**NAME : TARRUN .B**

**CLASS & SEC : AI&DS – ‘FD’**

**Reg No : 241801291**

**Ex No : 4**

**Ex Name : A\* Search**

**Date : 26.04.2025**

**Program :** 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 \_\_lt\_\_(self, other): return self.f < other.f

def heuristic(a, b):

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

def a\_star(grid, start, goal):

rows, cols = len(grid), len(grid[0]) open\_list = [] heapq.heappush(open\_list, Node(start, None, 0, heuristic(start, goal))) closed\_set = set()

while open\_list:

current\_node = heapq.heappop(open\_list) if current\_node.position == goal: path = [] while current\_node:

path.append(current\_node.position) current\_node = current\_node.parent return path[::-1]

closed\_set.add(current\_node.position) for dr, dc in [(-1, 0), (1, 0), (0, -1), (0, 1)]:

new\_pos = (current\_node.position[0] + dr, current\_node.position[1] + dc) if (0 <= new\_pos[0] < rows and 0 <= new\_pos[1] < cols and grid[new\_pos[0]][new\_pos[1]] == 0 and new\_pos not in closed\_set):

new\_node = Node(new\_pos, current\_node, current\_node.g + 1, heuristic(new\_pos, goal))

heapq.heappush(open\_list, new\_node) return None

warehouse\_grid = [

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

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

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

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

[0, 0, 0, 0, 0]

]

start\_position = (0, 0) goal\_position = (4, 4) path = a\_star(warehouse\_grid, start\_position, goal\_position) print("Optimal Path:", path)

**Output :**

