EAST WEST UNIVERSITY
Department: Computer Science & Engineering

Assignment – Lab 5

Semester : Fall 2021
Course Number : CSE366
Course Title : Artificial Intelligence
Course Instructor : Md Al-Imran

Student ID: 2019-1-60-093
Student Name: Md. Asad Chowdhury Dipu
Section: 01
Date of Submission: 1/12/2021

A*

In [1]:

```
def aStar(start, stop):
    open_set = set([start])
    closed_set = set([])
    g = {}
    parents = {}
    g[start] = 0
    parents[start] = start

    while len(open_set) > 0:
        n = None
        for v in open_set:
            if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):
                n = v

        if n == stop or graph_nodes[n] == None:
            pass

        else:

            for(m, weight) in get_neighbors(n):
                if m not in open_set and m not in closed_set:
                    open_set.add(m)
                    parents[m] = n
                    g[m] = g[n] + weight
            else:
                if g[m] > g[n] + weight:
                    g[m] = g[n] + weight
                    parents[m] = n

                    if m in closed_set:
                        closed_set.remove(m)
                        open_set.add(m)

        if n == None:
            print ('Path des not exist')
            return None
        if n == stop:
            path = []

            while parents[n] != n:
                path.append(n)
                n = parents[n]
            path.append(start)
            path.reverse()

            print("Path found: {}".format(path))
            return path

        open_set.remove(n)
        closed_set.add(n)
    print('path does not exist')
    return
```

In [2]:

```
def get_neighbors(v):
    if v in graph_nodes:
        return graph_nodes[v]
    else:
        return None
```

In [3]:

```
def heuristic(n):
    h_dist = {
        'Arad' : 366,
        'Zerind' : 374,
        'Timisoara' : 329,
        'Sibiu' : 253,
        'Oradea' : 380,
        'Lugoj' : 244,
        'RimnicuVilcea' : 193,
        'Mehadia' : 241,
        'Craiova' : 160,
        'Pitesti' : 98,
        'Fagaras' : 178,
        'Dobreta' : 242,
        'Bucharest' : 0,
        'Giurgiu' : 77,
    }
    return h_dist[n]
```

In [4]:

```
graph_nodes = {

    'Arad' : [('Zerind', 75), ('Timisoara', 118), ('Sibiu', 140)],
    'Zerind' : [('Oradea', 71), ('Arad', 75)],
    'Timisoara' : [('Arad', 118), ('Lugoj', 111)],
    'Sibiu' : [('Arad', 140), ('Oradea', 151), ('Fagaras', 99), ('RimnicuVilcea', 80)],
    'Oradea' : [('Zerind', 71), ('Sibiu', 151)],
    'Lugoj' : [('Timisoara', 111), ('Mehadia', 70)],
    'RimnicuVilcea' : [('Sibiu', 80), ('Pitesti', 97), ('Craiova', 146)],
    'Mehadia' : [('Lugoj', 70), ('Dobreta', 75)],
    'Craiova' : [('Dobreta', 120), ('RimnicuVilcea', 146), ('Pitesti', 138)],
    'Pitesti' : [('RimnicuVilcea', 97), ('Craiova', 138), ('Bucharest', 101)],
    'Fagaras' : [('Sibiu', 99), ('Bucharest', 211)],
    'Dobreta' : [('Mehadia', 75), ('Craiova', 120)],
    'Bucharest' : [('Fagaras', 211), ('Pitesti', 101), ('Giurgiu', 90)],
    'Giurgiu' : [('Bucharest', 90)],
}
```

In [5]:

```
aStar('Arad','Bucharest')
```

```
Path found: ['Arad', 'Sibiu', 'RimnicuVilcea', 'Pitesti', 'Bucharest']
```

Out[5]:

```
['Arad', 'Sibiu', 'RimnicuVilcea', 'Pitesti', 'Bucharest']
```

In [6]:

```
def heuristic(n):
    h_dist = {
        'A' : 11,
        'B' : 6,
        'C' : 99,
        'D' : 1,
        'E' : 7,
        'G' : 0,
    }
    return h_dist[n]
```

In [7]:

```
graph_nodes = {
    'A' : [('B', 2), ('E', 3)],
    'B' : [('C' , 1), ('G', 9)],
    'C' : None,
    'E' : [('D', 6)],
    'D' : [('G',1)]
}

```

In [8]:

```
aStar('A', 'G')
```

```
Path found: ['A', 'E', 'D', 'G']
```

Out[8]:

```
['A', 'E', 'D', 'G']
```