# A* Algorithm

## Aim:

To implement A* Algorithm

## Code:

```
import heapq

class Node:
    def __init__(self, position, parent=None, g=0, h=0):
        self.position = position
        self.parent = parent
        self.g = g
        self.h = h
        self.f = g + h

    def __eq__(self, other):
        return self.position == other.position

    def __lt__(self, other):
      return self.f < other.f

def astar(maze, start, end):
    rows, cols = len(maze), len(maze[0])
    open_list = []
    closed_set = set()

    start_node = Node(start, g=0, h=heuristic(start, end))
    heapq.heappush(open_list, start_node)

    while open_list:
        current_node = heapq.heappop(open_list)
        if current_node.position == end:
            path = []
            while current_node:
                path.append(current_node.position)
```

```
                current_node = current_node.parent
            return path[::-1]

        closed_set.add(current_node.position)

        for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0)]:
            new_x, new_y = current_node.position[0] + dx,
current_node.position[1] + dy

            if 0 <= new_x < rows and 0 <= new_y < cols and maze[new_x][new_y]
== 0:
                neighbor = (new_x, new_y)

                if neighbor in closed_set:
                    continue
                temp_g = current_node.g + 1
                neighbor_node = Node(neighbor, current_node, temp_g,
heuristic(neighbor, end))

                if neighbor_node not in open_list:
                  heapq.heappush(open_list, neighbor_node)
    return None
def heuristic(a, b):
    return abs(a[0] - b[0]) + abs(a[1] - b[1])
maze = [
    [0, 0, 0, 0, 0],
    [0, 1, 1, 1, 0],
    [0, 0, 0, 0, 0],
    [0, 1, 1, 1, 0],
    [0, 0, 0, 0, 0]
]
start = (0, 0)
end = (4, 4)
path = astar(maze, start, end)
if path:
  print("Path found:", path)
else:
  print("No path found.")
```

## Output:

Path found: [(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 4), (2, 4), (3, 4), (4, 4)]

## Result:

A* Algorithm implemented successfully.