



**Course Title :**

**Artificial Intelligence and Expert Systems Lab**

**Course Code : CSE 404**

**Submitted By:**

**Humayra Tabassum**

Section: A(A2)

ID:18201025

**Submitted to:**

**Dr. Nasima Begum**

Assistant Professor

Department of CSE, UAP

## Project Name: Implementation of a small address map using A\* search algorithm

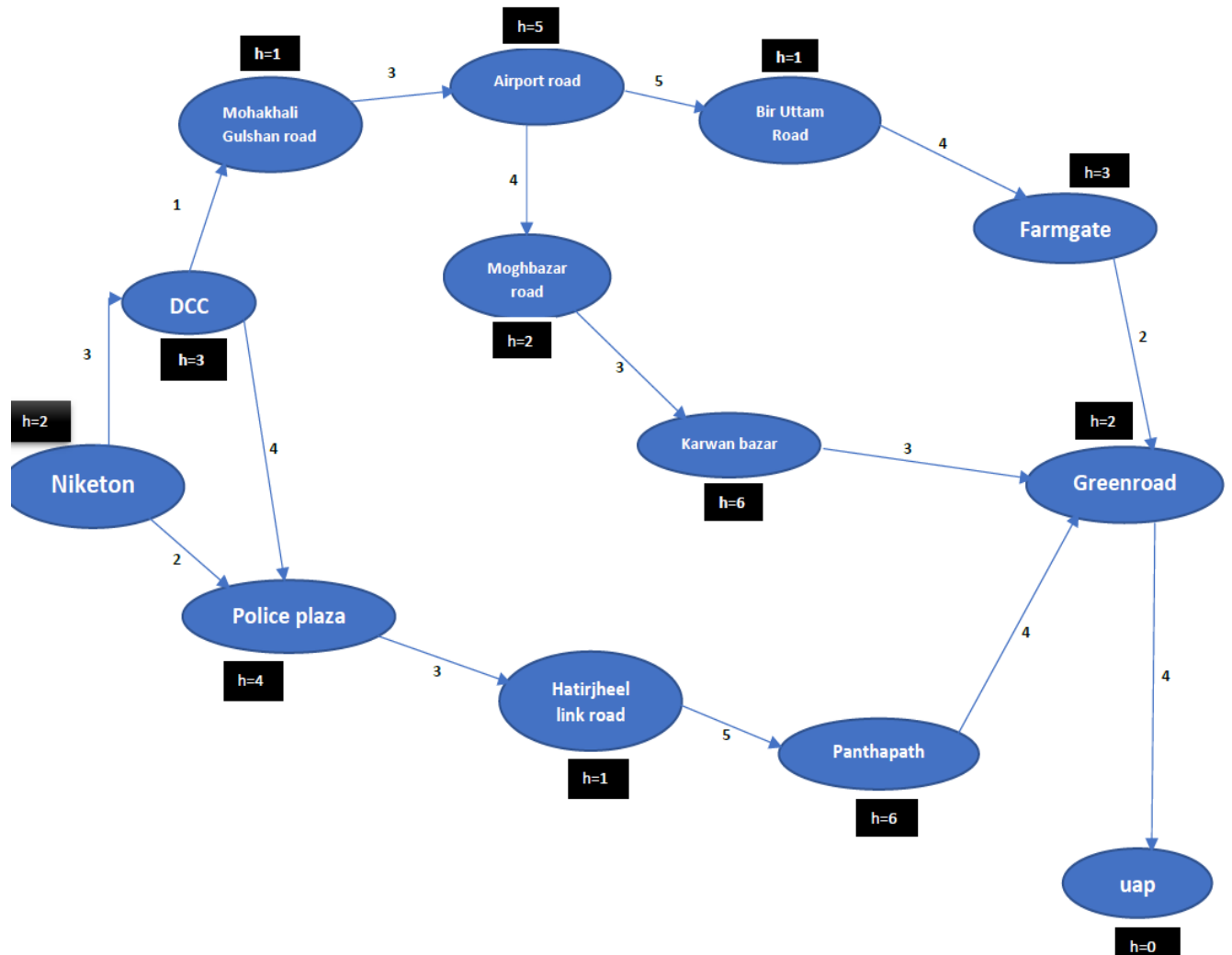
### Project Description:

The project problem is to implementation of a small address map from my home to UAP, using A\* search algorithm and then find out the optimal path.

### A\* search algorithm :

It is an informed search algorithm. Informed search algorithm contains an array of knowledge such as how far a node is from the goal, path cost, how to reach to goal node etc. A\* search algorithm combines both UCS and greedy best first search algorithms.

Designed map from my home to uap:



Here,

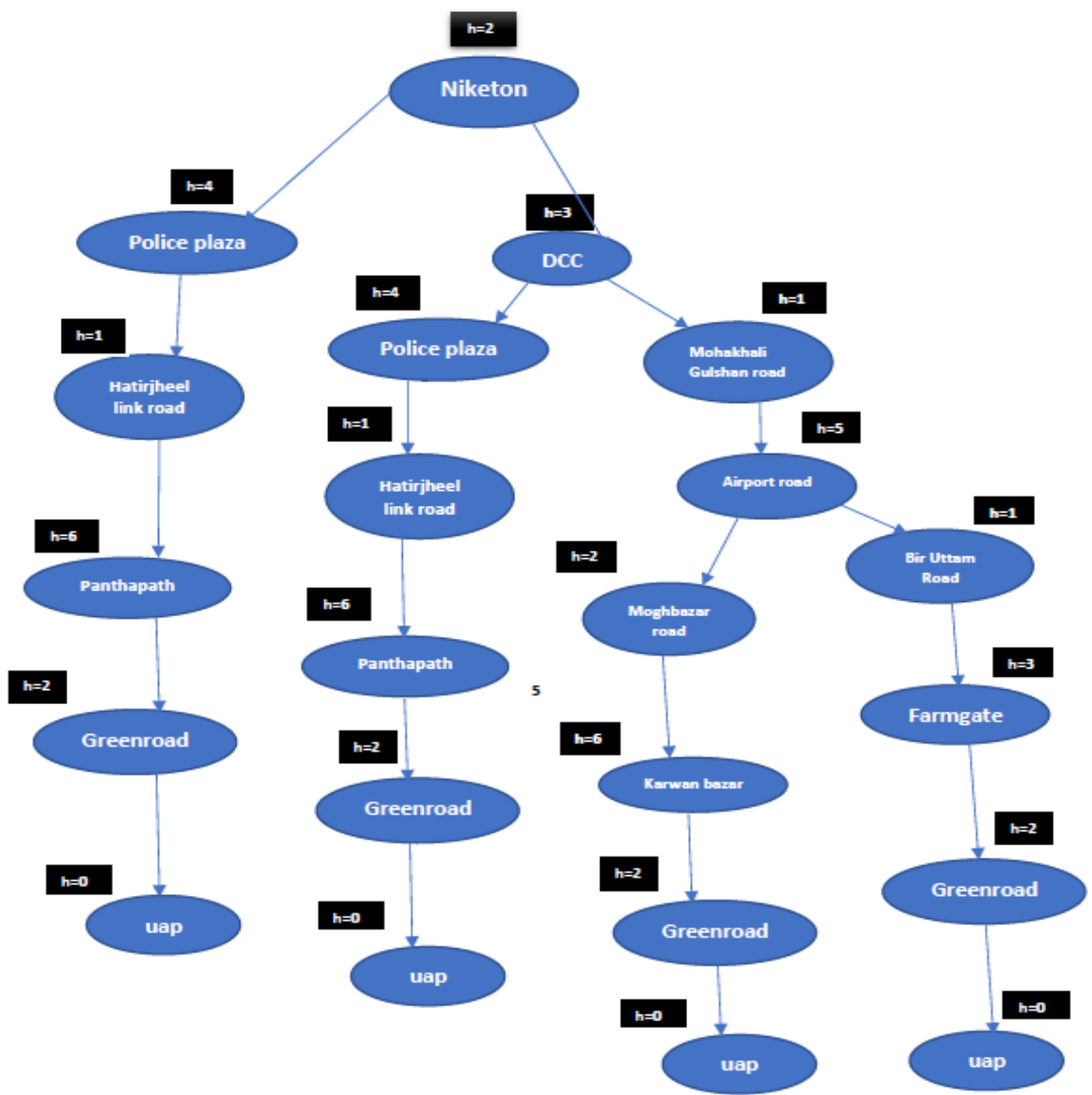
$g(n)$  = Path cost

$h(n)$  = Heuristic cost

$f(n)$  = Evaluation function = Estimated cost of the cheapest solution.

For A\* algorithm,  $f(n) = g(n) + h(n)$ .

Search Tree for the designed map:



## Implementation:

```
Project a search.py X
C: > Users > USER > Desktop > Project a search.py > a_star_search

1  def a_star_search(start, goal):
2      open_fringe = set(start)
3      close_fringe = set()
4      g = {}
5      parents = {}
6
7      g[start] = 0
8
9      parents[start] = start
10
11     while len(open_fringe) > 0:
12         n = None
13
14         for v in open_fringe:
15             if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):
16                 n = v
17
18         if n == goal or Graph_nodes[n] == None:
19             pass
20         else:
21             for (m, weight) in get_neighbors(n):
22
23                 if m not in open_fringe and m not in close_fringe:
24                     open_fringe.add(m)
25                     parents[m] = n
26
27                     g[m] = g[n] + weight
28
29                 else:
30                     if g[m] > g[n] + weight:
31                         g[m] = g[n] + weight
32                         parents[m] = n
33
34                         if m in close_fringe:
35                             close_fringe.remove(m)
36                             open_fringe.add(m)
37
38             if n == None:
39                 print('Path does not exist!')
40                 return None
41
42             if n == goal:
43                 path = []
44                 path_cp = []
45                 full = {
46                     'FG': [('GR', 2)],
47                     'GR': [('U', 4)],
48                     'MR': [('KB', 3)],
49                     'KB': [('GR', 3)],
50                     'PZ': [('HLR', 3)],
51                     'HLR': [('PP', 5)],
52                     'PP': [('GR', 4)],
53                     'U': None
54                 }
55
56                 path = a_star_search('H', 'U')
57
58                 path_cost = 0.0
59
60                 for i in range(len(path)-1):
61                     for key, value in Graph_nodes[path[i]]:
62                         if key == path[i+1]:
63                             path_cost += value
64                             break
65
66                 print("The path cost is %.2f " % path_cost)
```

### Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\python\humayra Tabassum> & "C:/Program Files/Python38/python.exe" "c:/Users/USER/Desktop/Project a search.py"
Path found: ['Niketon (Home)'--> 'Police Plaza'--> 'Hatirjheel Link Road'--> 'Panta Path'--> 'Green Road'--> 'UAP']
The path cost is 18.00
PS E:\python\humayra Tabassum>
```

### Result:

After implementation of A\* algorithm my shortest path cost is 18 and the path is home to policeplaza to hatirjheel link road to pantapath to greenroad to uap.

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