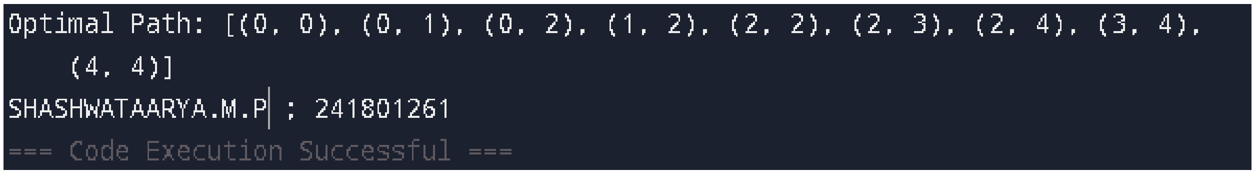
A* SEARCH ALGORITHM

NAME : SHASHWATAARYA.M.P **ROLL NO :** 241801261

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:



```
Optimal Path: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 3), (2, 4), (3, 4),
    (4, 4)]
SHASHWATAARYA.M.P ; 241801261
=== Code Execution Successful ===
```