Tutorial Projeto 4
Alunos:

Rafael Bastos Saito - 726580
Renata Sarmet Smiderle Mendes - 726586
Rodrigo Pesse de Abreu - 726588

Árvore de Caminhos Mínimos e Agrupamento de Dados

Projeto sugerido pelo professor

A partir de um dataset específico (grafo ponderado armazenado em arquivo .gml, .graphml, .txt, .net, etc) e implementar o algoritmo de Dijkstra para extrair uma árvore de caminhos mínimos de G.

Metodologia

Após a implementação do algoritmo, uma forma de crescer várias subárvores de caminhos mínimos é inicializar várias sources, ou seja, atribuir custo inicial zero a um número K de vértices. O restante do algoritmo permanece intacto. O que irá acontecer é um processo de disputa entre cada uma das raízes para verificar qual delas irá conquistar cada vértice de G. Ao final da execução um vértice estará "pendurado" apenas a uma única subárvore, fazendo com que tenhamos vários grupos de nós, similar ao que acontecia com as MST's. Porém aqui há supervisão no processo de formação dos grupos, uma vez que o usuário pode definir de onde as subárvores irão iniciar o crescimento (esses pontos devem ser escolhidos de forma a definir o centro dos agrupamentos).

Questionamento

Considerando o grafo em questão, mostre os resultados (plote graficamente) obtidos para:

a) 2 agrupamentos (K = 2)
b) 3 agrupamentos (K = 3)

Desenvolvimento

Após o grupo analisar o embasamento teórico apresentado em sala e analisado o pseudocódigo:

Definição das variáveis
$\lambda(v)$: menor custo até o momento para o caminho s-v
$\pi(v)$: predecessor de v na árvore de caminhos mínimos
Q: fila de prioridades dos vértices (maior prioridade = menor $\lambda(v)$)

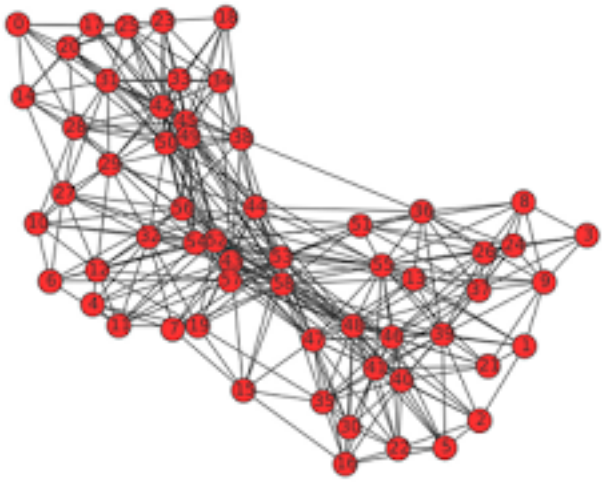
PSEUDOCODIGO

```
Dijkstra(G, w, s)
{
      λ(s)=0
      π(s)=0
    for each  v∈V
    {
          λ(v)=∞
          π(v)=nil
    }
    Q = V (fila de prioridades)
    while  Q≠∅
    {
        u = ExtractMin(Q)
        S=S∪[u]
        for each  v∈N(u)
            relax(u, v, w)
    }
}
```

Algoritmos e estruturas de dados
BFS – Fila
DFS – Pilha
Dijkstra – Fila de prioridades

Pseudocódigo apresentado em aula

A partir do grafo apresentado, que possui 59 cidades da Alemanha Ocidental e suas distâncias, pode-se desenvolver um algoritmo que contempla-se a metodologia de Dijkstra para resolver o problema:



Grafo apresentado - 59 cidades da Alemanhã Ocidenal

```python
import networkx as nx
import numpy as np
from matplotlib import pyplot as plt

def dijkstra(graph, sources):

    tmp = graph.copy()
    push = heappush
    pop = heappop
    nodes = tmp.nodes()

    for x in nodes:
        tmp.node[x][l] = np.inf # l = lambda do algoritmo passado em sala
        tmp.node[x]['pi'] = None

    for x in souces:
        tmp.node[x][l] = 0

    fp = []
    s = []

    for x in nodes:
        push(fp, (tmp,node[x][l],x))

    while fp:

        u = pop(fp)
        u = u[1]
        s.append(u)

        for x in tmp.neighbors(u):
            if v not in s and tmp.node[v][l] > (tmp.node[u][l] + g[u][v]['s.weight']):
                q.remove(tmp.node[x][l],x)
                tmp.node[x][l] = tmp.node[u][l] + g[u][v]['s.weight']
                tmp.node[x]['pi']=u

    r = nx.Graph()

    for u in tmp.nodes():
        r.add_node(u)
        if tmp.node[u]['pi'] is not None:
            r.add_edge(u,tmp.node[u]['pi'])
            r[u][tmp.node[u]['pi']]['s.weight'] = tmp[u][tmp.node[u]['pi']]['s.weight']

    return r
```

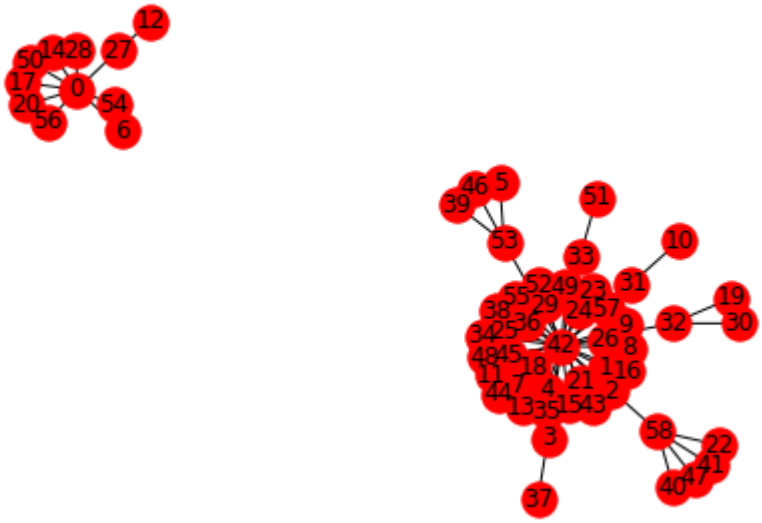
Algoritmo de Dijkstra

```python
def matriz_t(trail):

    data = np.loadtxt(trail)
    rows, cols = np.where(data > 0)
    edges = zip(rows,cols)
    graf = nx.Graph(edges)

    for x,y in zip(rows, cols):
        graf[x][y]['weight'] = data[x][y]
        graf[x][y]['s.weight'] = data[x][y]
    return graf

    alemanha = matriz_adj("data/wg59_dist.txt")
    dijkstra_b = dijkstra(alemanha, [0, 42])
    dijkstra_c = dijkstra(alemanha, [0, 26, 53])


    draw(alemanha,True,"alemanha")
    draw(dijkstra_b,True,"dijkstra_b")
    draw(dijkstra_c,True,"dijkstra_c")
```

Selecionando os agrupamentos desejados (K=2 e K=3)

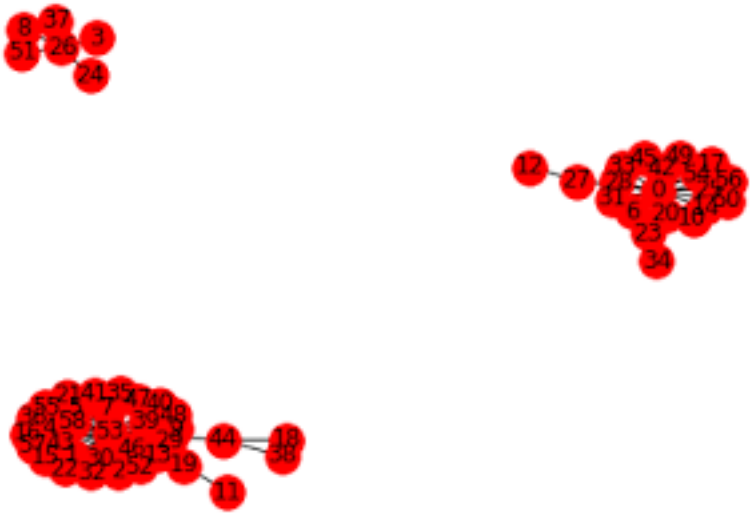
Resultados

2 agrupamentos K=2)



Pegando os vértices 0 e 42

3 agrupamentos K = 3)



Pegando os vértices 0, 26 e 53