

**SIN110 – Algoritmos e Grafos**  
**Matheus Luz - 2020032426**  
**ATIVIDADE 1 (ATV1) : Entrega 29/08**  
**Código: [Github](#)**

A proposta é criar um protótipo de software que faça a leitura de um arquivo de uma dada instância, exiba os resultados obtidos e salve estes em um outro arquivo. No caso, foi necessário o uso das funções da biblioteca Numpy para a criação de matrizes e para a obtenção das dimensões destas.

O código foi estruturado conforme as indicações de organização do Slide 1, ou seja, dividindo as etapas / funções do software em pastas e arquivos para agilizar o desenvolvimento e facilitar o processo de modularização.

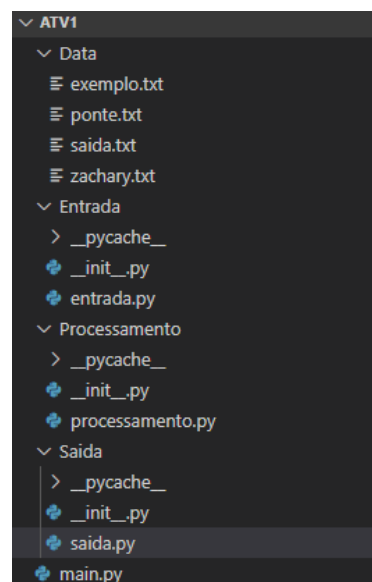


Figura 1. Estrutura do código.

Como pode ser observado, o código foi dividido em 4 pastas (além do arquivo main.py), sendo estas Data, Entrada, Processamento e Saída.

### **Pasta Data**

É onde armazeno os arquivos que serão lidos pelo software e a saída (saida.txt) que armazena os resultados obtidos após o processamento das matrizes e obtenção das suas dimensões.

### **Pasta Entrada**

É onde armazeno o arquivo responsável por ler o conteúdo das instâncias e retornar uma matriz numpy como resposta para a main. O nome do arquivo que será aberto é

recebido como parâmetro e será usado após a concatenação para encontrar este no disco. O arquivo é aberto, a função “loadtxt” do Numpy é usada e o resultado é armazenado na variável matriz. Por fim, o arquivo é fechado e a matriz é retornada.

```
1  # Importação de bibliotecas e/ou arquivos
2  import numpy as np
3
4  # A função abre o arquivo cujo nome é
5  # parâmetro, lê o conteúdo e retorna
6  # uma matriz numpy como resposta
7  def entrada(nome):
8
9      # Armazena a localização do arquivo
10     local = 'Data/' + nome + '.txt'
11
12     # Abre o arquivo para leitura ('r')
13     arquivo = open(local, 'r')
14
15     # A função 'loadtxt' armazena os dados
16     # do arquivo na matriz Numpy
17     matriz = np.loadtxt(arquivo)
18
19     # Fecha o arquivo
20     arquivo.close()
21
22     # Retorna a matriz
23     return matriz
```

Figura 2. Arquivo entrada.py

### Pasta Processamento

É onde armazeno o arquivo responsável por processar a matriz que foi obtida anteriormente no arquivo de entrada. O nome do arquivo que foi aberto e a matriz obtida são usados de parâmetro, então a quantidade de linhas e colunas são obtidas através da função “len” e são armazenadas nas variáveis de nome iguais. Por fim, o código exibe o nome do arquivo e as dimensões da matriz e retorna as linhas e colunas para a main.

```
1  # A função obtém a quantidade de linhas
2  # e a quantidade de colunas da matriz
3  # passada como parâmetro, depois exibe
4  # no console e retorna os valores
5  def processamento(nome, matriz):
6
7      # Armazena a quantidade
8      # de linhas da matriz
9      linhas = len(matriz[0])
10
11     # Armazena a quantidade
12     # de colunas da matriz
13     colunas = len(matriz)
14
15     # Exibe os valores
16     print(nome + ' (' + str(linhas) + ', ' + str(colunas) + ')') + '\n')
17
18     return linhas, colunas
```

Figura 3. Arquivo processamento.py

### Pasta Saída

É onde armazeno o arquivo responsável por armazenar os resultados obtidos através do processamento da matriz Numpy. De modo simples, a função recebe o nome do arquivo que foi lido, a quantidade de linhas e colunas, então armazena estes no arquivo de saída e fecha este

```
1 # A função é responsável por armazenar
2 # os resultados obtidos do processamento
3 # da matriz Numpy. Os resultados são
4 # as dimensões da matriz, ou seja,
5 # a quantidade de linhas e colunas
6 def saida(nome, linhas, colunas):
7
8     # Abre o arquivo para escrita ('a+')
9     arquivo = open('Data/saida.txt', 'a+')
10
11     # Faz a escrita no arquivo
12     arquivo.write(nome + ' (' + str(linhas) + ', ' + str(colunas) + ') ' + '\n')
13
14     # Fecha o arquivo
15     arquivo.close
```

Figura 4. Arquivo saida.py

### Arquivo main.py

É onde fica a função principal do software, nesta os arquivos necessários para o funcionamento são inicialmente importados. Dentro da função principal, foi criado um laço para processar os três arquivos “.txt” que deveriam ser usados como exemplo. Então as funções de entrada, processamento e saída são chamados em ordem respectiva.

```
1 # Importação de bibliotecas e/ou arquivos
2 import Processamento as p
3 import Entrada as e
4 import Saída as s
5
6 # Função principal
7 if __name__ == '__main__':
8
9     # Loop para automatizar os testes
10    for i in range(3):
11
12        # Condicional para definir
13        # qual arquivo será aberto
14        if i == 0:
15            nome = 'exemplo'
16
17        elif i == 1:
18            nome = 'ponte'
19
20        else:
21            nome = 'zachary'
22
23        # 'matriz' é a matriz que será
24        # obtida a partir da função de entrada
25        matriz = e.entrada(nome)
26
27        # A função obtém a quantidade de linhas
28        # e a quantidade de colunas da matriz,
29        # exibe as dimensões obtidas na tela e
30        # retorna estes valores
31        linhas, colunas = p.processamento(nome, matriz)
32
33        # A função
34        s.saida(nome, linhas, colunas)
```

Figura 5. Arquivo main.py