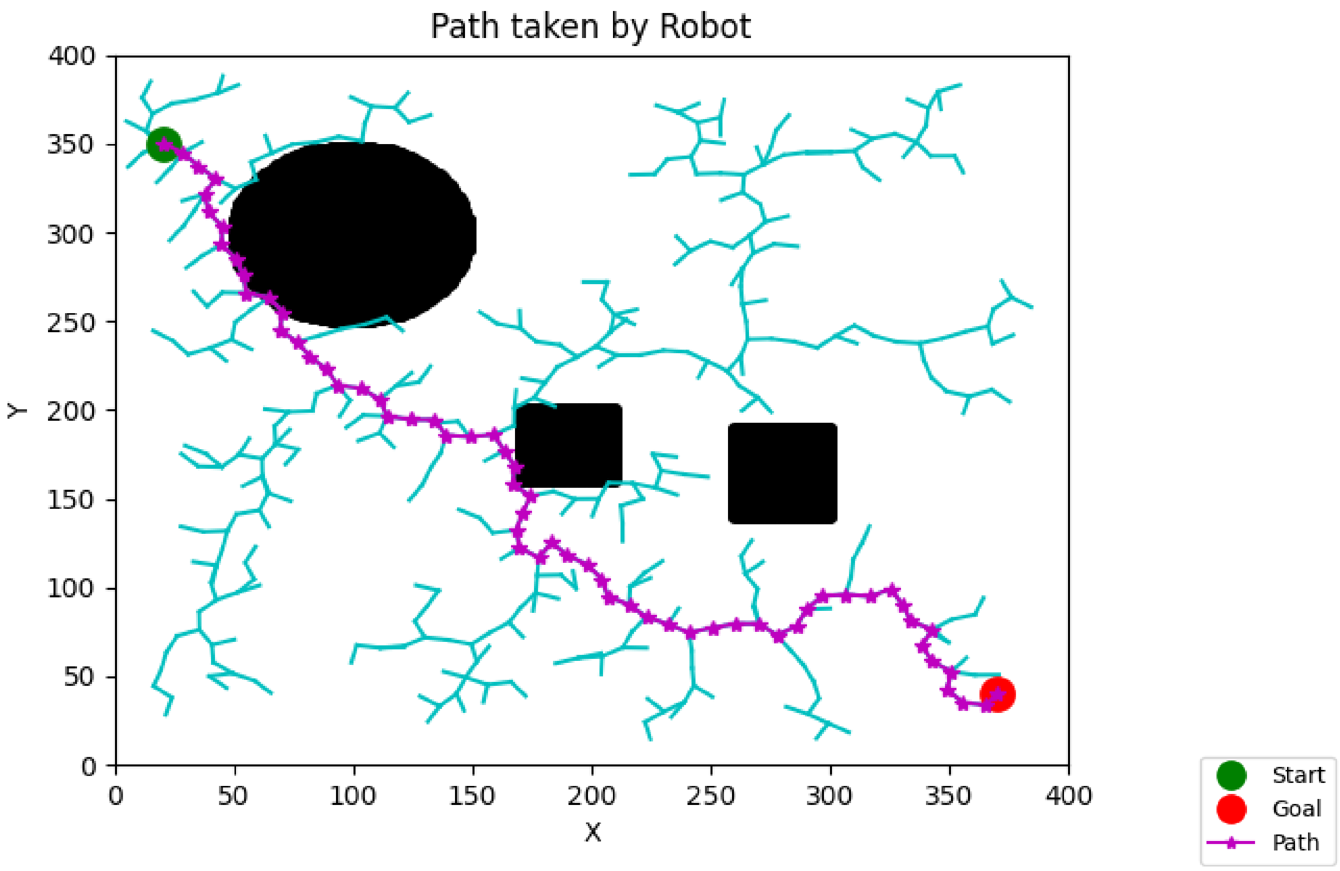
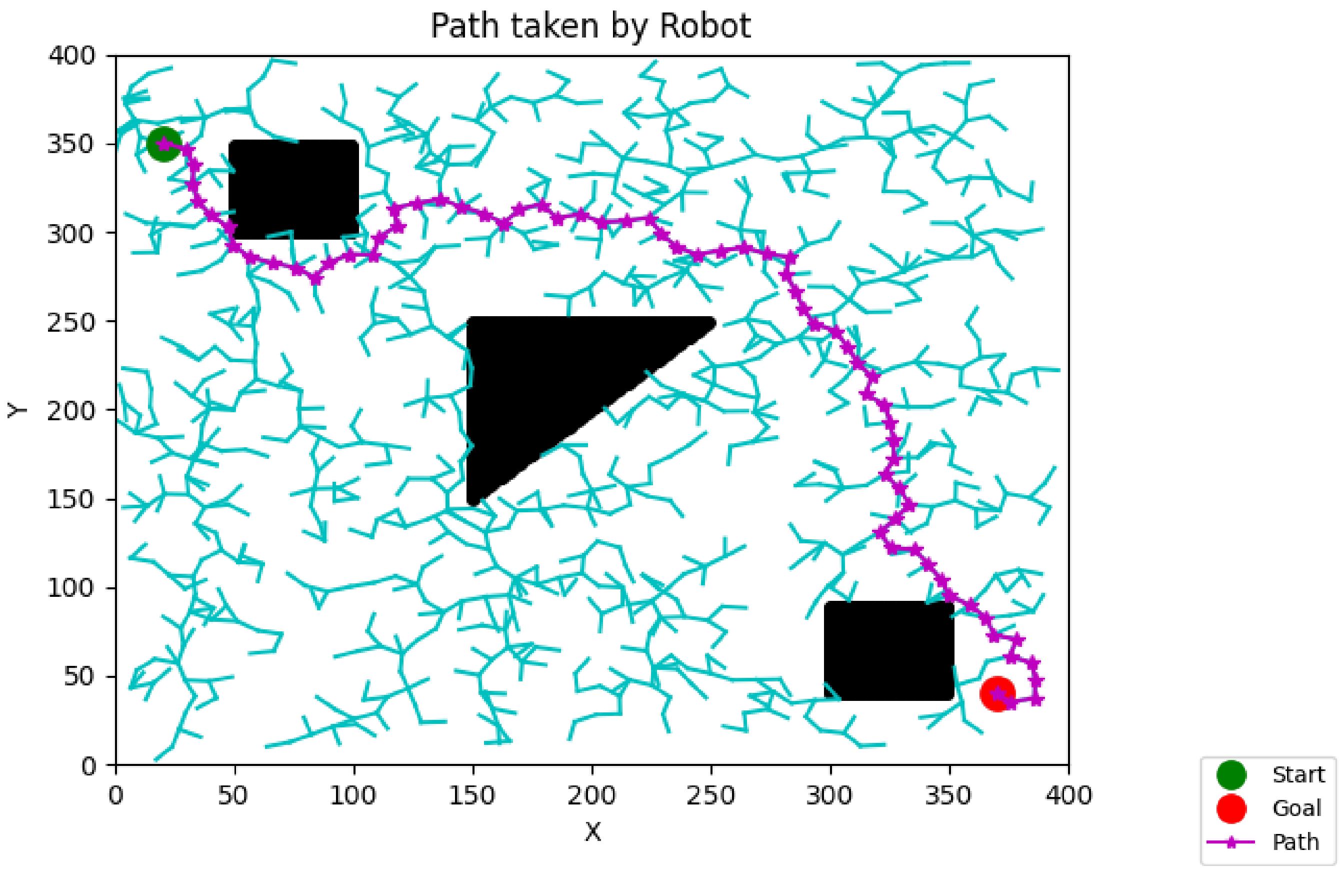
```
import numpy as np                                                    # Submitted by- Sourit Saha, 200998
import matplotlib.pyplot as plt
from shapely.geometry import LineString

class Node:
    def __init__(self, point, parent=None):
        self.point = point
        self.parent = parent

def solveforRRT(start, goal, obstacles, size_x, size_y, max_iter=10000, step_size=10):
    start_node = Node(start)
    goal_node = Node(goal)

    nodes = [start_node]
    tree=[]

    for i in range(max_iter):
        # Sample a random point in the space
        point = np.array([np.random.randint(size_x), np.random.randint(size_y)])

        # Find the nearest node to the sampled point
        nearest_node = None
        min_dist = float('inf')
        for node in nodes:
            dist = np.linalg.norm(node.point - point)
            if dist < min_dist:
                nearest_node = node
                min_dist = dist

        # Compute the new point by moving towards the sampled point
        direction = point - nearest_node.point
        direction = step_size * direction / np.linalg.norm(direction)
        new_point = nearest_node.point + direction

        # Check if the new point is outside the workspace
        if new_point[0] < 0 or new_point[0] >= size_x or new_point[1] < 0 or new_point[1] >= size_y:
            continue

        # Check if the new point collides with any obstacles
        if obstacles[int(new_point[0]), int(new_point[1])] == 1:
            continue

        # Create a new node for the new point and add it to the tree
        new_node = Node(new_point, parent=nearest_node)
        nodes.append(new_node)

        # Generate the tree
        p1=new_node.point
        p2=nearest_node.point
        line=LineString([p1, p2])
        tree.append(line)

        # Check if the new node is close enough to the goal
        if np.linalg.norm(new_point - goal_node.point) < step_size:
            goal_node.parent = new_node
            nodes.append(goal_node)
            break

    # Extract the path from the tree

    # If path not found
    if goal_node.parent is None:
        return None, np.array(tree)

    # If path is found
    path = []
    node = goal_node
    while node is not None:
        path.append(node.point)
        node = node.parent
    path.reverse()
    return np.array(path), np.array(tree)


# generating workspace
size_x = 400
size_y = 400
x, y = np.meshgrid(np.arange(1, size_x + 1), np.arange(1, size_y + 1))

# defining start and goal points
start = np.array([20, 350])
goal = np.array([370, 40])

# defining obstacles
obstacles = np.zeros((size_x, size_y))

#obstacles where path is found, Set 1:
obstacles[260:301, 140:191] = 1
obstacles[170:211, 160:201] = 1
t = ((x - 300)**2 + (y - 100)**2) < 50**2
obstacles[t] = 1

#obstacles where path is found, Set 2:
'''obstacles[50:100, 300:350] = 1
obstacles[300:350, 40:90] = 1
for i in range(150,250):
    for j in range(150,250):
        if i<=j:
            obstacles[i,j] = 1'''

#obstacles where path is found, Set 3:
'''t = ((x - 300)**2 + (y - 100)**2) < 50**2
obstacles[t] = 1
obstacles[260:350, 100:200] = 1
for i in range(150,250):
    for j in range(150,250):
        if i<=j:
            obstacles[i,j] = 1'''

#obstacles where path is found, Set 4:
'''obstacles[50:70, 50:390] = 1
obstacles[100:120, 10:350] = 1
obstacles[150:170, 50:390] = 1
obstacles[200:220, 10:350] = 1
obstacles[250:270, 50:390] = 1
obstacles[300:320, 10:350] = 1
obstacles[340:360, 50:390] = 1'''

#obstacles where path not found:
'''obstacles[50:70, 50:400] = 1
obstacles[100:120, 0:350] = 1
obstacles[150:170, 50:400] = 1
obstacles[200:220, 0:350] = 1
obstacles[250:270, 50:400] = 1
obstacles[300:320, 0:350] = 1
obstacles[340:360, 50:390] = 1'''

#solving for RRT
path,tree = solveforRRT(start, goal, obstacles, size_x, size_y)

#plotting obstacles
points = np.argwhere(obstacles == 1)
plt.plot(points[:, 0], points[:, 1], 'k.', markersize=8,)

#plotting start and goal points
plt.plot(start[0], start[1], 'g.', markersize=25)
plt.plot(goal[0], goal[1], 'r.', markersize=25)

#plotting tree
for i in range(len(tree)):
    x,y=tree[i].xy
    plt.plot(x,y,'c')
```

```
#plotting path traced by robot
if path is None:
    plt.title('Failed Case. Path not found')
else:
    plt.plot(path[:, 0], path[:, 1], 'm-*')
    plt.title('Path taken by Robot')

plt.xlim([0, size_x])
plt.ylim([0, size_y])
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

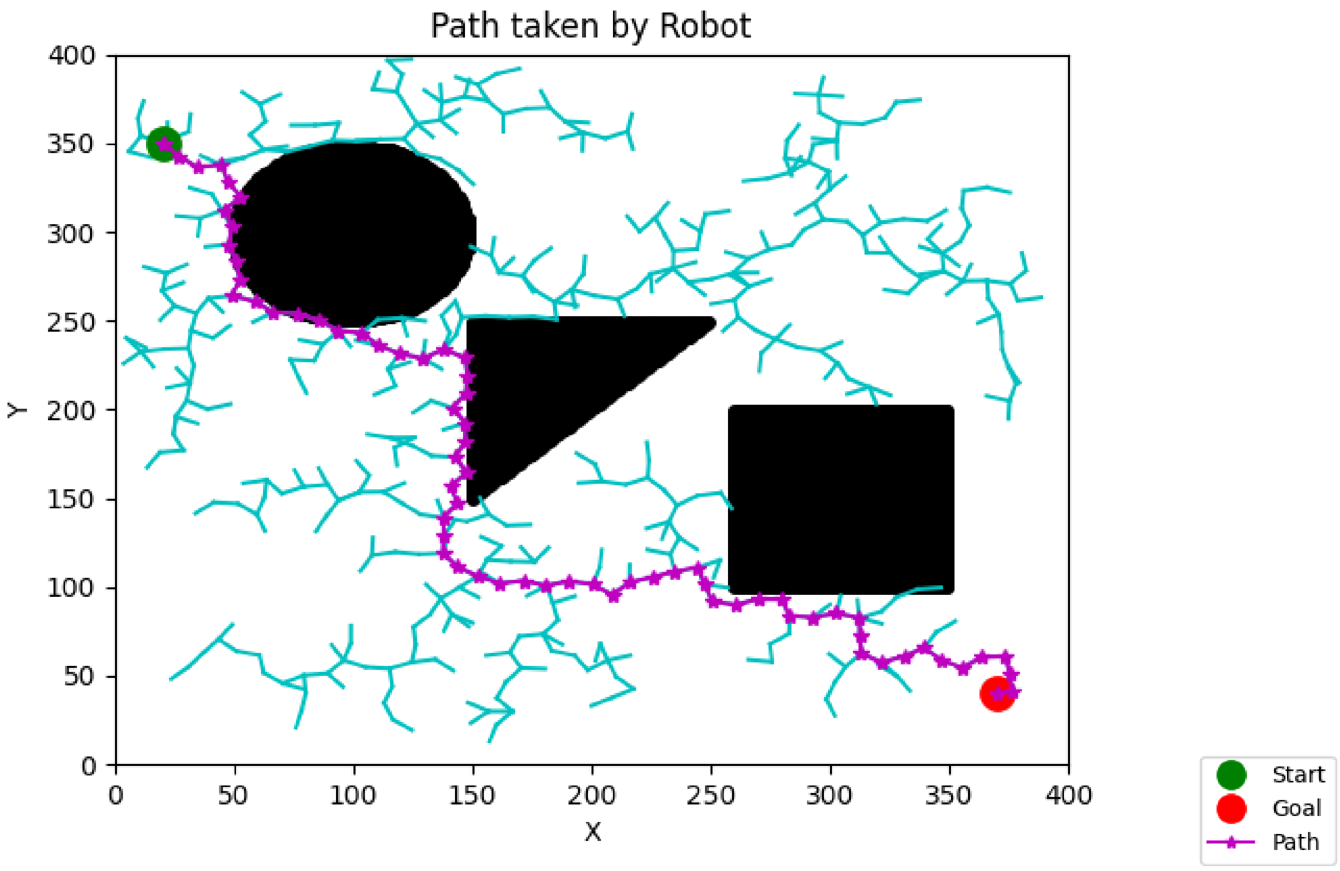
- Path Traced by Robot for Obstacle Set 1:



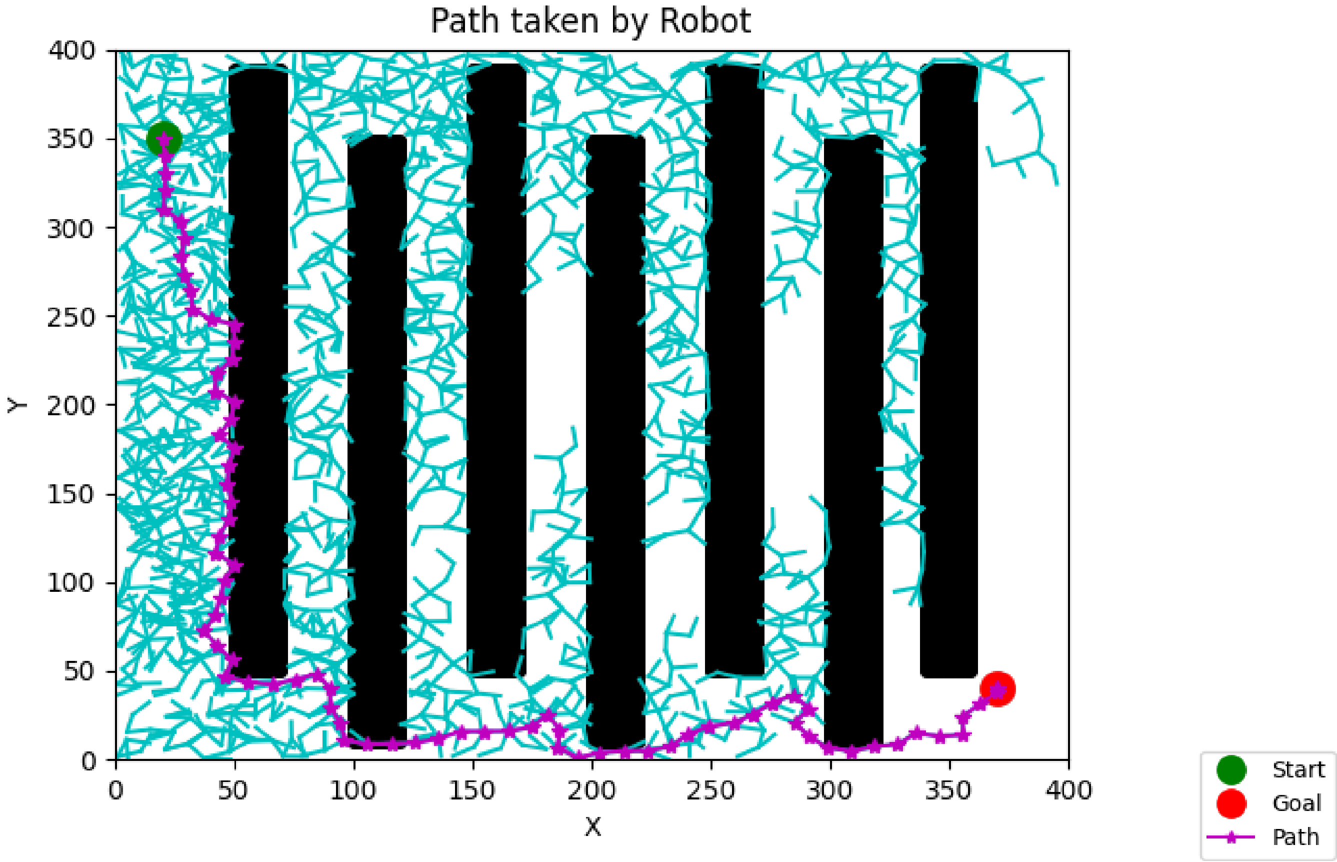
- Path Traced by Robot for Obstacle Set 1:



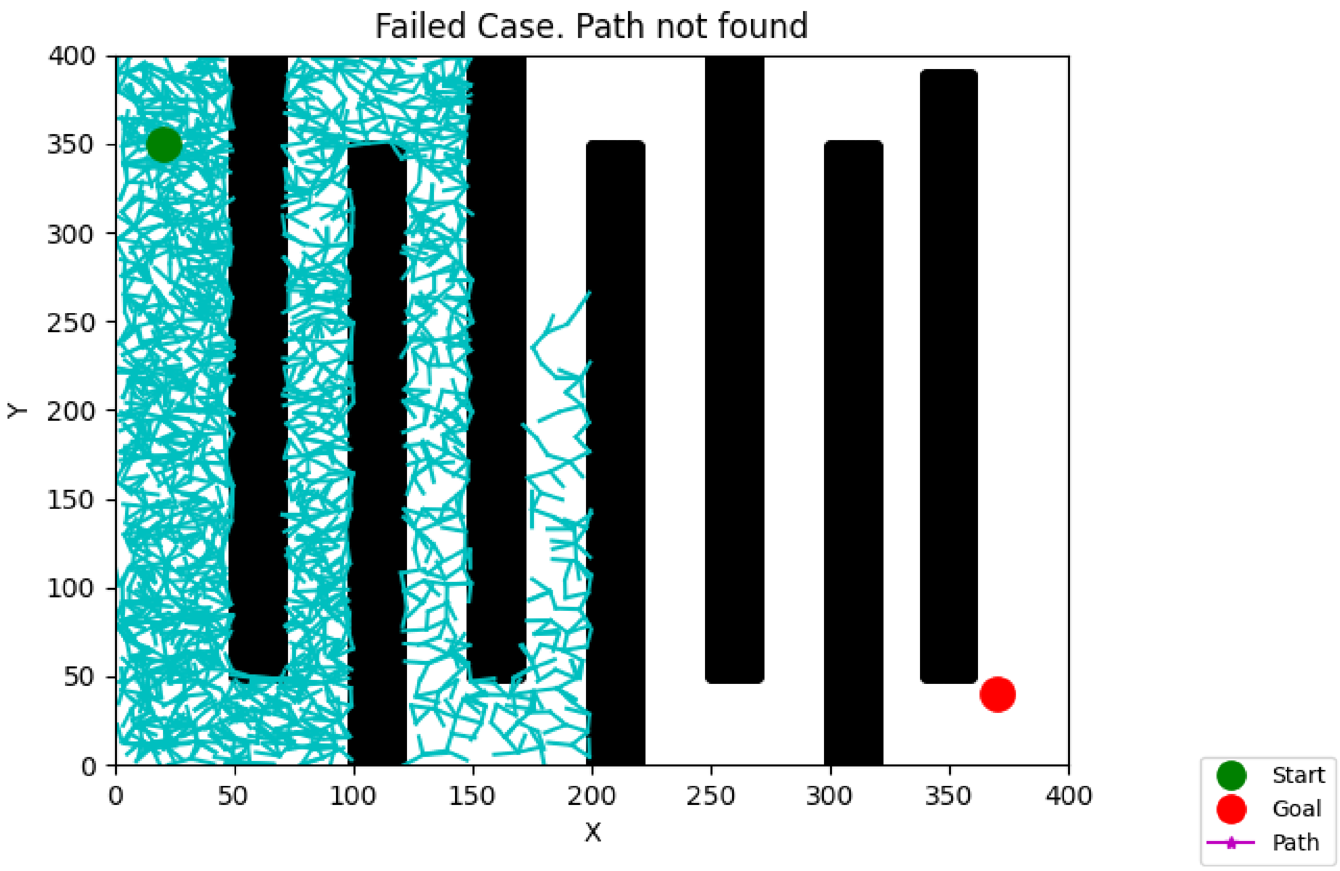
- Path Traced by Robot for Obstacle Set 1:



- Path Traced by Robot for Obstacle Set 1:



- Failed Case, Path not Found:


Failed Case. Path not found
Y
X
Start
Goal
Path