Relatório

Atividade 1 - INE5413

Maria Eduarda Hang de Melo
Thiago Sant' Helena

Questão 1 Nessa questão deveríamos implementar um tipo estruturado ou uma classe para representar um grafo. Nesta solução, criamos a representação na forma de uma classe, implementando todos os métodos requisitados no enunciado da atividade, exceto o método `ler()`, que foi implementado junto ao construtor da classe.

Para a nossa representação, utilimos a linguagem Python, versão 3.7, com as seguintes especificações:

- Matriz para representar os arcos/arestas e seus pesos

  Dessa forma, a verificação de ligação entre dois vértices e a do peso da aresta seria O(1)m de fácil implementação. O valor de cada M(u, v) equivale a um valor real finito caso exista a aresta e a representação de infinito em Python, dada por `float('inf')`, para arestas inexistentes.

- Um um `dict()` de Python, para mapear o identificador numérico de cada vértice para um rótulo especificado.

  Dessa forma, o acesso ao rótulo também é O(1), além de termos a contagem de vértices disponível no tamanho deste dicionário.

- Um contador inteiro, interno a classe, para manter o acompanhamento do número de arestas

  E portanto, termos o acesso ao número de arestas por O(1).

- O construtor recebe necessariamente como argumento o caminho para um arquivo que descreva o grafo como especificado no enunciado

  Assim desempenhando o papel da função `ler()` descrita no enunciado.

Nota: Para todas as questões a seguir, utilizamos um arquivo executável separado para cada uma, as instruções de execução destes está contida em cada um deles. Cada algoritmo utilizado para a resolução das questões está implementado no arquivo `functions.py` na pasta `src`.

Questão 2 Aqui deveríamos implementar o algoritmo de busca em largura, que está em `src/functions.py` com assinatura `bfs(G, s)`. Ao algoritmo é executado e no arquivo do programa é feito um pequeno tratamento para mostrar os dados na tela da maneira requisitada. Se nenhum ciclo euleriano foi encontrado, o valor mostrado na tela será `0`.

Questão 3 Nesta questão implementamos o algoritmo de Hierholzer dividido em 2 partes, com assinaturas `hierholzer(G)` e `search_subcicle(G, v, C)`. No programa apenas chamamos o primeiro, que chama o segundo, e tratamos o retorno para mostrar os dados na tela da maneira requisitada.

Questão 4 Nesta questão implementamos o algoritmo de Bellman-Ford, além de uma função auxiliar para organizar o retorno do algoritmo para ser mostrado na tela. Nenhum motivo em especial para termos implementado Bellman-Ford ao invés de Dijkstra, apenas que foi o visto primeiro em aula.

Questão 5 Aqui implementamos o algoritmo de Floyd-Warshall da maneira mais simples possível e organizamos a saída na tela.