Universidade Federal de Viçosa
Roteiro do Projeto (40 pts)

SIN 213 - Projeto de Algoritmos
Professor: Clausius Duque Gonçalves Reis
email: clausius@ufv.br
Sala LAE 279

April 19, 2021

Informações Gerais

1. Este projeto DEVE ser desenvolvido individualmente.
2. Os algoritmos solicitados devem ser implementados na linguagem Python.
3. A **ÚNICA** função do Python que pode ser importada é a função "**time**". No final desse texto deixei um exemplo.
4. Sempre que o aluno mencionar um dos códigos no relatório, referencie o mesmo pelo nome do arquivo do código fonte (ex. arquivo1.py).
5. A entrega deste trabalho deve ser realizada até o dia **09 de Maio de 2021 até às 23:59** via PVANet na aba - **Projeto 2021**.

Exercício 1 - (20 Pontos)

1. Considerando os algoritmos apresentados nas **Aulas 4 e 5 (selection sort, insertion sort, shell sort** e **merge sort)**, implemente estes algoritmos na linguagem Python e execute-os sobre os casos de teste fornecidos na página da disciplina (**ArquivosOrdenacao.zip**) seguindo a descrição abaixo:
   (a) Com 10, 1000, 10000, 100000 e 1000000 de elementos (todas com elementos em ordem aleatórias);
   (b) Com 10, 1000, 10000, 100000 e 1000000 de elementos (todas com elementos em ordem crescente);
   (c) Com 10, 1000, 10000, 100000 e 1000000 de elementos (todas com elementos em ordem decrescente);
2. Façam o aferimento do tempo de execução de cada algoritmo para cada entrada dos casos de teste (10, 1000, 10000, 100000 e 1000000 de elementos). Escreva um breve relatório **discutindo** os resultados obtidos. Elaborem **um gráfico** e planilhas para comparar o comportamento dos algoritmos de ordenação em cada um dos casos de teste.
3. Um **exemplo de relatório** (**ExemploRelatorio.pdf**) está disponível na página da disciplina.
4. Os códigos fonte e relatório deveram ser submetidos ao PVANet (em formato compactado) até a data estipulada nas informações gerais.

Exercício 2 - (20 Pontos)

1. Considerando o algoritmo de Huffman na Aula 17, o objetivo deste trabalho é criar um compactador/descompactador de arquivos em modo texto plano, ou seja, sem formatação.

2. Implemente na linguagem Python um sistema que, recebendo um arquivo texto (.txt) contendo uma cadeia de caracteres quaisquer, compacte este arquivo para um novo arquivo com extensão "**.dvz**" (Darth Vader Zip).

3. A primeira linha deste novo arquivo (extensão "**.dvz**"), deve conter as informações necessárias para recriar a **árvore de Huffman** usada na compressão do mesmo. O formato desta linha é proprietário, ou seja, a forma de gravação da árvore de Huffman no início do arquivo ".dvz" fica a critério do aluno. A partir da segunda linha, será armazenada a cadeia de caracteres comprimida do arquivo com extensão ".txt".

4. De posse do arquivo ".dvz", quero ser capaz de descompactar o mesmo, para isso, a árvore de huffmann deve ser recriada a partir da leitura da primeira linha do arquivo.

5. Façam o aferimento da "**taxa de compressão**" antes e depois da compressão (tamanho do arquivo em KB com duas casas decimais) pelo Darth Vader Zip para arquivos com diferentes número de caracteres, fornecidos na página da disciplina, com 1, 2, 3, 4 e 5 parágrafos (**ArquivosCompressor.zip**). Escreva um breve relatório **discutindo** os resultados obtidos. Elaborem **um gráfico** e planilhas para comparar o comportamento dos algoritmos de ordenação em cada um dos casos de teste.

6. Um **exemplo de relatório** (**ExemploRelatorio.pdf**) está disponível na página da disciplina.

7. Os códigos fonte e relatório deveram ser submetidos ao PVANet (em formato compactado) até a data estipulada nas informações gerais.

Exemplo de como calcular o tempo usando a função "time"

```python
def funcao():
    for i in range(10000):
        print(i)

import time

inicio = time.time()
funcao()
fim = time.time()
print(fim - inicio)
```