04-03: Graph Algorithms

Networkx module

In [2]:

```
# Importing Networkx module
import networkx as nx
```

Representing Connections Between Cities

In [3]:

```
# First create a list of weighted edges in Python
edgelist = [['Mannheim', 'Frankfurt', 85], ['Mannheim', 'Karlsruhe', 80], ['Erfu
rt', 'Wurzburg', 186],
            ['Munchen', 'Numberg', 167], ['Munchen', 'Augsburg', 84], ['Munchen'
, 'Kassel', 502],
            ['Numberg', 'Stuttgart', 183], ['Numberg', 'Wurzburg', 103], ['Numbe
rg', 'Munchen', 167],
            ['Stuttgart', 'Numberg', 183], ['Augsburg', 'Munchen', 84], ['Augsbu
rg', 'Karlsruhe', 250],
            ['Kassel', 'Munchen', 502], ['Kassel', 'Frankfurt', 173], ['Frankfur
t', 'Mannheim', 85],
            ['Frankfurt', 'Wurzburg', 217], ['Frankfurt', 'Kassel', 173], ['Wurz
burg', 'Numberg', 103],
            ['Wurzburg', 'Erfurt', 186], ['Wurzburg', 'Frankfurt', 217], ['Karls
ruhe', 'Mannheim', 80],
            ['Karlsruhe', 'Augsburg', 250],["Mumbai", "Delhi",400],["Delhi", "Ko
lkata",500],
            ["Kolkata", "Bangalore",600], ["TX", "NY",1200],["ALB", "NY",800]]
# Creating a graph
g = nx.Graph()
for edge in edgelist:
    g.add_edge(edge[0], edge[1], weight = edge[2])
```

Finding Distinct Continents & Their Cities From This Graph

In [4]:

```
for index, component in enumerate(nx.connected_components(g)):
    print("cc"+str(index)+":", component)
```

```
cc0: {'Stuttgart', 'Frankfurt', 'Augsburg', 'Numberg', 'Erfurt', 'Ka
ssel', 'Munchen', 'Mannheim', 'Karlsruhe', 'Wurzburg'}
cc1: {'Delhi', 'Kolkata', 'Bangalore', 'Mumbai'}
cc2: {'NY', 'ALB', 'TX'}
```

Dijkstra's Shortest Path

Shortest Path between Stuttgart & FrankFurt

In [6]:

```
print (nx.shortest_path(g, 'Stuttgart','Frankfurt', weight = 'weight'))
print(nx.shortest_path_length(g, 'Stuttgart', 'Frankfurt',weight = 'weight'))
```

```
['Stuttgart', 'Numberg', 'Wurzburg', 'Frankfurt']
503
```

All Pairs of Shortest Paths:

You want to find out how to go from Frankfurt (The starting node) to Munchen by covering the shortest distance.

In [8]:

```
for x in nx.all_pairs_dijkstra_path(g,weight='weight'):
    #print(x)
    #uncomment this line for your output to explode
```

```
  File "<ipython-input-8-76e6ee5eefad>", line 3
    #uncomment this line for your output to explode.
                                                    ^
SyntaxError: unexpected EOF while parsing
```

Jarnik Prim's Minimum Spanning Tree

We work for a water pipe laying company or an internet fiber company. We need to connect all the cities in the graph we have using the minimum amount of wire/pipe.

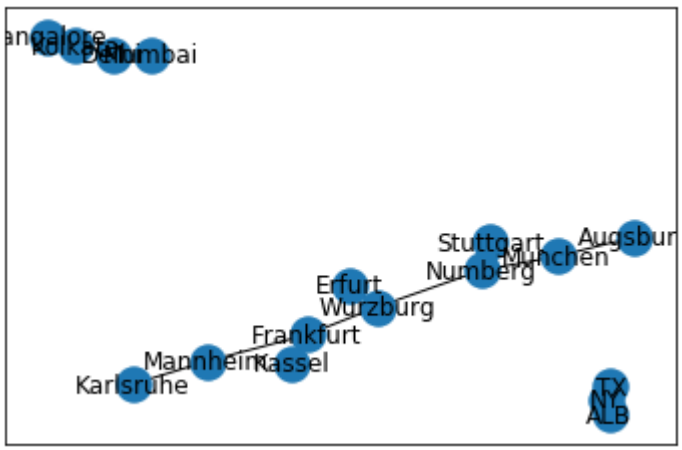
Applications

- Minimum spanning trees have direct applications in the design of networks, including computer networks, telecommunications networks, transportation networks, water supply networks, and electrical grids (which they were first invented for)
- MST is used for approximating the traveling salesman problem
- Clustering — First construct MST and then determine a threshold value for breaking some edges in the MST using Intercluster distances and Intracluster distances.
- Image Segmentation — It was used for Image segmentation where we first construct an MST on a graph where pixels are nodes and distances between pixels are based on some similarity measure(color, intensity, etc.)

In [12]:

```
#nx.minimum_spanning_tree(g) returns an instance of type graph. Call the draw_ne
tworkx function to see the network.
print(nx.draw_networkx(nx.minimum_spanning_tree(g)))
```

```
None
```



PageRank

This is the page sorting algorithm that powered google for a long time. It assigns scores to pages based on the number and quality of incoming and outgoing links.

Applications

- Pagerank can be used anywhere where we want to estimate node importance in any network.
- It has been used for finding the most influential papers using citations.
- Has been used by Google to rank pages
- It can be used to rank tweets- User and Tweets as nodes. Create Link between user if user A follows user B and Link between user and Tweets if user tweets/retweets a tweet.
- Recommendation engines

Centrality Measures

There are a lot of centrality measures which you can use as features to your machine learning models. I will talk about two of them. You can look at other measures here.

- Betweenness Centrality: It is not only the users who have the most friends that are important, the users who connect one geography to another are also important as that lets users see content from diverse geographies. Betweenness centrality quantifies how many times a particular node comes in the shortest chosen path between two other nodes.
- Degree Centrality: It is simply the number of connections for a node. Applications Centrality measures can be used as a feature in any machine learning model.

In [ ]: