Ciências da Computação – Universidade Federal de Santa Catarina

Prof: Rafael de Santiago

Alunos: Henrique Pereira, Luiz Slongo e Matheus Antonio

Relatório Atividade Prática A1 – Grafos (INE5413)

1. Representação

```python
class GrafoMatrizAdjacencia:
    def __init__(self):
        self.vertices = set()
        self.matriz = [[float('inf')] for _ in range(1)]
        self.nomes_vertices = {}
```

vertices: É utilizado o *set()* do python, por ser uma representação abstrata de um conjunto. Não há ordenação nem repetições, além de fornecer métodos úteis como *union()* e *intersection()*.

matriz: Matriz de adjacências é representada por uma lista de listas. Acesso e escrita aos elementos é feito em O(1), e a lista de listas é uma representação comum de uma matriz em python.

nomes_vertices: O nome dos vértices é armazenado em um dict do python. Ele mapeia os vértices(que são representados por inteiros) em *strings*. O acesso ao método *rotulo* da classe faz um *get* no dicionário, onde o acesso a elementos chave-valor é O(1)

2. Buscas

```
    c_visitados = [False for _ in range(grafo.qtdVertices()+1)]
    d_niveis = defaultdict(list)

    # c_visitados.add(s)
    c_visitados[s] = True
    fila = deque()
    fila.append((0, s)) # (nivel, vertice)
```

c_visitados: Uma lista de booleanos onde o elemento do indice **i** armazena a informação de se o vértice **i** já foi visitado pelo algoritmo. É utilizado a lista por ser mutável e de acesso O(1)

d_niveis: Um *defaultdict* de lista do python. É um *hashmap* com acesso O(1) a elementos chave-valor, onde os valores possuem como padrão uma lista mesmo quando não inicializados. As chaves representam o nível, e os valores representam quais vértices estão no determinado nível.

fila: a fila é criada com *deque()*, utilizando os elementos *popleft()* para retirar da fila e *append()* para adicionar ao final da fila. Para busca em largura é utilizado uma fila, ao contrário da busca em profundidade que utiliza uma pilha.

3. Ciclo Euleriano

Para o ciclo Euleriano, foi desenvolvido um programa baseado no algoritmo de Hierholzer disponibilizado na apostila da disciplina, onde são determinados o método Hierholzer() e o buscarSubcicloEuleriano().

Ce: Dicionário que armazena todas as arestas visitadas do grafo como chaves e são setadas inicialmente como False.

Ciclo: Lista que armazena os vértices do Ciclo Euleriano.

4. Algoritmo de Bellman-Ford ou de Dijkstra

Na questão 4, foi escolhido o algoritmo de Dijkstra. O primeiro método do programa em python foi implementado de acordo com o algoritmo de

Dijkstra apresentado em aula e retorna dois dicionários: das distâncias e dos antecessores de cada vértice. Após, o método reconstruir_caminho utiliza estes valores para reconstruir o caminho mínimo percorrido. Por fim, no método imprimir_resultados, são impressos o número de cada caminho, os vértices ordenados percorridos, e a distância do vértice de origem para cada vértice alcançável do grafo.

5. Algoritmo de Floyd-Warshall

```
    # Inicializa matriz de distancias com peso das arestas
    D_matriz = [[grafo.peso(u, v) for v in range(grafo.qtdVertices()+1)] for u in range(grafo.qtdVertices()+1)]


    # Inicializa matriz de predecessores
    PI_matriz = [[None for _ in range(grafo.qtdVertices()+1)] for _ in range(grafo.qtdVertices()+1)]
```

D_matriz e PI_matriz: As matrizes de distanciamento e predecessores são representadas por lista de listas de inteiros, por serem mutáveis e uma representação comum em pyhon de matrizes 2D.