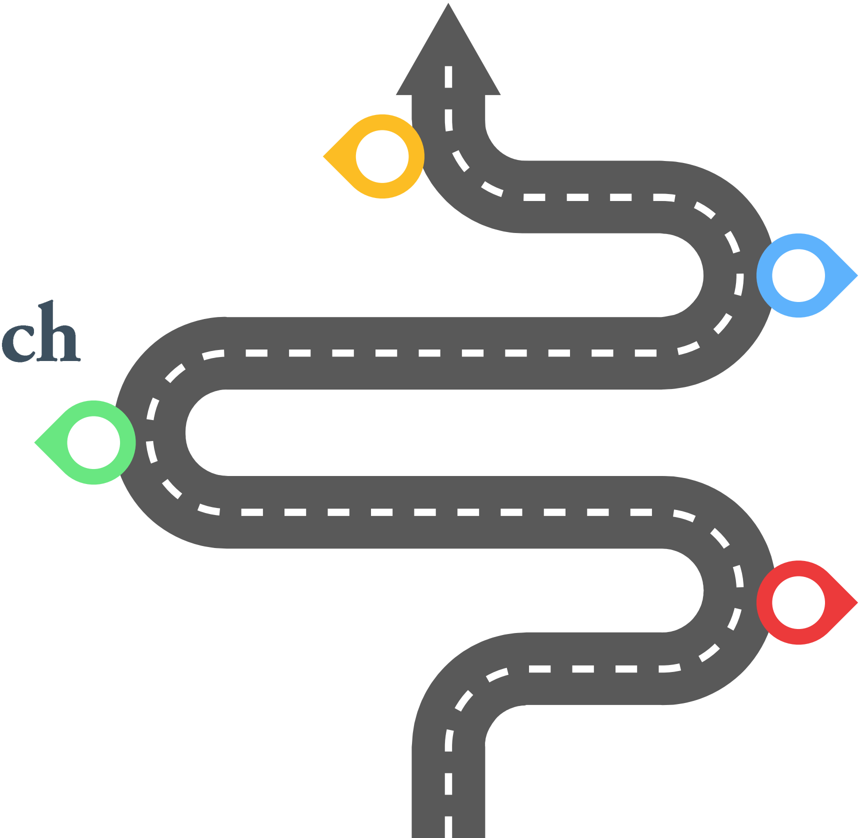
Artificial intelligence and Experts Systems Lab
CSE 404

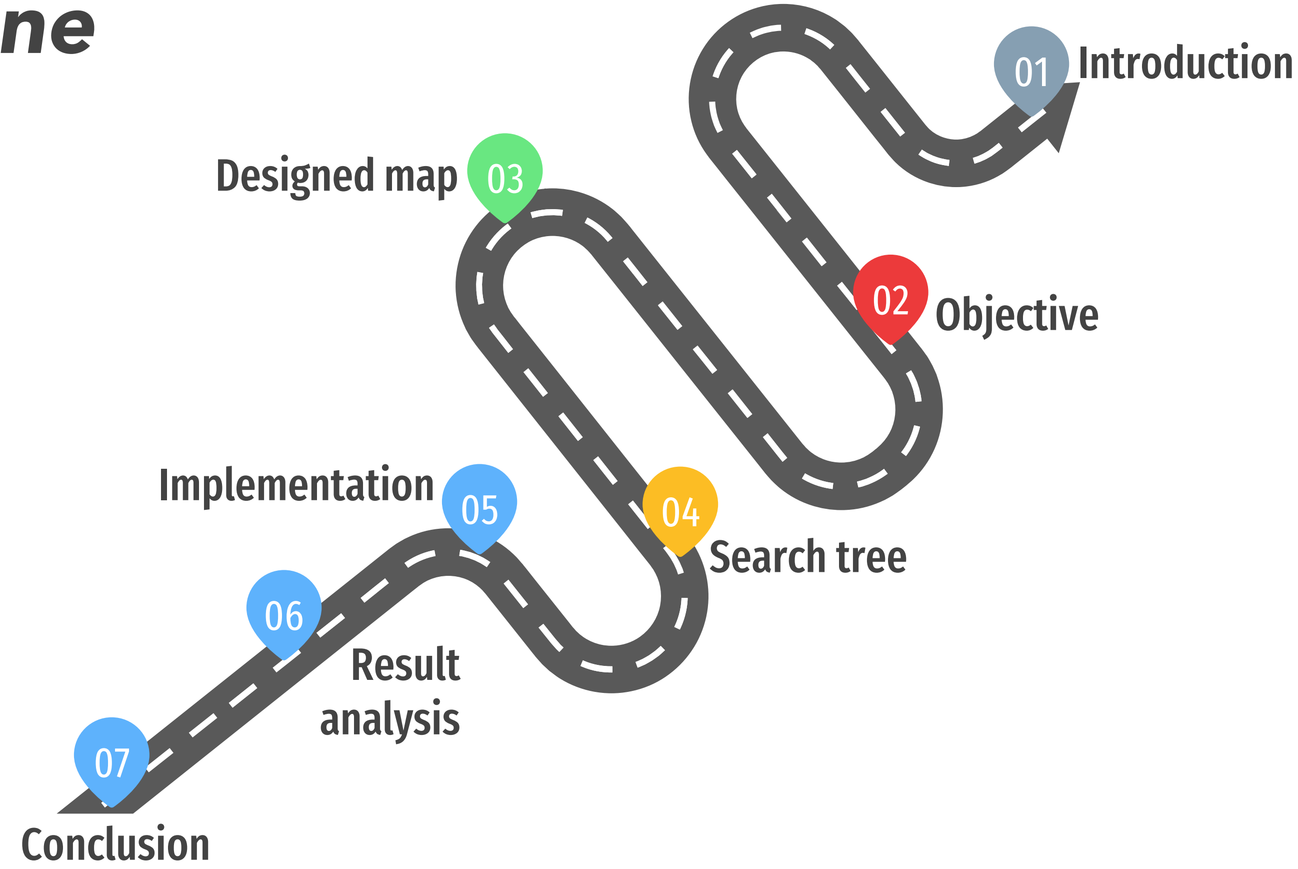
Project-1

Implementation of a small address Map using A* Search Algorithm



Submitted By
Rifa Tasfia
19201045(A2)

Outline



01 Introduction

02 Objective

03 Designed map

04 Search tree

05 Implementation

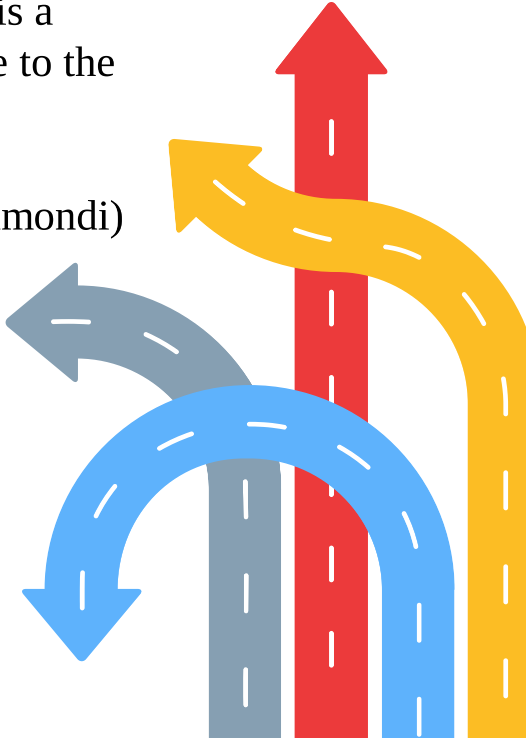
06 Result analysis

07 Conclusion

Introduction

The assigned problem is implementation of a small address map from my home to UAP, using A* search algorithm and find out the optimal path. A* algorithm is a searching algorithm that searches for the shortest path between the initial state to the final state.

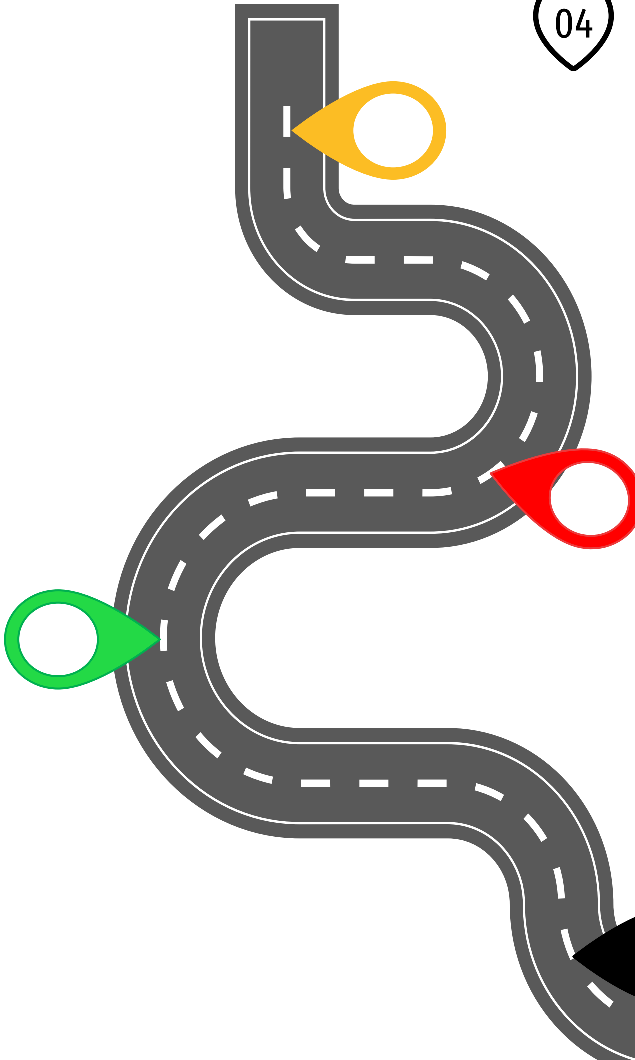
So, here in this project I will find the most optimal path from my home (Dhanmondi) to my university (UAP) using A* search algorithm.



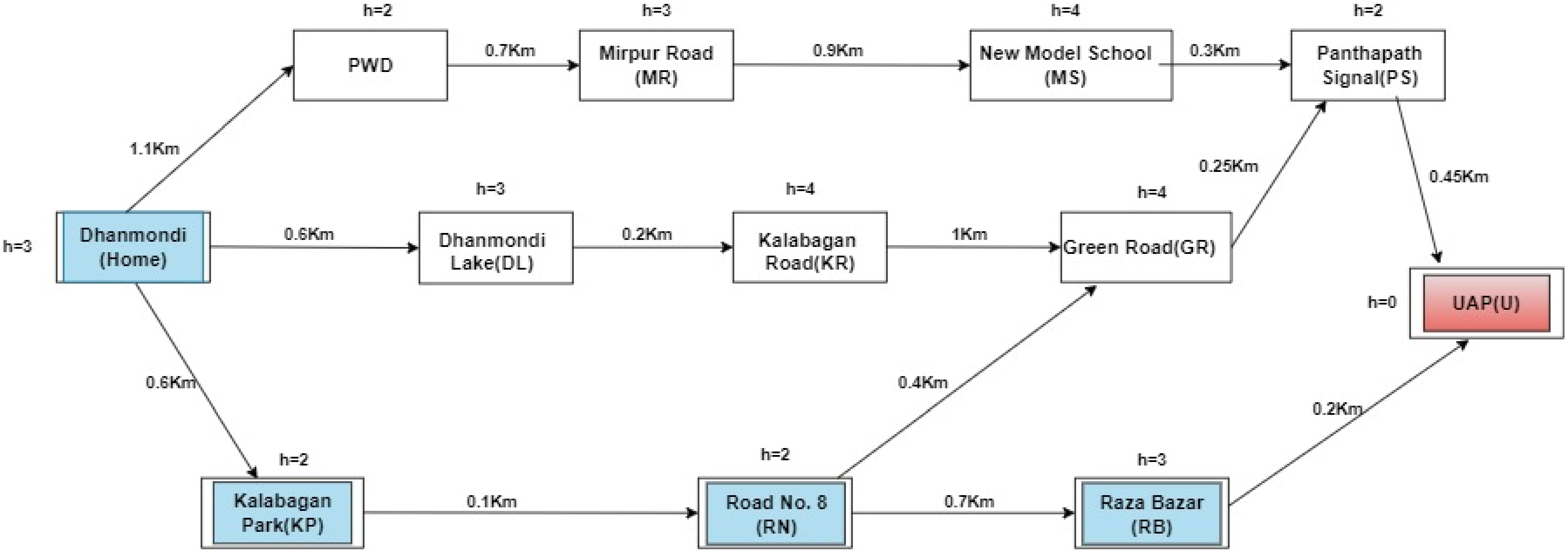
Objective

In this project, I have to reach UAP from my home Dhanmondi by using the shortest path.
There are several path between Dhanmondi to UAP. But not all of those path are optimal. So I need to find out the optimal path. For finding, I’ve used the A*(A-star) search algorithm.

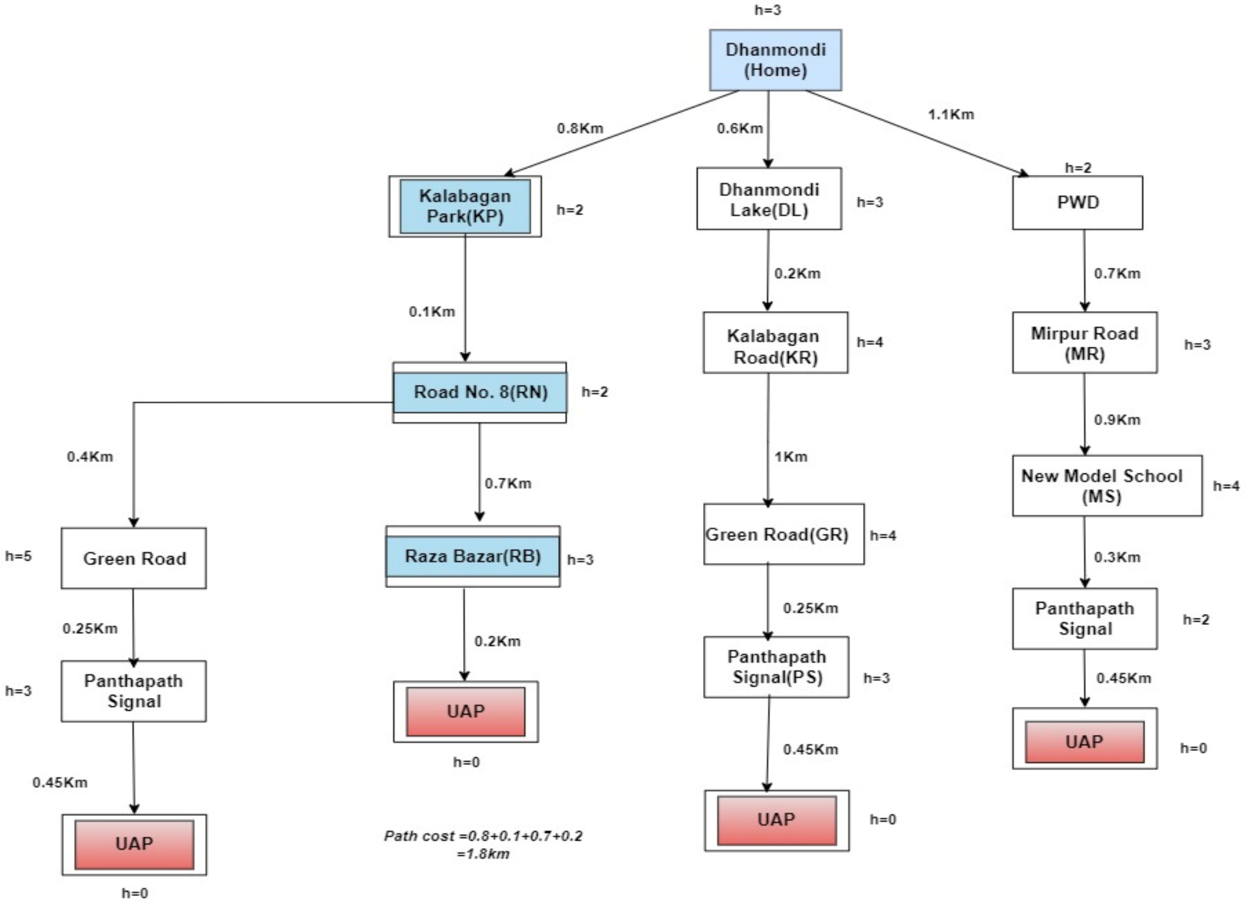
The objective of this project is to find an optimal path from my home (Dhanmondi) to my university (UAP).



Designed Map



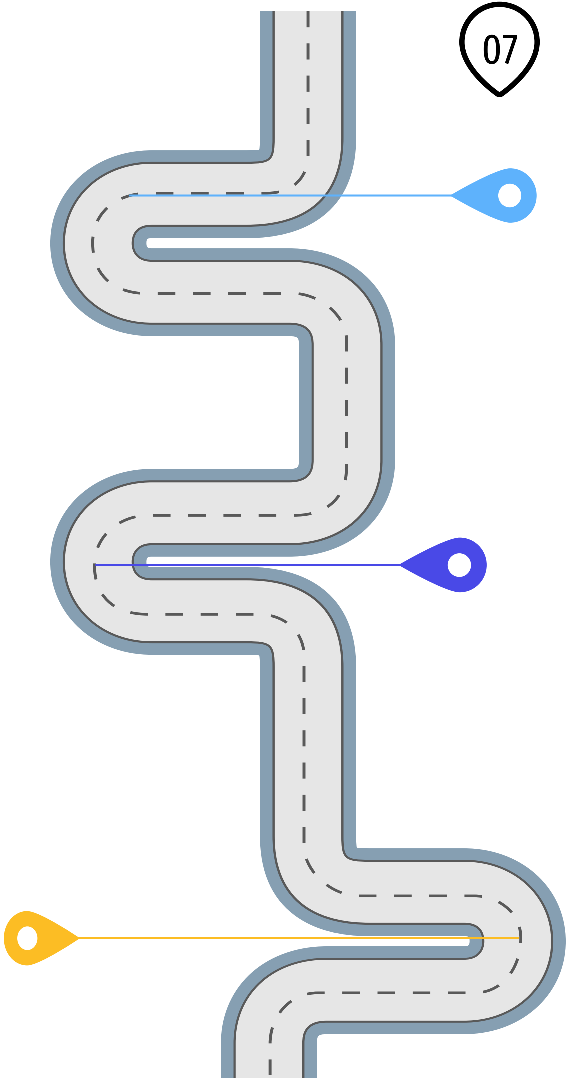
Search Tree


h=3
Dhanmondi (Home)
0.8Km
0.6Km
1.1Km
Kalabagan Park(KP)
h=2
Dhanmondi Lake(DL)
h=3
h=2
PWD
0.1Km
0.2Km
0.7Km
Kalabagan Road(KR)
h=4
Mirpur Road (MR)
h=3
Road No. 8(RN)
h=2
0.4Km
0.7Km
1Km
0.9Km
New Model School (MS)
h=4
h=5
Green Road
Raza Bazar(RB)
h=3
Green Road(GR)
h=4
0.3Km
0.25Km
0.2Km
0.25Km
Panthapath Signal
h=2
h=3
Panthapath Signal
Panthapath Signal(PS)
h=3
0.45Km
UAP
h=0
UAP
h=0
0.45Km
0.45Km
UAP
h=0
UAP
h=0
Path cost =0.8+0.1+0.7+0.2 =1.8km

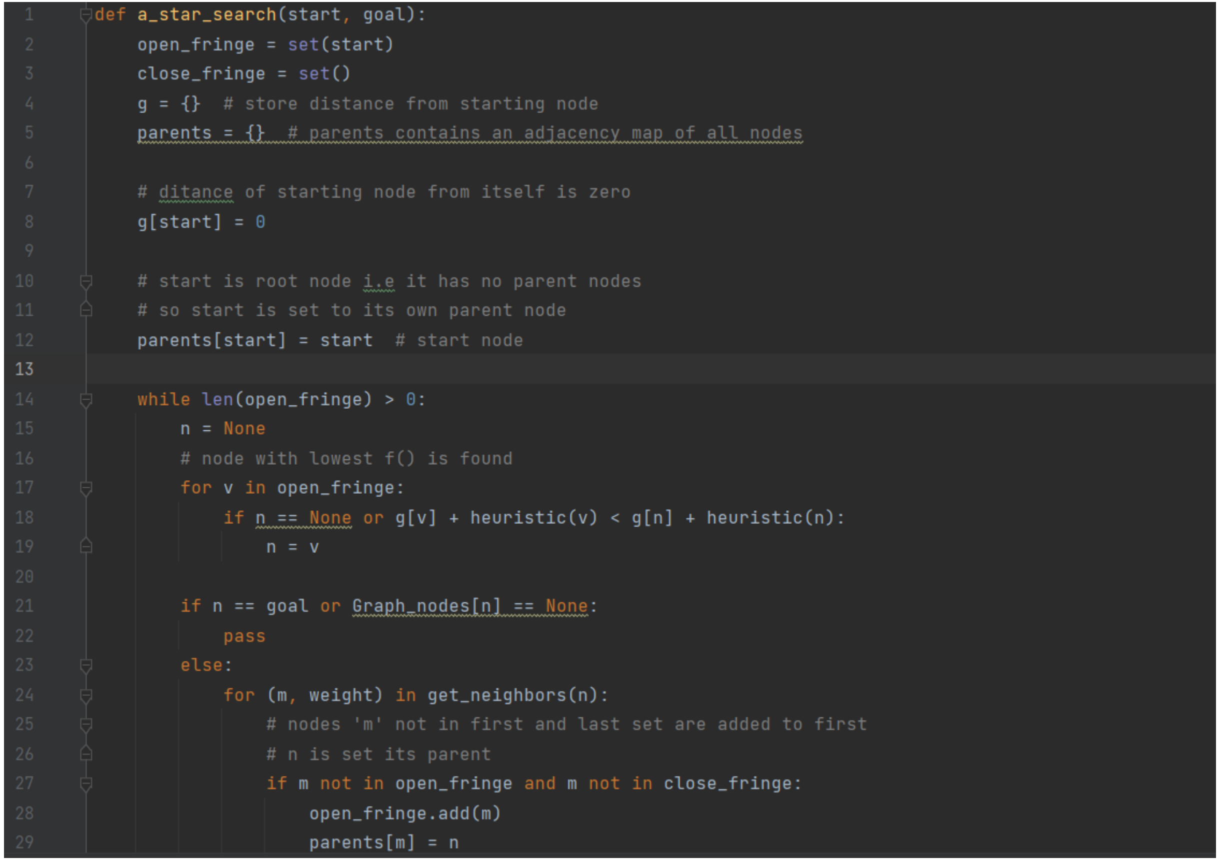
Implementation

Now I'll explain implementation part of this project.

I've used python for the programming language and implement it in PyCharm IDE.

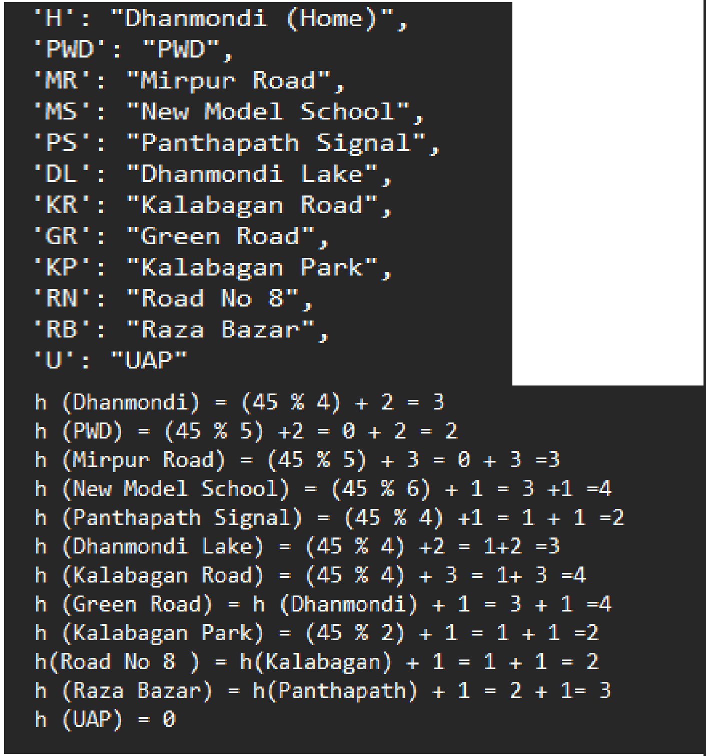


Implementation



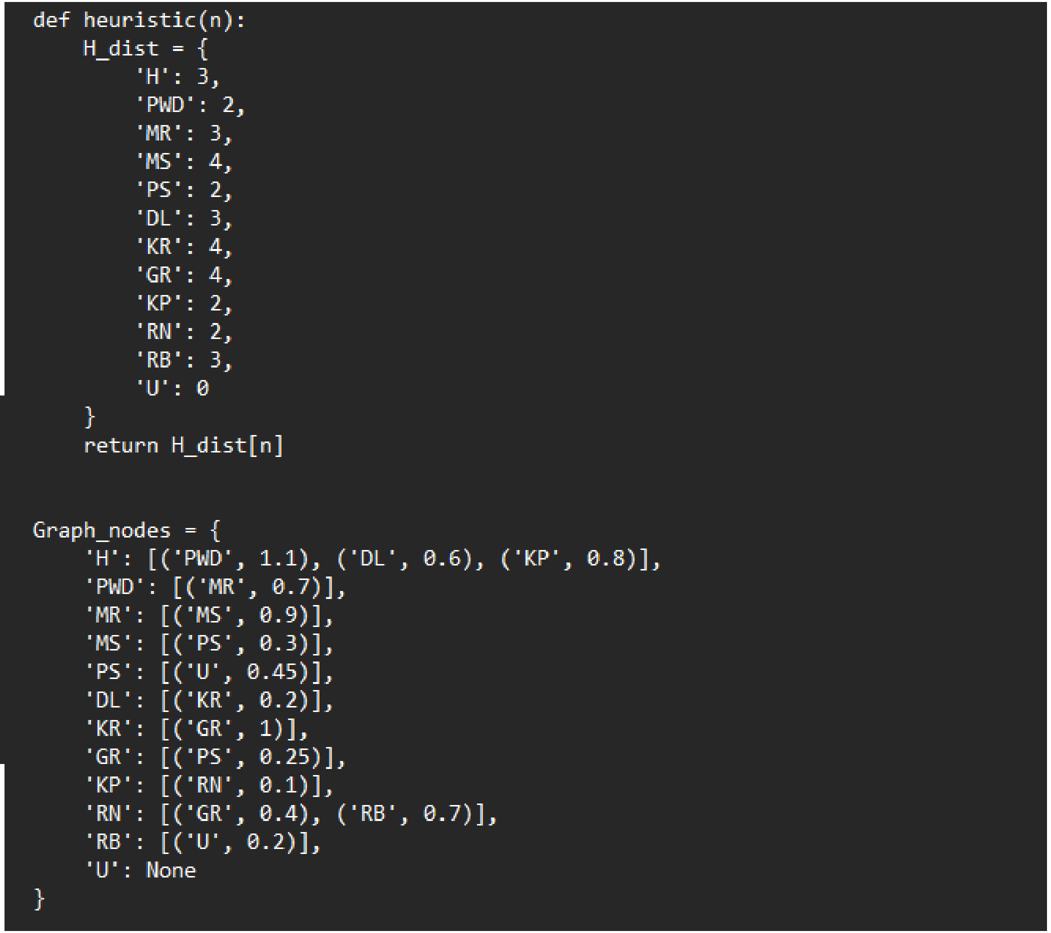
```python
def a_star_search(start, goal):
    open_fringe = set(start)
    close_fringe = set()
    g = {}  # store distance from starting node
    parents = {}  # parents contains an adjacency map of all nodes

    # ditance of starting node from itself is zero
    g[start] = 0

    # start is root node i.e it has no parent nodes
    # so start is set to its own parent node
    parents[start] = start  # start node

    while len(open_fringe) > 0:
        n = None
        # node with lowest f() is found
        for v in open_fringe:
            if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):
                n = v

        if n == goal or Graph_nodes[n] == None:
            pass
        else:
            for (m, weight) in get_neighbors(n):
                # nodes 'm' not in first and last set are added to first
                # n is set its parent
                if m not in open_fringe and m not in close_fringe:
                    open_fringe.add(m)
                    parents[m] = n
```

Input



```
'H': "Dhanmondi (Home)",
'PWD': "PWD",
'MR': "Mirpur Road",
'MS': "New Model School",
'PS': "Panthapath Signal",
'DL': "Dhanmondi Lake",
'KR': "Kalabagan Road",
'GR': "Green Road",
'KP': "Kalabagan Park",
'RN': "Road No 8",
'RB': "Raza Bazar",
'U': "UAP"
```

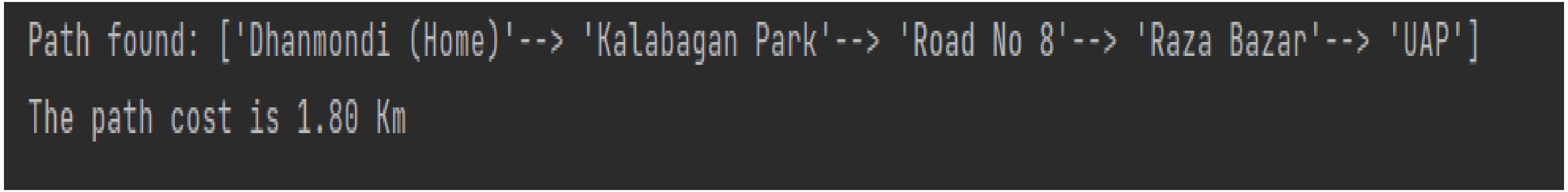
```
h (Dhanmondi) = (45 % 4) + 2 = 3
h (PWD) = (45 % 5) +2 = 0 + 2 = 2
h (Mirpur Road) = (45 % 5) + 3 = 0 + 3 =3
h (New Model School) = (45 % 6) + 1 = 3 +1 =4
h (Panthapath Signal) = (45 % 4) +1 = 1 + 1 =2
h (Dhanmondi Lake) = (45 % 4) +2 = 1+2 =3
h (Kalabagan Road) = (45 % 4) + 3 = 1+ 3 =4
h (Green Road) = h (Dhanmondi) + 1 = 3 + 1 =4
h (Kalabagan Park) = (45 % 2) + 1 = 1 + 1 =2
h(Road No 8 ) = h(Kalabagan) + 1 = 1 + 1 = 2
h (Raza Bazar) = h(Panthapath) + 1 = 2 + 1= 3
h (UAP) = 0
```



```
def heuristic(n):
    H_dist = {
        'H': 3,
        'PWD': 2,
        'MR': 3,
        'MS': 4,
        'PS': 2,
        'DL': 3,
        'KR': 4,
        'GR': 4,
        'KP': 2,
        'RN': 2,
        'RB': 3,
        'U': 0
    }
    return H_dist[n]


Graph_nodes = {
    'H': [('PWD', 1.1), ('DL', 0.6), ('KP', 0.8)],
    'PWD': [('MR', 0.7)],
    'MR': [('MS', 0.9)],
    'MS': [('PS', 0.3)],
    'PS': [('U', 0.45)],
    'DL': [('KR', 0.2)],
    'KR': [('GR', 1)],
    'GR': [('PS', 0.25)],
    'KP': [('RN', 0.1)],
    'RN': [('GR', 0.4), ('RB', 0.7)],
    'RB': [('U', 0.2)],
    'U': None
}
```

Result Analysis

After Using A Star Search Algorithm on this designed map, on output we can find the shortest path



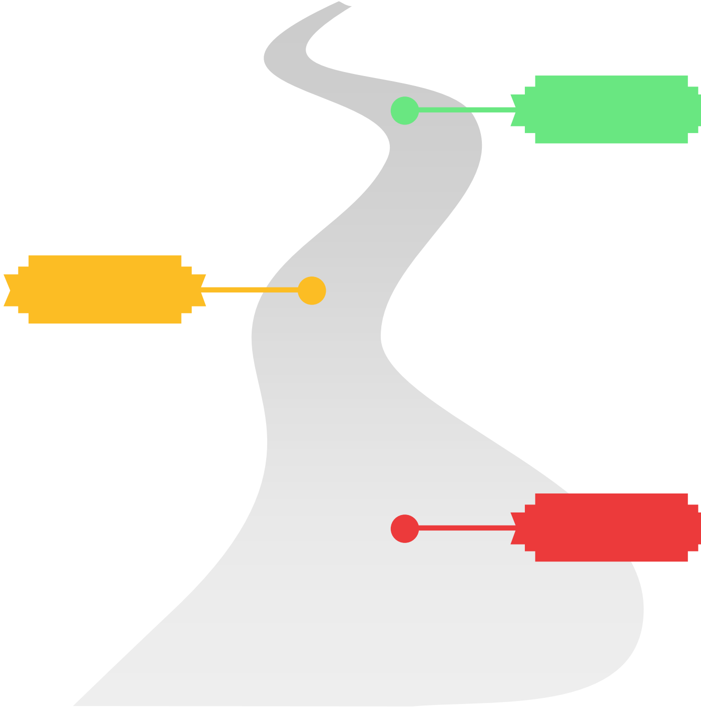
```
Path found: ['Dhanmondi (Home)'--> 'Kalabagan Park'--> 'Road No 8'--> 'Raza Bazar'--> 'UAP']

The path cost is 1.80 Km
```

So, we can say that that is the most optimal and shortest path.

Conclusion

- In this project, after successful implementation, A* search algorithm gives the most optimal path as output.

- In conclusion, A* search algorithm is a powerful and beneficial algorithm with all the potential. So we can use this algorithm for approximate the shortest path in real-life situation, like - in maps, games, robotics etc.



Thank You