TRABALHO DE ORDENAÇÃO

Disciplina SIN 213 - Projeto de Algoritmos

Data de Entrega: 10 de novembro de 2023
Valor: 8 Vistos

1. Ordenação: Algoritmo1, Merge Sort e Quick Sort*

Algoritmo mais popular de ordenação. Na maioria dos casos ele roda em $\mathcal{O}(n \log n)$ para ordenações internas ou em memória. Para ordenar informações em arquivos em disco existem métodos melhores

1.1. Objetivo:

3.1.1. Experimentar algoritmos.

3.1.2. Comparar os tempos dos experimentos.

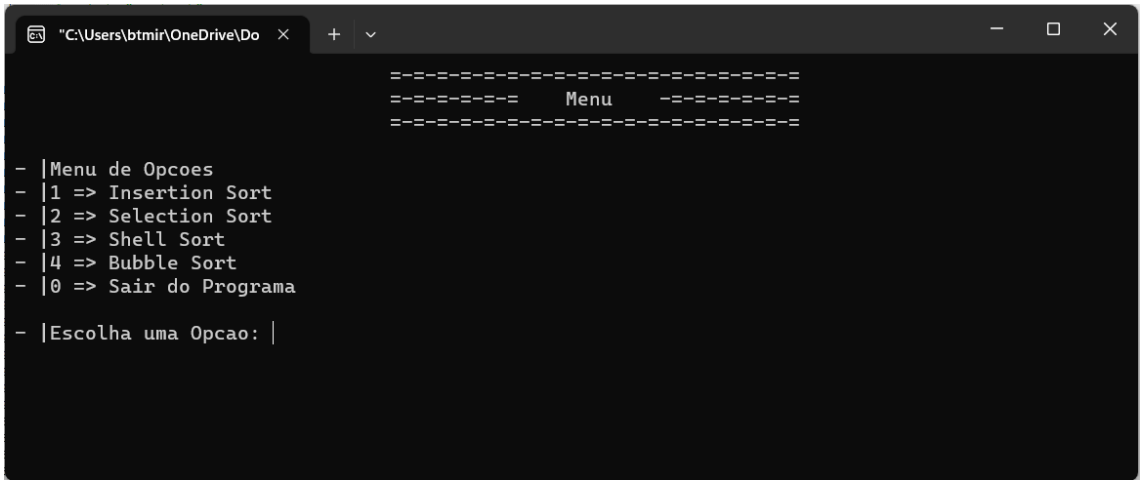
3.1.3. Analisar os tempos e especular as circunstâncias em que um algoritmo se sobressai ao outro.

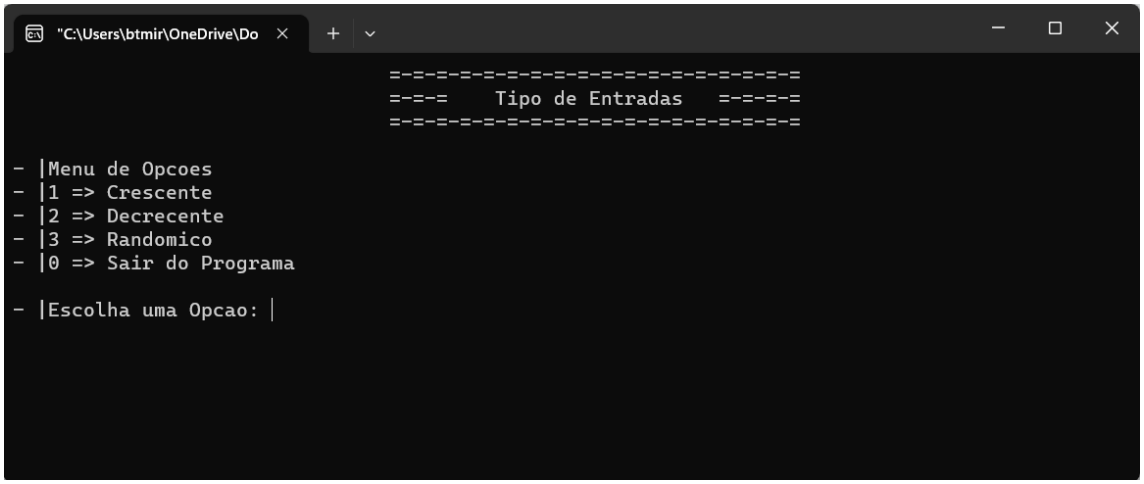
1.2. Código:

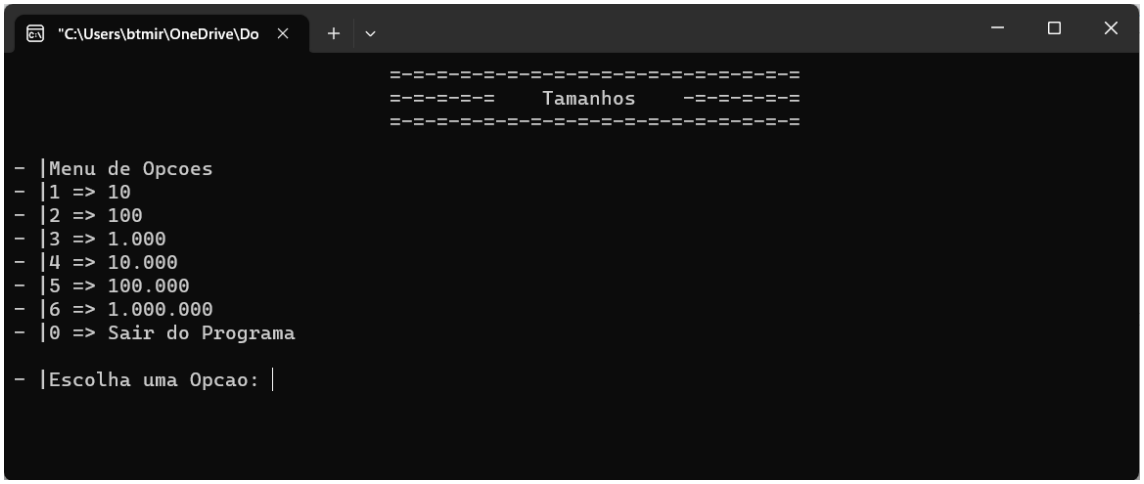
Implemente em **linguagem C ou C++** os algoritmos de ordenação (Usando vetor e arquivos):

- **Algoritmo1** *(Obs: o Algoritmo1 você deverá escolher 1 entre os algoritmos estudados: Insertion Sort, Selection Sort, Bubble Sort, etc.)*
- **Merge Sort**
- **Quick Sort**
    - **Quicksort_versao1** (algoritmo normal apresentado em sala de aula utilizando o primeiro elemento como pivô ou o do livro página 118, ou o do autor Ziviani).
    - **Quicksort_versao2** (segundo algoritmo utilizando a média para melhorar a escolha do pivô).
    - **Quicksort_versao3** (segundo algoritmo utilizando a mediana de três elementos para melhorar a escolha do pivô).
    - **Quick_versao4** (escolhendo o pivô de forma randômica).

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







1.2.2 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 decrescente)**, e execute o algoritmo implementado.

12.3 Geração de Pastas e Arquivos: O código deve gerar automaticamente as pastas e arquivos, 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.

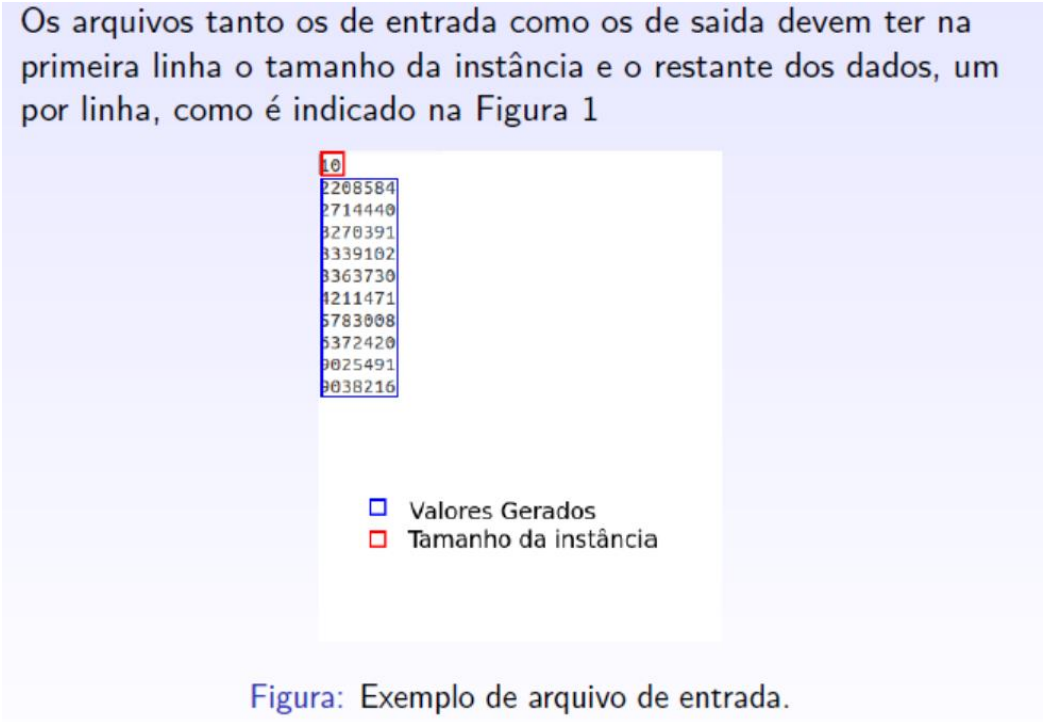


Figura 1: Exemplo de arquivo de entrada.

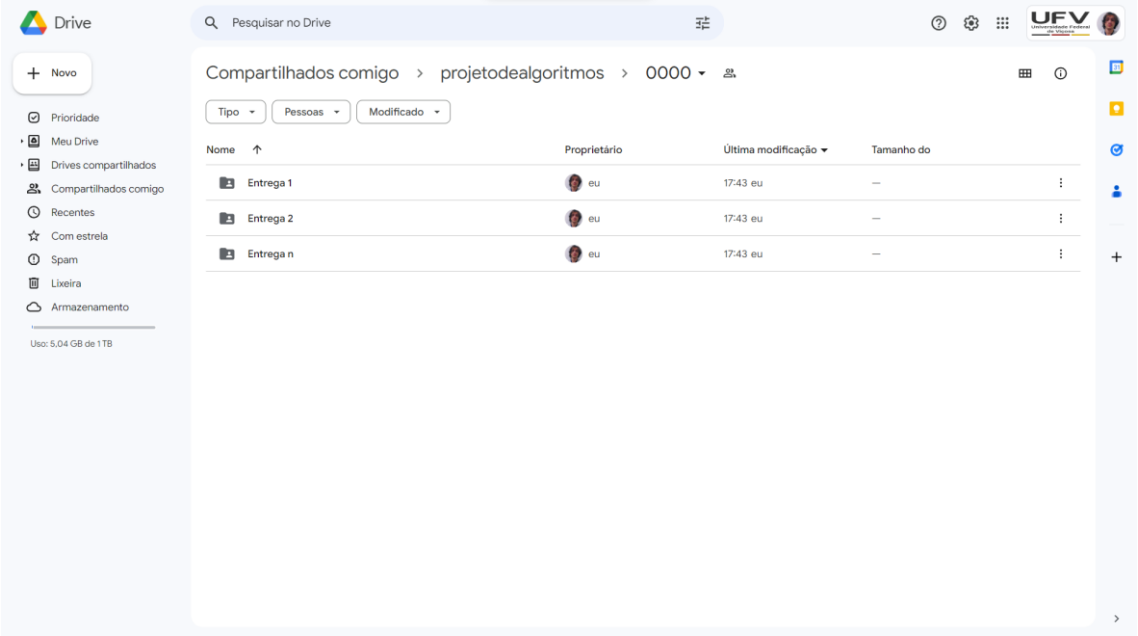
2. Entrega:

2.1. 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.

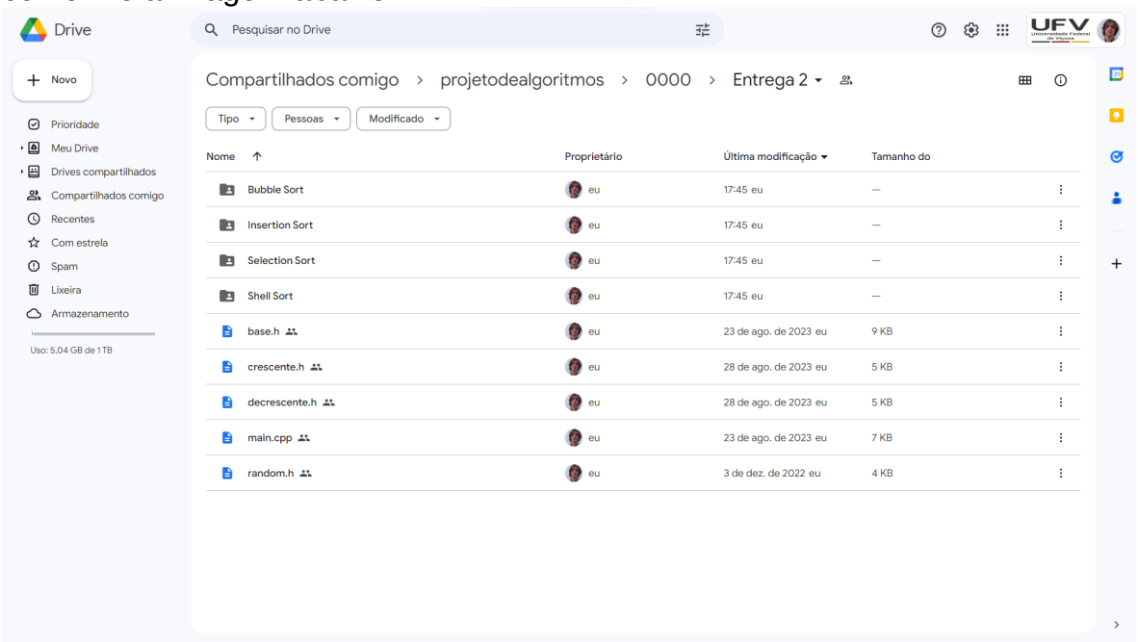
2.2. A entrega do relatório deverá ser enviada no Moodle em formato PDF.

3. Exemplo de Organização:

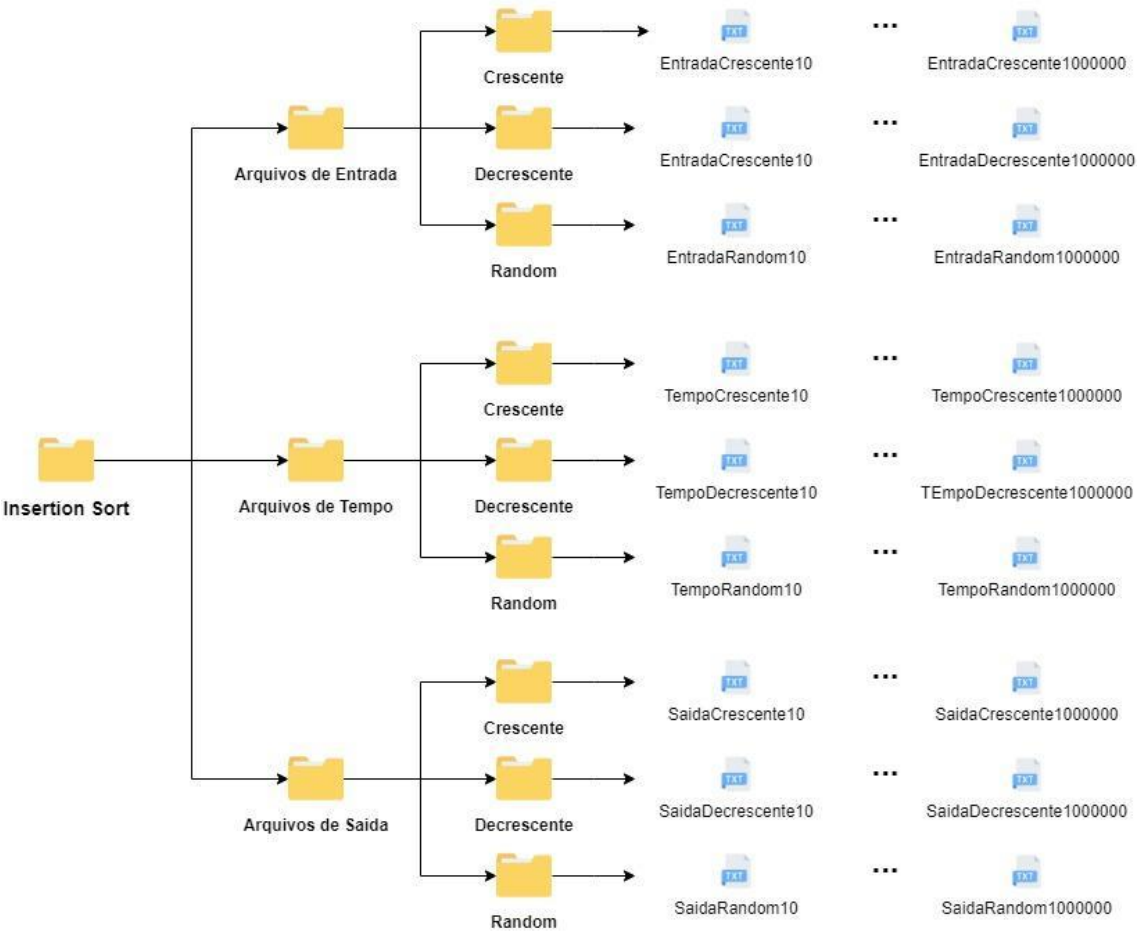
3.1. Crie pastas para diferenciar cada entrega do seu trabalho.



3.2 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.



3.3. 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 os outros algoritmos para a realização do 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.

4. Resultado Esperado

4.1. Crie um relatório seguindo os tópicos abaixo:

- Capa e Contracapa;
- Resumo;
- Sumário;
- Introdução;
- Algoritmo;
  - Algoritmo x;
- Análise de Complexidade;
  - Algoritmo x;

- Tabelas e Gráficos;
- 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)

Para o caso do Quick Sort, explicar se tem ou não a fase de combinação e o porquê.

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.

Faça uma tabela **para cada algoritmo**. 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 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.

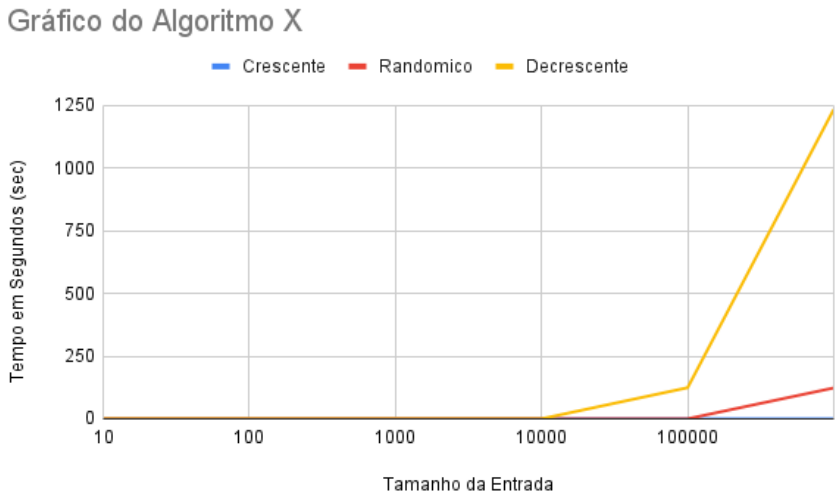


Figura 2: Exemplo de Gráfico do Algoritmo X com dados fictícios.

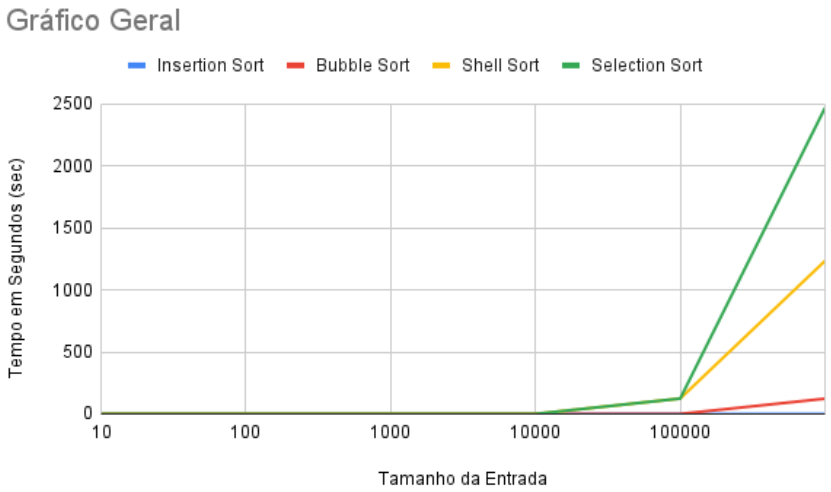


Figura 3: Exemplo de Gráfico Geral usando resultados de cada algoritmo usando dados fictícios.