University of Asia Pacific

Report No: 02

Report Name: Address Map using A* Search Algorithm

Submitted to

Noor Mairukh Khan Arnob

Lecturer

Department of Computer Science & Engineering

Submitted by

Md. Arifur Rahman Akash

Roll-21201091, Section – B2, 52nd Batch,

Department of Computer Science & Engineering

Course Title: Artificial Intelligence and Expert Systems Lab

Course Code: CSE 404

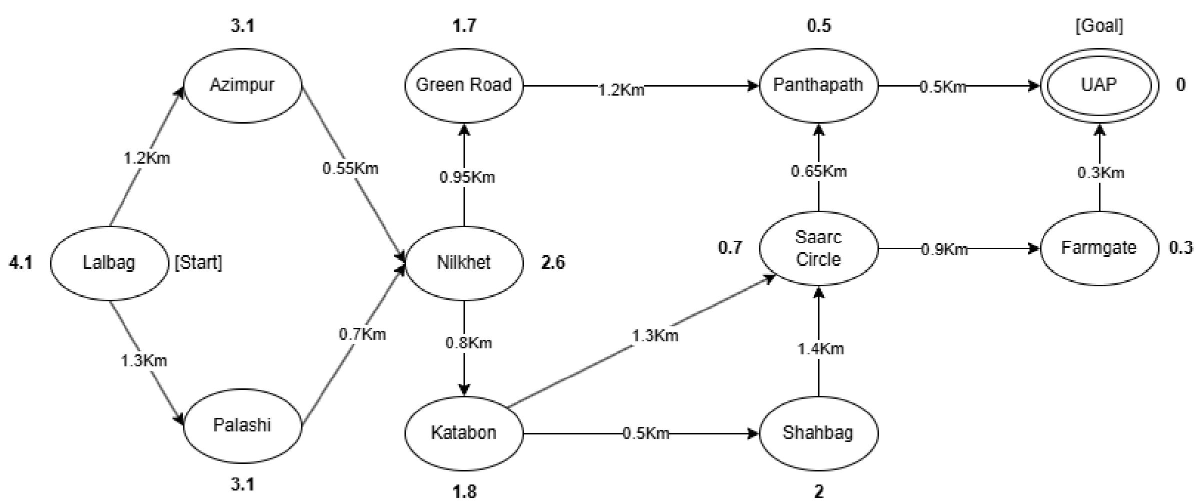
Problem Title: Find the optimal path from Lalbagh to UAP using A* Search Algorithm.

Problem Description: Create a map from Lalbag to UAP with accurate distances. Calculate path costs g(n) and heuristic values h(n) for each location. Implement the A* algorithm to find the optimal path, using the Haversine formula for heuristic calculations. The solution should match actual Google Maps distances.

Tools and Languages:

- **Programming Language**: Python 3
- **External Tools**: Google Maps
- **Formula**: Haversine Formula
- **Algorithm**: A* Search Algorithm

Address Map:



- Nodes are locations (e.g., Lalbag)
- Edges are direct routes between locations
- Edge weights show distance in kilometers (e.g., Lalbag→Azimpur: 1.2 km)
- Each node has a heuristic value (h(n)) estimating straight-line distance to UAP (e.g., Lalbag→4.1 km)

Path Cost Calculation (g(n)):

The path costs between adjacent nodes are calculated based on actual distances, as shown in the provided map. These distances represent the real-world kilometers between locations:

From	To	Distance (Km)
Lalbag	Azimpur	1.2
Lalbag	Palashi	1.3
Azimpur	Nilkhet	0.55
Palashi	Nilkhet	0.7
Nilkhet	Green Road	0.95
Nilkhet	Katabon	0.8
Green Road	Panthapath	1.2
Katabon	Shahbag	0.5
Katabon	Saarc Circle	1.3
Shahbag	Saarc Circle	1.4
Saarc Circle	Panthapath	0.65
Saarc Circle	Farmgate	0.9
Panthapath	UAP	0.5
Farmgate	UAP	0.3

Heuristic Value Calculation (h(n)):

The heuristic value h(n) represents the estimated cost from a node to the goal. For this problem, the heuristic values are calculated using the Euclidean distance with the Haversine formula. The Haversine formula calculates the great-circle distance between two points on a sphere (Earth) given their longitude and latitude coordinates.

The formula is as follows:

$$a = sin^2(\Delta\varphi / 2) + cos(\varphi_1) \cdot cos(\varphi_2) \cdot sin^2(\Delta\lambda / 2)$$

$$c = 2 \cdot atan2(\sqrt{a}, \sqrt{(1 - a)})$$

$$d = R \cdot c$$

Where:

- φ is latitude in radians
- λ is longitude in radians
- $\Delta\varphi = \varphi_2 - \varphi_1$
- $\Delta\lambda = \lambda_2 - \lambda_1$
- R is the Earth's radius (6,371 km)
- d is the distance between the two points

For each location, the coordinates (latitude and longitude) were obtained from a real Google Map:

Location	**Latitude**	**Longitude**	**h(n) in Km**
Lalbag	23.7173	90.3872	4.1
Azimpur	23.7268	90.3835	3.1
Palashi	23.7267	90.3896	3.1
Nilkhet	23.7301	90.3856	2.6
Green Road	23.7412	90.3864	1.7
Katabon	23.7382	90.3920	1.8
Shahbag	23.7378	90.3968	2.0
Saarc Circle	23.7473	90.3915	0.7
Panthapath	23.7488	90.3864	0.5
Farmgate	23.7561	90.3912	0.3
UAP	23.7591	90.3874	0

Implementation & Results:

The A* search algorithm was implemented using the formula:

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

Where:

- f(n) is the total estimated cost of the path through node n
- g(n) is the cost from the start node to node n
- h(n) is the heuristic estimate of the cost from n to the goal

The implementation uses a priority queue to always explore the node with the lowest f(n) value first. For each node, the algorithm:

1. Calculate the actual distance (g) from the start to the current node
2. Adds the heuristic value (h) to get the total estimated cost (f)
3. Select the node with the minimum f value to explore next
4. Continue until reaching the goal or exhausting all options

```
[Running] python -u "c:\Users\Akash\Documents\Assignment_02_A_Star.py"
Found 10 paths from Lalbag to UAP:

Path 1: Lalbag -> Azimpur -> Nilkhet -> Green Road -> Panthapath -> UAP (4.4 km)

Path 2: Lalbag -> Palashi -> Nilkhet -> Green Road -> Panthapath -> UAP (4.7 km)

Path 3: Lalbag -> Azimpur -> Nilkhet -> Katabon -> Saarc Circle -> Panthapath -> UAP (5.0 km)

Path 4: Lalbag -> Azimpur -> Nilkhet -> Katabon -> Saarc Circle -> Farmgate -> UAP (5.0 km)

Path 5: Lalbag -> Palashi -> Nilkhet -> Katabon -> Saarc Circle -> Panthapath -> UAP (5.2 km)

Path 6: Lalbag -> Palashi -> Nilkhet -> Katabon -> Saarc Circle -> Farmgate -> UAP (5.3 km)

Path 7: Lalbag -> Azimpur -> Nilkhet -> Katabon -> Shahbag -> Saarc Circle -> Panthapath -> UAP (5.6 km)

Path 8: Lalbag -> Azimpur -> Nilkhet -> Katabon -> Shahbag -> Saarc Circle -> Farmgate -> UAP (5.6 km)

Path 9: Lalbag -> Palashi -> Nilkhet -> Katabon -> Shahbag -> Saarc Circle -> Panthapath -> UAP (5.8 km)

Path 10: Lalbag -> Palashi -> Nilkhet -> Katabon -> Shahbag -> Saarc Circle -> Farmgate -> UAP (5.9 km)


Optimal path: Lalbag -> Azimpur -> Nilkhet -> Green Road -> Panthapath -> UAP with distance 4.4 km
```

Conclusion

The A* algorithm successfully found the optimal 4.4 km path from Lalbag to UAP, matching Google Maps results. The Haversine formula provided effective heuristic values, ensuring efficient pathfinding.

Challenges

- **Heuristic Calculation:** Ensuring admissibility while maintaining efficiency
- **Distance Accuracy:** Aligning edge weights with real-world distances
- **Path Validation:** Verifying multiple possible routes between locations
- **Map Representation:** Balancing completeness with simplicity