**AI FOR LLM- CSA1704**

**10. A\* (A-star)**

**CODE:**

import heapq

class Node:

def \_\_init\_\_(self, position, parent=None, g=0, h=0):

self.position = position

self.parent = parent

self.g = g # Cost from start to current

self.h = h # Heuristic cost to goal

self.f = g + h # Total cost

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

return self.f < other.f

def heuristic(a, b):

# Manhattan distance

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

def astar(grid, start, goal):

open\_list = []

closed\_set = set()

start\_node = Node(start, None, 0, heuristic(start, goal))

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

while open\_list:

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

if current\_node.position == goal:

# Reconstruct path

path = []

while current\_node:

path.append(current\_node.position)

current\_node = current\_node.parent

return path[::-1]

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

x, y = current\_node.position

neighbors = [(x+1,y), (x-1,y), (x,y+1), (x,y-1)]

for nx, ny in neighbors:

if 0 <= nx < len(grid) and 0 <= ny < len(grid[0]) and grid[nx][ny] == 0:

neighbor\_pos = (nx, ny)

if neighbor\_pos in closed\_set:

continue

g = current\_node.g + 1

h = heuristic(neighbor\_pos, goal)

neighbor\_node = Node(neighbor\_pos, current\_node, g, h)

heapq.heappush(open\_list, neighbor\_node)

return None # No path found

# Example grid (0 = free, 1 = obstacle)

grid = [

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

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

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

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

[0, 0, 0, 1, 0]

]

start = (0, 0)

goal = (4, 4)

path = astar(grid, start, goal)

# Display result

if path:

print("Path found:")

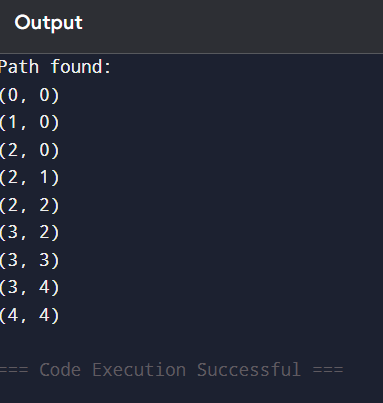
for step in path:

print(step)

else:

print("No path found.")

**OUTPUT:**

****