UNIVERSITY OF ASIA PACIFIC

Department of Computer Science & Engineering

Artificial Inteligence Lab

Course Code: CSE 404

Assignment -2

Student Name	:	Md. Sohanuzzaman Soad
Student ID	:	18101064
Section	:	B1
Date of Submission	:	09.06.2021

INDEX

Introduction:

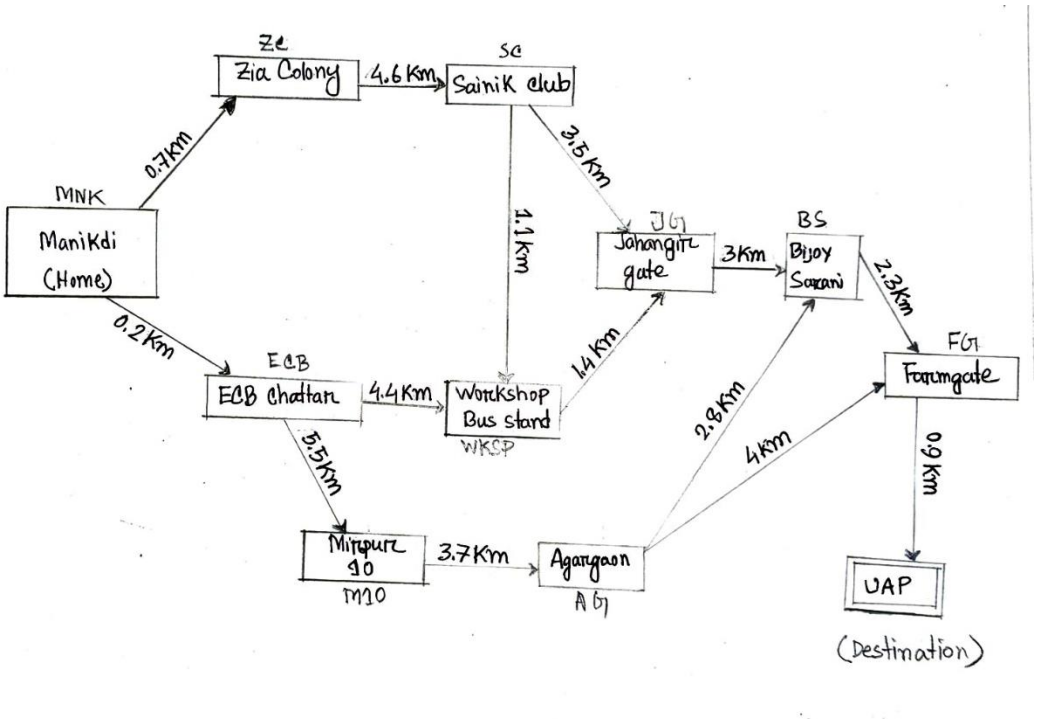
Nowadays most of the time in our daily life we use Google Map as a Navigator System. Google map suggest us shortest path when we choose our current location and destination. This shortest path is generated based on distance of source and destination.

Objective:

In this project our objective is implement a small version of google maps with specific source and destination location with some nodes between source and destination.

Designed Map:

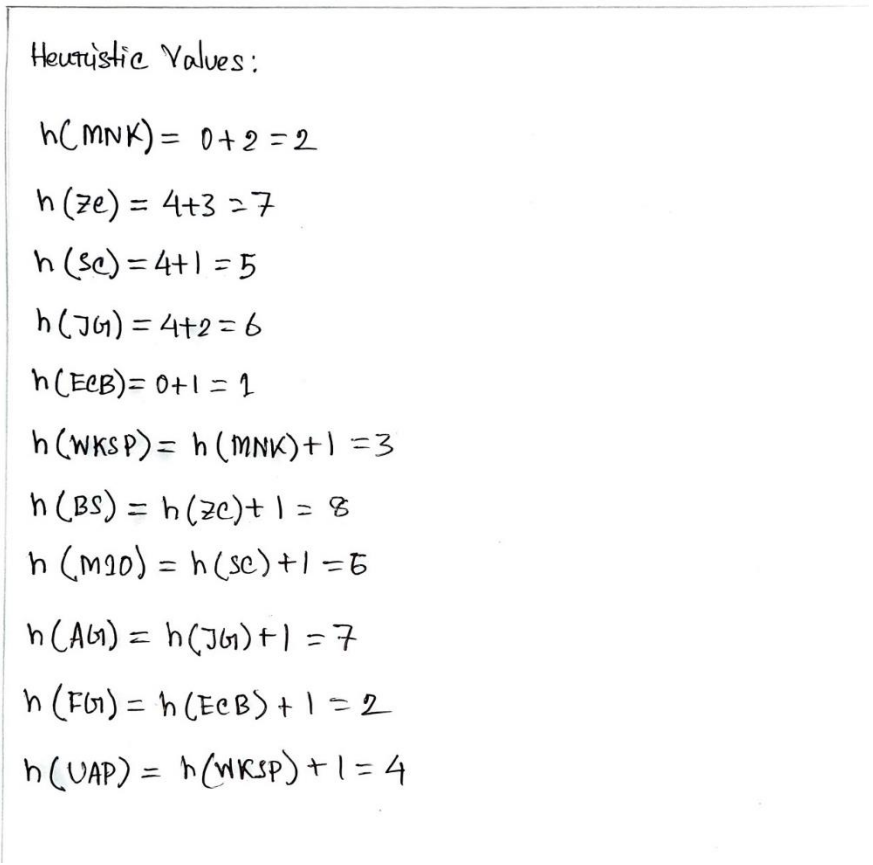
Map For Home to UAP:



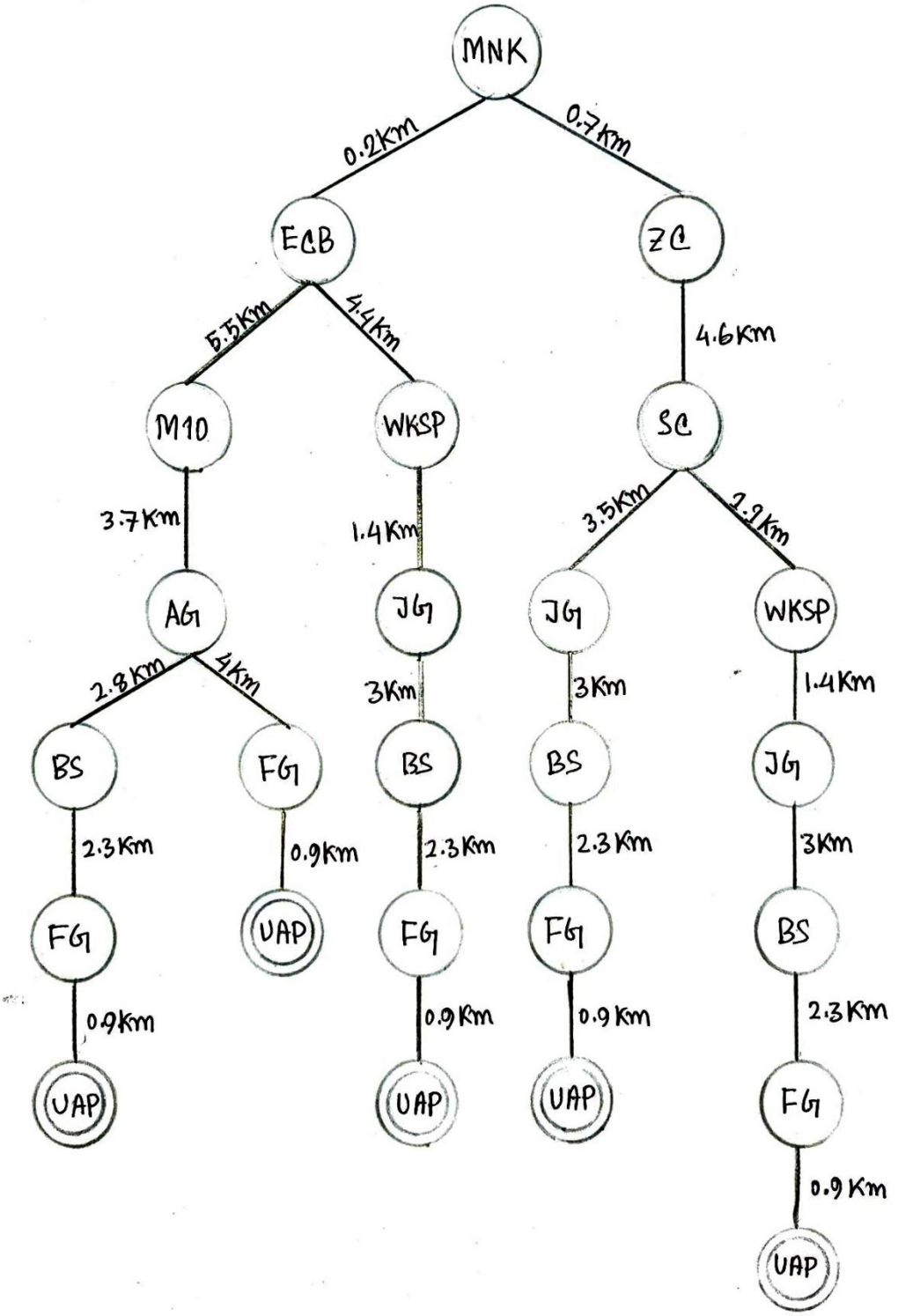
Short Name and Corresponding Full Name of Nodes:

Short Name	**Full Name**
MNK	Manikdi
ZC	Zia Colony
SC	Sainik Club
JG	Jahangir Gate
ECB	ECB Chattar
WKSP	Workshop Bus Stand
M10	Mirpur 10
AG	Agargaon
BS	Bijoy Sharani
FG	Farmgate
UAP	University of Asia Pacific

Heuristic Values:



Search Tree of The Map:



Implementation:

```
class Graph:
    def __init__(self):
        self.edges = {}
        self.heuristics = {}

    def add_edge(self, node_a, node_b, cost):
        try:
            neighbors = self.edges[node_a]
        except KeyError:
            neighbors = {}
        neighbors[node_b] = cost
        self.edges[node_a] = neighbors

    def set_heuristics(self, heuristics={}):
        self.heuristics = heuristics

    def get_neighbors(self, node):
        try:
            return self.edges[node]
        except KeyError:
            return []

    def cost(self, node_a, node_b):
        try:
            return self.edges[node_a][node_b]
        except:
            return inf
```

```
def a_star_search(self, start, goal):
        found, fringe, visited, came_from, cost_so_far = False, [
            (self.heuristics[start], start)], set([start]), {start: None}, {start: 0}
        while not found and len(fringe):
            _, current = heappop(fringe)

            if current == goal:
                found = True
                break
            for node in self.get_neighbors(current):
                new_cost = cost_so_far[current] + self.cost(current, node)
                if node not in visited or cost_so_far[node] > new_cost:
                    visited.add(node)
                    came_from[node] = current
                    cost_so_far[node] = new_cost
                    heappush(fringe, (new_cost + self.heuristics[node], node))

        if found:
            return came_from, cost_so_far[goal]
        else:
            print('No path from {} to {}'.format(start, goal))
            return None, inf
@staticmethod
    def print_path(came_from, goal):
        parent = came_from[goal]
        if parent:
            Graph.print_path(came_from, parent)
        else:
            print(full_name[goal], end='')
            return
        print(' =>', full_name[goal], end='')
```

Result Analysis:

From the Map Shortest Path is :

Manikdi => ECB Chattar => Workshop Bus Stand => Jahangir Gate => Bijoy Sharani => Farmgate => University of Asia Pacific

Output from Implementation:

```
Path: Manikdi => ECB Chattar => Workshop Bus Stand => Jahangir Gate => Bijoy Sharani => Farmgate => University of Asia Pacific
Cost: 12.2 km
```

Conclusion:

Shortest Path for Start location to Destination location in this Map is Completely same with our Output of Implementation. So, that we can say we use this program for getting optimal path in any given map.