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

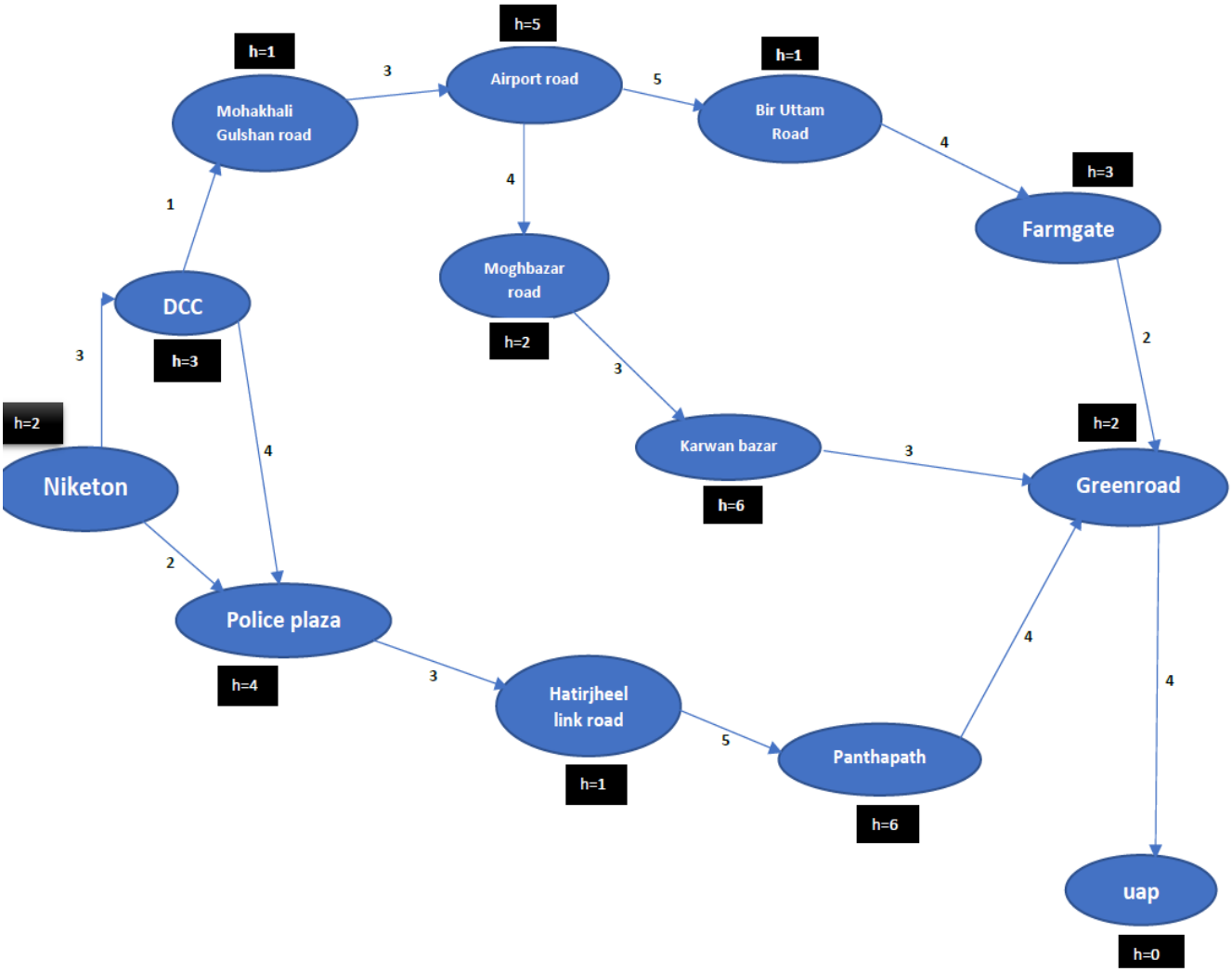
Project Description:

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

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

```
def a_star_search(start, goal):
        open_fringe = set(start)
        close_fringe = set()
        g = {}
        parents = {}

        g[start] = 0

        parents[start] = start

        while len(open_fringe) > 0:
            n = None

            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):

                    if m not in open_fringe and m not in close_fringe:
                        open_fringe.add(m)
                        parents[m] = n
                        g[m] = g[n] + weight



                    else:
                        if g[m] > g[n] + weight:

                            g[m] = g[n] + weight

                            parents[m] = n

                            if m in close_fringe:
                                close_fringe.remove(m)
                                open_fringe.add(m)

            if n == None:
                print('Path does not exist!')
                return None

            if n == goal:
                path = []
                path_cp = []
                full = {
```

Project a search.py

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

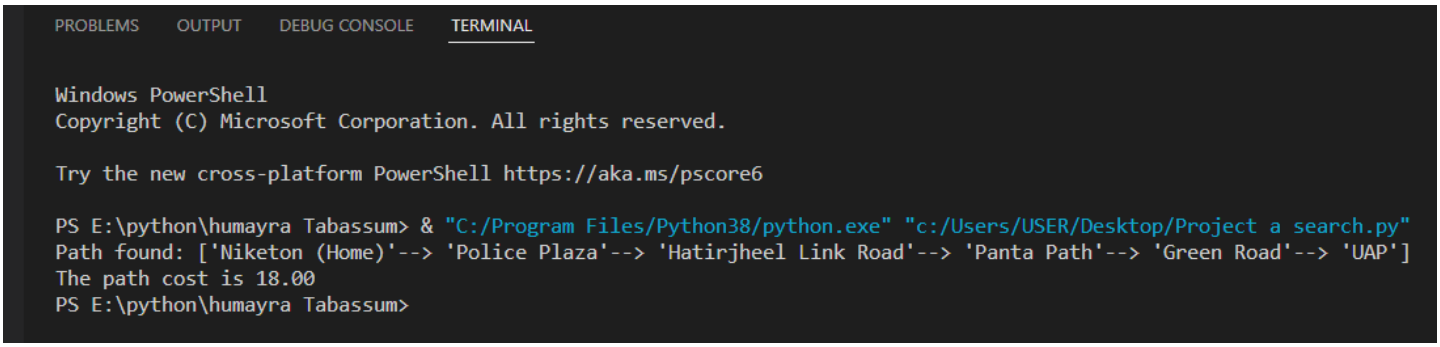
```
            'FG': [('GR', 2)],
            'GR': [('U', 4)],
            'MR': [('KB', 3)],
            'KB': [('GR', 3)],
            'PZ': [('HLR', 3)],
            'HLR': [('PP', 5)],
            'PP': [('GR', 4)],
            'U': None
    }

    path = a_star_search('H', 'U')

    path_cost = 0.0

    for i in range(len(path)-1):
        for key, value in Graph_nodes[path[i]]:
            if key == path[i+1]:
                path_cost += value
                break
    print("The path cost is %.2f " % path_cost)
```

Output:



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.