Department of Computer Science & Engineering

Title: Artificial Intelligence System & Expert System Lab

Course Code: CSE 404

Lab Report: 02

Assignment: Implementation of a small Address Map (from my own home to UAP) using A* Search Algorithm.

Submission Date: 06/03/2025

Submitted To:	**Submitted By:**
Noor Mairukh Khan Arnob	Name: Humaira Ahmed Proma
Lecturer,	Reg no: 21201178
Department of CSE, UAP	Sec: B2

Problem Title:

Determining and Finding the Optimal Path and Most Efficient Route from Moghbazar Ideal Point to UAP Using the A* Search Algorithm.

Problem Description:

The objective of this task is to find the optimal path from Moghbazar Ideal Point (starting point) to the University of Asia Pacific (UAP) using the A* search algorithm. The A* algorithm evaluates possible paths by combining the actual cost of traveling to a node with an estimate of the cost to reach the destination, ensuring that the chosen route is both efficient and practical.

The algorithm calculates the best path by evaluating a function defined as:

$f(n)=g(n)+h(n)$

f(n):- f(n) represents the total cheapest cost for reaching the destination.

g(n):- g(n) is the cost to travel from the starting point to node n.

h(n):- h(n) is the estimated cost to travel from node n to goal node.

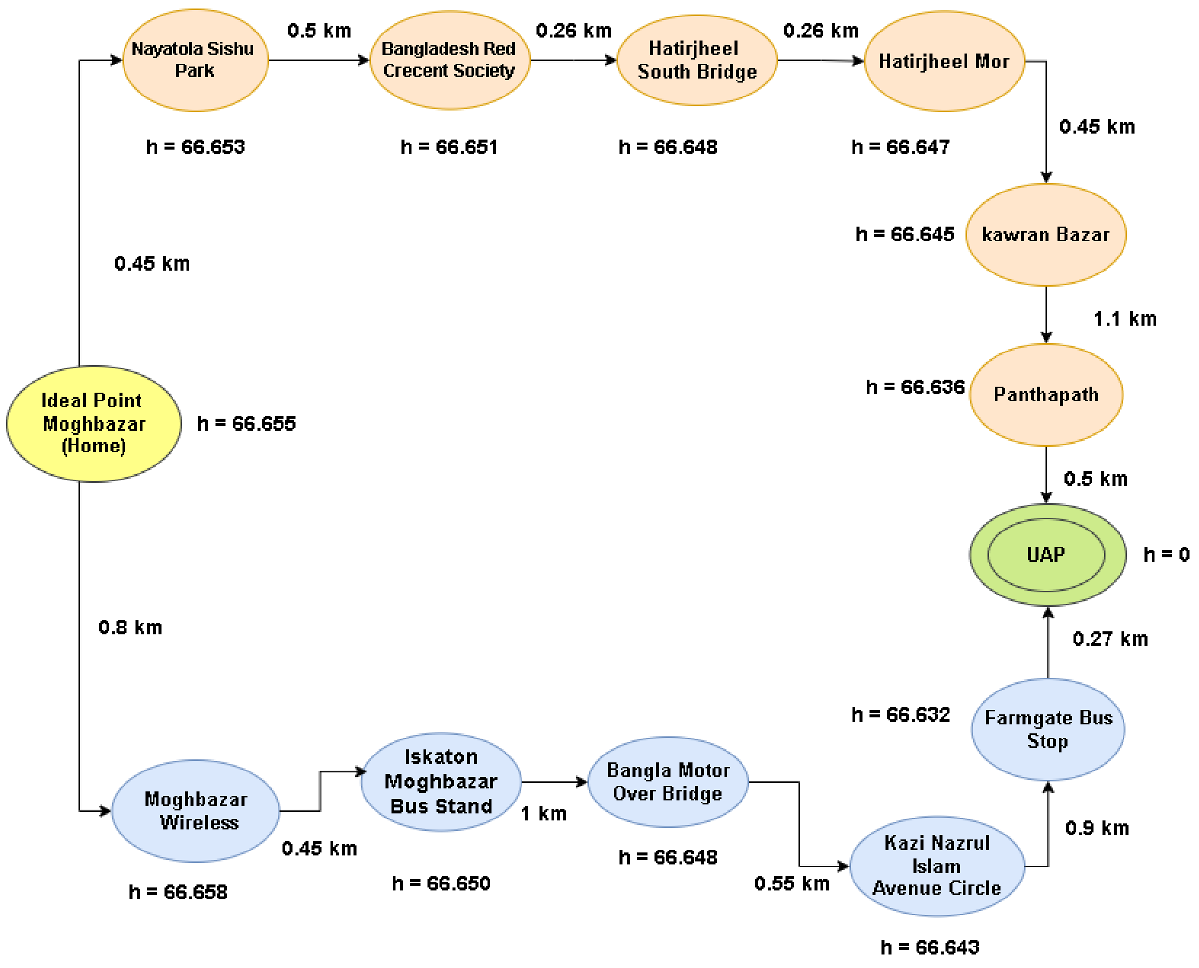
In this task, the goal node is (UAP)

Tools & Language Used:

- For Diagram: draw.io
- Programming Language: Python
- Tools: Google Colab Notebook

Diagram:

Designed Map/Graph:



Graph Overview

Here,

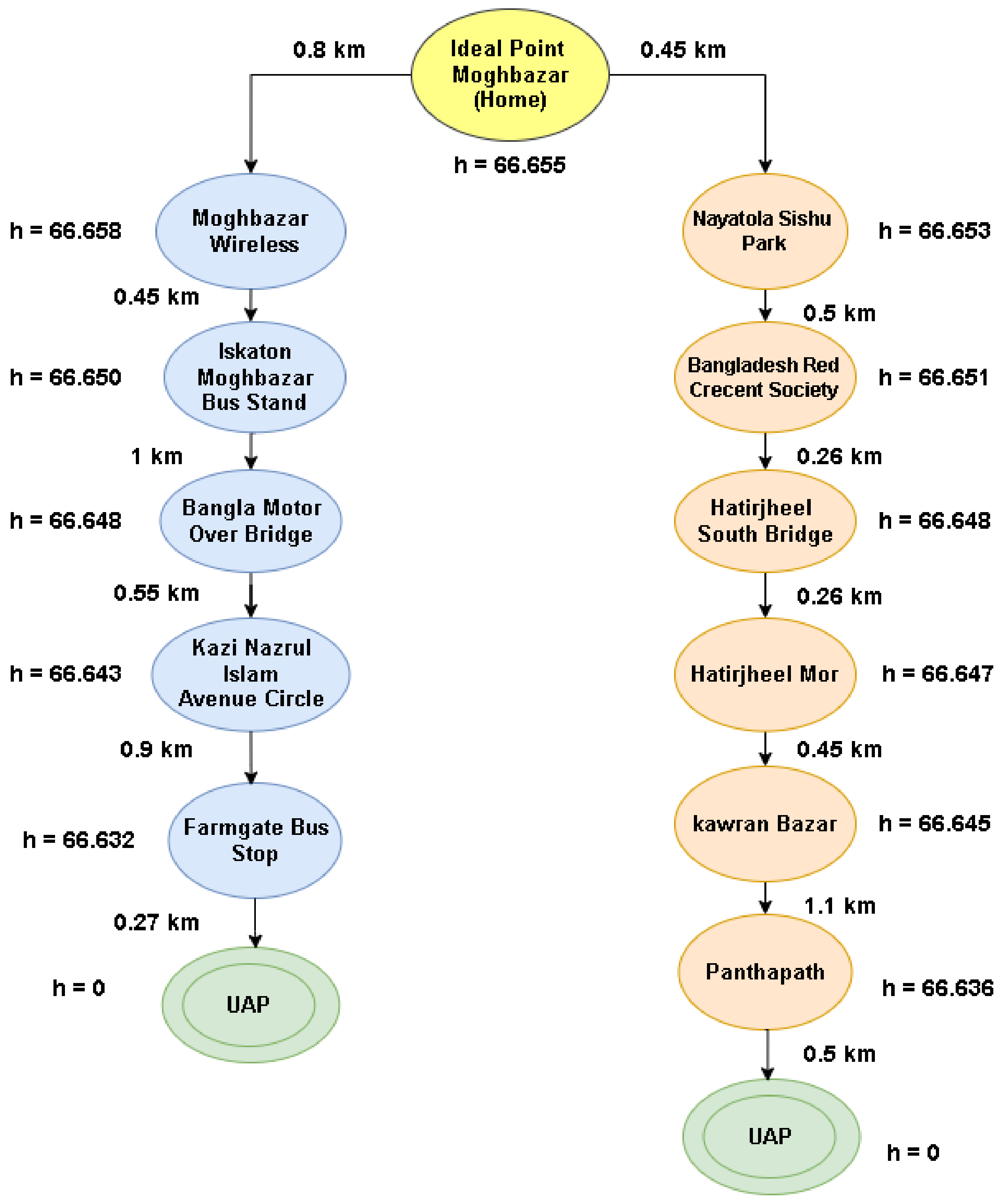
Start Node: Moghbazar Ideal Point (Home)

Goal Node: UAP(University of Asia Pacific)

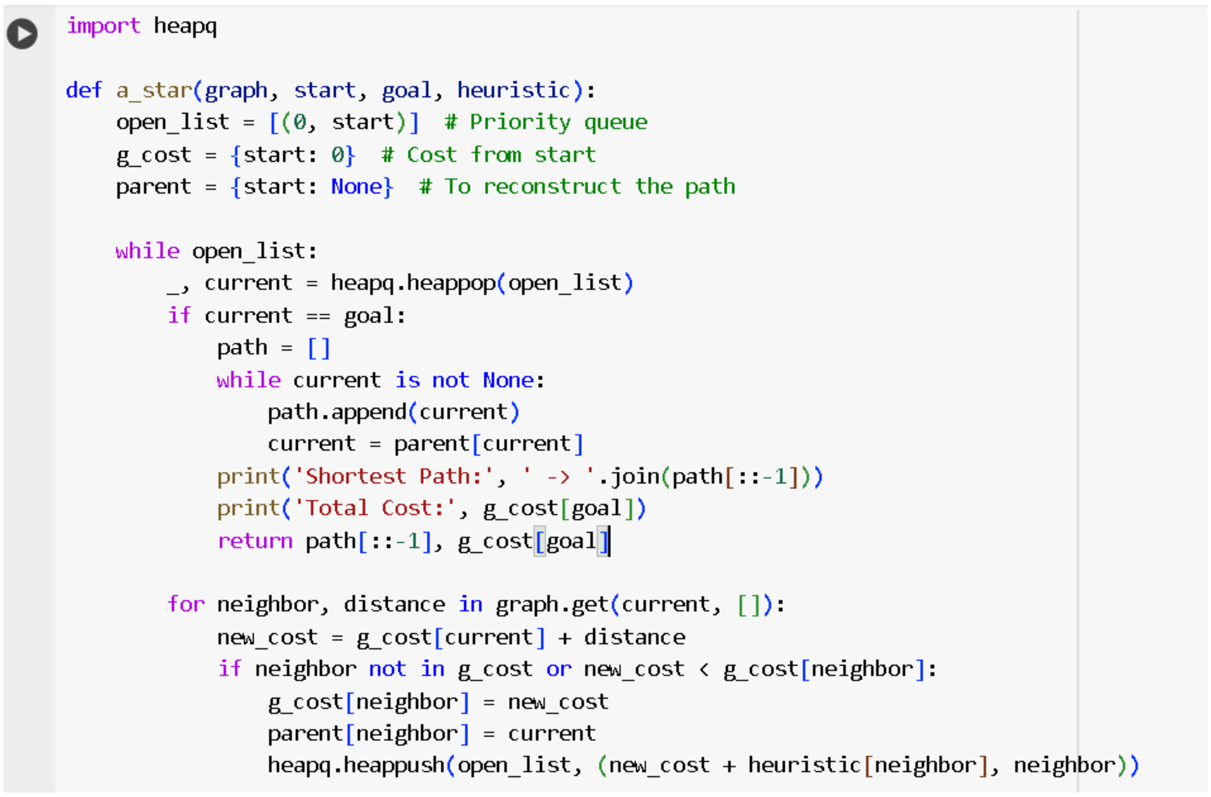
g(n): This represents the actual path cost between nodes, measured in kilometers using Google Maps.

h(n): This is the heuristic estimate to the goal, using the Manhattan Distance. In this task, the heuristic is derived from the absolute differences in longitude and latitude values obtained from Google Maps (Longitude - Latitude).

Designed Search Tree:



Source Code:

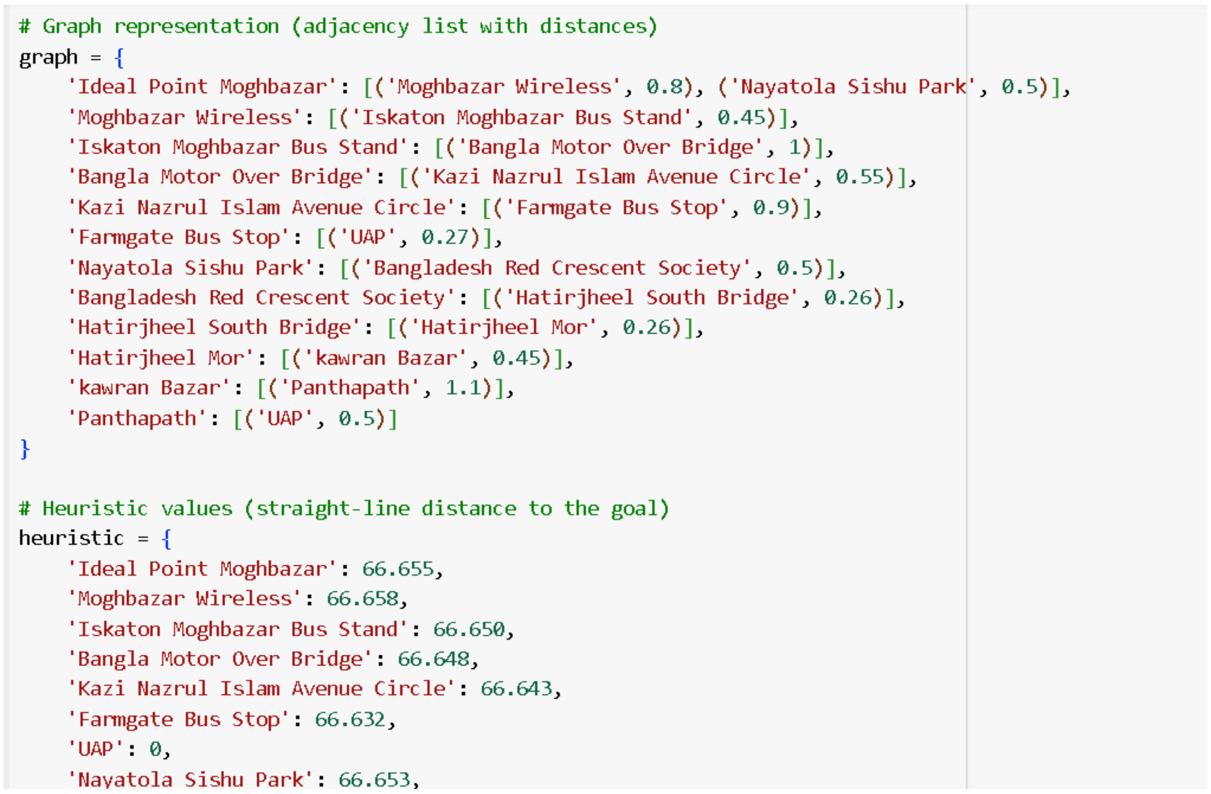


```python
import heapq

def a_star(graph, start, goal, heuristic):
    open_list = [(0, start)]  # Priority queue
    g_cost = {start: 0}  # Cost from start
    parent = {start: None}  # To reconstruct the path

    while open_list:
        _, current = heapq.heappop(open_list)
        if current == goal:
            path = []
            while current is not None:
                path.append(current)
                current = parent[current]
            print('Shortest Path:', ' -> '.join(path[::-1]))
            print('Total Cost:', g_cost[goal])
            return path[::-1], g_cost[goal]

        for neighbor, distance in graph.get(current, []):
            new_cost = g_cost[current] + distance
            if neighbor not in g_cost or new_cost < g_cost[neighbor]:
                g_cost[neighbor] = new_cost
                parent[neighbor] = current
                heapq.heappush(open_list, (new_cost + heuristic[neighbor], neighbor))
```



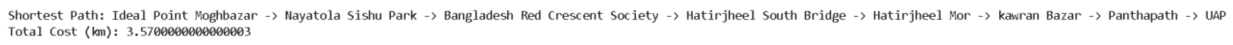
```python
# Graph representation (adjacency list with distances)
graph = {
    'Ideal Point Moghbazar': [('Moghbazar Wireless', 0.8), ('Nayatola Sishu Park', 0.5)],
    'Moghbazar Wireless': [('Iskaton Moghbazar Bus Stand', 0.45)],
    'Iskaton Moghbazar Bus Stand': [('Bangla Motor Over Bridge', 1)],
    'Bangla Motor Over Bridge': [('Kazi Nazrul Islam Avenue Circle', 0.55)],
    'Kazi Nazrul Islam Avenue Circle': [('Farmgate Bus Stop', 0.9)],
    'Farmgate Bus Stop': [('UAP', 0.27)],
    'Nayatola Sishu Park': [('Bangladesh Red Crescent Society', 0.5)],
    'Bangladesh Red Crescent Society': [('Hatirjheel South Bridge', 0.26)],
    'Hatirjheel South Bridge': [('Hatirjheel Mor', 0.26)],
    'Hatirjheel Mor': [('kawran Bazar', 0.45)],
    'kawran Bazar': [('Panthapath', 1.1)],
    'Panthapath': [('UAP', 0.5)]
}

# Heuristic values (straight-line distance to the goal)
heuristic = {
    'Ideal Point Moghbazar': 66.655,
    'Moghbazar Wireless': 66.658,
    'Iskaton Moghbazar Bus Stand': 66.650,
    'Bangla Motor Over Bridge': 66.648,
    'Kazi Nazrul Islam Avenue Circle': 66.643,
    'Farmgate Bus Stop': 66.632,
    'UAP': 0,
    'Nayatola Sishu Park': 66.653,
# Heuristic values (straight-line distance to the goal)
heuristic = {
    'Ideal Point Moghbazar': 66.655,
    'Moghbazar Wireless': 66.658,
    'Iskaton Moghbazar Bus Stand': 66.650,
    'Bangla Motor Over Bridge': 66.648,
    'Kazi Nazrul Islam Avenue Circle': 66.643,
    'Farmgate Bus Stop': 66.632,
    'UAP': 0,
    'Nayatola Sishu Park': 66.653,
    'Bangladesh Red Crescent Society': 66.651,
    'Hatirjheel South Bridge': 66.648,
    'Hatirjheel Mor': 66.647,
    'kawran Bazar': 66.645,
    'Panthapath': 66.636
}

# Find and print the shortest path

print('Shortest Path:', ' -> '.join(path))
print('Total Cost (km):', cost)
```

Output:



```
Shortest Path: Ideal Point Moghbazar -> Nayatola Sishu Park -> Bangladesh Red Crescent Society -> Hatirjheel South Bridge -> Hatirjheel Mor -> kawran Bazar -> Panthapath -> UAP
Total Cost (km): 3.5700000000000003
```

Shortest Path: Ideal Point Moghbazar -> Nayatola Sishu Park -> Bangladesh Red Crescent Society -> Hatirjheel South Bridge -> Hatirjheel Mor -> kawran Bazar -> Panthapath -> UAP

Total Cost (km): 3.5700

Conclusion:

Through the implementation of the A* search algorithm, we identified the most efficient route from Moghbazar Ideal Point to UAP, successfully minimizing the overall travel distance. The algorithm adeptly combines the actual travel cost /path cost(g(n)) with the heuristic estimate (h(n)), ensuring that the selected path is both optimal and reliable. This balance between measured and estimated distances confirms the method's effectiveness in navigating complex route networks.