Algoritmos e Técnicas de Programação II

# Relatório Projeto 5: Ordenação

**Prof.** Aleardo Manacero Junior **Aluno:** Gabriel Henrique da Silva Gava

## 1. Introdução

Na computação, ordenação é o ato de se colocar os elementos de uma sequência de informações, ou dados, em uma ordem predefinida. Existem diversos algoritmos conhecidos para ordenação de elementos, cada um com diferentes complexidade de tempo.

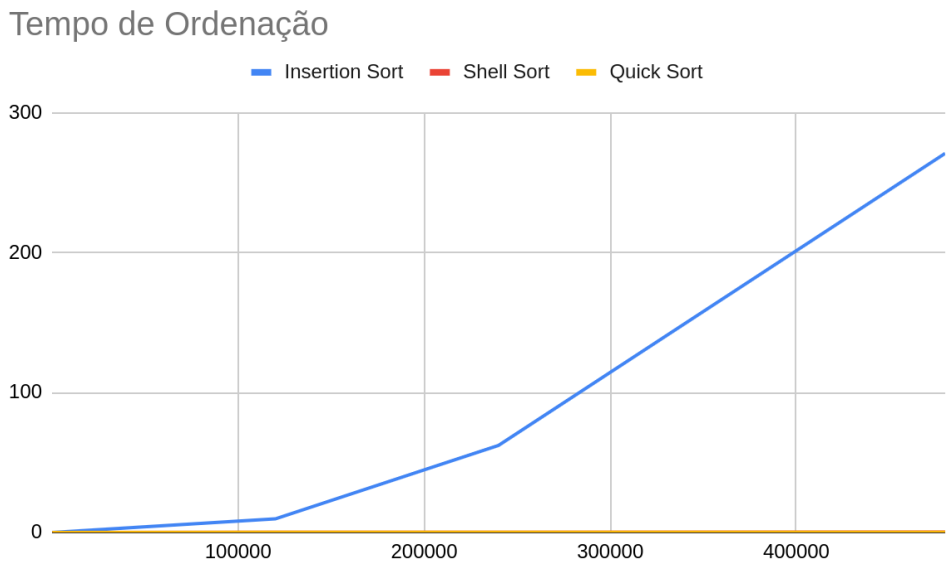
## 2. Objetivo

Neste trabalho foi desenvolvido um software em C capaz de receber um arquivo de entrada com uma lista de números, ordená-los de acordo com o método de ordenação selecionado e também efetuar uma busca a partir de outro arquivo. Analisaremos o tempo levado para três tipos diferentes de ordenação: Insertion Sort, Shell Sort e Quick Sort. Além disso, também analisaremos o tempo de busca para o algoritmo sequencial e binário. Por fim, iremos analisar a diferença entre cada método com relação a sua entrada.

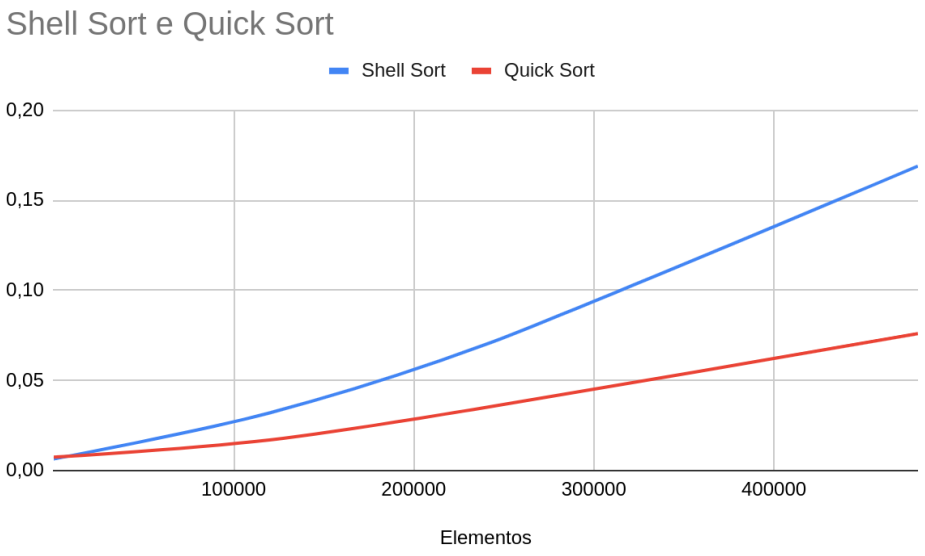
## 3. Resultados

Para realização dos testes foi utilizado um processador Ryzen 1800x, com oito núcleos de  3.6GHz. Primeiramente, analisaremos o tempo levado por cada algoritmo para ordenar de forma crescente cada um dos conjuntos de dados, iniciando com 50 e finalizando com 480.000 elementos. Para cada combinação, foi testado 5 vezes e feito uma média entre os valores obtidos em cada teste. Todos os resultados estão em segundos.

| N de Elementos | Insertion Sort | Shell Sort | Quick Sort |
|---|---|---|---|
| **50** | 0,005311 | 0,006463 | 0,007413 |
| **120.000** | 9,648 | 0,032 | 0,017 |
| **240.000** | 62,142 | 0,07 | 0,035 |
| **480.000** | 270,953 | 0,169 | 0,076 |

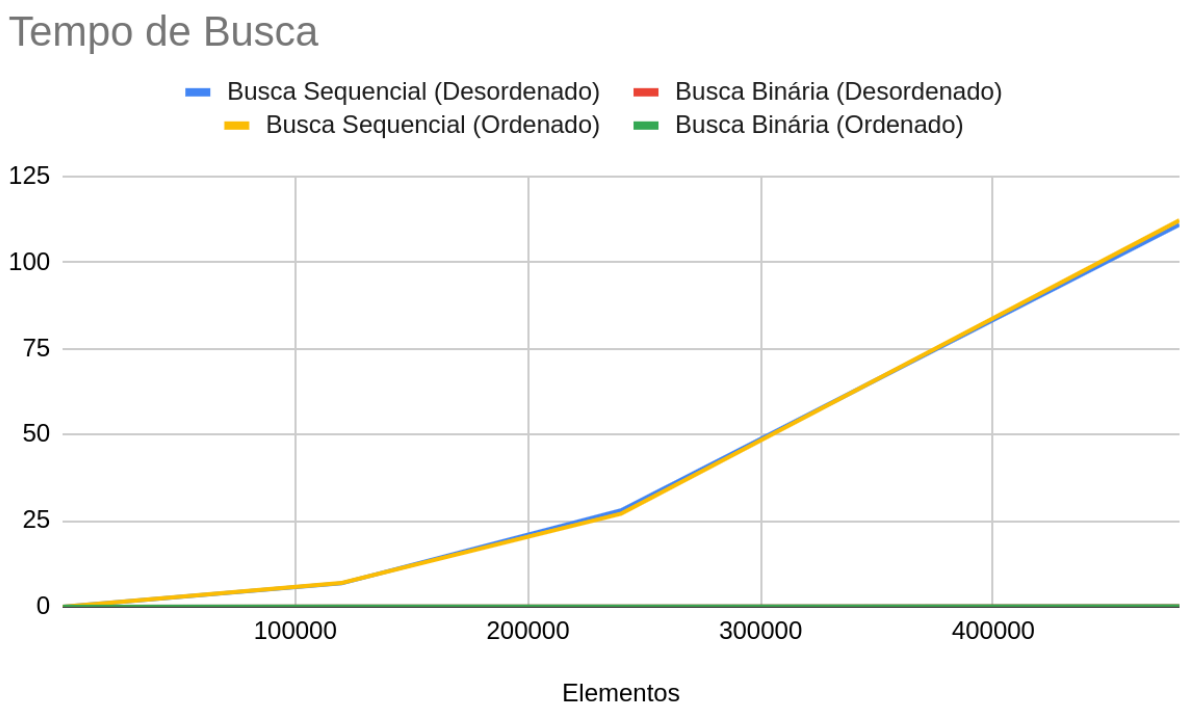



Como pode ser observado, no gráfico acima é difícil identificar a diferença de tempo entre o algoritmo ShellSort e o Quicksort devido a grande diferença de tempo gasto entre eles e o InsertionSort. Por esta razão, foi separado em um novo gráfico apenas estes dois métodos de ordenação



Em seguida, vamos analisar o tempo demandado para realizar os dois métodos de busca em um vetor ordenado e outro desordenado.

| Quantidade de Elementos no Vetor | Busca Sequencial (Desordenado) | Busca Binária (Desordenado) | Busca Sequencial (Ordenado) | Busca Binária (Ordenado) |
|---|---|---|---|---|
| 50 | 0,000369647 | 0,000425684 | 0,000416814 | 0,000377047 |
| 120.000 | 6,79 | 0,031 | 6,876 | 0,035 |
| 240.000 | 27,948 | 0,068 | 27,02 | 0,068 |
| 480.000 | 110,916 | 0,146 | 112,164 | 0,148 |




Como pode ser observado, não existe diferença de tempo significativa em ambos os métodos de busca caso o vetor já esteja ordenado ou não. Também podemos verificar a diferença de tempo para cada método, enquanto a Busca Sequencial leva 110 segundos para fazer a busca, a Binária leva apenas 0,146 segundos, uma diferença de até 753 vezes. **Obs: Um detalhe importante é que a busca binária assume que o vetor já está ordenado, por esta razão os resultados gerados em um vetor desordenado não serão corretos.**

## 4. Conclusão

Neste trabalho conseguimos observar a grande diferença de tempo de ordenação para cada um dos três algoritmos. Como o Insertion Sort possui uma complexidade de tempo de igual a $O(n^2)$, em comparação ao ShellSort com $O(n\,log_2\,n)$ e o QuickSort com $O(n\,log\,n)$, é natural que ele demande mais tempo para finalizar em relação aos outros dois. Quando utilizado um vetor de 480.000 elementos o QuickSort é 3565 vezes mais rápido comparado ao InsertionSort. Apenas no caso com 50 elementos em que o InsertionSort demonstrou realizar a ordenação mais rápido que os demais.

Também podemos observar a diferença de tempo entre a busca sequencial e a busca binária, onde a busca binária leva 743 vezes menos tempo que a busca sequencial (com 480.000 elementos no vetor). Além disso, podemos concluir que a diferença de tempo de busca com o vetor ordenado ou desordenado é praticamente nula nesses casos, como pode ser observado nos gráficos da seção anterior.