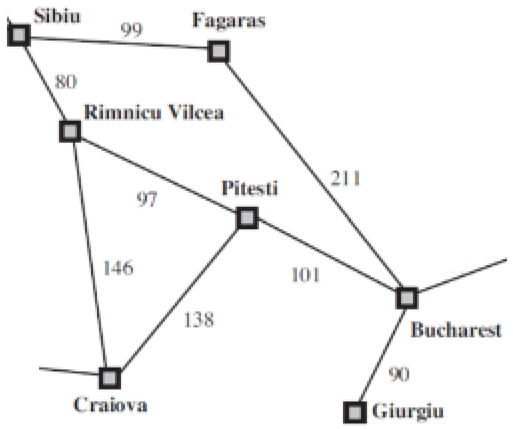
Assignment 1

Chapter 3 - Implement the breadth-first search (BFS) algorithm

With the help of the code blocks provided below, implement the BFS algorithm (in Python) to find the shortest path from Sibiu to Bucharest in the following map. Note that in a standard BFS algorithm/implementation, we ignore the connection weights.



__Queue basics in Python:__

```
# Initializing a queue
queue = []
# Adding elements to the queue
queue.append('a')
queue.append('b')
queue.append('c')
# Print
print(queue)
# Removing elements from the queue
print("\nElements dequeued from queue")
print(queue.pop(0))
print(queue.pop(0))
print(queue)
```

__Representing a graph using dictionary (values are lists of neighbors):__

```
graph = {}
graph['Sibiu'] = ['Fagaras', 'Rimnicu Vilcea']
graph['Fagaras'] = ['Sibiu', 'Bucharest']
graph['Rimnicu Vilcea'] = ['Sibiu', 'Pitesti', 'Craiova']
```

```
graph['Pitesti'] = ['Rimnicu Vilcea', 'Craiova', 'Bucharest']
graph['Craiova'] = ['Rimnicu Vilcea', 'Pitesti']
graph['Bucharest'] = ['Fagaras', 'Pitesti', 'Giurgiu']
graph['Giurgiu'] = ['Bucharest']
```

<u>Shortest path using BFS:</u>

```
def shortest_path_BFS(graph, start, goal):
  To Do:
  To Do:


# Call shortest_path_BFS
shortest_path_BFS(graph, 'Sibiu', 'Bucharest')
```

<u>Expected output:</u>

```
['Sibiu', 'Fagaras', 'Bucharest']
```