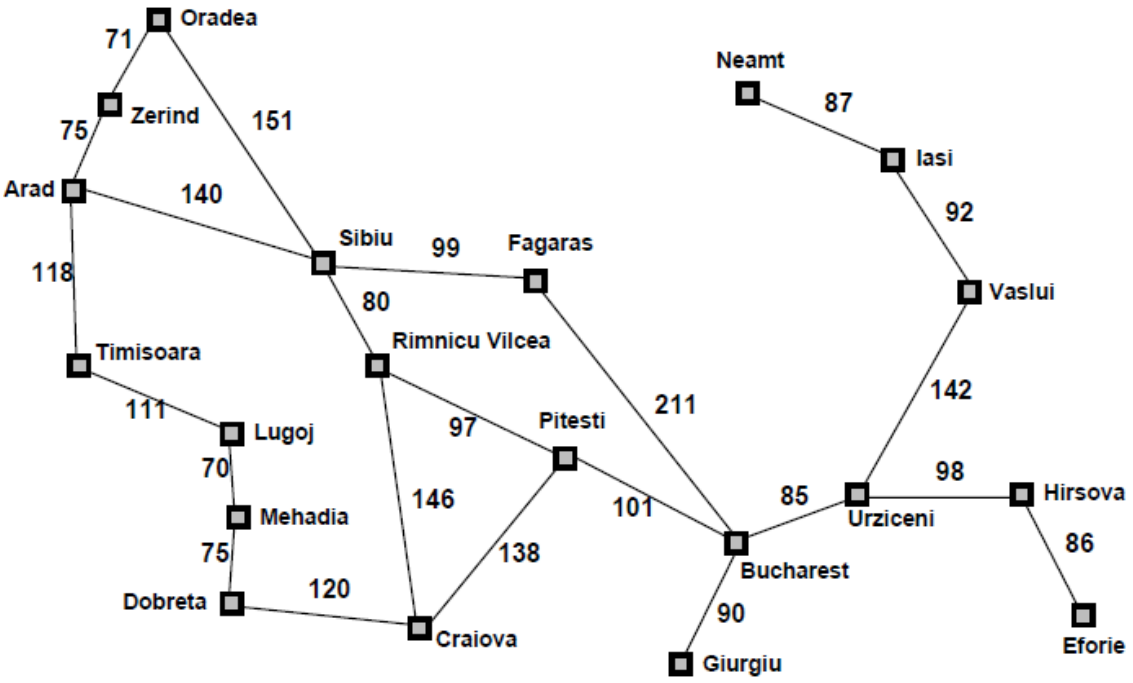
TahaAbbasAli_P20-0119_6A_AILab_10

May 19, 2023

[18]:
```
import networkx as nx
import matplotlib.pyplot as plt
from queue import PriorityQueue
```

Suppose that you plan to spend your summer vacations in Romania. Following is the map of Romania.



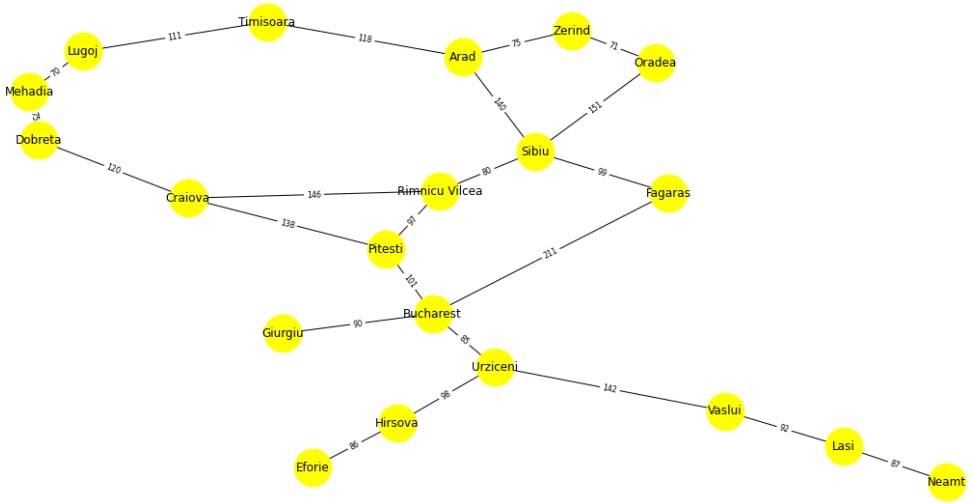
[19]:
```
G = nx.Graph()

G.add_nodes_from(["Arad", "Bucharest", "Oradea", "Zerind", "Timisoara",␣
↪"Lugoj", "Mehadia", "Dobreta", "Craiova",
                  "Rimnicu Vilcea", "Sibiu", "Fagaras", "Pitesti", "Giurgiu",␣
↪"Urziceni", "Vaslui", "Lasi","Neamt",
                  "Hirsova", "Eforie"]) #add remaining nodes to the list
```

[20]:
```
edges = [("Arad", "Zerind", 75),("Arad", "Sibiu", 140), ("Arad", "Timisoara",␣
↪118),("Bucharest", "Urziceni",85),
```

```
        ("Bucharest", "Giurgiu",90),("Bucharest", "Pitesti",101),("Bucharest",␣
 ↪"Fagaras",211),
        ("Craiova", "Dobreta",120),("Craiova",␣
 ↪"Pitesti",138),("Craiova","Rimnicu Vilcea", 146),
        ("Dobreta", "Mehadia", 75),("Eforie", "Hirsova", 86),("Fagaras",␣
 ↪"Sibiu", 99),("Hirsova", "Urziceni", 98),
        ("Lasi", "Neamt", 87),("Lasi", "Vaslui", 92),("Lugoj", "Mehadia",␣
 ↪70),("Lugoj", "Timisoara",111),
        ("Oradea", "Zerind", 71),("Oradea", "Sibiu", 151),("Pitesti", "Rimnicu␣
 ↪Vilcea", 97),
        ("Rimnicu Vilcea", "Sibiu",80),("Urziceni", "Vaslui",142)]

for edge in edges:
    G.add_edge(edge[0], edge[1], weight=edge[2])
```

[21]:
```
# Set node positions using Kamada-Kawai layout
pos = nx.kamada_kawai_layout(G)
```

[22]:
```
# Draw graph with labels and edge weights
plt.figure(figsize=(16, 8))
nx.draw(G, pos, with_labels=True, font_size=12, node_size= 1500, node_color␣
 ↪="yellow")

edge_labels = nx.get_edge_attributes(G, "weight")
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=8)

plt.show()
```



[23]:
```
# heuristic_values are given
heuristic_values = {"Arad" : 366, "Bucharest":0,"Oradea":380,"Zerind":
 ↪374,"Sibiu":253,"Timisoara":329,"Lugoj":244,"Mahadia":241,"Dobreta":
 ↪242,"Rimnicu Vilcea":193,"Craiova":160,"Pitesti":100,"Fagaras":176,"Giurgiu":
 ↪ 77,"Urziceni":80,"Hirsova":151,"Eforie":161,"Vaslui":199,"Iasi":226,"Neamt":
 ↪234}

heuristic_values
```

[23]:
```
{'Arad': 366,
 'Bucharest': 0,
 'Oradea': 380,
 'Zerind': 374,
 'Sibiu': 253,
 'Timisoara': 329,
 'Lugoj': 244,
 'Mahadia': 241,
 'Dobreta': 242,
 'Rimnicu Vilcea': 193,
 'Craiova': 160,
 'Pitesti': 100,
 'Fagaras': 176,
 'Giurgiu': 77,
 'Urziceni': 80,
 'Hirsova': 151,
 'Eforie': 161,
 'Vaslui': 199,
 'Iasi': 226,
 'Neamt': 234}
```

[24]:
```
def bestfs(start_node, goal_node):
    PQ = PriorityQueue()


    visited = []
    closed = []

    PQ.put((heuristic_values[starting],starting))

    while PQ.empty() == False:

        n = PQ.get()

        h = n[0]
        city = n[1]

        closed.append(city)
```

```
        visited.append(city)

        successors = [i for i in G.neighbors( city)]

        if goal in successors:

            visited.append(goal)
            cost = nx.path_weight(G, visited, "weight")
            print("Visited cities:", visited)
            print("Cost = ", cost)
            break

        successor_queue = PriorityQueue()

        for i in successors:
            successor_queue.put((heuristic_values[i], i))

        for i in successors:
            s = successor_queue.get()
            if s not in closed and s not in visited:
                PQ.put(s)
                break


start_node = "Arad"
goal_goal = "Bucharest"
bestfs(start_node, goal_goal)
```

```
Visited cities: ['Arad', 'Sibiu', 'Fagaras', 'Bucharest']
Cost =  450
```

[ ]: