

NAME: SIDDHARTH R

REGISTER NO: 241801269

EXP NO: 4

EXP NAME: IMPLEMENTATION OF A* SEARCH

PROGRAM:

```
import heapq
```

```
# Define the grid and movements
```

```
class Node:
```

```
    def __init__(self, position, parent=None, g=0, h=0):
```

```
        self.position = position # (row, col)
```

```
        self.parent = parent    # Parent node
```

```
        self.g = g              # Cost from start node
```

```
        self.h = h              # Heuristic cost to goal
```

```
        self.f = g + h          # Total cost
```

```
    def __lt__(self, other):
```

```
        return self.f < other.f # Priority queue comparison
```

```
def heuristic(a, b):
```

```
    return abs(a[0] - b[0]) + abs(a[1] - b[1]) # Manhattan Distance
```

```
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) # Get node with lowest f-value

    if current_node.position == goal:

        path = []

        while current_node:

            path.append(current_node.position)

            current_node = current_node.parent

        return path[::-1] # Return reversed path

    closed_set.add(current_node.position)

    for dr, dc in [(-1, 0), (1, 0), (0, -1), (0, 1)]: # Possible moves

        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 # No path found

```

```
# Example grid: 0 = free space, 1 = obstacle
```

```
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]
```

```
]
```

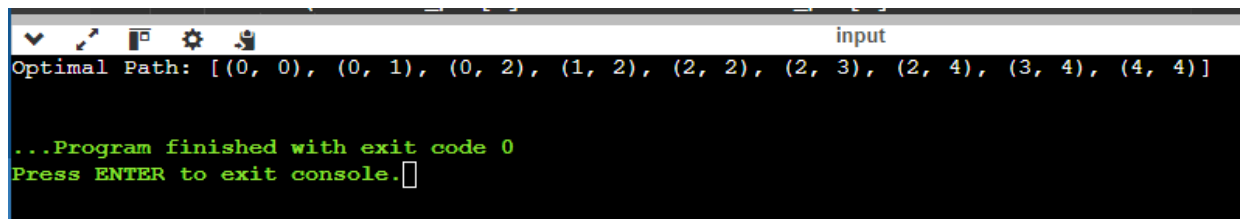
```
start_position = (0, 0)
```

```
goal_position = (4, 4)
```

```
path = a_star(warehouse_grid, start_position, goal_position)
```

```
print("Optimal Path:", path)
```

OUTPUT

A screenshot of a terminal window with a dark background. The title bar at the top shows standard window controls and the word "input". The terminal displays the output of a program: "Optimal Path: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 3), (2, 4), (3, 4), (4, 4)]". Below this, it shows "...Program finished with exit code 0" and "Press ENTER to exit console." with a cursor. The text is in a monospaced font, with the first line in red and the subsequent lines in green.

```
input
Optimal Path: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 3), (2, 4), (3, 4), (4, 4)]
...Program finished with exit code 0
Press ENTER to exit console.
```