Relatório Atividade I

Fernanda Narloch, 18102712 14 de março de 2021

1 Representação

O trabalho foi desenvolvido em Python e, por isso, a representação do grafo foi feita através da classe *grafo.py*. O construtor recebe o conteúdo lido e já adaptado do arquivo e é estruturado da seguinte forma:

```
    def __init__(self, vertices, arestas, mapa_arestas):
        self.vertices = vertices
        self.arestas = arestas
        self.mapa_arestas = mapa_arestas
```

Assim, **vertices** é uma lista de tamanho n, guardando o rótulo de cada vértice do grafo. **arestas** é uma lista das duplas de vértices que possuem arestas e **mapa arestas** é a matriz de adjacência do grafo.

A primeira estrutura foi selecionada por ser uma forma simples de armazenar o rótulo do vértice junto com seu índice (por se tratar de uma lista em python, o valor do índice é a posição + 1). Já as arestas são armazenadas em duplas para facilitar no desenvolvimento de algoritmos que iteram pelas arestas, como Bellman-Ford. E, por último, o mapa arestas funciona como uma matriz de adjacência para a obtenção do peso dos vértices.

2 Buscas

Para a implementação da Busca em Largura, foi utilizada a estrutura de Grafo previamente criada bem como uma lista de vértices conhecidos, uma lista de distâncias e uma fila já utilizada no algoritmo tradicional. A lista de conhecidos é utilizada para avaliar se o algoritmo já percorreu o vértice, a de distância armazena em que nível está o vértice em questão a partir do inicial e a fila é utilizada para enfileirar os próximos vértices a serem analisados.

3 Algoritmo de Bellman-Ford

Neste algoritmo, era necessária a análise das arestas do grafo para encontrar o caminho mínimo de cada vértice com relação ao inicial. Para isso, utilizou-se a classe de Grafo criada, em especial a lista de arestas que foi concatenada com a permutação dos vértices de cada dupla para contemplar o fato de que o grafo é não-dirigido. Também foi utilizada uma lista de distância necessária para o algoritmo e uma lista de listas para armazenar o caminho mínimo de cada vértice.

4 Algoritmo de Floyd-Warshall

Para este caso foi utilizada a atribuição da matriz de adjacências para uma variável que representaria a distância para cada par de vértices. Assim, o algoritmo pode ser executado facilmente através da iteração sobre a matriz.

5 Informação adicional sobre o trabalho

O trabalho foi desenvolvido em Python 3 e os algoritmos podem ser rodados por linha de comando da seguinte forma:

```
python3 nome_algoritmo.py nome_arquivo_grafo [indice do vertice]
```