**Batch: IAI-2 Experiment Number: 3**

**Roll Number: 16010422234 Name: Chandana Ramesh Galgali**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Aim of the Experiment:** Implementation of Informed search algorithm - A\*

**Program/Steps:**

1. Implement A\* algorithm as discussed for graph traversal.
2. Print the contents of fringe/OPEN, CLOSED/Visited and the solution.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code:**

***class* Node:**

***def* \_\_init\_\_(self, state, parent=None, g=0, h=0):**

**self.state = state**

**self.parent = parent**

**self.g = g**

**self.h = h**

**self.f = g + h**

***def* a\_star(start, goal, successors, heuristic):**

**open\_list = [Node(start, None, 0, heuristic[start])]**

**closed\_set = set()**

***while* open\_list:**

**open\_list.sort(key=*lambda* x: x.f)**

**current\_node = open\_list.pop(0)**

***if* current\_node.state == goal:**

***return* reconstruct\_path(current\_node)**

**closed\_set.add(current\_node.state)**

***for* successor in successors(current\_node.state):**

***if* successor in closed\_set:**

***continue***

**g = current\_node.g + 1**

**h = heuristic[successor]**

**f = g + h**

**existing\_node = find\_node(open\_list, successor)**

***if* not existing\_node or f < existing\_node.f:**

***if* existing\_node:**

**open\_list.remove(existing\_node)**

**new\_node = Node(successor, current\_node, g, h)**

**open\_list.append(new\_node)**

***return* None**

***def* find\_node(node\_list, state):**

***for* node in node\_list:**

***if* node.state == state:**

***return* node**

***return* None**

***def* reconstruct\_path(node):**

**path = []**

***while* node:**

**path.insert(0, node.state)**

**node = node.parent**

***return* path**

***def* get\_graph():**

**graph = {}**

***while* True:**

**node = input("Enter a node (or 'done' to finish): ")**

***if* node.lower() == 'done':**

***break***

**successors = input("Enter successors separated by commas: ").split(',')**

**graph[node] = successors**

***return* graph**

***def* get\_heuristic\_values():**

**heuristic = {}**

**nodes = input("Enter nodes separated by commas: ").split(',')**

***for* node in nodes:**

**heuristic[node] = int(input(f"Enter heuristic value for {node}: "))**

***return* heuristic**

**graph = get\_graph()**

**heuristic\_values = get\_heuristic\_values()**

**start\_state = input("Enter the start state: ")**

**goal\_state = input("Enter the goal state: ")**

**path = a\_star(start\_state, goal\_state, graph.get, heuristic\_values)**

***if* path:**

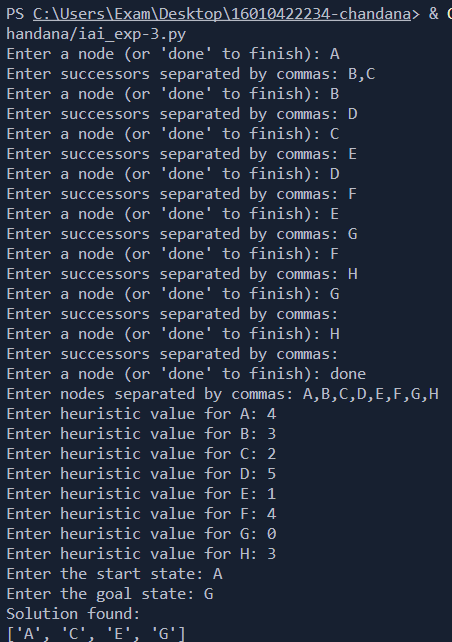
***print*("Solution found:")**

***print*(path)**

***else*:**

***print*("No solution found.")**

**Output/Result:**

****

**Outcomes: Analyze and formalize the problem (as a state space, graph, etc.) and select the appropriate search method and write the algorithm**

**Conclusion (Based on the Results and outcomes achieved):**

The successful implementation of the A\* algorithm demonstrated its capability to efficiently navigate through a search space, considering both the current cost and the estimated future cost. The algorithm is widely applicable in various fields, including artificial intelligence, robotics, and pathfinding applications.

**References:**

1. Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Second Edition, Pearson Publication
2. Luger, George F. Artificial Intelligence : Structures and strategies for complex problem solving, 2009, 6th Edition, Pearson Education