UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática
Prof. Dr. Ivan Carlos Alcântara de Oliveira
Disciplina: Teoria dos Grafos

Implementação

AGCM- Algoritmo de Kruskal ou Pim (Grupo do Projeto)

Nome do(a) aluno(a)	RA
Igor Benites Moura	10403462
Gabriel Gonzaga Chung	10403025
Rodrigo Machado de Assis Oliveira de Lima	10401873

A partir dos algoritmos de Kruskal e Prim estudados em aula sobre Árvore Geradora de Custo Mínimo, escolher um deles e fazer a implementação na linguagem de programação selecionada para o desenvolvimento do seu projeto. Para isso, utilize a classe grafo com matriz de adjacência ou lista de adjacência disponibilizada em aula.

Realizar pelo menos dois testes, sendo: 1) grafo do material da aula; e 2) grafo da atividade solicitada anteriormente, colocá-los no relatório a seguir. Em uma seção final do relatório, denominada Apêndice, inclua o código fonte para o algoritmo desenvolvido, devidamente documentado.

Os códigos fontes desenvolvidos e o relatório devem ser encaminhados pelo Ambiente Virtual compactados.

Relatório

Obs: usamos o início da contagem dos vértices ‘0’ ao invés do ‘1’.

1) Teste do grafo do material disponibilizado em aula:

```
Testando para um grafo do material de aula professor
([(0, 1), (1, 5), (5, 8), (1, 2), (2, 3), (3, 4), (4, 6), (4, 7)], 33)
 n:  9 m: 13
 ∞    4  ∞   ∞   ∞    5  ∞   ∞   ∞
 4  ∞    7  ∞   ∞    3  ∞   ∞   ∞
 ∞    7  ∞    5  ∞   ∞    6  ∞   ∞
 ∞   ∞    5  ∞    3  ∞   ∞   ∞   ∞
 ∞   ∞   ∞    3  ∞   ∞    2   4  ∞
 5   3  ∞   ∞   ∞   ∞    7  ∞    5
 ∞   ∞    6  ∞    2   7  ∞    6  ∞
 ∞   ∞   ∞   ∞    4  ∞    6  ∞    8
 ∞   ∞   ∞   ∞   ∞    5  ∞    8  ∞

fim da impressao do grafo.
```

2) Teste do grafo da atividade solicitada anteriormente (Implementação Djikstra ou Bellman-Ford):

```
Testando para um grafo da atividade anterior

([(0, 1), (1, 3), (3, 2)], 50)

 n:  4 m:  6

 ∞   20  30  50

 20  ∞   40  15

 30  40  ∞   15

 50  15  15  ∞

fim da impressao do grafo.
```

3) Teste de um grafo aleatório para esta atividade:

```
Testando para um grafo aleatório

([(0, 1), (1, 3), (1, 2), (3, 4)], 117)

 n:  5 m:  5

 ∞   20  30  ∞   ∞

 20  ∞   25  22  ∞

 30  25  ∞   ∞   ∞

 ∞   22  ∞   ∞   50

 ∞   ∞   ∞   50  ∞

fim da impressao do grafo.
```

Apêndice

Código fonte:

```
# Algoritmo de Prim
    def prim(self, origin):
        # Inicializa as variaveis
        V = list(range(self.n))
        T = [origin]
        E = []
        custo = 0

        # Versao iterativa em vez de recursiva
        while set(T) != set(V):

            # Atualiza o conjunto de vertices que nao estao em T
            KminusT = [v for v in V if v not in T]
            valor = float('inf')

            # Encontra a aresta de menor peso
            for k in T:
                for i in KminusT:
                    if self.adj[k][i] < valor:
                        valor = self.adj[k][i]
                        vint = k
                        vext = i
            # Adiciona o vertice novo ao conjunto T
            T.append(vext)

            # Adiciona a aresta nova ao conjunto E
            E.append((vint, vext))

            # Atualiza o custo
            custo += valor

        return E, custo
```