Exp. No: 4 A* Search.

Date:

Aim:

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

Algorithm:

Step 01: create open & closed sets: start with the initial node.

Step 02: Add the start node to the open set with an initial cost g0.

Step 03: If they remove the node, with the lowest f value from the open set.

Step 04. If the current ~~beg~~ node is the goal node, reconstruct the path.

Step 05: For each neighbour, calculate g.h & f values.

Step 06: If the neighbour is not in the open set on a lower cost path, is found, update costs & patent.

Step 07: Add the neighbour to the open set if it is not already in the closed set.

Step 08: Repeat until the open set is empty on the goal is found.

Program:

```
import heapq
def a_star(star, 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 neighbour in neighbours (current)
          tentative_g = g-score [current] +1
      if neighbour not in g-score or tentative_g <
         g-score [neighbour]:
            came_from [neighbour] = current
            g-score [neighbour] = tentative_g
            f-score [neighbour] = tentative_g + h(neighbour
     if neighbour not in [i[2] for i in open_set]:
        heapq.heappush(open_set. (f-score [neighbour],
            tentative_g.neigbour))
```

```
return None,

def heuristic (node):
    goal - position = (5,5)
return abs (node[0] - goal.position[0]) +
        abs(node[1] - goal.position[1])
def neighbours (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, neighbour)
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 A* search program is executed and the output is verified successfully.