**Experiment: 1.1**

**Student Name:** SANJIV GUPTA **UID:** 21BCS-*3478*

**Branch:** CSE **Section/Group:** 21BCS-IOT-602B

**Semester:** 5th **Date:** 17/08/23

**Subject Name**: AIML Lab **Subject Code:** 21CSH-316

1. **AIM:** *Evaluate the performance and effectiveness of the A\* algorithm implementation in Python*
2. **Objective:** *The objective is to assess how well the A\* algorithm performs in solving a specific problem or scenario, and to analyze its effectiveness in comparison to other algorithms or approaches.*
3. **Tools/Resource Used:**

*1. Python programming language.*

*2. A\* algorithm implementation in Python.*

*3. Relevant data or problem scenario for testing the algorithm.*

1. **Algorithm:**

*1. Define the problem scenario or task for which the A\* algorithm will be used.*

*2. Implement the A\* algorithm in Python, taking into accounts the specific problem requirements and constraints.*

*3. Provide necessary data structures, such as graphs or grids, to represent the problem space.*

*4. Write code to initialize the start and goal states or nodes.*

*5. Implement the A\* algorithm, including the heuristic function and the necessary data structures, such as priority queues or heaps.*

*6. Run the algorithm on the given problem scenario and record the execution time.*

*7. Monitor and log the nodes expanded, the path generated, and any other relevant information during the algorithm's execution.*

*8. Repeat steps 4-7 for multiple problem scenarios or test cases, if applicable.dq*

1. **Program Code:**

*import heapq*

*class Node:*

*def \_\_init\_\_(self, position, parent=None):*

*self.position = position*

*self.parent = parent*

*self.g = 0 # Cost from start node to current node*

*self.h = 0 # Heuristic (estimated cost) from current node to goal node*

*self.f = 0 # Total cost (g + h)*

*def \_\_lt\_\_(self, other):*

*return self.f < other.f*

*def heuristic(node, goal):*

*return abs(node.position[0] - goal[0]) + abs(node.position[1] - goal[1])*

*def astar(grid, start, goal):*

*open\_list = []*

*closed\_set = set()*

*start\_node = Node(start)*

*goal\_node = Node(goal)*

*heapq.heappush(open\_list, start\_node)*

*while open\_list:*

*current\_node = heapq.heappop(open\_list)*

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

*path = []*

*while current\_node is not None:*

*path.append(current\_node.position)*

*current\_node = current\_node.parent*

*return path[::-1]*

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

*for next\_position in [(0, -1), (0, 1), (-1, 0), (1, 0)]:*

*node\_position = (current\_node.position[0] + next\_position[0], current\_node.position[1] + next\_position[1])*

*if node\_position[0] < 0 or node\_position[0] >= len(grid) or node\_position[1] < 0 or node\_position[1] >= len(grid[0]):*

*continue*

*if grid[node\_position[0]][node\_position[1]] == 1:*

*continue*

*if node\_position in closed\_set:*

*continue*

*new\_node = Node(node\_position, current\_node)*

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

*new\_node.h = heuristic(new\_node, goal\_node.position)*

*new\_node.f = new\_node.g + new\_node.h*

*for node in open\_list:*

*if new\_node.position == node.position and new\_node.f >= node.f:*

*break*

*else:*

*heapq.heappush(open\_list, new\_node)*

*return None # No path found*

*# Example usage*

*grid = [*

*[0, 0, 0, 0],*

*[0, 1, 1, 0],*

*[0, 0, 0, 0],*

*[0, 0, 1, 0]*

*]*

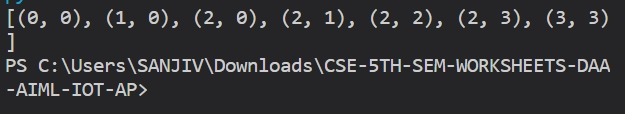
*start\_point = (0, 0)*

*goal\_point = (3, 3)*

*path = astar(grid, start\_point, goal\_point)*

*print(path)*

1. **Output/Result:**

****

1. **Learning Outcomes:**
2. *Record the execution time of the A\* algorithm for each problem scenario.*
3. *Note the number of nodes expanded during the algorithm's execution.*
4. *Record the optimal path generated by the A\* algorithm.*
5. *Evaluate the correctness of the generated path by comparing it with known optimal solutions, if available.*
6. *Analyze the efficiency and effectiveness of the A\* algorithm based on the execution time, number of nodes expanded, and the quality of the generated paths.*
7. *Compare the performance of the A\* algorithm with other algorithms or approaches, if applicable.*