

# TRABALHO DE ORDENAÇÃO

Disciplina SIN 213 - Projeto de Algoritmos

**Data de Entrega:** 27 de Setembro de 2023

**Valor:** 8 Vistos

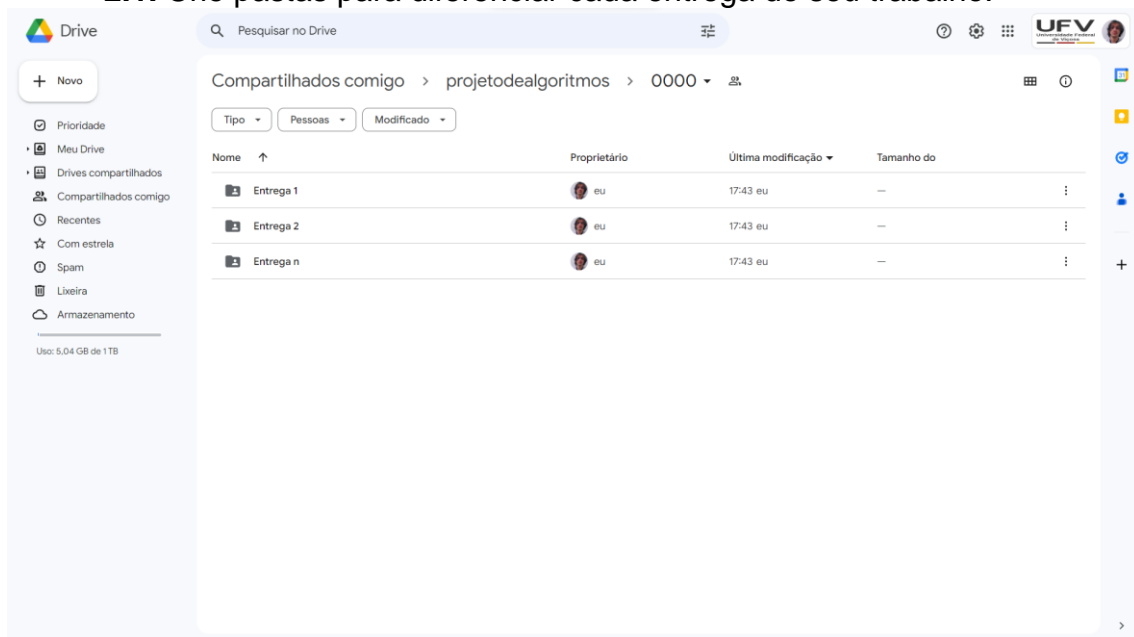
## 1. Entrega:

**1.1.** A entrega dos códigos (Código fonte e arquivos de entrada, tempo e o de saída, para cada instância do trabalho testando cada algoritmo deste trabalho) deverão ser colocadas em suas respectivas pastas do Drive, diferenciando por pastas de cada entrega.

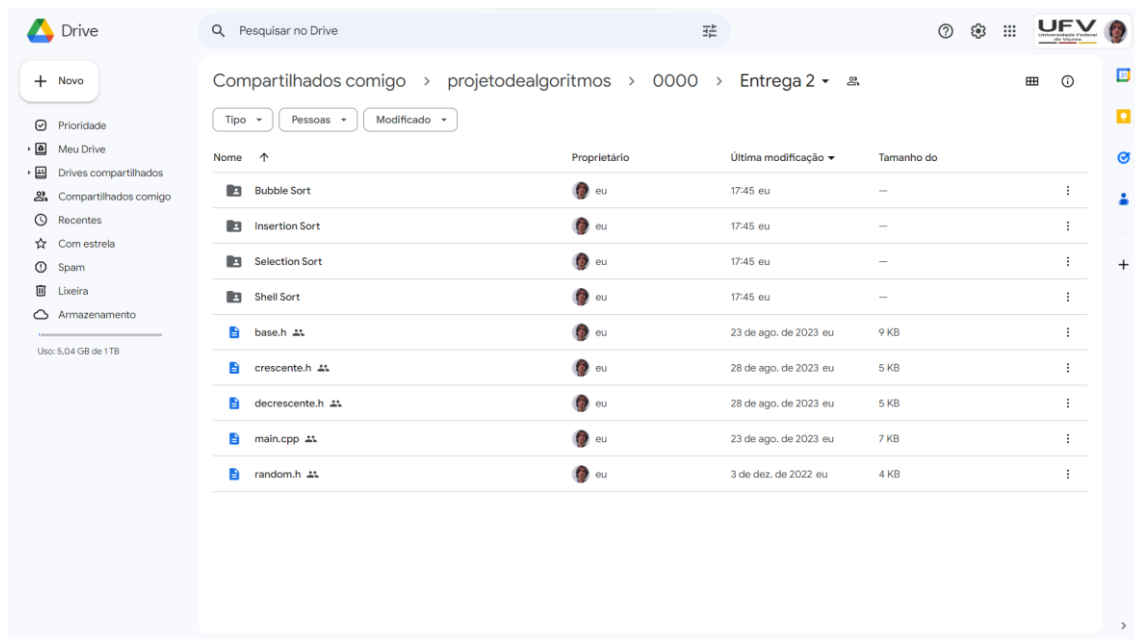
**1.2.** A entrega do relatório deverá ser enviada no Moodle em formato PDF.

## 2. Exemplo de Organização:

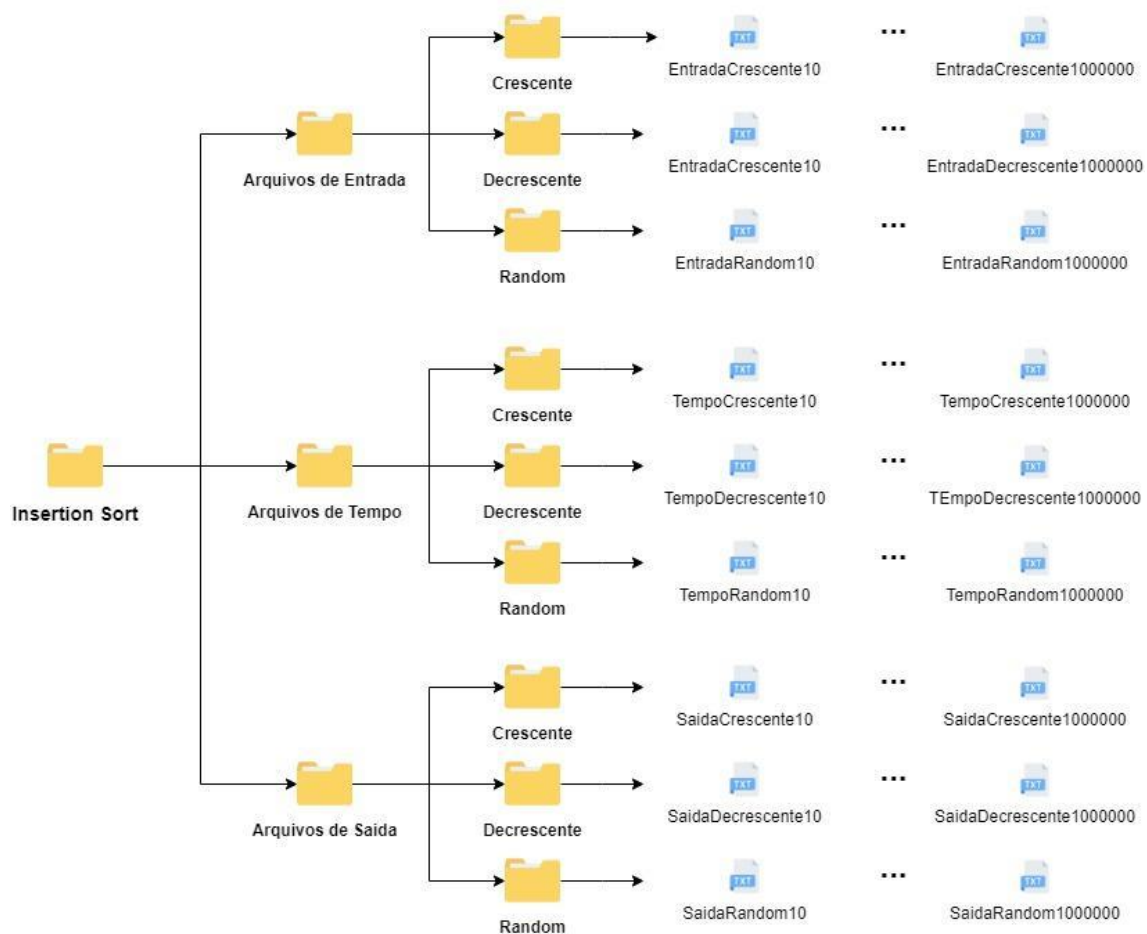
**2.1.** Crie pastas para diferenciar cada entrega do seu trabalho.



**2.2** Dentro da pasta deve ser enviado o seu código fonte e as pastas dos respectivos Algoritmos feitos, contendo os arquivos de entrada, saída e tempo conforme a imagem abaixo.



**2.3. Cada pasta deve seguir o seguinte formato: NÃO ESQUEÇA QUE DEVE SER FEITO A CRIAÇÃO DAS PASTAS E ARQUIVOS PELO CÓDIGO.**



Faça a criação do mesmo para o **insertion**, **selection**, **bubble** e **shellsort**, para poder fazer o relatório. **NÃO ESQUEÇA QUE DEVE SER FEITO A CRIAÇÃO DAS PASTAS E ARQUIVOS PELO CÓDIGO.**

**Sugestão:** Gere as entradas em um vetor e salve elas no arquivo de entrada com seu respectivo tamanho. Em seguida faça a ordenação do vetor, calculando o tempo gasto e salvando ela em um arquivo de tempo com seu respectivo tamanho. Por último salve o vetor organizado no arquivo de Saída com o seu respectivo tamanho.

### 3. Ordenação: Selection Sort , Bubble Sort, Shell Sort e Insertion Sort:

Ordenação é uma atividade necessária a vários sistemas de informação. Há vários algoritmos de ordenação, cada um deles com pontos fortes e pontos fracos. Os algoritmos **bolha**, **Inserção** e **Seleção** tem complexidade  **$O(N^2)$**  com relação a número de comparações de chave, mas dependendo dos dados, ambos podem ser **mais rápidos do que algoritmos com complexidade menor**.

#### 3.1. Objetivo:

3.1.1. Experimentar algoritmos.

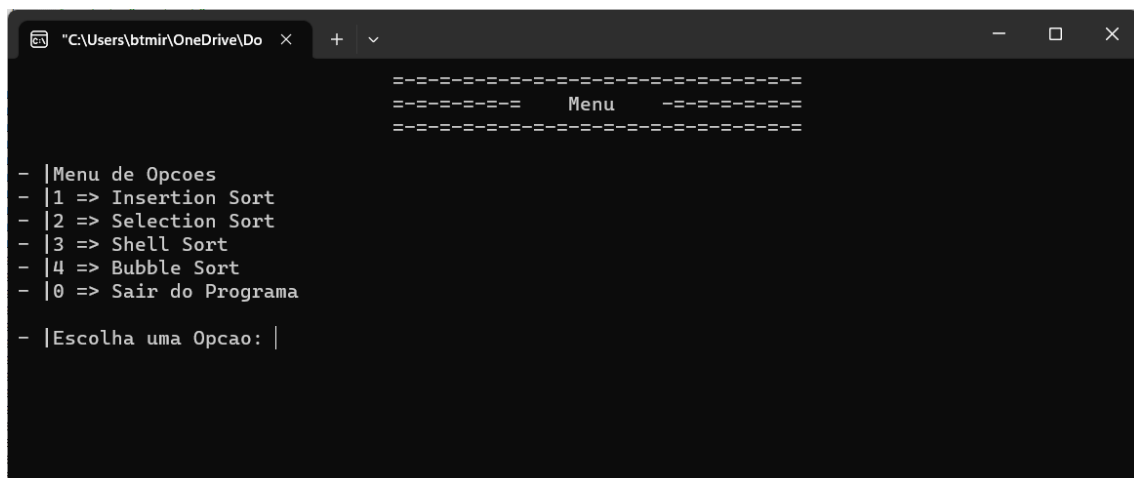
3.1.2. Comparar os tempos dos experimentos.

3.1.3. Analisar os tempos e especular as circunstâncias em que um algoritmo se sobressai ao outro.

#### 3.2. Código:

Implemente em **linguagem C ou C++** os algoritmos de ordenação **Insertion**, **Selection** , **Shell** e **Bubble**. Usando vetor e arquivos.

3.2.1. **Sugestão de Menu:** implemente um **Menu** na aplicação para melhor interação com o usuário. Este Menu poderá conte:



```
"C:\Users\btmir\OneDrive\Do  × + v
-----
Menu
-----

- |Menu de Opcoes
- |1 => Insertion Sort
- |2 => Selection Sort
- |3 => Shell Sort
- |4 => Bubble Sort
- |0 => Sair do Programa

- |Escolha uma Opcao: |
```

```
"C:\Users\btmir\OneDrive\Do x + v
=====
----- Tipo de Entradas -----
=====

- |Menu de Opcoes
- |1 => Crescente
- |2 => Decrecente
- |3 => Randomico
- |0 => Sair do Programa

- |Escolha uma Opcao: |
```

```
"C:\Users\btmir\OneDrive\Do x + v
=====
----- Tamanhos -----
=====

- |Menu de Opcoes
- |1 => 10
- |2 => 100
- |3 => 1.000
- |4 => 10.000
- |5 => 100.000
- |6 => 1.000.000
- |0 => Sair do Programa

- |Escolha uma Opcao: |
```

**3.2.2 Geração de Entradas:** O código deve gerar instâncias de tamanhos: **10, 100, 1.000, 10.000, 100.000 e 1000,000** para todas as formas (**randômico, crescente e random**), e execute o algoritmo implementado.

**3.2.3 Geração de Pastas e Arquivos:** O código deve gerar automaticamente as pastas e arquivos, como citado no tópico **2.3**, salvando seus respectivos valores de cada.

Na parte de arquivo, gere:

- Um **Arquivo de Entrada** contendo as instâncias geradas.
- Um **Arquivo de Saída** com as instâncias ordenadas.
- Um **Arquivo de Tempo** gasto pela ordenação.

Os arquivos tanto os de **Entrada** como o de **Saída** devem ter na primeira linha o **tamanho da instância** e o restante dos dados, um por linha, como é indicado na Figura 1. O **Arquivo de Tempo** salve apenas o tempo final gasto em segundos, ou formate em sua preferência.

Os arquivos tanto os de entrada como os de saída devem ter na primeira linha o tamanho da instância e o restante dos dados, um por linha, como é indicado na Figura 1

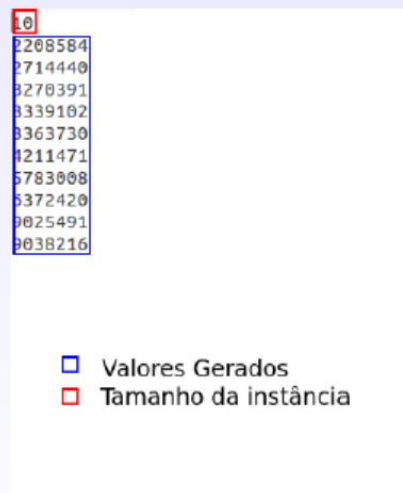


Figura: Exemplo de arquivo de entrada.

Figura 1: Exemplo de arquivo de entrada.

## 4. Resultado Esperado

4.1. Crie um relatório seguindo os tópicos abaixo:

- Capa e Contra capa;
- Resumo;
- Sumário;
- Introdução ;
- Algoritmo;
  - Algoritmo x;
- Análise de Complexidade;
  - Algoritmo x;
- Tabela e Gráfico;
- Conclusão ;
- Referências;

Faça uma boa **introdução** de **uma página** falando sobre como funciona cada algoritmo de forma resumida.

**Provar a Análise de Complexidade** de cada algoritmo em seus respectivos casos: o melhor, médio e o pior caso. ( exatamente como foi feito em sala de aula para o insertion sort, coloque um custo para cada linha e o número de vezes que ela irá ser executada e faça os cálculos)

*Obs: Pesquisar a análise da complexidade do algoritmo Shellsort e implementar no relatório.*

É necessário fazer a **conclusão** que você obteve sobre os algoritmos, e comparar caso possível com os demais algoritmos feitos uns com os outros apontando suas qualidades e defeitos e explicando o porquê isso acontece, comparando as entradas, tempos, e algoritmos e casos.

Fazer uma tabela de comparação dos algoritmos **insertion** , **selection**, **bubble** e **shellsort**. Deve ser feito a execução do código proposto gerando as saídas de tempo das instâncias: 10, 100, 1.000, 10.000 , 100.000 e 1.000.000 que serão utilizadas para fazer a tabela de comparação e os gráficos.

Faça um gráfico **para cada algoritmo** e também um **gráfico geral** contendo os algoritmos propostos. Cada um dos gráficos deve possuir duas curvas: uma representando o comportamento do algoritmo quando submetidos a dados em ordem decrescente e outra quando submetido a dados já ordenados.

Gráfico do Algoritmo X

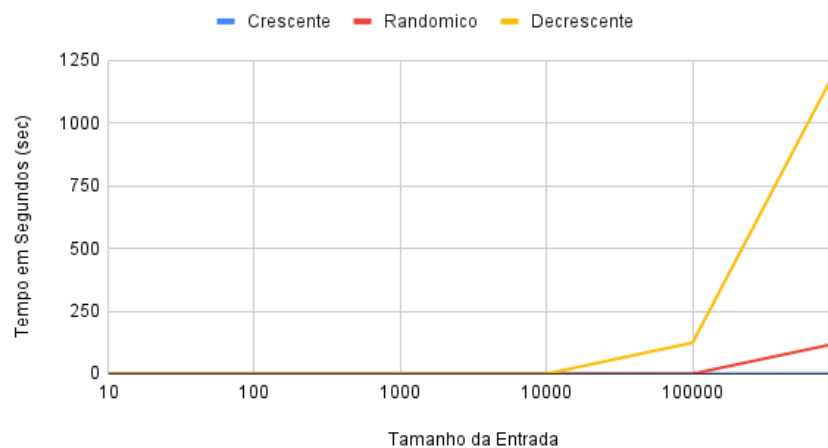


Figura 2: Exemplo de Gráfico do Algoritmo X com dados fictícios.

Gráfico Geral

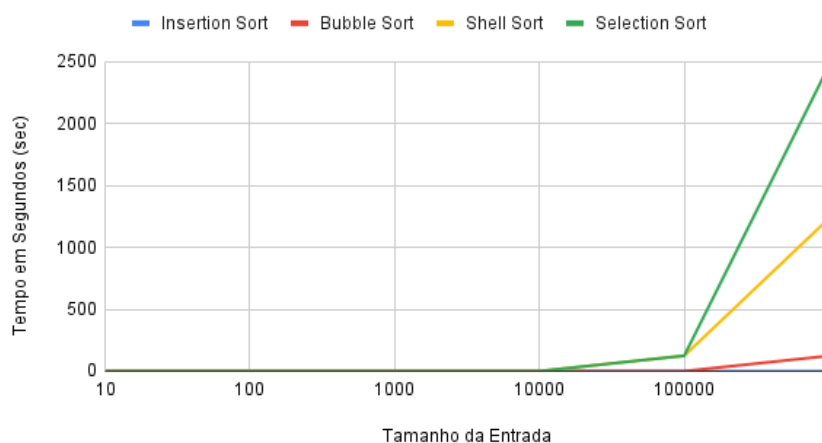


Figura 3: Exemplo de Gráfico Geral usando resultados de cada algoritmo usando dados fictícios.