Assignment 2
Teoria de Grafos e Computabilidade

Iyan Lucas Duarte Marques[1], **Samir do Amorim Cambraia**[1]

[1]Instituto de Ciências Exatas e Informática - Pontifícea Universidade Católica Minas Gerais (PUC-MG)

1. Problema

"O problema de se determinar o número máximo de caminhos disjuntos em arestas existentes em um grafo apresenta várias aplicações. Neste trabalho você deverá implementar um método de resolução deste problema que receba um grafo e um par de vértices (isto é, origem e destino) exiba ao final a quantidade de caminhos disjuntos em arestas entre os dois vértices dados, além de listar cada um dos caminhos encontrados. "

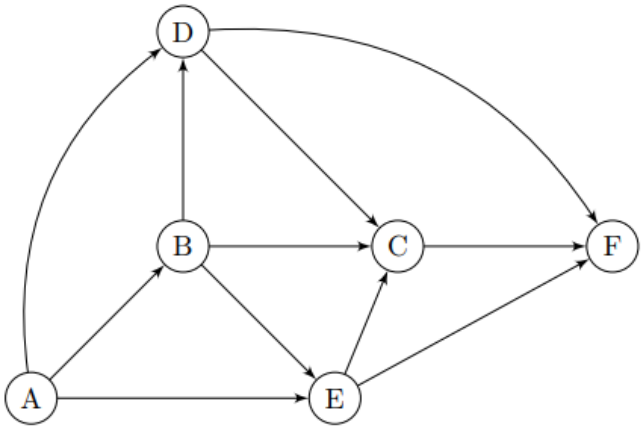

Figura 1: Exemplo de grafo com 6 vértices e 11 arestas.

2. Implementação

2.1. A classe

2.1.1. Construtor

O objeto recebe um object, que é na verdade um dicionário de dicionários, onde cada vértice é um dicionário. A chave do dict por sua vez é o rótulo do vértice e os values são os outros vértices onde há uma aresta que incide nos mesmos. Desta forma, se o object está vazio, é gerado um dicionário vazio como variável local da classe para substituí-lo.

2.1.2. Vertices e Arestas

Quando nós adicionamos um object na construção do objeto de classe, ele atribui automaticamente os vertices e arestas. Entretanto, quando quisermos adicionar um vértice, o

código simplesmente adiciona-o no dicionário de vértices. Já com as arestas, é um pouco mais complicado, o código recebe as arestas, sendo as arestas um parâmetro, e atribui como um valor no dicionário do vértice X. O valor é uma tupla, onde é o (incidente, origem) e guardado no object.

2.1.3. *Path*

Para achar um caminho, há o vertice inicial e o destino. Se o caminho existe e o vértice inicial não é o final, a função viaja recursivamente por cada vértice que há uma ligação. Ou seja, a função é chamada internamente passando pelo parâmetro o vértice atual, e o vértice de destino original e o array de caminho (que é anexado a cada iteração). Para achar todos os caminhos é literalmente a forma que se roda mais vezes só que o array de caminho diferente (isso possibilita a ele escolher caminhos que não foram selecionados) e então anexados em um array de caminhos.

3. Utilização

3.1. Execução

Para executar o código, precisa-se estar no diretório e digitar o comando *python main.py*.

3.2. Arquivos

- ***main.py:*** Arquivo que contem a declaração do grafo e a chamada dos métodos.
- ***grafo.py:*** Arquivo que contem a classe grafo declarada e com seus métodos.