Problem Title:

Implementation of A* search algorithm.

Problem Description:

First, I need to construct a map from my home to SU. Afterward, I will find the most optimal path from my home (Shymoli) to my university (SU) using the A* search algorithm.

Tools and Language:

1. Vs Code
2. Python
3. Microsoft Word
4. Paint
5. Google Maps
6. Web Whiteboard

Google Map View:

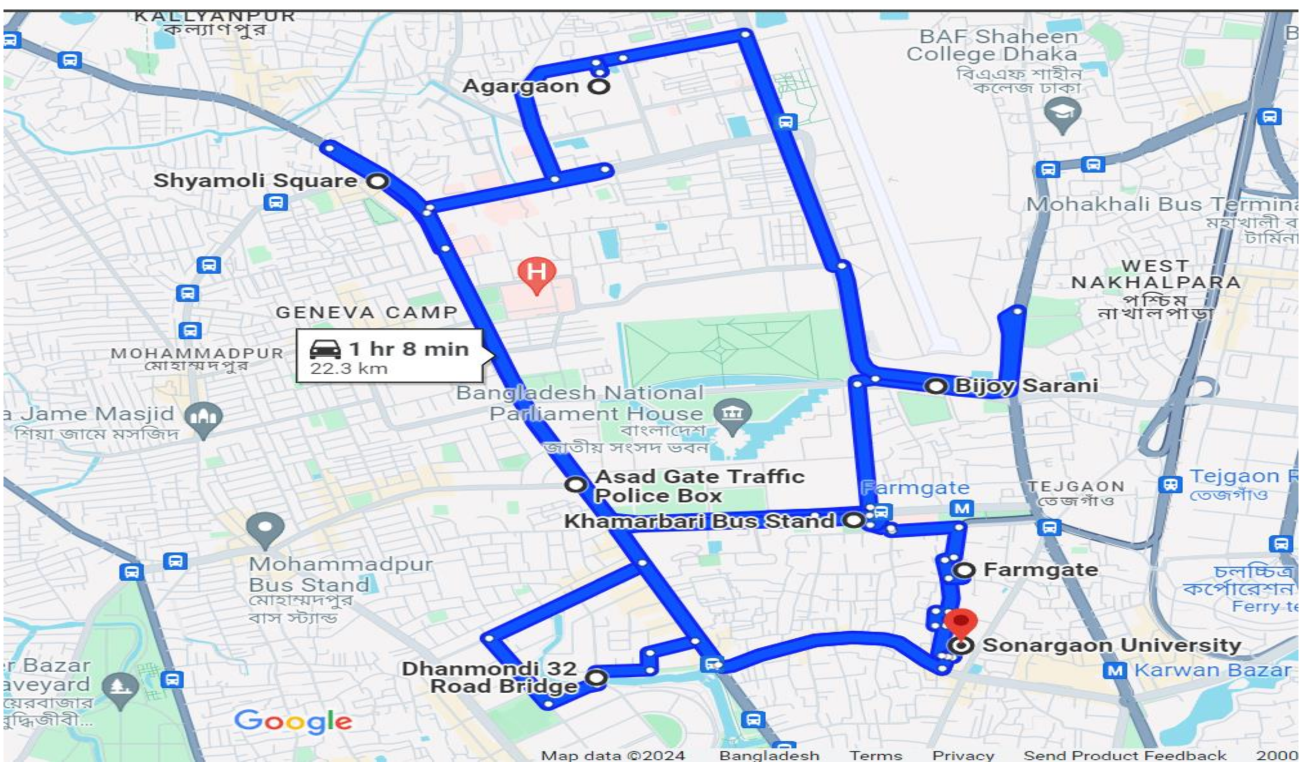
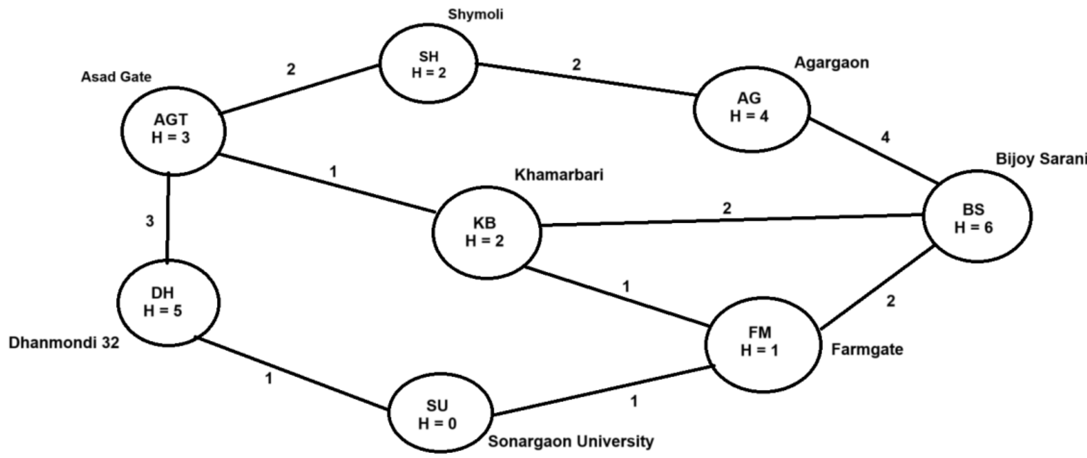


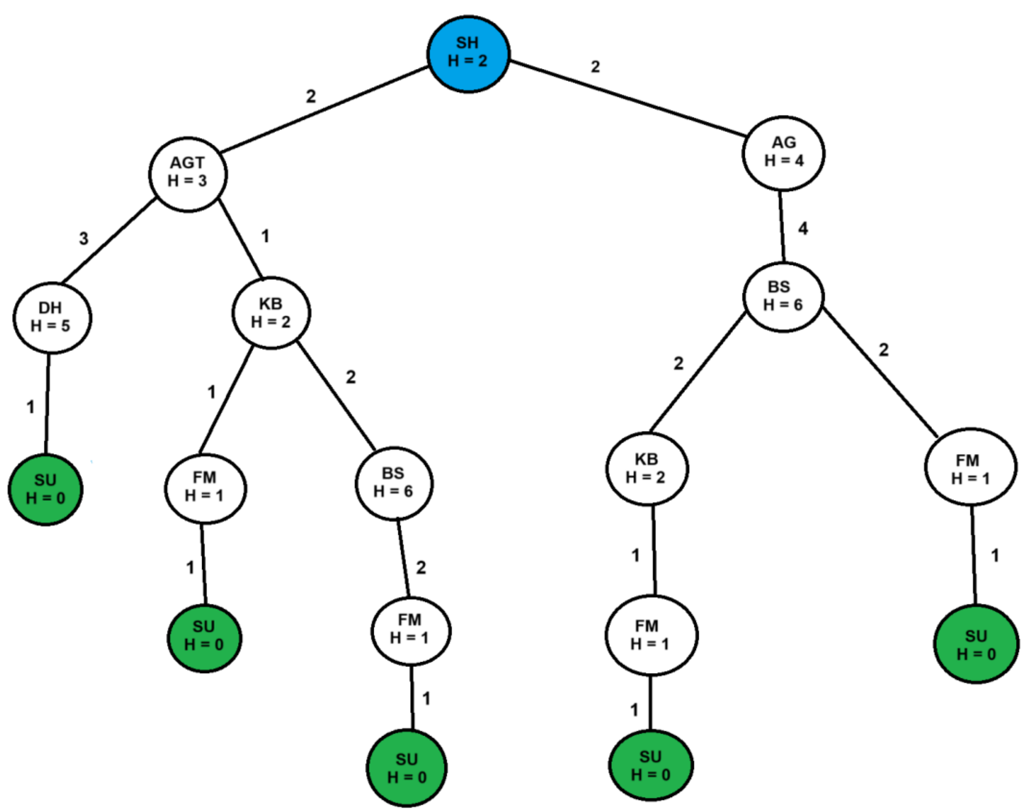
Fig: Map

The above diagram is a map from my home to SU with different possible routes.

State Graph:



Search Tree:



Here,

'SH' = 'Shymoli',
'AGT' = 'Asad Gate',
'DH' = 'Dhanmondi 32',
'KB' = 'Khamarbari',
'AG' = 'Agargaon',
'BS' = 'Bijoy Sarani',
'FM' = 'Farmgate',
'SU' = 'Sonargaon University'

Code:

```
def get_child(current_node):
    if current_node in directions:
        return directions[current_node]
    else:
        return []

def heuristic(n):
    Heuristic_distance = {
        'SH': 2,
        'AGT': 3,
        'DH': 5,
        'KB': 2,
        'AG': 4,
        'BS': 6,
        'FM': 1,
        'SU': 0
    }
    return Heuristic_distance[n]

directions = {
    'SH': [('AGT', 2), ('AG', 2)],
    'AGT': [('KB', 1), ('DH', 3)],
    'DH': [('SU', 1)],
    'KB': [('AGT', 1), ('BS', 2), ('FM', 1)],
    'AG': [('BS', 4)],
```

```
    'BS': [('KB', 2), ('FM', 2)],
    'FM': [('SU', 1)],
    'SU': []
}

def aStarSearch(start, goal):
    o_fringe = {start}
    c_fringe = set()
    g_pathCost = {}
    root_node = {}
    g_pathCost[start] = 0
    root_node[start] = start

    while len(o_fringe) > 0:
        temp = None
        for current_node in o_fringe:
            if temp is None or g_pathCost[current_node] + heuristic(current_node) <
g_pathCost[temp] + heuristic(temp):
                temp = current_node
        if temp == goal:
            path_list = []
            routes_list = []
            routes = {
                'SH': "Shymoli (Home)",
                'AGT': "Asad Gate",
                'DH': "Dhanmondi 32",
                'KB': "Khamarbari",
                'AG': "Agargaon",
                'BS': "Bijoy Shorani",
                'FM': "Farmgate",
                'SU': "Sonargaon University"
            }
            while root_node[temp] != temp:
                path_list.append(temp)
                routes_list.append(routes[temp])
                temp = root_node[temp]
            path_list.append(start)
            routes_list.append(routes[start])
            path_list.reverse()
```

```
        routes_list.reverse()
        print('The most optimal path : {}'.format(str(routes_list).replace(",",
">>>")))
        return path_list

      o_fringe.remove(temp)
      c_fringe.add(temp)

      for (child_node, node_path_cost) in get_child(temp):
        if child_node not in o_fringe and child_node not in c_fringe:
          o_fringe.add(child_node)
          root_node[child_node] = temp
          g_pathCost[child_node] = g_pathCost[temp] + node_path_cost
        else:
          if g_pathCost[child_node] > g_pathCost[temp] + node_path_cost:
            g_pathCost[child_node] = g_pathCost[temp] + node_path_cost
            root_node[child_node] = temp
          if child_node in c_fringe:
            c_fringe.remove(child_node)
            o_fringe.add(child_node)

  print('No such route.')
  return 0

path = aStarSearch('SH', 'SU')
path_cost = 0.0

for i in range(len(path) - 1):
  for key, value in directions[path[i]]:
    if key == path[i + 1]:
      path_cost += value
      break

print("Path cost = %d km." % path_cost)
```

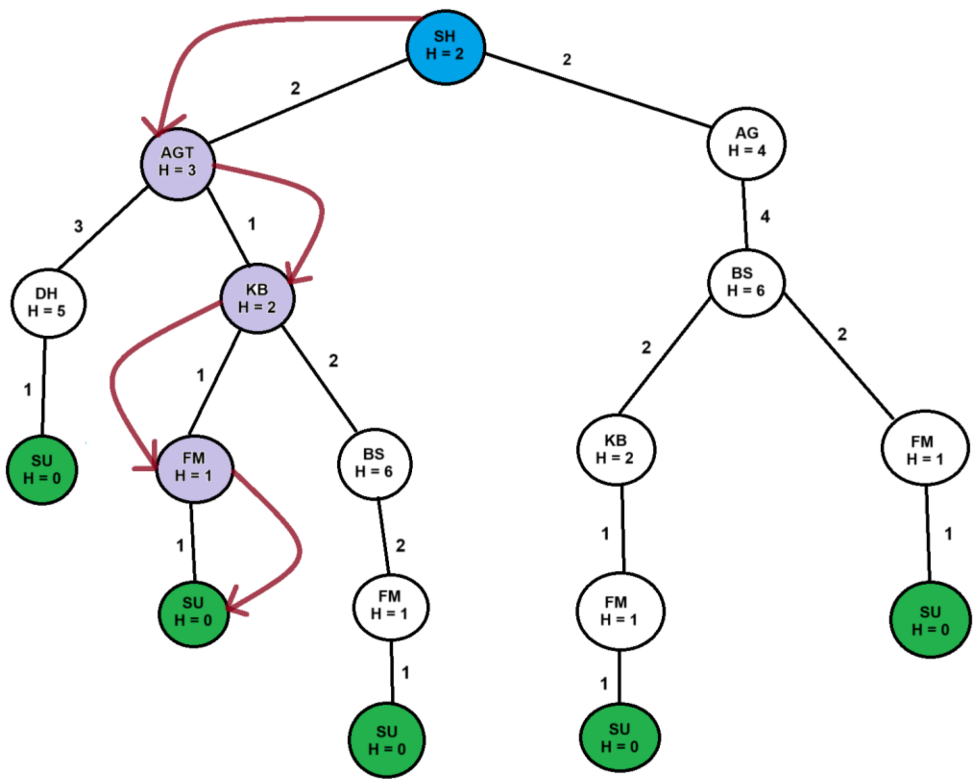
Output:

```
PS D:\10th semester all\10th-semester-all\sub_AI\coding\Academic Task\assignment_02> cd "d:\10th semester all\10th-semester-all\sub_AI\coding\Academi
c Task\assignment_02"
PS D:\10th semester all\10th-semester-all\sub_AI\coding\Academic Task\assignment_02> python -u "d:\10th semester all\10th-semester-all\sub_AI\coding\
Academic Task\assignment_02\maping.py"
The most optimal path : ['Shymoli (Home)'>>> 'Asad Gate'>>> 'Khamarbari'>>> 'Farmgate'>>> 'Sonargaon University']
Path cost = 5 km.
PS D:\10th semester all\10th-semester-all\sub_AI\coding\Academic Task\assignment_02>
```

Using the A* search algorithm we got the above result.

Output tree:

The return path is shown using the red arrow.



The most optimal path :
['Shymoli (Home)'>>> 'Asad Gate'>>> 'Khamarbari'>>> 'Farmgate'>>> 'Sonargaon University']
Path cost = 5 km.

Conclusion:

The provided code is a robust implementation of the A* search algorithm. It effectively constructs a map, uses heuristics to guide the search, and correctly finds and outputs the most optimal path and its cost. This method can be applied to various real-world routing and navigation problems where finding the shortest path is essential. The careful management of open and closed sets, along with accurate heuristic values, ensures that the algorithm performs efficiently and effectively.

Name : Salman Johir Tonoy

ID : CSE2102023012

Subject Name : artificial intelligence