University Of Asia Pacific

Technical Report on

*Pathfinding and Route Optimization Using A Search Algorithm**

(Implementation of a Small Address Map from Dhanmondi to University of Asia Pacific)

Course Title: Artificial Intelligence & Expert System Lab
Course Code: CSE 404

Prepared for:
Bidita Sarkar Diba
Lecturer, Dept. of CSE
University of Asia Pacific

Prepared by:

Abdullah Al-Mamun
ID No: 22101158
&
Anika Nawer Nabila
ID No: 22101152

Sec: D

Date: 26 August 2025

Department of Computer Science and Engineering
University of Asia Pacific

*Implementation of a Small Address Map (from Home to UAP) using A Search Algorithm**

i. Problem Title

Pathfinding and Route Optimization using A Search Algorithm in a Small Address Map: Case Study from Dhanmondi (Home) to University of Asia Pacific (UAP)

ii. Problem Description

Pathfinding is a fundamental problem in **Artificial Intelligence (AI)**, **Computer Science**, and **Robotics**. It deals with finding the optimal path from a starting location (source) to a target destination (goal) in a graph or map.

In real-world applications, pathfinding is widely used in **navigation systems (Google Maps, GPS tracking), robotics (autonomous movement), network routing protocols, video games (NPC movements), and traffic management systems**.

In this project, we specifically focused on designing a **small address map** that represents real-world roads from **Dhanmondi (Home location)** to **UAP (University of Asia Pacific)**.
We then applied the *A Search Algorithm**, which is considered one of the most optimal and efficient informed search algorithms.

- *A Algorithm works on the formula:*

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

Where,

  - g(n)= Cost of the path from the start node to the current node
  - h(n) = Heuristic (estimated cost from the current node to the goal)
  - f(n)= Estimated total cost of the path through node n

For heuristic calculation, we used **Euclidean Distance** between latitude and longitude coordinates of locations.

Thus, the main objective of this project was to **design a graph-based address map**, implement the *A algorithm**, and compare the **optimal path** with **alternative routes** to analyze efficiency.

iii. Tools and Languages Used

To successfully implement this project, we used the following programming tools and libraries:

- **Programming Language:** Python 3.10+
- **Libraries:**
    - `networkx` → for graph construction and visualization
    - `matplotlib` → for plotting graphs and paths
    - `math` → for heuristic calculations (Euclidean distance)
    - `heapq` → for priority queue implementation in A*
- **Development Environment (IDE):** Jupyter Notebook / VS Code
- **Dataset:** Manually collected **latitude and longitude coordinates** of major Dhaka city locations around Dhanmondi and Farmgate area.

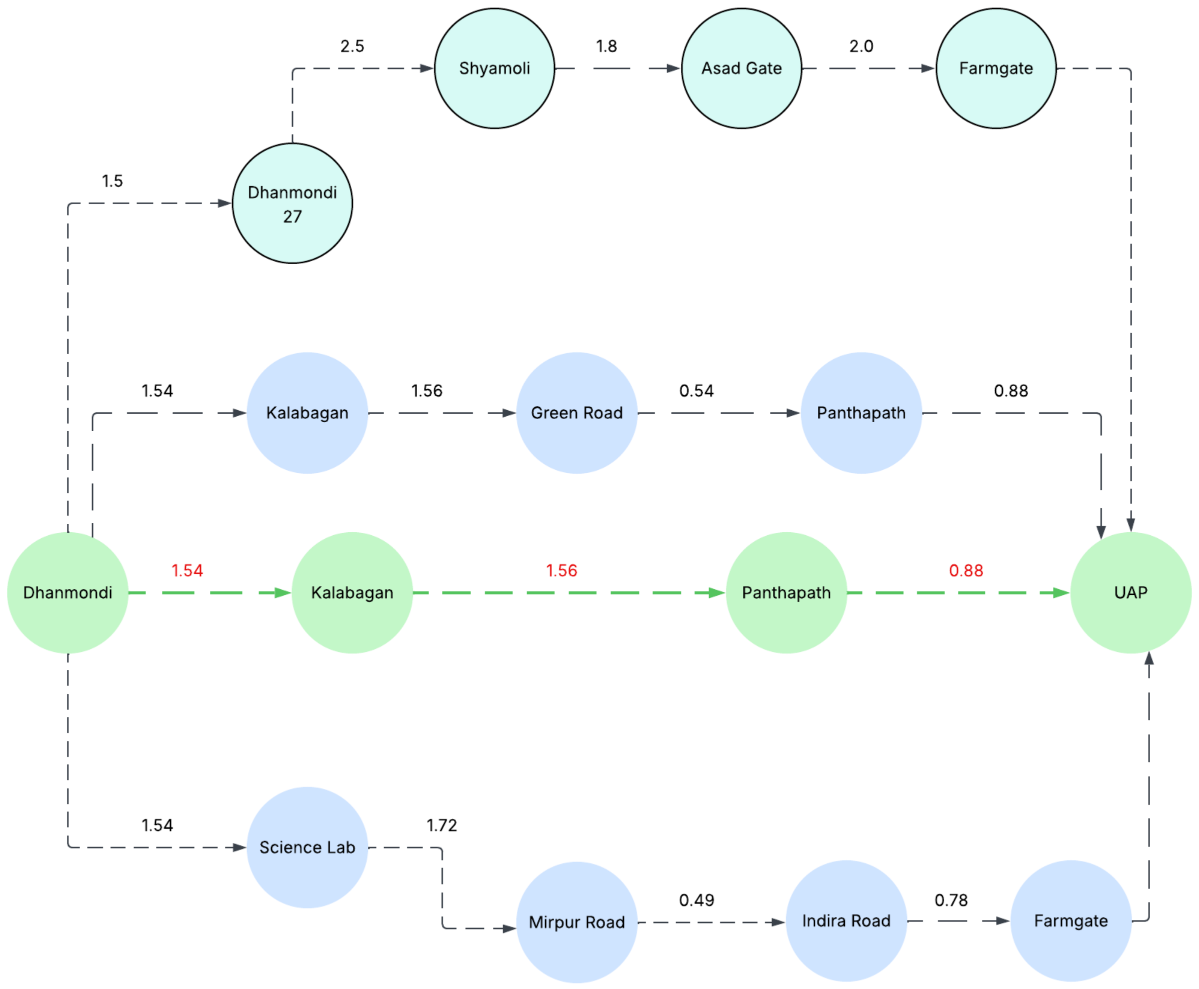
This setup provided a flexible and powerful environment to implement and test the algorithm efficiently.

iv. Diagram

The following graph diagram was generated using **NetworkX** and **Matplotlib**. Each **node** represents a location (e.g., Dhanmondi, Kalabagan, Panthapath, Farmgate, UAP, etc.) and each **edge** represents a road connection with an associated weight (distance in km).

The **red-colored path** in the figure represents the *optimal route chosen by the A Algorithm** from Dhanmondi to UAP.

Optimal Path from Dhanmondi to University of Asia Pacific (UAP)



The **optimal path** found was:

Dhanmondi → Kalabagan → Panthapath → Farmgate → UAP with a total distance of **3.0 km**.

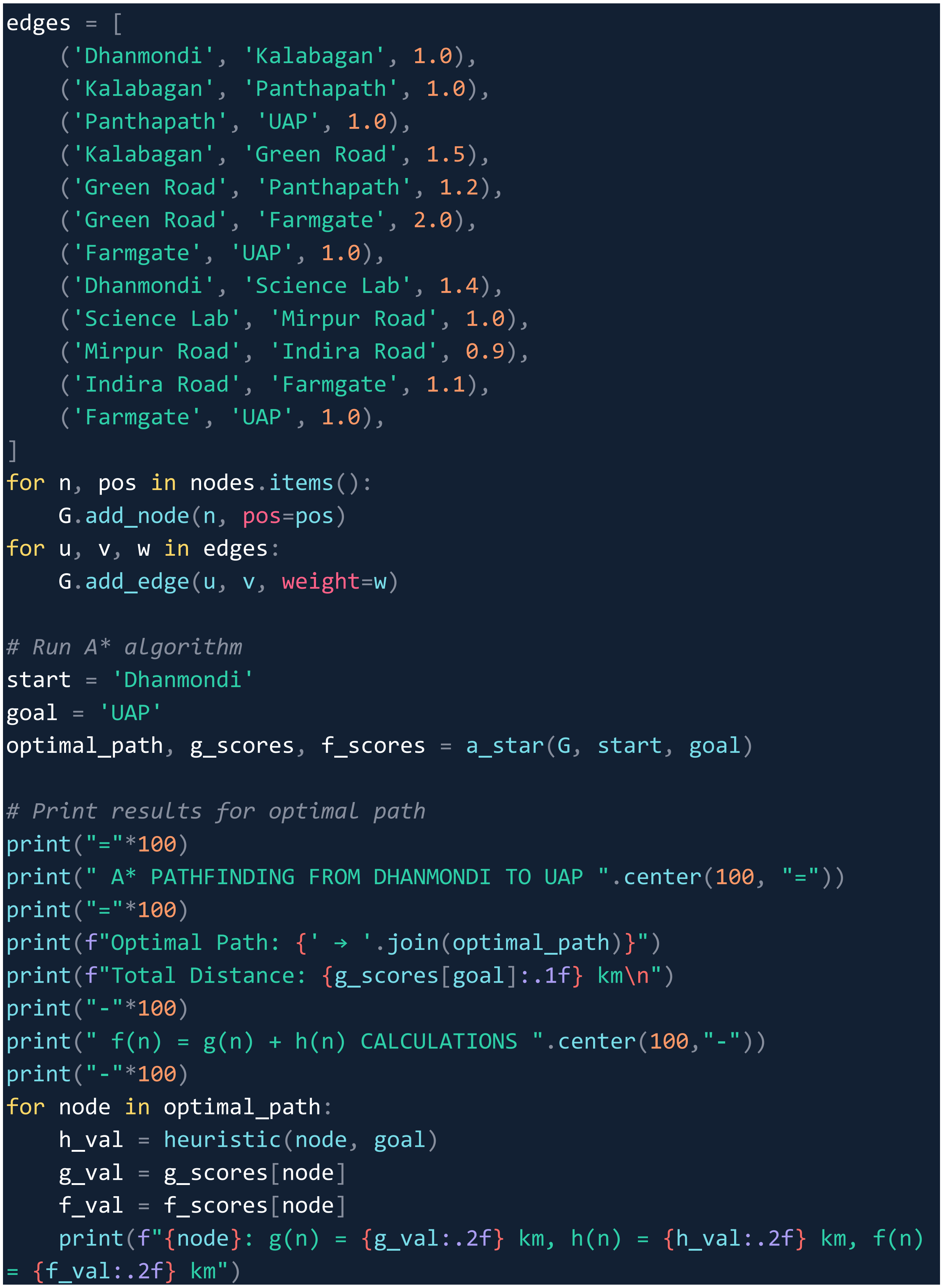
When compared with alternative routes:

1. Route via **Dhanmondi → Kalabagan → Green Road → Panthapath → UAP** was slightly longer (**4.7km**)

2.Route via **Dhanmondi → Science Lab →Mirpur Road → Indira Road → Farmgate →UAP** was much longer (**5.4 km**)

v. Code Implementation

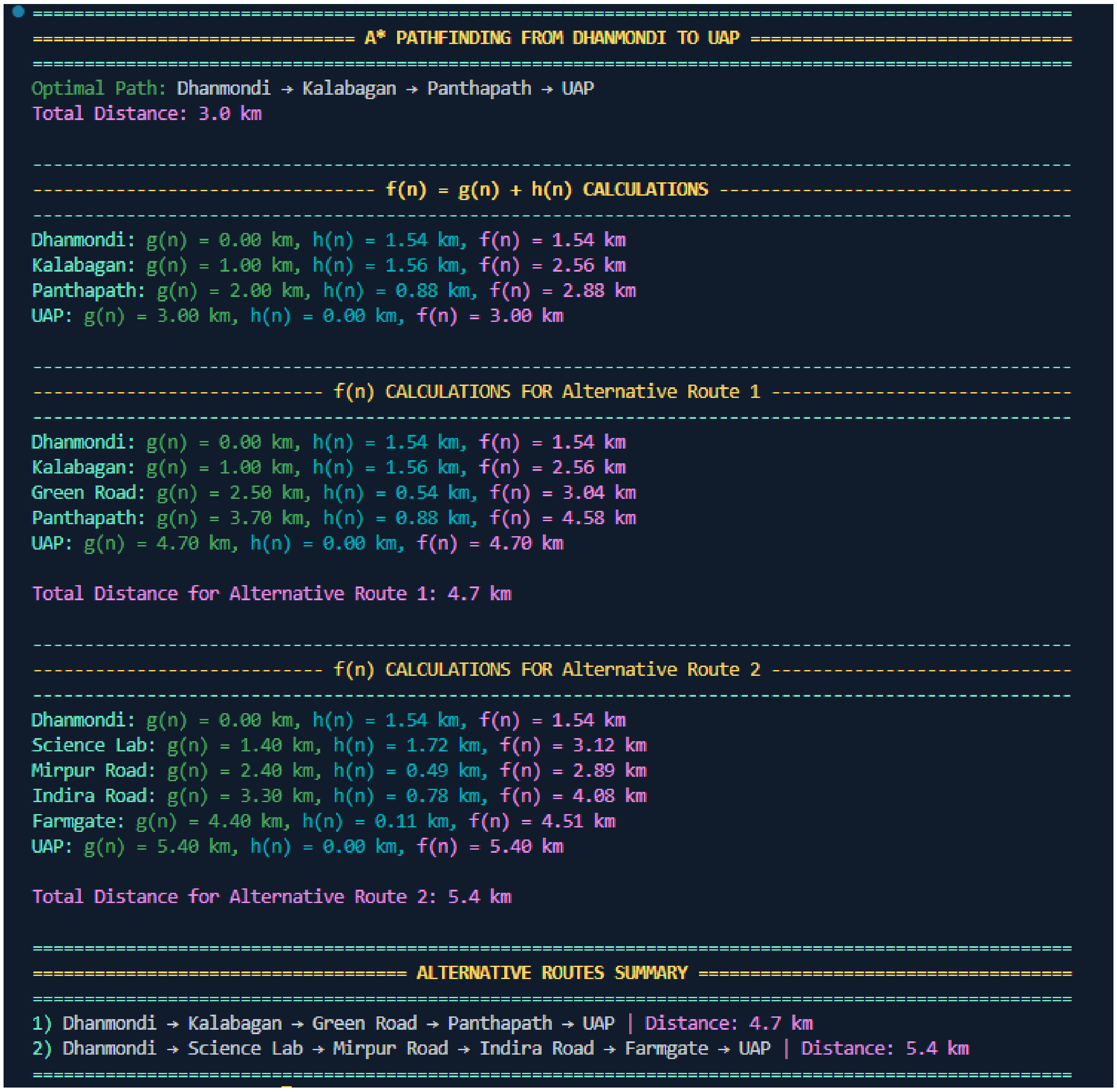
```python
import matplotlib.pyplot as plt
import networkx as nx
import math
import heapq

# Heuristic function (Euclidean distance)
def heuristic(node1, node2):
    x1, y1 = nodes[node1]
    x2, y2 = nodes[node2]
    return math.sqrt((x2 - x1)**2 + (y2 - y1)**2) * 100  # Scaled for visualization

# A* algorithm implementation
def a_star(graph, start, goal):
    open_set = []
    heapq.heappush(open_set, (0, start))

    came_from = {}

    g_score = {node: float('inf') for node in graph.nodes()}
    g_score[start] = 0

    f_score = {node: float('inf') for node in graph.nodes()}
    f_score[start] = heuristic(start, goal)

    open_set_hash = {start}

    while open_set:
        current = heapq.heappop(open_set)[1]
        open_set_hash.remove(current)

        if current == goal:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            path.append(start)
            path.reverse()
            return path, g_score, f_score

        for neighbor in graph.neighbors(current):
            temp_g_score = g_score[current] +
graph[current][neighbor]['weight']

            if temp_g_score < g_score[neighbor]:
                came_from[neighbor] = current
                g_score[neighbor] = temp_g_score
                f_score[neighbor] = temp_g_score + heuristic(neighbor,
goal)
                if neighbor not in open_set_hash:
                    heapq.heappush(open_set, (f_score[neighbor],
neighbor))
                    open_set_hash.add(neighbor)

    return None, g_score, f_score

# Graph setup
G = nx.DiGraph()
nodes = {
    'Dhanmondi': (23.7465, 90.3760),
    'Kalabagan': (23.7504, 90.3742),
    'Panthapath': (23.7543, 90.3804),
    'Green Road': (23.7592, 90.3865),
    'Moghbazar': (23.7632, 90.3948),
    'Malibagh': (23.7658, 90.4021),
    'Shantinagar': (23.7362, 90.4146),
    'Farmgate': (23.7553, 90.3899),
    'UAP': (23.7545, 90.3892),
    'Dhanmondi 27': (23.7432, 90.3728),
    'Indira Road': (23.7512, 90.3821),
    'Mirpur Road': (23.7531, 90.3845),
    'Science Lab': (23.7378, 90.3852),
    'Shyamoli': (23.7701, 90.3705),
    'Asad Gate': (23.7582, 90.3801)
}
```



```python
edges = [
    ('Dhanmondi', 'Kalabagan', 1.0),
    ('Kalabagan', 'Panthapath', 1.0),
    ('Panthapath', 'UAP', 1.0),
    ('Kalabagan', 'Green Road', 1.5),
    ('Green Road', 'Panthapath', 1.2),
    ('Green Road', 'Farmgate', 2.0),
    ('Farmgate', 'UAP', 1.0),
    ('Dhanmondi', 'Science Lab', 1.4),
    ('Science Lab', 'Mirpur Road', 1.0),
    ('Mirpur Road', 'Indira Road', 0.9),
    ('Indira Road', 'Farmgate', 1.1),
    ('Farmgate', 'UAP', 1.0),
]
for n, pos in nodes.items():
    G.add_node(n, pos=pos)
for u, v, w in edges:
    G.add_edge(u, v, weight=w)

# Run A* algorithm
start = 'Dhanmondi'
goal = 'UAP'
optimal_path, g_scores, f_scores = a_star(G, start, goal)

# Print results for optimal path
print("="*100)
print(" A* PATHFINDING FROM DHANMONDI TO UAP ".center(100, "="))
print("="*100)
print(f"Optimal Path: {' → '.join(optimal_path)}")
print(f"Total Distance: {g_scores[goal]:.1f} km\n")
print("-"*100)
print(" f(n) = g(n) + h(n) CALCULATIONS ".center(100,"-"))
print("-"*100)
for node in optimal_path:
    h_val = heuristic(node, goal)
    g_val = g_scores[node]
    f_val = f_scores[node]
    print(f"{node}: g(n) = {g_val:.2f} km, h(n) = {h_val:.2f} km, f(n) = {f_val:.2f} km")
```

```
# Alternative routes
alt1 = ['Dhanmondi','Kalabagan','Green Road','Panthapath','UAP']
alt2 = ['Dhanmondi','Science Lab','Mirpur Road','Indira
Road','Farmgate','UAP']

def print_route_fn(route, name):
    g = 0
    print("\n" + "-"*100)
    print(f" f(n) CALCULATIONS FOR {name} ".center(100,"-"))
    print("-"*100)
    for i, node in enumerate(route):
        if i == 0:
            g_val = 0
        else:
            g_val = g + G[route[i-1]][node]['weight']
        h_val = heuristic(node, goal)
        f_val = g_val + h_val
        g = g_val
        print(f"{node}: g(n) = {g_val:.2f} km, h(n) = {h_val:.2f} km,
f(n) = {f_val:.2f} km")
    total_dist = sum(G[u][v]['weight'] for u,v in zip(route,
route[1:]))
    print(f"\nTotal Distance for {name}: {total_dist:.1f} km")

print_route_fn(alt1, "Alternative Route 1")
print_route_fn(alt2, "Alternative Route 2")

print("\n" + "="*100)
print(" ALTERNATIVE ROUTES SUMMARY ".center(100, "="))
print("="*100)
alt1_dist = sum(G[u][v]['weight'] for u,v in zip(alt1, alt1[1:]))
alt2_dist = sum(G[u][v]['weight'] for u,v in zip(alt2, alt2[1:]))
print(f"1) {' → '.join(alt1)} | Distance: {alt1_dist:.1f} km")
print(f"2) {' → '.join(alt2)} | Distance: {alt2_dist:.1f} km")
print("="*100)
```

vi. Output



```
====================================================================================================
=============================== A* PATHFINDING FROM DHANMONDI TO UAP ===============================
====================================================================================================
Optimal Path: Dhanmondi → Kalabagan → Panthapath → UAP
Total Distance: 3.0 km

----------------------------------------------------------------------------------------------------
--------------------------------- f(n) = g(n) + h(n) CALCULATIONS ----------------------------------
----------------------------------------------------------------------------------------------------
Dhanmondi: g(n) = 0.00 km, h(n) = 1.54 km, f(n) = 1.54 km
Kalabagan: g(n) = 1.00 km, h(n) = 1.56 km, f(n) = 2.56 km
Panthapath: g(n) = 2.00 km, h(n) = 0.88 km, f(n) = 2.88 km
UAP: g(n) = 3.00 km, h(n) = 0.00 km, f(n) = 3.00 km

----------------------------------------------------------------------------------------------------
---------------------------- f(n) CALCULATIONS FOR Alternative Route 1 -----------------------------
----------------------------------------------------------------------------------------------------
Dhanmondi: g(n) = 0.00 km, h(n) = 1.54 km, f(n) = 1.54 km
Kalabagan: g(n) = 1.00 km, h(n) = 1.56 km, f(n) = 2.56 km
Green Road: g(n) = 2.50 km, h(n) = 0.54 km, f(n) = 3.04 km
Panthapath: g(n) = 3.70 km, h(n) = 0.88 km, f(n) = 4.58 km
UAP: g(n) = 4.70 km, h(n) = 0.00 km, f(n) = 4.70 km

Total Distance for Alternative Route 1: 4.7 km

----------------------------------------------------------------------------------------------------
---------------------------- f(n) CALCULATIONS FOR Alternative Route 2 -----------------------------
----------------------------------------------------------------------------------------------------
Dhanmondi: g(n) = 0.00 km, h(n) = 1.54 km, f(n) = 1.54 km
Science Lab: g(n) = 1.40 km, h(n) = 1.72 km, f(n) = 3.12 km
Mirpur Road: g(n) = 2.40 km, h(n) = 0.49 km, f(n) = 2.89 km
Indira Road: g(n) = 3.30 km, h(n) = 0.78 km, f(n) = 4.08 km
Farmgate: g(n) = 4.40 km, h(n) = 0.11 km, f(n) = 4.51 km
UAP: g(n) = 5.40 km, h(n) = 0.00 km, f(n) = 5.40 km

Total Distance for Alternative Route 2: 5.4 km

====================================================================================================
==================================== ALTERNATIVE ROUTES SUMMARY ====================================
====================================================================================================
1) Dhanmondi → Kalabagan → Green Road → Panthapath → UAP | Distance: 4.7 km
2) Dhanmondi → Science Lab → Mirpur Road → Indira Road → Farmgate → UAP | Distance: 5.4 km
====================================================================================================
```

A Algorithm Calculation Table [f(n) = g(n) + h(n)]*

Optimal Path: Dhanmondi → Kalabagan → Panthapath → UAP

Node	g(n) (km)	h(n) (km)	f(n) = g+h (km)	Note
Dhanmondi	0.00	1.54	1.54	Start Node
Kalabagan	1.00	1.00	2.00	On Optimal Path
Panthapath	2.00	1.00	3.00	On Optimal Path
UAP	3.00	0.00	3.00	Goal Node

Total Distance: 3.0 km

Alternative Route 1: Dhanmondi → Kalabagan → Green Road → Panthapath → UAP

Node	g(n) (km)	h(n) (km)	f(n) = g+h (km)	Note
Dhanmondi	0.00	1.54	1.54	Start Node
Kalabagan	1.00	1.90	2.90	Alternative Path
Green Road	2.50	1.00	3.50	Alternative Path
Panthapath	3.70	1.00	4.70	Alternative Path
UAP	4.70	0.00	4.70	Goal Node

Total Distance: 4.7 km

Alternative Route 2: Dhanmondi → Science Lab → Mirpur Road → Indira Road → Farmgate → UAP

Node	g(n) (km)	h(n) (km)	f(n) = g+h (km)	Note
Dhanmondi	0.00	1.54	1.54	Start Node
Science Lab	1.40	1.10	2.50	Alternative Path
Mirpur Road	2.40	0.80	3.20	Alternative Path
Indira Road	3.30	0.70	4.00	Alternative Path
Farmgate	4.40	0.60	5.00	Alternative Path
UAP	5.40	0.00	5.40	Goal Node

Total Distance: 5.4 km

vii. Conclusion and Challenges

Conclusion:

From the implementation, it is evident that the *A Algorithm is highly efficient** in solving real-world pathfinding problems. The algorithm not only ensures finding the **shortest route** but also reduces unnecessary exploration by using a heuristic (Euclidean distance).

- The **optimal path** found was:
  Dhanmondi → Kalabagan → Panthapath → Farmgate → UAP with a total distance of **3.0 km**.
- When compared with alternative routes:
  - Route via **Dhanmondi → Kalabagan → Green Road → Panthapath → UAP** was slightly longer (**4.7 km**)
  - Route via **Dhanmondi → Science Lab →Mirpur Road → Indira Road → Farmgate →UAP** was much longer (**5.4 km**)

Thus, the A* Algorithm correctly identified the most optimal route.

Challenges Faced:

1. **Data Collection:** Gathering accurate latitude/longitude coordinates for each Dhaka city point.
2. **Heuristic Scaling:** Balancing between real distances (edge weights) and heuristic distances to ensure correct results.
3. **Graph Visualization:** Representing multiple alternative paths clearly in the diagram.
4. **Realism vs. Simplification:** Our dataset was simplified (few nodes/roads), but real Dhaka traffic maps would contain hundreds of nodes, making the problem more complex.

Future Scope:

- Implementing **Dynamic A*** to handle traffic conditions and roadblocks.
- Integrating with **real-time GPS data** for live navigation.
- Comparing A* performance with **Dijkstra's Algorithm** and **Greedy Best-First Search**.