Relatório — Atividade 1

Nome: Eduardo Vicente Petry **Matrícula: 17100513**

Questão 1) Representação

Implementei uma classe para representar grafos não-direcionados, sejam eles ponderados ou não. Os atributos da classe são: número de vértices, número de arestas, boolean que sinaliza se o grafo é ponderado, lista de vértices e matriz de arestas (ou de adjacências).

A lista de vértices armazena nas posições $i - 1$ os rótulos de cada vértice v_i. A matriz de arestas é uma matriz triangular inferior sem diagonal principal, visto que essa classe representa apenas grafos não-direcionados. Devido a esta otimização de memória, cada aresta $\{i, j\}$ é armazenada nas posições $(\max(i, j) - 2, \min(i, j) - 1)$ da matriz. Os elementos dessa matriz são 0's e 1's se o grafo for não-ponderado, sinalizando a existência das arestas. Caso o grafo seja ponderado, os elementos da matriz são os pesos das arestas, sendo infinito para arestas não existentes.

A classe contém todas as funções requeridas pelo enunciado. Vale mencionar que a função que lê arquivos de grafo recebe o nome dos arquivos pela execução atráves do *prompt de comando*. Os arquivos de grafo a serem lidos devem estar contidos na pasta *networks*.

Questão 2) Buscas

O programa recebe pelo *prompt de comando* o nome do arquivo de grafo a ser lido e o índice do vértice de origem, e então instancia um grafo não-ponderado. (Exemplo de execução: python busca_em_largura.py facebook_santiago.net 99)

São utilizadas três listas: de vértices já *conhecidos*, de *distâncias* até cada vértice e dos *antecessores* que levaram até cada vértice. Também é utilizada uma fila para armazenar as *visitas* durante a busca em largura. O algoritmo de busca em largura foi reproduzido tal como consta na p. 26 das Anotações da Disciplina. Por fim, foi utilizado um dicionário para indexar os vértices que foram percorridos em cada nível da busca, mostrando-os em ordem ascendente de níveis.

Questão 4) Algoritmo de Bellman-Ford

O programa recebe pelo *prompt de comando* o nome do arquivo de grafo a ser lido e o índice do vértice de origem, e então instancia um grafo ponderado. (Exemplo de execução: python bellman_ford.py fln_pequena.net 1)

São utilizadas três listas: de vértices já *conhecidos*, de *distâncias* até cada vértice e dos *antecessores* que levaram até cada vértice. O algoritmo de Bellman-Ford foi reproduzido tal como consta na p. 50 das Anotações da Disciplina. Por fim,

percorreram-se todos os vértices do grafo, reconstruindo os caminhos mínimos através da lista de antecessores e mostrando-os juntamente com suas distâncias.