Lesson 1
Introduction to Path Planning
Graph Searches: BFS and DFS

DASL Summer Program

Path Planning

References: http://robotics.mem.drexel.edu/mhsieh/Courses/MEM380I/index.html
http://dasl.mem.drexel.edu/Hing/BFSDFSTutorial.htm
http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/16311/www/current/syllabus.html
http://www.cs.cmu.edu/~motionplanning/syllabus%202010a-2.html

Course Description

- 6-week course to introduce path planning and related algorithms
- Course consists of lectures with practical implementations using Matlab and the NXT Tribot Robot
- Course will culminate with a path planning challenge using a robot to navigate a course with obstacles

Syllabus

- Week 1 – BFS and DFS
- Week 2 – A* and Dijkstra's Algorithm
- Week 3 – Potential Fields
- Week 4 – Randomized Path Planning
- Week 5 – SLAM
- Week 6 – Path Planning Challenge

Homework

- One assignment per week
- Matlab implementation of planning technique or algorithm
- Should require ~2 hours to complete

Student Introductions

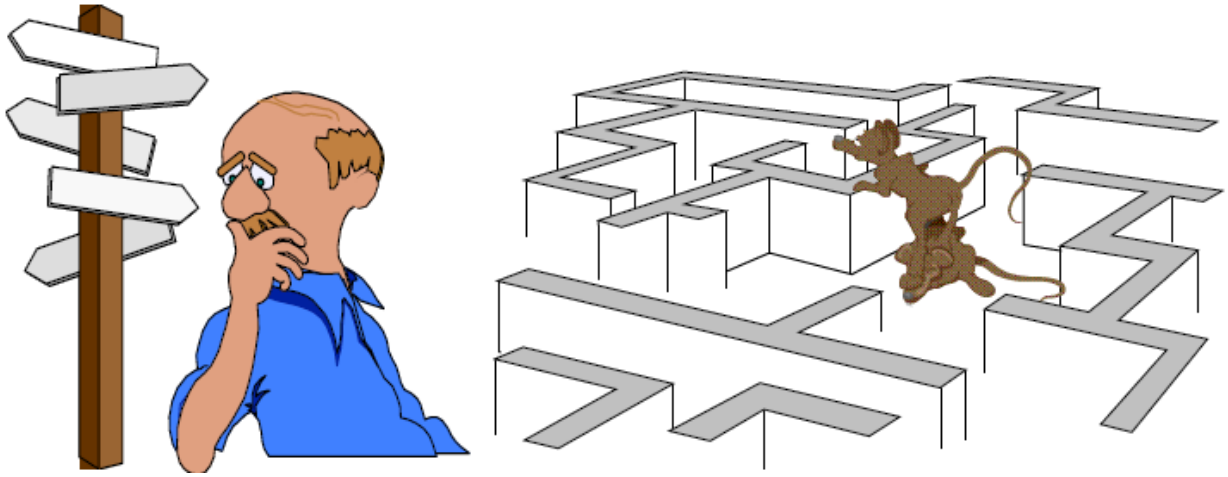
- Name
- Major, School, Year
- Hometown
- Hobbies
- Something interesting about you
- What you hope to get out of the course

Instructor

- Christopher Korpela
- cmk325@drexel.edu, christopher.korpela@gmail.com
- 1st year PhD Student in EE
- Bel Air, MD
- Soccer and running

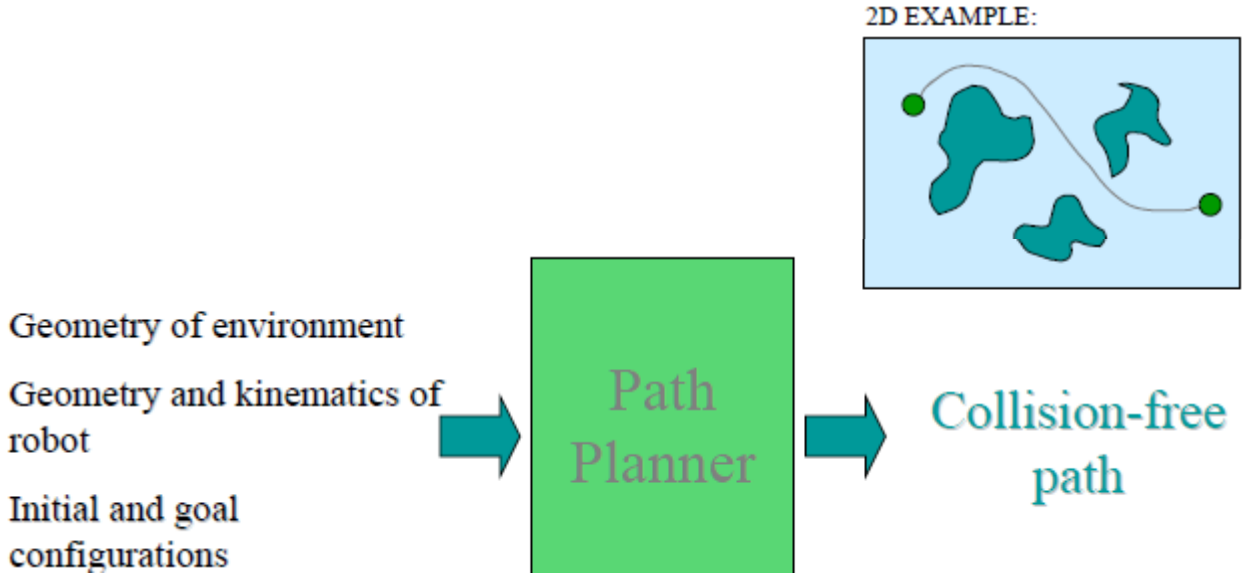
What is Path (Motion) Planning?

- Process of detailing a task into discrete motions
- Example: navigating a mobile robot from a start point to a distant end point



Basic Path Planning

- Problem Statement: Compute a continuous sequence of collision-free robot configurations connecting the initial and goal configurations



Live Motion Planning Experiments

- Person 1 walks through some obstacles
- Person 1, looking at Person 2, directs Person 2 through obstacles
- Person 1, looking at Person 2 with eyes closed, directs Person 2 through obstacles
- Person 1, looking at a map and not Person 2, whose eyes are still closed, directs Person 2 through obstacles
- Person 1, looking at an object and Person 2, whose eyes are closed, directs Person 2 to grab an obstacle
- How do we do the last experiment with a map?

What did we assume?

- Perfect sensors?
  - What information
  - Uncertainty
- Perfect control?
  - What controls?
  - Uncertainty
- Perfect thinking?
  - Knowledge of the world? Complete?
  - Processing the world? Everything?

Path Planning Considerations

- Planning tasks
  - Navigation
  - Coverage
  - Localization
  - Mapping
- Properties of the robot
  - degrees of freedom
  - holonomic or not
  - kinematic vs. dynamic
- Properties of Algorithms
  - Optimality
  - Computational complexity
  - Completeness
- Resolution completeness
- Probabilistic completeness
  - Online vs. offline
  - Sensor-based vs. not
  - Feedback or not

Basics: Metrics

- There are many different ways to measure a path:
  - Time
  - Distance traveled
  - Expense
  - Distance from obstacles
  - Etc…

Mathematical Rigor

Symbol	Meaning
$\exists$	there exists
$\forall$	for all
∞	infinity
$\in$	element
$\notin$	not in
s.t.	such that
$\mathbb{R}$	real numbers
$\mathbb{R}^m$	m-dimensioned real numbers
$\bigcup$	union
$\bigcap$	intersection
$\setminus$	set difference
$\Rightarrow$	implies. $p \rightarrow q$ is p implies q
$\Leftarrow$	implies. $q \rightarrow p$ is q implies p
$\Longleftrightarrow$	if and only if
S^1	a circle
∇	gradient
D	differential or distance to closest obstacle (depending on context)
d_i	distance to obstacle i in either the workspace or configuration space (depending on context)
$d(x, y)$	distance between the two points x and y
Null	null space

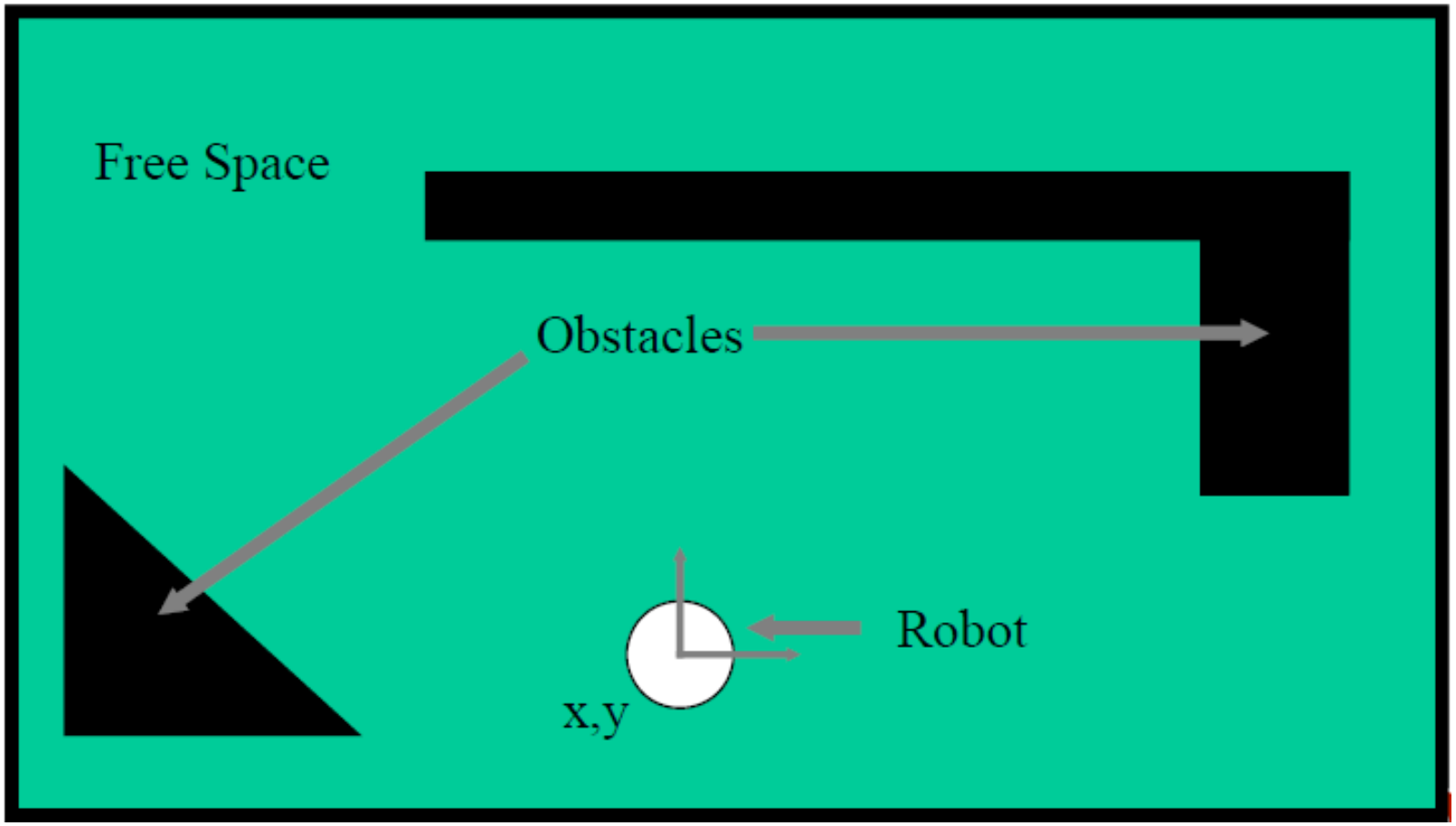
J	Jacobian
Γ	Christoffel symbol
RM	roadmap
$\mathcal{W}$	workspace
$\mathcal{Q}$	configuration space
$\mathcal{Q}_{\text{free}}$	free space
$x(k)$	state at time k
$\|\|x\|\|$	norm of x
$\subseteq$	subset of
$\subset$	strict subset of
cl(A)	closure of A
T^n	n-dimensional torus
S^n	n-dimensional sphere in $\mathbb{R}^{n+1}$
$SO(n)$	special orthogonal group
$SE(n)$	special Euclidean group
$B_\epsilon(q)$	open ball of radius ϵ centered at q
Df	differential of f
∇f	gradient of f
∇	affine connection
$\nabla_{Y_1} Y_2$	covariant derivative of Y_2 with respect to Y_1
C^0	continuous
C^n	n times differentiable
$\langle x, y \rangle$	inner product of x and y
$\mathcal{I}$	identity matrix
atan2(y, x)	returns angle to (x, y) in the plane in range $[-\pi, \pi)$
$T_x\mathcal{M}$	tangent space of $\mathcal{M}$ at x
$T\mathcal{M}$	tangent bundle of $\mathcal{M}$
$[f, g]$	Lie bracket of vector fields f, g
$\overline{\text{Lie}}(\mathcal{G})$	the Lie algebra of a set of vector fields $\mathcal{G}$
$\overline{\mathcal{D}}$	involutive closure of the distribution $\mathcal{D}$
$\mathcal{U}_\pm$	control set positively spanning $\mathbb{R}^m$
$\mathcal{U}_+$	control set spanning $\mathbb{R}^m$
$\langle Y_1 : Y_2 \rangle$	the symmetric product of vector fields Y_1 and Y_2

The World consists of

- Obstacles
  - Already occupied spaces of the world
  - In other words, robots can’t go there
- Free Space
  - Unoccupied space within the world
  - Robots “might” be able to go here
  - To determine where a robot can go, we need to discuss what a *Configuration Space* is

Robots

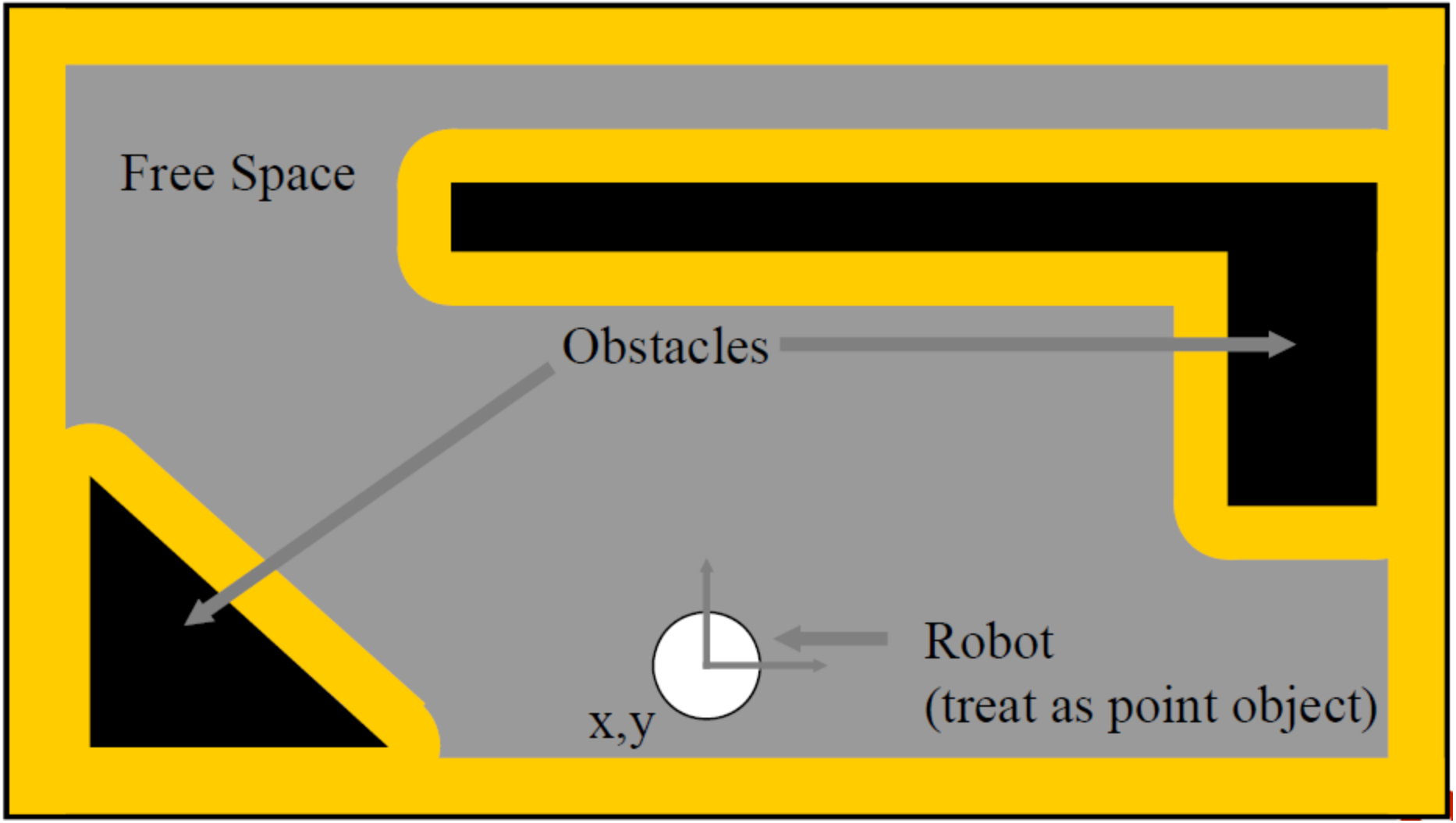
Robot and Environment



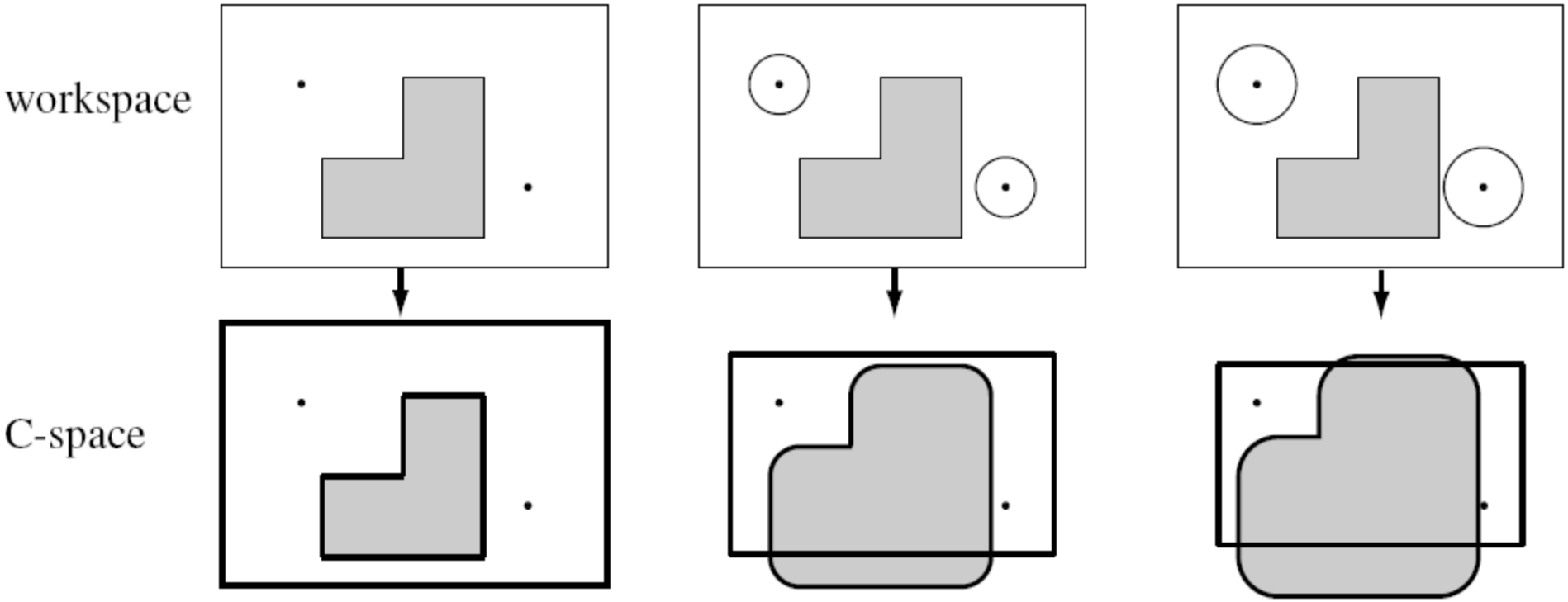
The Configuration Space

- What it is
  - A set of “reachable” areas constructed from knowledge of both the robot and the world
- How to create it
  - First abstract the robot as a point object. Then, enlarge the obstacles to account for the robot’s footprint and degrees of freedom
  - In our example, the robot was circular, so we simply enlarged our obstacles by the robot’s radius (*note the curved vertices*)

Configuration Space: Accommodate Robot Size



Trace Boundary of Workspace



Motion Planning Statement

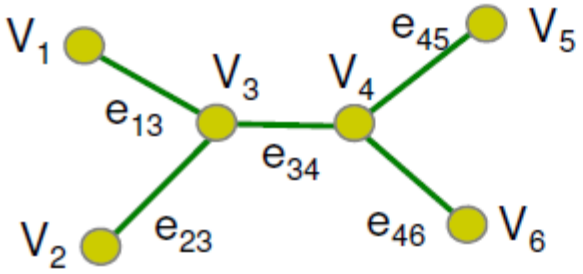
- If **W** denotes the robot's workspace,
- And $\mathbf{C_i}$ denotes the i'th obstacle,
- Then the robot's free space, **FS**, is defined as:

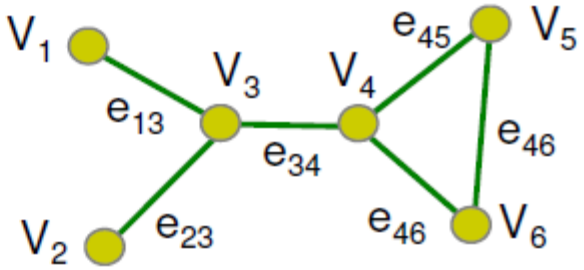
$$\mathbf{FS = W \setminus (\bigcup C_i)}$$

- And a path $\mathbf{c \in C^0}$ is $\mathbf{c : [0,1] \rightarrow FS}$
- where **c(0)** is $\mathbf{q_{start}}$ and **c(1)** is $\mathbf{q_{goal}}$

Motion Planning as Graph Searches

- Abstract representation of a *collection of items*
- A collection of vertices & edges

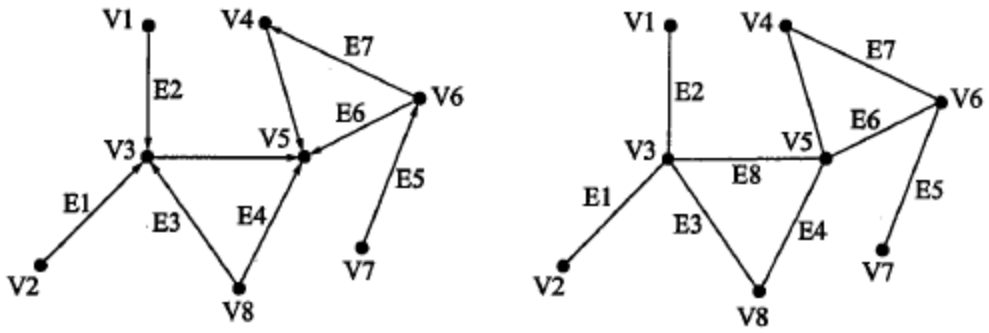




- Terminology
  - Adjacency
  - Paths and Cycles
  - Degree of a vertex
  - Fully connected graphs

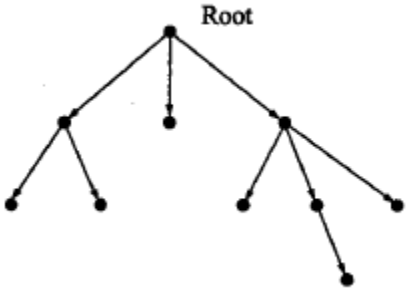
Types of Graphs

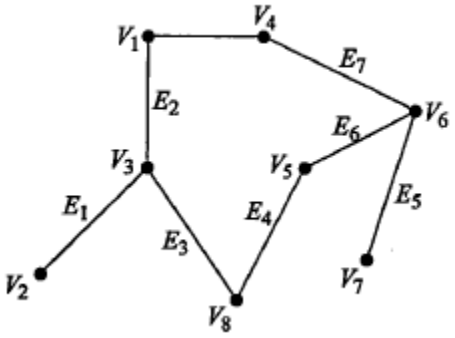
- Directed vs. Undirected
- Weighted vs. Unweighted



Types of Graphs

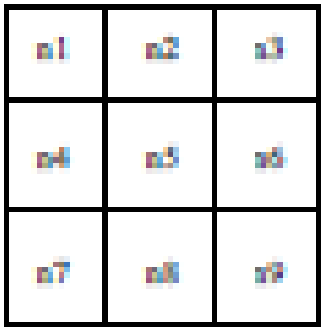
- Cyclic vs. Acyclic
- Trees

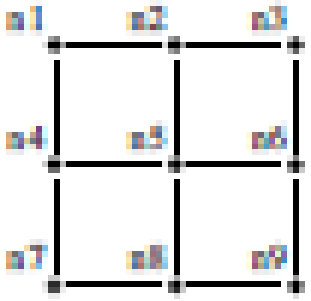


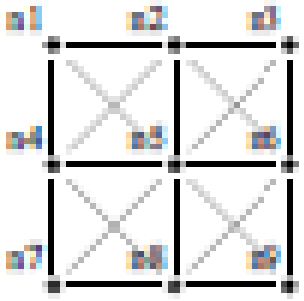


Grids


n1
n2
n3
n4
n5
n6
n7
n8
n9

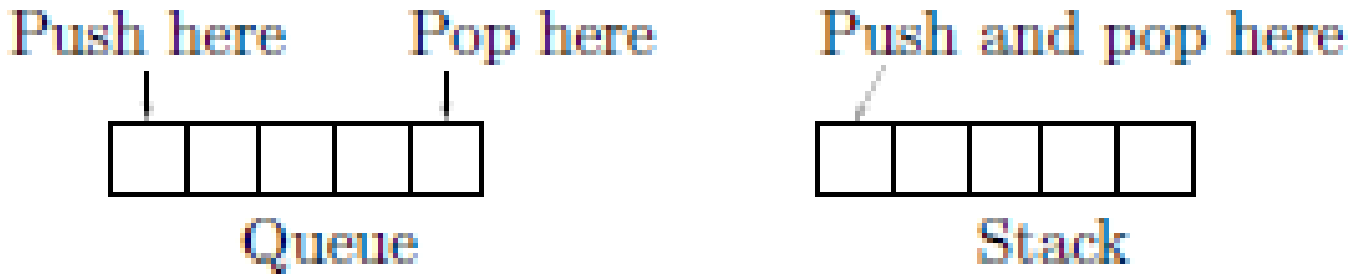

n1
n2
n3
n4
n5
n6
n7
n8
n9


n1
n2
n3
n4
n5
n6
n7
n8
n9

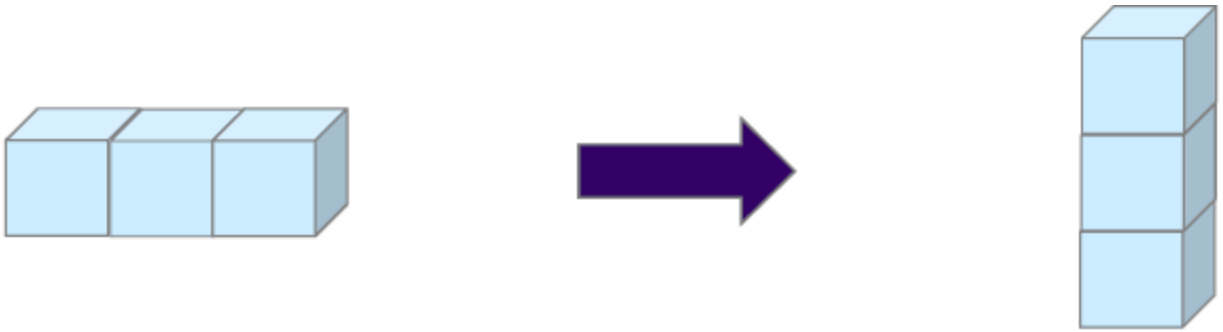

n1
n2
n3
n4
n5
n6
n7
n8
n9

Stacks and Queues

- Stack: First in, Last out (FILO)
- Queue: First in, First out (FIFO)



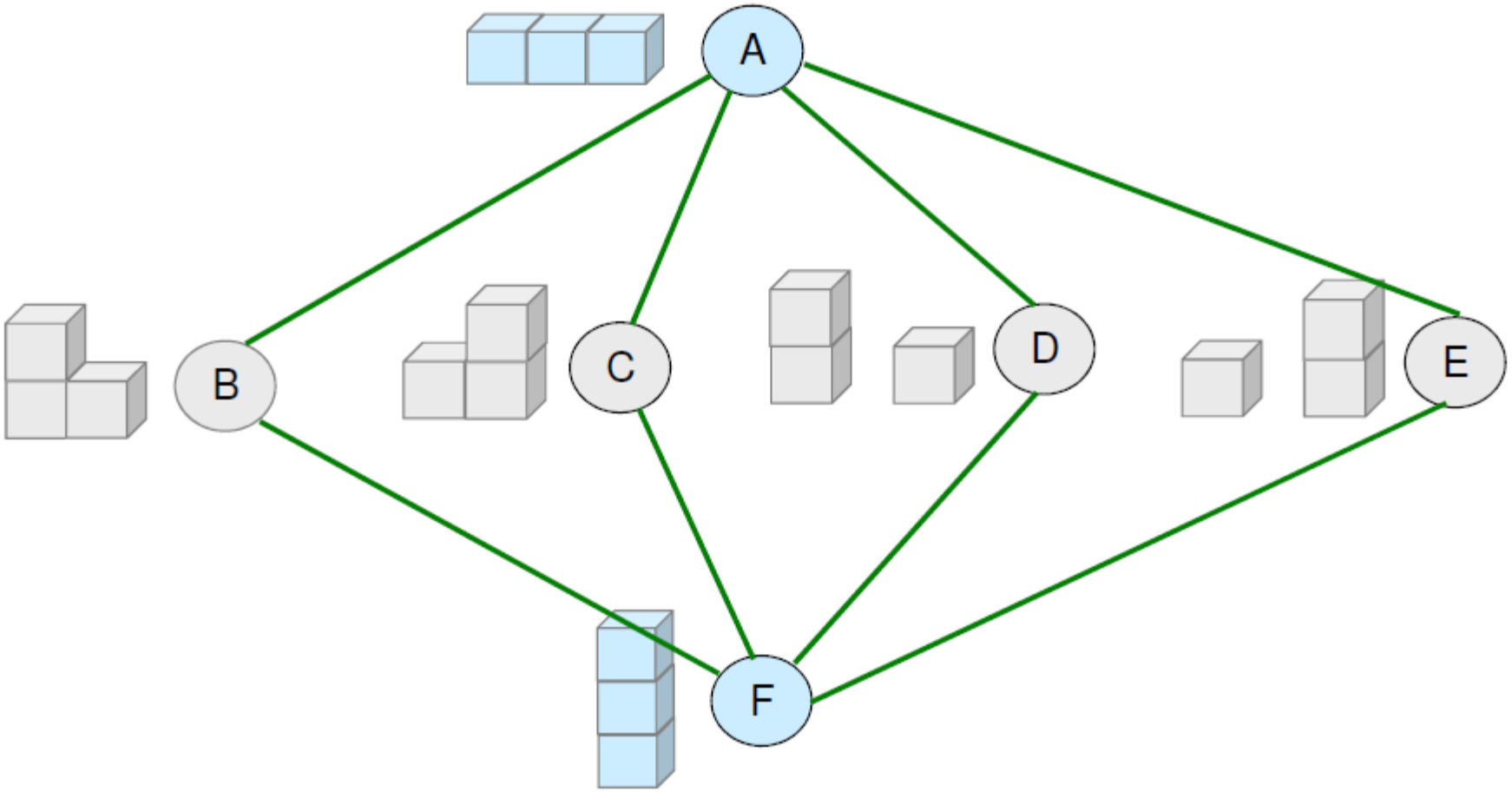
Graph Representation: Example



- Objective: We want to represent all possible ways to get from configuration A to B using a graph

HOW?

Graph Representation: Example



Search in Path Planning

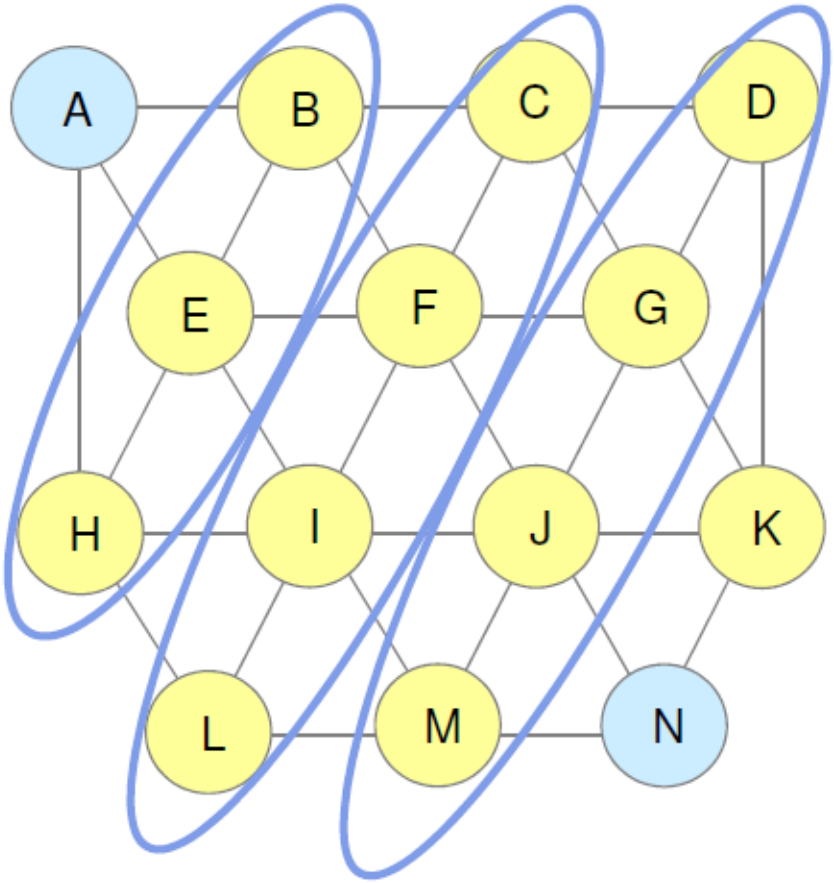
- Find a path between two locations in an *unknown, partially known, or known environment*
- Search Performance
  - Completeness
  - Optimality
  - Operating Costs
  - Space Complexity
  - Time Complexity

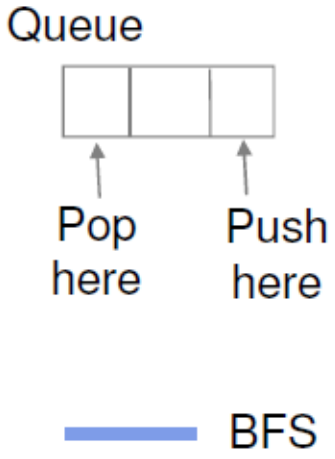
Search

- Uninformed Search
  - Use no information obtained from the environment
  - Blind search: BFS (Wavefront), DFS
- Informed Search
  - Use evaluation function
  - More efficient
  - Heuristic Searchers: A*, D*, etc.

Uniformed Search: BFS

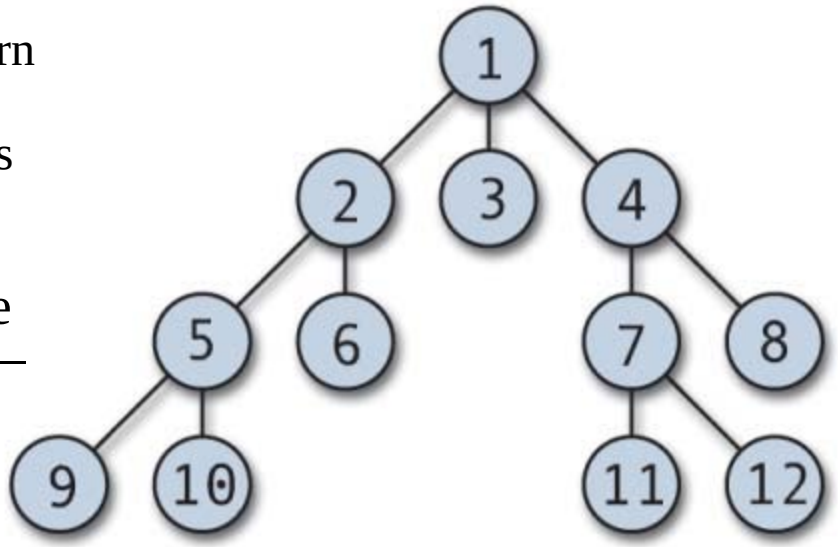
- Graph search from A to N





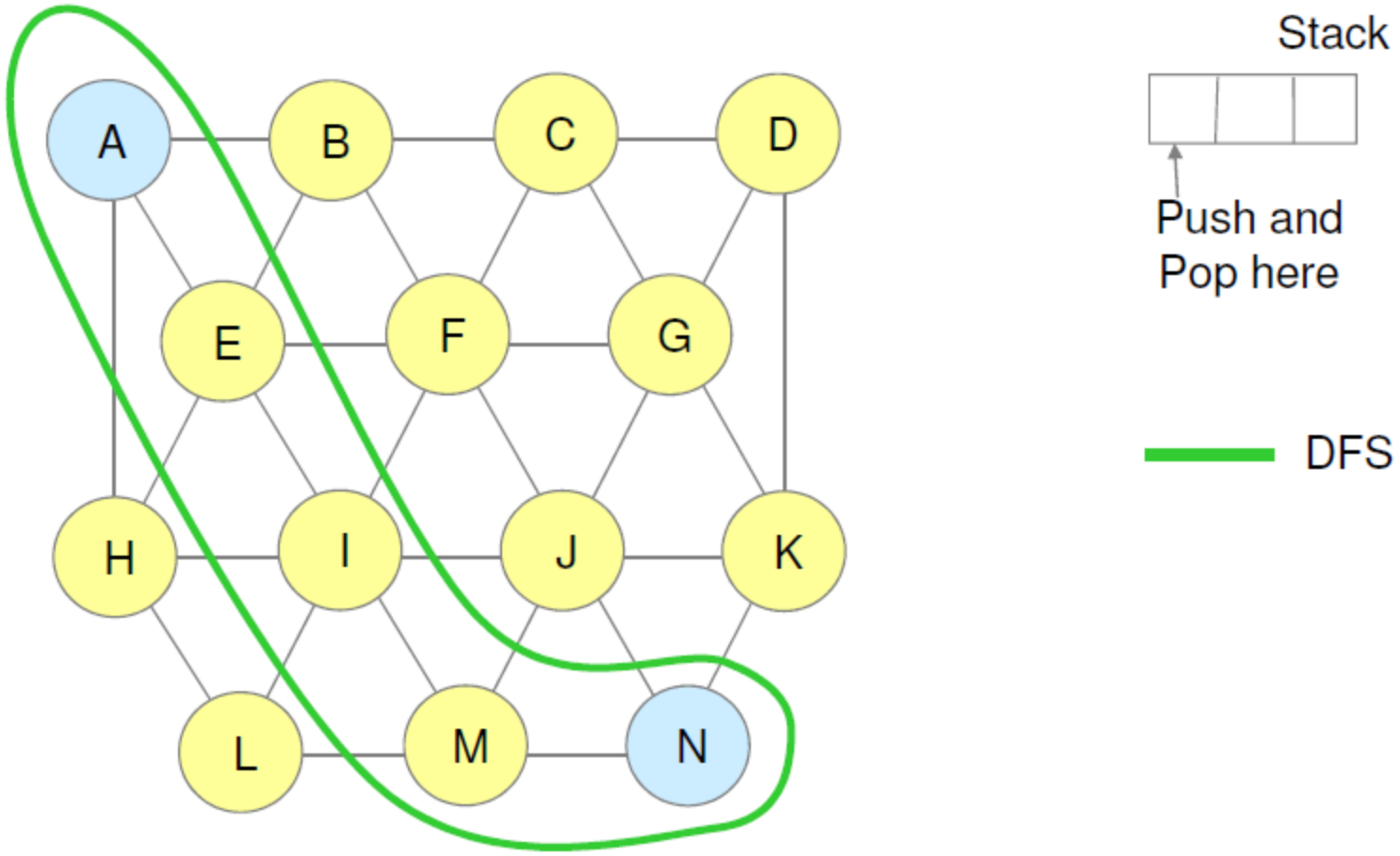
Breadth First Search

1. Enqueue the root node.
2. Dequeue a node and examine it.
   - If the element sought is found in this node, quit the search and return a result.
   - Otherwise enqueue any successors (the direct child nodes) that have not yet been discovered.
3. If the queue is empty, every node on the graph has been examined – quit the search and return "not found".
4. If the queue is not empty, repeat from Step 2.



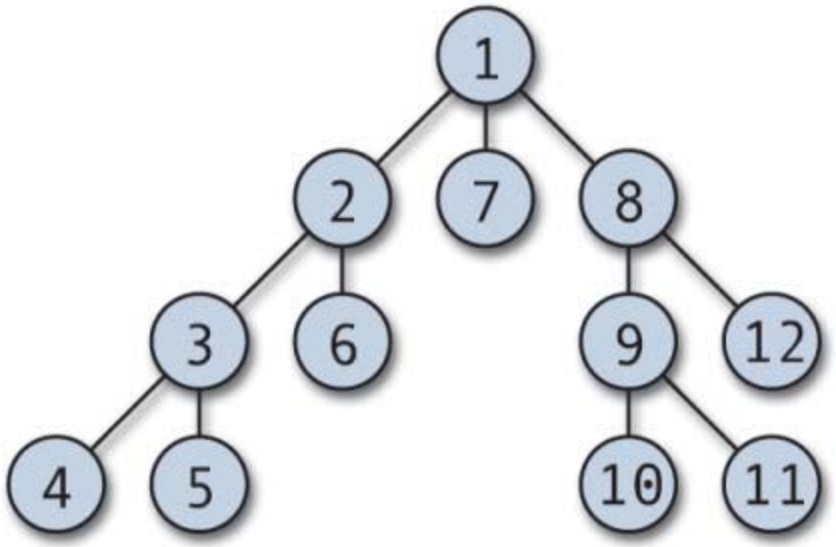
Uninformed Search: DFS

- Graph search from A to N

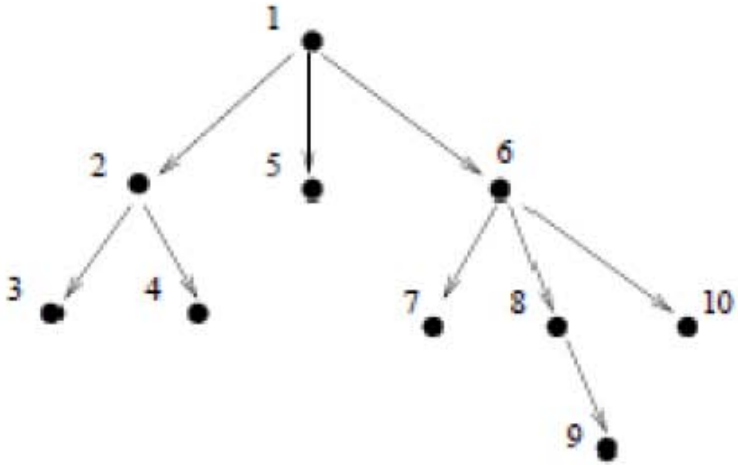


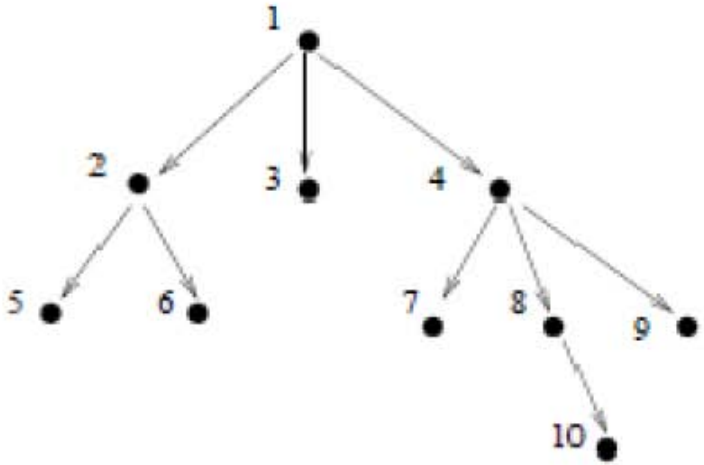
Depth First Search

```
algorithm dft(x)
   visit(x)
   FOR each y such that (x,y) is an edge
      IF y was not visited yet
         THEN dft(y)
```



Depth First and Breadth First Searches





DASL BFS-DFS Tutorial

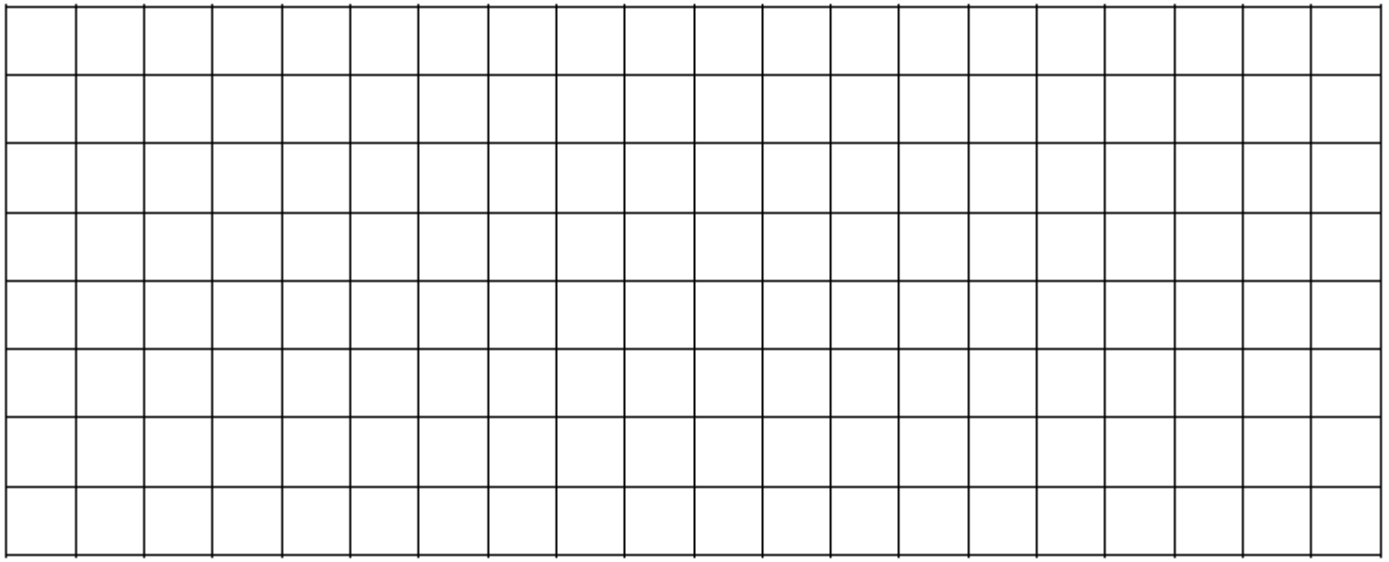
- http://dasl.mem.drexel.edu/Hing/BFSDFSTutorial.htm
- Download m-files
- Change obstacle configuration

The Wavefront Planner

- A common algorithm used to determine the shortest paths between two points
  - In essence, a breadth first search of a graph
- For simplification, we'll present the world as a two-dimensional grid
- Setup: Label free space with 0
- Label start as START
- Label the destination as 2

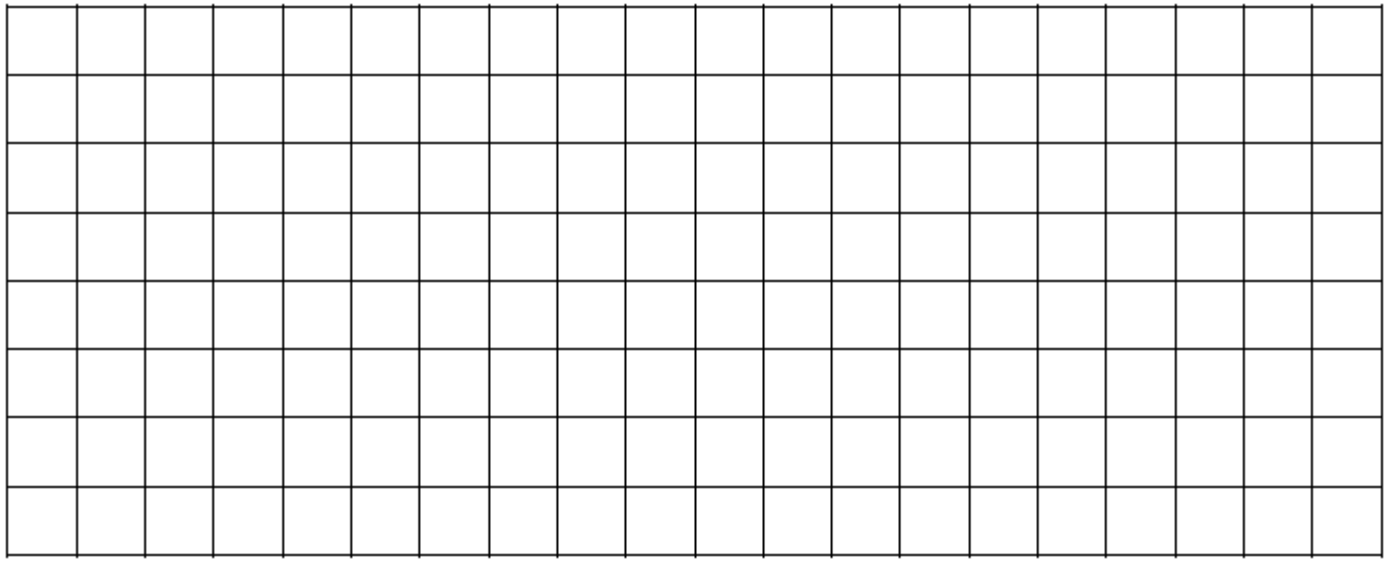
Representations

- World Representation
  - You could always use a large region and distances
  - However, a grid can be used for simplicity



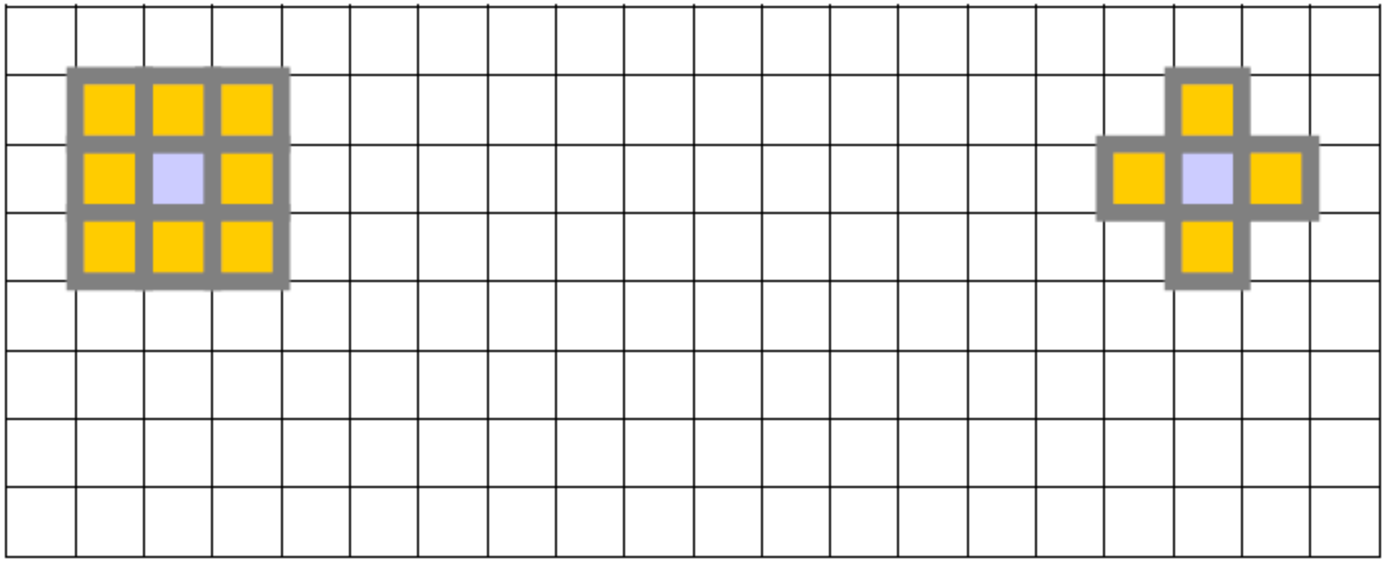
Representations: A Grid

- Distance is reduced to discrete steps
  - For simplicity, we'll assume distance is uniform
- Direction is now limited from one adjacent cell to another
  - Time to revisit Connectivity



Representations: Connectivity

- 8-Point Connectivity 4-Point Connectivity

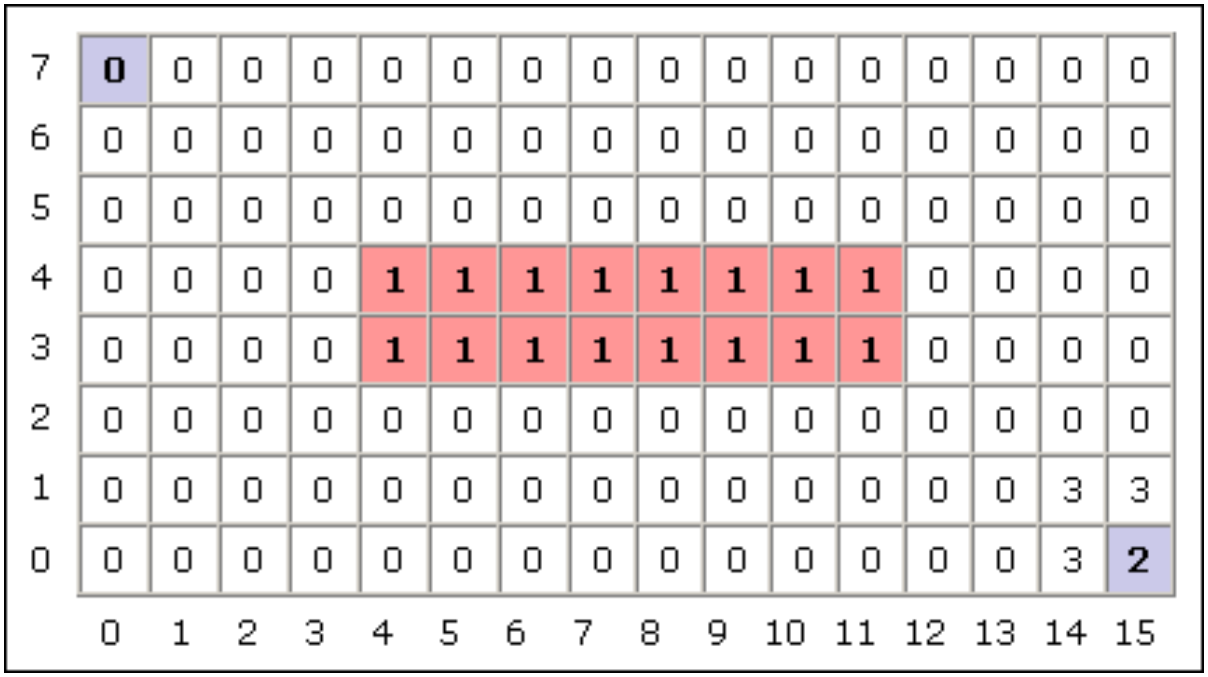


The Wavefront Planner: Setup

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**

The Wavefront in Action (Part 1)

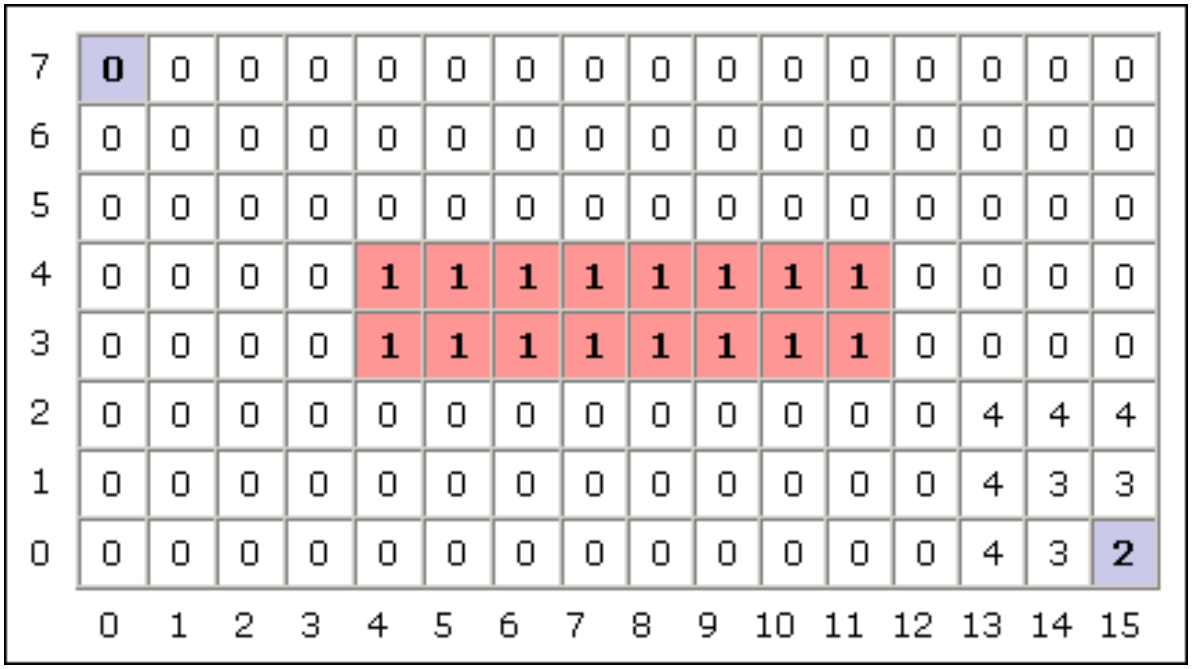
- Starting with the goal, set all adjacent cells with “0” to the current cell + 1
  - 4-Point Connectivity or 8-Point Connectivity?
  - Your Choice. We’ll use 8-Point Connectivity in our example



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	**2**

The Wavefront in Action (Part 2)

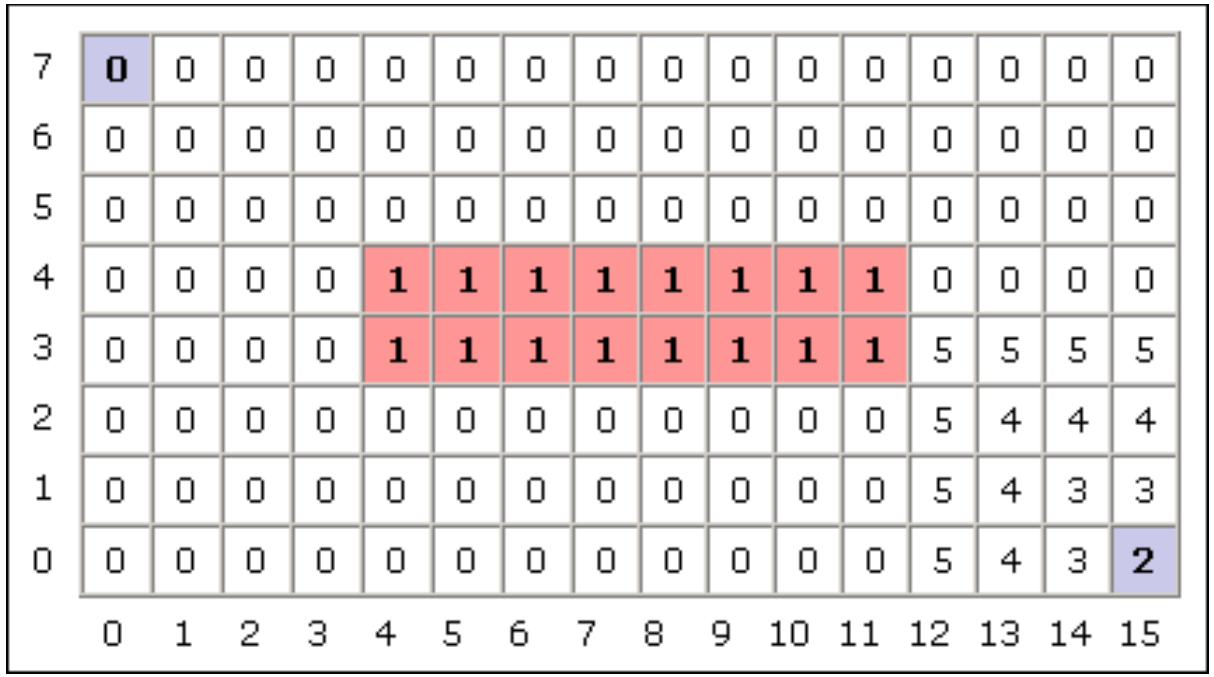
- Now repeat with the modified cells
  - This will be repeated until no 0's are adjacent to cells with values >= 2
    - 0's will only remain when regions are unreachable



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	0	0	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	3	**2**

The Wavefront in Action (Part 3)

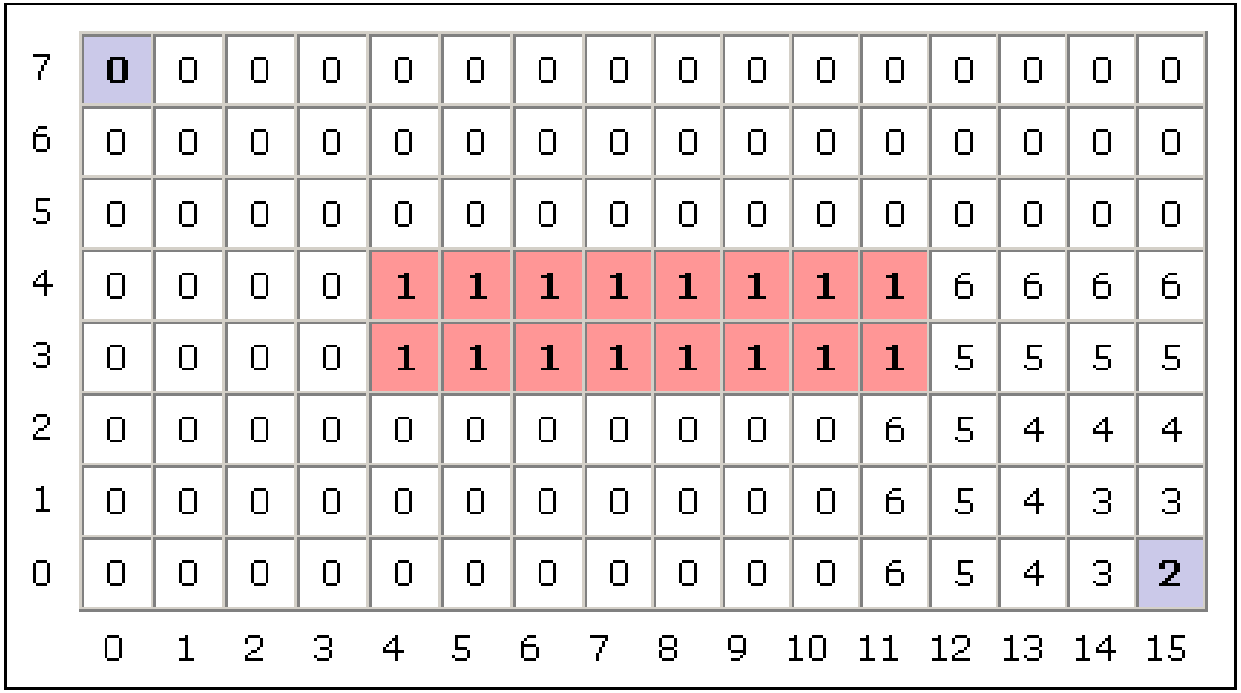
- Repeat again…



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	0	0	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	0	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	5	4	3	**2**

The Wavefront in Action (Part 4)

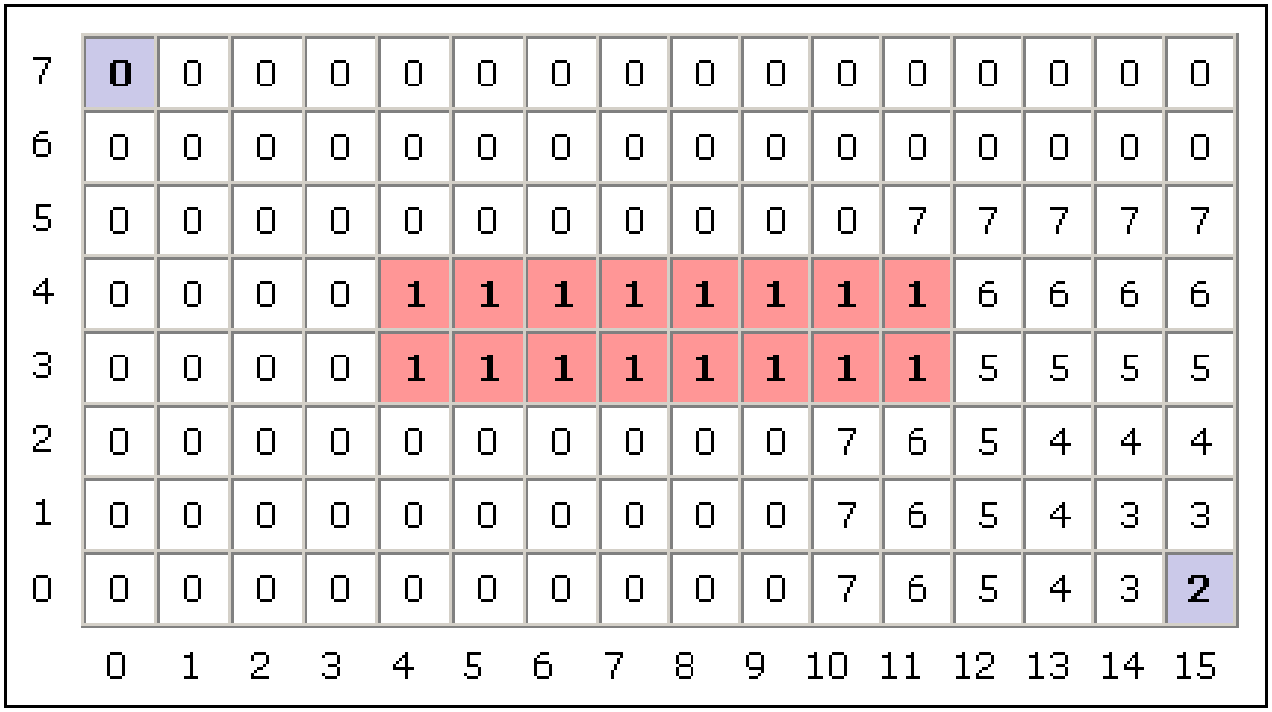
- And again…



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	0	6	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	6	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	6	5	4	3	**2**

The Wavefront in Action (Part 5)

- And again until…



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	7	7	7	7	7
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	7	6	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	7	6	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	7	6	5	4	3	**2**

The Wavefront in Action (Done)

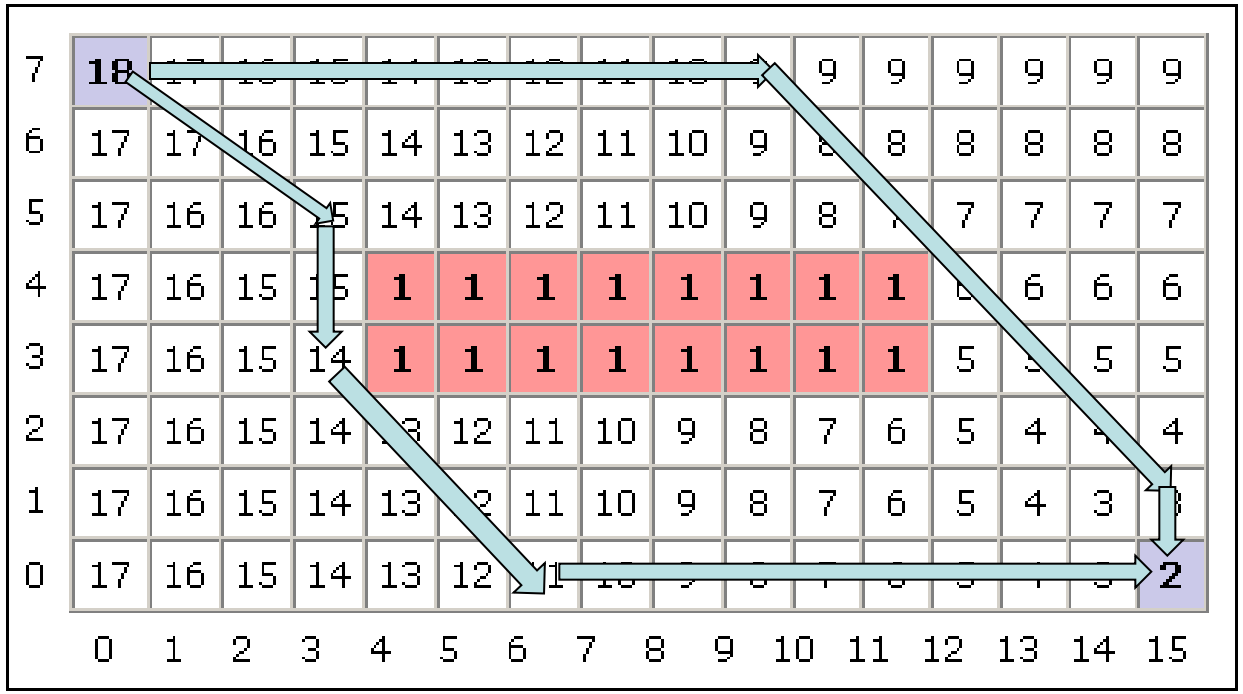
- You're done
  - Remember, 0's should only remain if unreachable regions exist

7	**18**	17	16	15	14	13	12	11	10	9	9	9	9	9	9	9
6	17	17	16	15	14	13	12	11	10	9	8	8	8	8	8	8
5	17	16	16	15	14	13	12	11	10	9	8	7	7	7	7	7
4	17	16	15	15	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	17	16	15	14	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	17	16	15	14	13	12	11	10	9	8	7	6	5	4	4	4
1	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	3
0	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront, Now What?

- To find the shortest path, according to your metric, simply always move toward a cell with a lower number
  - The numbers generated by the Wavefront planner are roughly proportional to their distance from the goal

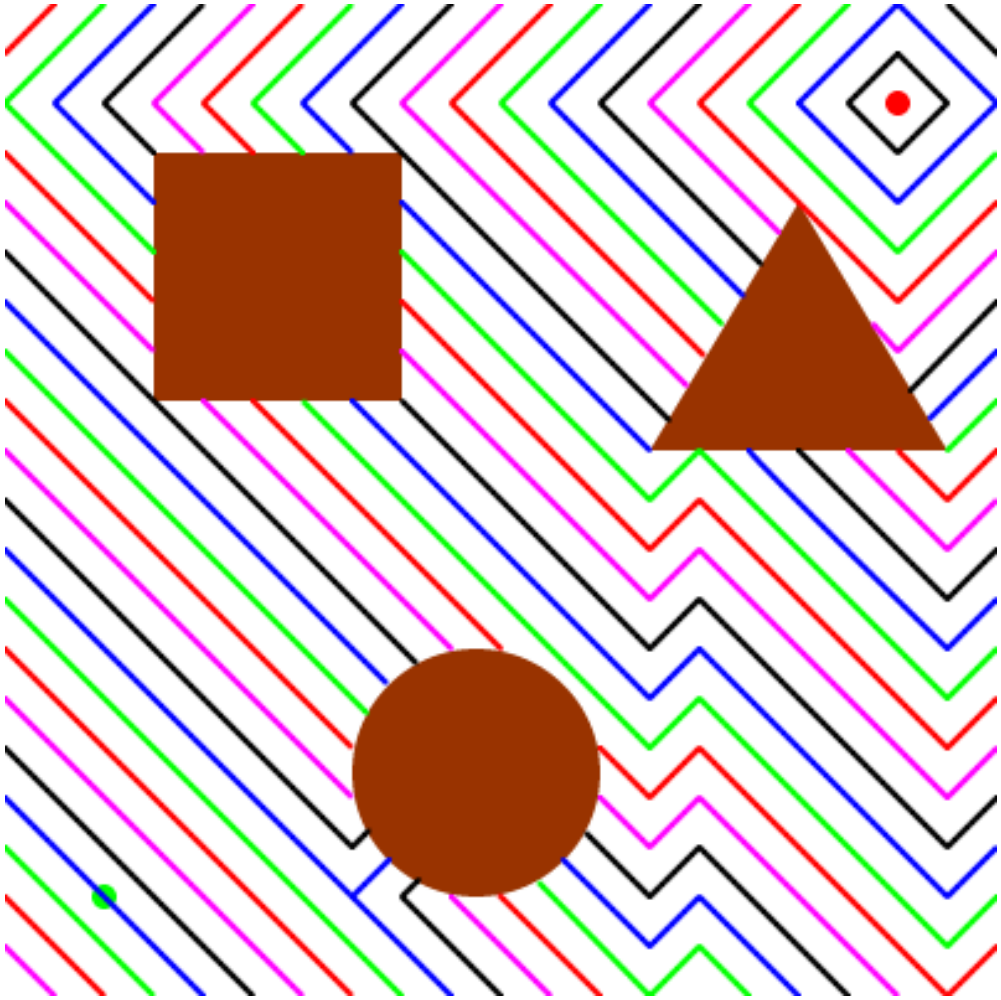
Two possible shortest paths shown



Wavefront (Overview)

- Divide the space into a grid.
- Number the squares starting at the start in either 4 or 8 point connectivity starting at the goal, increasing till you reach the start.
- Your path is defined by any uninterrupted sequence of decreasing numbers that lead to the goal.

Wavefront Slide



Wavefront References

- http://www.societyofrobots.com/programming_wavefront.shtml

Final Notes

- Introduction to path planning challenge
  - Motion capture, grid based, obstacles, map vs. no map