Algoritmos e Grafos - Relatório de atividade

Atividade 1 - Preparação do ambiente para atividades

> Joel Vítor Torres de Andrade Pinto (2021003527)
> Sistemas de Informação

> Data máxima de entrega: 29/08

Para acessar o código disponível no github clique aqui

A atividade consiste na elaboração de um código que seja capaz de ler um arquivo armazenando uma instância, contabilizar a quantidade de linhas e colunas da última e, então, mostrar essa informação no terminal, também armazenando o resultando em um arquivo.

Usei a biblioteca **Numpy**, a qual possui métodos especializados para realização desse tipo de atividade.

Estrutura do repositório

```
projeto/
├─ instances/
├─ Output/
│  ├─ output_results.py
├─ Read/
│  ├─ get_input.py
├─ main.py
```

Detalhamento do código

```
 # main.py
# Os comentários originais foram ocultados para uma melhor visualização
def main():
    datasets: str = sys.argv[1]

    # Nesse trecho de código, eu itero pelos nomes das instancias que recebi como parametro
    for dataset in datasets.split(','):
        # Utilizo as funções que criei para cada instancia
        matrix, shape = read_instance(dataset)

        nrows, ncols = shape

        result = f'{dataset};{nrows};{ncols}'

        # E faço uma pequena formatação para a apresentação do resultado no terminal
        print(result.replace(';', ' = ', 1).replace(';', ' '))

        output_results([result], 'results.csv')
```

Se tratando da leitura do arquivo, fiz uma breve pesquisa na documentação do Numpy, buscando uma maneira de fazer leitura de arquivos com representações de grafos no formato de matriz. Como resultado, encontrei o método np.loadtxt(), usado

abaixo.

```
 # Read/get_input.py
# As duas linhas seguintes foram essenciais para a execução da atividade

# O método loadtxt() foi utilizado para ler o arquivo
# com formato de matriz de adjacência ou incidência
matrix = np.loadtxt(file_path)

# Já o método shape() me retorna uma tupla que representa a forma (nesse caso as dimensões)
# de um np.array, que no caso armazenou a matriz lida pelo método acima
matrix_shape = np.shape(matrix)
```

O método np.shape() já era de meu conhecimento, visto que a um tempo atrás explorei um pouco a biblioteca Numpy, então foi só uma questão de usá-lo passando a matriz do tipo np.array como parâmetro.

```
 # Read/get_input.py
# Ainda nesse arquivo, eu fiz uma pequena verificação para que pudesse obter
# as dimensões (nrows e ncols) corretamente

# Caso o tamanho da tupla fosse 1, o np.array se trataria de um array unidimensional
# e a saída do meu programa não estaria no formato correto. Por isso converti a saída
# em uma tupla de tamanho 2, com 1 linha e matriz_shape[0] colunas
if len(matrix_shape) == 1:
    return matrix, (1, matrix_shape[0])
```

Implementar a parte de leitura foi simples. Fiz uma função que aceita uma lista de strings, caso quisesse escrever mais de um dado em uma única execução, e então iterei sobre ela. A cada iteração, escrevi o dado em uma linha no arquivo determinado, isso por meio da função open() no modo de leitura **a+**.

```
 # Output/output_results.py

with open(file_path, 'a+') as f:
    for result in results:
        f.write(result + '\n')
```