# 21162101012_CBA_Yash_Lakhtariya

## AI Practical 4

**Aim - Flyhigh Travel, Inc. provides its customers rental charter helicopters between eight cities of India. To help them plan a trip you are given a task to create a software module that finds optimal routes between any two cities. Connectivity between cites and cost to reach there is given in the attached image.**

- Since you only have limited information about the routes you will have to use your search agent and various search algorithms to find the way - Implement the A-star algorithm.
- Proper visualization must be there for the path. To implement that you can use various libraries for map visualization.

[1]
```
!pip install matplotlib
```

20s
```
import matplotlib.pyplot as plt

class Node:
    def __init__(self, p, pos, c, h, goal):
        self.p = p
        self.pos = pos
        self.c = c
        self.f = {t: 0 for t in h}
        self.update_f(h, goal)

    def update_f(self, h, goal):
        for t in h:
            self.f[t] = self.c + heuristic(t, self.pos, goal)

def heuristic(t, pos, goal):
    return h_t[t][pos.lower()]

def a_star(g, start, goal, h_t, chosen_h):
    open_set = []
    closed_set = set()
    start_node = Node(None, start, 0, h_t, goal)
    open_set.append(start_node)
    while open_set:
        current_node = min(open_set, key=lambda node: node.f[chosen_h])
        open_set.remove(current_node)
        closed_set.add(current_node.pos)
        if current_node.pos.lower() == goal.lower():
            path = []
            while current_node:
                path.append(current_node.pos)
                current_node = current_node.p
            path_cost = calculate_path_cost(g, path)
            return path[::-1], path_cost
        for n, mc in g[current_node.pos].items():
            if n.lower() in closed_set:
                continue
            tentative_cost = current_node.c + mc
            new_node = Node(current_node, n, tentative_cost, h_t, goal)
            if new_node not in [node for node in open_set if
                                node.pos.lower() == new_node.pos.lower() and new_node.c >= tentative_cost]:
                new_node.p = current_node
                new_node.c = tentative_cost
                new_node.update_f(h_t, goal)
                open_set.append(new_node)
    return None, None

def calculate_path_cost(g, path):
    cost = 0
    for i in range(len(path) - 1):
        cost += g[path[i]][path[i + 1]]
    return cost

def visualize_path(g, path):
    plt.figure()
    for i in range(len(path) - 1):
        plt.plot([path[i], path[i + 1]], [0, 0], '->', color='#dd7878', linewidth=3)
    plt.xlabel("Locations")
    plt.title("Path Visualization")
    plt.show()

g = {
    "Delhi": {"Jaipur": 9.1},
    "Jaipur": {"Delhi": 9.1, "Ahmedabad": 8.1},
    "Ahmedabad": {"Mumbai": 10.9, "Jaipur": 8.1, "Nagpur": 7.9},
    "Nagpur": {"Kolkata": 9.7, "Ahmedabad": 7.9, "Hyderabad": 7.9},
    "Kolkata": {"Nagpur": 9.7, "Hyderabad": 8.8},
    "Hyderabad": {"Kolkata": 8.8, "Mumbai": 8.1, "Nagpur": 7.9},
    "Mumbai": {"Ahmedabad": 10.9, "Goa": 10.6, "Hyderabad": 8.1},
    "Goa": {"Mumbai": 10.6}
}

h_t = {
    "delhi": {"delhi": 0, "jaipur": 235, "ahmedabad": 775, "nagpur": 852, "kolkata": 1305, "hyderabad": 1266,
              "mumbai": 1153, "goa": 1515},
    "jaipur": {"delhi": 235, "jaipur": 0, "ahmedabad": 534, "nagpur": 722, "kolkata": 1359, "hyderabad": 1088,
               "mumbai": 914, "goa": 1297},
    "ahmedabad": {"delhi": 775, "jaipur": 534, "ahmedabad": 0, "nagpur": 706, "kolkata": 1617, "hyderabad": 876,
                  "mumbai": 441, "goa": 871},
    "nagpur": {"delhi": 852, "jaipur": 722, "ahmedabad": 706, "nagpur": 0, "kolkata": 964, "hyderabad": 422,
               "mumbai": 844, "goa": 845},
    "kolkata": {"delhi": 1305, "jaipur": 1359, "ahmedabad": 1617, "nagpur": 964, "kolkata": 0, "hyderabad": 1180,
                "mumbai": 1652, "goa": 2187},
    "hyderabad": {"delhi": 1266, "jaipur": 1088, "ahmedabad": 876, "nagpur": 422, "kolkata": 1180, "hyderabad": 0,
                  "mumbai": 617, "goa": 530},
    "mumbai": {"delhi": 1153, "jaipur": 914, "ahmedabad": 441, "nagpur": 844, "kolkata": 1652, "hyderabad": 617,
               "mumbai": 0, "goa": 435},
    "goa": {"delhi": 1515, "jaipur": 1297, "ahmedabad": 871, "nagpur": 845, "kolkata": 2187, "hyderabad": 530,
            "mumbai": 435, "goa": 0},
}

s = input("\n\tEnter source location : ")
d = input("\n\tEnter destination location : ")
h = d.lower()
p, c = a_star(g, s, d, h_t, h)

if p:
    print("\n\tPath to reach : ", end='')
    for i, x in enumerate(p):
      print((x + '') if i == len(p) - 1 else (x + ' --> '), end='')

    print("\n\n\tMinimum cost : ", c)
    visualize_path(g, p)
else:
    print("No path found")
```

```
        Enter source location : Delhi

        Enter destination location : Goa

        Path to reach : Delhi --> Jaipur --> Ahmedabad --> Mumbai --> Goa

        Minimum cost :  38.7
```