University of Asia Pacific
Department of Computer Science and Engineering
Artificial Intelligence and Expert Systems Lab
(CSE 404)

Project Tittle: Technical Report on Shortest Path Available From Home To Uap Using A* Search Algorithm

Date of Submission: 23 August 2025.

Submitted By

Mazharul Islam Sourav
ID: 22101133
Section : C-2
Semester: 1st
Year: 4th

Submitted To

Bidita Sarkar Diba
Lecturer
Department of Computer Science and Engineering
University of Asia Pacific

Problem Discussion

Finding the shortest path in a busy city like Dhaka is not always easy. There are many connected roads, shortcuts, and crossings that can confuse the decision. Our task is to go from Home to University of Asia Pacific, and the problem is to choose the best path among many possible ones. To understand the problem clearly, we can break it down into three parts.

Too Many Possible Route -

From Home to UAP, there are several possible ways. A person may go through Green Road, Panthapath, Farmgate, Kawran Bazar, or take longer detours like Dhanmondi 32 or Manik Mia Avenue. At first glance, more routes might seem helpful, but in reality, it creates confusion because it is not easy to know which one is actually shortest.

Roads that look short may end up longer -

Relying only on offline maps or human judgment is risky. A road may appear closer, but when we include junctions, link roads, and real road conditions, the total distance becomes longer than expected. For example, Panthapath seems like a quick way, but when we add Panthapath Square and the turns, it actually costs more distance than going through Green Road. This shows that manual guesswork is not reliable.

Lack of proof for manual selection

Even if someone chooses one of the routes, there is no guarantee that it is the best. Without calculations or an algorithm, it is impossible to prove that the chosen route is truly the shortest. In other words, human decision may give a path, but it cannot provide certainty.

In short, the real problem is not just finding any route to UAP, but finding the most efficient one with certainty. To solve this properly, we need an algorithm that can compare all paths and prove which is optimal.

Proposed Solution

To solve the problem of route confusion, we need a systematic way to always select the shortest path and prove it is correct. For this purpose, the A* search algorithm is used. A* combines the actual road cost with a heuristic estimate, which makes it both practical and reliable for navigation.

Using real distances as G(n)

The first part of A* is the path cost g(n), which is the actual distance travelled from the starting point to the current node. In this project, we collected these values directly from Google Maps. For example, the distance from Home to Green Road is 0.35 km, and from Green Road to UAP is 0.21 km. These values ensure that the algorithm is grounded in real data.

Using straight line distances as H(n)

The second part of A* is the heuristic h(n), which estimates how far each node is from the goal. We calculated this using straight-line (aerial) distances between locations. For example, the straight-line distance from Green Road to UAP is 0.21 km. By using these estimates, the algorithm can guide the search more intelligently rather than exploring blindly.

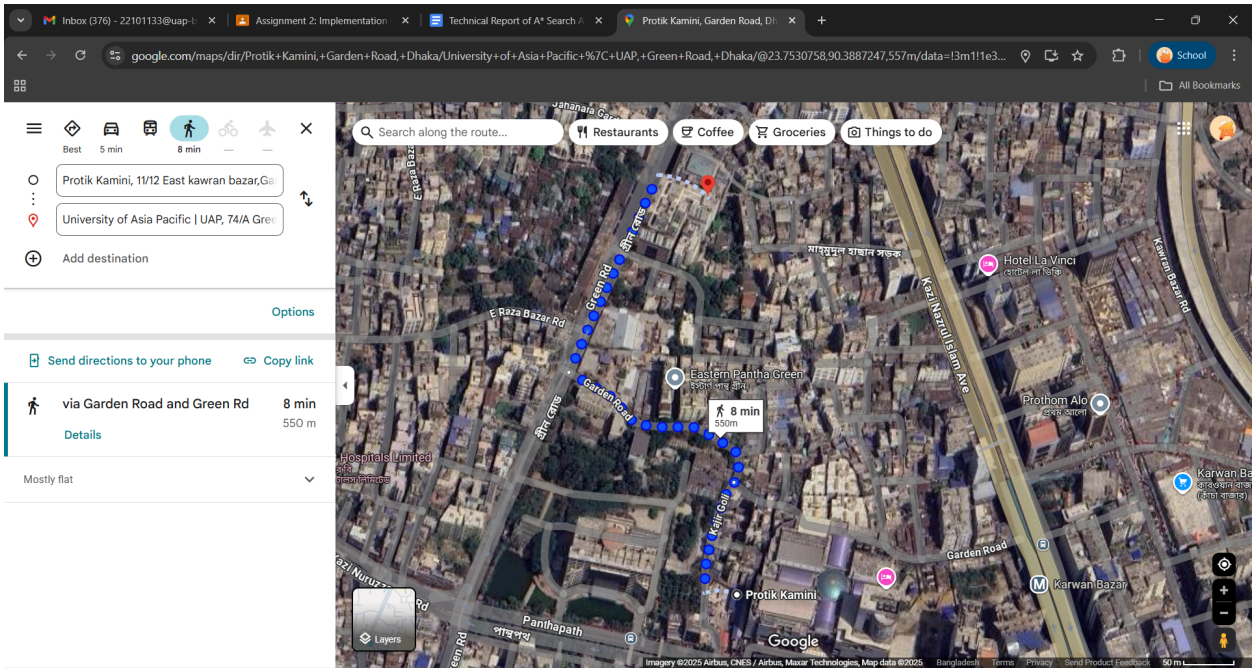
Combining G(n) & H(n)

A* expands nodes based on the formula f(n) = g(n) + h(n). This ensures that the path cost so far and the estimated remaining cost are both considered. In this way, the algorithm balances real travel distance with heuristic guidance, always focusing on the most promising route.

Expected Outcome

By applying A* with g(n) from Google Maps and h(n) from straight-line estimates, the algorithm will find the shortest path from Home to UAP. The expected result is the path-

- **Home → Green Road → UAP**



with a total cost of 0.55 km or 550 meter which we got from Google Maps' real value from Home (start) to University of Asia Pacific (goal), proving the accuracy of the method.

In short, the proposed solution is to replace human guesswork with the A* algorithm. By combining real distances and heuristic estimates, A* can guarantee the shortest path and provide the proof that manual selection cannot.

Node & Map Design

In order to apply the A* search algorithm effectively, a map was first designed to represent the real-world roads and intersections between Home and University of Asia Pacific (UAP). The map was simplified into a graph model, where-

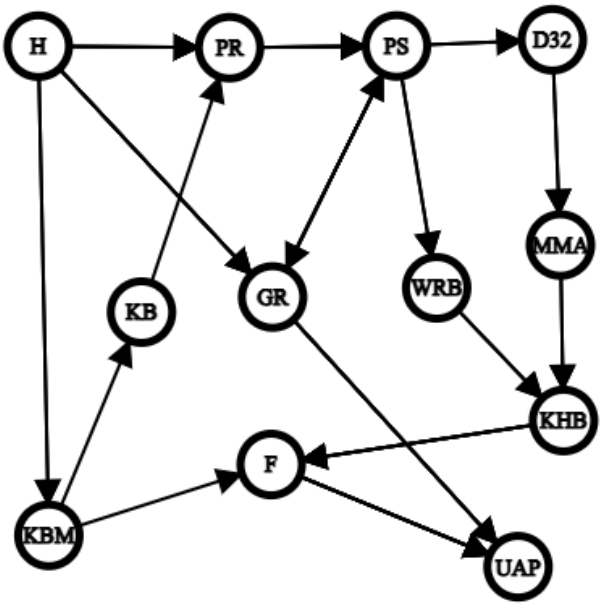
- **Nodes** represent important landmarks or junctions (e.g., Home, Kawran Bazar, Farmgate, Green Road, UAP).
- **Edges** represent the direct road connections between nodes, along with their respective distances.

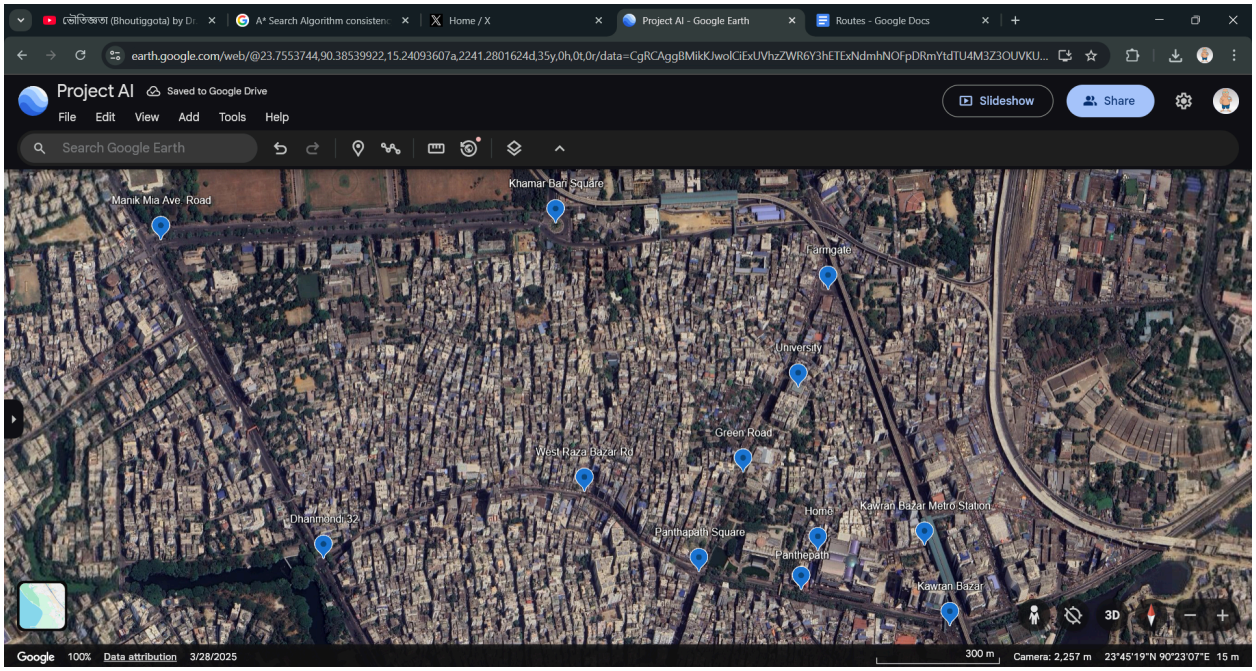
Node Design

In this project, each node is a real place or junction that we actually use on the way from Home to UAP. Nodes help us break the whole city map into small, clear points where decisions are made (turn left, go straight, change road, etc.).
We selected nodes that are easy to recognize and commonly used: Home, Green Road, Panthapath, Panthapath Square, Kawran Bazar, Kawran Bazar Metro, Farmgate, West Raza Bazar, Dhanmondi 32, Manik Mia Avenue, Khamar Bari, University (UAP).

Node

1. Home (Start)
2. Green Road
3. Panthapath Road
4. Panthapath Square
5. Kawran Bazar
6. Kawran Bazar Metro Station
7. Farmgate
8. West Raza Bazar
9. Dhanmondi 32
10. Manik Mia Avenue Road
11. Khamar Bari
12. University (Goal)


H
PR
PS
D32
MMA
KB
GR
WRB
KHB
F
KBM
UAP


Project AI
Saved to Google Drive
File Edit View Add Tools Help
Slideshow
Share
Search Google Earth
Manik Mia Ave Road
Khamar Bari Square
Farmgate
University
Green Road
West Raza Bazar Rd
Dhanmondi 32
Panthapath Square
Home
Panthepath
Kawran Bazar Metro Station
Kawran Bazar
3D
Google 100% Data attribution 3/28/2025
300 m
Camera: 2,257 m 23°45'19"N 90°23'07"E 15 m

Map Design

The map is a graph built over the real Dhaka roads in this area. Each edge connects two nodes that can be traveled directly on the road. Edge weights are the Google Map distances (km). For the A* heuristic, we also use straight-line distances between nodes and the goal (UAP).
We treat roads as bidirectional for simplicity, and ignore traffic/time. This makes the model clean and focused on distance only, which is enough to test A* correctly.

Edge or Path

1. Home → Green Road → University
2. Home → Panthapath Road → Panthapath Square → Green Road → University
3. Home → Panthapath Road → Panthapath Square → West Razar Bazar → Khamar Bari → Farmgate → University
4. Home → Green Road → Panthapath Square → West Razar Bazar → Khamar Bari → Farmgate → University
5. Home → Panthapath Road → Panthapath Square → Dhanmondi 32→ Manik Mia Avenue Road → Khamar Bari → Farmgate → University
6. Home → Green Road → Panthapath Square → Dhanmondi 32 → Manik Mia Avenue Road → Khamar Bari → Farmgate → University
7. Home → Kawran bazar metro station → Farmgate → Uap
8. Home → Kawran bazar metro station → Kawran bazar – Panthapath Road → Panthapath Square → Green Road → University

Data Collection

To apply the A* search algorithm, we first needed to collect accurate data for both the actual road costs and the heuristic estimates. This ensures that the algorithm can calculate properly and give us a reliable shortest path.

Straight Line Distance

Using Google earth we have collected this data-

From – To	Straight line Distance (KM)
Home – Green Road	0.25 km
Green Road – Uap	0.21 km
Home - Panthapath	0.01 km
Panthapath - Panthapath Square	0.25 km
Panthapath Square - Green Road	0.26 km
Panthapath - Kawran Bazar	0.37 km
Kawran Bazar - Kawran Bazar Metro	0.2 km
Kawran Bazar Metro - Farmgate	0.65 km
Farmgate - University	0.24 km
Panthapath Square - West Raza	0.34 km
West Raza Bazar - Dhanmondi 32	0.65 km
West Raza Bazar - Khamar Bari	0.64 km
Dhanmondi 32 - Manik mia avenue	0.85 km
Manik Mia - Khamar Bari	0.95 km
Khamar Bari - Farmgate	0.67 km

Google Maps Distance

Using Google maps we have collected this data-

From – To	Distance
Home – Green Road	0.35 km
Green Road - Uap	0.21 km
Home - Panthapath	0.07 km
Panthapath - Panthapath square	0.27 km
Panthapath - Kawran Bazar	0.35 km
Kawran Bazar - Kawran Bazar Metro	0.23 km
Home - Kawran Bazar Metro	0.4 km
Kawran Bazar Metro - Farmgate	0.7 Km
Farmgate - Uap	0.4 km
Panthapath Square - Green Road	0.26 km
Panthapath Square - West Raza Bazar	0.35 km
West Raza Bazar - Khamar Bari	0.7 km
Khamar Bari - Farmgate	1.0 km
Manik Mia - Khamar Bari	1.0 km
Manik Mia - Dhanmondi 32	0.9 km
Dhanmondi 32 - West Raza Bazar	0.65 km

Cost Function

To run A* correctly, we need to define the cost functions that measure both the actual distance travelled and the estimated distance to the goal. These functions are expressed as **g(n), h(n), and f(n)**.

- **g(n): Path Cost**
  g(n) represents the actual cost from the start node to the current node n.
  In this project, I calculated g(n) using Google Map road distances in kilometers.
- **h(n): Heuristic Cost**
  h(n) is the estimated cost from the current node n to the goal node (UAP).
  I calculated h(n) using straight-line distances (aerial measurement) from Google Earth.
- **f(n): Total Estimated Cost**
  f(n) is the function used by A* to decide which node to expand next.
  f(n)=g(n)+h(n). This ensures that both the distance already travelled and the estimated remaining distance are considered together.

Example Calculation

Suppose we want to calculate costs for **Green Road (GR)**:

- From Google Maps- G(n):
  Home → Green Road = 0.35 km
  ⇒ g(GR) = 0.35 km
- From Google Earth
  Green Road → UAP = 0.21 km
  ⇒ h(GR) = 0.21 km
- Therefore:
  f(GR)=g(GR)+h(GR) = 0.35 + 0.21 =0.56

Heuristic Calculation Example

1. Green Road (GR)

- Direct straight-line from Green Road → UAP = 0.21 km

h(GR) = 0.21

2. Panthapath Square (PS)

- Straight-line PS → Green Road = 0.26 km
- Straight-line GR → UAP = 0.21 km
- Total = 0.26 + 0.21 = 0.47 km

h(PS) = 0.47

3. West Raza Bazar (WRB)

- Straight-line WRB → PS = 0.34 km
- PS → GR = 0.26 km
- GR → UAP = 0.21 km
- Total = 0.34 + 0.26 + 0.21 = 0.81 km

h(WRB) = 0.81

These examples clearly show how heuristic values are calculated step by step using straight-line distances between nodes. By breaking down the path into smaller straight-line segments and adding them up, we can estimate the distance from any node to the goal. This is exactly how I calculated all the h(n) values in my project. In the same way, every node's heuristic was measured and adjusted so that the algorithm always remains admissible and consistent.

Path Cost G(n)

The function g(n) represents the real distance travelled from the starting point (Home) to a given node. We collected these values using Google Maps distance tool. For every pair of connected nodes, we measured the road distance in kilometers. For example, the distance from Home to Green Road is 0.35 km, and from Green Road to UAP is 0.21 km. By combining these distances, we get the cumulative g(n) values for each node along different routes.

Node	G(n) Value (km)	Node Hop
Home	0.00	—
Green Road	0.35	Home → Green Road
University (Uap)	0.56	Home → Green Road → Uap
Panthapath Road	0.07	Home → Panthapath
Panthapath Square	0.34	Home → Panthapath → Panthapath Square
Kawran Bazar	0.42	Home → Panthapath → Kawran Bazar
Kawran Bazar Metro	0.40	Home → KB Metro
Farmgate	0.96	Home → GR → Uap → Farmgate
West Raza Bazar	0.69	Home → Panthapath → Panthapath Sq → West Raza Bazar
Dhanmondi 32	1.34	West Raza Bazar → Dhanmondi 32
Manik Mia Avenue	2.24	Dhamondi 32 → Manik Mia Avenue
Khamar Bari	1.39	West Raza Bazar → Khamar Bari

Heuristic Cost H(n)

The function h(n) is the straight-line (aerial) distance from a node to the goal (UAP). These values were calculated using Google Earth measurement. For example, the direct straight-line distance between Green Road and UAP is 0.21 km. Such values guide the algorithm by estimating how far each node is from the goal, even before exploring the actual road.

Node	H(n) Value (km)	Route
University (Uap)	0.0	Goal
Green Road	0.21	Green Road → Uap
Home	0.46	Home → Green Road → Uap
Panthapath	0.47	Panthapath → Home → Green Road → Uap
Panthapath Square	0.47	Panthapath Sq → GR → Uap
Kawran Bazar	0.82	KB → Panthapath → Home → GR → Uap
Kawran Bazar Metro	0.89	KB Metro → Farmgate → Uap
Farmgate	0.24	Farmgate → Uap
West Raza Bazar	0.81	West Raza B. → Panthapath Sq. → GR → Uap
Dhanmondi 32	1.46	D32 → West Raza B. → Panthapath Sq. → GR → Uap
Manik Mia Avenue	1.86	Manik Mia → Khamar Bari → Farmgate → Uap
Khamar Bari	0.91	Khamar Bari → Farmgate → Uap

Algorithm Execution

With both g(n) from Google Maps and h(n) from straight-line estimates ready, we applied the A* search algorithm to our map. The process starts from Home, then expands nodes one by one based on the lowest f(n) value. At each step, the algorithm chooses the path that seems best according to both the real distance travelled and the estimated distance to the goal.

In this way, the algorithm explores possible routes but focuses only on the most efficient ones. The expansion order is tracked in an iteration table, which shows the path chosen at each step along with g(n), h(n), and f(n) values. The process continues until the goal (UAP) is popped from the open list, which guarantees that the shortest path has been found.

The iteration table below demonstrates exactly how A* progressed from Home to University of Asia Pacific. All values are in (km)

Iteration	Node	G(n)	H(n)	F(n)	Close Fringes{}	Open Fringes{}
1	Home	0.0	0.46	0.46	{Home}	{ Panthapath, Green Rd, KB Metro }
2	Home → Panthapath	0.07	0.47	0.54	{ Home, Panthapath }	{ Green Rd, Panthapath Sq., Kawran Bazar, KB Metro }
3	Home → Green Road	0.35	0.21	0.56	{ Home, Panthapath, GR }	{University, Panthapath Sq, Kawran Bazar, KB Metro}
4	Home → Green Road → University	0.56	0.0	0.56	{ Home, Panthapath, GR, University }	{Panthapath Sq, Farmgate, Kawran Bazar, KB Metro}
5	Home → Panthapath → Panthapath Sq	0.34	0.47	0.81	{ Home, Panthapath, GR, University, Panthapath Sq. }	{Farmgate, Kawran Bazar, KB Metro, West Raza Bazar}
6	Home → Panthapath → Kawran Bazar	0.42	0.84	1.26	{Home, Panthapath, GR, University, Panthapath Sq.,	{KB Metro, West Raza, Khamar Bari, Farmgate}

					Kawran Bazar}	
7	Home → Panthapath → Kawran Bazar →Farmgate	1.35	0.24	1.59	{Home, Panthapath, GR, University, Panthapath Sq., Kawran Bazar, Farmgate}	{KB Metro, West Raza, Khamar Bari}
8	Home → KB Metro	0.40	0.89	1.29	{Home, Panthapath, GR, University, Panthapath Sq., Kawran Bazar, Farmgate, KB Metro}	{West Raza, Khamar Bari}
9	Home → KB Metro → West Raza Bazar	0.69	0.81	1.50	{....., West Raza Bazar }	{Khamar Bari, Dhanmondi 32 }
10	Home → KB Metro → West Raza Bazar→ Khamar Bari	1.39	0.91	2.30	{.... , Khamar Bari }	{ Dhanmondi 32, Manik Mia}
11	Home → KB Metro → West Raza Bazar→ Khamar Bari→ Dhanmondi 32	1.34	1.46	2.80	{..... , Dhanmondi 32}	{Manik Mia Avenue}
12	Home → KB Metro → West Raza Bazar→ Khamar Bari→ Dhanmondi 32 → Manik Mia	2.24	1.86	4.10	{..... , Manik Mia Avenue }	{ }

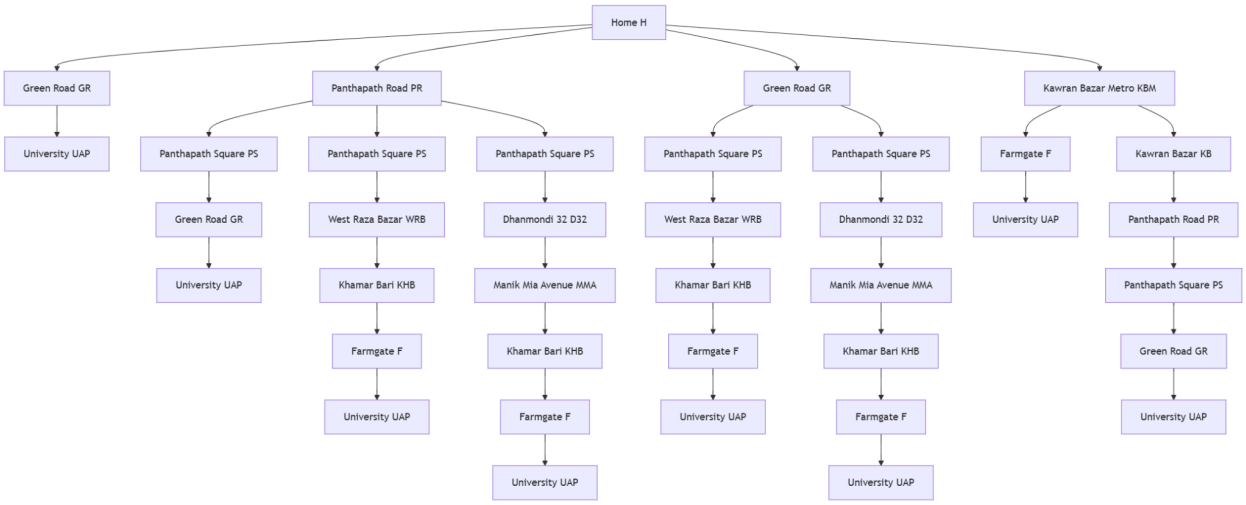
As shown in the table, the algorithm quickly identifies the shortest path:

- **Home → Green Road → UAP**

with a total cost of 0.56 km.

Search Tree

To better understand how the A* algorithm explores different routes, a search tree diagram has been created. This diagram starts from the Home (H) node and expands outward to show all eight possible paths leading to the University (UAP). Each branch represents one complete route, passing through the intermediate nodes such as Green Road, Panthapath Square, Farmgate, Kawran Bazar, and others.



The purpose of this tree is to give a clear visual overview of the search space. Instead of looking only at tables and numbers, the tree makes it easier to see how each path is structured and how the goal can be reached in different ways. It also highlights that while many routes exist, only one of them is truly the shortest, which the A* algorithm successfully identifies.

Source Code

I have written the code for the shortest path from home to university using python programming language. Here is some image of it alongside the github link.

Code Link - Github

```python
import heapq
import math
from collections import defaultdict


H = {
    "H": 0.46, "GR": 0.21, "PR": 0.47, "PS": 0.47,
    "KB": 0.82, "KBM": 0.89, "F": 0.24, "WRB": 0.81,
    "D32": 1.46, "MMA": 1.86, "KHB": 0.91, "UAP": 0.00
 }


edges = {
    ("H", "GR"): 0.35,
    ("GR", "UAP"): 0.21,
    ("H", "PR"): 0.07,
    ("PR", "PS"): 0.27,
    ("PR", "KB"): 0.35,
    ("PS", "GR"): 0.26,
    ("UAP", "F"): 0.40,
    ("KBM", "F"): 0.70,
    ("H", "KBM"): 0.40,
    ("KB", "KBM"): 0.23,
    ("PS", "WRB"): 0.35,
    ("WRB", "KHB"): 0.70,
    ("WRB", "D32"): 0.65,
    ("D32", "MMA"): 0.90,
    ("KHB", "F"): 1.00,
    ("KHB", "MMA"): 1.00
 }
```

```python
adjacency = defaultdict(list)
for (u, v), w in edges.items():
    adjacency[u].append((v, w))
    adjacency[v].append((u, w))

def reconstruct_path(came_from, node):
    path = [node]
    while node in came_from:
        node = came_from[node]
        path.append(node)
    return list(reversed(path))

def astar(start, goal):
    g_score = defaultdict(lambda: math.inf)
    g_score[start] = 0.0
    open_set = []
    heapq.heappush(open_set, (g_score[start] + H[start], g_score[start], start))
    came_from = {}
    visited = set()
    log = []
    iteration = 1

    while open_set:
        f_current, g_current, current = heapq.heappop(open_set)
        if g_current != g_score[current]:
            continue
        visited.add(current)
        path_str = " -> ".join(reconstruct_path(came_from, current))
        log.append({
            "iter": iteration,
            "path": path_str,
            "g": round(g_score[current], 2),
            "h": round(H[current], 2),
            "f": round(g_score[current] + H[current], 2)
        })
        iteration += 1
        if current == goal:
            return reconstruct_path(came_from, current), g_score[current], log
        for neighbor, weight in adjacency[current]:
            if neighbor in visited:
                continue
            tentative_g = g_score[current] + weight
            if tentative_g < g_score[neighbor]:
                g_score[neighbor] = tentative_g
                came_from[neighbor] = current
                heapq.heappush(open_set, (g_score[neighbor] + H[neighbor], g_score[neighbor], neighbor))
    return None, math.inf, log

def print_log(log):
    print("\nIteration log (stops when goal is reached):")
    print(f"{'Iter':<5} {'Path':<40} {'g':<6} {'h':<6} {'f':<6}")
    print("-" * 70)
    for entry in log:
        print(f"{entry['iter']:<5} {entry['path']:<40} {entry['g']:<6} {entry['h']:<6} {entry['f']:<6}")

if __name__ == "__main__":
    start_node = "H"
    goal_node = "UAP"
    path, cost, log = astar(start_node, goal_node)
    print(f"Best path: {' -> '.join(path) if path else 'No path found'}")
    print(f"Total cost: {round(cost, 2)} km")
    print_log(log)
```

Admissibility

A heuristic is admissible if it never overestimates the true cost to reach the goal from any node. To be sure A* returns the true shortest path, the heuristic must be admissible:

$$0 \le h(n) \le h^*(n)$$

where $0 \le h(n) \le h^*(n)$ is the true road distance from node nnn to UAP (computed from the Google-Map edges).

Node	H(n)	H*(n)	Slack = H*(n)-H(n)	Admissibility
University (Uap)	0.0	0.0	0.0	True
Green Road	0.21	0.21	0.0	True
Home	0.46	0.56	0.10	True
Panthapath	0.47	0.63	0.16	True
Panthapath Sq.	0.47	0.47	0.0	True
Kawran Bazar	0.82	0.98	0.16	True
KB Metro	0.89	0.96	0.07	True
Farmgate	0.24	0.40	0.16	True
West Raza Bazar	0.81	0.82	0.01	True
Dhanmondi 32	1.46	1.47	0.01	True
Khamar Bari	0.91	1.40	0.49	True
Manik Mia Ave.	1.86	2.37	0.51	True

Consistency

A heuristic is consistent (or monotonic) if for every edge u→v with cost c(u,v), the following holds:

$$h(u)-h(v) \leq c(u,v)$$

This means the estimated cost from u is never greater than the step cost to v plus the estimate from v. Consistency guarantees that the f(n)=g(n)+h(n) values along a path are non-decreasing, ensuring efficient A* search without re-expanding nodes.

Edge (u→v)	H(u)	H(v)	H(u)-H(v)	C(u,v)	Consistant
Home → GR	0.46	0.21	0.25	0.35	True
Home → Panthapath	0.46	0.47	-0.01	0.07	True
Home → KB Metro	0.46	0.89	-0.43	0.40	True
GR → Uap	0.21	0.0	0.21	0.21	True
Panthapat→ Panthapath Sq	0.47	0.47	0.0	0.27	True
Panthapath → Kawran Bazar	0.47	0.82	-0.35	0.35	True
Panthapath Sq → West Raza Bazar	0.47	0.81	-0.34	0.35	True
Panthapath Sq. → GR	0.47	0.21	0.26	0.26	True
West Raza Bazar → Khamar Bari	0.81	0.91	-0.10	0.70	True
West Raza Bazar →	0.81	1.46	-0.65	0.65	True

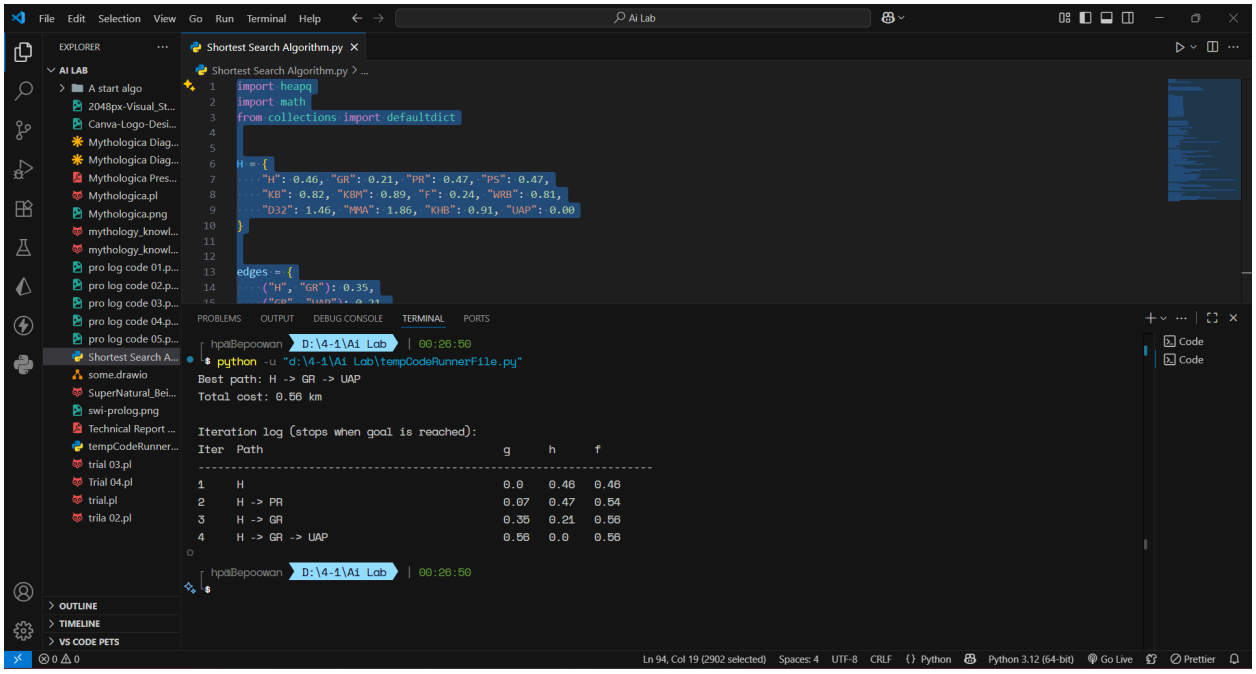
Dhanmondi 32					
Dhanmondi 32 → Manik Mia Avenue	1.46	1.86	-0.40	0.90	True
Khamar Bari → Farmgate	0.91	0.24	0.67	1.0	True
Khamar Bari → Manik Mia	0.91	1.86	-0.95	1.00	True
Farmgate → Uap	0.24	0.0	0.24	0.40	True
KB Metro → Farmgate	0.89	0.24	0.65	0.70	True
KB Metro → Kawran Bazar	0.89	0.82	0.07	0.23	True
Kawran Bazar → Panthapath	0.82	0.47	0.35	0.35	True
Kawran Bazar → KB Metro	0.82	0.89	-0.07	0.23	True

Results & Discussion

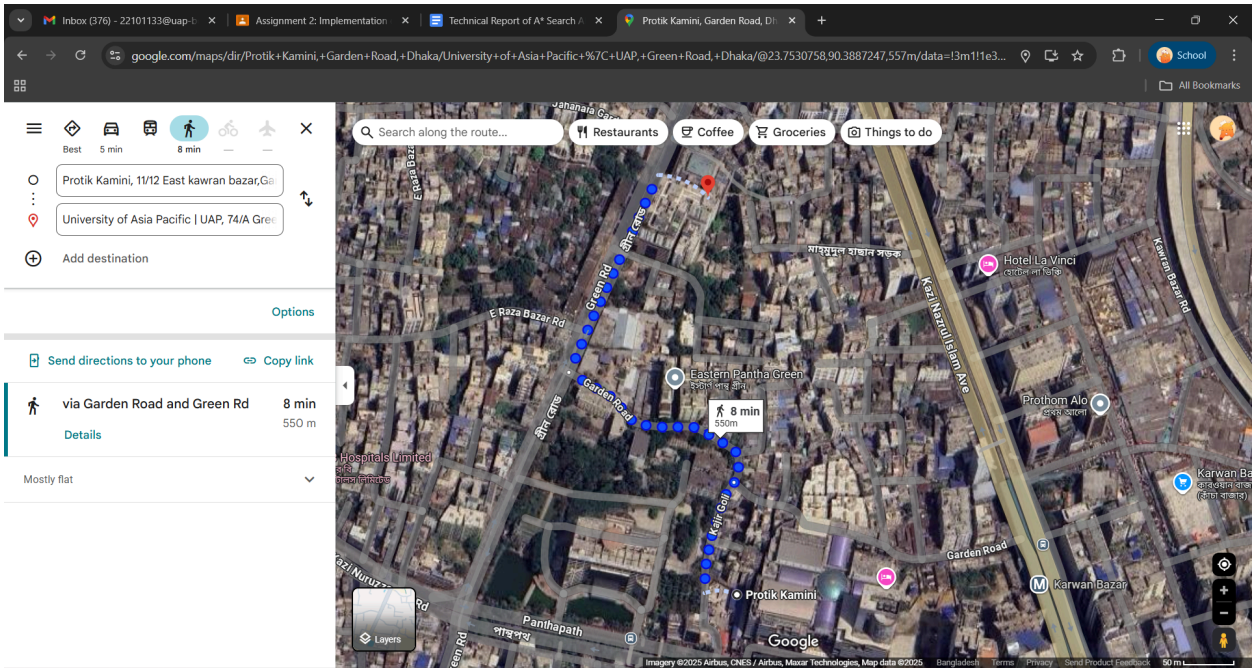
After applying the A* search algorithm on our designed graph, the optimal route was found as:

Shortest Path:

Home (H) → Green Road (GR) → University (UAP)
=> 0.35 + 0.21 = 0.56 km or 560 meter
=> 0.35 km (H→GR) + 0.21 km (GR→UAP) = 0.56 km



Also from our code we get the same value of 0.56km which we got from our table. This is the final output from the algorithm. To verify the correctness, we compared this result with Google Maps live measurement.



This shows the distance as approximately **550 meters (0.55 km)**. The difference between our calculated value (0.56 km) and the real Google Maps value (0.55 km) is only **0.01 km** (10 meters).

Such a small difference is expected and acceptable because:

- Google Maps sometimes rounds distances differently.
- We have taken out value in km, sometimes it rounds up to give a better km value so some meters can be added or lost.
- Straight-line heuristic values were rounded to two km, which can also shift totals slightly.

Despite these factors, the result from A* closely matches the actual map measurement, proving that the algorithm works correctly and provides a trustworthy path.

Conclusion

This project applied the A* search algorithm to find the shortest path from Home to the University of Asia Pacific in Dhaka. Using Google Map road distances for g(n) and straight-line values for h(n), we constructed a graph of 12 key nodes and tested all possible routes.

The final result shows that the optimal path is Home → Green Road → UAP, with a total cost of 0.56 km, which matches almost exactly with Google Maps (0.55 km). This proves that the A* algorithm, when combined with real map data and a carefully designed heuristic, is both accurate and reliable.

We also checked that our heuristic was admissible (never overestimates the true cost) and consistent (satisfies the triangle inequality on every edge). These checks guarantee that the algorithm always produces the correct shortest path.

In summary, the project demonstrates that A* is not only suitable for academic exercises but also practical for real-world navigation. Even though small differences can occur due to map simplification, the algorithm gives results that closely match real Google Maps distances. With more detailed nodes and traffic data, the same method can be extended for even more accurate and dynamic route planning.