CE3004/CZ3004/CPE279/CSC279 MULTIDISCIPLINARY DESIGN PROJECT

Algorithms Briefing

Huang Shell Ying

MDP TASK

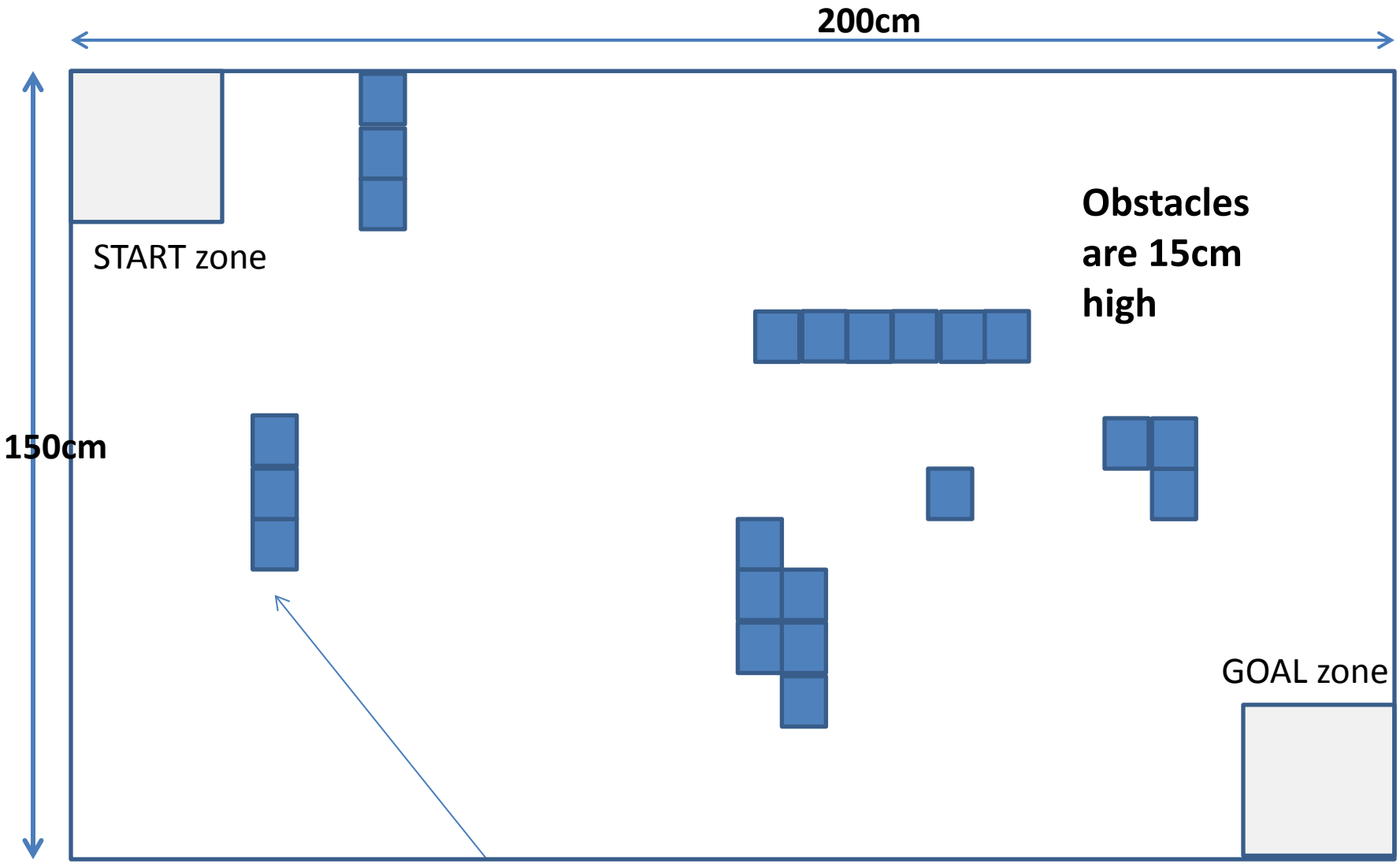
Build a robotic system that can

- **Autonomously explore + traverse an unknown area, avoiding obstacles**
- **Plan and follow fastest path from X to Y**
- Transmit telemetry + receive control signals from mobile device
- **Simulate physical robot and algorithms in software**

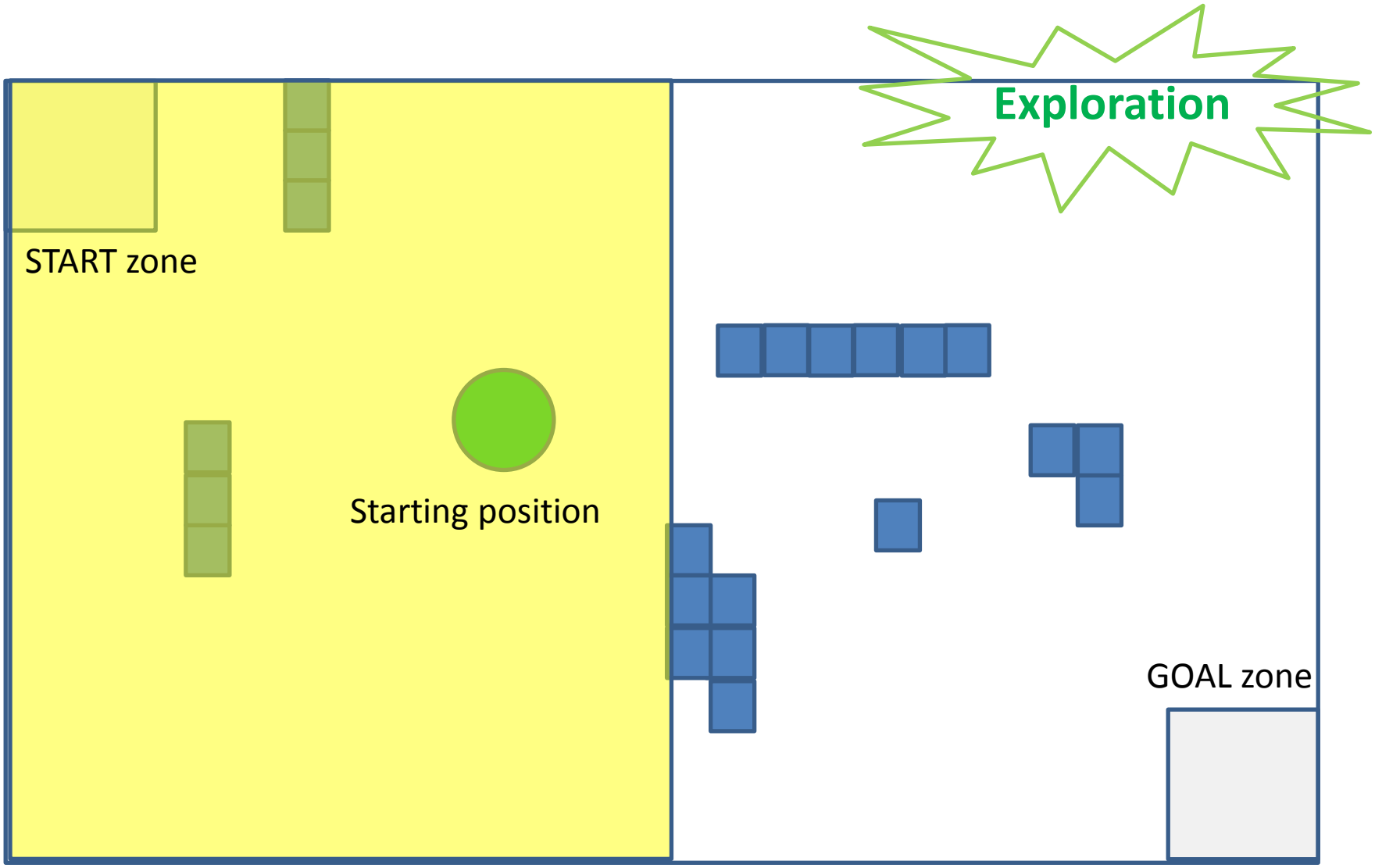
Robot's Environment

- Arena with internal dimensions 200cm x 150cm, completely surrounded by 15cm high boundary walls on four sides.
- **Rectangular** obstacles scattered in the arena.
- The rectangular obstacles are either in the **horizontal** or **vertical** orientation to the destination wall.
- The top left corner is called the START zone and the diagonally opposite corner is called the GOAL zone.
- START and GOAL zones are 30cm x 30cm each.
- Robot has a diameter of 20cm.

Sample Arena Layout

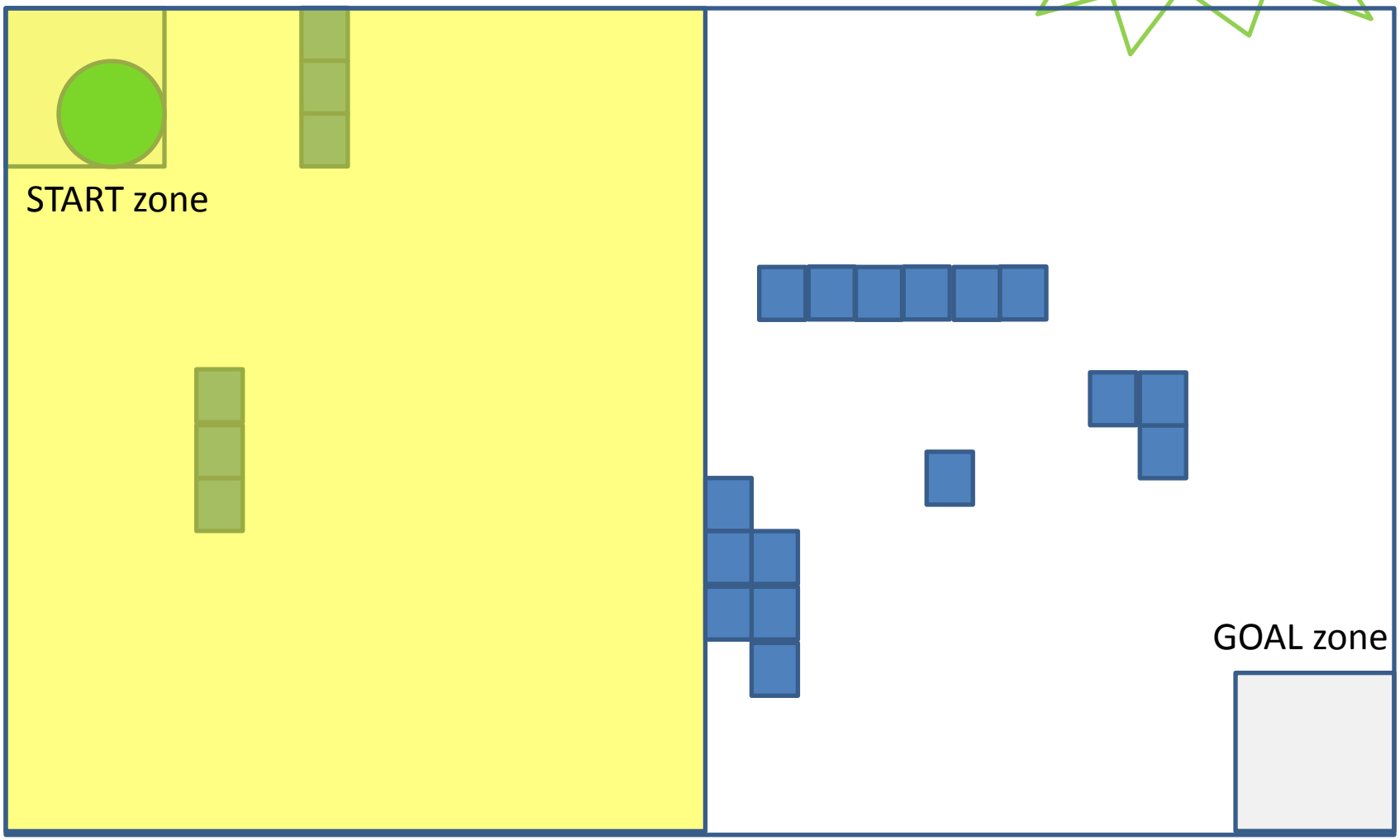


An obstacle is composed of one or more blocks, each block with ~10x10cm footprint, forming a rectangular shape



1. Place the robot anywhere within **YELLOW** Zone (first half of maze) with **ANY** orientation and update the coordinates of Starting position through Android Tablet.
2. Robot knows the positions of the START and GOAL zones and explores the arena.
3. Robot **MUST** enter **GOAL** Zone and explore back to **START** Zone within the Exploration Time Limit

Fastest Path Run



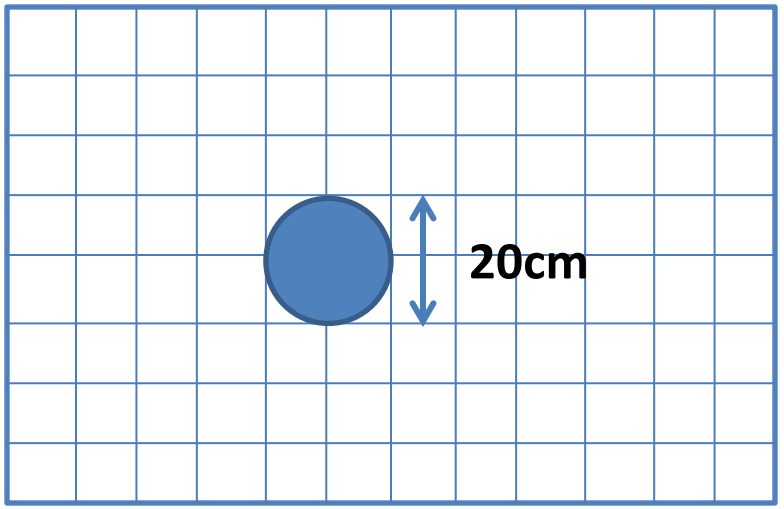
Robot to move from START to GOAL in shortest possible time.

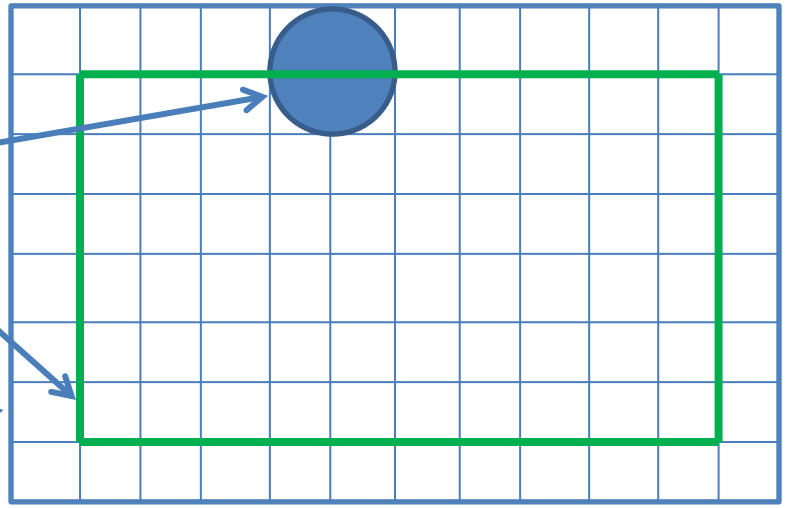
Problems/tasks to solve

1. How to represent the navigational area, the robot and the obstacles
2. How to navigate
3. How to explore
4. How to find the shortest/fastest path from START to GOAL.

1. How to represent the navigational area, the robot and the obstacles

- By using a 15x20 grid (150x200cm arenas using 10x10cm grid cells))
- Robot has a diameter of 20cm – 2x2 grid cells
- Space needed for the robot to move without contact with a wall





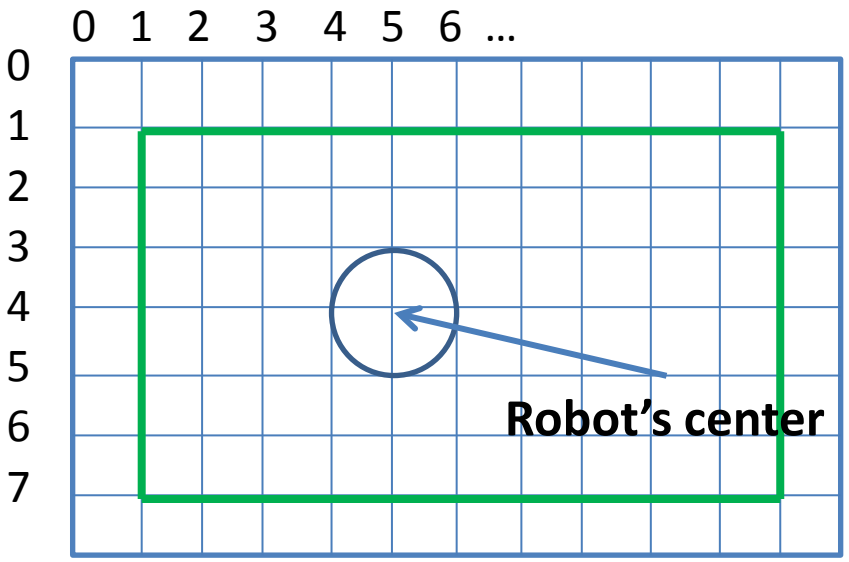
1. How to represent the navigational area, the robot and the obstacles

- The robot occupies 4 cells
- Robot is treated as a dot
- If the 4 cells are (a,b), (a,b+1), (a+1, b) and (a+1,b+1) the center is

  ((a+1)x10, (b+1)x10)

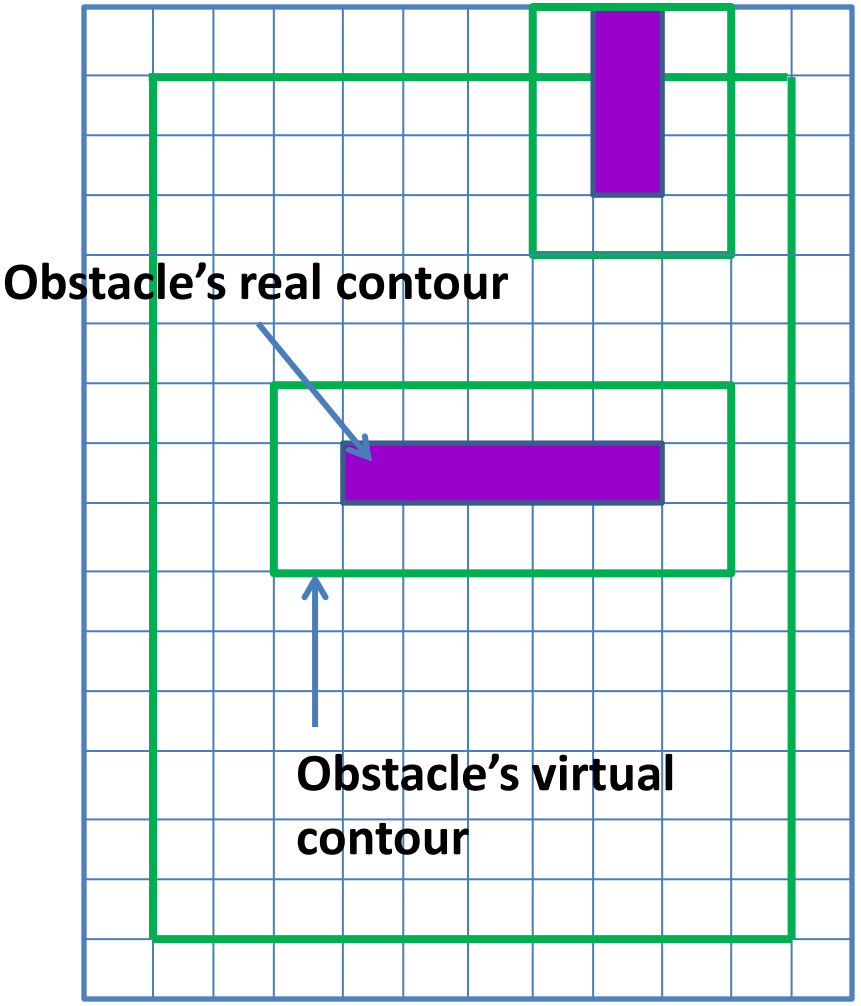
- E. g the robot occupying cells (4,3), (4,4), (5,3), (5,4) has the center at:

  ((4+1)x10, (3+1)x10)



1. How to represent the navigational area, the robot and the obstacles

- An obstacle is composed of one or more blocks, forming a rectangular shape
- each block with 1x1 grid cell footprint
- Space needed for the robot to move without contact with an obstacle

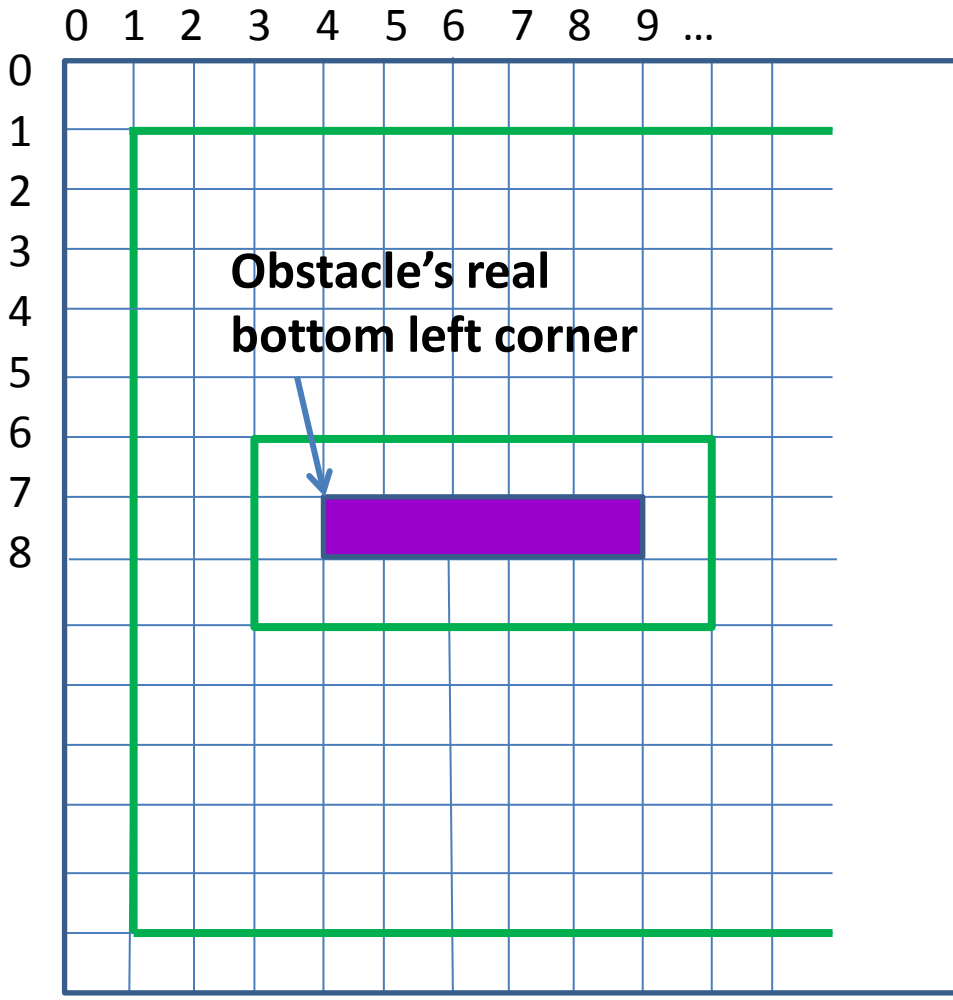


1. How to represent the navigational area, the robot and the obstacles

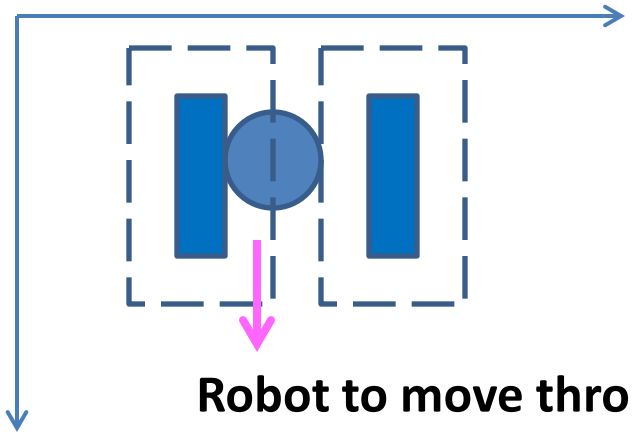
- The obstacle's 4 real corners clockwise from bottom left:
  ((3+1)x10, (6+1)x10)
  ((8+1)x10, (6+1)x10)
  ((8+1)x10, (7+1)x10)
  ((3+1)x10, (7+1)x10)

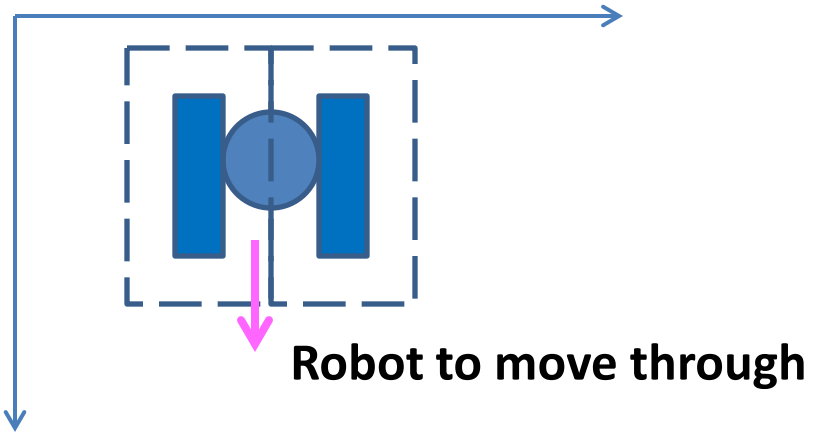
- The virtual corners:
  (3x10, 6x10), (10x10, 6x10)
  (10x10, 9x10), (3x10, 9x10)

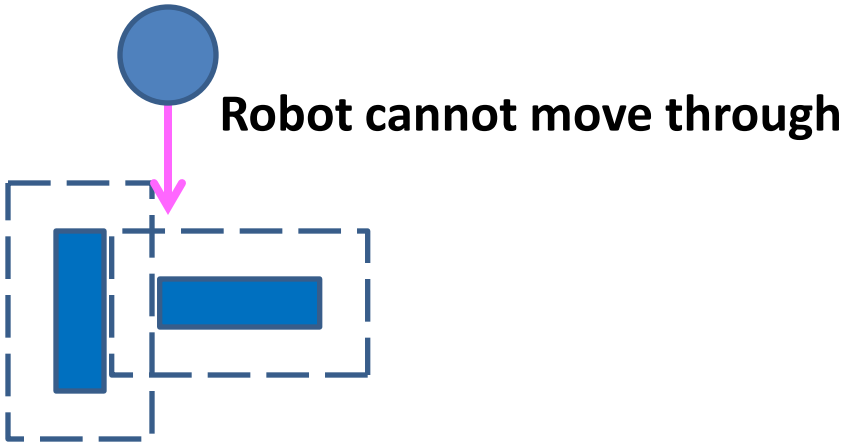
- If you want to have bigger clearance between robot and obstacle, enlarge obstacles by one more row of cells at each side

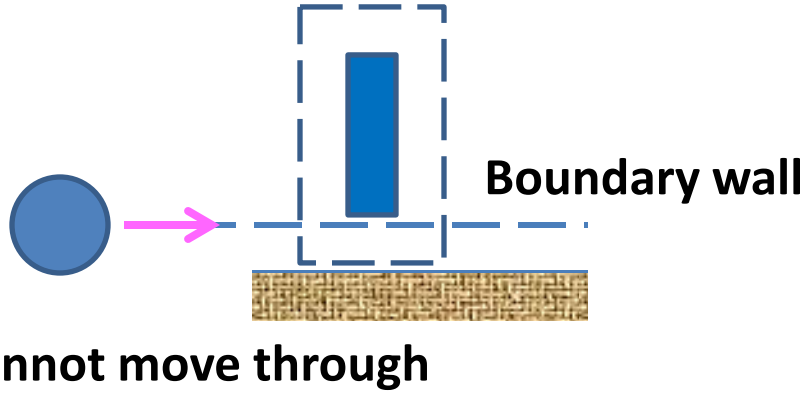


2. How to Simulate the Robot and the Algorithm



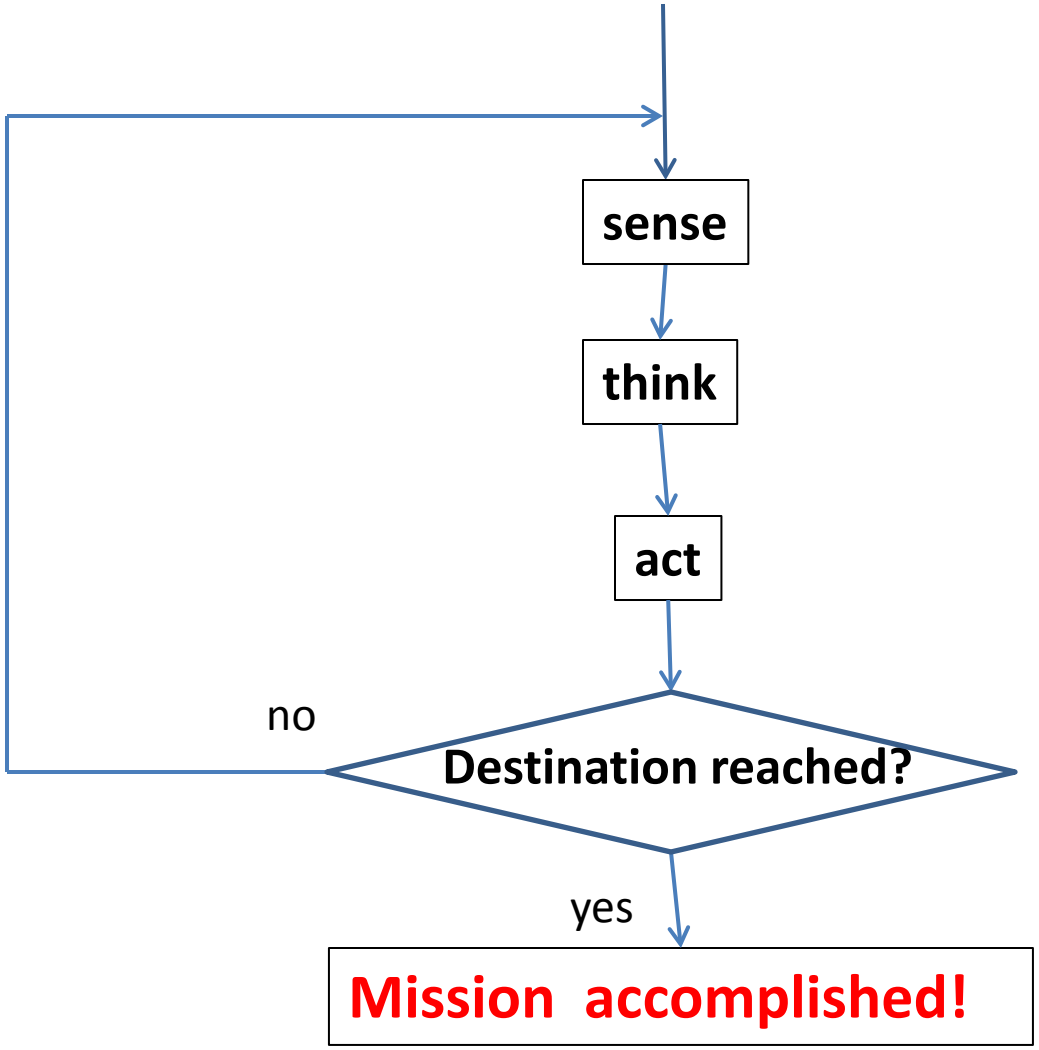




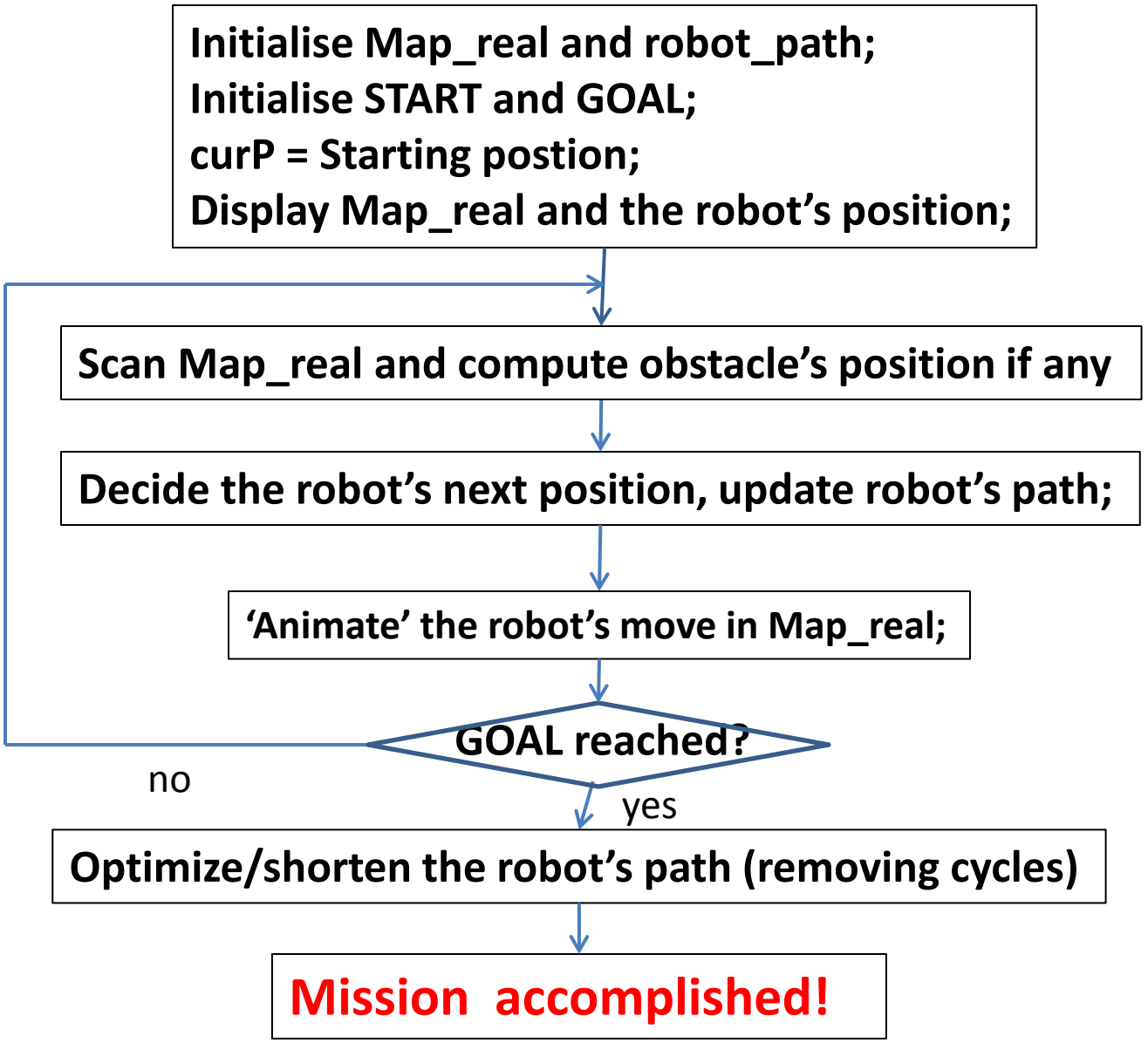


2. How to navigate

General control flow for robot navigation



2. How to navigate (more details)

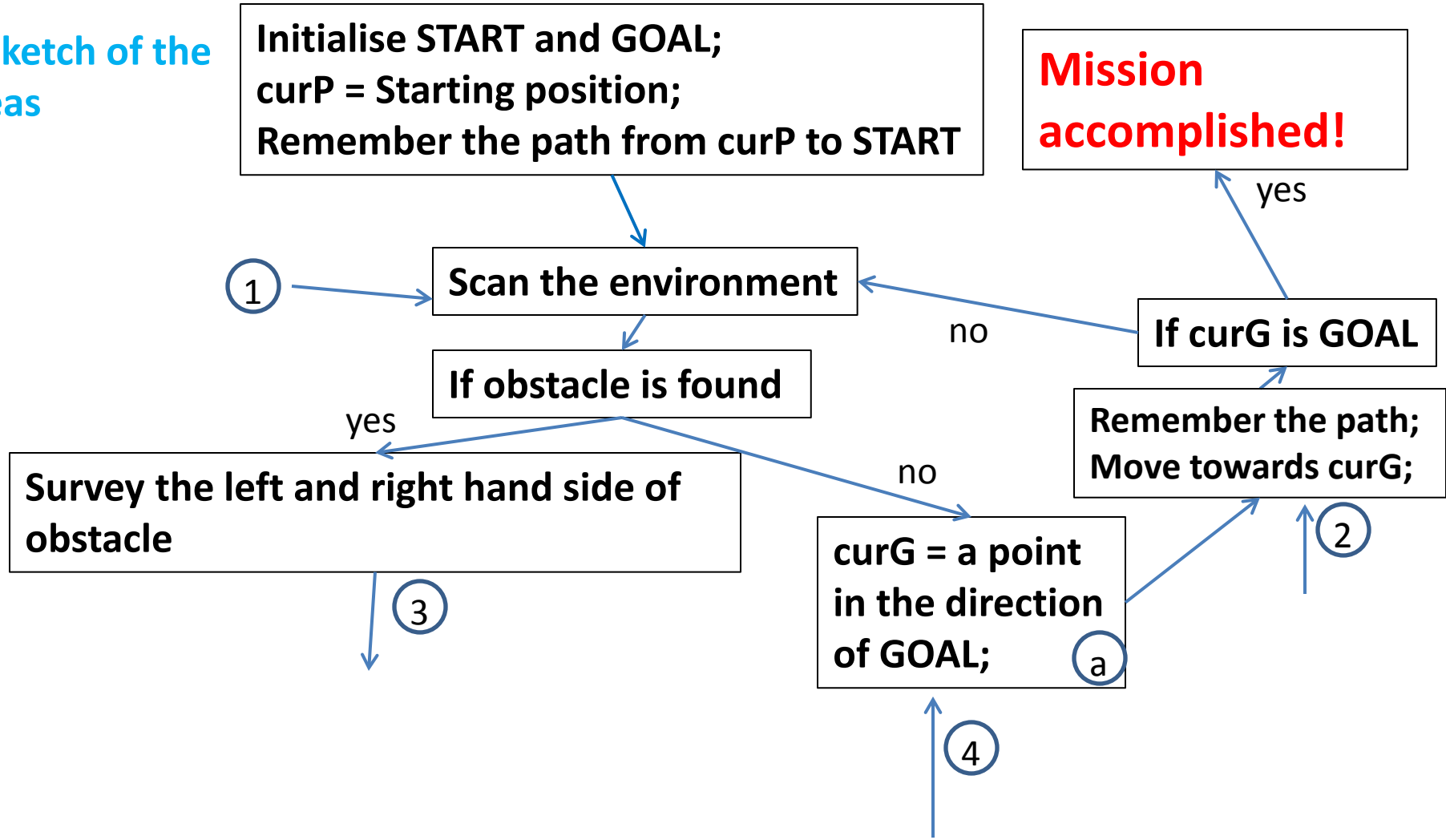


3. How to explore

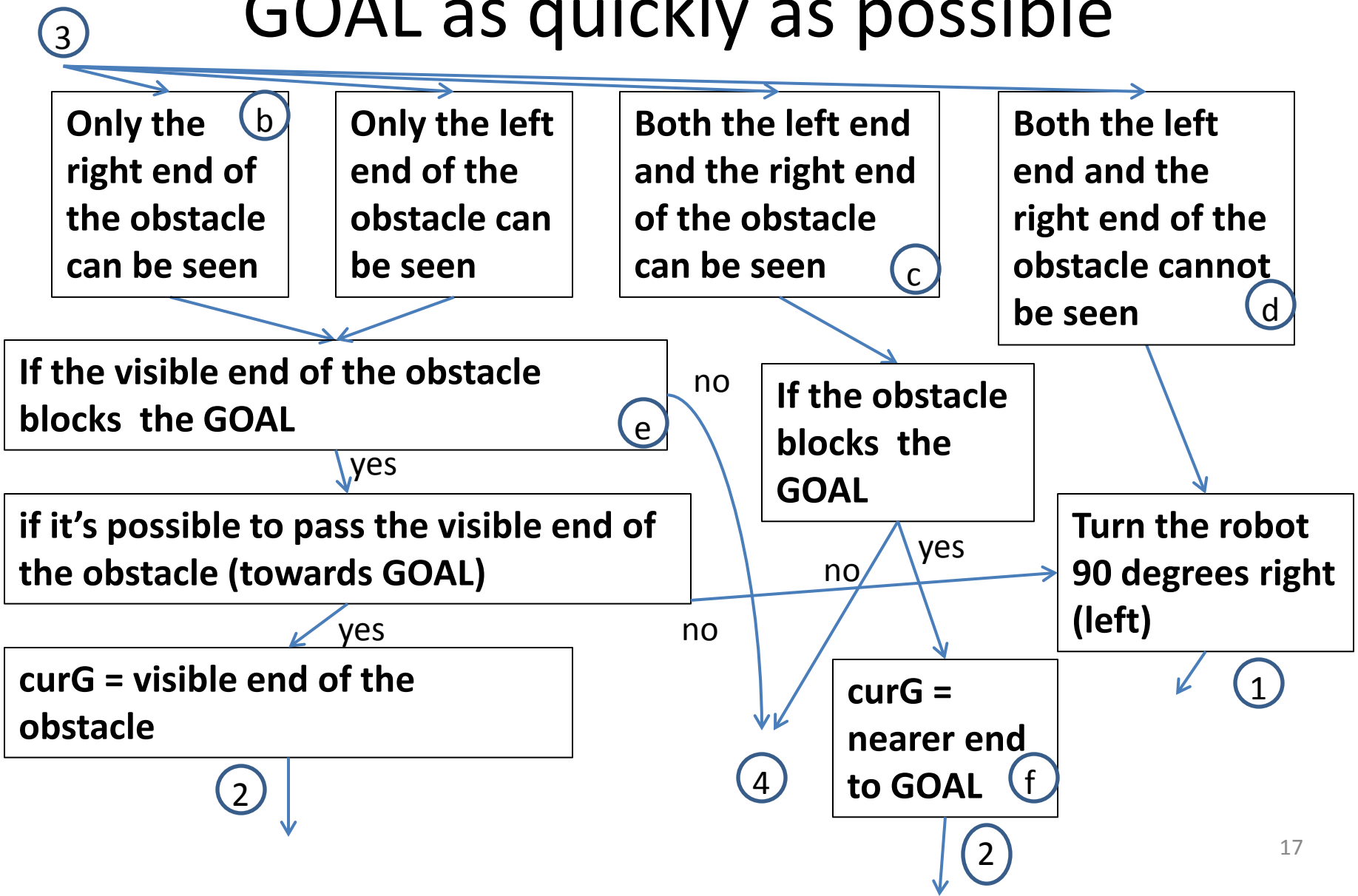
- First attempt:
  1. Find a good path to GOAL
  2. Reverse the path to travel from GOAL to START
- More explorations
  1. Explore more to build a map
  2. Shortest path algorithm to find better quality solution
- Possible to combine into one attempt

First attempt: find a good path to GOAL as quickly as possible

A sketch of the ideas



First attempt: find a good path to GOAL as quickly as possible



Some explanations

- How to scan the Environment
  - In simulation: search your map – any obstacle within sensor range?
  - In real operation, any obstacle visible?
- How to move towards curG
  - In simulation: nothing to do, assume the robot can do it. You may also display (animation) robot's moves from curP to curG.
  - In real operation, needs the commands to direct the robot and wait till it has reached curG

Some explanations

curG = a point in the direction of GOAL; a

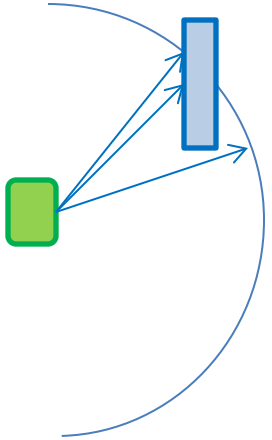
Only the right end of the obstacle can be seen b

- Using a grid, this is the node nearest to the GOAL within sensor range, say (h, k):

  distance(curP, (h, k)) < sensor distance and

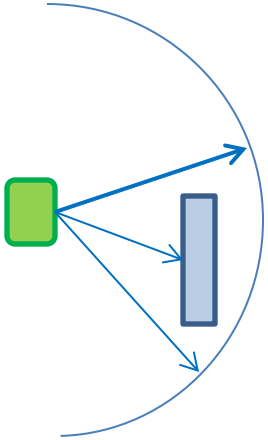
  distance((h, k), GOAL) is minimum

- A cell along the line from current position to GOAL within the sensor range

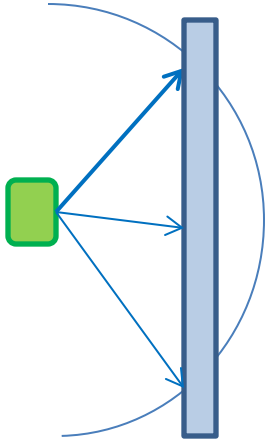


Some explanations

Both the left end and the right end of the obstacle can be seen c

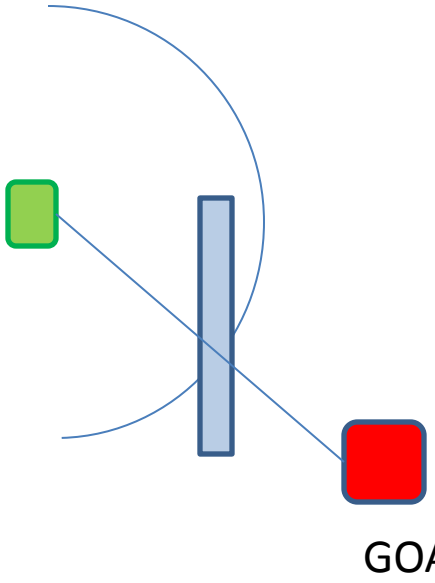


Both the left end and the right end of the obstacle cannot be seen d



Some explanations

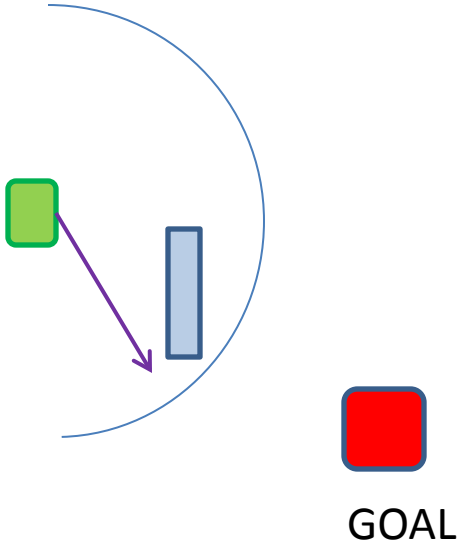
If the visible end of the obstacle blocks the GOAL e



The line linking the robot and GOAL intersect with an edge of the obstacle

Some explanations

Choose the nearer end to GOAL f

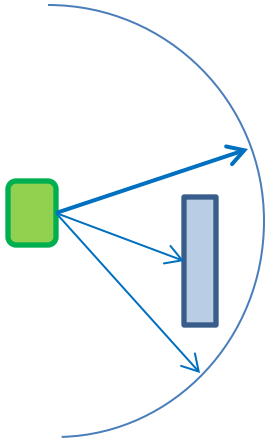


- Using a grid, choose between the two nodes near the two ends of the obstacle the one nearer to the GOAL

More explorations

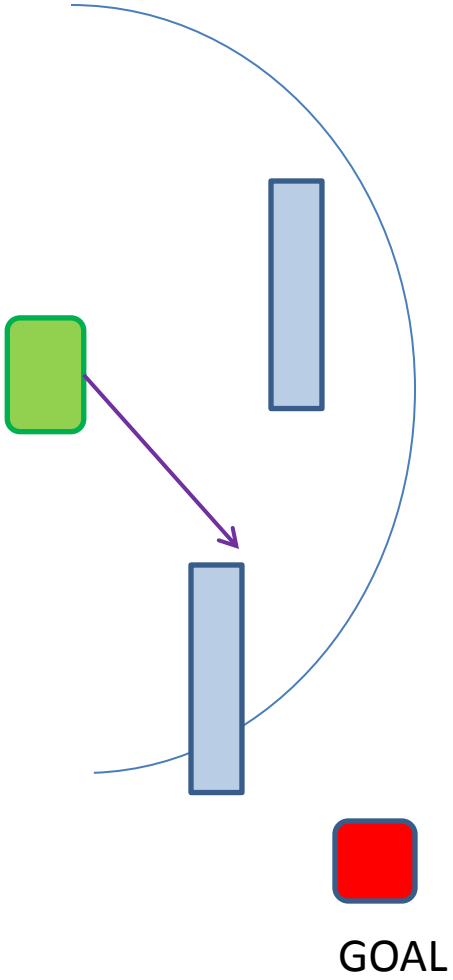
- We may choose to explore more without mapping out the whole arena.
- May sense more directions before each move.
- In the situation as shown in the right hand side, get the robot to move to both ends and continue the exploration

Both the left end and the right end of the obstacle can be seen C



More explorations

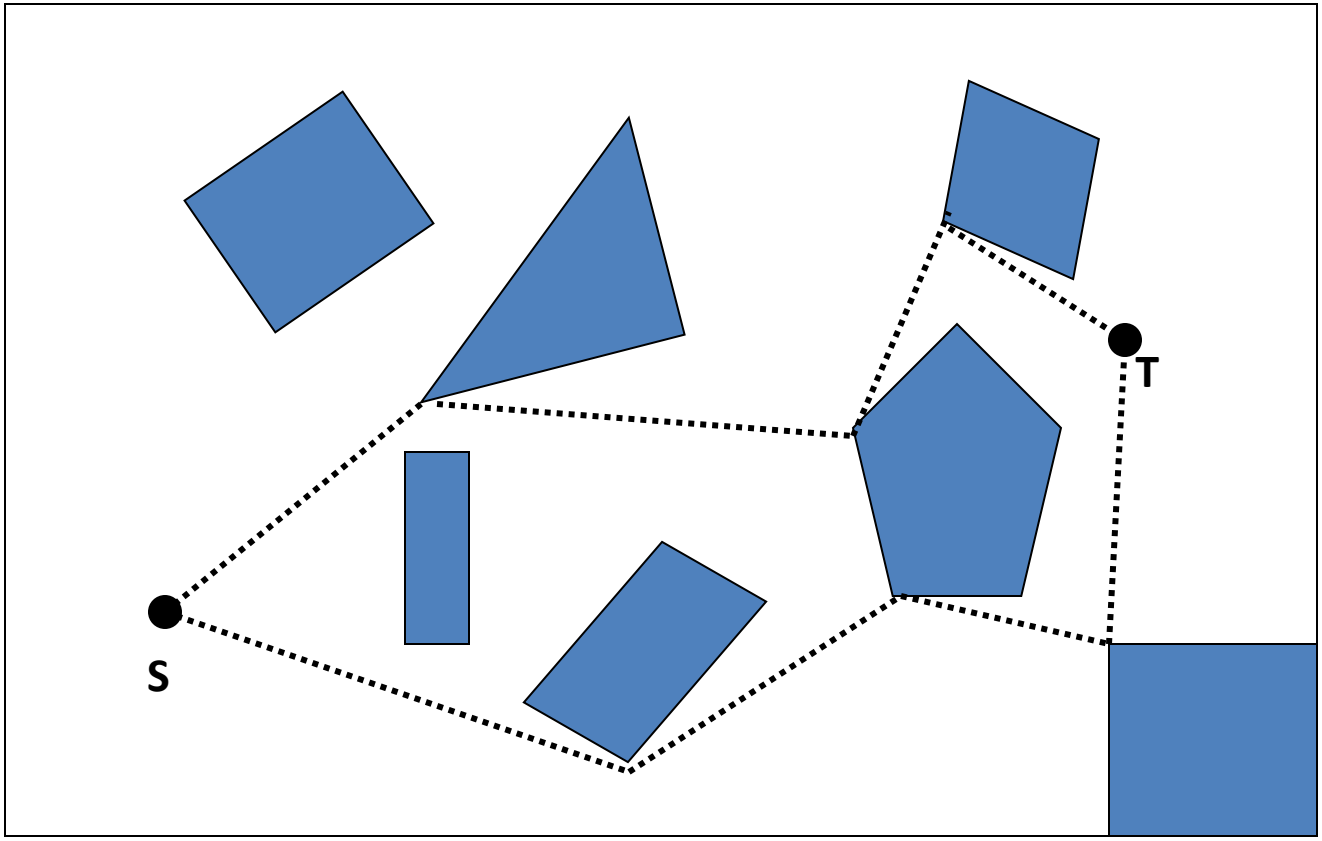
- In the situation where multiple obstacles are detected, get the robot to move along a path closer to ‘the shortest path’. Also explore the alternative choices which may result in a better path.
- Remember the time limit for exploration



4. How to find the shortest/fastest path from START to GOAL

- **Represent a robot as a point in a space called configuration space.**
- **The robot is the only moving object in the space.**
- **Move around the area so that the whole area is covered by sensor to find the locations of the obstacles.**
- **The starting position and the target position of the robot are given.**
- **We are to find the path from the starting to the target position.**

4. How to find the shortest/fastest path from START to GOAL



4. How to find the shortest/fastest path from START to GOAL

- **The Roadmap approach to path planning consists of capturing the connectivity of the robot's free space .**
- **The basic algorithm has three steps:**

  (1) construct the visibility graph G.

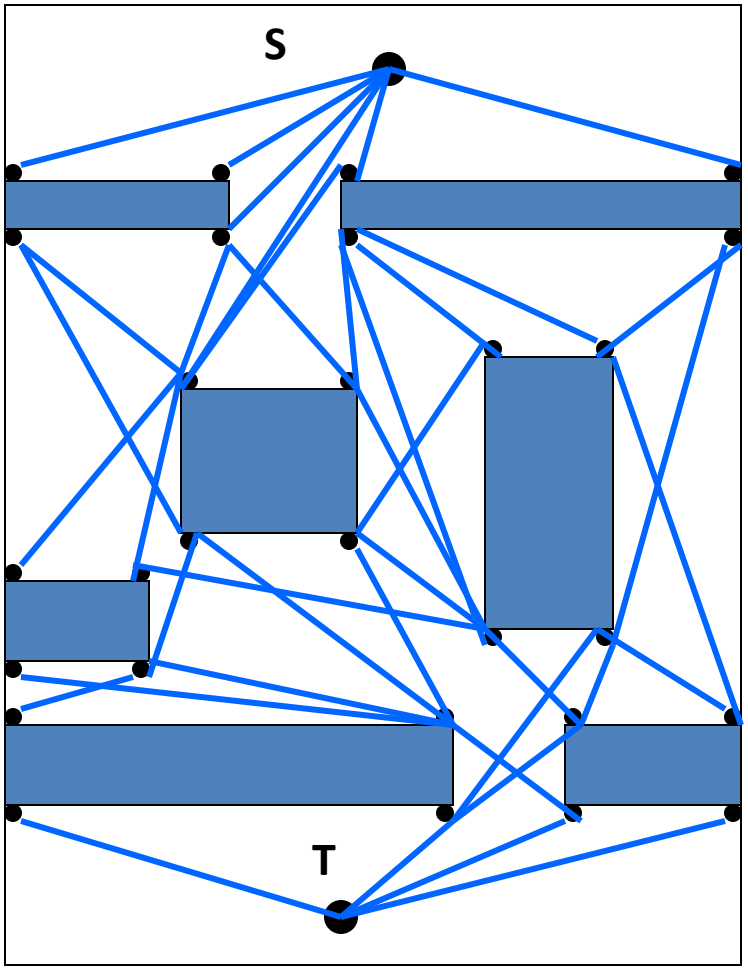
  (2) search G for a path from the starting to the target point.

  (3) if path is found, return it; otherwise, indicate failure.

4. How to find the shortest/fastest path from START to GOAL

Construction of a (partial) visibility graph G:

- **The vertices of the obstacles are considered as nodes.**
- **The starting and target points are considered as nodes.**
- **The nodes are linked with straight line segments if they do not intersect the interior of any obstacle region**.



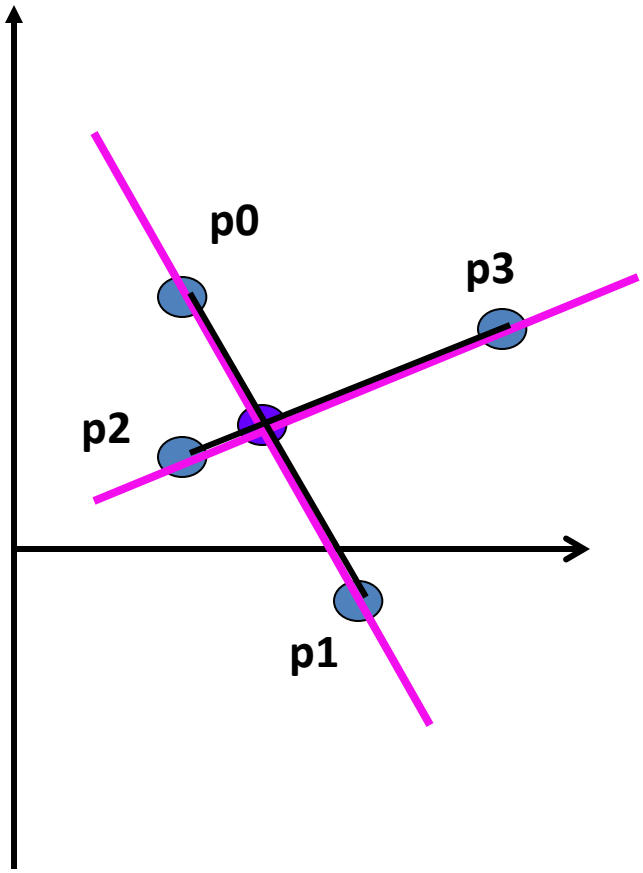
Appendix A

This shows a method to do a full exploration of the whole arena. You may not have the time to do it. However, some of the techniques are useful. For example, how to decide whether two line segments cross each other.

Determine whether 2 line segments intersect (method 1)

- We are given line segments (p0, p1) and (p2, p3) on the plane and we have to determine whether they intersect or not.

- We find the intersection point of the 2 lines that contain the 2 line segments respectively, and then check that the intersection is on the segments.



Recap:

- **The straight line through two points (x_0, y_0), (x_1, y_1) in the coordinate plane can be defined by the equation**

  $$ax + by = c,$$

  where $a = y_1 - y_0$, $b = x_0 - x_1$, $c = x_0y_1 - y_0x_1$

- **So we find the 2 lines**

  $a_1x + b_1y = c_1$, L_1 **through p0, p1**

  $a_2x + b_2y = c_2$ L_2 **through p2, p3**

  and compute the intersection (x, y).

- **For example, we substitute**

  $$y = \frac{c_1 - a_1 x}{b_1}$$

  into

  $$x = \frac{c_2 - b_2 y}{a_2}$$

- **Finally we check that:**

  $\min(x0, x1) \leq x \leq \max(x0, x1)$ and

  $\min(x2, x3) \leq x \leq \max(x2, x3)$

- **Cases that need special treatment:**

  (1) The system of 2 equations has no solution

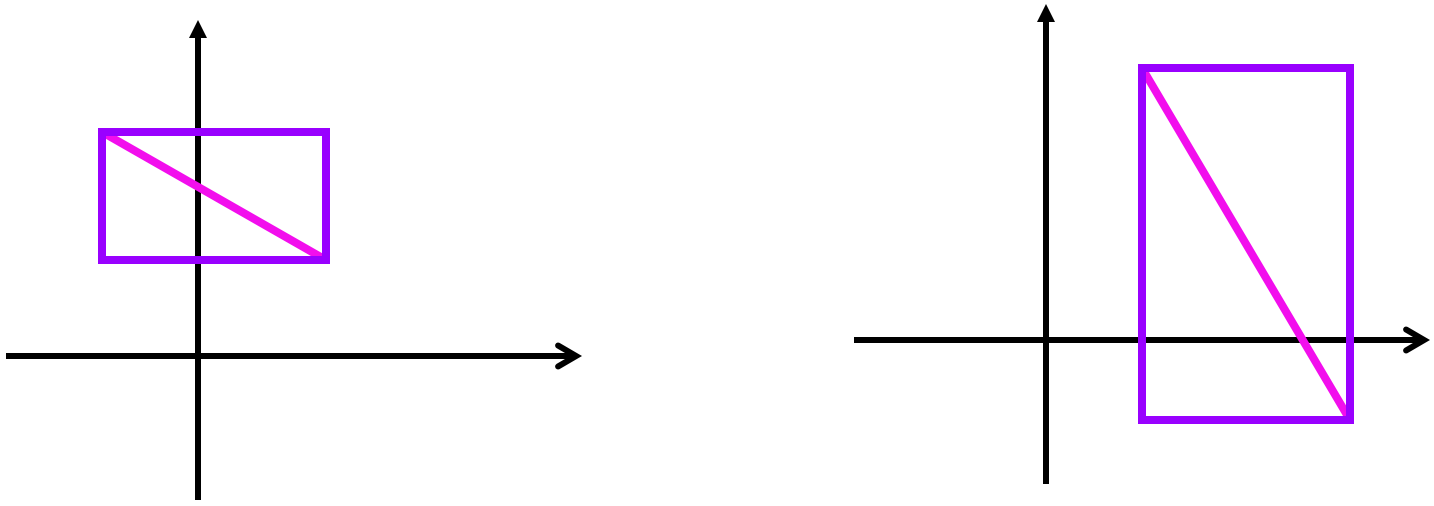
  (2) The system of 2 equations has infinite solutions

- **This method needs divisions in several places**
- **Division generates truncation errors that can produce the wrong final results**
- **However we do not need to find the intersection point in order to detect its existence**

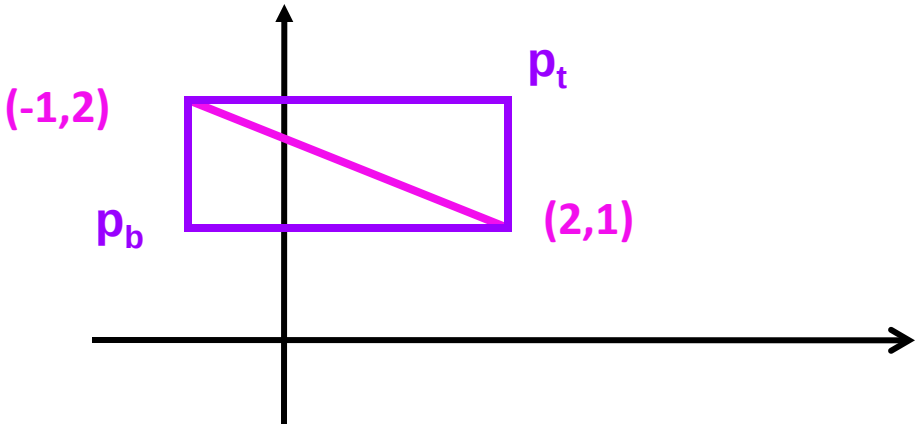
Determine whether 2 line segments intersect (method 2)

- A two-stage process is used to determine whether 2 line segments intersect.
- The 1st stage is quick rejection: 2 line segments cannot intersect if their bounding boxes do not intersect.
- The 2nd stage decides whether each segment "straddles" the line containing the other.

<u>Definition</u>: The bounding box of a geometric figure is the smallest rectangle that contains the figure and whose segments are parallel to the x-axis and y-axis.



The bounding box of a line segment is represented by the rectangle (p_b, p_t) with the lower left point $p_b = (x_b, y_b)$ and upper right point $p_t = (x_t, y_t)$ where $x_b = \min(x_1, x_2)$, $y_b = \min(y_1, y_2)$, $x_t = \max(x_1, x_2)$ and $y_t = \max(y_1, y_2)$.



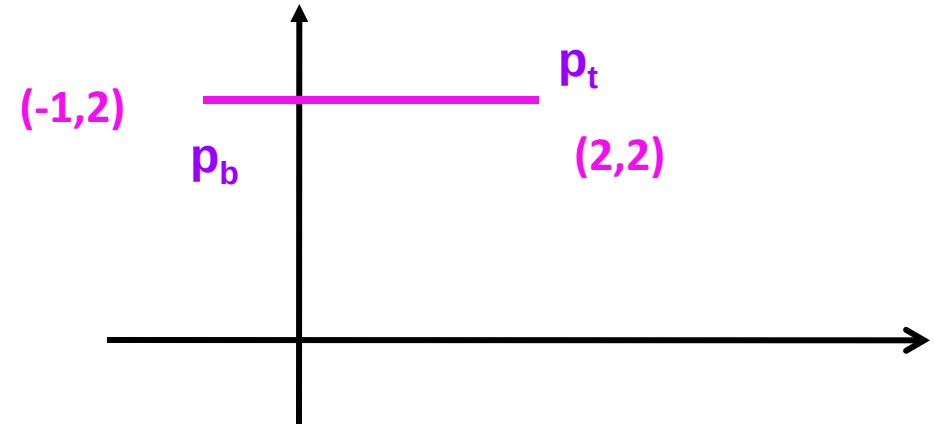
$p_b = (-1, 1)$

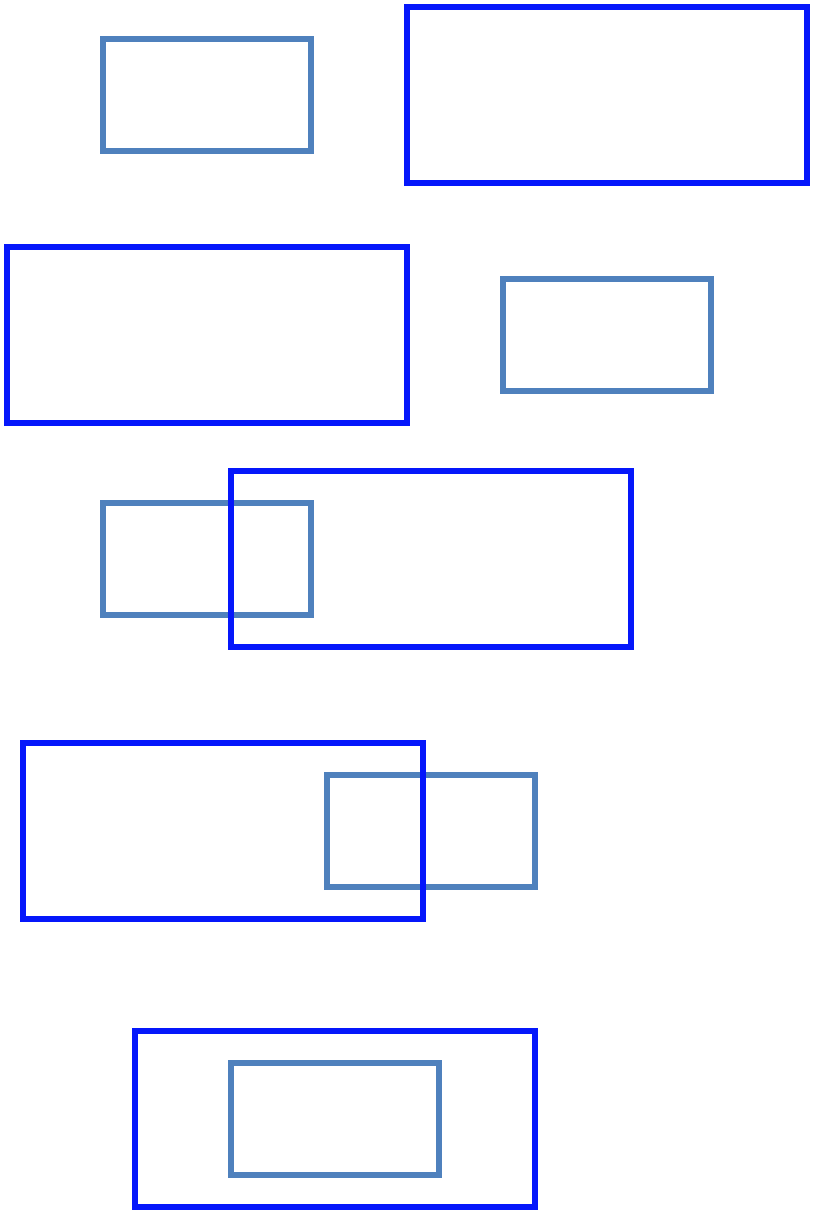
$p_t = (2, 2)$

A special case:

$p_b = (-1, 2)$

$p_t = (2, 2)$





$x_1 < x_2$ and $x_3 < x_4$

$x_1 < x_2 < x_3 < x_4$

$x_3 < x_4 < x_1 < x_2$

$x_1 < x_3 < x_2 < x_4$

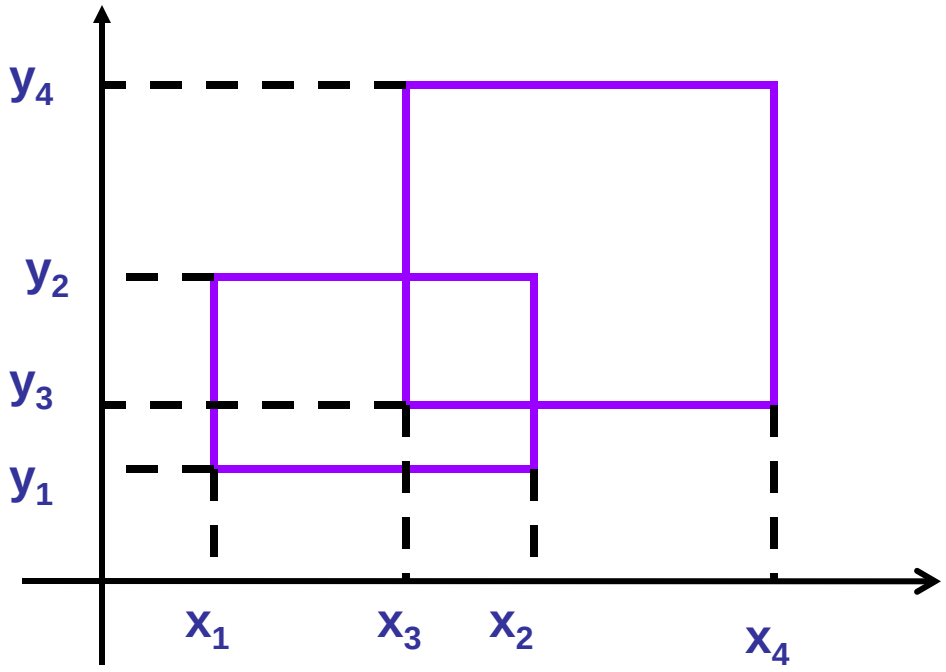
$x_3 < x_1 < x_4 < x_2$

$x_3 < x_1 < x_2 < x_4$

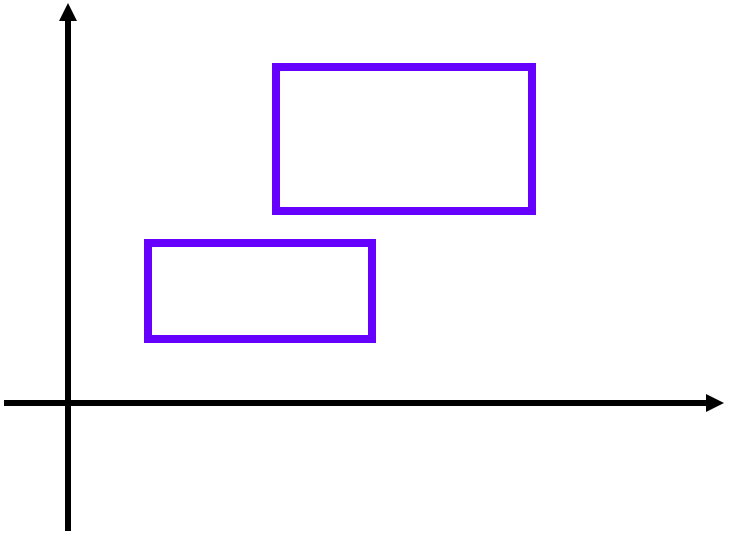
$x_1 < x_3 < x_4 < x_2$

Two rectangles, represented by their lower left and upper right points (p_b, p_t) and (p_b', p_t') respectively, intersect if and only if

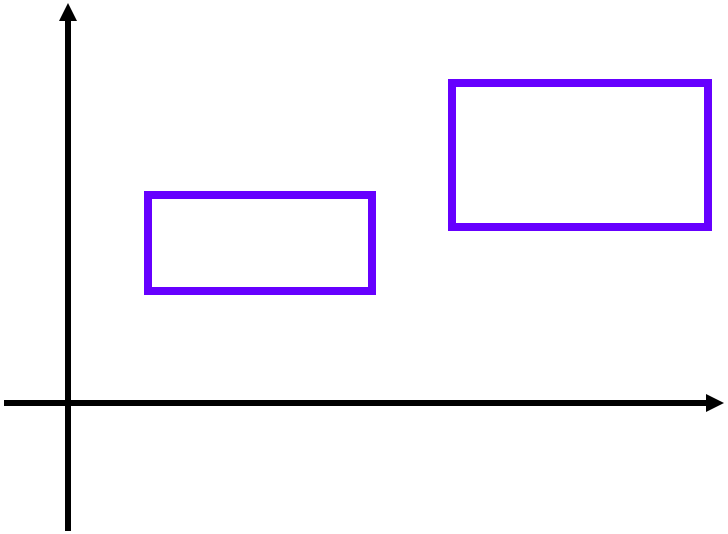
$$(x_2 \geq x_3) \wedge (x_4 \geq x_1) \wedge (y_2 \geq y_3) \wedge (y_4 \geq y_1)$$ **is true.**



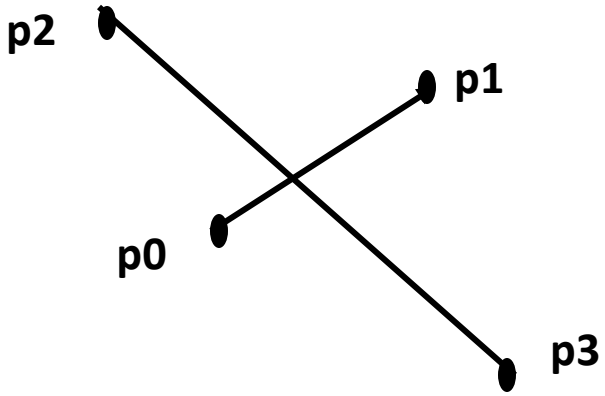
$(x_2 \geq x_3) \wedge (x_4 \geq x_1)$ only:

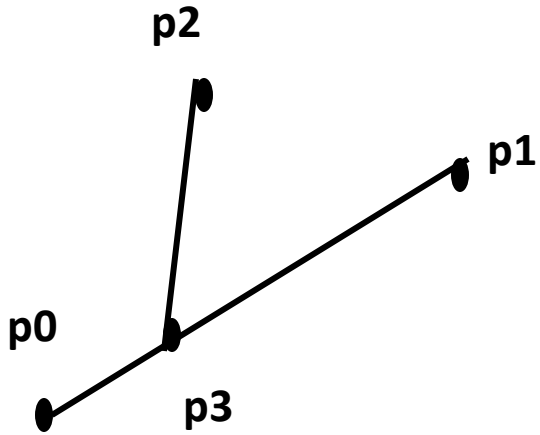


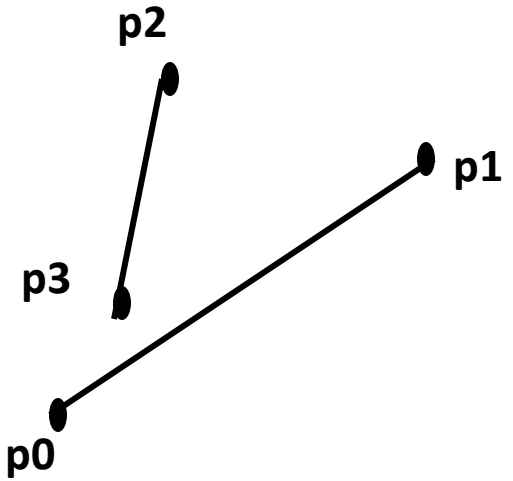
$(y_2 \geq y_3) \wedge (y_4 \geq y_1)$ only:

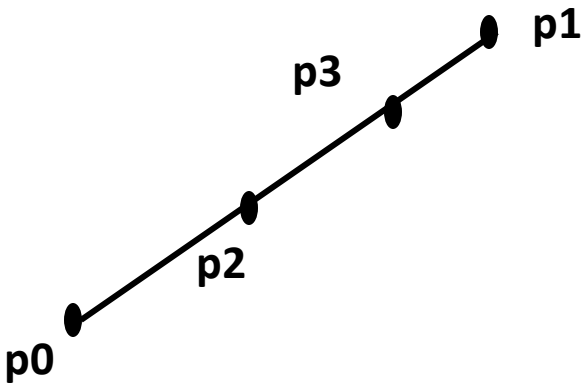


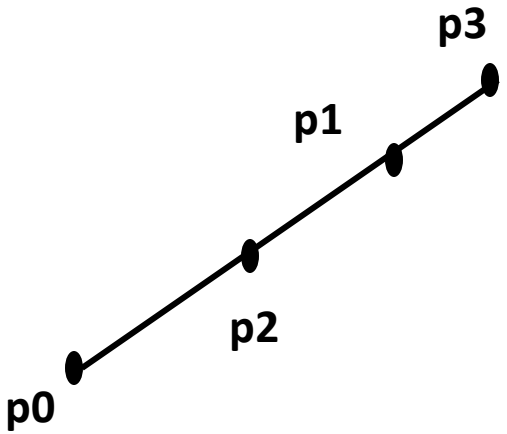
After the 2 line segments pass the rejection test (so two bounding boxes do intersect): check whether each line segment “straddles” the line containing the other





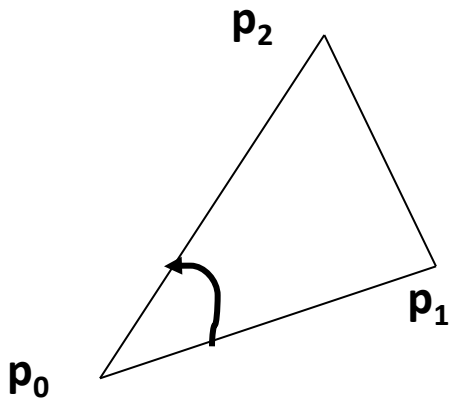






Determining whether p_2 is on the left of line segment $\overline{p_0 p_1}$

For examples



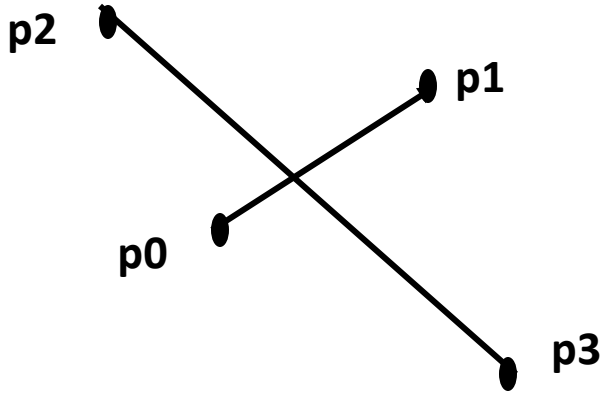
p_0 = (2, 3), p_1 = (5, 4), p_2 = (4, 7),

$(p_2 - p_0) \times (p_1 - p_0)$

$$= \begin{vmatrix} 4-2 & 5-2 \\ 7-3 & 4-3 \end{vmatrix}$$

= 2 – 12 =-10

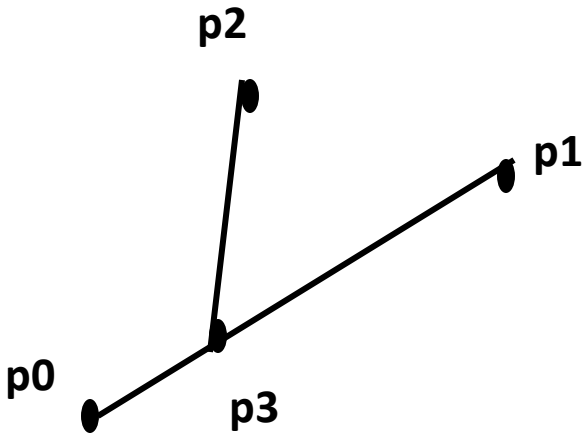
p_2 is on the left of line segment $\overline{p_0 p_1}$.



$$(p2 - p0) \times (p1 - p0) < 0$$

$$(p3 - p0) \times (p1 - p0) > 0$$

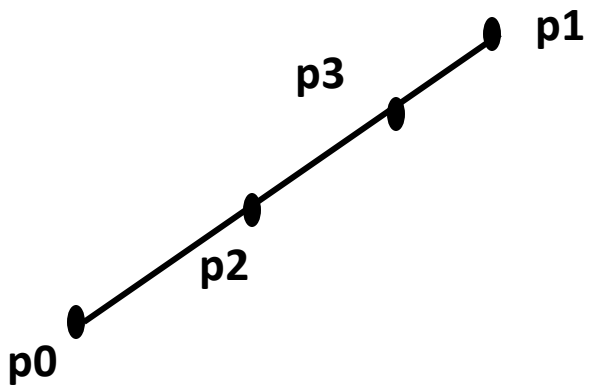
P2 is on the left of p0p1, p3 is on the right of p0p1



$$(p2 - p0) \times (p1 - p0) < 0$$

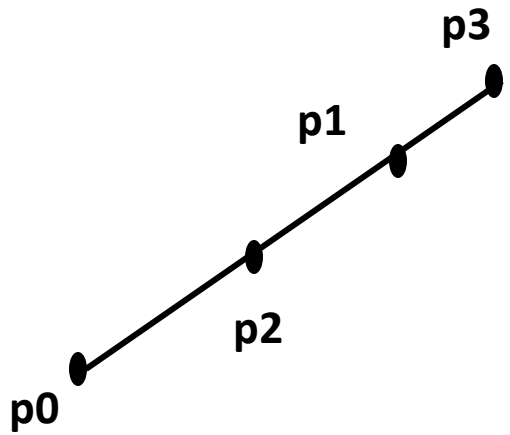
$$(p3 - p0) \times (p1 - p0) = 0$$

P2 is on the left of p0p1, p3 is collinear to p0p1



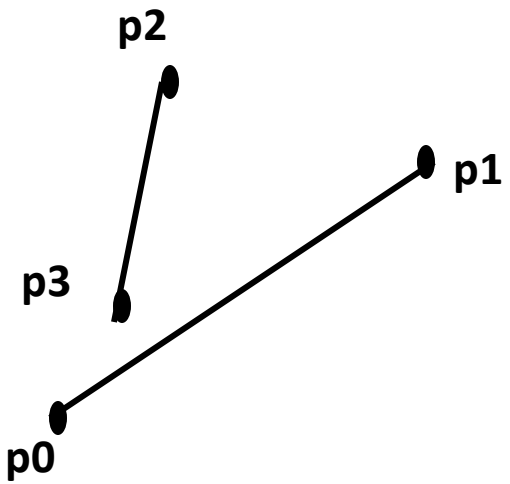
$$(p2 - p0) \times (p1 - p0) = 0$$

$$(p3 - p0) \times (p1 - p0) = 0$$



$$(p2 - p0) \times (p1 - p0) = 0$$

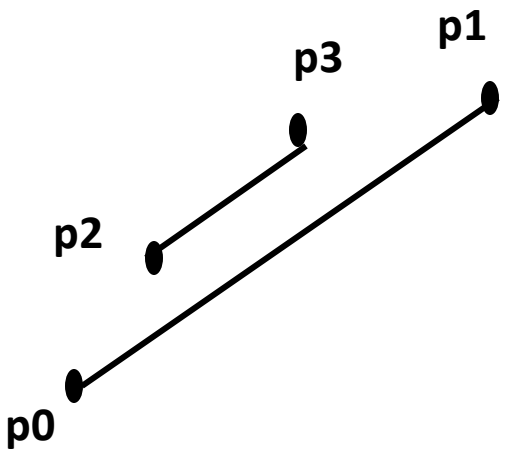
$$(p3 - p0) \times (p1 - p0) = 0$$



$$(p2 - p0) \times (p1 - p0) < 0$$

$$(p3 - p0) \times (p1 - p0) < 0$$

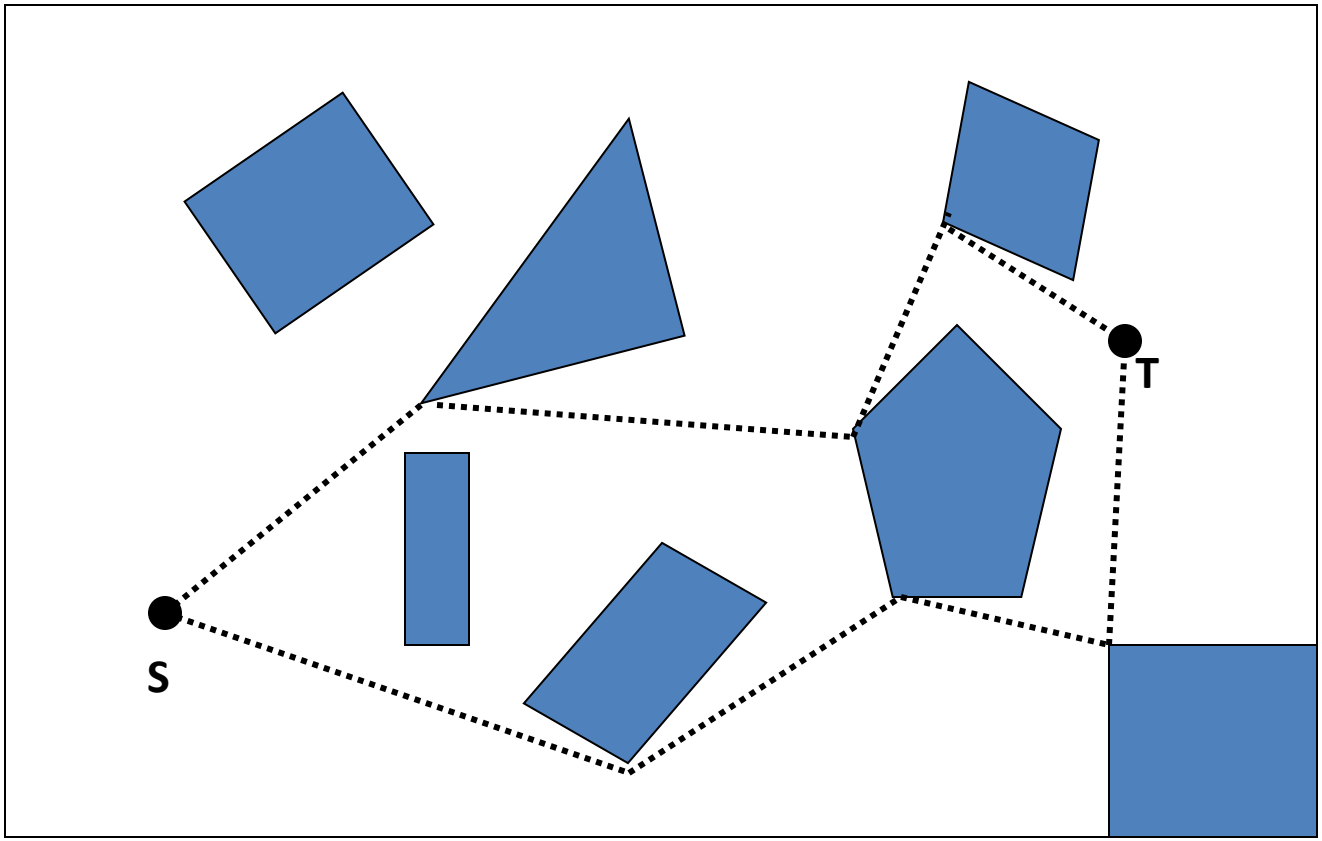
P2 and p3 are both on the left of p0p1. BUT p1 is on the left of p2p3 and p0 is on the right of p2p3



Conclusion:
Two line segments intersect if and only if
(i) they pass the rejection test and
(ii) $(p2 - p0) \times (p1 - p0)$ and $(p3 - p0) \times (p1 - p0)$ do not have the same signs (should test from both segments)

Basic Motion Planning Problem

- Represent a robot as a point in a space called configuration space.
- The robot is the only moving object in the space.
- Move around the area so that the whole area is covered by sensor to find the locations of the obstacles.
- The starting position and the target position of the robot are given.
- We are to find the path from the starting to the target position.


T
S

The Roadmap approach

- **The Roadmap approach to path planning consists of capturing the connectivity of the robot's free space .**
- **The basic algorithm has three steps:**

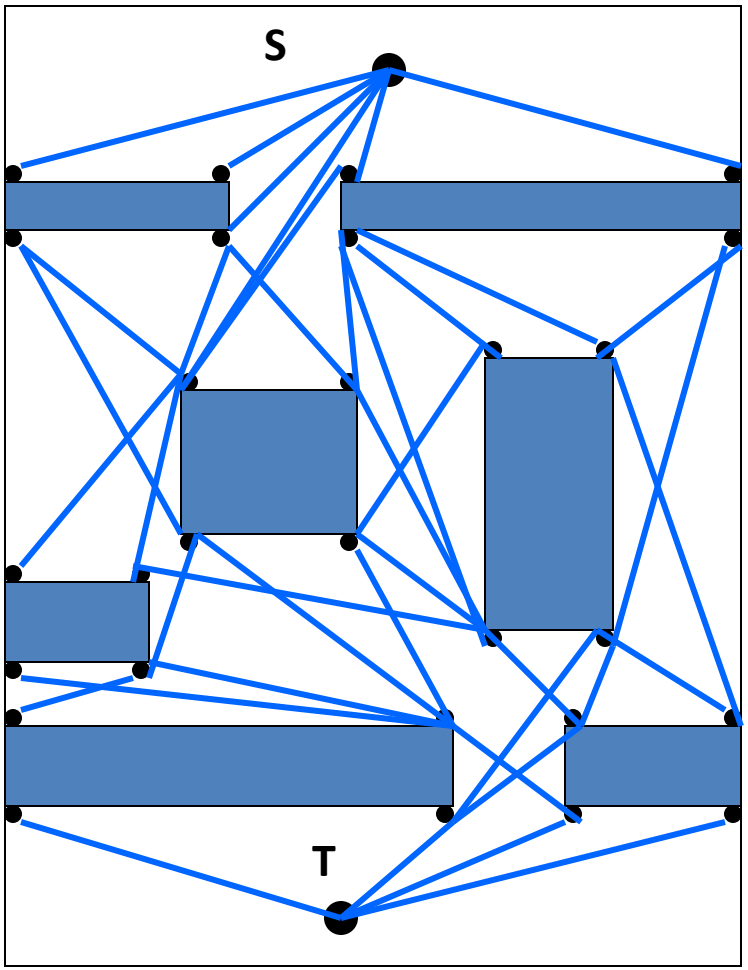
  (1) construct the visibility graph G.

  (2) search G for a path from the starting to the target point.

  (3) if path is found, return it; otherwise, indicate failure.

Construction of the visibility graph G

- The vertices of the obstacles are considered as nodes .
- The starting and target points are considered as nodes.
- The nodes are linked with straight line segments if they do not intersect the interior of any obstacle region

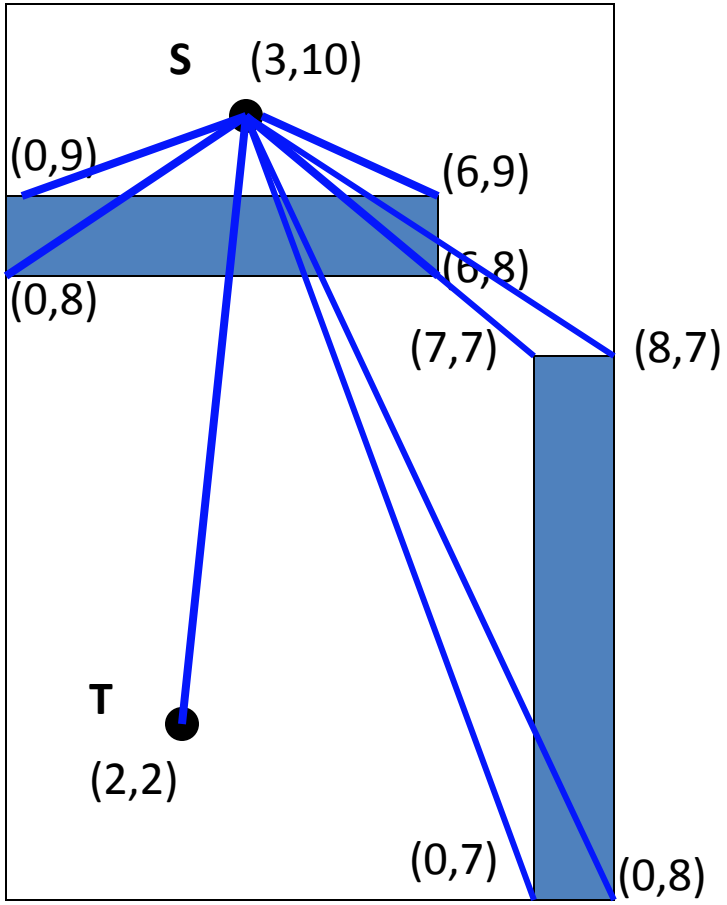


- **To decide whether a line segment intersects the interior of any obstacle region, we check whether the line segment crosses any side of any obstacle.**
- **The algorithm given above runs in time $O(n^3)$ where n is the total number of vertices in the visibility graph G.**

  -- there are at most $O(n^2)$ number of edges in G.

  -- there are O(n) obstacle sides
- **There are more efficient algorithms**

Example: demonstrate how a path from S to T is found



A path:

(3,10) -> (6, 9) -> (6, 8) -> (2, 2)

Appendix B

This shows a method how to decide which point is closest to a line segment among a number of points

To decide which point is closer to a line segment

- **A useful fact from analytical geometry:**

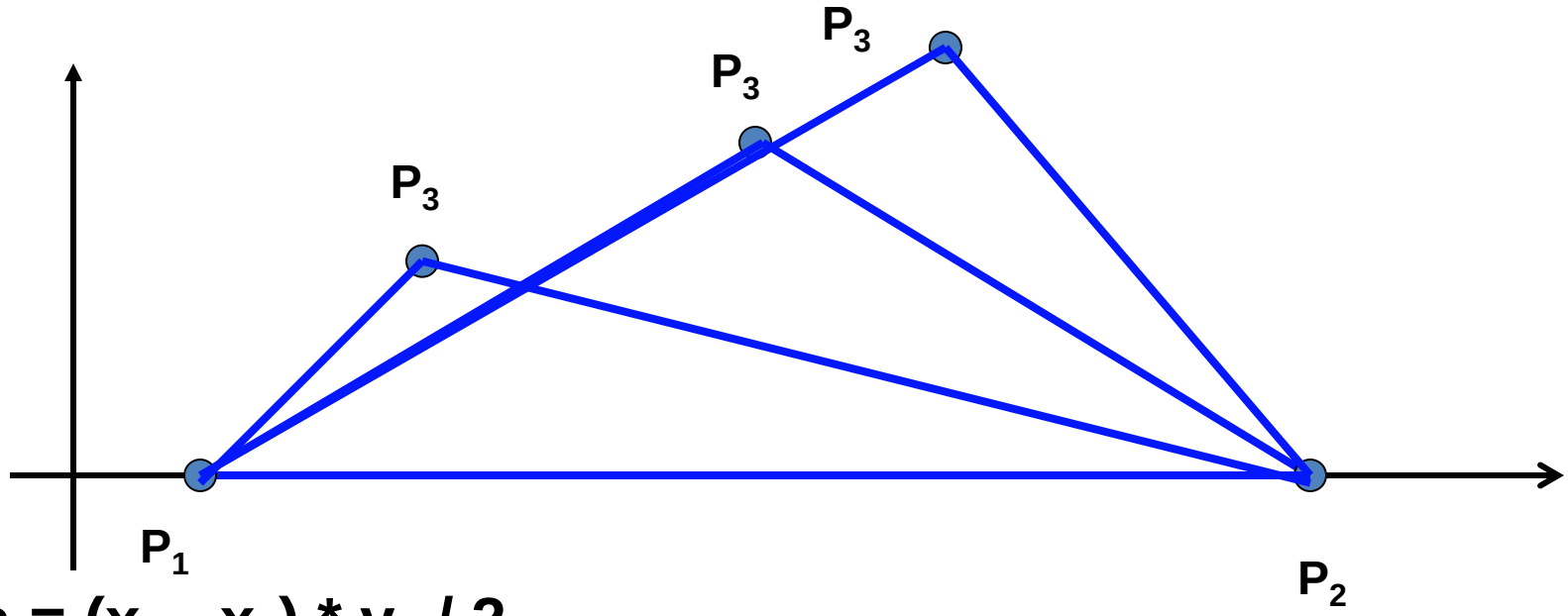
  $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ and $P_3 = (x_3, y_3)$ are three arbitrary points in the plane, then the area of the triangle $\Delta P_1P_2P_3$ is equal to one half of the magnitude of the determinant

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1y_2 + x_3y_1 + x_2y_3 - x_3y_2 - x_2y_1 - x_1y_3$$

To decide which point is closer to a line segment

1. The sign of the determinant is positive if and only if the point P_3 is to the left of the line from P_1 to P_2 .
2. The area of the triangle is directly proportional to the distance of the point P_3 from the line $P_1 P_2$. This is a general property for any three points.
3. So to choose the closest point to a line segment among a number of points, we just need to see which triangle has the smallest area.

Consider a triangle with $P_1 = (x_1, 0)$, $P_2 = (x_2, 0)$ and $P_3 = (x_3, y_3)$ and $y_3 > 0$, for example,



Area = $(x_2 - x_1) * y_3 / 2$

where y_3 is the distance of the point P_3 from the line P_1P_2 .

So the area of the triangle is directly proportional to the distance of the point P_3 from the line $P_1 P_2$. This is a general property for any three points.

Appendix C

- Focus on slide 14 – 22 of “potential-fields.pdf”