

## PROGRAM-11

### A\* ALGORITHM PROBLEM

#### AIM:-

To write and execute the python program for the A\* algorithm program.

#### PROCEDURE:-

- **Inputs:**
  - graph: Dictionary representing the graph where keys are nodes and values are dictionaries containing neighboring nodes and their associated costs.
  - start: Start node.
  - goal: Goal node.
- **Variables:**
  - frontier: Priority queue (heap) containing nodes to be explored, ordered by their estimated total cost (f-score).
  - came\_from: Dictionary that maps each explored node to the node it was reached from.
  - cost\_so\_far: Dictionary that maps each node to the cost of reaching that node from the start node.
- **Algorithm:**
  - The algorithm iterates over the frontier until it becomes empty.
  - After finding the goal node, it reconstructs the path from the start node to the goal node using the came\_from dictionary.
- **Main section:**
  - Defines a sample graph represented as a dictionary.
  - Specifies the start and goal nodes.
  - Calls the astar function with the provided graph, start, and goal nodes, and prints the resulting path.

#### CODING:-

```
import heapq
```

```
def heuristic(node, goal):
```

```

    return abs(node[0] - goal[0]) + abs(node[1] - goal[1])

def astar(graph, start, goal):

    frontier, came_from, cost_so_far = [(0, start)], {start: None}, {start: 0}

    while frontier:

        _, current = heapq.heappop(frontier)

        if current == goal: break

        for next_node in graph[current]:

            new_cost = cost_so_far[current] + graph[current][next_node]

            if next_node not in cost_so_far or new_cost < cost_so_far[next_node]:

                cost_so_far[next_node] = new_cost

                heapq.heappush(frontier, (new_cost + heuristic(next_node, goal), next_node))

                came_from[next_node] = current

    path, current = [], goal

    while current != start: path.append(current); current = came_from[current]

    return path + [start]

if __name__ == "__main__":

    graph = {(0, 0): {(0, 1): 1, (1, 0): 1}, (0, 1): {(0, 0): 1, (1, 1): 1},

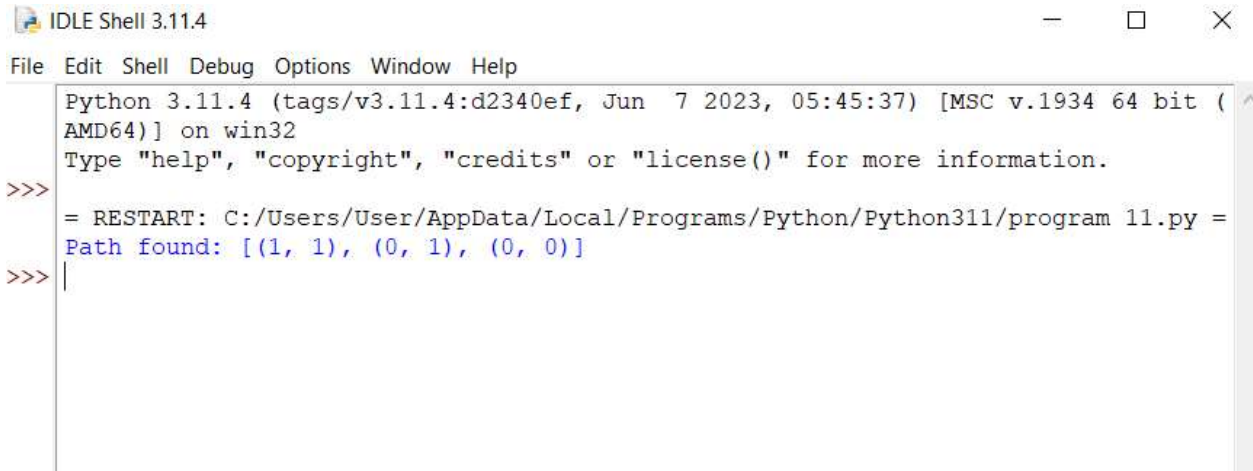
            (1, 0): {(0, 0): 1, (1, 1): 1}, (1, 1): {(0, 1): 1, (1, 0): 1}}

    start, goal = (0, 0), (1, 1)

    print("Path found:", astar(graph, start, goal))

```

**OUTPUT:-**



```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/User/AppData/Local/Programs/Python/Python311/program 11.py =
Path found: [(1, 1), (0, 1), (0, 0)]
>>> |
```

## RESULT:-

Hence the program has been successfully executed and verified.