

▼ DAA Lab - Practical - 2

Name : *Rishabh Jain*

Roll No : *A-54*

Aim:

Construction of Minimum Spanning Tree

Problem Statement:

A telecommunications organization has offices spanned across multiple locations around the globe. It has to use leased phone lines for connecting all these offices with each other. The organization, wants to use minimum cost for connecting all its offices. This requires that all the offices should be connected using a minimum number of leased lines so as to reduce the effective cost.

A. Consider the following for deciding connections in same state in India:

- i. Find the latitude and longitude of cities in same state. Consider 4 to 6 cities.
- ii. Calculate the cost of connecting each pair of offices by computing the distance between different pair of different cities (as considered in part A) and construct a fully connected graph.
- iii. Compute a minimum spanning tree using either Prim's or Kruskal's Method to find the cost of connecting offices in different cities.

B. Repeat the above for cities in different states.

```
import time;
time_ws = [];
time_os = [];
time_ws.clear();
time_os.clear();

def min_in_mat(mat):
    min = 1000000000000;
    min_i=-1;
    min_j=-1;
    for i in range(0,len(mat)):
        for j in range(0,len(mat)):
            if mat[i][j]<min:
                min = mat[i][j];
                min_i = i;
                min_j = j;
    return min_i,min_j;
```

```

def prims(matrix):
    ttf = [];
    near = [];
    k,l = min_in_mat(matrix);
    mincost = matrix[k][l];
    t1 = [k,l,matrix[k][l]];
    ttf.append(t1);

    for i in range(0,len(matrix)):
        if i==k or i==l:
            near.append(99);
        elif(matrix[i][k] < matrix[i][l]):
            near.append(k);
        else:
            near.append(l);

    print("Near array of all steps:");
    print(near);
    min_idx = -1;
    for i in range(1,len(matrix)-1):
        min = 999999999;
        for j in range(0,len(near)):
            if(near[j]!=99):
                if(matrix[j][near[j]]<min):
                    min = matrix[j][near[j]];
                    min_idx = j;
        t1 = [min_idx,near[min_idx],matrix[min_idx][near[min_idx]]];
        mincost = mincost + matrix[min_idx][near[min_idx]];
        near[min_idx] = 99;
        ttf.append(t1);

        for k in range(0,len(matrix)):
            if(near[k]!=99 and matrix[k][near[k]]>matrix[k][min_idx]):
                near[k] = min_idx;
        print(near);
    print("\nTTF:");
    for i in range(0,len(ttf)):
        print(ttf[i]);
    print("\nTotal Minimum Distance: ",round(mincost,2));

```

▼ For 4 cities within same state.

```

from geopy.distance import geodesic
from geopy.geocoders import Nominatim

matrix4 = [];
city = ["Nagpur","Amravati","Pune","Latur"];

geolocator = Nominatim(user_agent="Rishabh");

for i in range(0,len(city)):

```

```

arr = [];
for j in range(0,len(city)):
    if(i == j):
        arr.append(9999);
    else:
        loc1 = geolocator.geocode(city[i]);
        loc2 = geolocator.geocode(city[j]);
        city1 = (loc1.latitude,loc1.longitude);
        city2 = (loc2.latitude,loc2.longitude);
        dis=geodesic(city1, city2).miles;
        arr.append(round(dis,2));
matrix4.append(arr);

print("Input matrix:");
for i in range(0,len(matrix4)):
    print(matrix4[i]);

```

```

Input matrix:
[9999, 83.6, 385.28, 252.24]
[83.6, 9999, 307.64, 195.44]
[385.28, 307.64, 9999, 172.89]
[252.24, 195.44, 172.89, 9999]

```

```

start=time.perf_counter()
prims(matrix4);
end= time.perf_counter()
timetaken=end-start
time_ws.append(timetaken);

```

```

Near array of all steps:
[99, 99, 1, 1]
[99, 99, 3, 99]
[99, 99, 99, 99]

```

```

TTF:
[0, 1, 83.6]
[3, 1, 195.44]
[2, 3, 172.89]

```

```
Total Minimum Distance: 451.93
```

▼ For 5 cities within same state.

```

from geopy.distance import geodesic
from geopy.geocoders import Nominatim

matrix5 = [];
city = ["Nagpur","Amravati","Pune","Latur","Mumbai"];

geolocator = Nominatim(user_agent="Rishabh");

for i in range(0,len(city)):

```

```

arr = [];
for j in range(0,len(city)):
    if(i == j):
        arr.append(9999);
    else:
        loc1 = geolocator.geocode(city[i]);
        loc2 = geolocator.geocode(city[j]);
        city1 = (loc1.latitude,loc1.longitude);
        city2 = (loc2.latitude,loc2.longitude);
        dis=geodesic(city1, city2).miles;
        arr.append(round(dis,2));
matrix5.append(arr);

print("Input matrix:");
for i in range(0,len(matrix5)):
    print(matrix5[i]);

WARNING:urllib3.connectionpool:Retrying (Retry(total=1, connect=None, read=None, redi
Input matrix:
[9999, 83.6, 385.28, 252.24, 433.14]
[83.6, 9999, 307.64, 195.44, 351.58]
[385.28, 307.64, 9999, 172.89, 74.09]
[252.24, 195.44, 172.89, 9999, 242.74]
[433.14, 351.58, 74.09, 242.74, 9999]

start=time.perf_counter()
prims(matrix5);
end= time.perf_counter()
timetaken=end-start
time_ws.append(timetaken);

Near array of all steps:
[2, 2, 99, 2, 99]
[3, 3, 99, 99, 99]
[1, 99, 99, 99, 99]
[99, 99, 99, 99, 99]

TTF:
[2, 4, 74.09]
[3, 2, 172.89]
[1, 3, 195.44]
[0, 1, 83.6]

Total Minimum Distance: 526.02

```

▼ For 6 cities within same state.

```

from geopy.distance import geodesic
from geopy.geocoders import Nominatim

matrix6 = [];
city = ["Nagpur", "Amravati", "Pune", "Latur", "Mumbai", "Chandrapur"];

```

```
geolocator = Nominatim(user_agent="Rishabh");
```

```
for i in range(0,len(city)):
    arr = [];
    for j in range(0,len(city)):
        if(i == j):
            arr.append(9999);
        else:
            loc1 = geolocator.geocode(city[i]);
            loc2 = geolocator.geocode(city[j]);
            city1 = (loc1.latitude,loc1.longitude);
            city2 = (loc2.latitude,loc2.longitude);
            dis=geodesic(city1, city2).miles;
            arr.append(round(dis,2));
    matrix6.append(arr);
```

```
print("Input matrix:");
for i in range(0,len(matrix6)):
    print(matrix6[i]);
```

```
WARNING:urllib3.connectionpool:Retrying (Retry(total=1, connect=None, read=None, redi
WARNING:urllib3.connectionpool:Retrying (Retry(total=1, connect=None, read=None, redi
Input matrix:
[9999, 83.6, 385.28, 252.24, 433.14, 883.11]
[83.6, 9999, 307.64, 195.44, 351.58, 962.96]
[385.28, 307.64, 9999, 172.89, 74.09, 1268.3]
[252.24, 195.44, 172.89, 9999, 242.74, 1121.18]
[433.14, 351.58, 74.09, 242.74, 9999, 1314.31]
[883.11, 962.96, 1268.3, 1121.18, 1314.31, 9999]
```

```
start=time.perf_counter()
prims(matrix6);
end= time.perf_counter()
timetaken=end-start
time_ws.append(timetaken);
```

```
Near array of all steps:
[2, 2, 99, 2, 99, 2]
[3, 3, 99, 99, 99, 3]
[1, 99, 99, 99, 99, 1]
[99, 99, 99, 99, 99, 0]
[99, 99, 99, 99, 99, 99]
```

```
TTF:
[2, 4, 74.09]
[3, 2, 172.89]
[1, 3, 195.44]
[0, 1, 83.6]
[5, 0, 883.11]
```

```
Total Minimum Distance: 1409.13
```

▼ For 4 cities in different state.

```

from geopy.distance import geodesic
from geopy.geocoders import Nominatim

matrix41 = [];
city = ["Mumbai","Delhi","Kolkata","Chennai"];

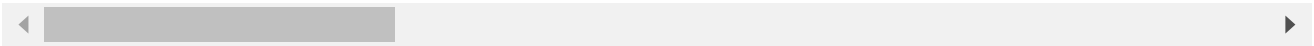
geolocator = Nominatim(user_agent="Rishabh");

for i in range(0,len(city)):
    arr = [];
    for j in range(0,len(city)):
        if(i == j):
            arr.append(9999);
        else:
            loc1 = geolocator.geocode(city[i]);
            loc2 = geolocator.geocode(city[j]);
            city1 = (loc1.latitude,loc1.longitude);
            city2 = (loc2.latitude,loc2.longitude);
            dis=geodesic(city1, city2).miles;
            arr.append(round(dis,2));
    matrix41.append(arr);

print("Input matrix:");
for i in range(0,len(matrix41)):
    print(matrix41[i]);

WARNING:urllib3.connectionpool:Retrying (Retry(total=1, connect=None, read=None, redi
Input matrix:
[9999, 718.88, 1034.63, 639.24]
[718.88, 9999, 812.84, 1087.83]
[1034.63, 812.84, 9999, 842.14]
[639.24, 1087.83, 842.14, 9999]

```



```

start=time.perf_counter()
prims(matrix41);
end= time.perf_counter()
timetaken=end-start
time_os.append(timetaken);

Near array of all steps:
[99, 0, 3, 99]
[99, 99, 1, 99]
[99, 99, 99, 99]

TTF:
[0, 3, 639.24]
[1, 0, 718.88]
[2, 1, 812.84]

```

Total Minimum Distance: 2170.96

▼ For 5 cities in different state.

```

from geopy.distance import geodesic
from geopy.geocoders import Nominatim

matrix51 = [];
city = ["Mumbai", "Delhi", "Kolkata", "Chennai", "Hyderabad"];

geolocator = Nominatim(user_agent="Rishabh");
for i in range(0, len(city)):
    arr = [];
    for j in range(0, len(city)):
        if(i == j):
            arr.append(9999);
        else:
            loc1 = geolocator.geocode(city[i]);
            loc2 = geolocator.geocode(city[j]);
            city1 = (loc1.latitude, loc1.longitude);
            city2 = (loc2.latitude, loc2.longitude);
            dis = geodesic(city1, city2).miles;
            arr.append(round(dis, 2));
    matrix51.append(arr);

print("Input matrix:");
for i in range(0, len(matrix51)):
    print(matrix51[i]);

WARNING:urllib3.connectionpool:Retrying (Retry(total=1, connect=None, read=None, redi
WARNING:urllib3.connectionpool:Retrying (Retry(total=1, connect=None, read=None, redi
Input matrix:
[9999, 718.88, 1034.63, 639.24, 387.38]
[718.88, 9999, 812.84, 1087.83, 779.73]
[1034.63, 812.84, 9999, 842.14, 736.04]
[639.24, 1087.83, 842.14, 9999, 317.55]
[387.38, 779.73, 736.04, 317.55, 9999]

start=time.perf_counter()
prims(matrix51);
end= time.perf_counter()
timetaken=end-start
time_os.append(timetaken);

Near array of all steps:
[4, 4, 4, 99, 99]
[99, 0, 4, 99, 99]
[99, 99, 4, 99, 99]
[99, 99, 99, 99, 99]

```

```
TTF:
[3, 4, 317.55]
[0, 4, 387.38]
[1, 0, 718.88]
[2, 4, 736.04]
```

Total Minimum Distance: 2159.85

▼ For 6 cities in different state.

```
from geopy.distance import geodesic
from geopy.geocoders import Nominatim

matrix61 = [];
city = ["Mumbai", "Delhi", "Kolkata", "Chennai", "Hyderabad", "Patna"];

geolocator = Nominatim(user_agent="Rishabh");

for i in range(0, len(city)):
    arr = [];
    for j in range(0, len(city)):
        if(i == j):
            arr.append(9999);
        else:
            loc1 = geolocator.geocode(city[i]);
            loc2 = geolocator.geocode(city[j]);
            city1 = (loc1.latitude, loc1.longitude);
            city2 = (loc2.latitude, loc2.longitude);
            dis=geodesic(city1, city2).miles;
            arr.append(round(dis, 2));
    matrix61.append(arr);

print("Input matrix:");
for i in range(0, len(matrix61)):
    print(matrix61[i]);

WARNING:urllib3.connectionpool:Retrying (Retry(total=1, connect=None, read=None, redi
WARNING:urllib3.connectionpool:Retrying (Retry(total=1, connect=None, read=None, redi
Input matrix:
[9999, 718.88, 1034.63, 639.24, 387.38, 909.37]
[718.88, 9999, 812.84, 1087.83, 779.73, 531.99]
[1034.63, 812.84, 9999, 842.14, 736.04, 292.53]
[639.24, 1087.83, 842.14, 9999, 317.55, 917.7]
[387.38, 779.73, 736.04, 317.55, 9999, 710.58]
[909.37, 531.99, 292.53, 917.7, 710.58, 9999]

start=time.perf_counter()
prims(matrix61);
end= time.perf_counter()
timetaken=end-start
time_os.append(timetaken);
```


Near array of all steps:

```
[5, 5, 99, 2, 5, 99]
[1, 99, 99, 2, 5, 99]
[4, 99, 99, 4, 99, 99]
[4, 99, 99, 99, 99, 99]
[99, 99, 99, 99, 99, 99]
```

TTF:

```
[2, 5, 292.53]
[1, 5, 531.99]
[4, 5, 710.58]
[3, 4, 317.55]
[0, 4, 387.38]
```

Total Minimum Distance: 2240.03

Analysis on time taken by different number cities within state and in different states.

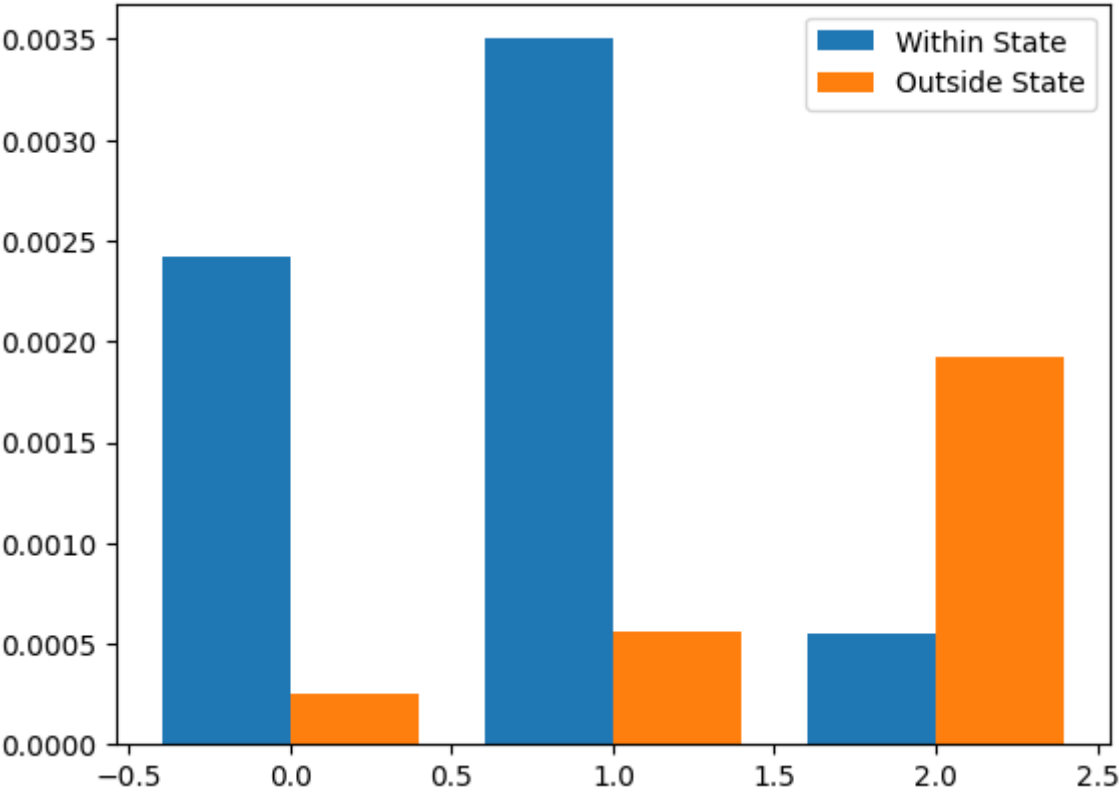
```
print(time_ws);
print(time_os);
```

```
250790000005509, 0.0014765070000066771, 0.0003716249999570209, 0.0004024649999792018]
5693999975083, 0.00901815899999292, 0.000903249999964828]
```



```
import matplotlib.pyplot as plt
import numpy as np
```

```
x = np.array(["4", "5", "6"])
tw = np.array(time_ws);
to = np.array(time_os);
X_axis = np.arange(len(x))
plt.bar(X_axis - 0.2, tw, 0.4, label='Within State')
plt.bar(X_axis + 0.2, to, 0.4, label='Outside State')
plt.legend()
plt.show()
```



!

0s

completed at 11:01 PM

●

×