Caminho Mínimo e Árvore de Geradora Mínima entre Duas Cidades*

Lucas Henrique Pereira[1],Wagner Antonio Pereira [2] e Michel Gomes de Souza[3]
Coordenação do Curso de Bacharelado em Ciência da Computação - COCIC
Universidade Tecnológica Federal do Paraná - UTFPR
Campus Campo Mourão
Campo Mourão, Paraná, Brasil

[1]lucas.pereiracm@gmail.com
[2]lwagner_jimmy@hotmail.com
[3] michels@alunos.utfpr.edu.br

1 Introdução

Um grafo é uma estrutura que inclui um conjunto de objetos na qual contém vértices e arestas, aonde os vértices estão ou não relacionados entre si [5], como demonstra a Figura 1.

Esta estrutura pode ser usada para os mais diversos fins e aplicações, como a obtenção da melhor rota ou caminho de uma estrada, maximar ou minimizar o fluxo de pacotes de dados em uma rede, verificação de ciclos em um banco de dados *(DeadLock)* dentre outras finalidades.

Em nosso trabalho apresentaremos dois algoritmos que implementam técnicas relacionadas a grafos, um algoritmo gerador de árvore miníma, e outro algoritmo de caminho mínimo, utilizando os valores de passagens de ônibus para ser o peso das arestas e o nome das cidades como os vértices do grafo, empregando um *parser* em python para obter as informações do site ClickBus[2].

2 Obtenção dos dados

Para a obtenção dos dados utilizamos o pacote *parser* de HTML em python chamado BeautifulSoup[1], assim fazemos uma requisição do tipo GET para as URLS para conseguir os valores de passagem de uma cidade origem para uma cidade destino.

Cidades de origem e destino estão armazenadas em um Json juntamente com um link, exemplo de entrada Json pode ser vista no Código 1.

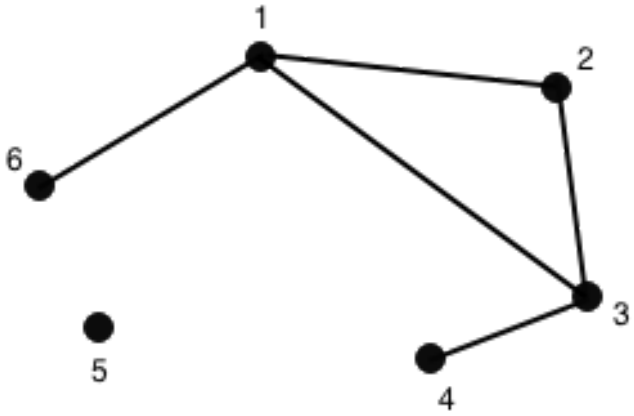

Figura 1: Exemplo simplificado de um Grafo

Código 1 - Exemplo de entrada para o Parser

```
1
2     "cidades": [{"origem": "Campo
          Mour o", "destino": "Peabiru", "
          link": "https://www.clickbus.com.
          br/onibus/campo-mourao-pr/peabiru
          -pr"},
3
4     {"origem": "Peabiru", "destino": "
          Eng Beltr o", "link": "https://
          www.clickbus.com.br/onibus/
          peabiru-pr/engenheiro-beltrao-pr?
          departureDate=2019-07-12"},
5
6     {"origem": "Campo Mour o", "destino
          ": "Maringa", "link": "https://
          www.clickbus.com.br/onibus/campo-
          mourao-pr/maringa-pr?
          departureDate=2019-07-12"},
7
```

*Trabalho desenvolvido para a disciplina de Grafos.

```
    {"origem": "Eng Beltr o", "destino"
        : "Ivailandia", "link": "https://
        www.clickbus.com.br/onibus/
        engenheiro-beltrao-pr/ivailandia-
        pr?departureDate=2019-07-12"},

    {"origem": "Ivailandia", "destino":
        "Maringa", "link": "https://www.
        clickbus.com.br/onibus/ivailandia
        -pr/maringa-pr?departureDate
        =2019-07-12"},

    {"origem": "Jussara", "destino": "
        Pai andu", "link": "https://www.
        clickbus.com.br/onibus/jussara-pr
        /paicandu-pr?departureDate
        =2019-07-12"},

    {"origem": "Pai andu", "destino": "
        Maringa", "link": "https://www.
        clickbus.com.br/onibus/paicandu-
        pr/maringa-pr?departureDate
        =2019-07-12"},

    {"origem": "Campo Mour o", "destino
        ": "Eng Beltr o", "link": "https
        ://www.clickbus.com.br/onibus/
        paicandu-pr/maringa-pr?
        departureDate=2019-07-12"},

    {"origem": "Maringa", "destino": "
        Londrina", "link": "https://www.
        clickbus.com.br/onibus/maringa-pr
        /engenheiro-beltrao-pr?
        departureDate=2019-07-12"}

    ]
```

Além do pacote *parser*, utilizamos uma biblioteca que permite montar um grafo com maior facilidade, a Networkx[4].

Antes de criar o grafo precisaremos de duas cidades para serem usadas como origem e destino, ambas contidas no arquivo JSON, com o link da rota das cidades obtido do JSON é feito um *scraping* para se obter o menor valor de passagem entre a origem e destino, este valor é colocado como peso da aresta que liga as duas cidades.

Depois do grafo criado, é usado mais 2 algoritmos, um para calcular o caminho de menor preço entre duas cidades, o outro é uma árvore geradora mínima. Todo essa lógica se encontra no Código 2.

Código 2 - Código responsável pelo parser e a criação do grafo.

```
# -*- coding: utf-8 -*-
import requests
from bs4 import BeautifulSoup
import json
import networkx as nx
import matplotlib.pyplot as plt

'''Carrega o Arquivo Json'''
arq = open('entradas.json')
cidades = json.load(arq)

'''Cria um Grafo Vazio'''
G = nx.Graph()
print("\n\n----------------------
    Come o do Parsing ---------------\n
    ")

'''Trecho que faz os Parsing e Scraping
    '''
for i in cidades['cidades']:

    print("Origem: "+i['origem']+"")

    '''Adiciona uma cidade Origem'''
    G.add_node(i['origem'])

    print("Destino: "+i['destino']+"")

    '''Adiciona uma cidade Destino'''
    G.add_node(i['destino'])

    print(i['link'])
    source = i['link']

    '''Faz um Request no Link do site
       paga Obter as informa es'''
    r = requests.get(source)
    soup = BeautifulSoup(r.content, '
       html.parser')
    precos  = soup.findAll(class_="
       price-value")
    resultados = list()
    for p in precos:
        l = (p.text.split('R$'))
        resultados.append(float((l[1].
           strip().replace(',','.'))))
    print("Preco: "+str(min(resultados)
       )+"\n\n")

    '''Adiciona uma Aresta entre os
       Vertices de Origem e Destino'''
    G.add_edge(i['origem'],i['destino
       '],weight=min(resultados))


print("\n\n---------------------- Fim
    do Parsing ---------------\n")

print("Caminho mais Curto entre duas
    Cidades: ")
path =list(nx.shortest_simple_paths(G,
```

```
    source="Peabiru", target="Maringa"))
print(path[0])

T = nx.minimum_spanning_tree(G)

print("\nGrafo Gerada: ")
print(list(G.edges(data=True)))

print("\nMST Gerada: ")
print(list(T.edges(data=True)))

subG= nx.Graph()


for n in path[0]:
    subG.add_node(n)

for i in range(len(path) -2):
    subG.add_edge(path[0][i], path[0][i
        +1])

plt.subplot(221)
nx.draw(G, with_labels=True,
    font_weight='bold')

plt.subplot(222)
nx.draw(T, with_labels=True,
    font_weight='bold')

plt.subplot(223)
nx.draw(subG, with_labels=True,
    font_weight='bold')

plt.show()
```

3 Resultados

A saída do Código - 2 retorna uma imagem gerada pelo pacote MatPlot[3], onde se encontra plotado o Grafo completo, o Grafo com a Árvore Geradora Miníma, e o Caminho Mínimo entre duas cidades. Como demonstrado na Figura 2.

E o Log dos vertices com seus pesos:

```

Caminho mais Curto entre duas Cidades:
['Peabiru', u'Campo Mour\xe3o', '
    Maringa']

Grafo Gerada:
[(u'Peabiru', u'Campo Mour\xe3o', {'
    weight': 5.94}), (u'Peabiru', u'Eng
    Beltr\xe3o', {'weight': 5.97}), (u'
    Londrina', u'Maringa', {'weight':
    23.46}), (u'Maringa', u'Ivailandia',
     {'weight': 11.76}), (u'Maringa', u'
    Pai\xe7andu', {'weight': 4.32}), (u'
    Maringa', u'Campo Mour\xe3o', {'
    weight': 30.94}), (u'Ivailandia', u'
    Eng Beltr\xe3o', {'weight': 4.57}),
    (u'Pai\xe7andu', u'Jussara', {'
    weight': 16.07}), (u'Eng Beltr\xe3o
    ', u'Campo Mour\xe3o', {'weight':
    4.32})]

MST Gerada:
[(u'Peabiru', u'Campo Mour\xe3o', {'
    weight': 5.94}), (u'Londrina', u'
    Maringa', {'weight': 23.46}), (u'
    Jussara', u'Pai\xe7andu', {'weight':
     16.07}), (u'Ivailandia', u'Maringa
    ', {'weight': 11.76}), (u'Ivailandia
    ', u'Eng Beltr\xe3o', {'weight':
    4.57}), (u'Pai\xe7andu', u'Maringa',
     {'weight': 4.32}), (u'Eng Beltr\
    xe3o', u'Campo Mour\xe3o', {'weight
    ': 4.32})]
```

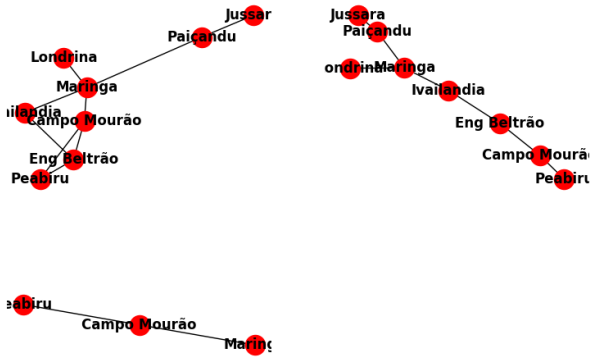

Figura 2: Imagem demonstrando o Grafo Completo, Árvore Geradora Mínima, e o Caminho Mínimo

Referências

[1] Beautiful soup. https://www.crummy.com/software/BeautifulSoup/bs4/doc/. Acessado: 10-07-2019.

[2] Click bus. https://www.clickbus.com.br/. Acessado: 07-07-2019.

[3] Mat Plot. https://matplotlib.org/. Acessado: 18-06-2019.

[4] Networkx. https://networkx.github.io/documentation/networkx-1.10/tutorial/tutorial.html. Acessado: 10-10-2019.

[5] O que é um grafo? IME. https://www.ime.usp.br/~pf/algoritmos_em_grafos/aulas/grafos.html. Acessado: 08-07-2019.