AIM:

The aim is to solve the algorithm program using python programming

PROGRAM:

import heapq

class Node:

def \_\_init\_\_(self, row, col, cost, heuristic):

self.row = row

self.col = col

self.cost = cost

self.heuristic = heuristic

self.parent = None

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

return (self.cost + self.heuristic) < (other.cost + other.heuristic)

def astar(grid, start, goal):

rows, cols = len(grid), len(grid[0])

open\_set = [Node(start[0], start[1], 0, heuristic(start, goal))]

closed\_set = set()

while open\_set:

current\_node = heapq.heappop(open\_set)

if (current\_node.row, current\_node.col) == goal:

path = []

while current\_node:

path.append((current\_node.row, current\_node.col))

current\_node = current\_node.parent

return path[::-1]

closed\_set.add((current\_node.row, current\_node.col))

for neighbor in get\_neighbors(current\_node.row, current\_node.col, rows, cols):

if neighbor not in closed\_set:

neighbor\_row, neighbor\_col = neighbor

cost = current\_node.cost + grid[neighbor\_row][neighbor\_col]

heuristic\_val = heuristic((neighbor\_row, neighbor\_col), goal)

new\_node = Node(neighbor\_row, neighbor\_col, cost, heuristic\_val)

new\_node.parent = current\_node

heapq.heappush(open\_set, new\_node)

return None

def heuristic(a, b):

return abs(a[0] - b[0]) + abs(a[1] - b[1])

def get\_neighbors(row, col, max\_row, max\_col):

neighbors = []

if row > 0:

neighbors.append((row - 1, col))

if row < max\_row - 1:

neighbors.append((row + 1, col))

if col > 0:

neighbors.append((row, col - 1))

if col < max\_col - 1:

neighbors.append((row, col + 1))

return neighbors

grid = [

[1, 3, 1, 2, 4],

[2, 5, 3, 1, 2],

[1, 1, 2, 5, 3],

[3, 2, 4, 1, 4],

[2, 4, 2, 3, 3]

]

start\_node = (0, 0)

goal\_node = (4, 4)

result = astar(grid, start\_node, goal\_node)

if result:

print("Shortest path:", result)

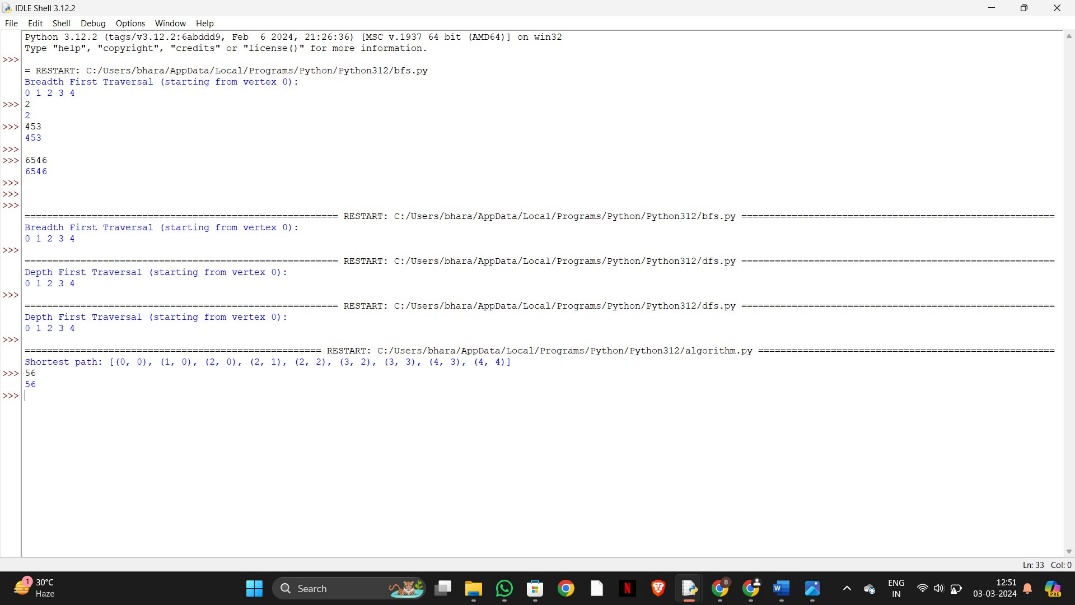
else:

print("No path found.")

INPUT:

Shortest path = …..

OUTPUT:



RESULT:

To solve the algorithm by using python program is successfully completed