Relatório - A1

Disciplina: INE5413 - Grafos
Discentes: Leonardo de Sousa Marques e Thayse Estevo Teixeira

1. [Representação]: Nesse algoritmo, utilizamos a biblioteca *dataclasses* do Python para criar classes para representar Vértice, Aresta e Grafo. A classe de Vértice não possui métodos, apenas os atributos índice: int, rótulo: str e vizinhos: List[Vértice]. Já a classe Aresta possui origem: Vértice, destino: Vértice, peso: float. A classe Grafo fará uso dessas outras estruturas para criar os métodos solicitados na atividade e demais métodos auxiliares.

2. [Buscas]: Nesse algoritmo, além da estrutura do grafo, também utilizamos uma fila, implementada como lista, para poder armazenar a ordem de acesso dos vértices. Além disso, utilizamos um dicionário, do tipo defaultdict, para guardar uma lista com vértices em determinado nível.

3. [Ciclo Euleriano]: Neste problema, além da estrutura do grafo, utilizamos o algoritmo de Hierholzer, indicado na apostila, e para isso utilizamos as estruturas de dados de dicionário, do tipo defaultdict e que tem como chave uma tupla de vértices (u, v) e armazena um valor inteiro que indica se a aresta já foi visitada. Para representar o ciclo, escolhemos a estrutura de dados de lista.

4. [Algoritmo de Bellman-Ford]: Nesse algoritmo, além da estrutura do grafo, também utilizamos a estrutura de dados de lista para representar o vetor de distâncias (D) e de predecessores (A). Como estavámos tratando de grafos não-orientados, é importante ressaltar que foi necessário aplicar o relaxamento em ambas direções.

5. [Algoritmo de Floyd-Warshall]: Nesse algoritmo, além da estrutura do grafo, também usamos um vetor de vetores de inteiros para representar a matriz D de distâncias mínimas entre todos os pares. Fizemos essa escolha pois é uma maneira prática de indexar e percorrer os valores por meio do cruzamento dos vértices que delimitam o caminho (os pares).