Name: Faraz Pathan

Rollno: COTB 70

**Program:**

class Graph:

    def \_\_init\_\_(self):

        self.graph = {}

    def add\_edge(self, u, v, w):

        if u in self.graph:

            self.graph[u].append((v, w))

        else:

            self.graph[u] = [(v, w)]

    def astar(self, start, goal):

        open\_set = [(0, start)]

        came\_from = {}

        g\_score = {vertex: float('inf') for vertex in self.graph}

        g\_score[start] = 0

        f\_score = {vertex: float('inf') for vertex in self.graph}

        f\_score[start] = self.heuristic(start, goal)

        while open\_set:

            current\_f\_score, current\_vertex = min(open\_set)

            open\_set.remove((current\_f\_score, current\_vertex))

            if current\_vertex == goal:

                return self.reconstruct\_path(came\_from, goal)

            for neighbor, weight in self.graph.get(current\_vertex, []):

                tentative\_g\_score = g\_score[current\_vertex] + weight

                if tentative\_g\_score < g\_score.get(neighbor, float('inf')):

                    came\_from[neighbor] = current\_vertex

                    g\_score[neighbor] = tentative\_g\_score

                    f\_score[neighbor] = tentative\_g\_score + self.heuristic(neighbor, goal)

                    open\_set.append((f\_score[neighbor], neighbor))

        return None

    def heuristic(self, current, goal):

        # This heuristic function can be replaced with any appropriate

        # heuristic function, such as Euclidean distance or Manhattan distance.

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

    def reconstruct\_path(self, came\_from, current):

        total\_path = [current]

        while current in came\_from:

            current = came\_from[current]

            total\_path.append(current)

        return total\_path[::-1]

# Example usage:

g = Graph()

g.add\_edge((0, 0), (0, 1), 1)

g.add\_edge((0, 0), (1, 0), 1)

g.add\_edge((0, 1), (1, 1), 1)

g.add\_edge((1, 0), (1, 1), 1)

g.add\_edge((1, 0), (2, 0), 1)

g.add\_edge((1, 1), (2, 1), 1)

g.add\_edge((2, 0), (2, 1), 1)

g.add\_edge((2, 0), (3, 0), 1)

g.add\_edge((2, 1), (3, 1), 1)

start = (0, 0)

goal = (3, 1)

path = g.astar(start, goal)

print("A\* Path from", start, "to", goal, ":", path)

**OUTPUT:**

A\* Path from (0, 0) to (3, 1) : [(0, 0), (0, 1), (1, 1), (2, 1), (3, 1)]