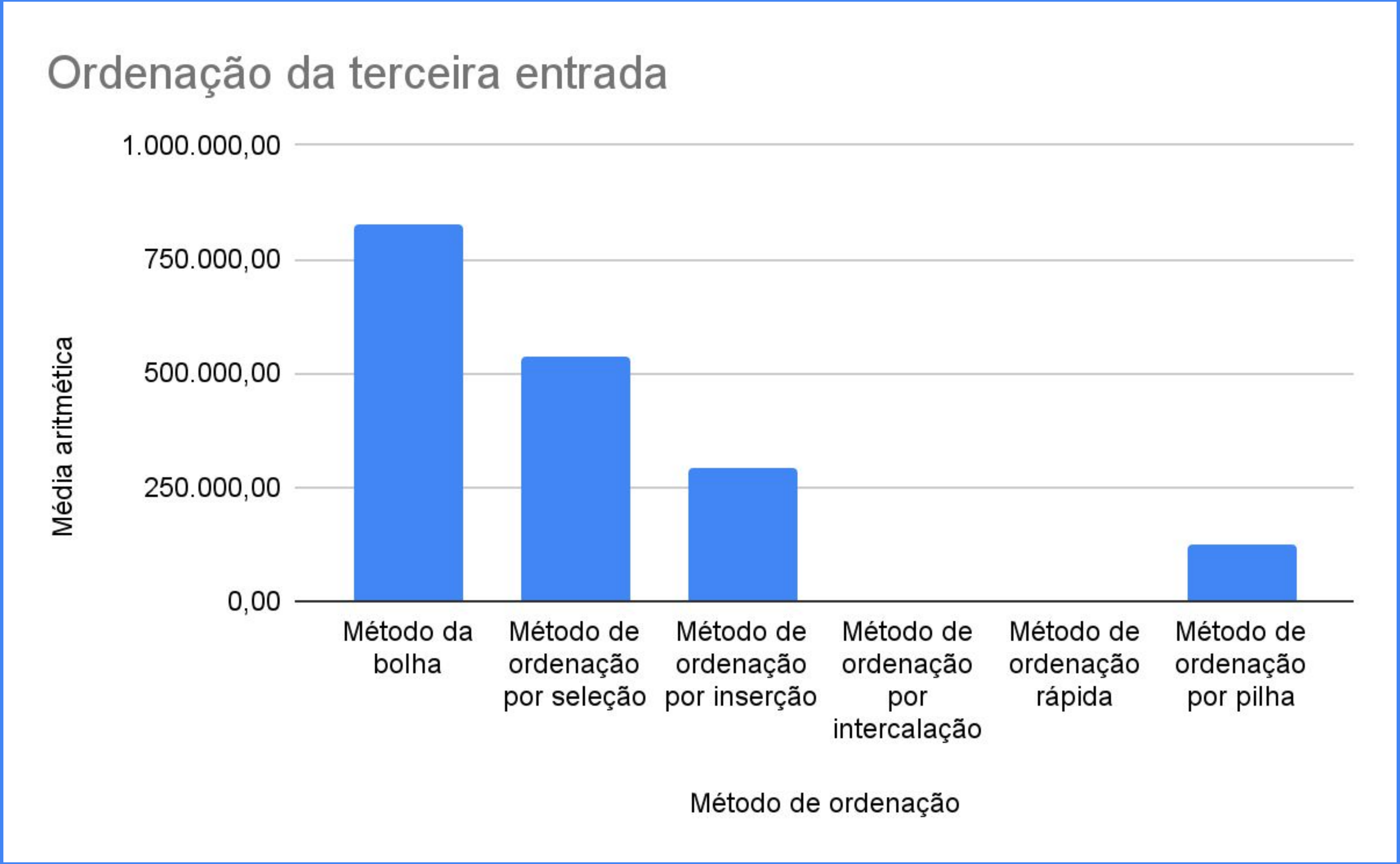


```
mergeSort, entry No. 4, loop No. 5: 29.882s
quickSort, entry No. 1, loop No. 1: 167.242ms
quickSort, entry No. 1, loop No. 2: 156.977ms
quickSort, entry No. 1, loop No. 3: 152.542ms
quickSort, entry No. 1, loop No. 4: 153.709ms
quickSort, entry No. 1, loop No. 5: 154.421ms
quickSort, entry No. 2, loop No. 1: 37.988ms
quickSort, entry No. 2, loop No. 2: 37.93ms
quickSort, entry No. 2, loop No. 3: 37.921ms
quickSort, entry No. 2, loop No. 4: 38.248ms
quickSort, entry No. 2, loop No. 5: 38.522ms
quickSort, entry No. 3, loop No. 1: 38.342ms
quickSort, entry No. 3, loop No. 2: 38.337ms
quickSort, entry No. 3, loop No. 3: 38.596ms
quickSort, entry No. 3, loop No. 4: 39.831ms
quickSort, entry No. 3, loop No. 5: 39.808ms
quickSort, entry No. 4, loop No. 1: 46.766ms
quickSort, entry No. 4, loop No. 2: 46.467ms
quickSort, entry No. 4, loop No. 3: 47.105ms
quickSort, entry No. 4, loop No. 4: 47.573ms
quickSort, entry No. 4, loop No. 5: 46.796ms
heapSort, entry No. 1, loop No. 1: 15:34.047 (m:ss.mmm)
heapSort, entry No. 1, loop No. 2: 15:45.806 (m:ss.mmm)
heapSort, entry No. 1, loop No. 3: 19:10.846 (m:ss.mmm)
heapSort, entry No. 1, loop No. 4: 15:44.001 (m:ss.mmm)
heapSort, entry No. 1, loop No. 5: 17:21.968 (m:ss.mmm)
heapSort, entry No. 2, loop No. 1: 2:03.007 (m:ss.mmm)
heapSort, entry No. 2, loop No. 2: 2:02.900 (m:ss.mmm)
heapSort, entry No. 2, loop No. 3: 2:02.967 (m:ss.mmm)
heapSort, entry No. 2, loop No. 4: 2:03.183 (m:ss.mmm)
heapSort, entry No. 2, loop No. 5: 2:04.238 (m:ss.mmm)
heapSort, entry No. 3, loop No. 1: 2:02.792 (m:ss.mmm)
heapSort, entry No. 3, loop No. 2: 2:02.758 (m:ss.mmm)
heapSort, entry No. 3, loop No. 3: 2:02.497 (m:ss.mmm)
heapSort, entry No. 3, loop No. 4: 2:02.452 (m:ss.mmm)
heapSort, entry No. 3, loop No. 5: 2:05.322 (m:ss.mmm)
heapSort, entry No. 4, loop No. 1: 2:03.191 (m:ss.mmm)
heapSort, entry No. 4, loop No. 2: 2:03.143 (m:ss.mmm)
heapSort, entry No. 4, loop No. 3: 2:03.206 (m:ss.mmm)
heapSort, entry No. 4, loop No. 4: 2:02.988 (m:ss.mmm)
heapSort, entry No. 4, loop No. 5: 2:03.012 (m:ss.mmm)
```


Ordenação da primeira entrada
5.000.000,00
4.000.000,00
3.000.000,00
2.000.000,00
1.000.000,00
0,00
Média aritmética
Método da bolha
Método de ordenação por seleção
Método de ordenação por inserção
Método de ordenação por intercalação
Método de ordenação rápida
Método de ordenação por pilha
Método de ordenação

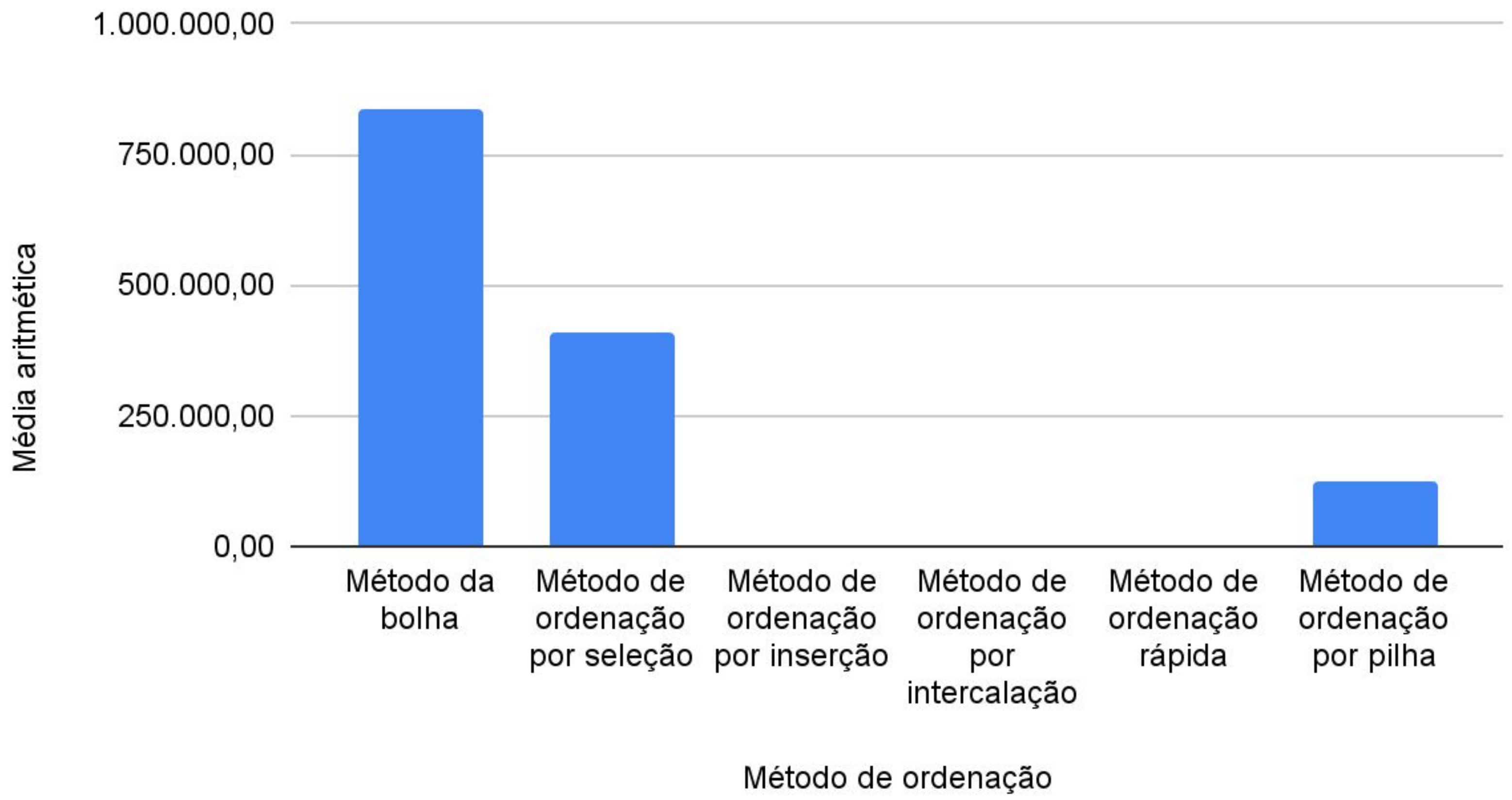
Ordenação da segunda entrada





Ordenação da terceira entrada
1.000.000,00
750.000,00
500.000,00
250.000,00
0,00
Média aritmética
Método da bolha
Método de ordenação por seleção
Método de ordenação por inserção
Método de ordenação por intercalação
Método de ordenação rápida
Método de ordenação por pilha
Método de ordenação

Ordenação da quarta entrada



Conclusão

- Uso de memória RAM e CPU no máximo mesmo com VB ativado(núcleo duplo para folga de cache)
- Teve uso de somente um núcleo sem distribuição para aliviar cache de memória
- Teste feitos com programas de inicialização e segundo plano fechados para não dar interferência no resultado
- Método de ordenação rápida se sobressai aos demais métodos
- Demorou mais da metade de um dia para terminar os teste tirando o fato que foi feito um teste anteriormente enquanto jogava jogo e fala no discord com navegador aberto em segundo plano
- Segundo teste foi feito com Wallpaper Engine em segundo plano houve uma possível interferência de resultado