CSE 404 - Artificial Intelligence and Expert System Lab

Project Report

Project Name : Implementation of a small address map (from Home to UAP) using A* search algorithm.

Submitted By:
Md. Shamaun Nabi
ID : 1820150 , Section: B1

4th Year 1st Semester
Department Of Computer Science,
University Of Asia Pacific, Dhaka.
March 23, 2022

Introduction

A* algorithm is a searching algorithm that searches for the shortest path between the initial state to the final state. 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. So, here in this project I will find the most optimal path from my home **(Bashiruddin Road)** to my university **(UAP)** using A* search algorithm.

Objective

In this project, I have to reach UAP from my home Bashiruddin Road Masjid by using the shortest path. There are several paths between Bashiruddin road to UAP. But not all of those paths 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 (Bashiruddin Road) to my university (UAP)

Tools And Languages

- **Map Designing**: Draw.io
- **Programing Language**: Python
- **IDE**: Vs Code
- **Distance Measurement**: Google Maps

<u>Designed Map</u>

Id: 18201050
Start Node : BashirUddin Road Masjid.
Goal Node: UAP

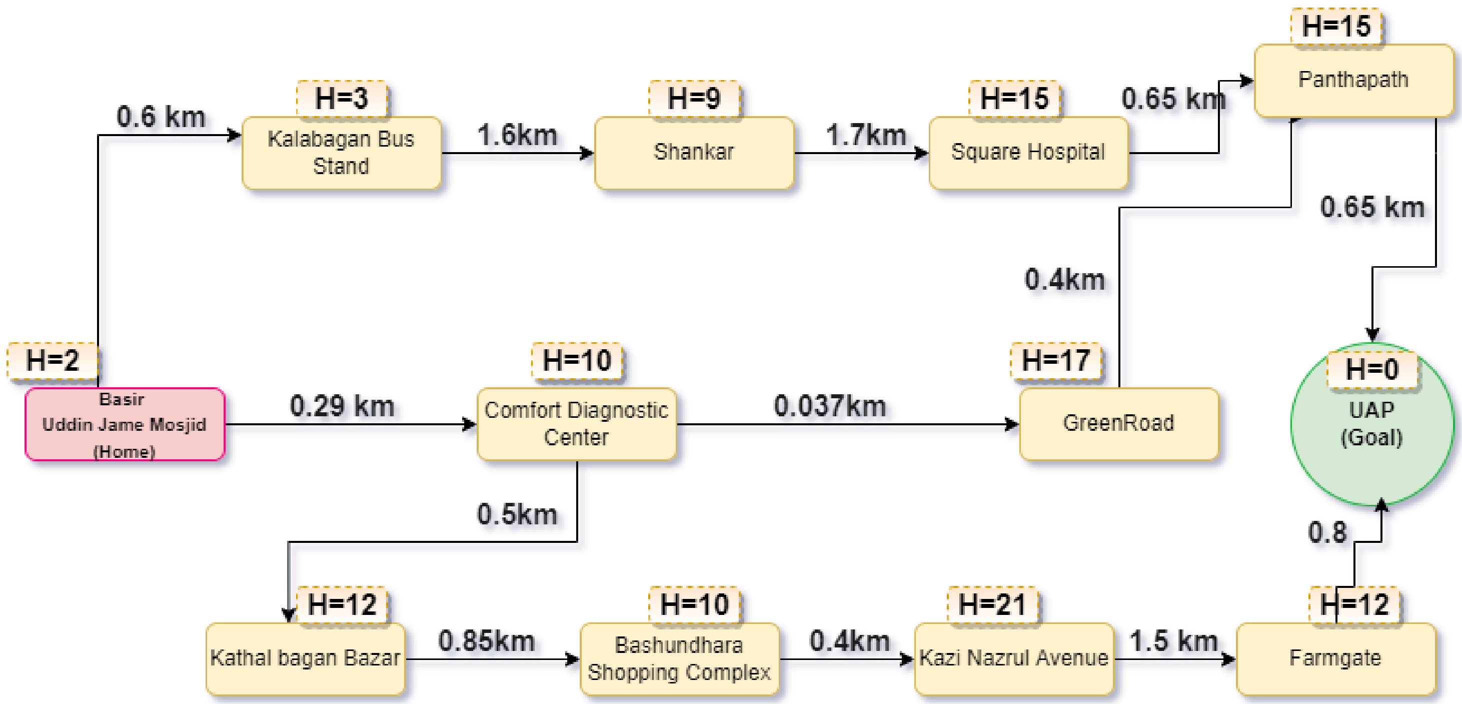

Fig: Designed Map From Home To UAP

Search Tree Of Designed Map

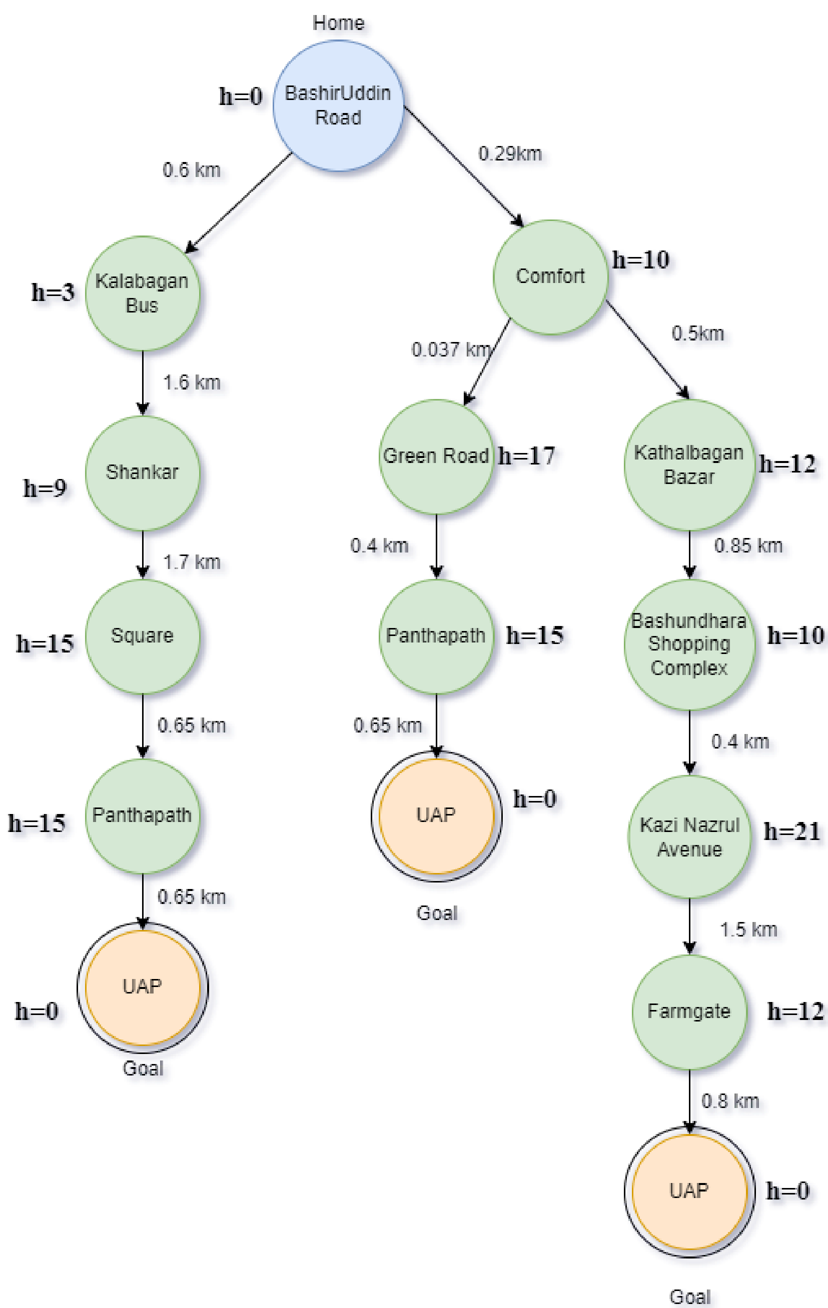

Fig: Tree for Designed Map

Implementation Using Python

```python
#18201050 Shamaun nabi Section:B 🔥

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
                        g[m] = g[n] + weight
```

```python
#18201050 Shamaun nabi Section:B 🔥

                    #for each node m,compare its distance from start i.e g(m) to the
                    #from start through n node
                    else:
                        if g[m] > g[n] + weight:
                            #update g(m)
                            g[m] = g[n] + weight
                            #change parent of m to n
                            parents[m] = n

                            #if m in closed set,remove and add to open
                            if m in close_fringe:
                                close_fringe.remove(m)
                                open_fringe.add(m)

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

            # if the current node is the goal
            # then  begin reconstructing the path from it to the start
            if n == goal:
                path = []
                path_cp = []
                full = {
                'B': "BashirUddin Road (Home)",
                'KBH': "Kalabagan Bus Stand",
                'SH': "Shankar",
                'SQ': "Square Hospital",
                'PS': "Panthapath Signal",
                'CMF': "Comfort Hospital",
                'GR': "Green Road",
                'KB': "KathalBagan Bazar",
                'BS': "Bashundhara Shopping Complex",
                'KZV': "Kazi Nazrul Avenue",
                'FRM': "Farmgate",
                'U': "UAP"
                }
```

```python
                while parents[n] != n:
                    path.append(n)
                    path_cp.append(full[n])
                    n = parents[n]

                path.append(start)
                path_cp.append(full[start])
                path.reverse()
                path_cp.reverse()
                print('Path found: {}'.format(str(path_cp).replace(",","-->")))
                return path

            open_fringe.remove(n)
            close_fringe.add(n)

        print('Path does not exist!')
        return None

def get_neighbors(v):
    if v in Graph_nodes:
        return Graph_nodes[v]
    else:
        return None

def heuristic(n):

        H_dist = {
            'B': 2,
            'KBH':3,
            'SH': 9,
            'SQ': 15,
            'PS': 15,
            'CMF': 10,
            'GR': 17,
            'KB': 12,
            'BS': 10,
            'KZV': 21,
            'FRM': 12,
            'U': 0
        }
        return H_dist[n]
```

```python
Graph_nodes = {
        'B': [('KBH', 0.6), ('CMF', 0.29)],
        'KBH': [('SH', 1.6)],
        'SH': [('SQ', 1.7)],
        'SQ': [('PS', 0.65)],
        'PS': [('U', 0.65)],
        'CMF': [('GR', 0.037),('KB',0.5)],
        'GR': [('PS', 0.4)],
        'PS': [('U', 0.65)],
        'KB': [('BS', 0.85)],
        'BS': [('KZV', 0.4)],
        'KZV': [('FRM', 1.5)],
        'FRM': [('U', 0.8)],
        'U': None
}

path = a_star_search('B', '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 Km" % path_cost)
```

Output:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                                    Code
[Running] python -u "h:\University of Asia Pasific\4-1_Spring_2021\CSE 404 - Artificial Intelligence  Lab\Lab Project\Project 2 -  A Star Search Map\18201050 My Road map.py"
Path found: ['BashirUddin Road (Home)'--> 'Comfort Hospital'--> 'Green Road'--> 'Panthapath Signal'--> 'UAP']
The path cost is 1.38 Km

[Done] exited with code=0 in 0.169 seconds
```

Result Analysis

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

['BashirUddin Road (Home)'--> 'Comfort Hospital'--> 'Green Road'--> 'Panthapath Signal'--> 'UAP']

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

Conclusion

In this project, after successful implementation, a star search algorithm gives the most optimal path as output. In conclusion, a star 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.

–End–