Exp No : 04

Date : 31/08/24

A * SEARCH

Aim :

To find the shortest path from a start node to a goal node using A* search algorithm.

Algorithm :

1) Create open and closed sets, start with the initial state.
2) Add the start node to the open set with an initial cost of 0.
3) Remove the node with the lowest 'f' value (cost + heuristic) from the open set.
4) If the current node is the goal node reconstruct the path.
5) For each neighbor, calculate 'g', 'h', 'f' values.
6) If the neighbour is not in the open set or a lowest cost path is found, update cost and percent.

7) Add the neighbour to the open set if it is not already in the closed set.

8) Repeat until the open set is empty or the goal is found.

Program:

```
import heapq
def a_star(start, goal, h, neighbours):
    open_set = []
    heapq.heappush(open_set, (0+h(start), 0, start))
    came_from = {}
    g_score = {start: 0}
    f_score = {start: h(start)}
    while open_set:
        _, current_g, current = heapq.heappop(open_set)
        if current == goal:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            path.append(start)
            return path[::-1]
```

```
for neighbor in neighbors(current):
    tentative_g = g_score[current]+1
    if neighbor not in g_score or tentative_g <
                    g_score[neighbor]:
        came_from[neighbor] = current
        g_score[neighbor] = tentative_g
        f_score[neighbor] = tentative_g+h(neighbor)
        if neighbor not in [i[2] for i in
                        open_set]:
            heapq = heappush(open_set,
                (f_score[neighbor], tentative_g,
                    neighbor))

return None

def heuristic(node):
    goal_position=(5,5)
    return abs(node[0]-goal_position[0])
            + abs(node[1]-goal_position[1])

def neighbors(node):
    x,y = node
```

```
return [(x+1, y), (x-1, y), (x, y+1), (x, y-1)]

start = (0,0)
goal = (5,5)
path = a_star (start, goal, heuristic, neighbors)
print(path)
```

Output:

[(0,0), (1,0), (2,0), (3,0), (4,0), (5,0) (5,1), (5,2), (5,3), (5,4) (5,5)]

Result:

Thus, the program is executed successfully and output is verified.