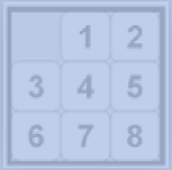
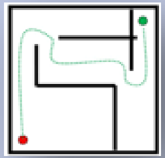
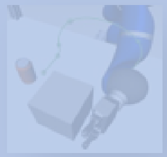
# Project 2: Implementation of Dijkstra algorithm for a Point and Rigid Robot

This is a group project, done in teams of 2. Groups will be announced separately.

Due Date – March 6th, 11.59 PM

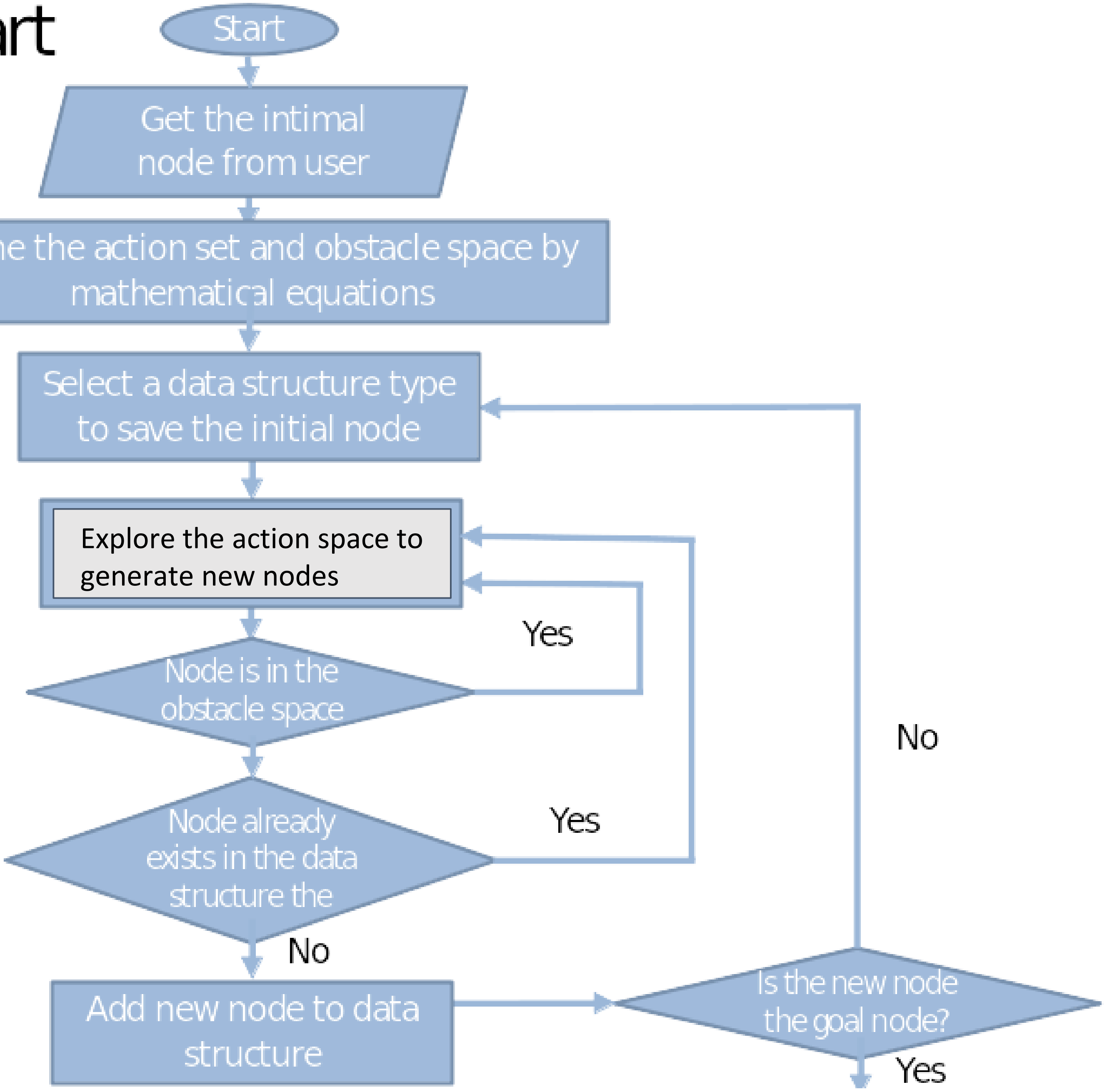



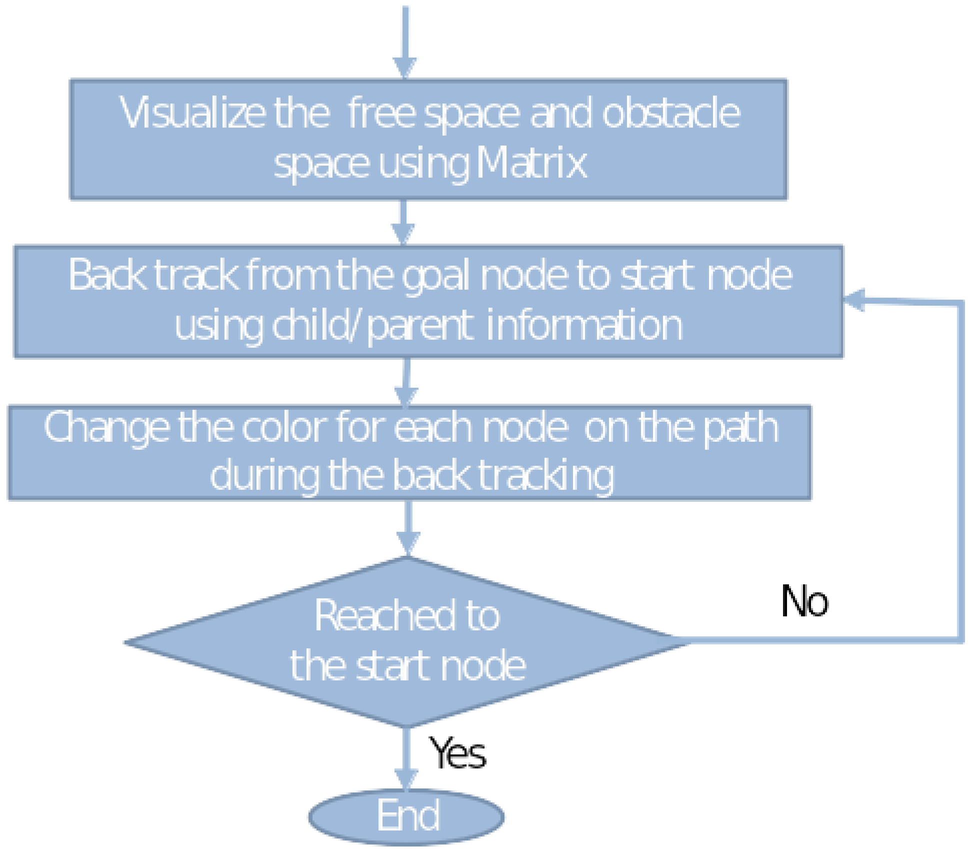


| | Project 1<br>8-puzzle problem | Project 2<br>Point robot | Project 3<br>Mobile robots | Project 4<br>Manipulators | Project 5<br>Final project |
|---|---|---|---|---|---|
| Step 1 | Define actions set & Generating tree | Define actions set & Generating tree | Define actions set & Generating tree | Learn V-rep and Move it/Gazebo | Define actions set & Generating tree |
| Step 2 | Implement simple Search algorithm* | Implement simple Search algorithm | Implement A* Search algorithm | Use V-REP and MOVE IT for path planning | Implement RRT and A* for your application |
| Step 3 | | | Implement:<br>1) nonholonomic cons.<br>2) Obstacle avoidance | Use these modules:<br>1) Inverse kinematics<br>2) Collision detection | Suggest incremental innovations for published papers |
| | | | | Implement project 3 or 4 on a real robot | Write a paper in IEEE format |
| Step 5 | | | | Robotic Competition* | 15- minute Presentation |
| Dates | MM/DD | MM/DD | MM/DD | MM/DD | MM/DD |
| Deliverables | - Source code<br>- Output file | - Source code<br>- Output file | - Source code<br>- Output file<br>- Visualization of the path | - Environment<br>- Video file | - Source code<br>- Output file<br>- Paper<br>- PPT file |

*Optional

# Flow chart


Start
Get the intimal node from user
Define the action set and obstacle space by mathematical equations
Select a data structure type to save the initial node
Explore the action space to generate new nodes
Node is in the obstacle space
Yes
Node already exists in the data structure the
Yes
No
Add new node to data structure
Is the new node the goal node?
No
Yes


Visualize the free space and obstacle space using Matrix
Back track from the goal node to start node using child/parent information
Change the color for each node on the path during the back tracking
Reached to the start node
No
Yes
End

# Project 2 Description

Your code must take following inputs from the user:

| Point Robot | Rigid Robot |
|---|---|
| 1. Start Point | 1. Start point |
| 2. Goal Point | 2. Goal point |
| | 3. Robot radius |
| | 4. Clearance |

**Project Assumption:** Workspace is a 8 connected space, that means now you can move the robot in up, down, left, right & diagonally between up-left, up-right, down-left and down-right directions.

## Project 2 Description

1) Check the feasibility of all inputs/outputs (if user gives start and goal nodes that are in the obstacle space they should be informed by a message and they should try again).

2) Implement Dijkstra’s Algorithm to find a path between start and end point on a given map for a point robot (radius = 0; clearance = 0) and for a rigid robot (radius = r; clearance = c)

3) Your code must output an animation of optimal path generation between start and goal point on the map. You need to show both the node exploration as well as the optimal path generated. (Some useful tools for simulation are OpenCV/Pygame/Matplotlib).

## Step 0) Practice BFS on Trial map

- Follow a similar approach as used in Project 1, to explore the nodes on the given obstacle map.
- Use the 8-action space method as described in the following slides.

This exercise is for your reference. You do not need to submit any codes for this. This is not graded.

## Step 1) Define the actions in a mathematical format

- Use can use the same data structure from project 1 to store the node information. You should also store the cost2come for each node.
- Write 8 subfunctions, one for each action. The output of each subfunction is the state of a new node after taking the associated action.

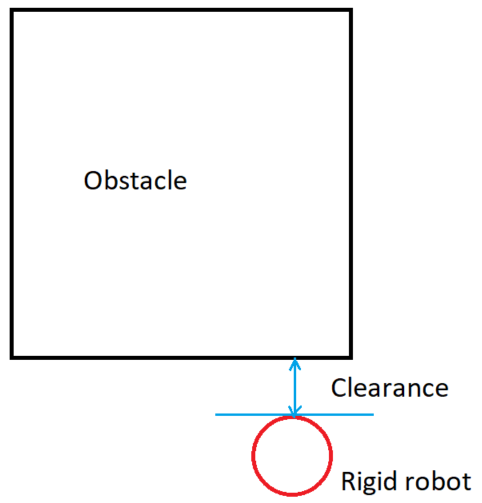
## Step 2) Find mathematical representation of free space

- Use Half planes and semi-algebraic models to represent the obstacle space. (Read Chapter 3: Geometric Representations and Transformations from Planning Algorithms by Steven M. LaValle)

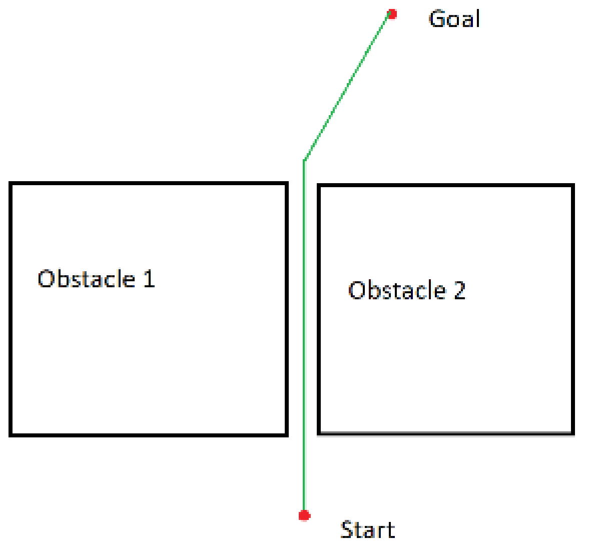
- Professor will cover this topic during the next class in details.

## Clearance

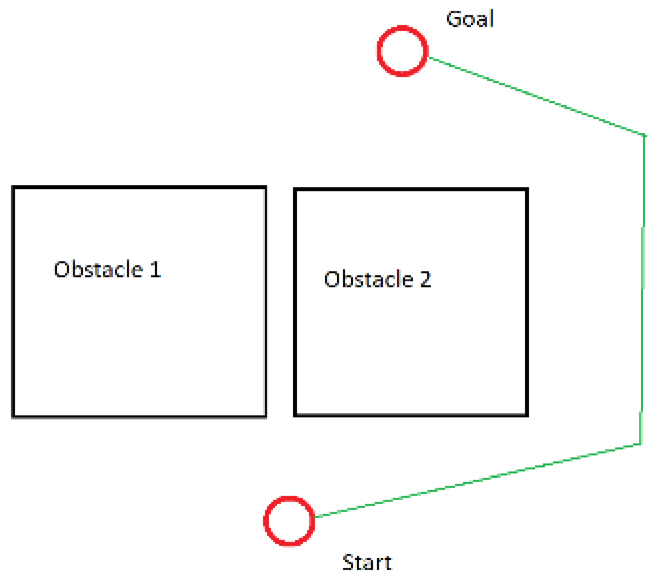
- Clearance is a maximum distance between the obstacle and the extreme point of the rigid robot.



# Difference between point and rigid robot



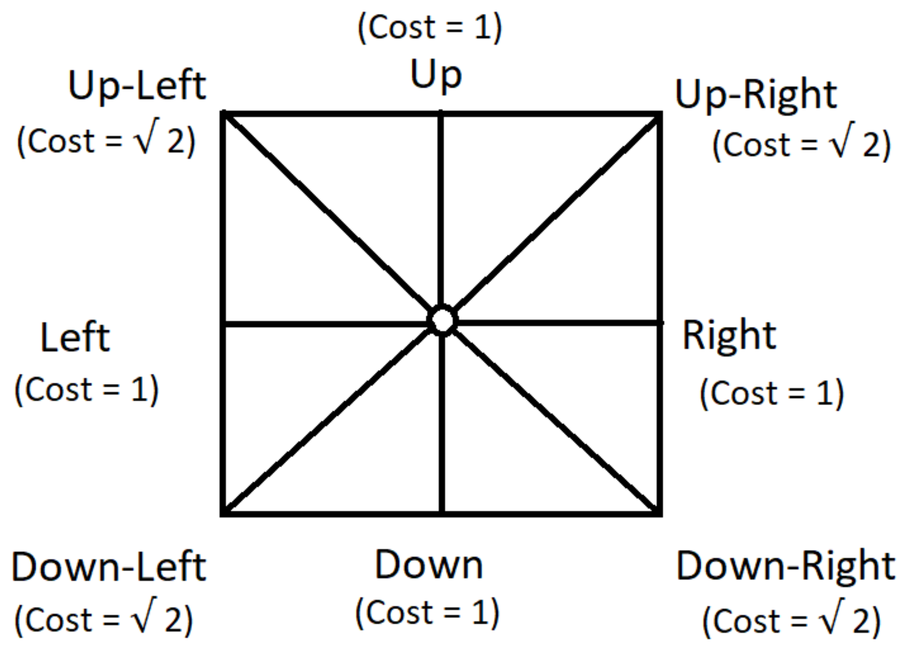
Navigation scenario for point robot



Navigation scenario for rigid robot

# Step 3) Generate the graph

- Generate the graph using action set for a 8-connected space and save in a data structure format
- Before saving the nodes, check for the nodes that are within the obstacle space and ignore them
- Cost of moving diagonally is $\sqrt{2}$.



## Step 4) Find the optimal path

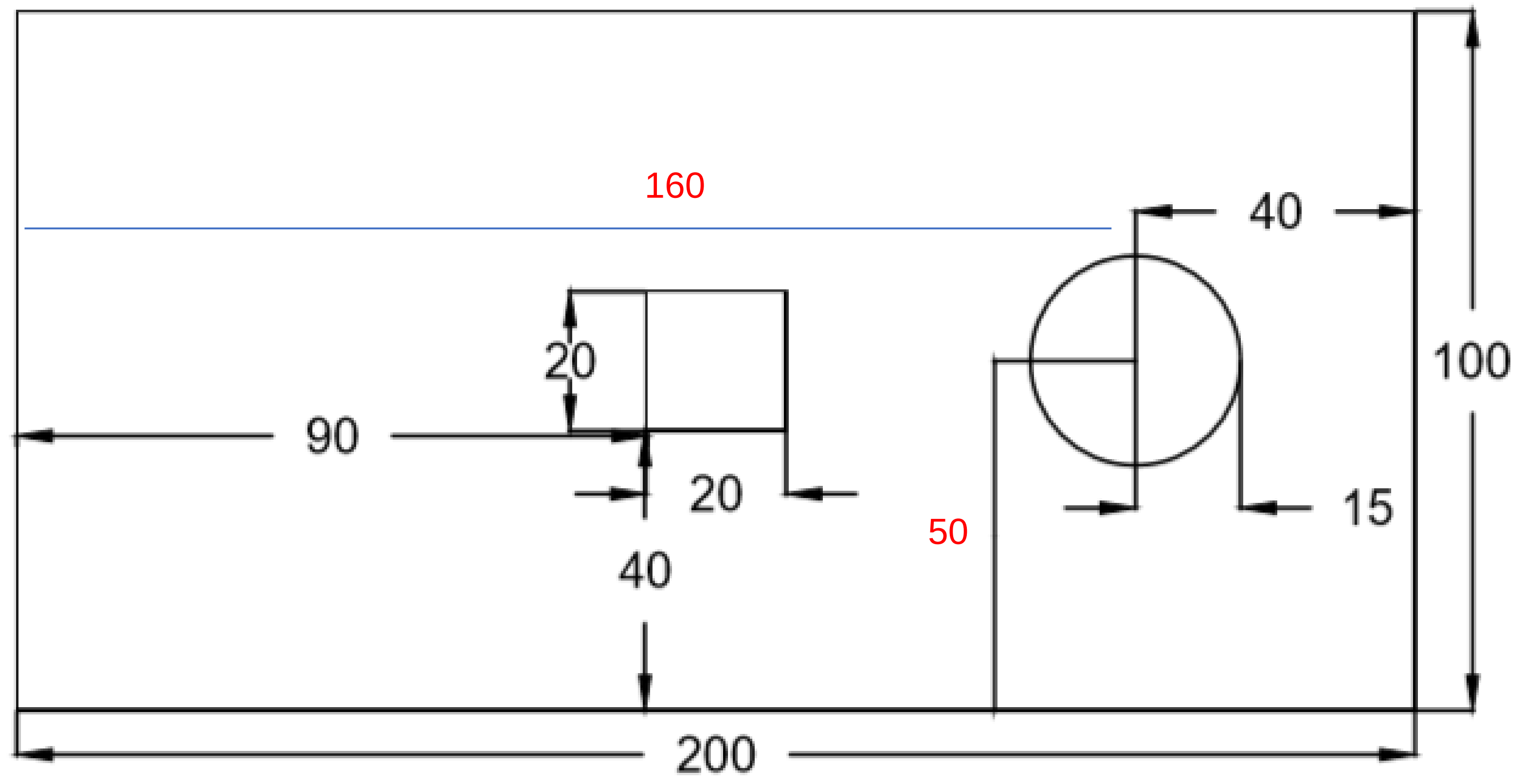
- Write a subfunction that compares the current node with the goal node and return TRUE if they are equal.
- While generating each new node this subfunction should be called
- Write a subfunction that once the goal node is reached, using the child and parent relationship, it backtracks from the goal node to initial node and outputs all the intermediate nodes.

## Step 5) Represent the optimal path

- Show optimal path generation animation between start and goal point using a simple graphical interface. You need to show both the node exploration as well as the optimal path generated.

***<u>The visualisation of (exploration and optimal path) should start only after the exploration is complete and optimal path is found.</u>***

# Trial Map

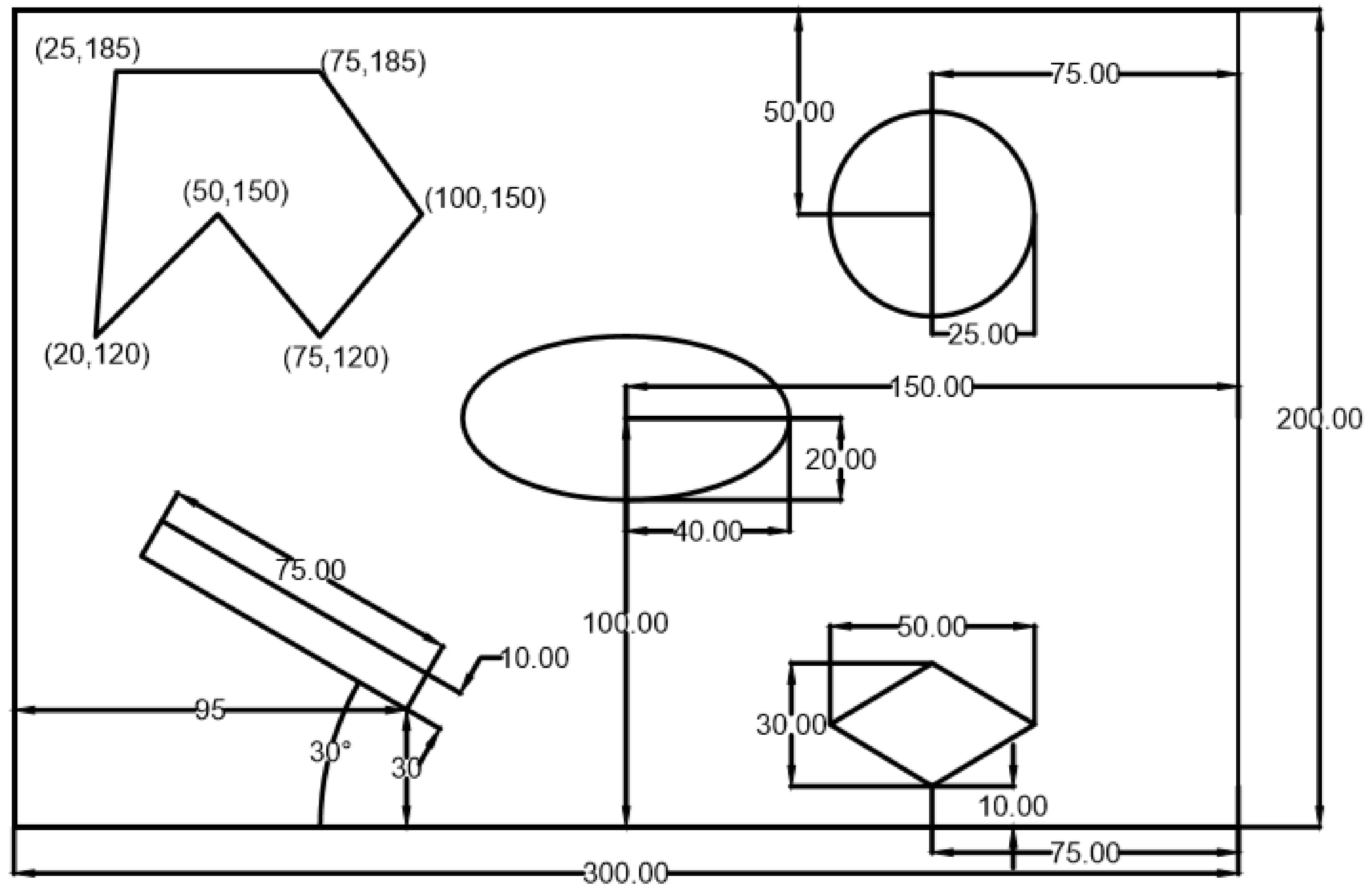


How to define obstacle:

for square: point (x,y) is in the obstacle space if x>=90 and x<= 110 and y>=40 and y<=60

for circle: point (x,y) is in the obstacle space if (x-160)^2+(y-50)^2 < 15^2

# Final Map (towards submission)



## Deliverables

Deliverables:

1. ReadMe.txt (Describing how to run the code in a txt format)
2. Source files for
   I. Dijkstra_point.py
   II. Dijkstra_rigid.py

3. GitHub repository link in the URL submission box (One repository link with commits from both team members)

## Submission Details

- You are required to submit a zip file with the file structure as shown

proj2_groupnumber_codingLanguage

— codes

— readme.txt