**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

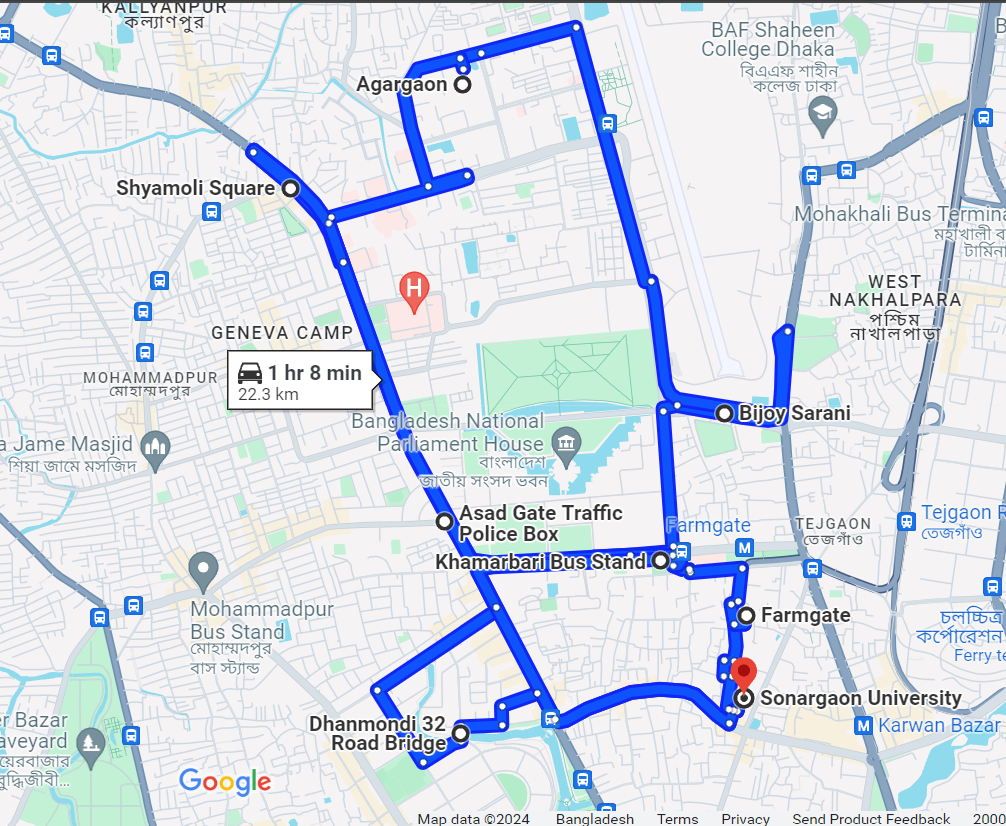
3. Microsoft Word

4. Paint

5. Google Maps

6. Web Whiteboard

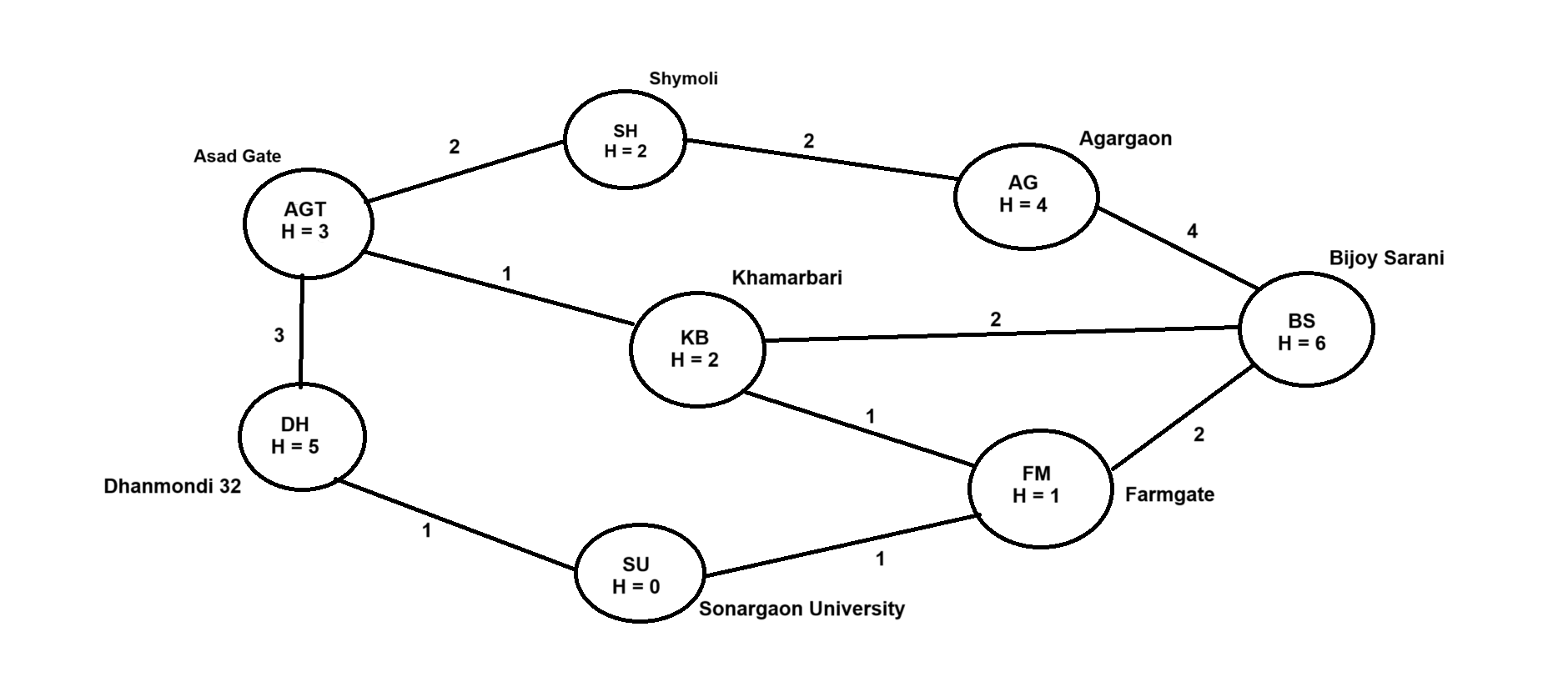
**Google Map View:**

**

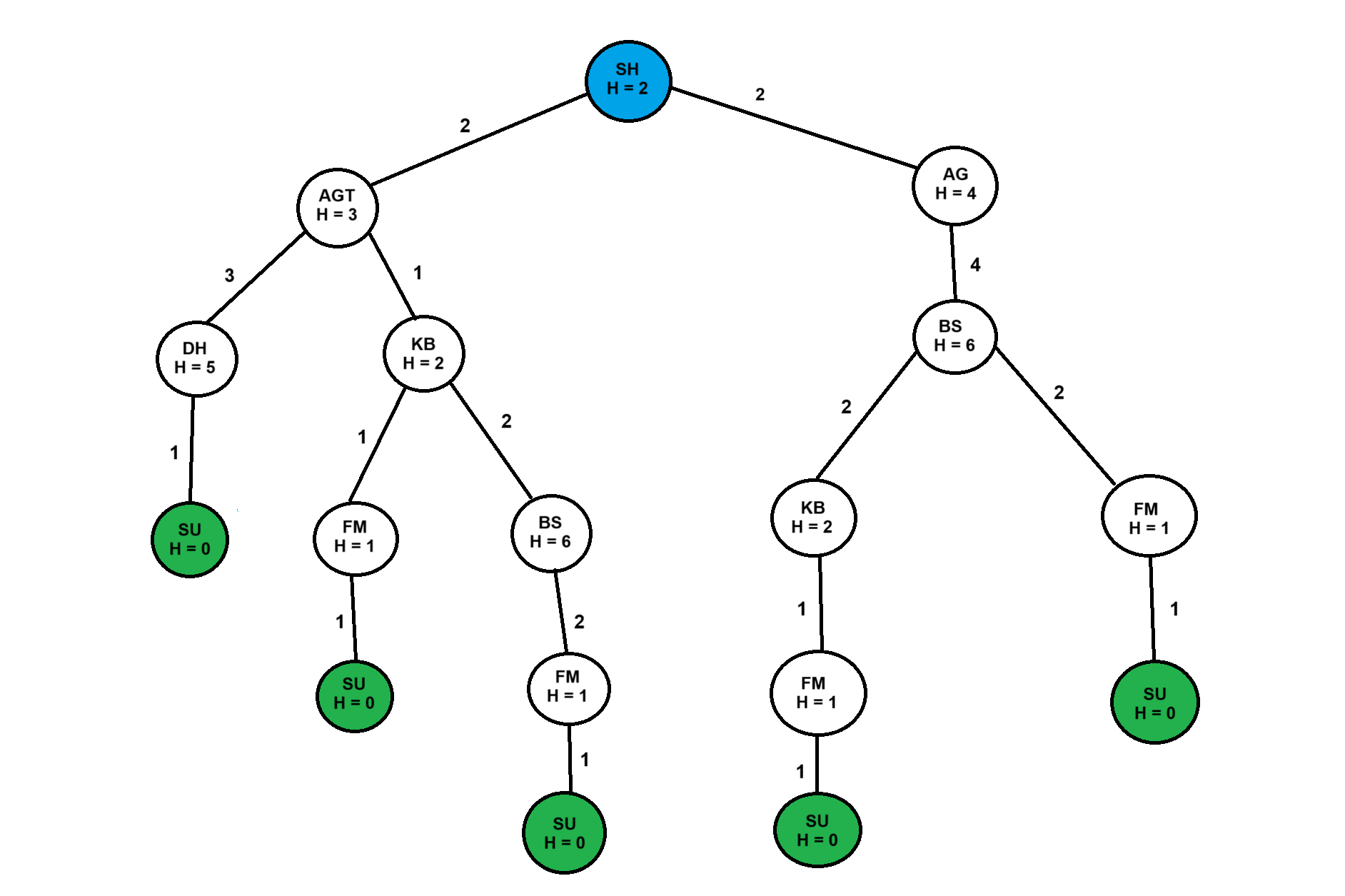
*Fig: Map*

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

**State Graph:**

****

**Search Tree:**

****

Here,



**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]]:**

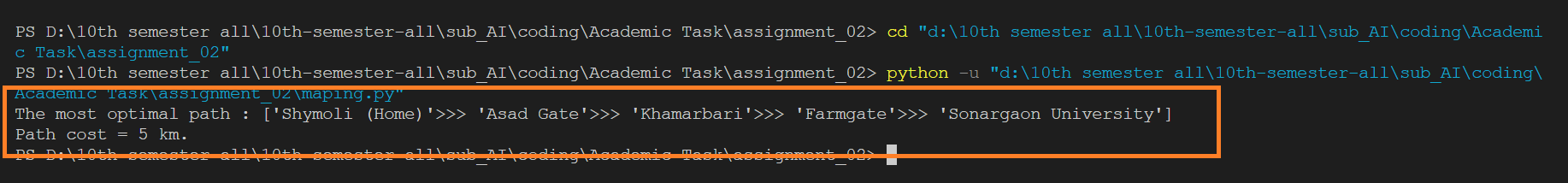
**if key == path[i + 1]:**

**path\_cost += value**

**break**

**print("Path cost = %d km." % path\_cost)**

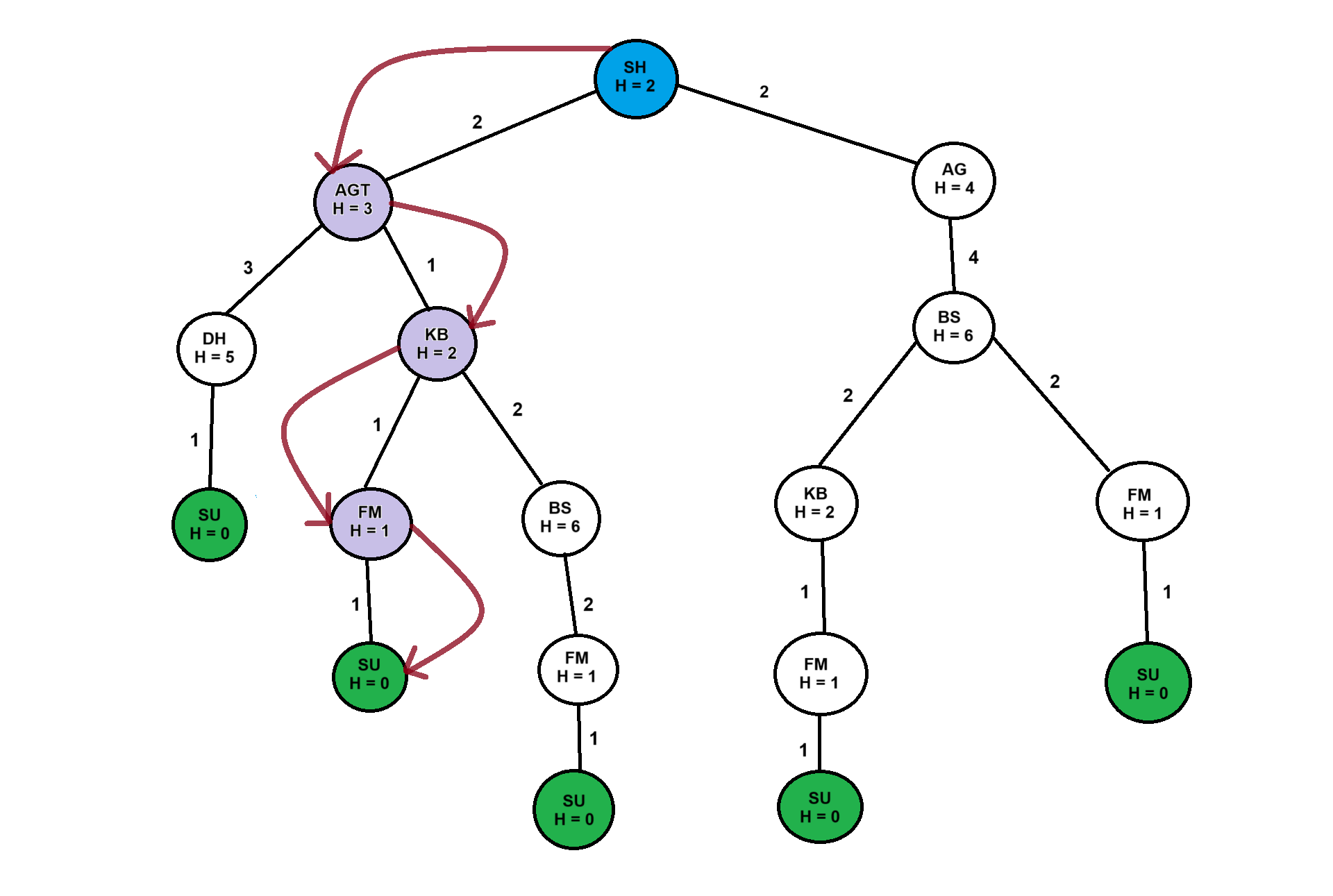
**Output:**

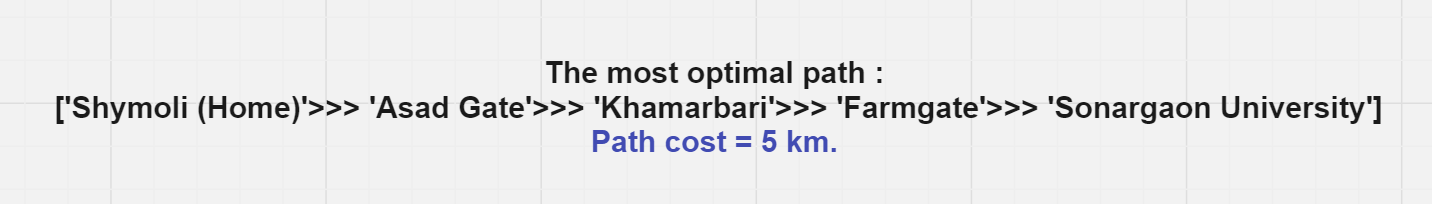
****

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

**Output tree:**

The return path is shown using the red arrow.



****

**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**