Department of Computer Science & Engineering

Course Title : Artificial Intelligence and Expert Systems Lab

Course Code : CSE 404

Lab Report : 02

Submission Date : 01.04.25

Submitted To:

Noor Mairukh Khan Arnob

Lecturer,

Department of CSE, UAP

Submitted By:

Susmita Roy

Reg: 21201199

Sec: B2

Problem Title:

Finding the Optimal Path from Azimpur Bus Stand to UAP Using A* Search Algorithm.

Problem Description:

The objective of this problem is to determine the optimal path from Azimpur Bus Stand(home) to UAP(University of Asia Pacific) using the A* search algorithm.

A* search algorithm,

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

Where,

f(n) = Fitness number

g(n) = Cost from start node to n-node

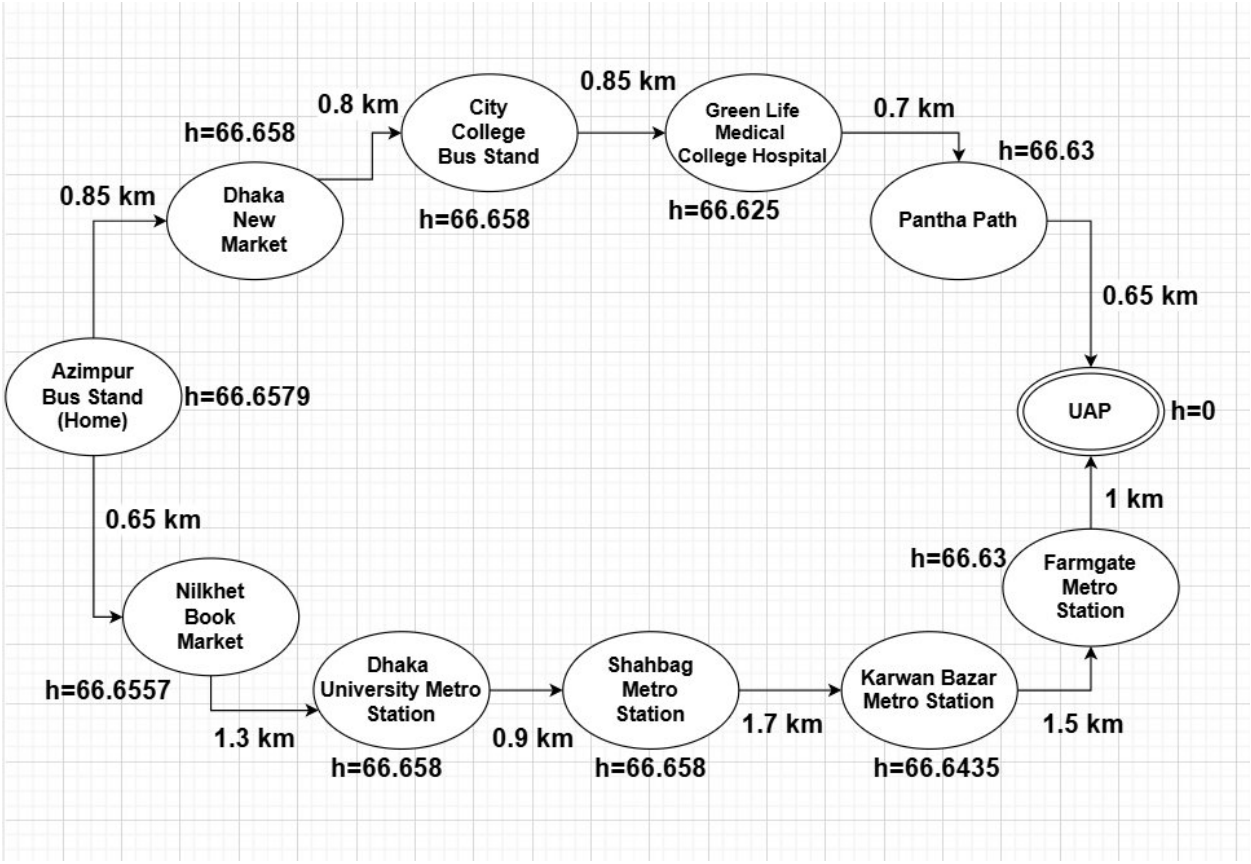
h(n) = Cost from n-node to goal node

Tools and Languages Used:

- Programming Language: Python
- Tools: Colab Notebook

Diagram:

Designed Graph



Here,

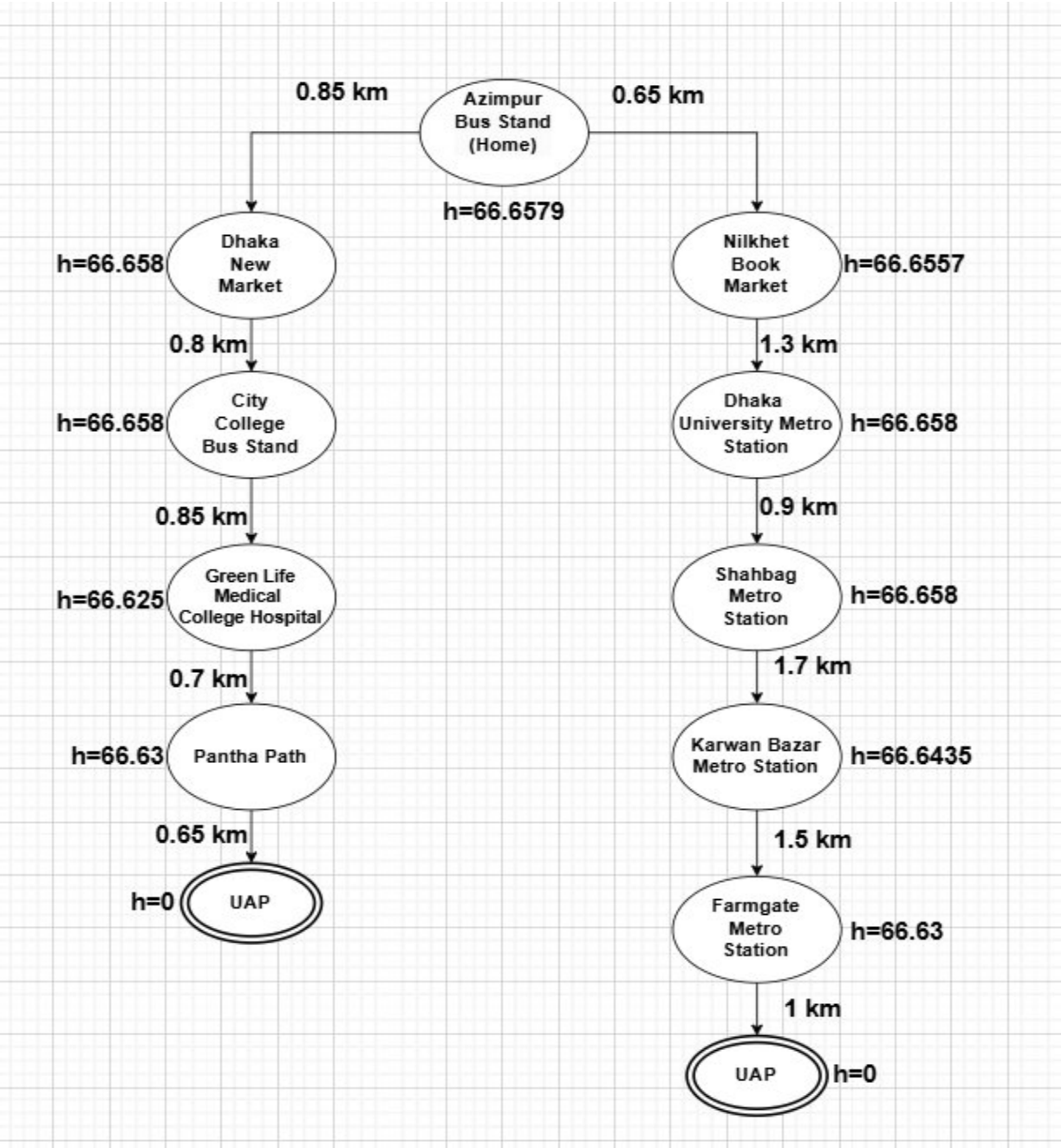
Start Node : Azimpur Bus Stand(Home)

Goal Node : UAP(University of Asia Pacific)

g(n) : Calculated in Kilo meter(km) from Google Maps

h(n) : Calculated from Google Maps(Longitude,Latitude),using Manhattan Distance(Longitude - Latitude).

Designed Search Tree



Sample Input/Output:

```
import heapq

def astar(graph, heuristic, start, goal):
    open_list = []  # Priority queue
    heapq.heappush(open_list, (0, start))  # (cost, node)
    came_from = {}  # Track path
    g_score = {node: float('inf') for node in graph}
    g_score[start] = 0
    f_score = {node: float('inf') for node in graph}
    f_score[start] = heuristic.get(start, float('inf'))

    while open_list:
        _, current = heapq.heappop(open_list)

        if current == goal:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            path.append(start)
            return path[::-1], g_score[goal]  # Return path and cost

        for neighbor, cost in graph.get(current, {}).items():
            tentative_g_score = g_score[current] + cost
            if tentative_g_score < g_score.get(neighbor, float('inf')):
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g_score
                f_score[neighbor] = tentative_g_score + heuristic.get(neighbor, float('inf'))
                heapq.heappush(open_list, (f_score[neighbor], neighbor))

    return None, float('inf')  # No path found
```

```
# Graph representation with distances (g(n))
graph = {
    "Azimpur": {"Dhaka New Market": 0.85, "Nilkhet": 0.65},
    "Dhaka New Market": {"City College": 0.8},
    "City College": {"Green Life Hospital": 0.85},
    "Green Life Hospital": {"Pantha Path": 0.7},
    "Pantha Path": {"UAP": 0.65},
    "Nilkhet": {"Dhaka Univ. Metro": 1.3},
    "Dhaka Univ. Metro": {"Shahbag Metro": 0.9},
    "Shahbag Metro": {"Karwan Bazar Metro": 1.7},
    "Karwan Bazar Metro": {"Farmgate Metro": 1.5},
    "Farmgate Metro": {"UAP": 1.0},
    "UAP": {}
}
# Heuristic values (h(n)) estimated based on given h values
heuristic = {
    "Azimpur": 66.6579,
    "Dhaka New Market": 66.658,
    "City College": 66.658,
    "Green Life Hospital": 66.625,
    "Pantha Path": 66.63,
    "UAP": 0,
    "Nilkhet": 66.6557,
    "Dhaka Univ. Metro": 66.658,
    "Shahbag Metro": 66.658,
    "Karwan Bazar Metro": 66.6435,
    "Farmgate Metro": 66.63,
}

# Finding the optimal path
start_node = "Azimpur"
goal_node = "UAP"
path, cost = astar(graph, heuristic, start_node, goal_node)
```

```
# Output results
if path:
    print("Optimal Path:", " -> ".join(path))
    print("Total Cost (km):", cost)
else:
    print("No valid path found.")
```

Output:

```
Optimal Path: Azimpur -> Dhaka New Market -> City College -> Green Life Hospital -> Pantha Path -> UAP
Total Cost (km): 3.85
```

Conclusion:

By implementing the A* search algorithm, we successfully determined the most optimal path from Azimpur Bus Stand to UAP while minimizing travel distance. The algorithm efficiently balances actual travel cost (g(n)) and estimated distance (h(n)), ensuring the shortest possible route.