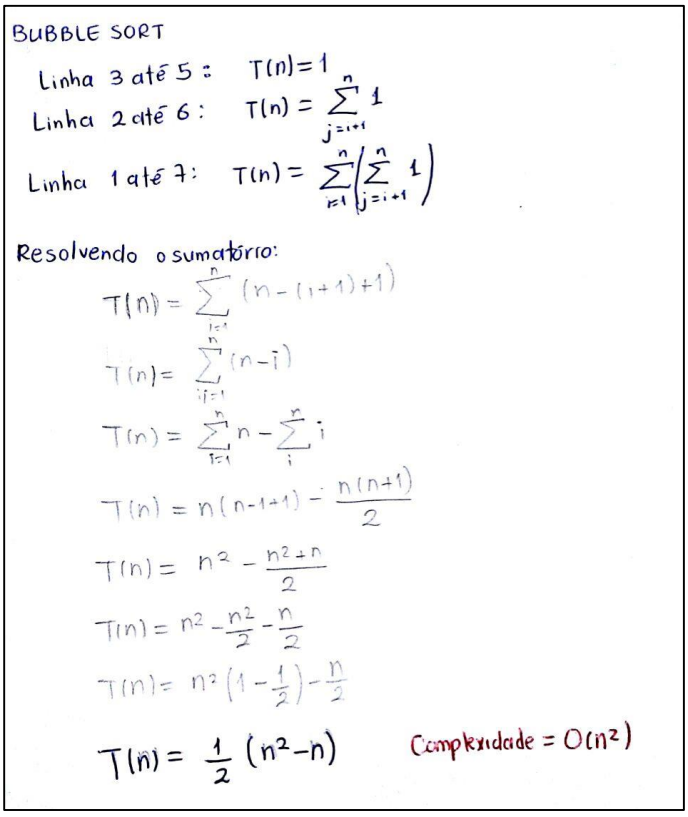
EXPERIMENTO: ALGORITMOS DE ORDENAÇÃO

Neste experimento se analisaram os algoritmos: bubble sort, insertion sort, merge sort, quick sort e shell sort. A continuação se expõem os resultados para cada caso.

Complexidade

O procedimento para achar a complexidade dos algoritmos baseados no pseudocódigo deles se pode ver nas seguintes imagens:

```
BUBBLE-SORT(A, n)
 1: for i ← 1 to n do
 2:   for j ← i + 1 to n do
 3:     if A[j] < A[i] then
 4:       troca(A[i], A[j]);
 5:     end if
 6:   end for
 7: end for
```



```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

INSERTION SORT

Linha 5 até 7: $T(n) = j - 1$

Linha 1 até 8: $T(n) = \sum_{j=2}^{n} (j-1)$

Resolvendo o somatório

$$T(n) = \sum_{j=2}^{n} (j-1)$$

$$T(n) = \sum_{j=2}^{n} j - \sum_{j=2}^{n} 1$$

$$T(n) = \frac{n(n+1)}{2} - 1 - (n-2+1)$$

$$T(n) = \frac{n(n+1)}{2} - 1 - n + 1$$

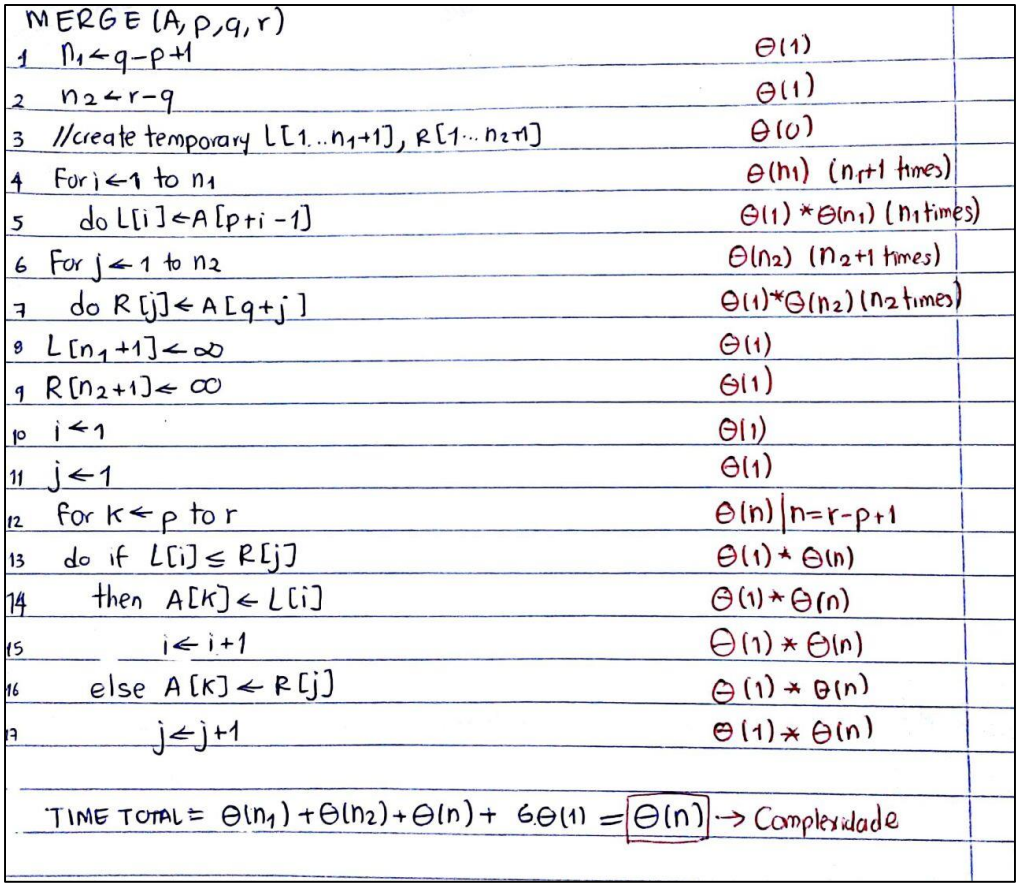
$$T(n) = \frac{n(n+1)}{2} - \frac{2n}{2}$$

$$T(n) = \frac{n^2+n-2n}{2}$$

$$T(n) = \frac{n^2-n}{2}$$

$$T(n) = \frac{n(n-1)}{2}$$

Complexidade: $O(n^2)$



Gráficos de comparação:

Os seguintes gráficos expõem a comparação dos tempos de execução analisados em cada um dos algoritmos com cada tipo de entrada.

ARRANJO ALEATÓRIO

A continuação, se expõem os dados e o comportamento dos algoritmos com um arranjo aleatório (caso médio).

Entrada	Booble Sort	Insertion Sort	Merge Sort	Quick Sort	Shell Sort
1000	0,00238462	0,00115385	0	0	0
5000	0,07207692	0,018	0	0,00123077	0,00361538
10000	0,33053846	0,07807692	0,00476923	0,00238462	0,00476923
100000	37,5715385	7,06569231	0,02161538	0,01676923	0,02892308
500000	945,612615	176,697846	0,07844615	0,108	0,16853846
1000000	3777,68154	701,516538	ERRO	0,26892308	0,38107692
1500000	8479,6	UNCOLLECTED	ERRO	0,46038462	0,583

O gráfico 1 mostra a comparação de todos os algoritmos:

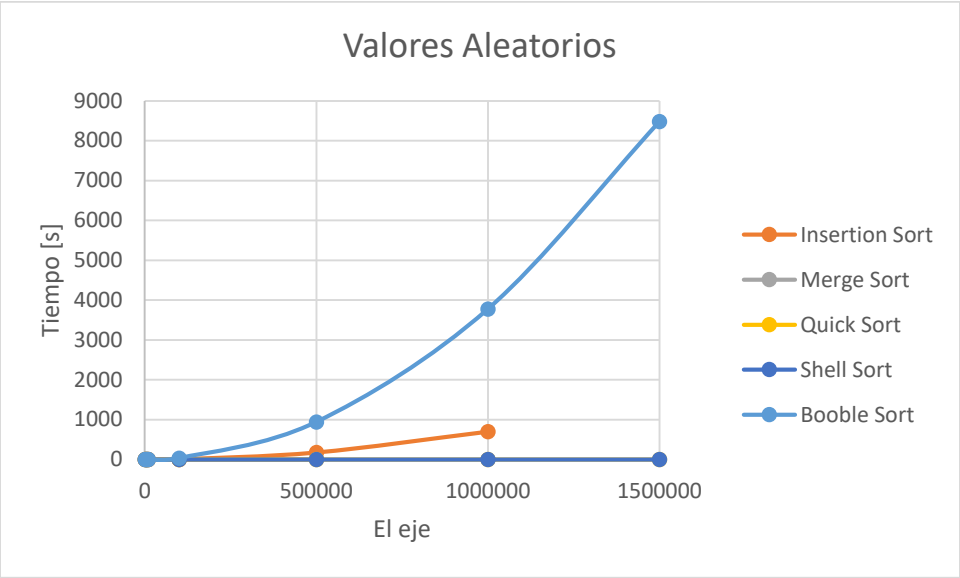


Gráfico 1. Valores Aleatórios

Se pode observar no gráfico 1 que o algoritmo que mais tempo gasta em ordenar os dados é o Bubble Sort.

A continuação se mostrara um gráfico para os algoritmos Merge Sort, Quick Sort e Shell Sort, o qual expõe suo comportamento com mais detalhe.

Gráfico 2. Valores Aleatorios (Merge Sort - Quick Sort - Shell Sort)

Devido a erros de locação de memória, não se tem os últimos dados de tempo para o algoritmo Merge Sort, porem, no gráfico se pode ver que o seu comportamento e mais ótimo que dos outros.

ARRANJO CRESCENTE

No gráfico 2 se expõe o comportamento dos algoritmos com entradas de arranjos ordenados (melhor caso).

Entrada	Booble Sort	Insertion Sort	Merge Sort	Quick Sort	Shell Sort
1000	0	0	0	0,00361538	0
5000	0	0	0	0,07092308	0
10000	0	0	0	0,29430769	0
100000	0,00023077	0,00123077	0,01330769	ERRO	0,00476923
500000	0,00238462	0	0,06253846	ERRO	0,042
1000000	0,00476923	0,00123077	ERRO	ERRO	0,06992308
1500000	0,00484615	0,006	ERRO	ERRO	0,11892308

Os algoritmos Merge Sort e Quick Sort tiveram erro devido a mal manejo da locação de memória. A continuação se mostra o gráfico com os dados coletados.

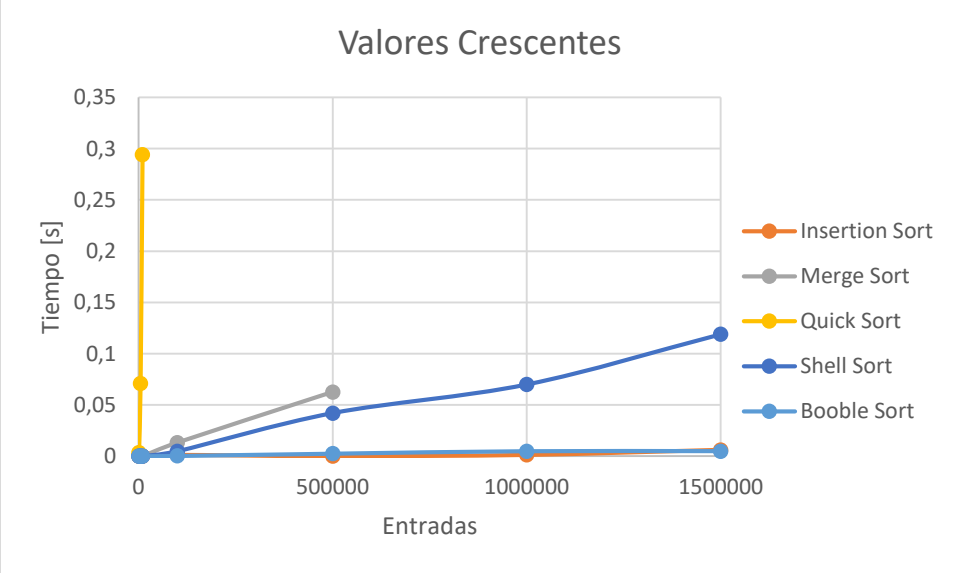


Gráfico 3. Valores Crescentes

Todos os algoritmos mostram um comportamento constante aproximado exceto o Quick Sort, porém nós não temos certeza dos dados devido aos erros gerados.

ARRANJO DECRESCENTE

Os seguintes resultados fazem parte da análise dos algoritmos trabalhando com um arranjo ordenado decrescentemente (pior caso).

Entrada	Booble Sort	Insertion Sort	Merge Sort	Quick Sort	Shell Sort
1000	0,006	0,003538462	0	0,00115385	0
5000	0,076923077	0,038384615	0	0,05169231	0,00115385

10000	0,298153846	0,140692308	0	0,20546154	0,00115385
100000	29,71192308	14,02669231	0,01207692	ERRO	0,006
500000	741,726	350,282	0,06123077	ERRO	0,03607692
1000000	2979,208889	1406,745	ERRO	ERRO	0,10346154
1500000	UNCOLLECTED	UNCOLLECTED	ERRO	ERRO	0,167

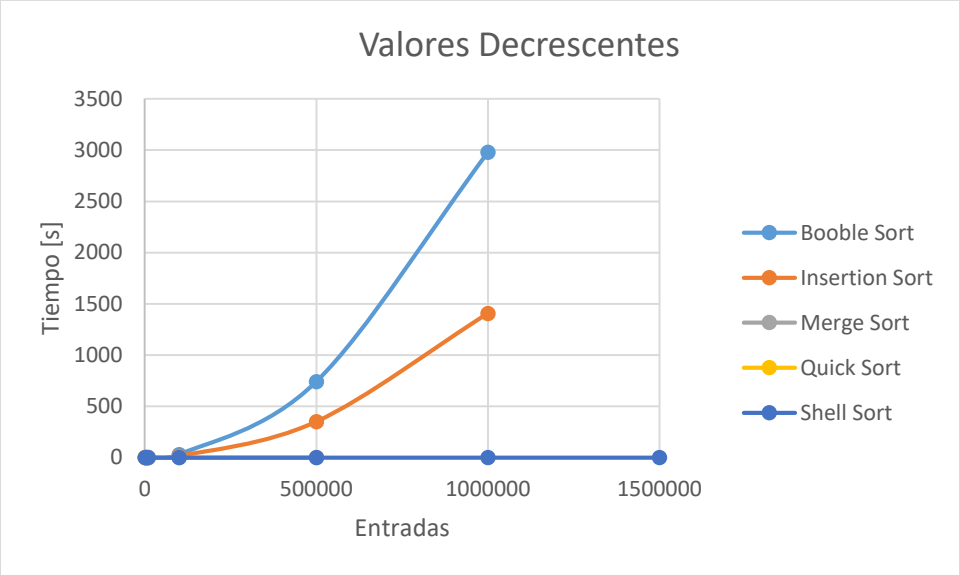


Gráfico 4. Valores decrescentes

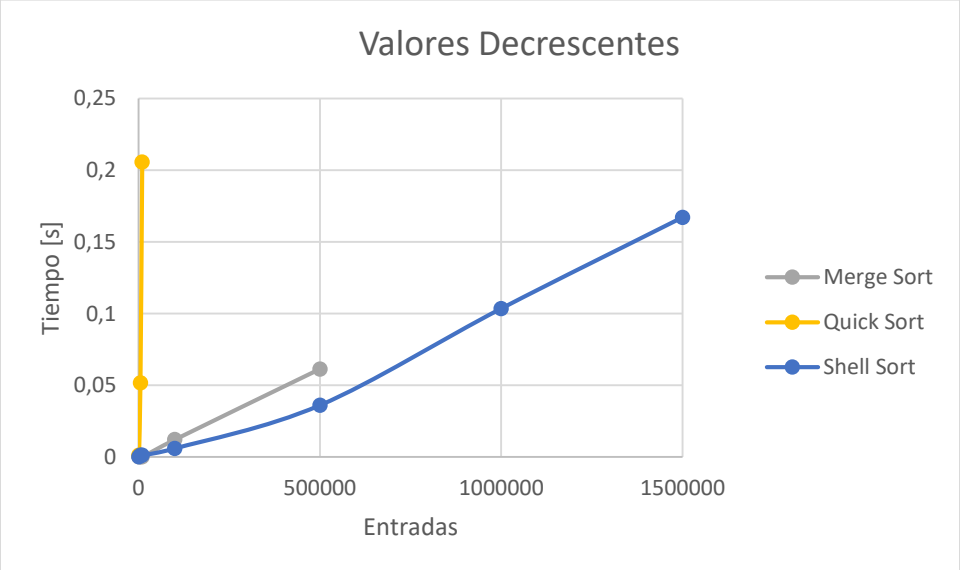


Gráfico 5.Valores decrescentes (Merge Sort - Quick Sort - Shell Sort)