UNIVERSIDADE FEDERAL DE SANTA CATARINA
Departamento de Informática e Estatística - INE
Ciências da Computação
INE5413 - Grafos

RELATÓRIO TÉCNICO

André William Régis (19200411)
Luiz Maurício do Valle Pereira (21104157)

Florianópolis, 28 de abril
2023

1 Introdução

Para visualizar o código fonte desse trabalho é necessário acessar github.com/Luiz276/grafos.

Para a execução do trabalho é necessário a instalação do `Poetry`. A seguir para a instalação das dependências é necessário executar na pasta do projeto:

```
$ poetry install
```

Finalmente para a execução do programa na raiz do projeto utilizasse:

```
$ poetry run python graph OPTION [VERTEX] [FILE...]
```

Ou para executar diretamente com o interpretador de python (dado que não há dependências indispensáveis):

```
$ python graph OPTION [VERTEX] [FILE...]$
```

Sendo `OPTION` substituído por um número e FILE por um caminho para o arquivo do grafo:

1. Apresenta as informações básicas do grafo
2. Apresenta a busca em largura
3. Apresenta a existência, ou não, de ciclo euleriano
4. Apresenta Bellman-Ford
5. Apresenta Floyd-Warshall

Sendo `[VERTEX]` podendo ser o índice ou label do vértice desejado. Essa opção é obrigatória para as opções 2 e 4, demais opções não aceitam a seleção de um vértice.

E `[FILE...]` admite nenhum, um ou vários arquivos de grafos, sendo os caminhos relativos à raiz do projeto. Caso nenhum arquivo seja especificado, o programa utilizará um grafo padrão "test/graph1.txt". "test/graph2.txt" é um grafo que contém ciclo euleriano, usado para testar o terceiro exercício.

2 Estrutura de Dados

Vértices, arestas e o grafo como um todo foram representados por classes, como explicado a seguir:

- Vertex: Representa um vértice, contendo duas variáveis:
  - index - Armazena o index de um vértice
  - label - armazena o label de um vértice
- Edge: Representa uma aresta não orientada do grafo, armazenando vértices de origem e destino
- Graph: classe que representa um grafo, com uma lista de vértices V chamado de "vertices", uma lista de arestas E chamada de "edges", e uma lista "weights" que mapeia o peso de cada aresta.
  - Seus métodos representam operações esperadas de um grafo.

  - Listas foram escolhidas devido a representarem os dados de maneira mais simples e intuitiva, mesmo não sendo o caso ótimo para todas as operações realizadas.

Cada uma das estruturas de dados foram escolhidas por representarem de maneira intuitiva e eficiente os dados do programa, tornando o código o mais próximo possível dos pseudocódigos de exemplo passados em aula. Durante o desenvolvimento do código, testes unitários foram utilizados para auxiliar no desenvolvimento de algumas partes. Testes se encontram no arquivo "`test/teste_ex1.py`".

3 Questões resolvidas

Todas as 6 questões foram resolvidas.

- Questão 1: Métodos da classe Graph implementam as operações pedidas, apenas a operação de criar um grafo a partir de um arquivo é resolvida no arquivo `reader.py`, através da função `import_graph()`.
- Questão 2: Resolvida no arquivo `t1_e2_breadth_first.py`, através da função `breadth_first_search`, que realiza a busca em largura em um grafo utilizando as variáveis:
  - `researcheds`: Para lembrar dos vértices já visitados
  - `depths`: Para lembrar o nível de cada vértice visitado
  - `max_depth`: Para saber o nível máximo (e otimizar a impressão dos dados)
- Questão 3: Resolvida no arquivo `t1_e3_eulerian_cycle.py`, através da função `eulerian_cycle`, que realiza a operação em um grafo uitilizando as váriaveis:
  - `edges_reached`: Para lembrar das arestas já visitas
  - `cycle`: Para armazenar os vértices do ciclo em ordem
  - `subcycle`: Para armazenar os vértices de um subciclo em ordem
- Questão 4: Resolvida no arquivo `t1_e4_bellman_ford.py`, através da função `search_minimal_` que realiza a operação em um grafo utilizando as variáveis:
  - `distances`: Um dicionário que mapeia a menor distância do vértice de origem selecionado e o vértice armazenado como chave
  - `antecessor` Um dicionário que mapeia um vértice até seu antecessor no caminho mínimo partindo do vértice selecionado como original, tanto o vértice chave (selecionado) como o valor (antecessor) são armazenados pelo índice, devido a limitação de `dict` em python.
- Questão 5: Resolvida no arquivo `t1_e5_floyd_warshall.py`, através das funções `print_floyd_warshall()` e `floyd_warshall_from_file()`, que realizam a operação em um grafo pré-existente e em um grafo em arquivo, respectivamente. Itens de interesse no código:
  - `dist`: Matriz que armazena as distâncias entre os vértices. Cada index dessa matriz representa o index de um vértice, de acordo com a regra a seguir: vértice 1 = index 0, vértice 2 = index 1, ... , vértice n = index n-1