

## Tutorial Projeto 5

### Alunos:

Rafael Bastos Saito - 726580

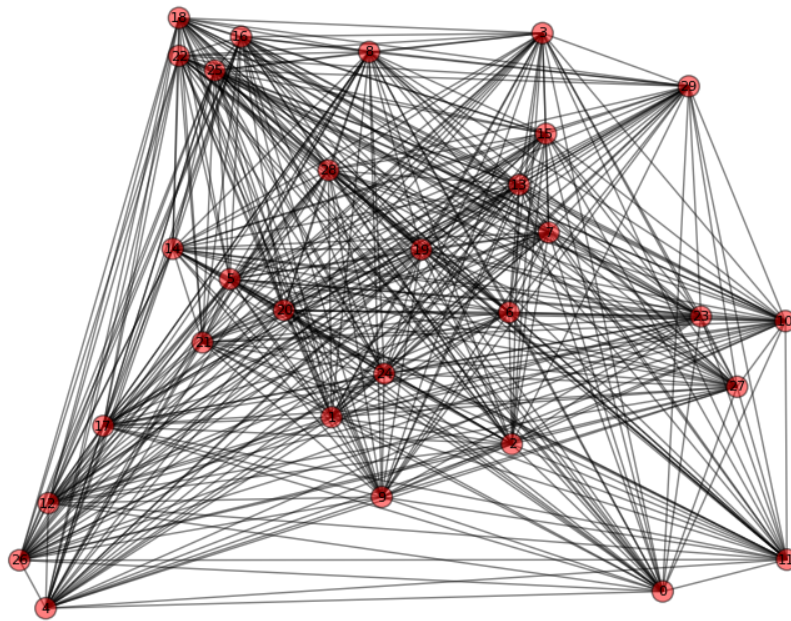
Renata Sarmet Smiderle Mendes - 726586

Rodrigo Pesse de Abreu - 726588

## Árvore Geradora Mínima

### Projeto sugerido pelo professor

A partir de um dataset específico (grafo ponderado armazenado em arquivo .gml, .graphml, .txt, .net, etc) implementar o algoritmo de Prim para extrair uma Minimum Spanning Tree (MST) de G.



Grafo apresentado - 30 cidades do mundo

A partir do pseudocódigo apresentado em sala, o algoritmo de Prim foi desenvolvido pelo grupo, resultando em:

```
4 import networkx as nx
5 import numpy as np
6 import matplotlib.pyplot as plt
7
8 toggle = True
9 G = nx.Graph()
10 H = {}
11 j = 0
12
13 def Prim(G = nx.Graph(), R = None):
14     i = 0
15     FilaPrioridade = {}
16     predecessor = {}
17
18     for v,data in G.nodes(data=True):
19         FilaPrioridade[v] = np.inf
20         predecessor[v] = 'null'
21
22     FilaPrioridade[R] = 0.0
23     MST = nx.create_empty_copy(G)
24
25     while FilaPrioridade:
26         u = min(FilaPrioridade,key = FilaPrioridade.get)
27         del FilaPrioridade[u]
28
29         for Vizinho in G[u]:
30             if Vizinho in FilaPrioridade:
31                 if G[u][Vizinho]['weight'] < FilaPrioridade[Vizinho]:
32                     predecessor[Vizinho] = u
33                     FilaPrioridade[Vizinho] = G[u][Vizinho]['weight']
34
35         if predecessor[u] is not 'null':
36             for v1,v2,data in G.edges(data=True):
37                 if (v1 == predecessor[u] and v2 == u):
38                     MST.add_edge(v1,v2, weight=data['weight'])
39                     H[i] = MST.copy()
40                     i = i + 1
41                 elif (v1 == u and v2 == predecessor[u]):
42                     MST.add_edge(v2,v1, weight=data['weight'])
43                     H[i] = MST.copy()
44                     i = i + 1
```

Algoritmo de Prim

O grupo desenvolveu uma forma, que, a partir de clicks, o grafo vá se formando, com seus pesos, vértices e arestas:

```
46 def onclick(event):
47     global toggle
48     global j
49
50     event.canvas.figure.clear()
51
52     if toggle:
53         labels = {}
54         nx.draw(G, pos, with_labels=True)
55         nx.draw_networkx_edge_labels(G, pos, labels)
56         toggle = not toggle
57
58     else:
59         labels = {}
60         for v1,v2,data in H[j].edges(data=True):
61             labels[(v1,v2)] = data['weight']
62             nx.draw(H[j], pos, with_labels=True)
63             nx.draw_networkx_edge_labels(H[j], pos, labels)
64             j = j + 1
65
66     event.canvas.draw()
67
68     A = np.loadtxt('ha30_dist.txt')
69     G = nx.from_numpy_matrix(A)
70     nx.draw_networkx(G)
71     plt.savefig('Inicial.pdf')
72     plt.show()
73
74     Prim(G, 0)
75
76     pos = nx.spring_layout(G)
77     fig = plt.figure()
78     fig.canvas.mpl_connect('button_press_event', onclick)
79
80     plt.show()
```

Algoritmo de plotagem a partir de clicks

## Resultado

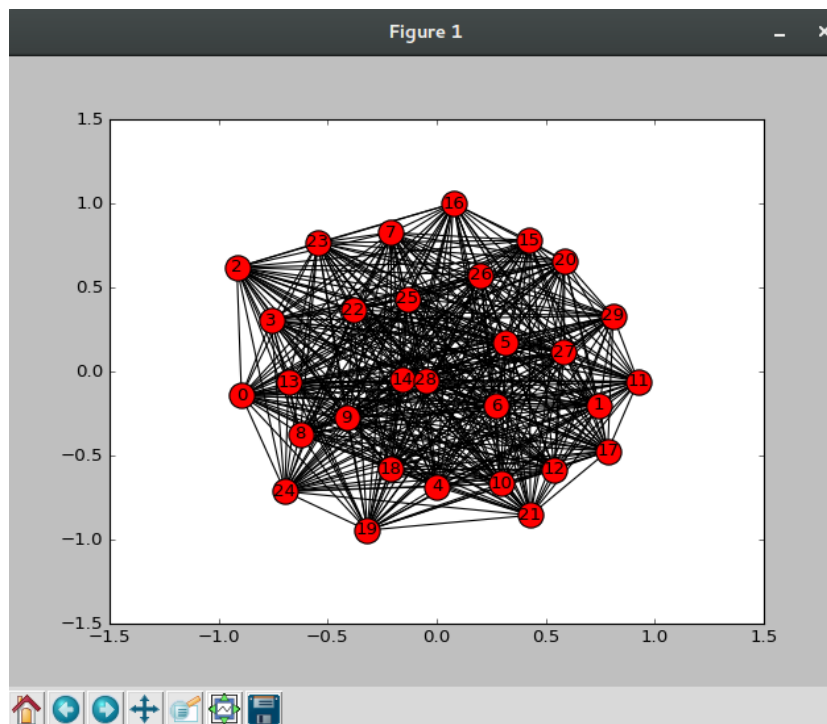
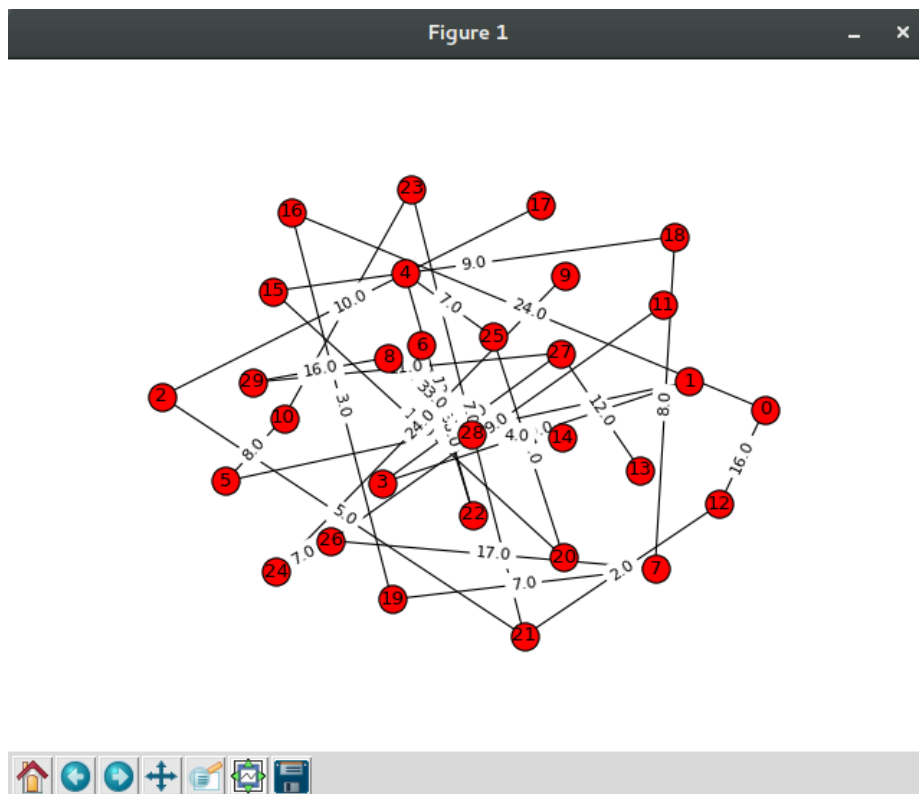


Imagem inicial



Resultado a partir da aplicação do algoritmo