



Department of Computer Science and Engineering

Project: 01

Course Title – Artificial Intelligence and Expert Systems Lab

Course Code- CSE 404

Year- 4th

Semester- 1st

Submitted to:

Dr. Nasima Begum
Associate Professor
University of Asia Pacific

Submitted by:

Name- Md. Abdur Rashid
Section- A1
ID- 19101008

Project Name: Implementation of a small address map (from my own home to UAP) using A* Search Algorithm.

Introduction:

The project problem is to implementation of a small address map from my own home to UAP, using A* search algorithm and then 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. Here, I will find the most optimal path from my home (Mugdapara) to my university (UAP) using A* search algorithm.

Objective:

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

Tools And Languages:

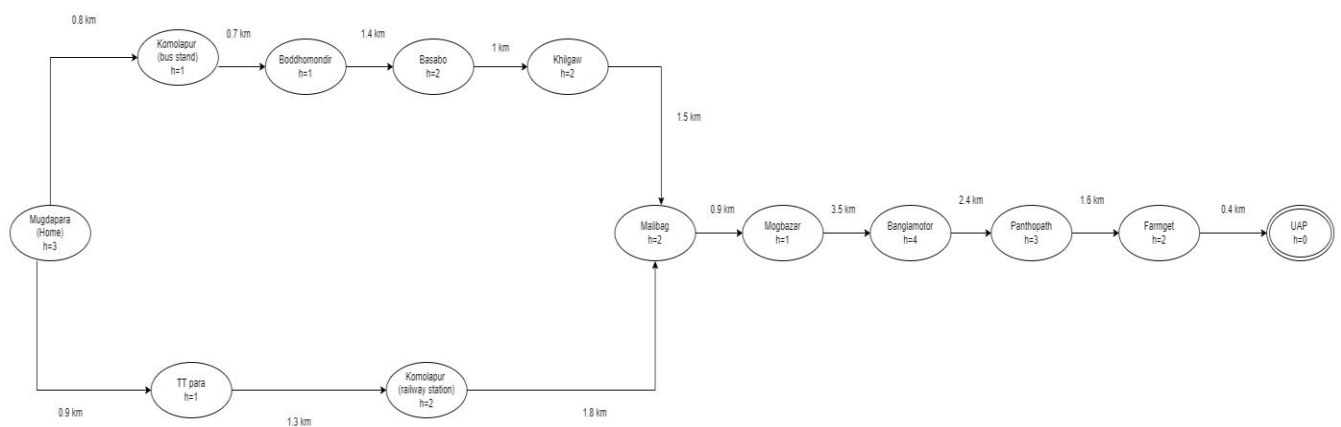
Distance Measurement: Google Maps

Map Designing: app diagram dot net

IDE: PyCharm

Programing Language: Python

Designed Map:



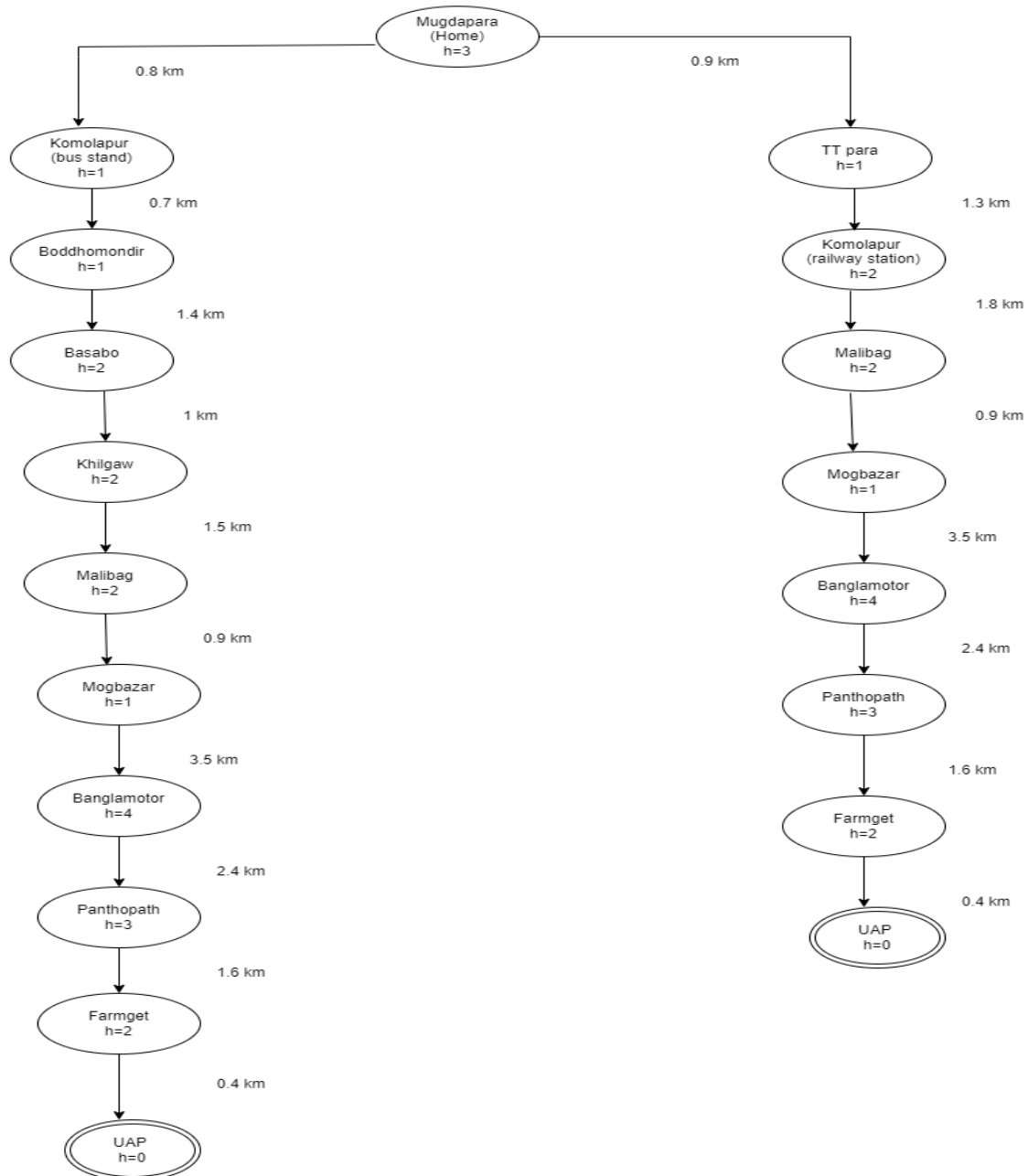
Here,

Start node: Mugdapara

Goal node: UAP

Cost in distance: Kilometer (km)

Search tree of designed map:



Implemented code using python:

```
FileEditor
pythonProject1 - a_search.py

a_search.py
1 def a_star_search(start, goal):
2     open_fringe = set(start)
3     close_fringe = set()
4     g = {} # here I store distance from starting node
5     parents = {} # here parents contains an adjacency map of all nodes
6
7     # starting node from 1st node is zero
8     g[start] = 0
9
10    parents[start] = start # start node
11
12    while len(open_fringe) > 0:
13        n = None
14
15        for v in open_fringe:
16            if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):
17                n = v
18
19        if n == goal or Graph_nodes[n] == None:
20            pass
21        else:
22            for (m, weight) in get_neighbors(n):
23                if m not in open_fringe and m not in close_fringe:
24                    open_fringe.add(m)
25                    parents[m] = n
26                    g[m] = g[n] + weight
27
28            else:
29                a_star_search() while len(open_fringe) > 0 else for (m, weight) in get_neighbor... if m not in open_fringe and m n...
```

```
FileEditor
pythonProject1 - a_search.py

a_search.py
28 else:
29     if g[m] > g[n] + weight:
30         # update g(m)
31         g[m] = g[n] + weight
32         # change parent of m to n
33         parents[m] = n
34         # if m in closed set, remove and add to open
35         if m in close_fringe:
36             close_fringe.remove(m)
37             open_fringe.add(m)
38
39     if n == None:
40         print('Path does not exist!')
41         return None
42
43     if n == goal:
44         path = []
45         path_cp = []
46         full = {
47             'H': "Mugdapara (Home)",
48             'KBS': "KomoLapur Bus Stand",
49             'Bod': "Boddhomondin",
50             'Bas': "Basabo",
51             'K': "Khilgaw",
52             'Mal': "Malibag",
53             'Mog': "Mogbazan",
54             'Ban': "Banglamotor",
55             'P': "Panthopath",
56         }
57
58     a_star_search() while len(open_fringe) > 0 else for (m, weight) in get_neighbor... if m not in open_fringe and m n...
```

```
Refactor Run Tools VCS Window Help pythonProject1 - a_search.py

a_search.py
55     'P': 'Panthopath',
56     'Farm': 'Farmget',
57     'TT': 'TT para',
58     'KRS': "Komalapur Railway station",
59     'U': "UAP"
60 }
61 while parents[n] != n:
62     path.append(n)
63     path_cp.append(full[n])
64     n = parents[n]
65
66 path.append(start)
67 path_cp.append(full[start])
68 path.reverse()
69 path_cp.reverse()
70 print('Path found: {}'.format(str(path_cp).replace(",", " ->")))
71 return path
72
73 open_fringe.remove(n)
74 close_fringe.add(n)
75
76 print('Path does not exist!')
77 return None
78
79 def get_neighbors(v):
80     if v in Graph_nodes:
81         return Graph_nodes[v]
82     else:
83         a_star_search() while len(open_fringe) > 0 else for (m, weight) in get_neighbor... if m not in open_fringe and m
```

```
Refactor Run Tools VCS Window Help pythonProject1 - a_search.py

a_search.py
82 else:
83     return None
84
85 def heuristic(n):
86     H_dist = {
87         'H': 3,
88         'KBS': 1,
89         'Bod': 1,
90         'Bas': 2,
91         'K': 2,
92         'MaL': 2,
93         'Mog': 1,
94         'Ban': 4,
95         'P': 3,
96         'Farm': 2,
97         'TT': 1,
98         'KRS': 2,
99         'U': 0
100     }
101     return H_dist[n]
102
103 Graph_nodes = {
104     'H': [('KBS', 0.8), ('TT', 0.9)],
105     'KBS': [('Bod', 0.7)],
106     'Bod': [('Bas', 1.4)],
107     'Bas': [('K', 1)],
108     'K': [('MaL', 1.5)],
109     'MaL': [('Mog', 0.9)],
110     'Mog': [('Ban', 0.9)],
111     'Ban': [('P', 0.9)],
112     'P': [('Farm', 0.9)],
113     'Farm': [('TT', 0.9)],
114     'TT': [('KRS', 0.9)],
115     'KRS': [('U', 0.9)]
116 }
```

```

186     'Bod': [('Bas', 1.4)],
187     'Bas': [('K', 1)],
188     'K': [('Mal', 1.5)],
189     'Mal': [('Mog', 0.9)],
190     'Mog': [('Ban', 3.5)],
191     'Ban': [('P', 2.4)],
192     'P': [('Farm', 1.6)],
193     'Farm': [('U', 0.4)],
194     'TT': [('KRS', 1.3)],
195     'KRS': [('MaL', 1.8)],
196     'PP': [('U', 0.5)],
197     'U': None
198 }
199
200 path = a_star_search('H', 'U')
201 path_cost = 0.0
202
203 for i in range(len(path) - 1):
204     for key, value in Graph_nodes[path[i]]:
205         if key == path[i + 1]:
206             path_cost += value
207             break
208 print("The path cost is %.2f Km" % path_cost)

```

Output:

```

Run: a_search x
"D:\python code\pythonProject1\venv\Scripts\python.exe" "D:/python code/pythonProject1/a_search.py"
Path found: ['Mugdapara (Home)'--> 'TT para'--> 'Komolapur Railway station'--> 'Malibag'--> 'Mogbazar'--> 'Banglamotor'--> 'Panthopath'--> 'Farmget'--> 'UAP']
The path cost is 12.80 Km
Process finished with exit code 0

```

Output Analysis:

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

Mugdapara (Home) --> TT para --> Komolapur Railway station --> Malibag --> Mogbazar --> Banglamotor --> Panthopath --> Farmget --> UAP.

Path cost: $(0.9+1.3+1.8+0.9+3.5+2.4+1.6+0.4)$ km = 12.80 km

Conclusion:

In this project, after successful implementation, A* search algorithm gives the most optimal path as output. So we can use this algorithm for approximate the shortest path in real-life situation, like - in maps, games etc.