Day 27

Bug Algorithms

Fundamental Problems

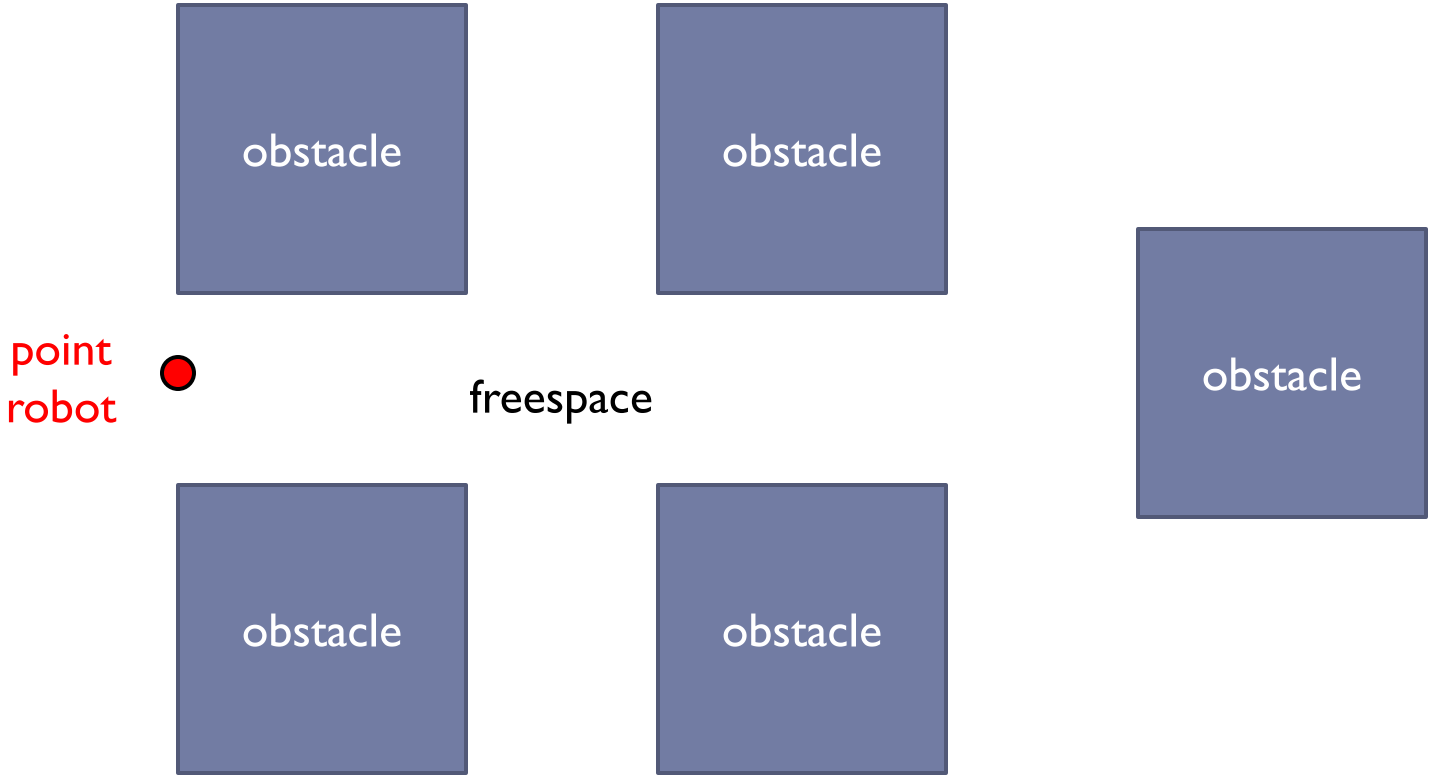
- Chapter 2 of Dudek and Jenkin begins:
  - "Before delving into the harsh realities of real robots…"
- lists 5 fundamental problems
  1. path planning
  2. localization
  3. sensing or perception
  4. mapping
  5. simultaneous localization and planning

A Point Robot

- represents a mobile robot as a point in the plane*
- the point P fully describes the state of the robot
  - called pose or configuration
- robot motion causes the state to change

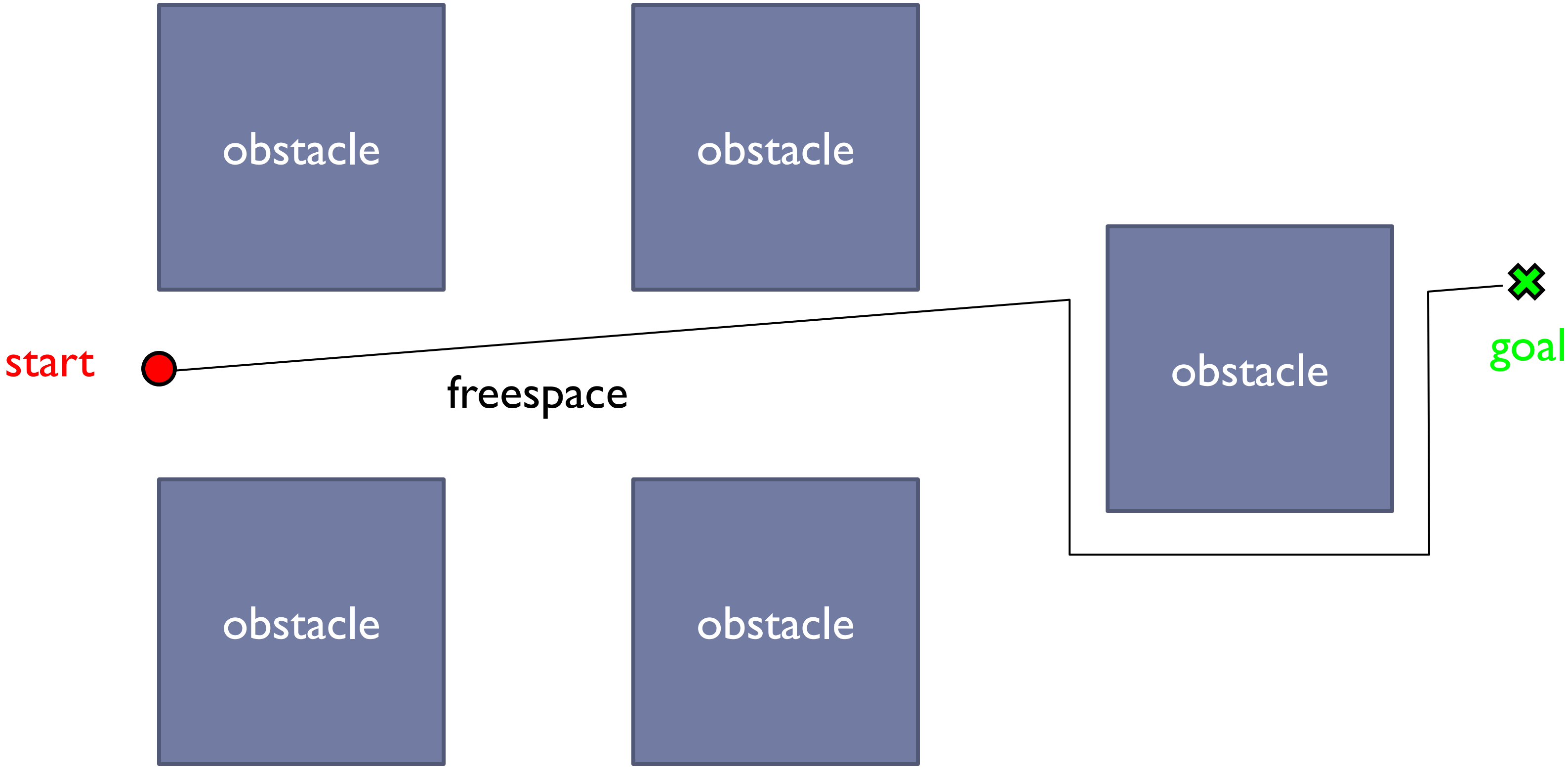
Free Space and Obstacles

- the set of valid poses is called the free space C_{free} of the robot
- the invalid poses are obstacles



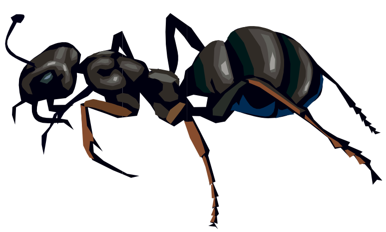
Path Planning

- is it possible for the robot to move to a goal configuration while remaining in C_{free} ?



Path Planning Using Bugs

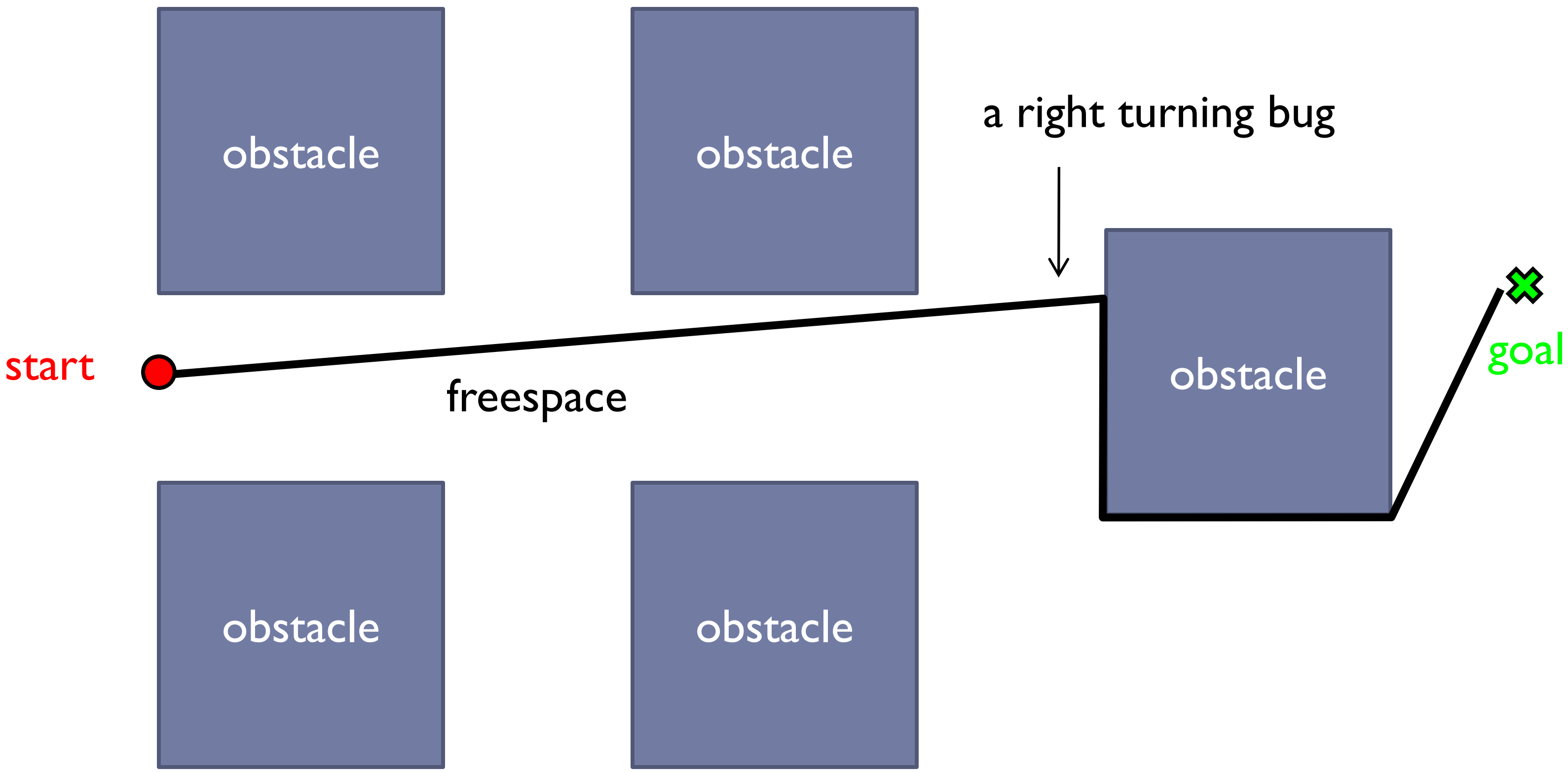
- bug algorithms assume:
  - point robot
  - known goal location
  - finite number of bounded obstacles
  - robot can perfectly sense its position at all times
  - robot can compute the distance between two points
  - robot can remember where it has been
  - robot can perfectly sense its local environment
  - robot can instantaneously change direction



Bug Zero

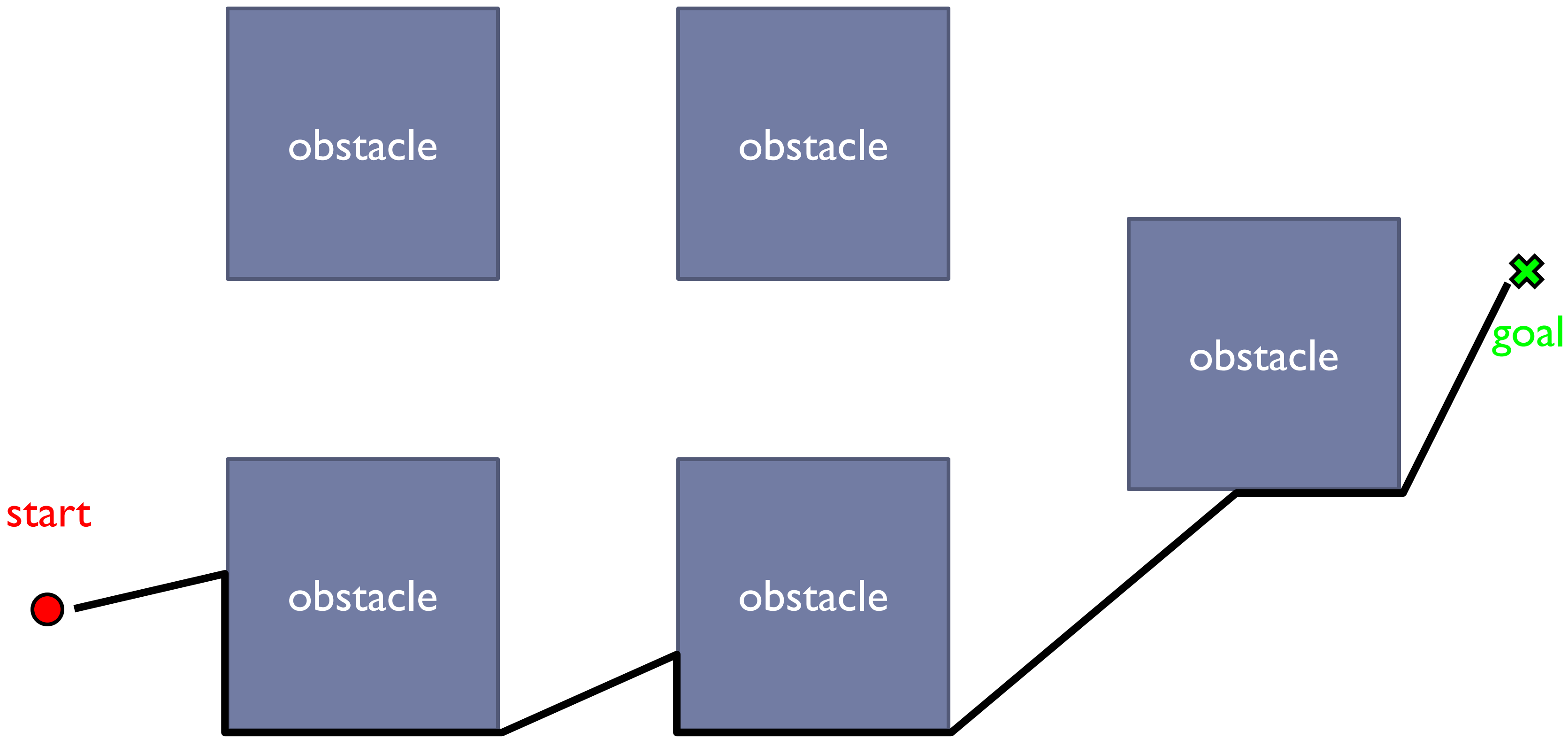
- assumes a perfect contact sensor
- repeat
  - head towards goal
  - if goal is reached then stop
  - if an obstacle is reached then follow the boundary until heading towards the goal is again possible

Bug Zero



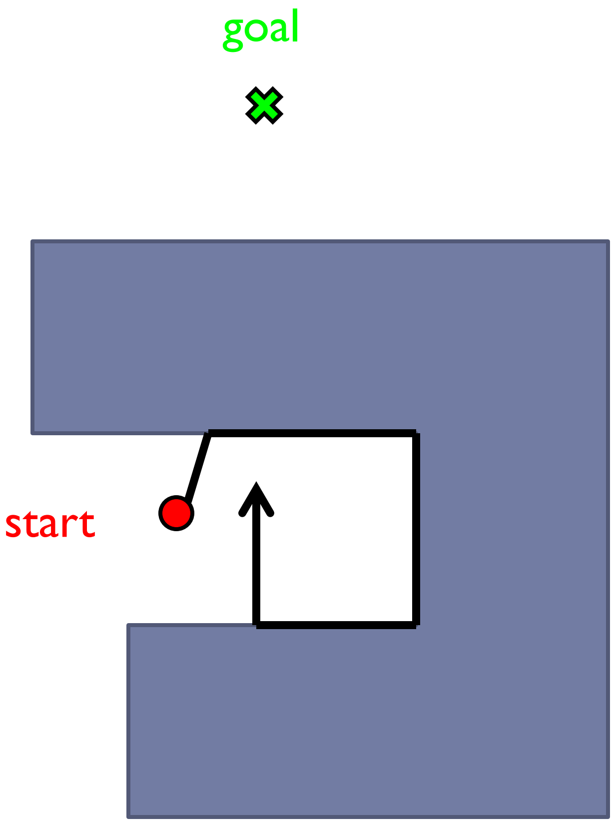
obstacle
obstacle
a right turning bug
obstacle
start
goal
freespace
obstacle
obstacle

Bug Zero



Bug Zero

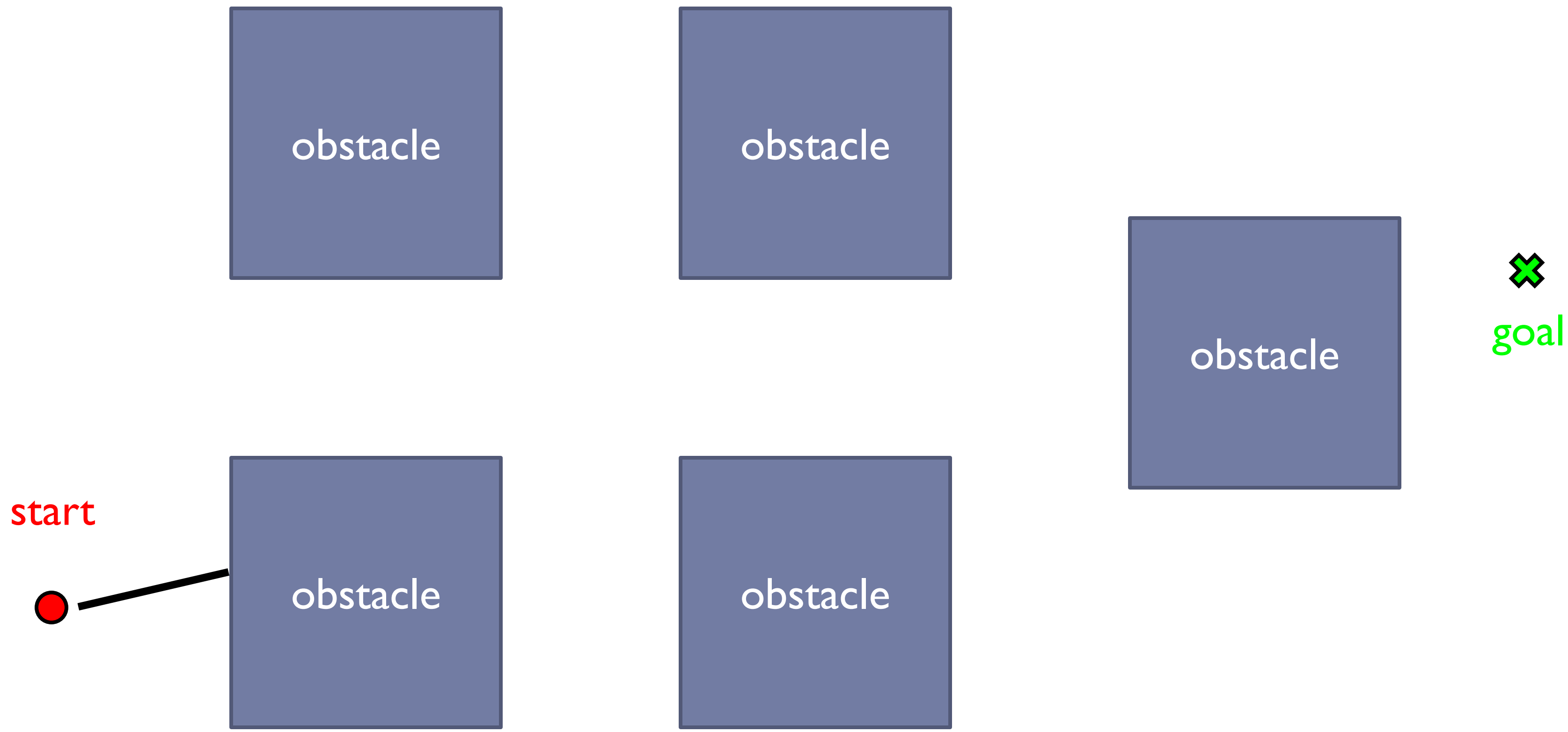
- not guaranteed to reach the goal



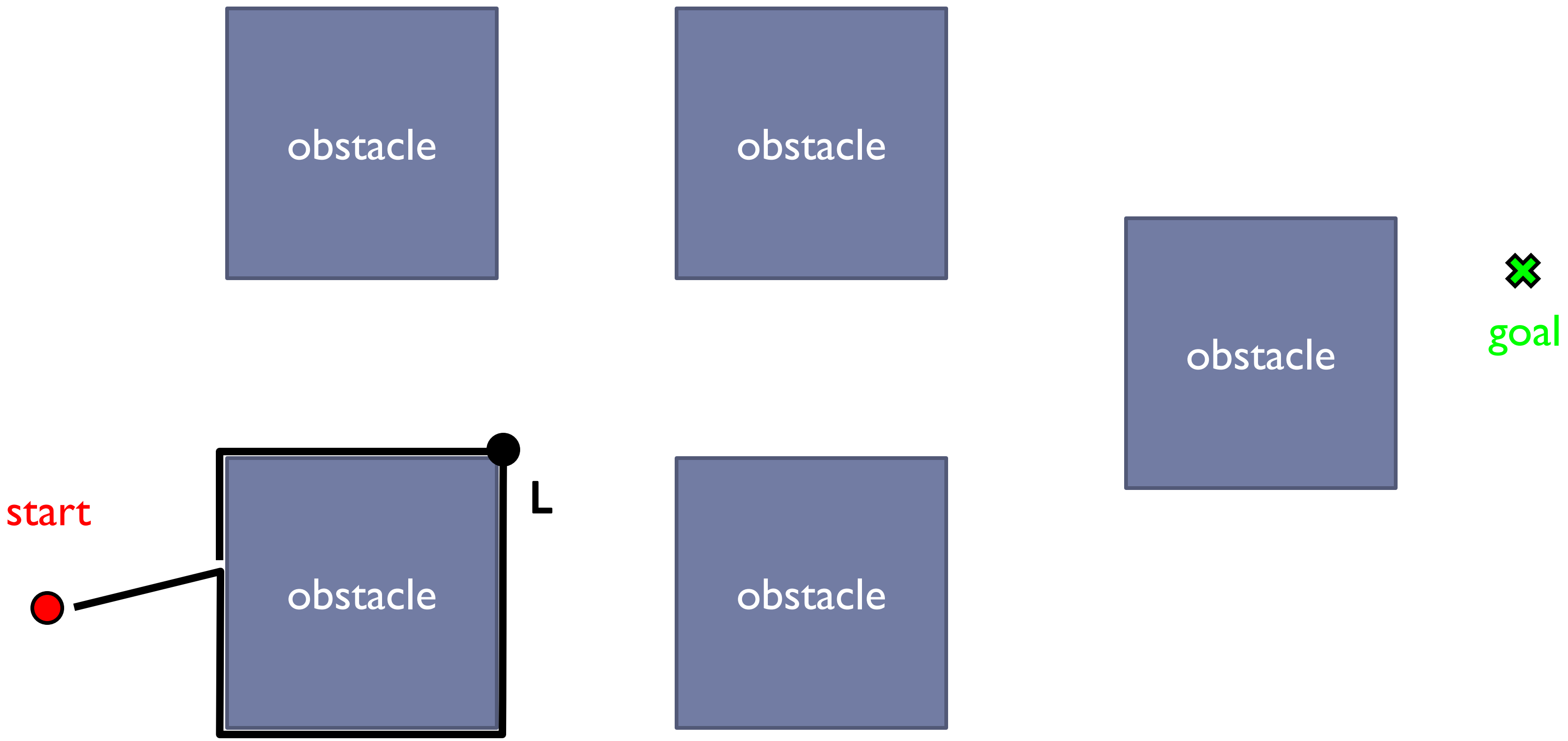
Bug One

- assumes a perfect contact sensor
- repeat:
  - head toward goal T
  - if goal is reached then stop
  - if an obstacle is reached then
    - remember the point of first contact H (the hit point)
    - follow the boundary of the obstacle until returning to H and remember the point L (the leave point) closest to T from which the robot can depart directly towards T
      - if no such point L exists then the goal is unreachable; stop
    - move to L using the shortest boundary following path

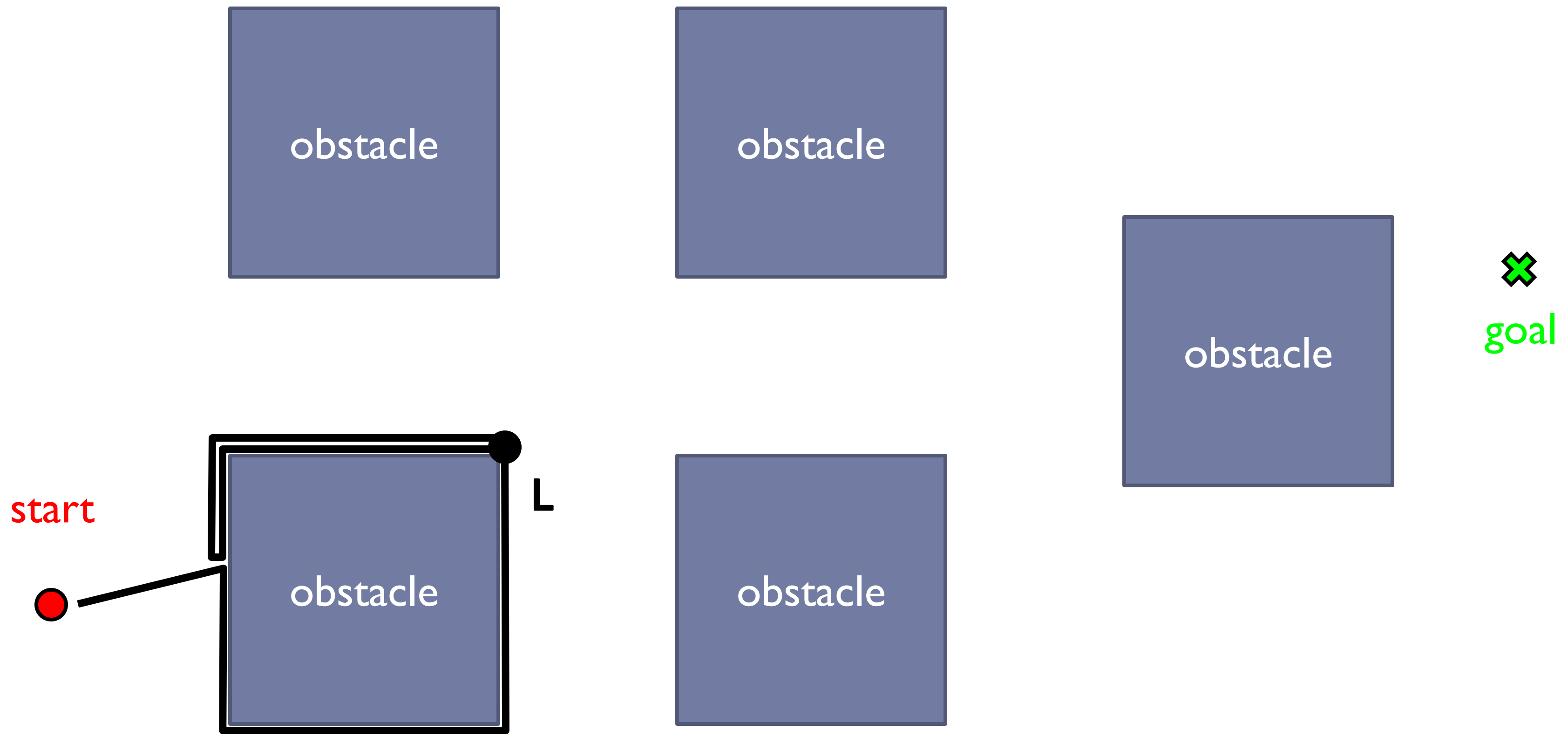
Bug One



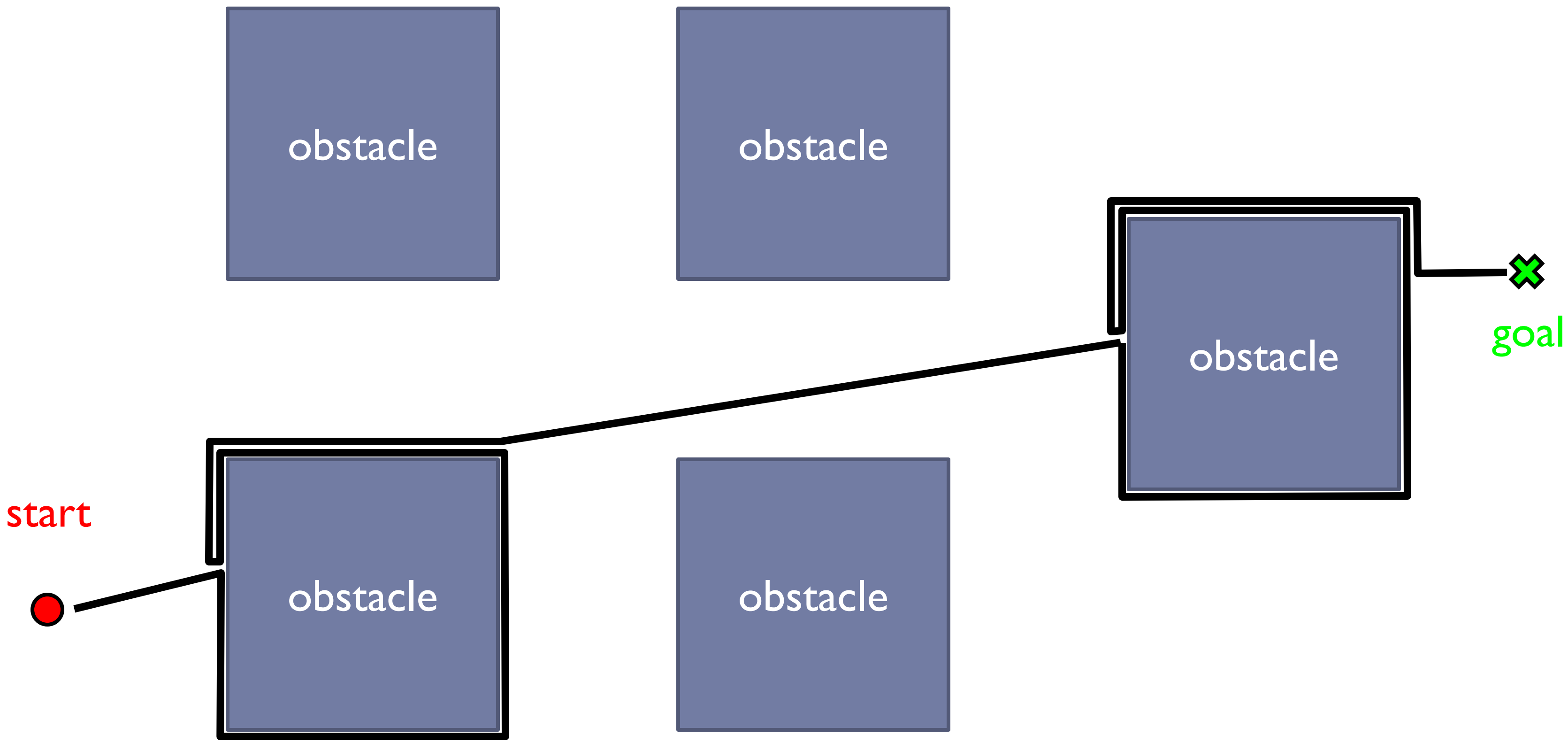
Bug One


obstacle
obstacle
obstacle
goal
L
start
obstacle
obstacle

Bug One

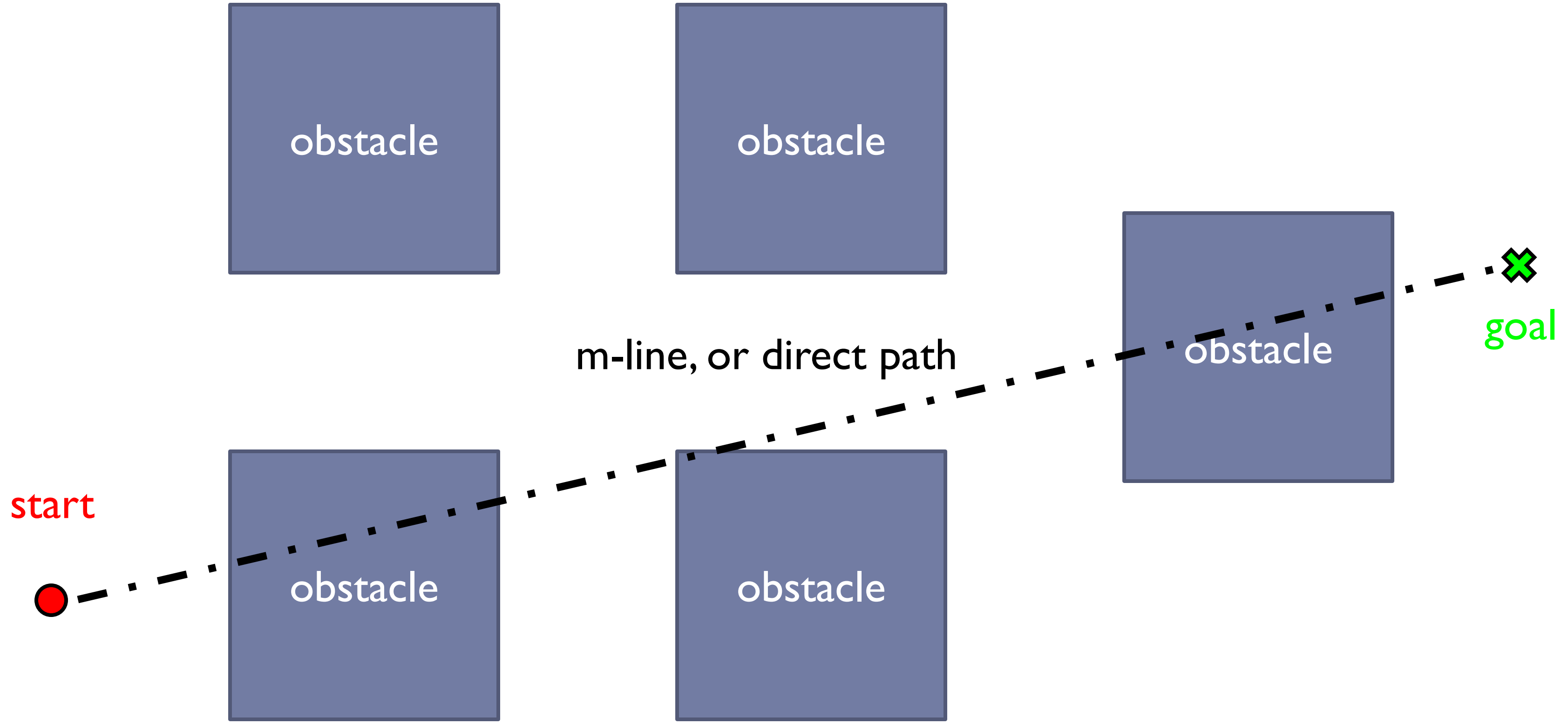

obstacle
obstacle
obstacle
goal
L
start
obstacle
obstacle

Bug One



Bug Two

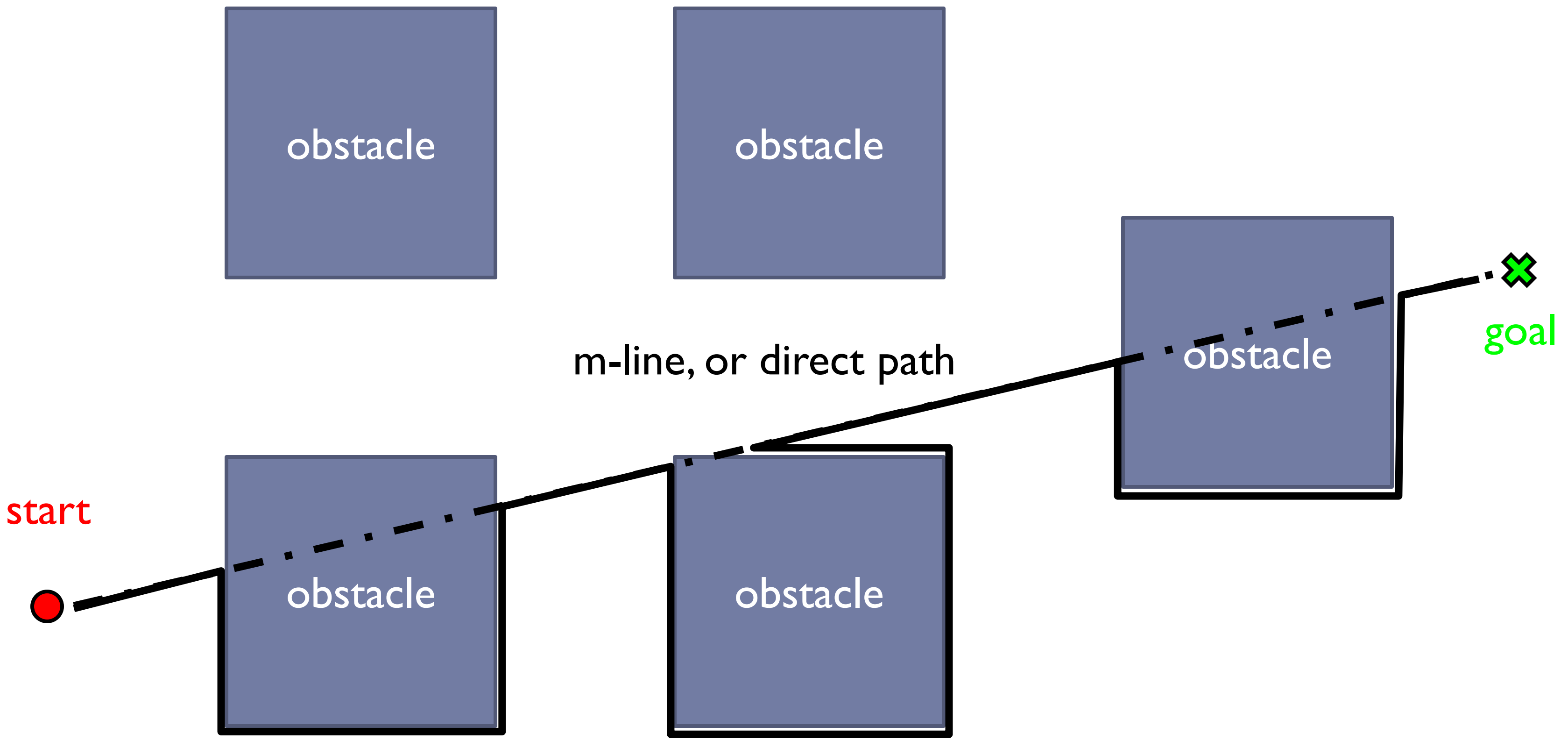
- Bug Two uses a line, called the *m-line*, from the start point to the goal
  - sometimes called the *direct path*



Bug Two

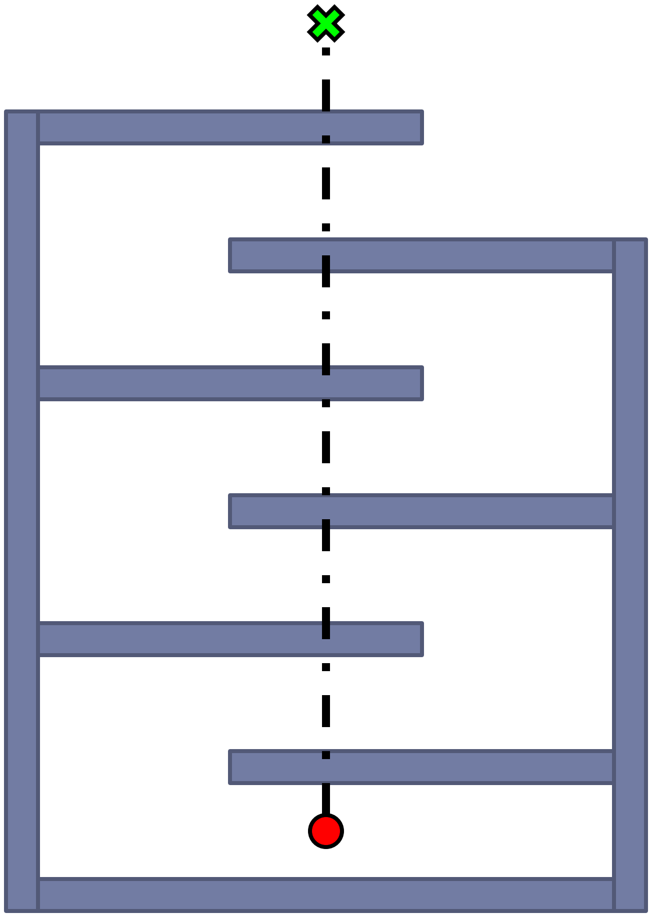
- assumes a perfect contact sensor
- repeat:
  - head toward goal T along the m-line
  - if goal is reached then stop
  - if an obstacle is reached then
    - remember the point of first contact H (the hit point)
    - follow the boundary of the obstacle until the m-line is crossed at a leave point closer to the goal than H
      - if no such point L exists then the goal is unreachable; stop
    - leave the obstacle and head toward T

Bug Two



obstacle
obstacle
obstacle
goal
m-line, or direct path
start
obstacle
obstacle

Bug Two



Bug One versus Bug Two

- Bug One uses exhaustive search
  - it considers all leave points before leaving the obstacle
- Bug Two uses greedy search
  - it takes the first leave point that is closer to the goal

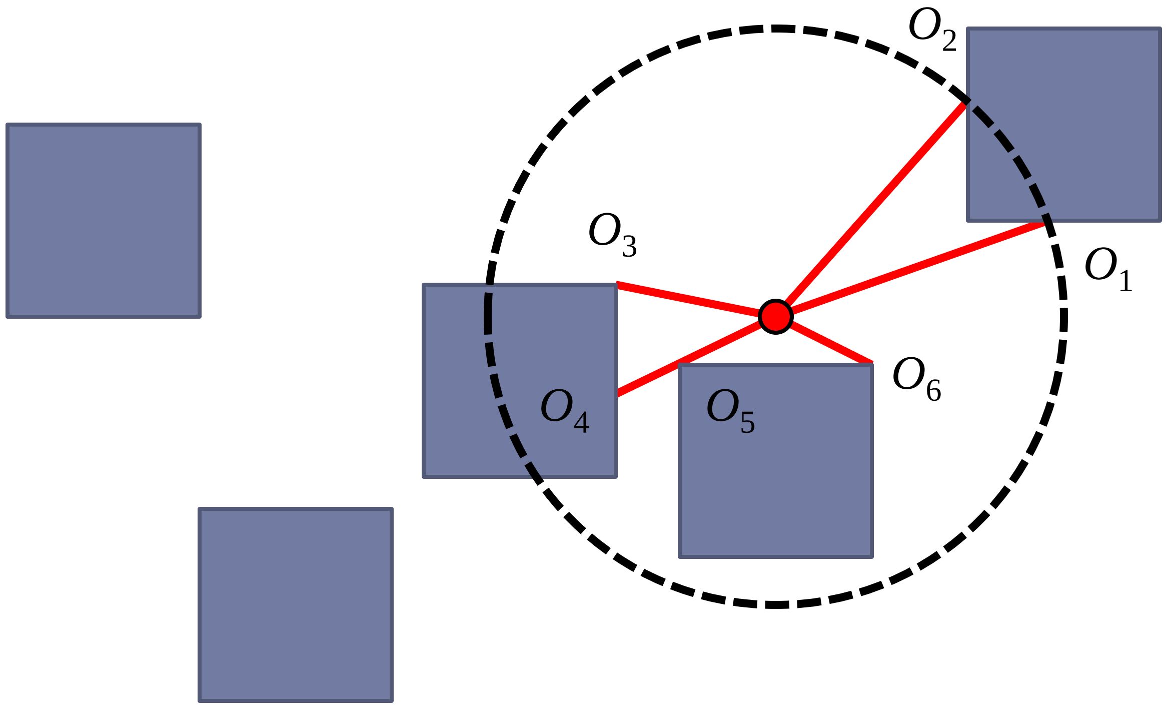
Sensing the Environment

- Bug1 and Bug2 use a perfect contact sensor
- we might be able to achieve better performance if we equip the robot with a more powerful sensor
- a range sensor measures the distance to an obstacle; e.g., laser range finder
  - emits a laser beam into the environment and senses reflections from obstacles
  - essentially unidirectional, but the beam can be rotated to obtain 360 degree coverage

Tangent Bug

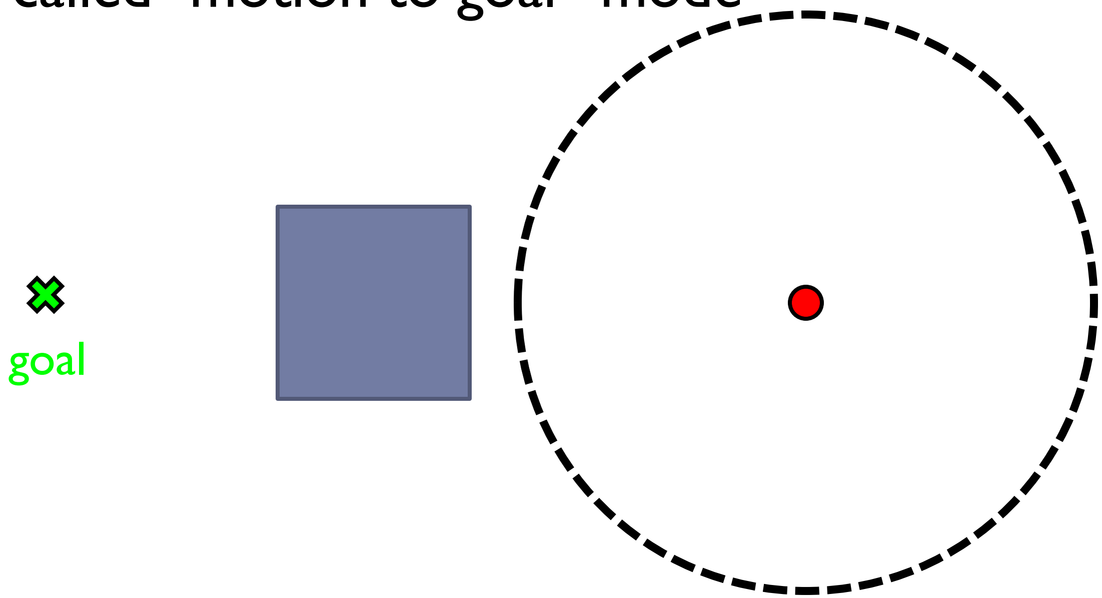
- assumes a perfect 360 degree range finder with a finite range
  - measures the distance $\rho(x, \theta)$ to the first obstacle intersected by the ray from x with angle θ
  - has a maximum range beyond which all distance measurements are considered to be $\rho = \infty$
- the robot looks for discontinuities in $\rho(x, \theta)$

Tangent Bug



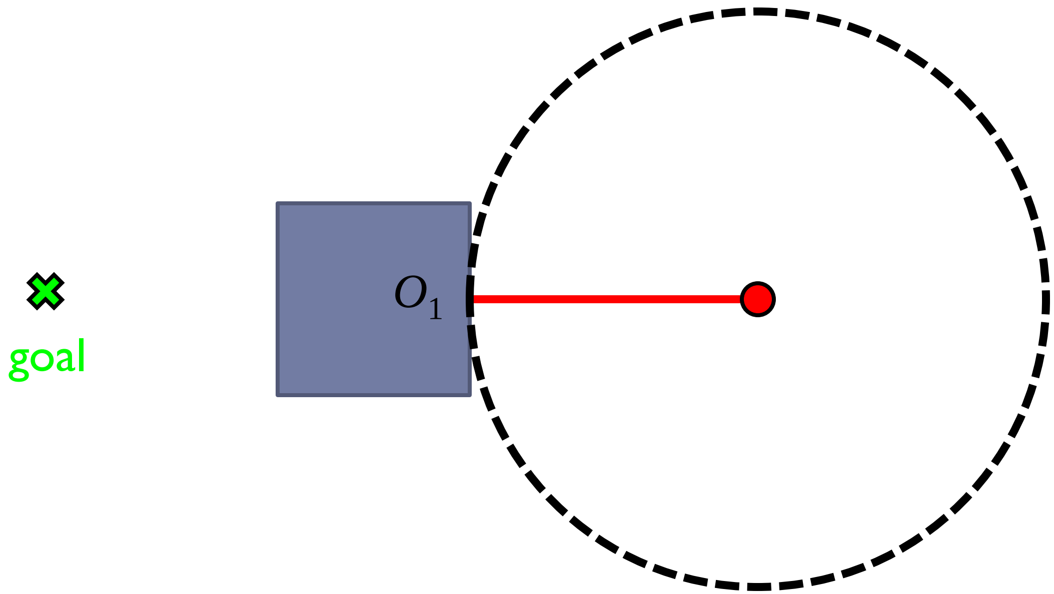
Tangent Bug

- currently, bug thinks goal is reachable so it moves toward the goal
  - called "motion to goal" mode



Tangent Bug

- once the obstacle is sensed, the bug needs to decide how to navigate around the obstacle

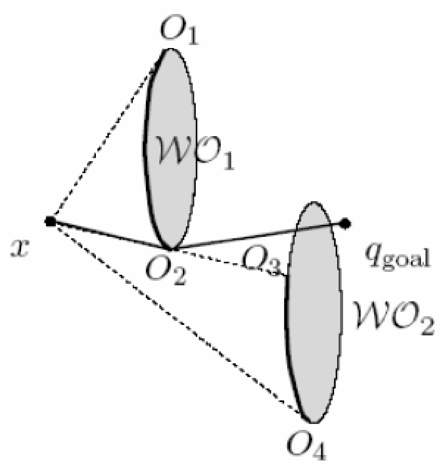


- move towards the sensed point O_i that minimizes the distance $d(x, O_i) + d(O_i, q_{\text{goal}})$ (called the heuristic distance)

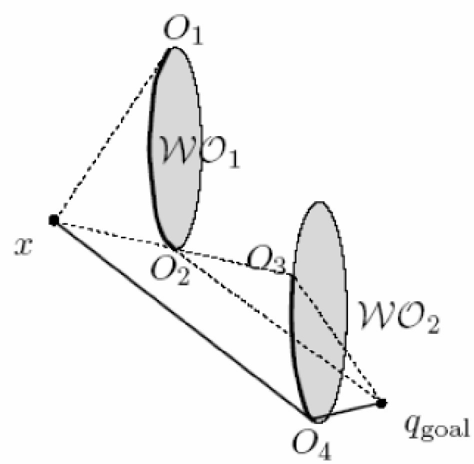
Tangent Bug

Minimize Heuristic Example

At x, robot knows only what it sees and where the goal is,



so moves toward O_2. Note the line connecting O_2 and goal pass through obstacle



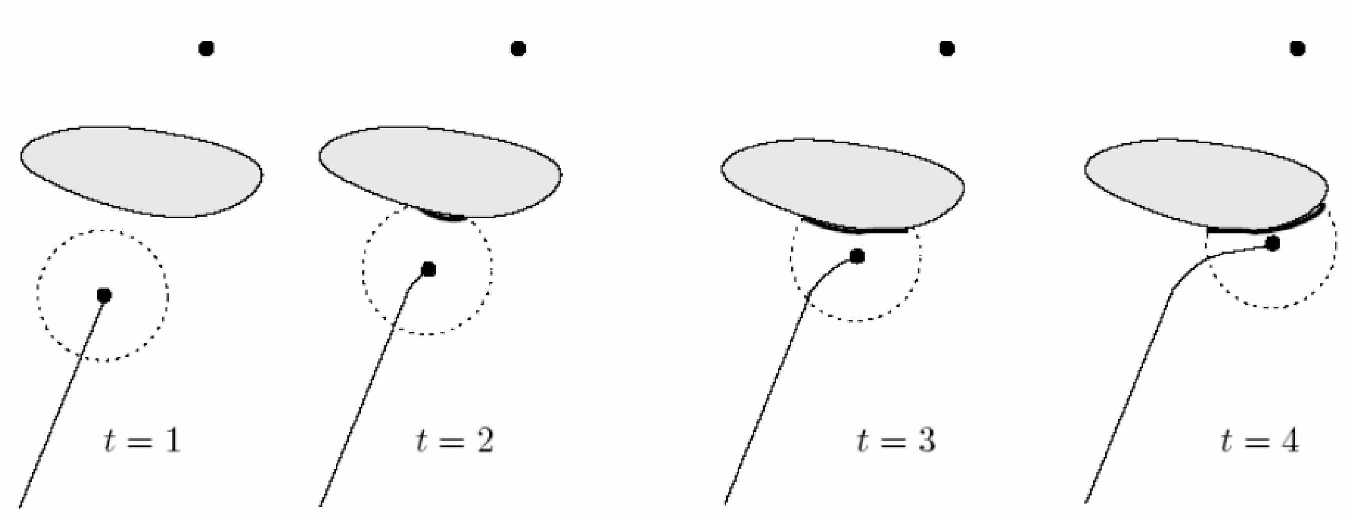
so moves toward O_4. Note some "thinking" was involved and the line connecting O_4 and goal pass through obstacle

Choose the pt O_i that minimizes $d(x,O_i) + d(O_i,q_{goal})$

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

Tangent Bug

Motion To Goal Example

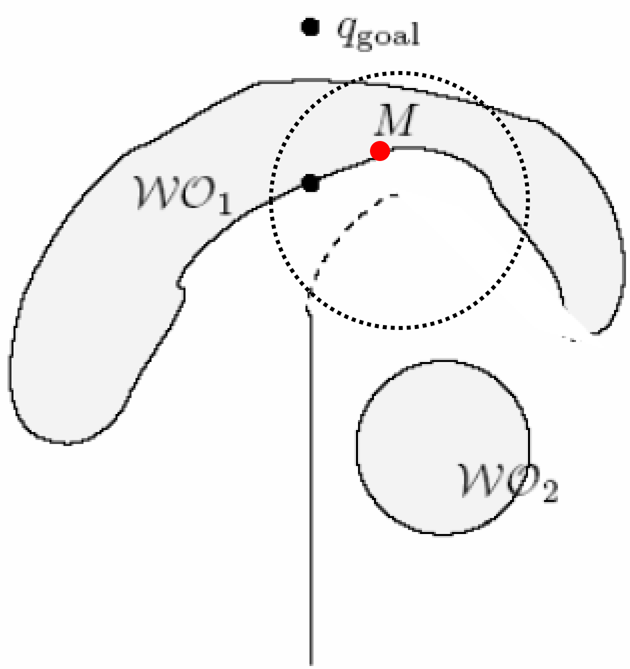


Choose the pt O_i that minimizes $d(x,O_i) + d(O_i,q_{goal})$

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

Tangent Bug

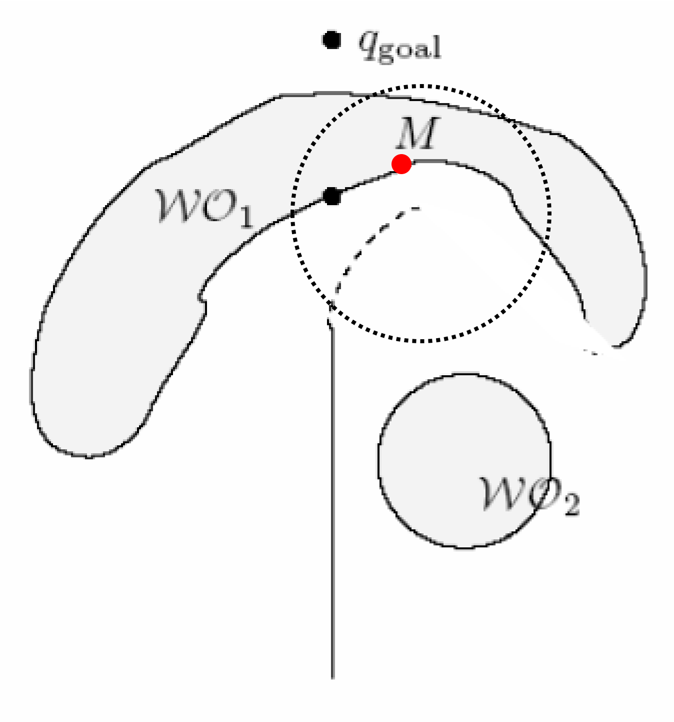
- problem with concave obstacles
  - eventually the robot reaches a point where $d(x, O_i) + d(O_i, q_{\text{goal}})$ starts to increase



  - once this happens, the robot switches to "boundary following" mode

Tangent Bug

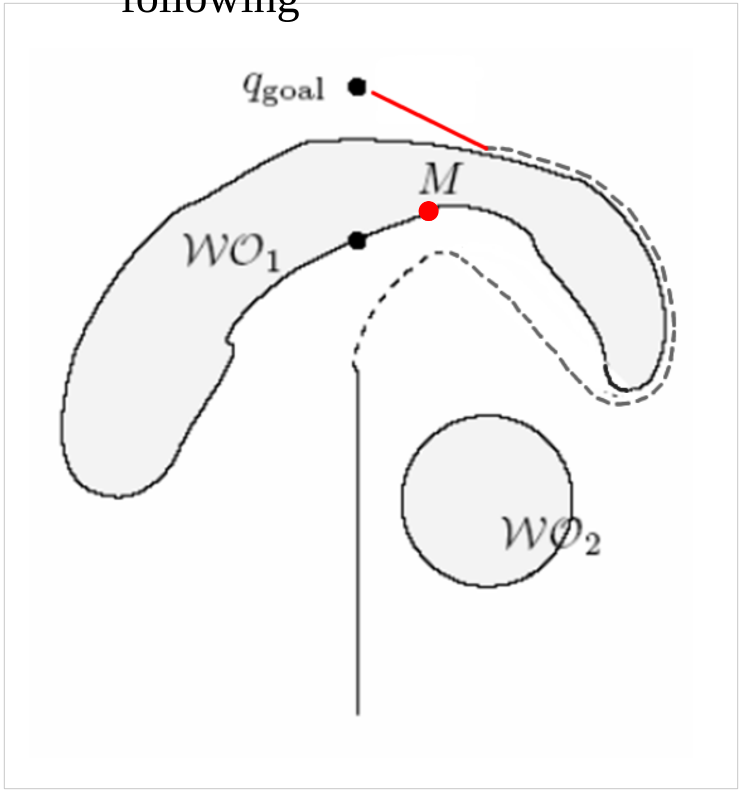
- trick here is that the robot remembers the distance $d_{\text{following}}$ between M and q_{goal}



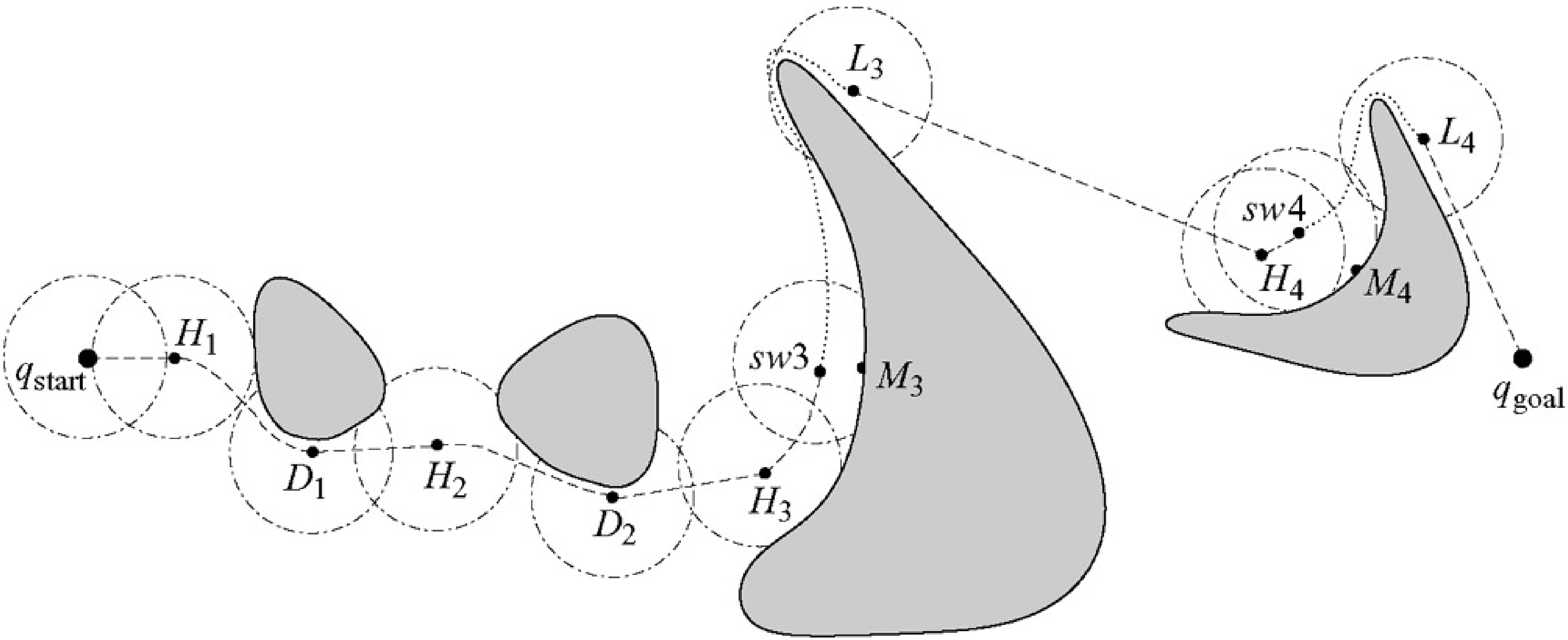
  - M is the point on the blocking obstacle that had the shortest distance to the goal when the heuristic distance started to increase

Tangent Bug

- the robot returns to "motion to goal" mode as soon as it reaches a point where the distance between a sensed point and the goal or the distance between the robot location and the goal is less than $d_{\text{following}}$



Tangent Bug



Tangent Bug

- full details
  - *Principles of Robot Motion: Theory, Algorithms, and Implementations*
  - http://www.library.yorku.ca/find/Record/2154237
- nice animation
  - http://www.cs.cmu.edu/~motionplanning/student_gallery/2006/st/hw2pub.htm