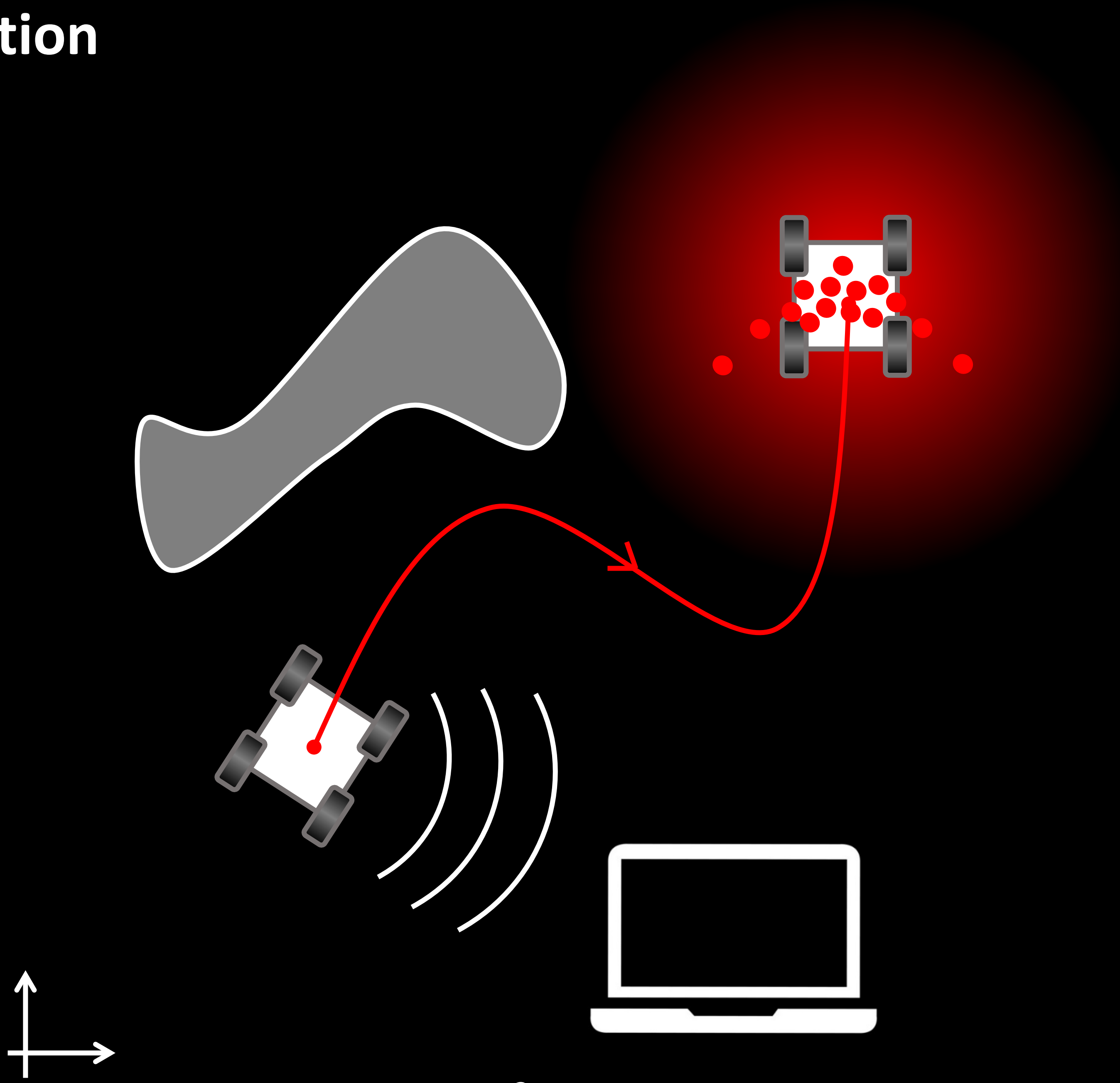
ECE 4160/5160
MAE 4910/5910

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Map Representations, Graphs and Graph Search

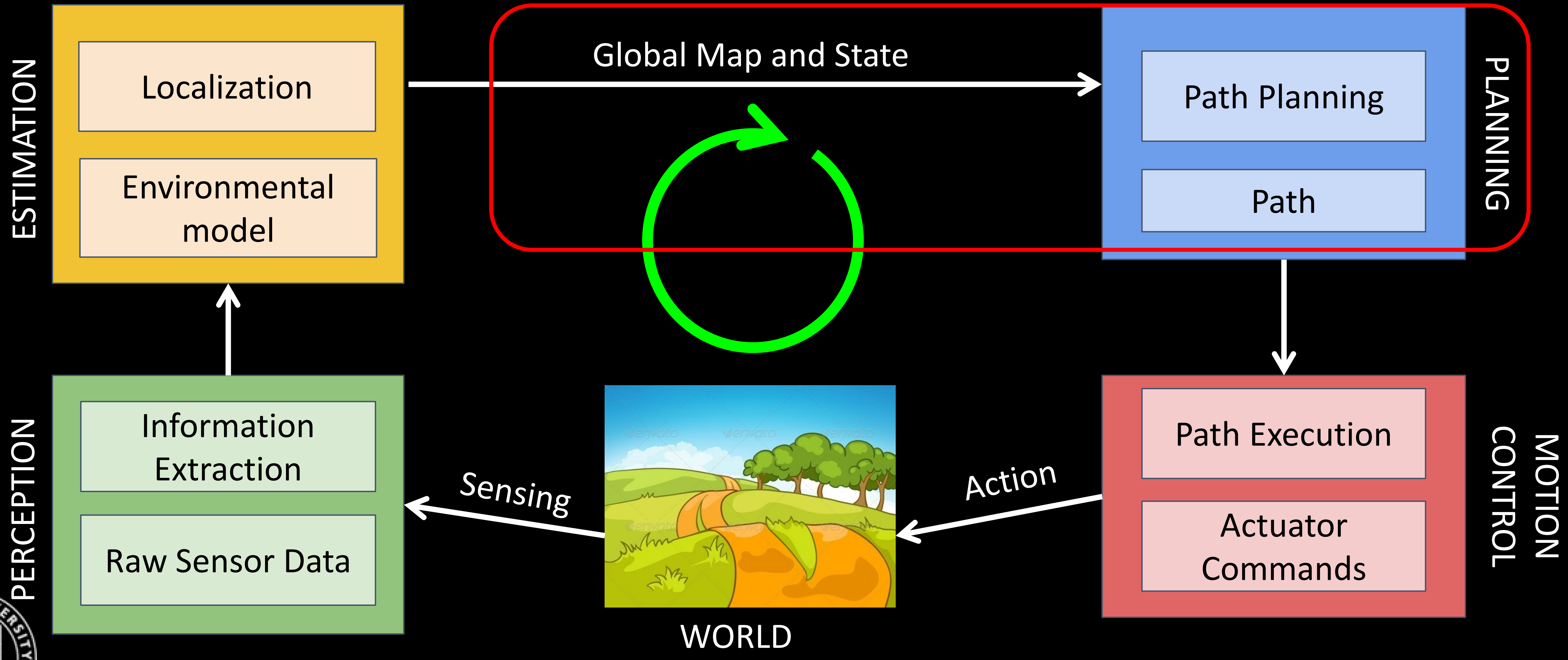
Outline of the next module on Navigation

- Local planners
- Global localization and planning
    - Map representations
        - Continuous
        - Discrete
        - Topological
  - Maps as graphs
    - Graph Search Algorithms
        - Breadth First Search
        - Depth First Search
        - Dijkstras
        - A*
    - Localization, Sensor and motion models, SLAM



Navigation

- Navigation breaks down to: Localization, Map Building, Path Planning



Map Representation

(a) Building plan

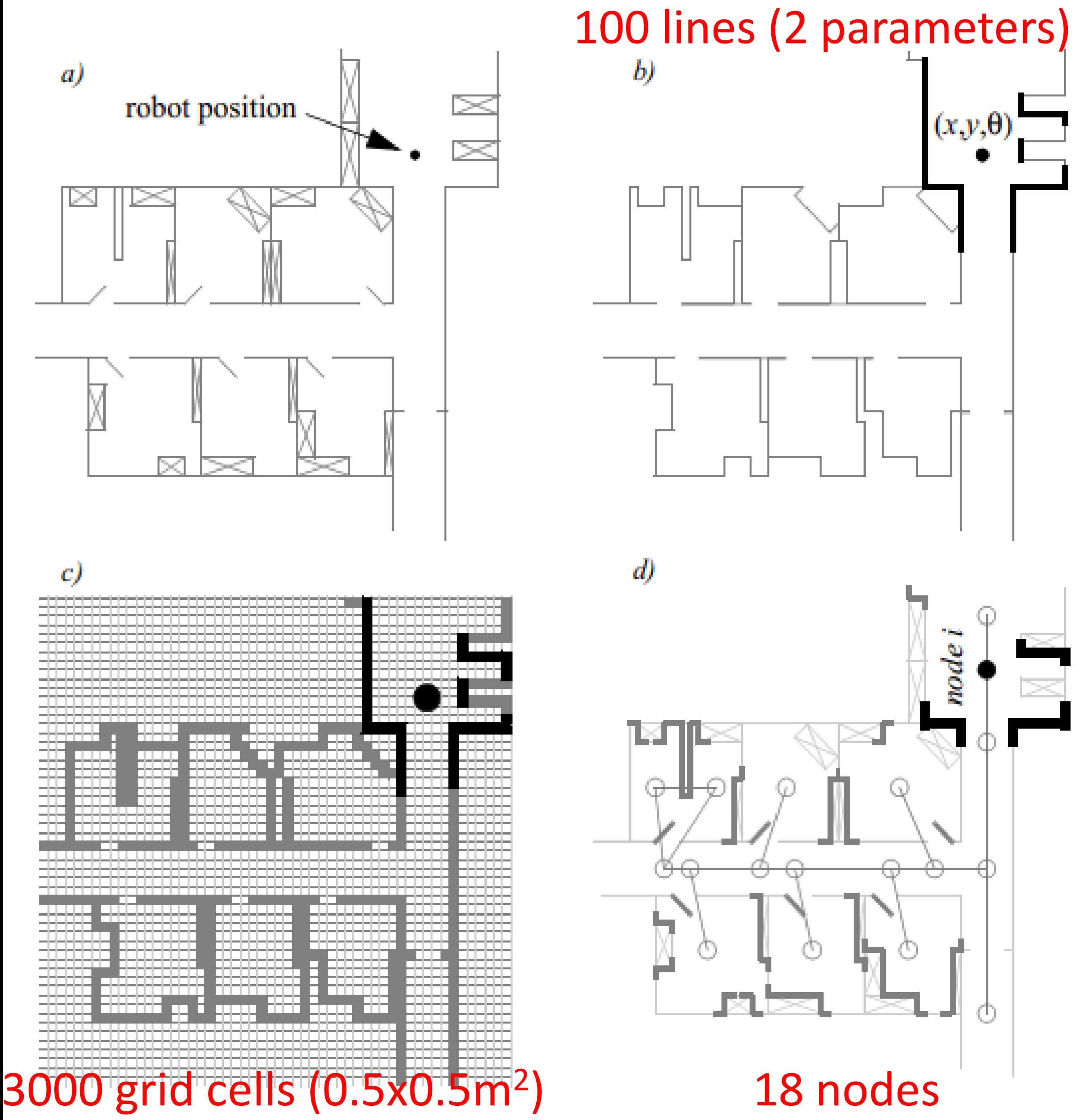
(b) line-based map

(c) occupancy grid-based map

(d) topological map

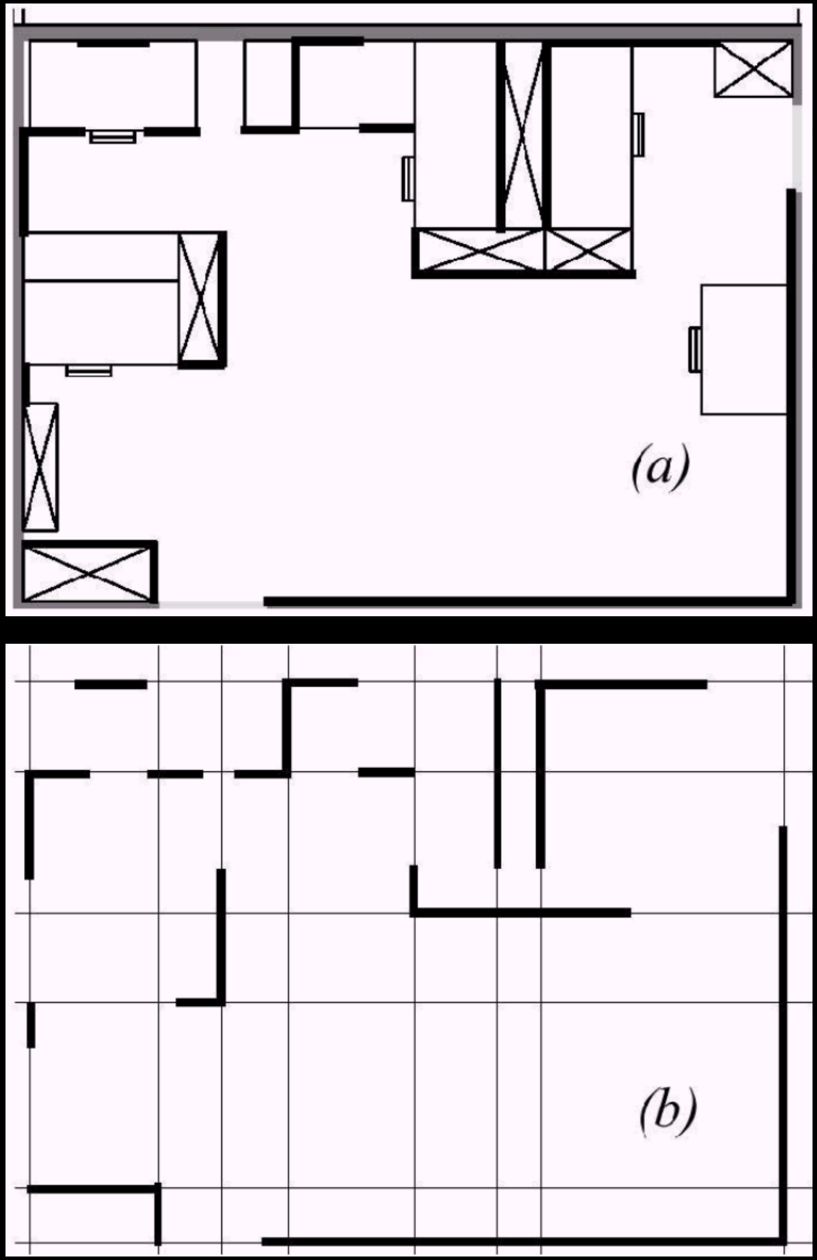
Important properties

- Memory allocation
- Computation



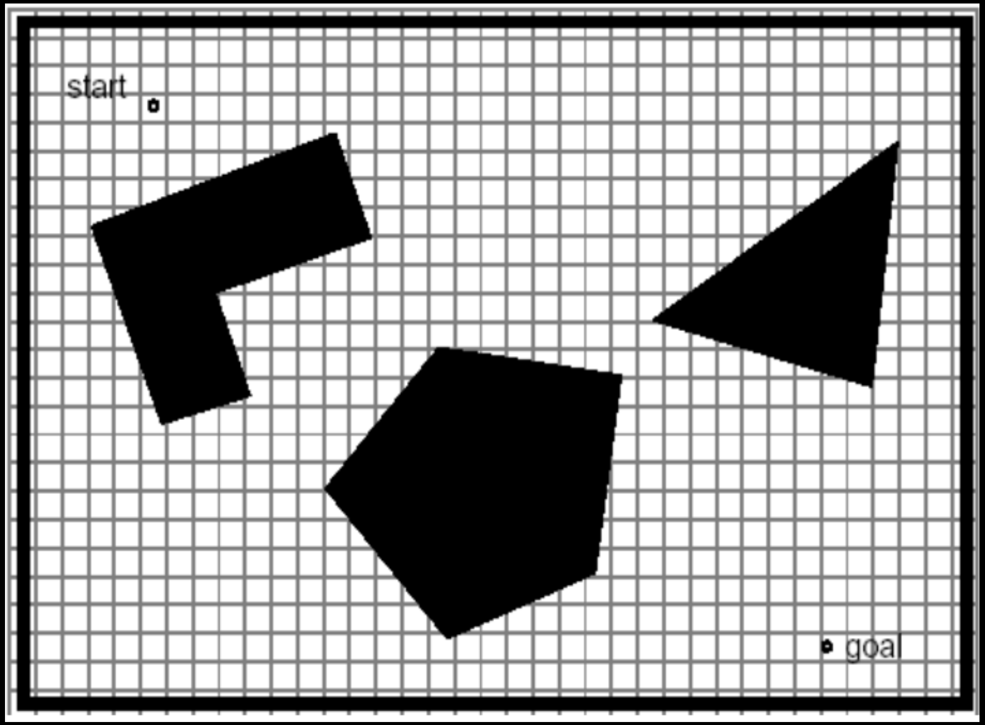
Continuous Representations

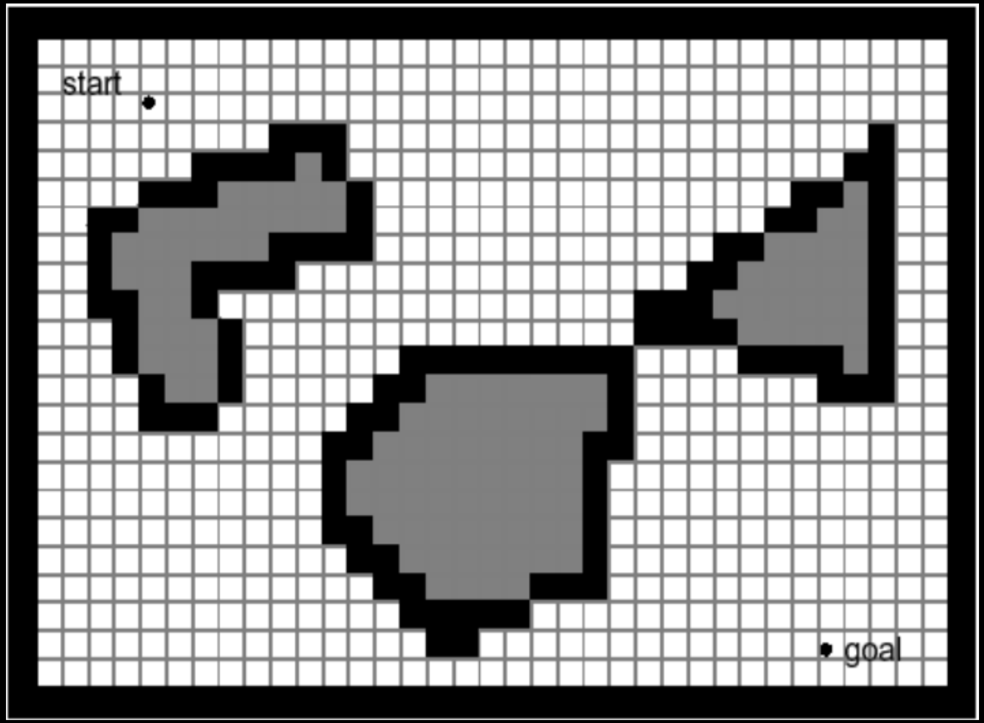
- Exact decomposition of the environment
- Used mainly in 2D representations
- Closed-world assumption
- Storage proportional to object density
- Example: Continuous line representations
  - Using range finders, we can extract lines/line segments in the environment



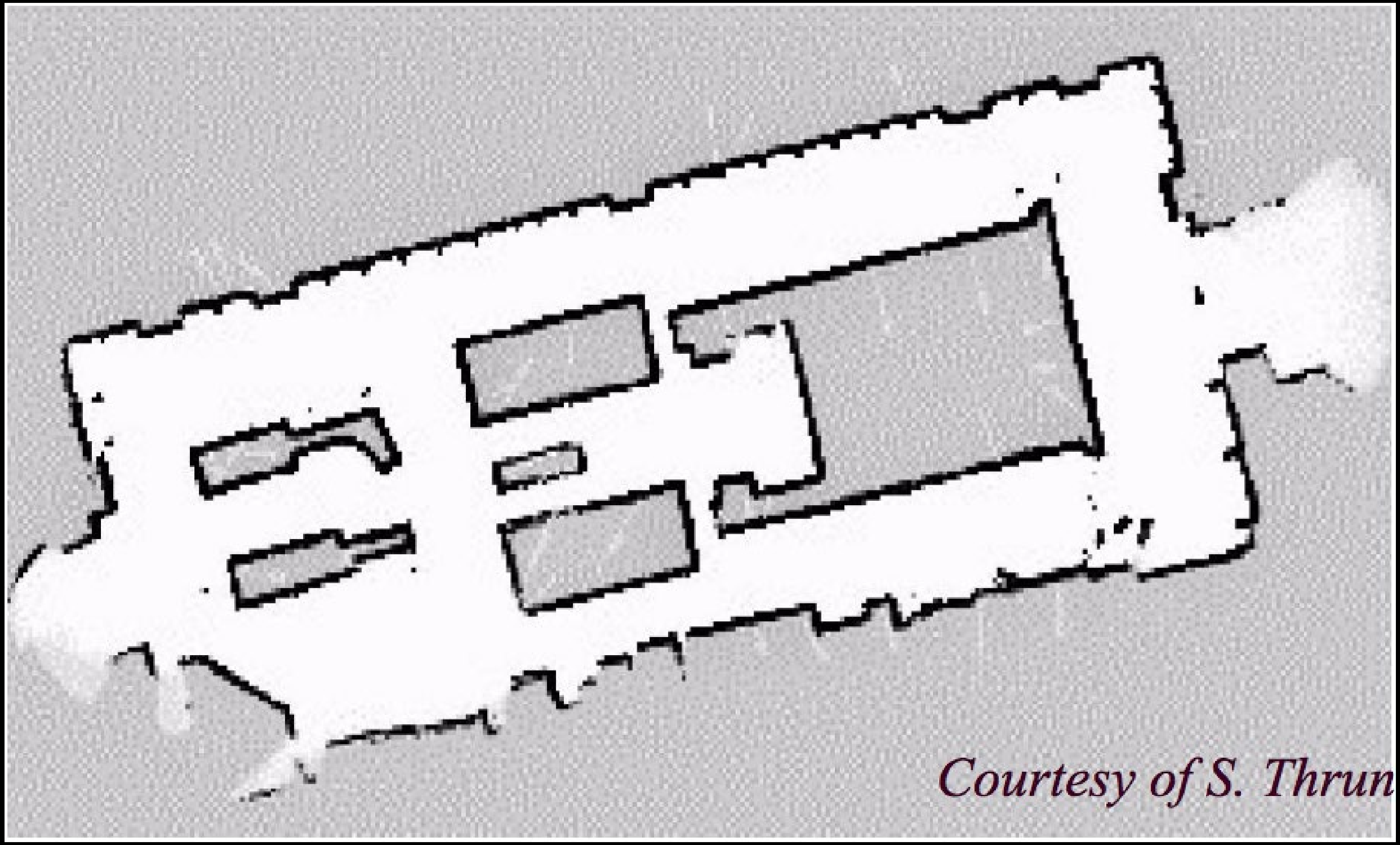
Fixed Decomposition

- Tessellate the world at a fixed resolution
- Approximate features given the resolution
- Most commonly used: Occupancy grid



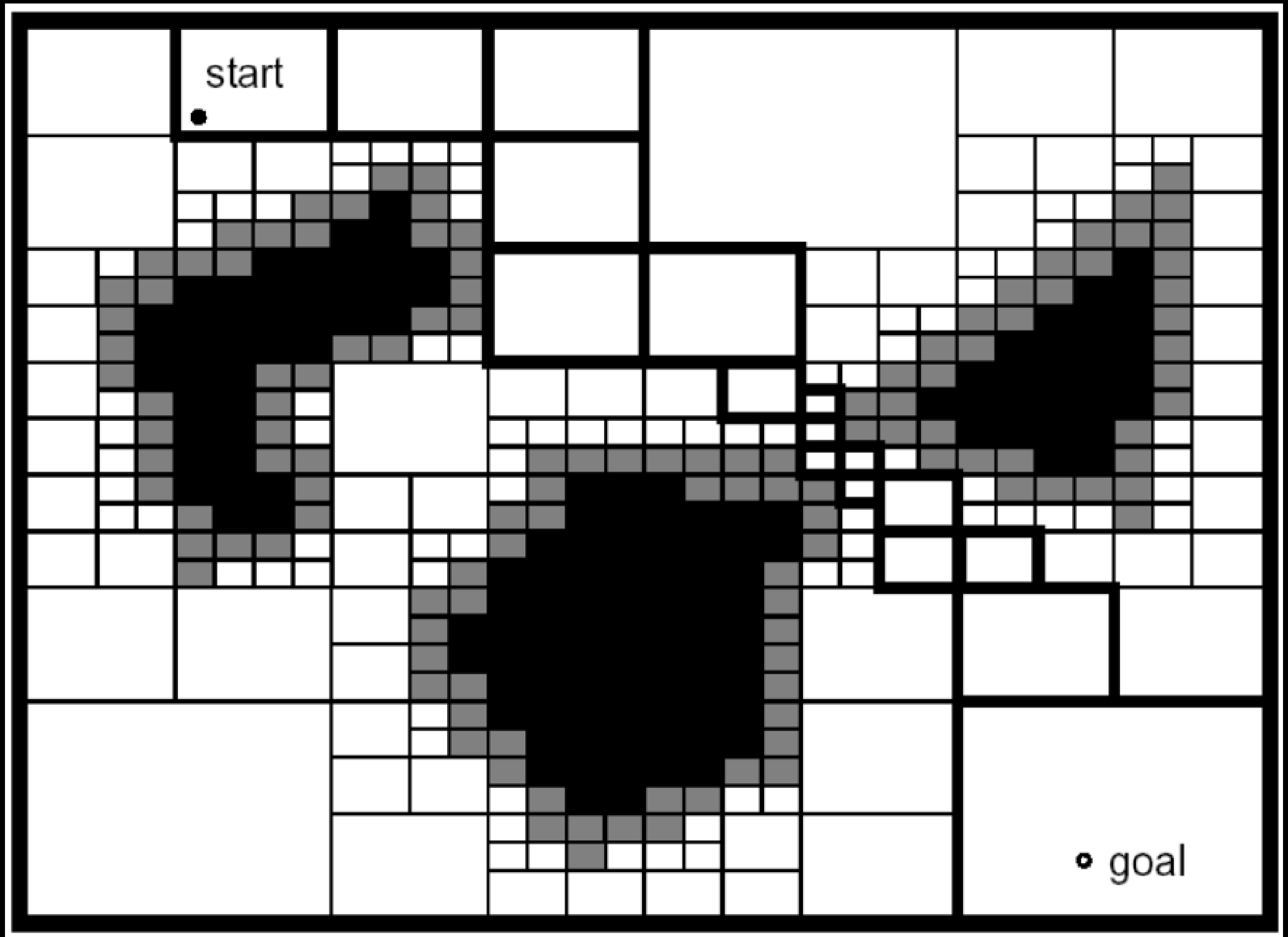


Fixed Decomposition



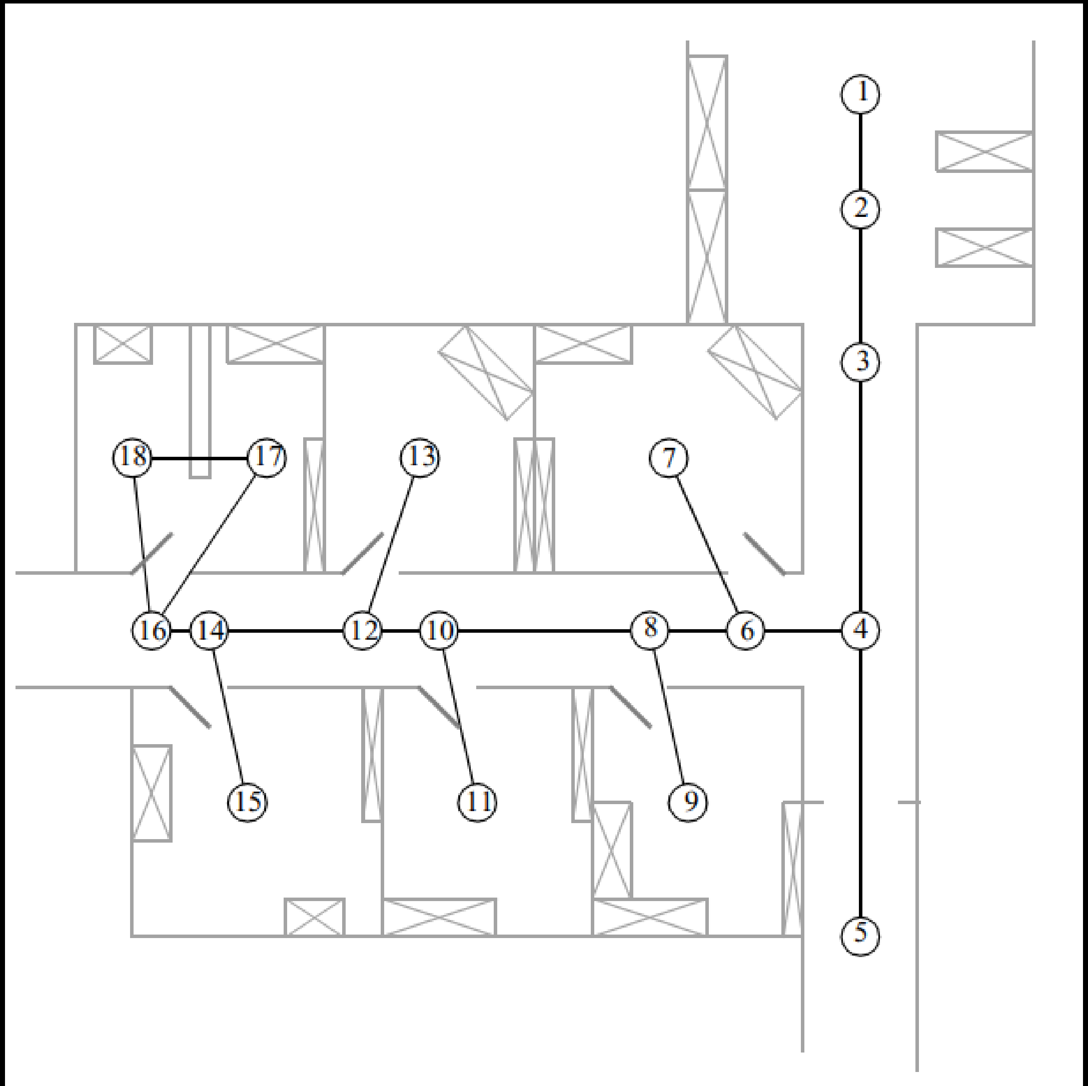
Adaptive Cell Decomposition

- Adapt cell size to features



Topological Decomposition

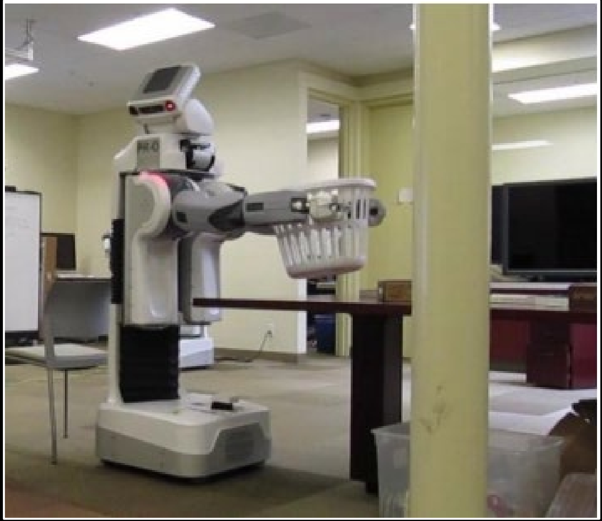
- A topological representation is a graph that specifies nodes and edges
  - Nodes denote areas in the environment
  - Edges describe environment connectivity
- Robots can…
  - ...detect their current position in terms of the nodes of the topological graph
  - …travel between nodes using robot motion



Topological Decomposition

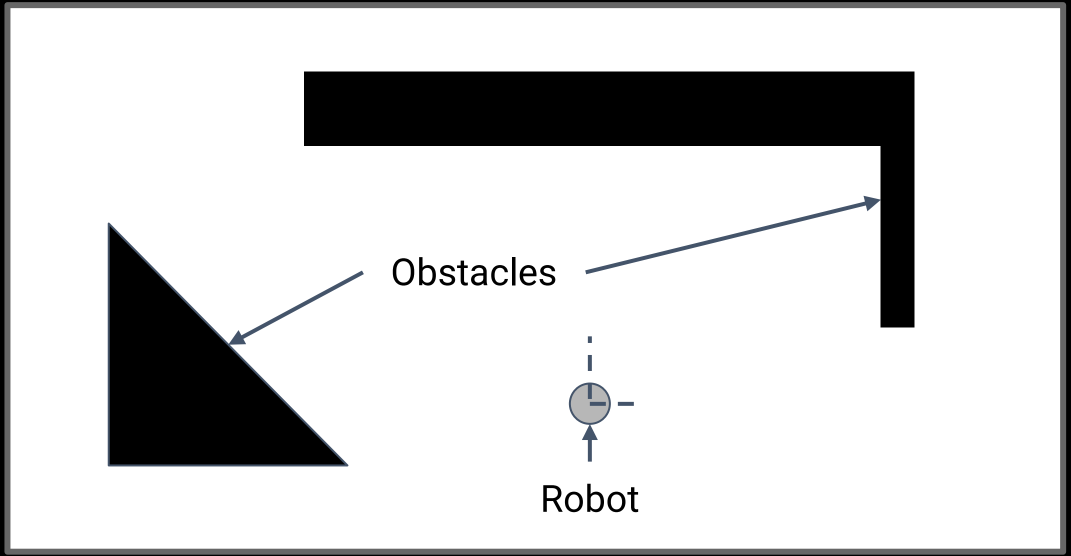
- A topological representation is a graph that specifies nodes and edges
  - Nodes denote areas in the environment
  - Edges describe environment connectivity
- Robots can…
  - ...detect their current position in terms of the nodes of the topological graph
  - …travel between nodes using robot motion

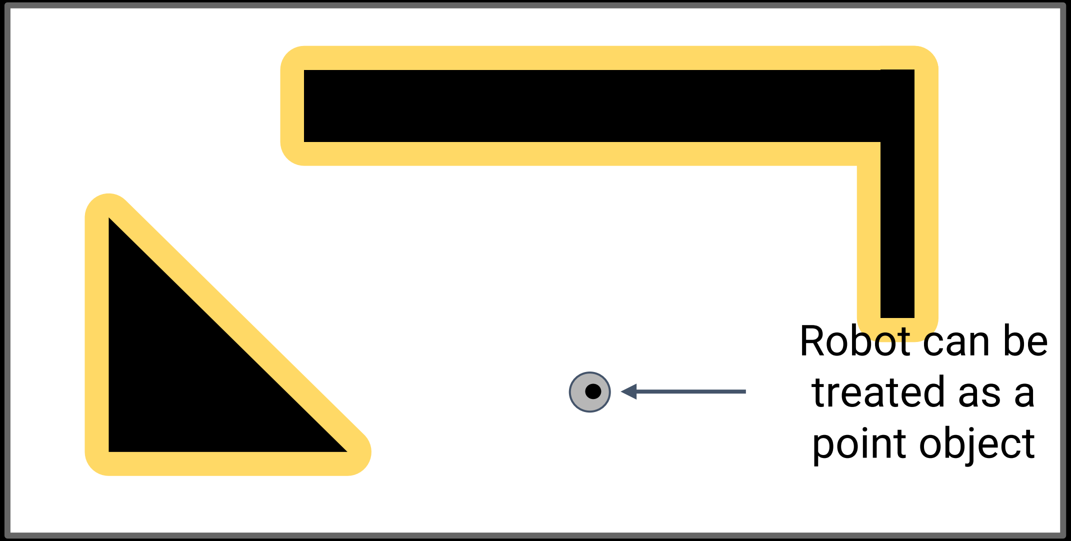
- Typical for 3D maps



How to represent the robot pose?

- Physical robots take up space
- Expand obstacles
- Represent maps in configuration space instead of Euclidean space





Map Representation Considerations

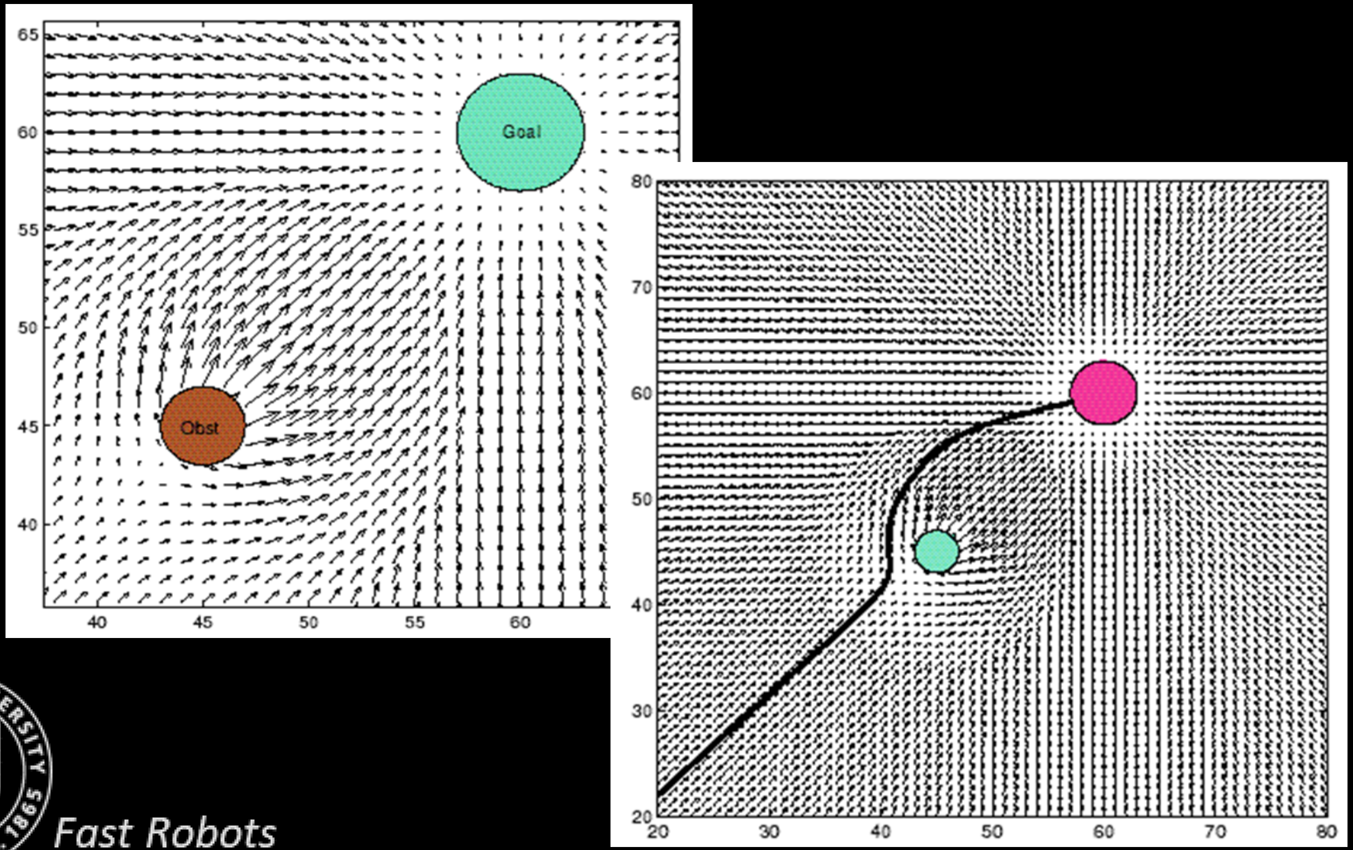
- The precision of the map must appropriately match the precision with which the robot needs to achieve its goals
- The precision of the map and the type of features represented must match the precision and data types returned by the robot's sensors
- The complexity of the map representation has direct impact on the computational complexity of reasoning about mapping, localization, and navigation

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Constructing Graphs

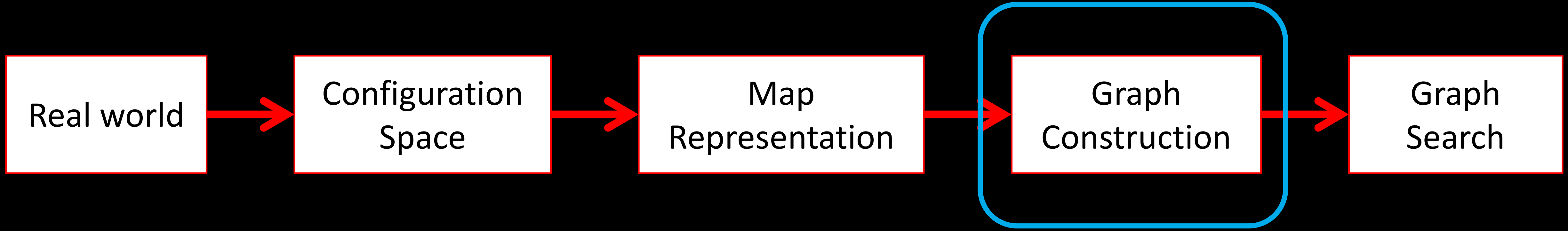
Modelling path planning as a graph search problem

Real world → Configuration Space → Map Representation → Graph Construction → Graph Search



Common alternatives
- Optimal control
- Potential fields

Modelling path planning as a graph search problem

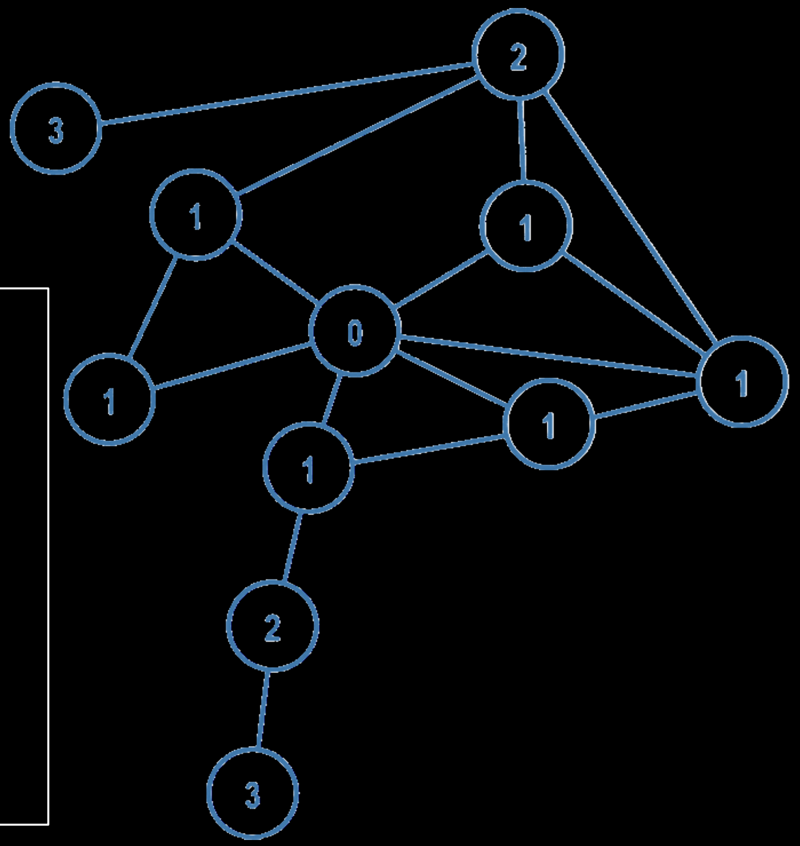


- Topological Graphs
- Cell decomposition
- Visibility Graphs
- RRT
- PRM

Graph Construction

- Transform continuous/discrete/topological maps to a discrete graph
- Why?
  - Model the path planning problem as a search problem
  - Graph theory has lots of tools
  - Real-time capable algorithms
  - Can accommodate for evolving maps

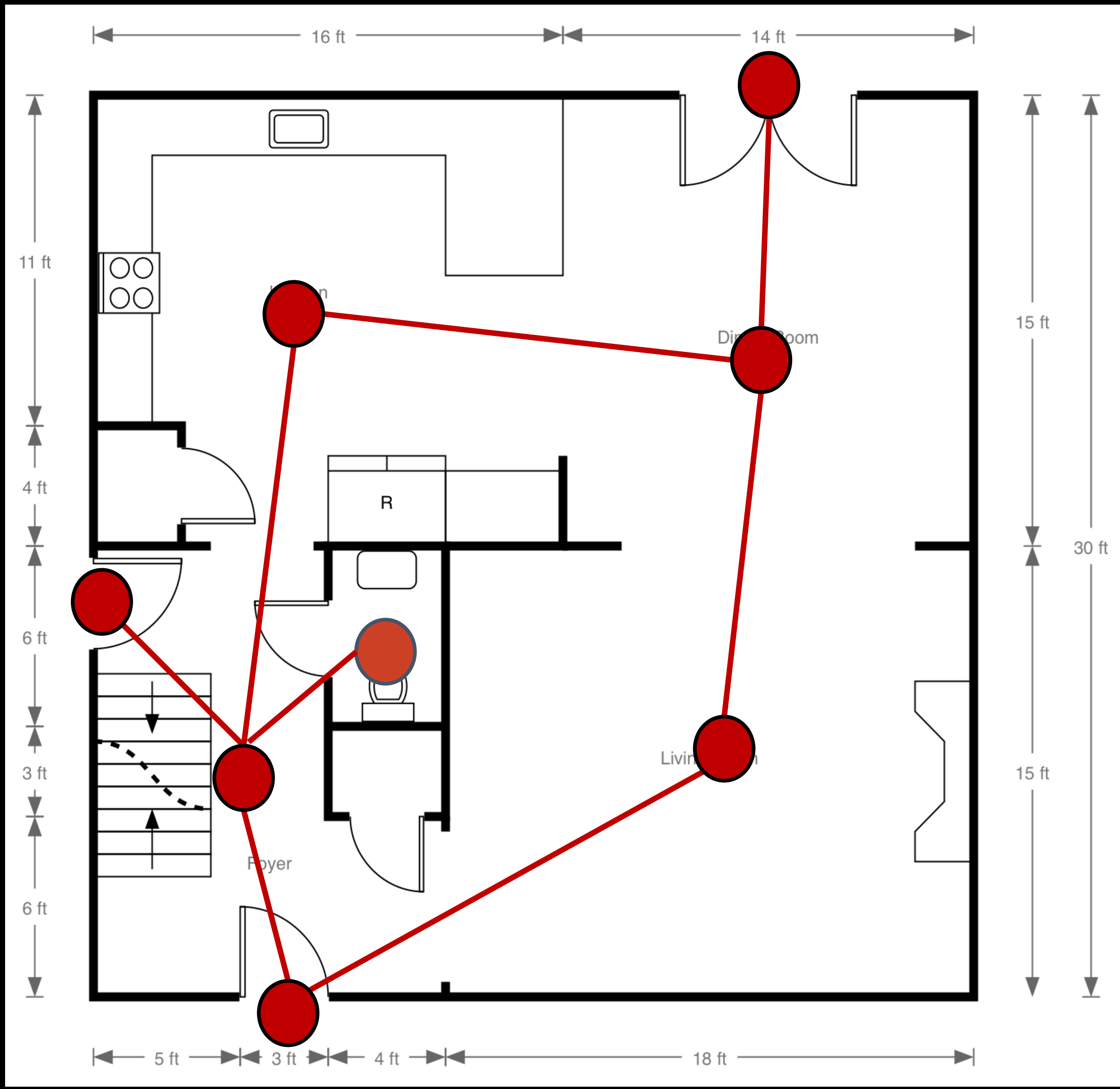
1. Divide space into simple, connected regions, or “cells”
2. Determine adjacency of open cells
3. Construct a connectivity graph
4. Find cells with initial and goal configuration
5. Search for a path in the connectivity graph to join them
6. From the sequence of cells, compute a path within each cell
   - e.g. passing through the midpoints of cell boundaries or by sequence of wall following movements



Geometry-Based Planners

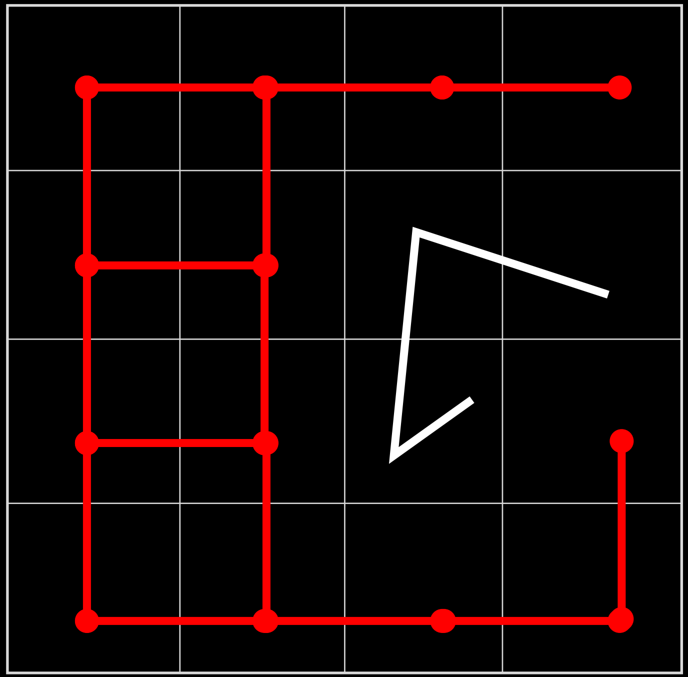
Topological Maps

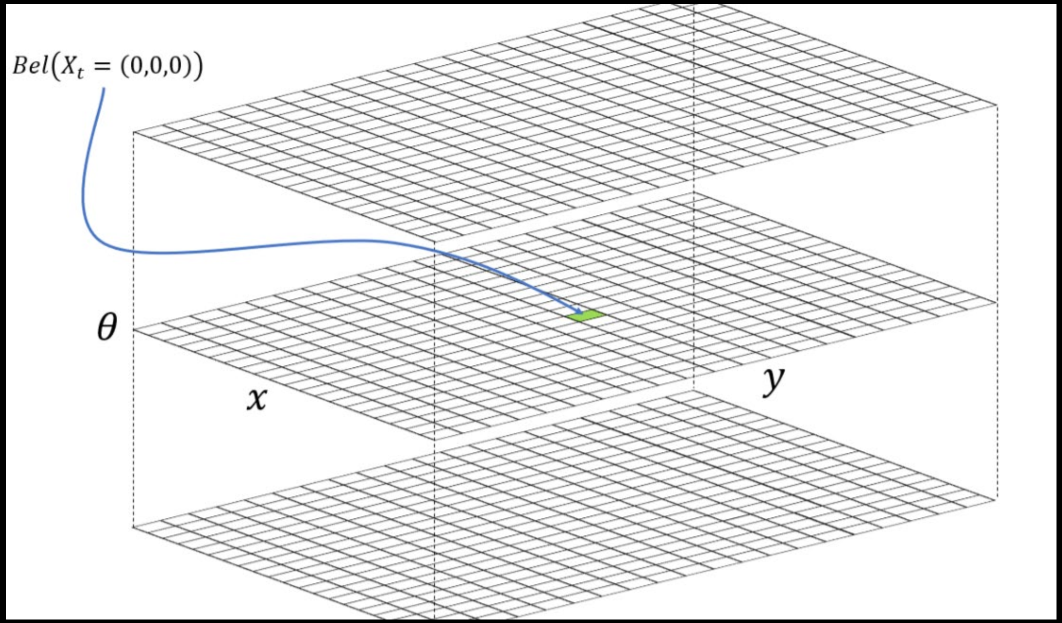
- Good abstract representation
- Tradeoff in # of nodes
  - Complexity vs. accuracy
  - Efficient in large, sparse environments
  - Loss in geometric precision
- Edges can carry weights
- Con: Limited information



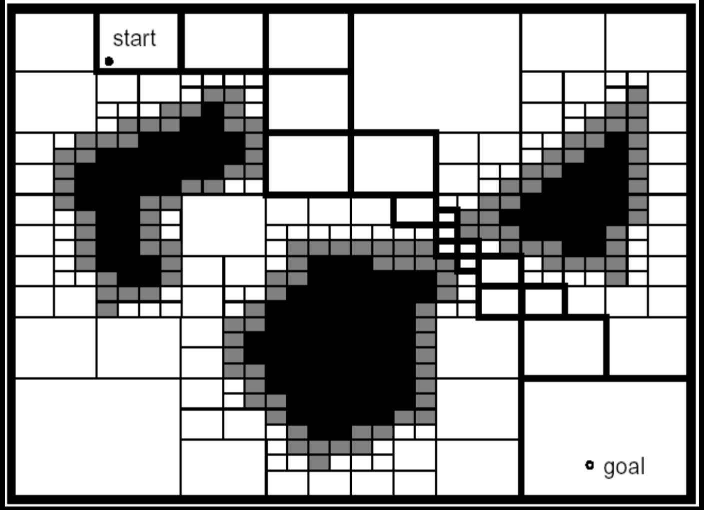
Fixed Cell Decomposition

(Lab 9-12)

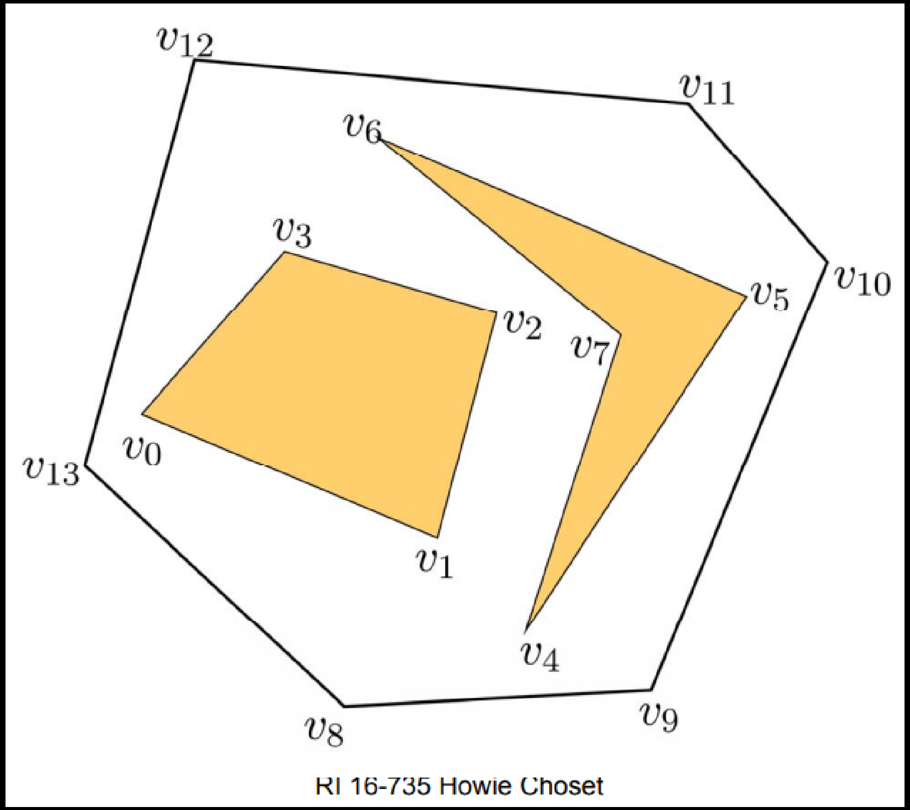


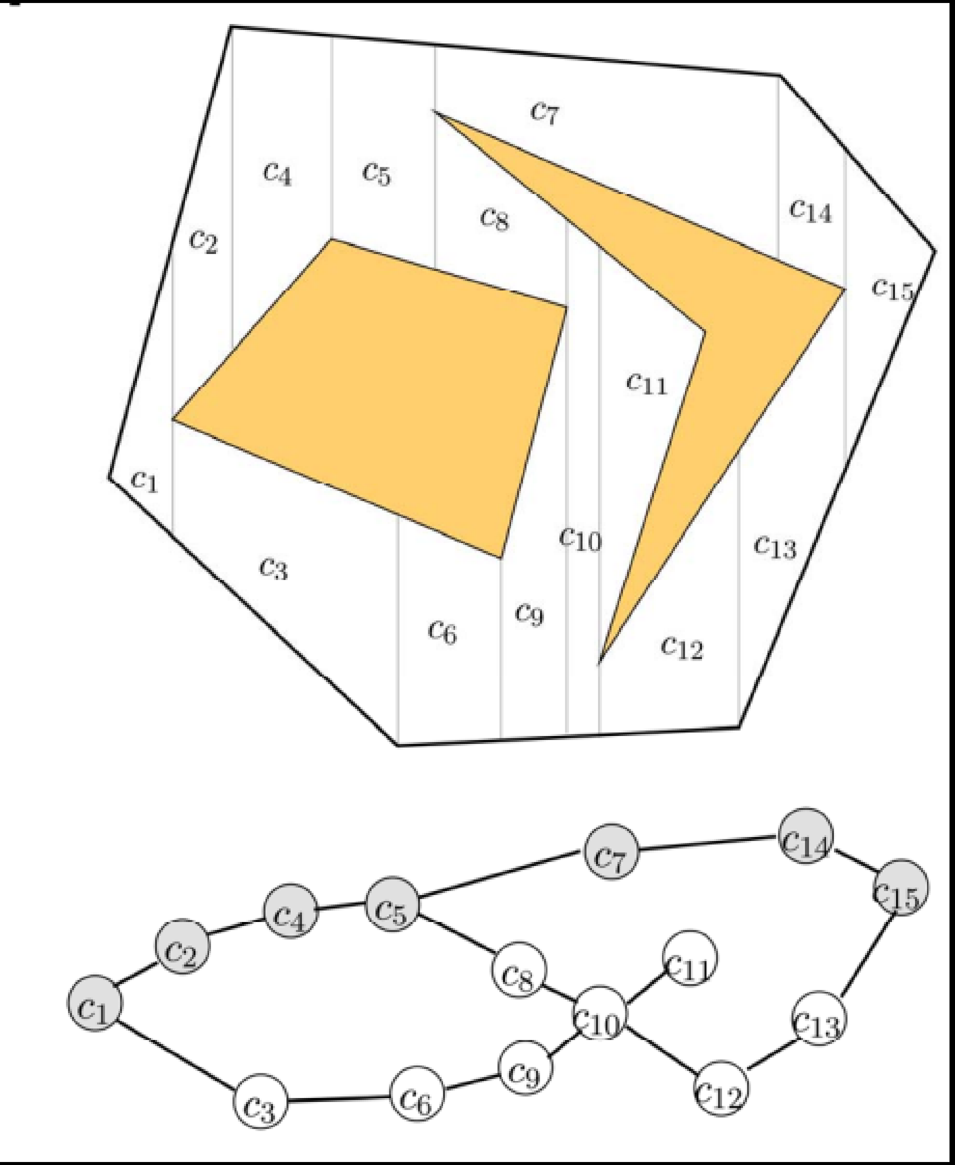


Adaptive Cell Decomposition



Trapezoidal Cell Decomposition

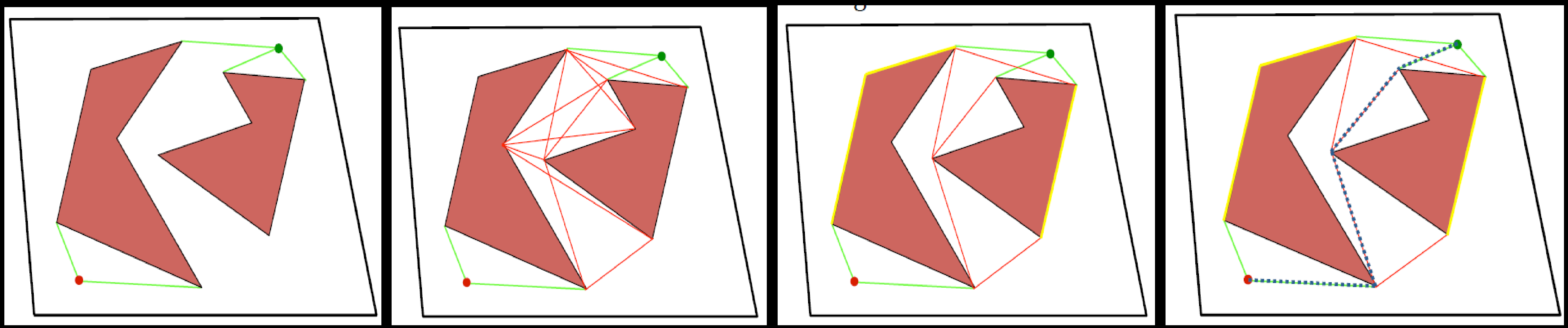




Visibility Graphs

Ioannis Rekleitis,
South Carolina

- Connect initial and goal locations with all visible vertices
- Connect each obstacle vertex to every visible obstacle vertex
- Remove edges that intersect the interior of an obstacle
- Plan on the resulting graph

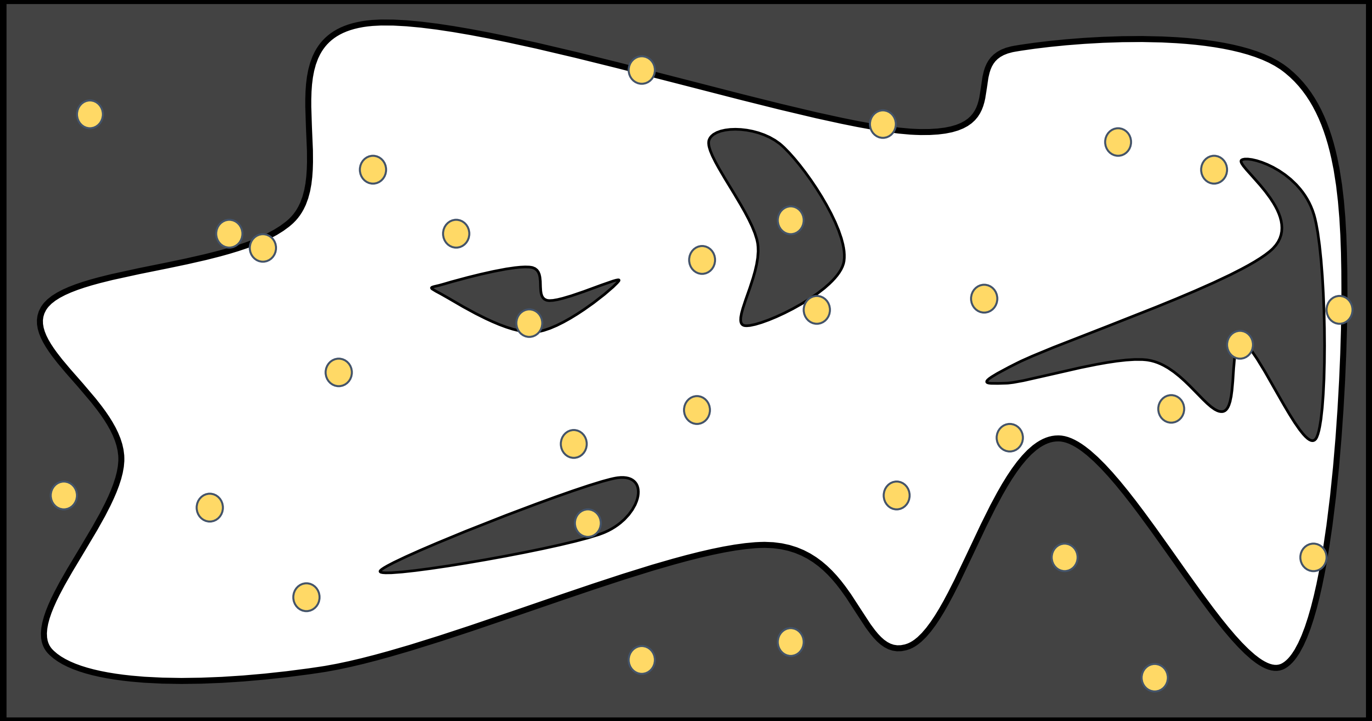


Sampling-Based Planners

- Rather than computing the C-Space explicitly, we sample it
- Often efficient in high dimensional spaces
- Compute if a robot configuration has collisions
    - Just requires forward kinematics
    - (Local path plans between configurations)
- Examples
    - Probabilistic Roadmaps (PRM)
    - Rapidly Exploring Random Trees (RRT)

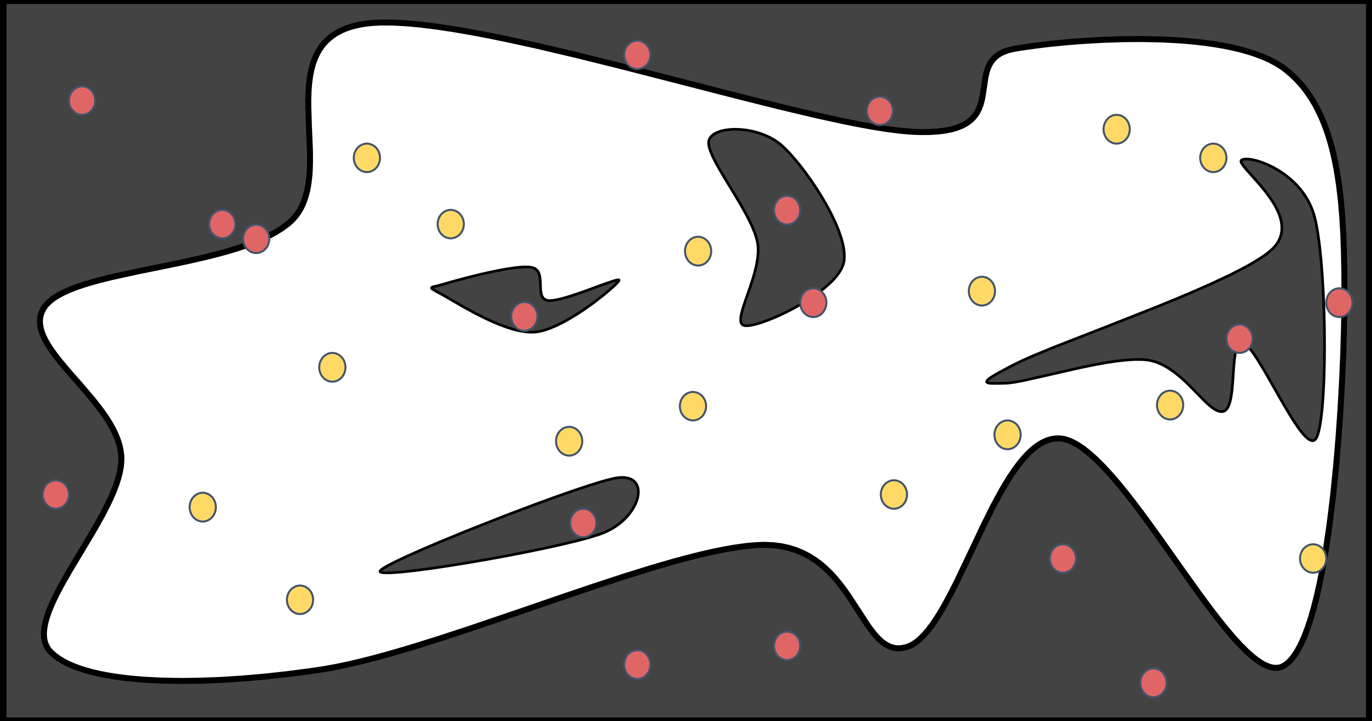
Probabilistic Roadmaps

- Configurations are sampled by picking coordinates at random



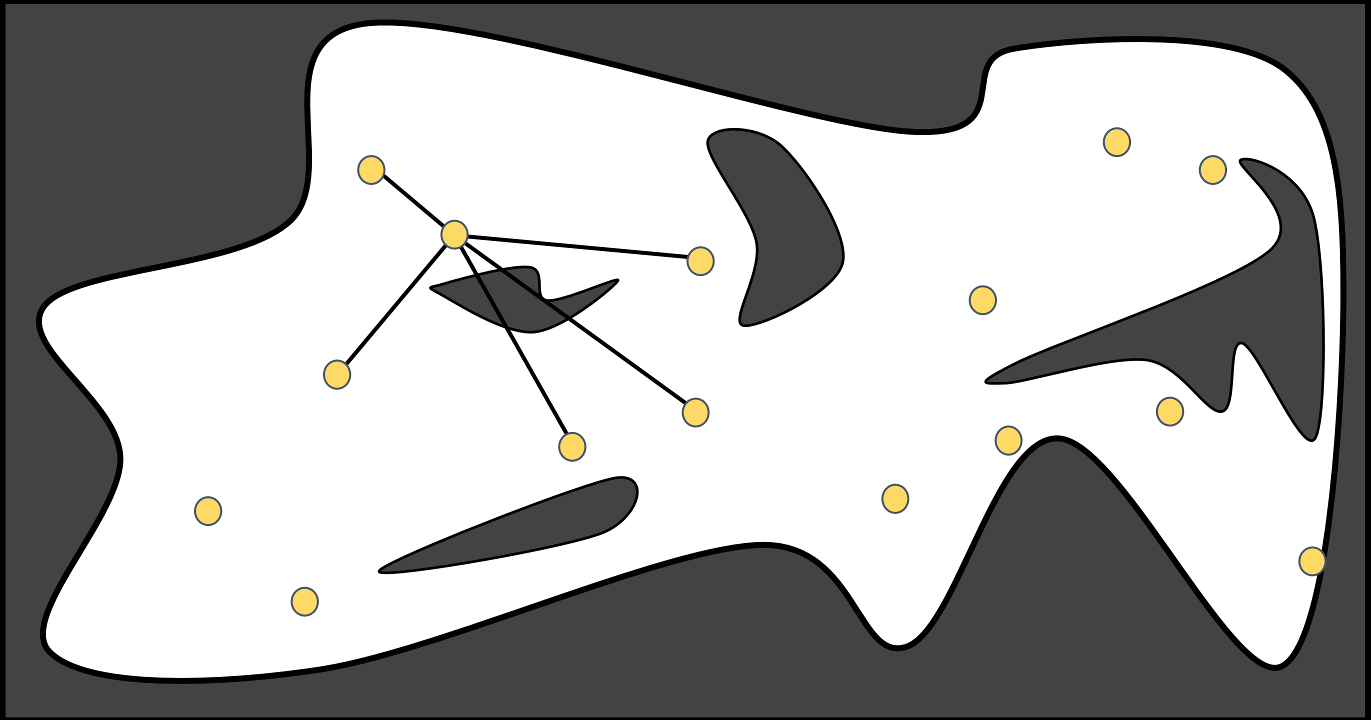
Probabilistic Roadmaps

- Configurations are sampled by picking coordinates at random
- Sampled configurations are tested for collision



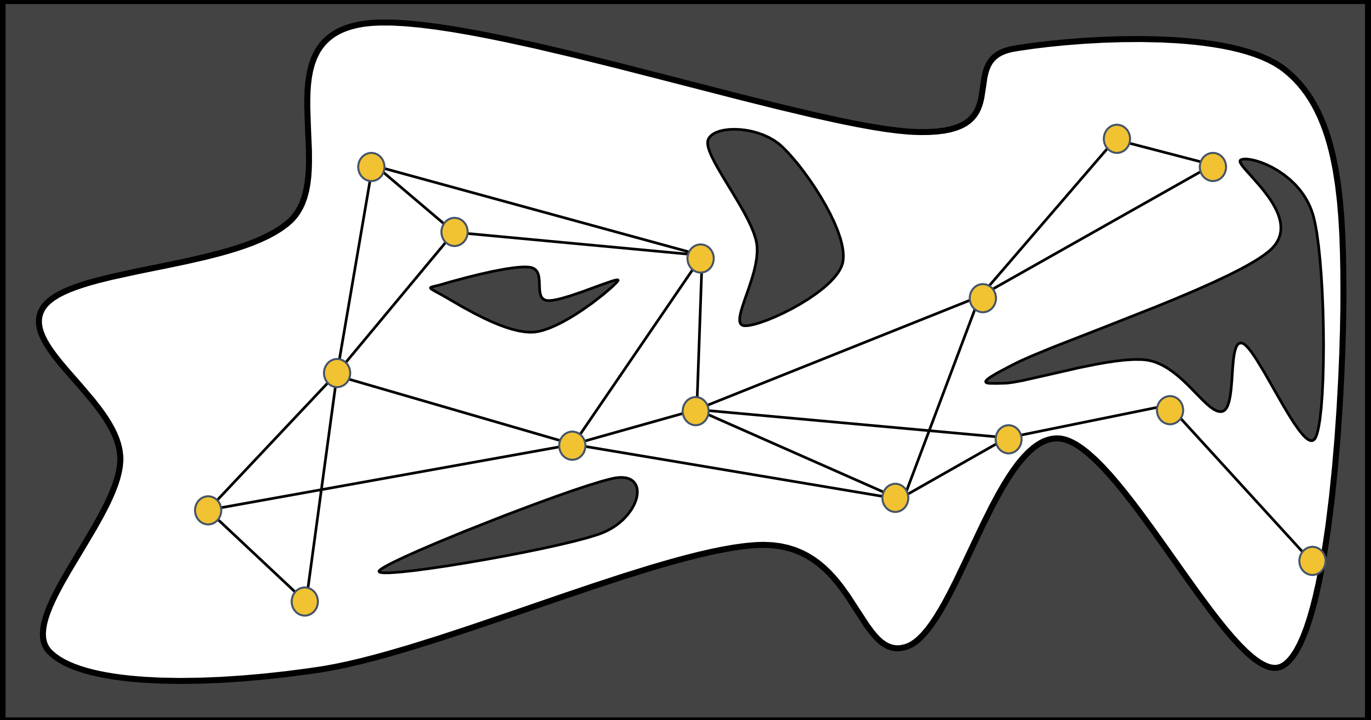
Probabilistic Roadmaps

- Configurations are sampled by picking coordinates at random
- Sampled configurations are tested for collision
- Each configuration is linked by straight paths to its nearest neighbors



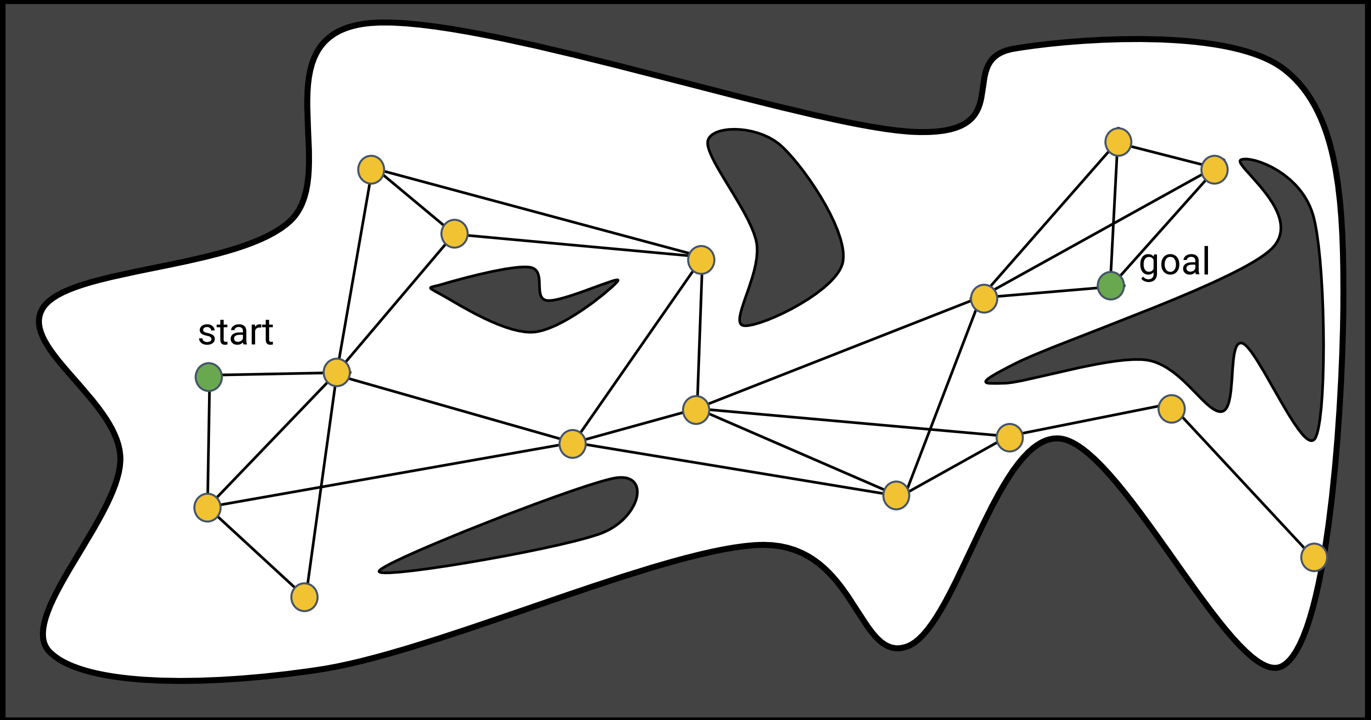
Probabilistic Roadmaps

- Configurations are sampled by picking coordinates at random
- Sampled configurations are tested for collision
- Each configuration is linked by straight paths to its nearest neighbors
- The collision-free links are retained as local paths to form the PRM



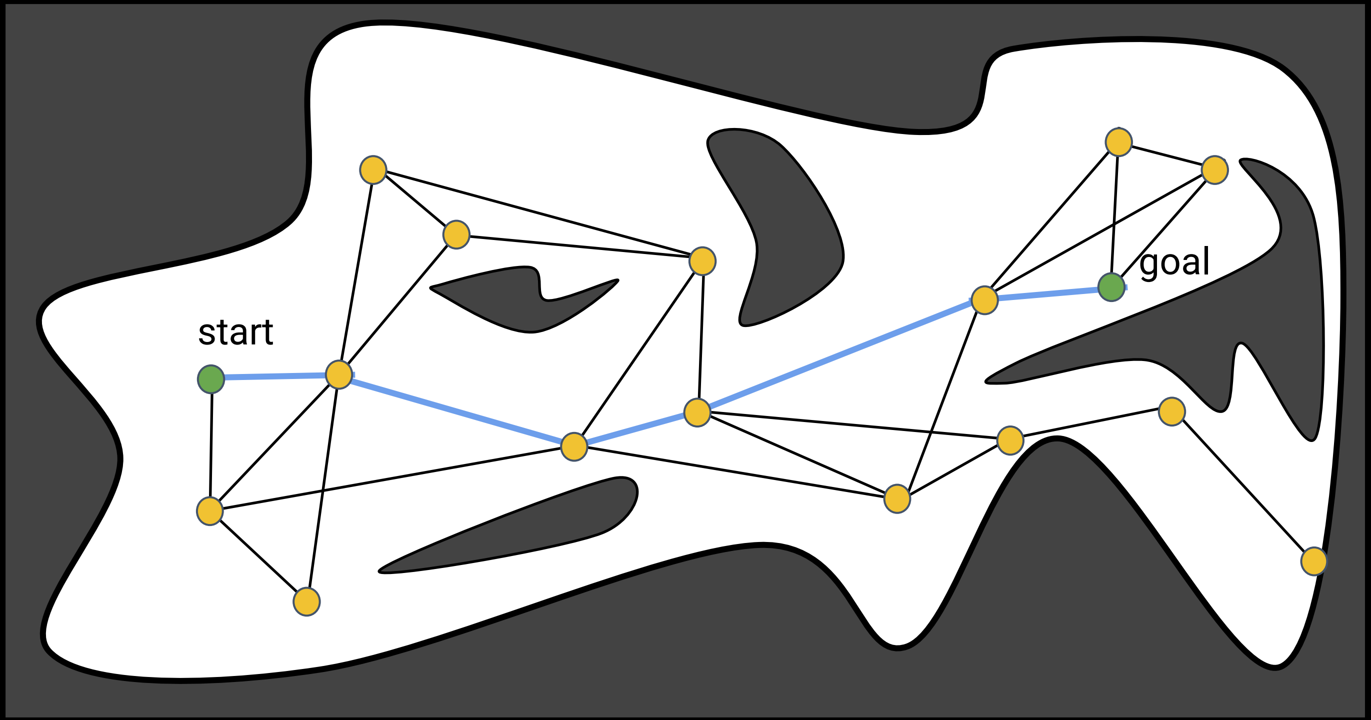
Probabilistic Roadmaps

- Configurations are sampled by picking coordinates at random
- Sampled configurations are tested for collision
- Each configuration is linked by straight paths to its nearest neighbors
- The collision-free links are retained as local paths to form the PRM
- The start and goal configurations are included as milestones



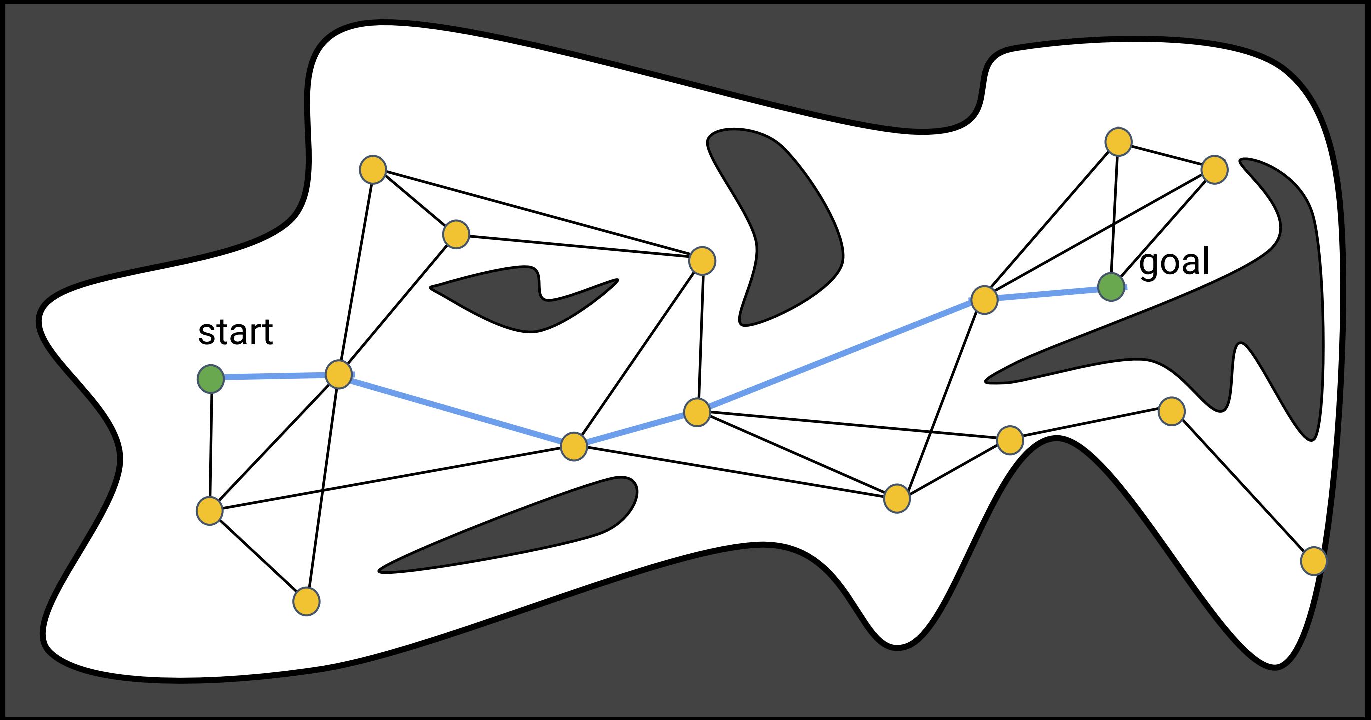
Probabilistic Roadmaps

- Configurations are sampled by picking coordinates at random
- Sampled configurations are tested for collision
- Each configuration is linked by straight paths to its nearest neighbors
- The collision-free links are retained as local paths to form the PRM
- The start and goal configurations are included as milestones
- The PRM is searched for a path from start to goal

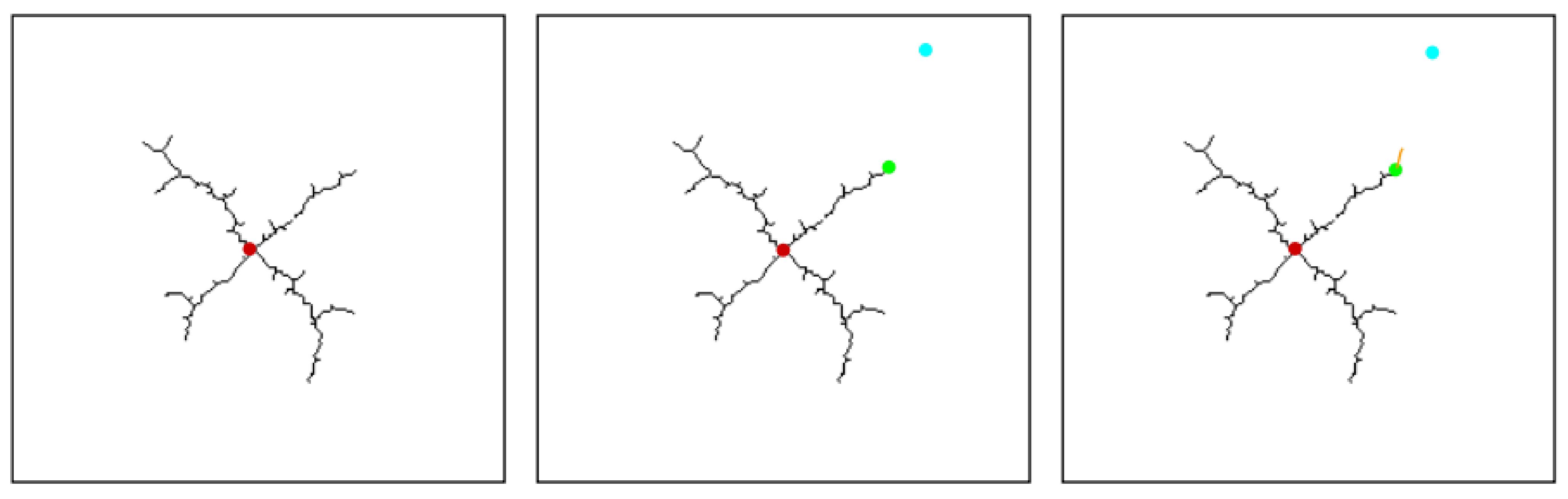


Probabilistic Roadmaps

- Considerations
  - Single query/multi query
  - How are nodes placed?
    - Uniform sampling strategies
    - Non-uniform sampling strategies
  - How are local neighbors found?
  - How is collision detection performed?
    - Dominates time consumption in PRMs



Rapidly Exploring Random Trees (RRT)



1. Maintain a tree rooted at the starting point
2. Choose a point at random from free space
3. Find the closest configuration already in the tree
4. Extend the tree in the direction of the new configuration

Rapidly Exploring Random Trees (RRT) – Uniform/biased sampling

Aaron Becker, UH, Wolfram Player example

Rapidly Exploring Random Trees (RRT) - Considerations

- **Sensitive to step-size (Δq)**
  - Small: many nodes, closely spaced, slowing down nearest neighbor computation
  - Large: Increased risk of suboptimal plans / not finding a solution
- **How are samples chosen?**
  - Uniform sampling may need too many samples to find the goal
  - Biased sampling towards goal can ease this problem
- **How are closest neighbors found?**
- **How are local paths generated?**
- **Variations**
  - RRT Connect, A*-RRT, Informed RRT*, Real-Time RRT*, Theta*-RRT, etc.

ECE 4160/5160
MAE 4910/5910

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

Fast Robots
Graph Search

Modelling path planning as a graph search problem

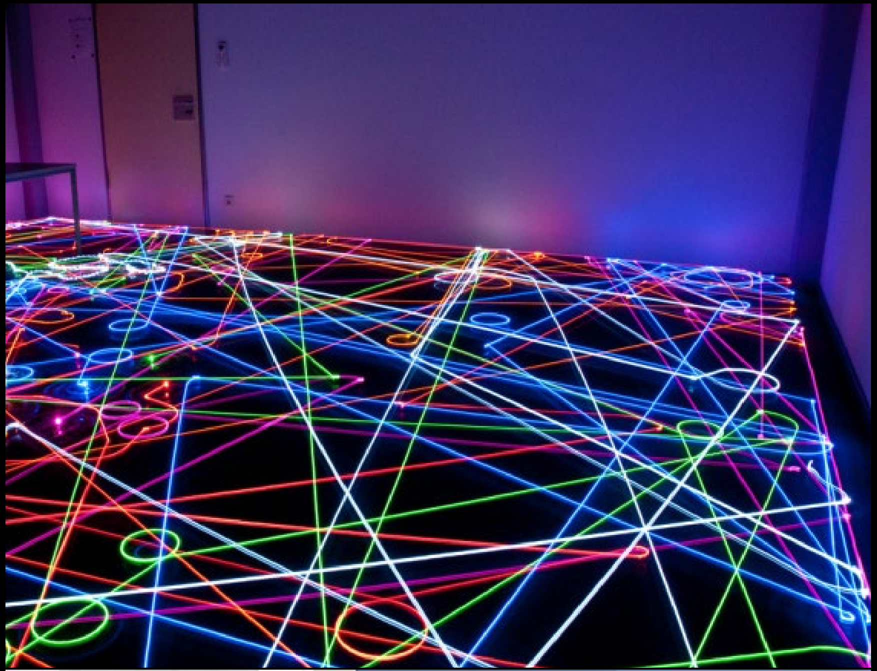
Real world → Configuration Space → Map Representation → Graph Construction → Graph Search

https://pythonrobotics.readthedocs.io/en/latest/modules/path_planning.html#basic-rrt

- Breadth first
- Depth first
- Dijstra
- A*

Graph Search

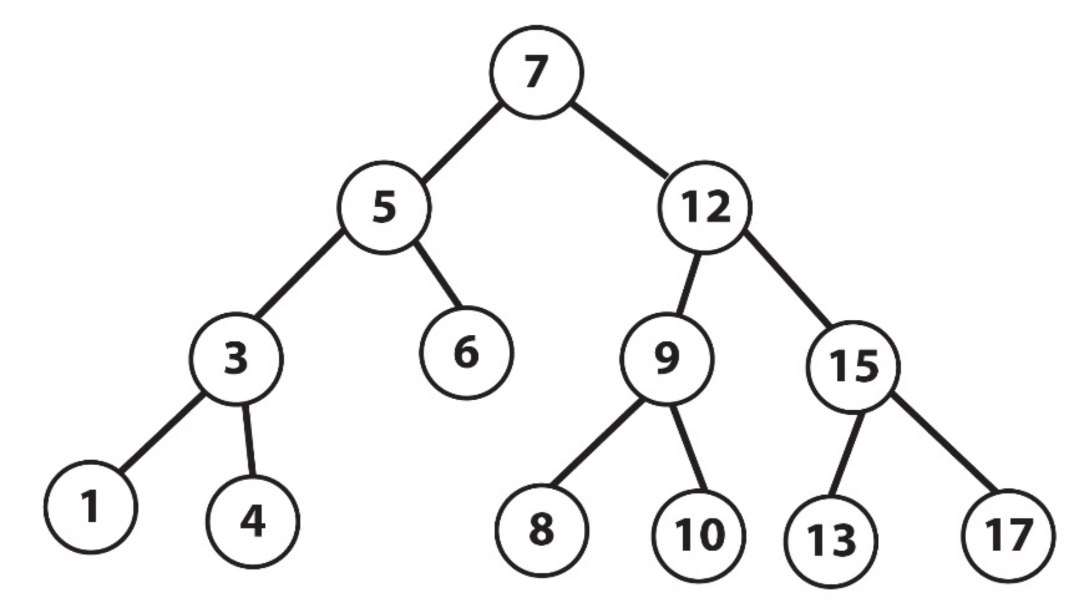
- **What is the simplest thing to do?**
  - Random or brute force search
- **Other methods?**
  - Uninformed search
    - Depth First Search (DFS)
    - Breadth First Search (BFS)
    - Dijsktra’s Search (LCFS)
  - Informed Search
    - Greedy
    - A*
    - (and many more)



Comparing Search Algorithms

Vocabulary

- Node, edge, parents/children, branching factor, depth



Definitions

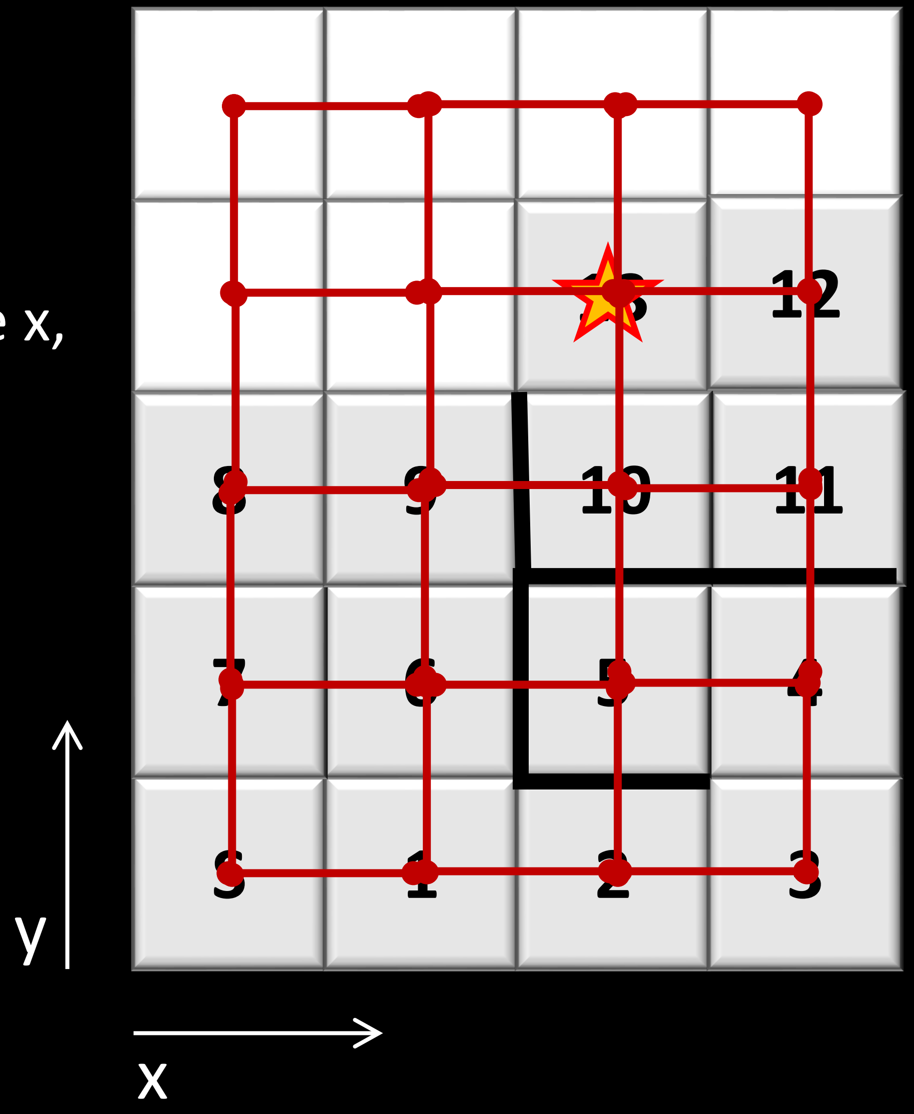
- Complete
  - Guaranteed to find a solution in finite time
- Time complexity
  - Worst-case run time
- Space complexity
  - Worst-case memory
- Optimality
  - A search is optimal if it is complete, and only returns cost-minimizing solutions

Search order: N, E, S, W

Algorithms and Search

- **What is the simplest thing to do?**
  - Random or brute force search
  - **How many grid traversals will brute force take?**
    - First establish a search order
    - Advance x first, then increment y and decrease x, etc.

Find a goal

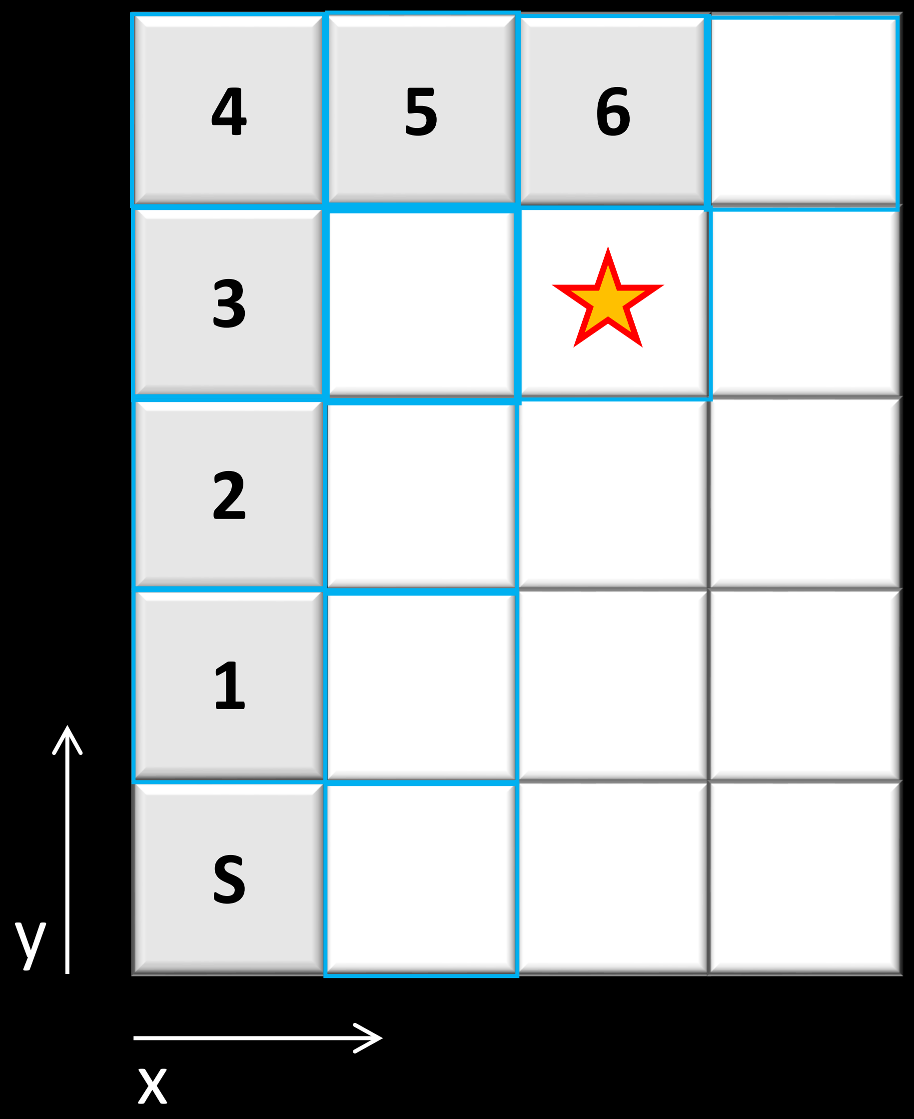


Algorithms and Search

- **What is the simplest thing to do?**
  - Random or brute force search
- **Other methods?**
  - Depth First Search (DFS)

Search order: N, E, S, W

Find a goal



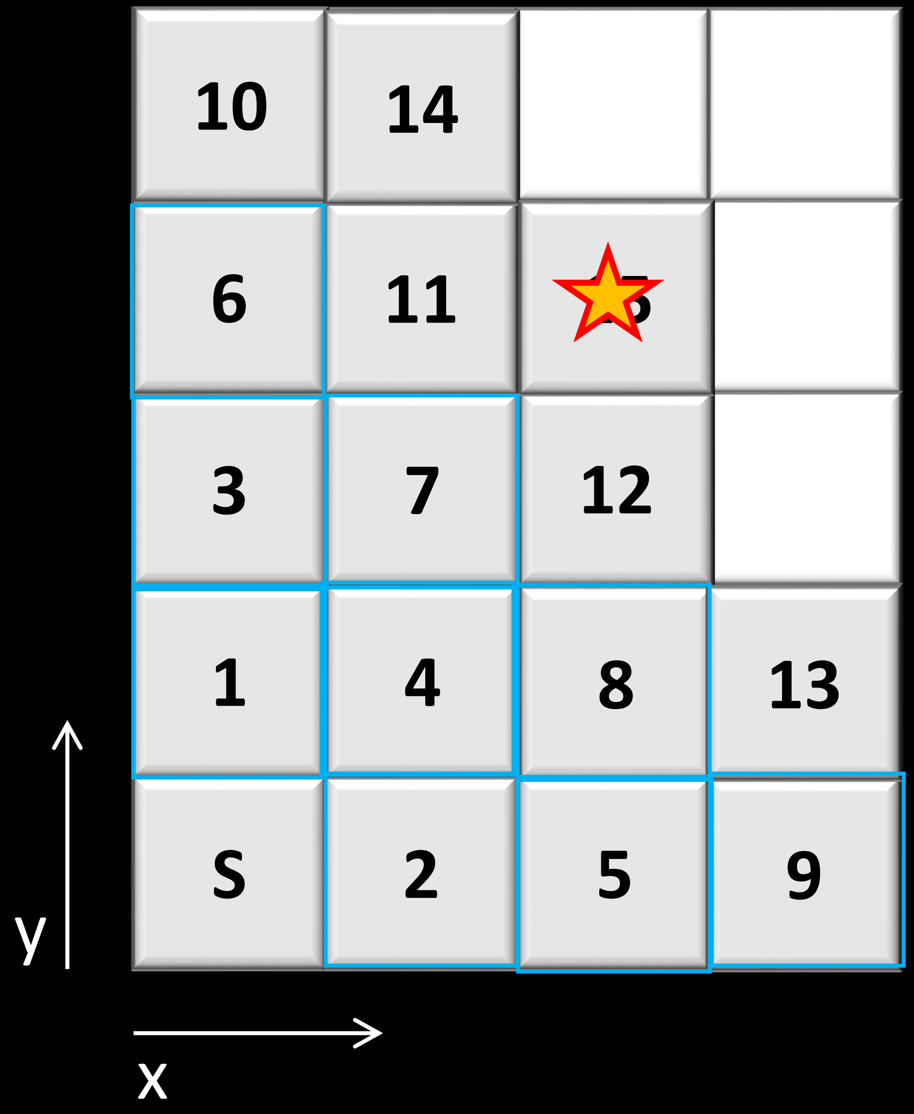
Fast Robots

Algorithms and Search

- **What is the simplest thing to do?**
  - Random or brute force search
- **Other methods?**
  - Depth First Search (DFS)
  - Breadth First Search (BFS)

Search order: N, E, S, W

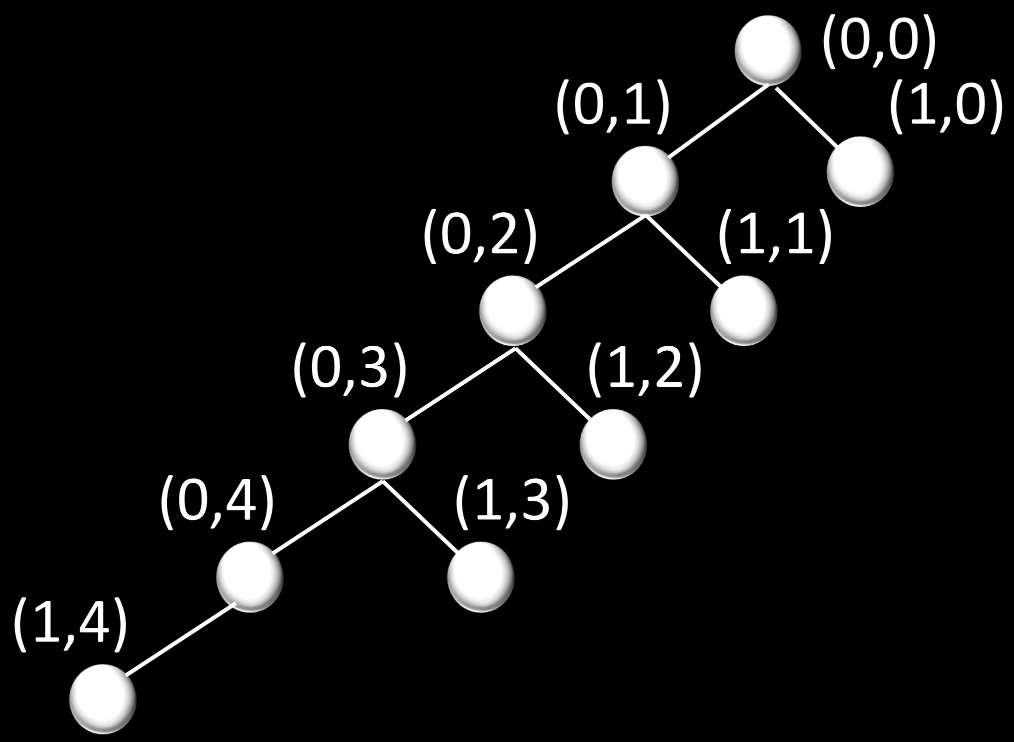
Find a goal



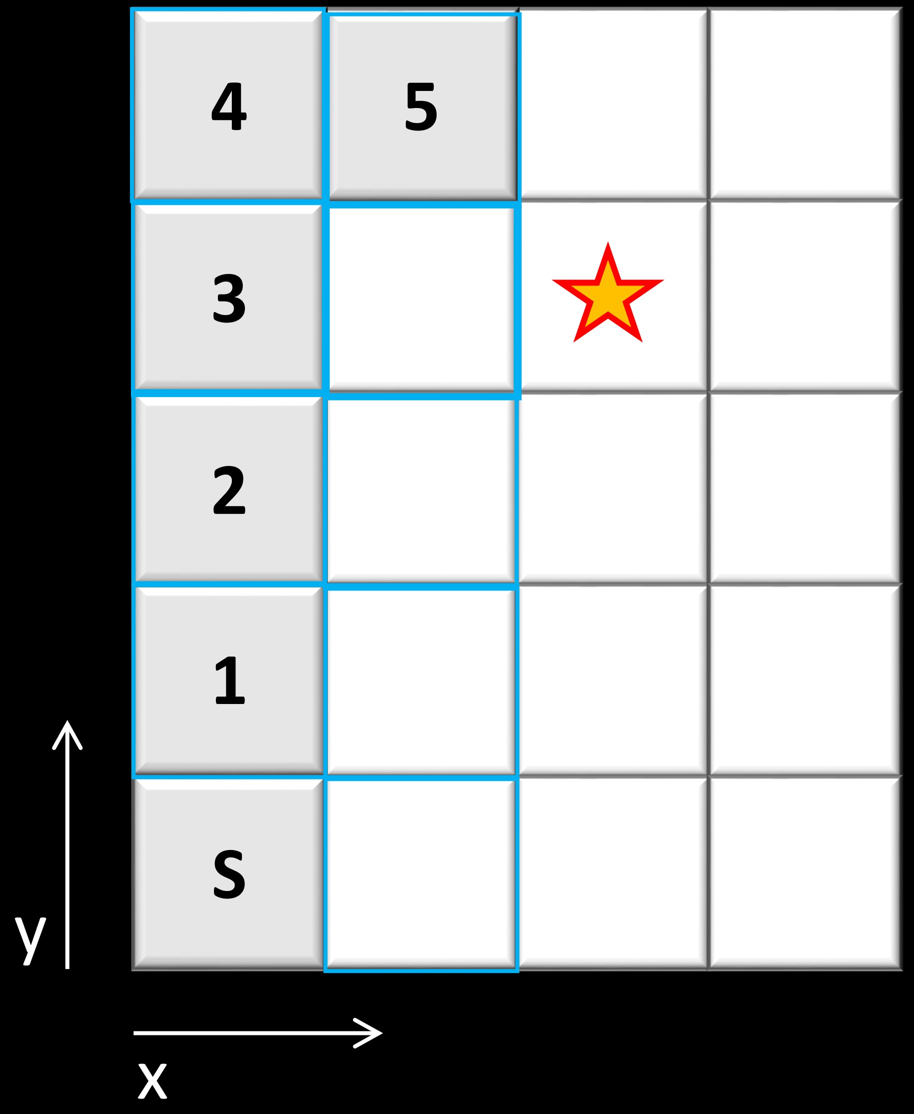
Depth First Search (DFS)

Search order: N, E, S, W

Find a goal

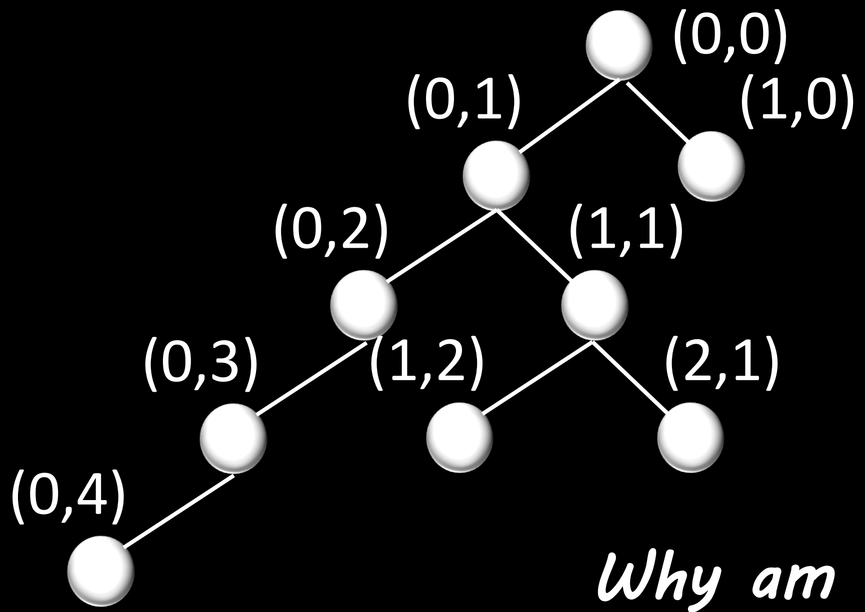


and so on…



Depth First Search (DFS)

Search order: N, E, S, W

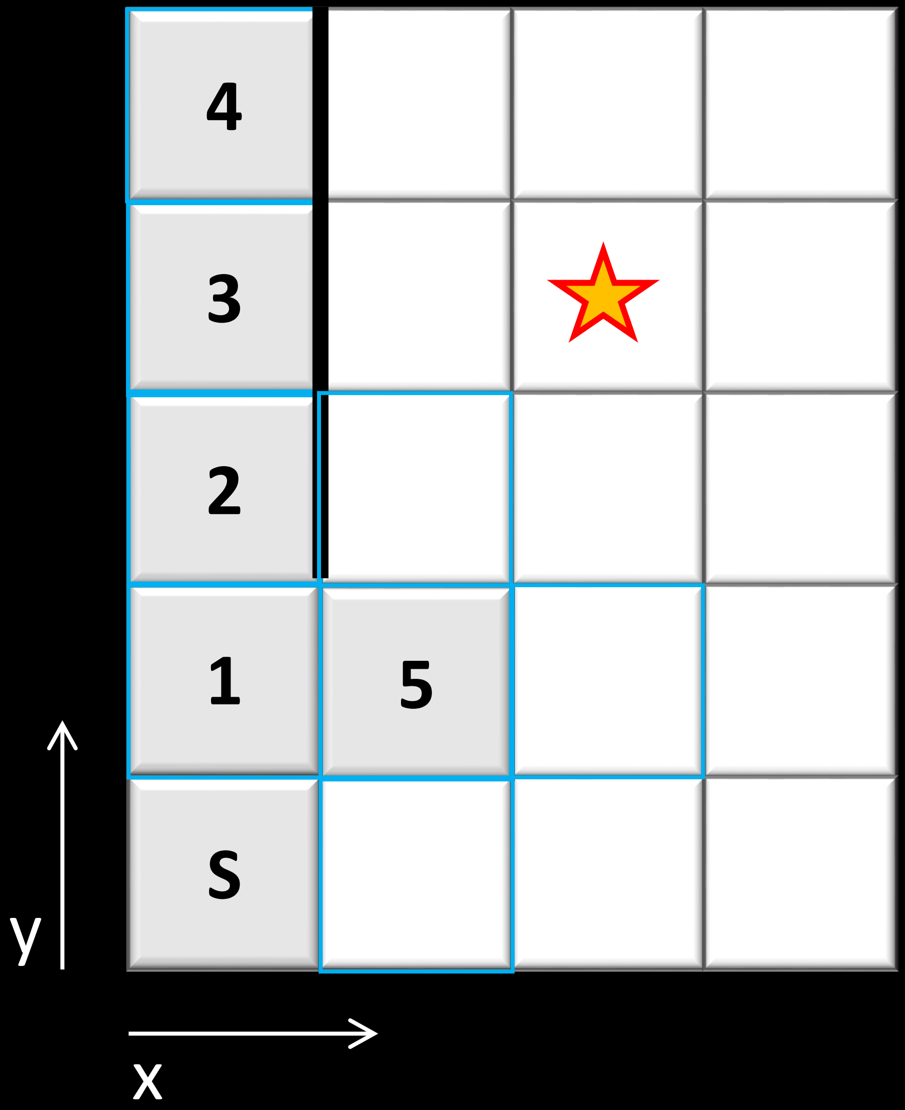


and so on…

Why am I not also adding (1,0)?

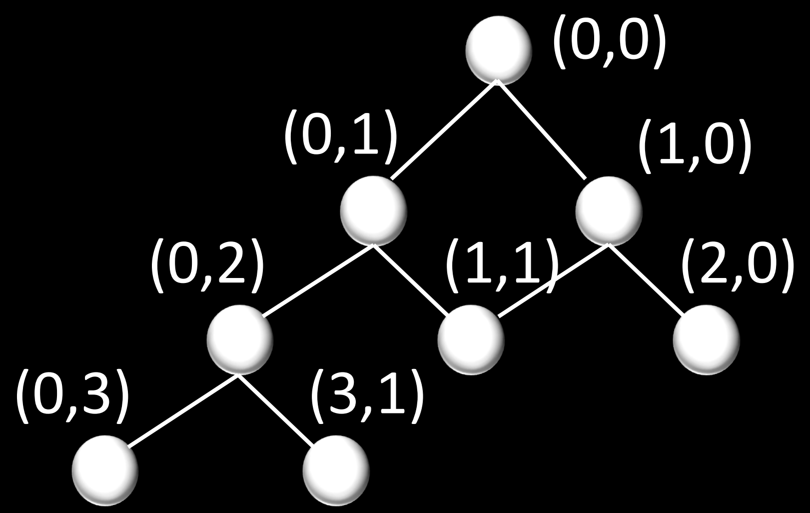
- Keep track of what is already on the frontier

Find a goal



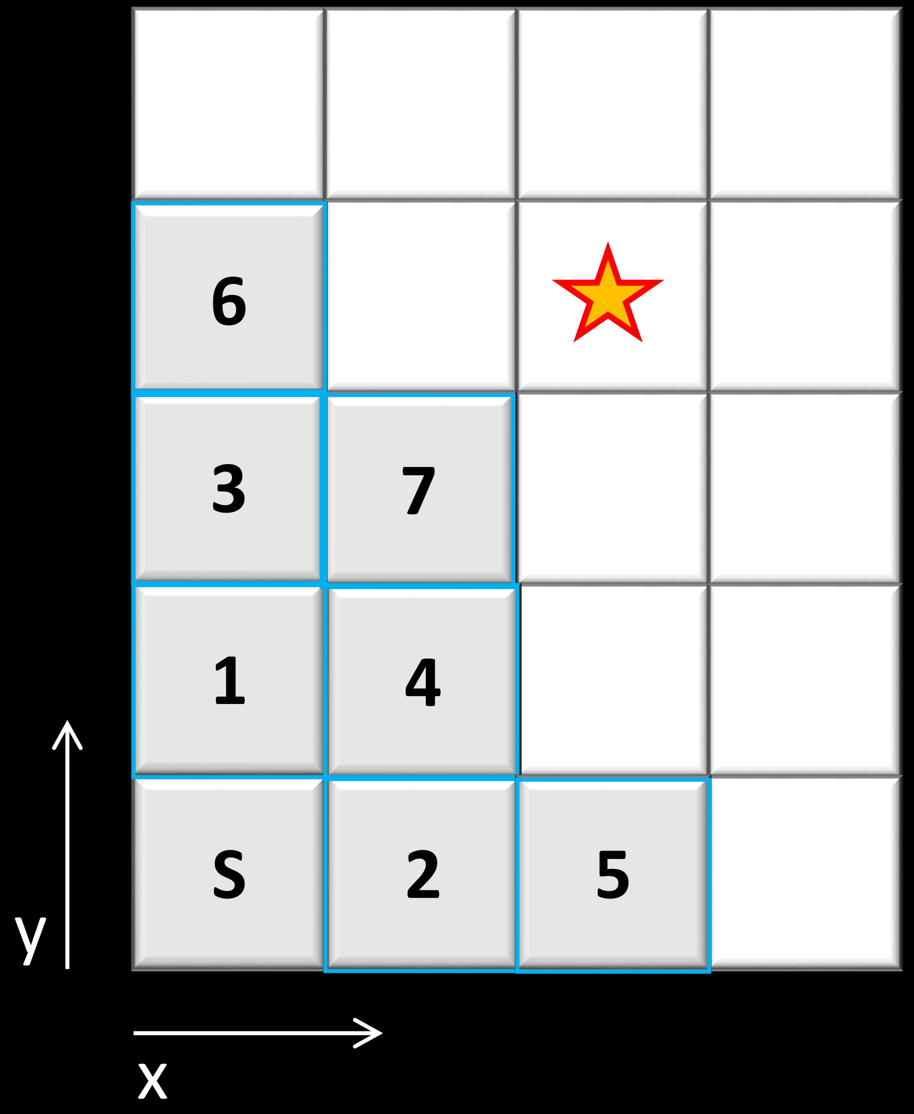
Breadth First Search (BFS)

Search order: N, E, S, W



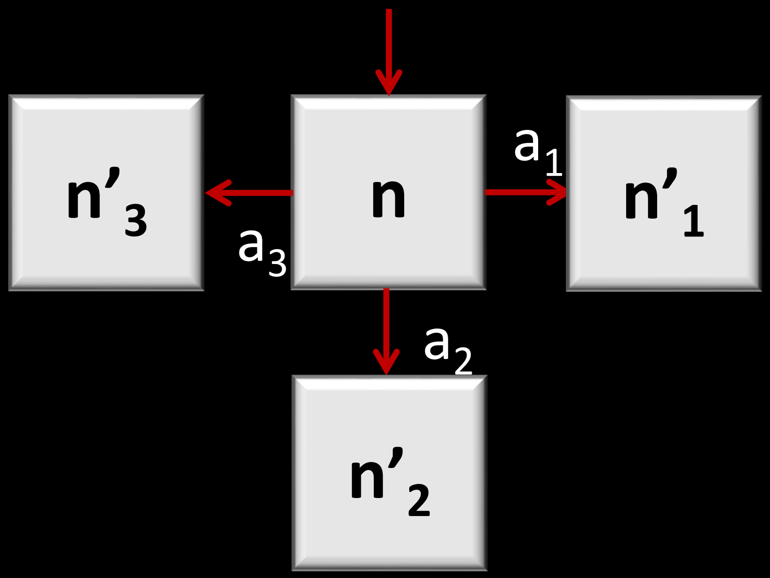
and so on…

Find a goal

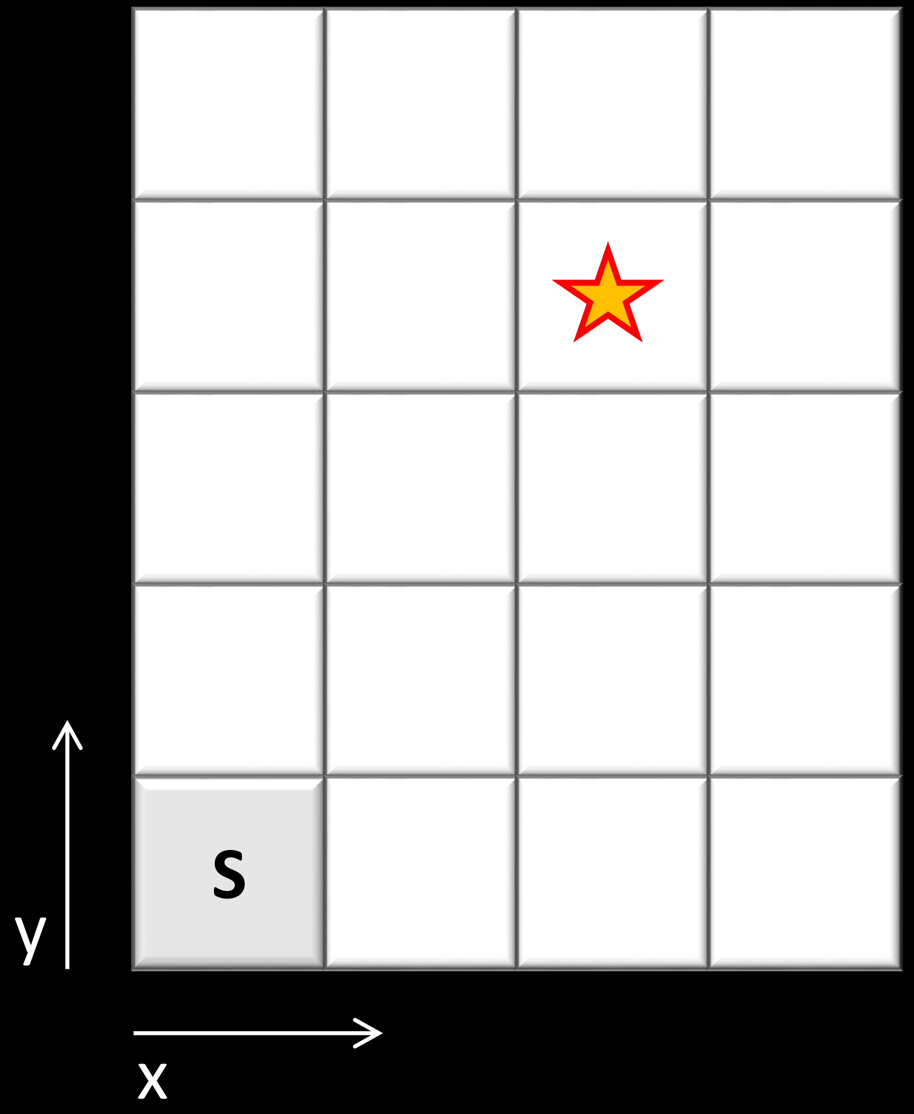


Search Algorithms, General

- Common graph structure
  - For every node, n
  - you have a set of actions, a
  - that moves you to a new node, n’



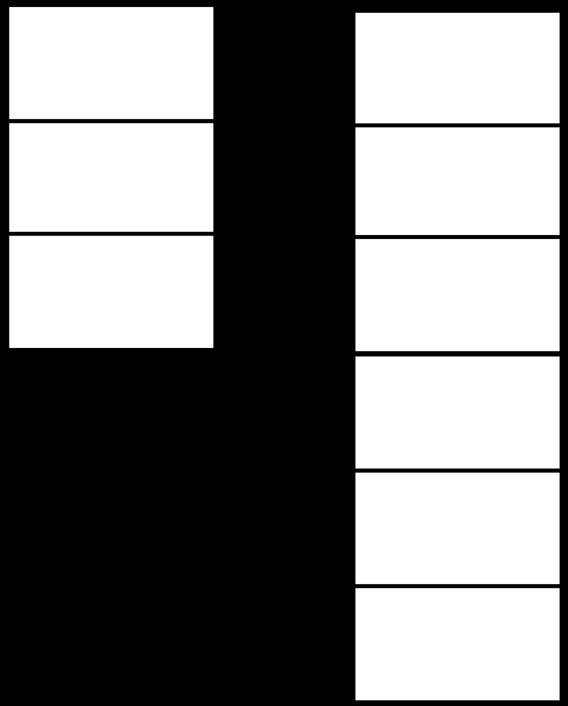
Find a goal

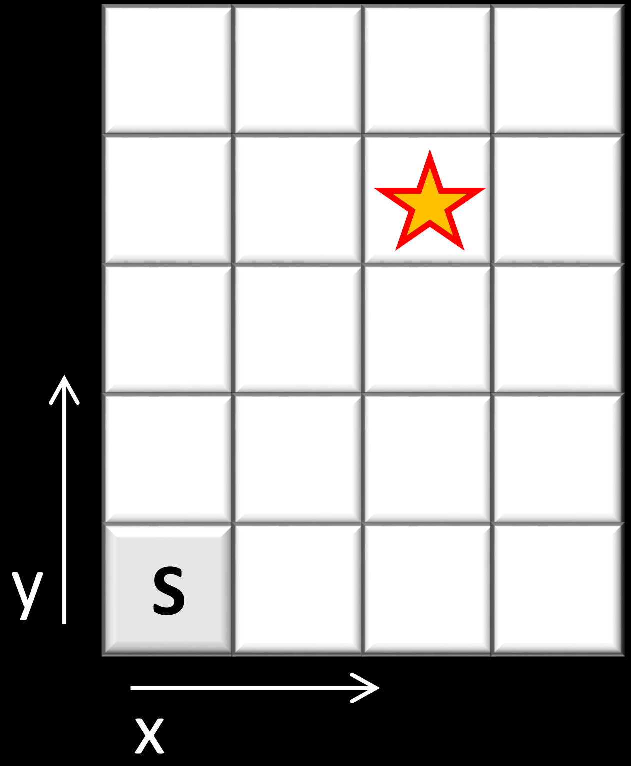


Search Algorithms, General

```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   append n to visited
   if n = goal, return solution
   for all actions in n
      n' = a(n)
      if n' not visited
            append n' to frontier
```

frontier visited

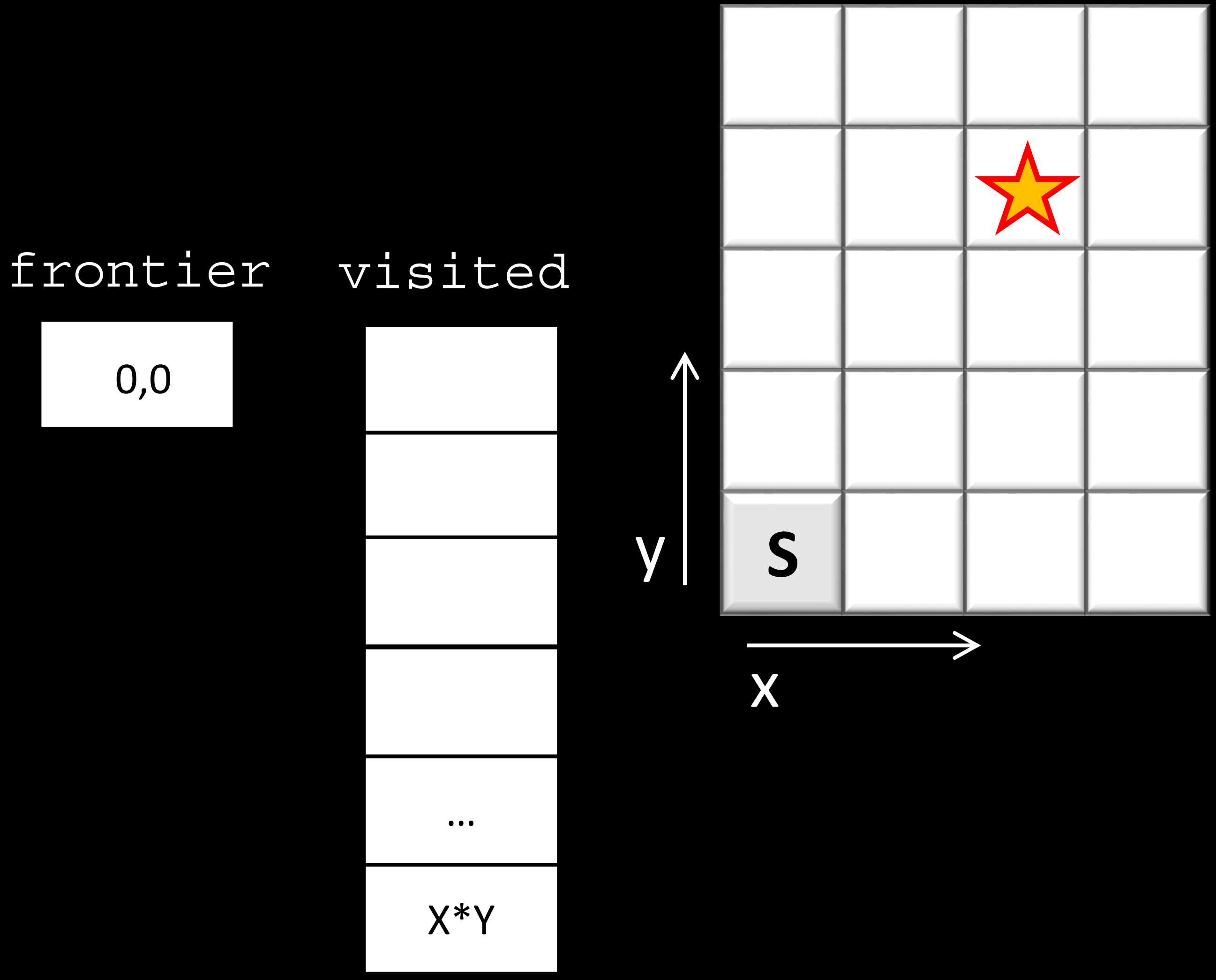




How much space to allocate to your buffers?

Depth First Search (DFS)

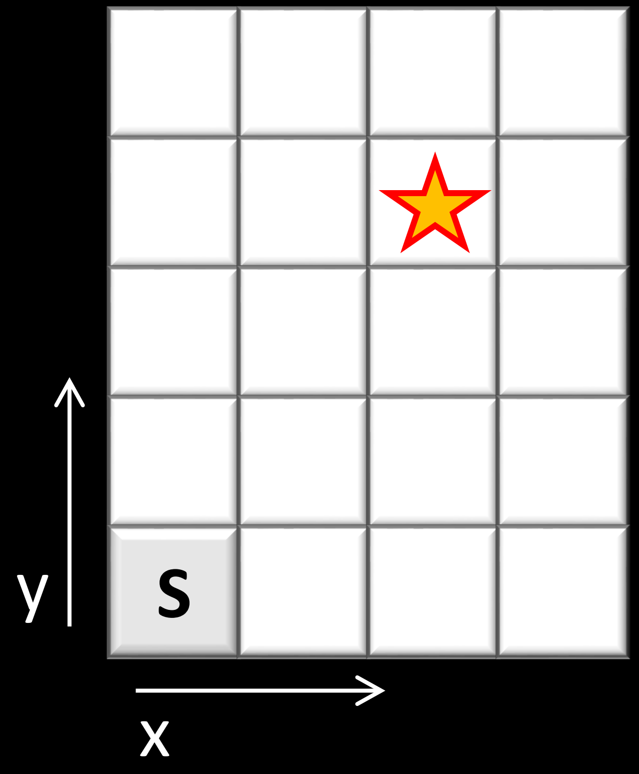
```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   append n to visited
   if n = goal, return solution
   for all actions in n
      n’ = a(n)
      if n’ not visited
            append n’ to frontier
```



Depth First Search (DFS)

```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   append n to visited
   if n = goal, return solution
   for all actions in n
      n' = a(n)
      if n' not visited
            append n' to frontier
```

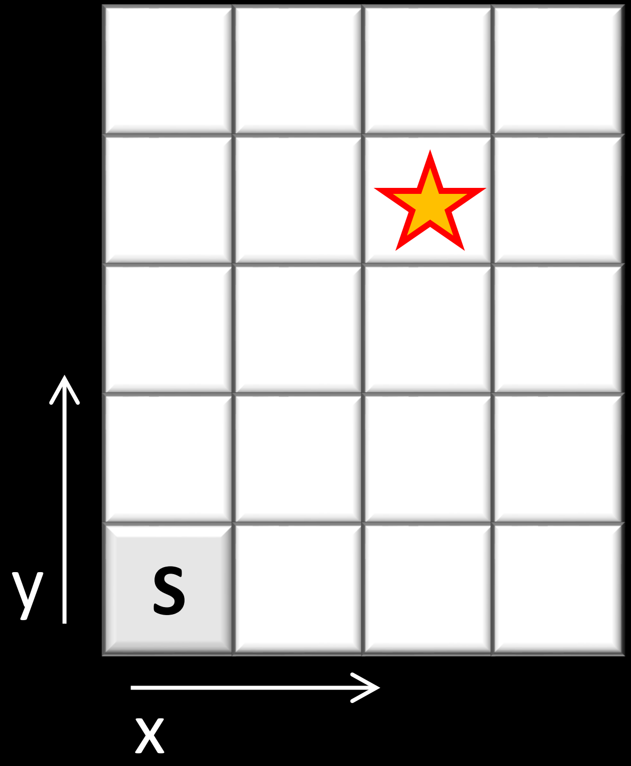
frontier	visited
0,1	0,0
1,0	
	…
	X*Y



Depth First Search (DFS)

```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   append n to visited
   if n = goal, return solution
   for all actions in n
      n’ = a(n)
      if n’ not visited
            append n’ to frontier
```

frontier	visited
0,1	0,0
1,0	0,1
1,0	
	…
	X*Y

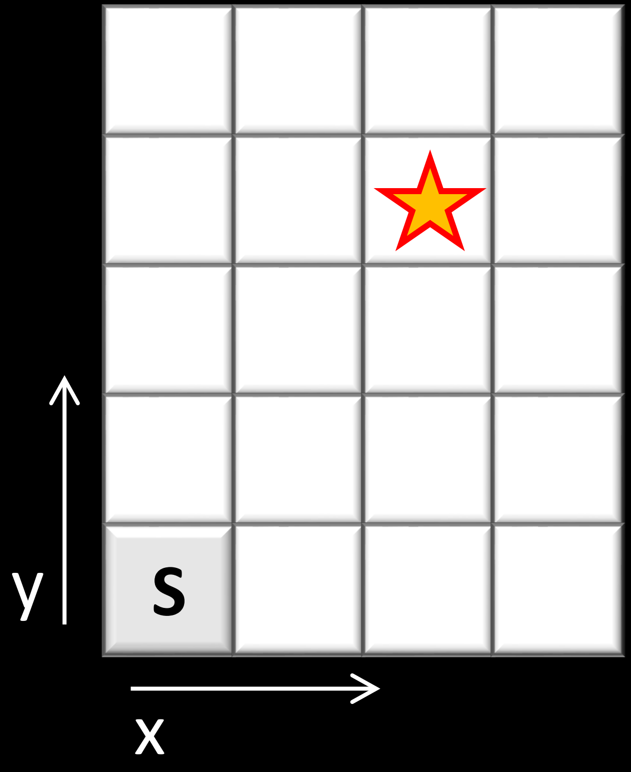


Depth First Search (DFS)

```
n = state(init)
frontier.append(n)
while(frontier not empty)
    n = pull state from frontier
    append n to visited
    if n = goal, return solution
    for all actions in n
        n' = a(n)
        if n' not visited
                append n' to frontier
```

frontier
0,2
1,1
1,0

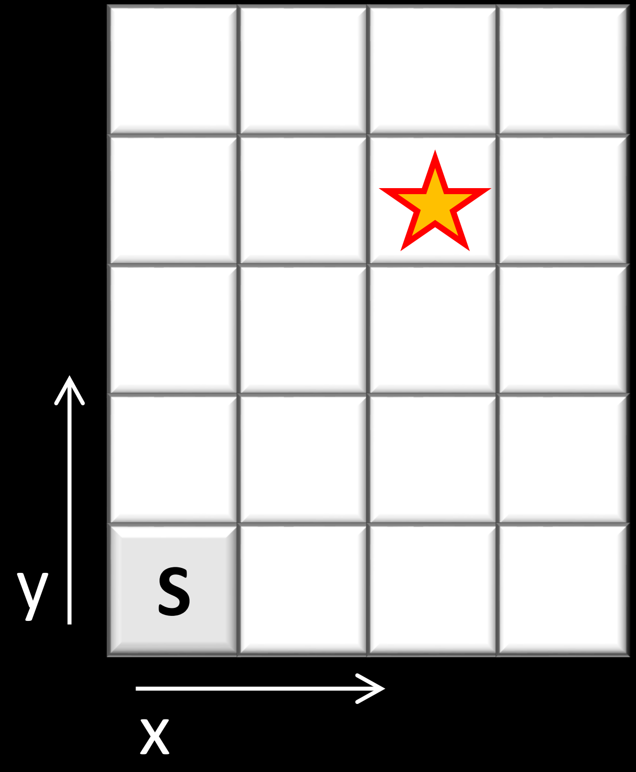
visited
0,0
0,1
0,2
…
X*Y



Depth First Search (DFS)

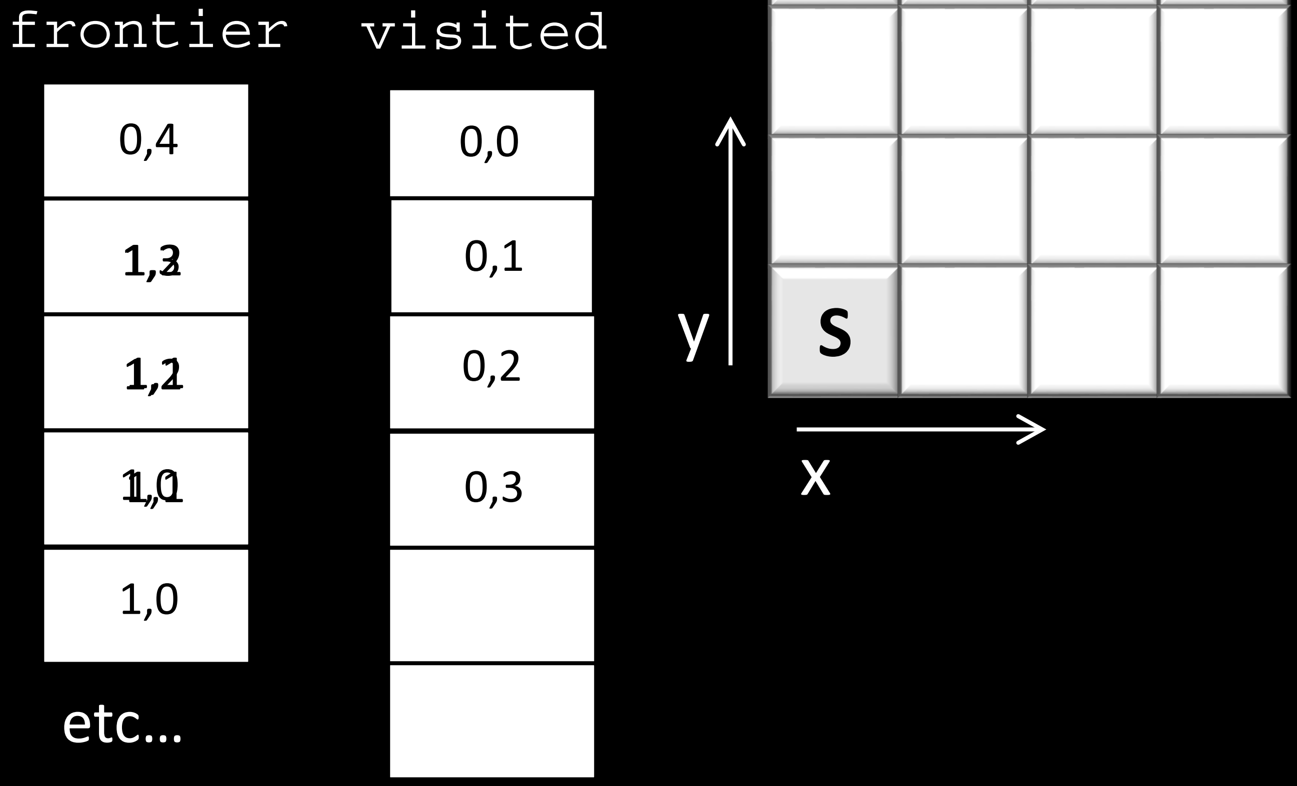
```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   append n to visited
   if n = goal, return solution
   for all actions in n
      n’ = a(n)
      if n’ not visited
            append n’ to frontier
```

frontier	visited
0,3	0,0
1,2	0,1
1,0	0,2
1,0	0,3



Depth First Search (DFS)

```
n = state(init)
frontier.append(n)
while(frontier not empty)
    n = pull state from frontier
    append n to visited
    if n = goal, return solution
    for all actions in n
        n' = a(n)
        if n' not visited
                append n' to frontier
```

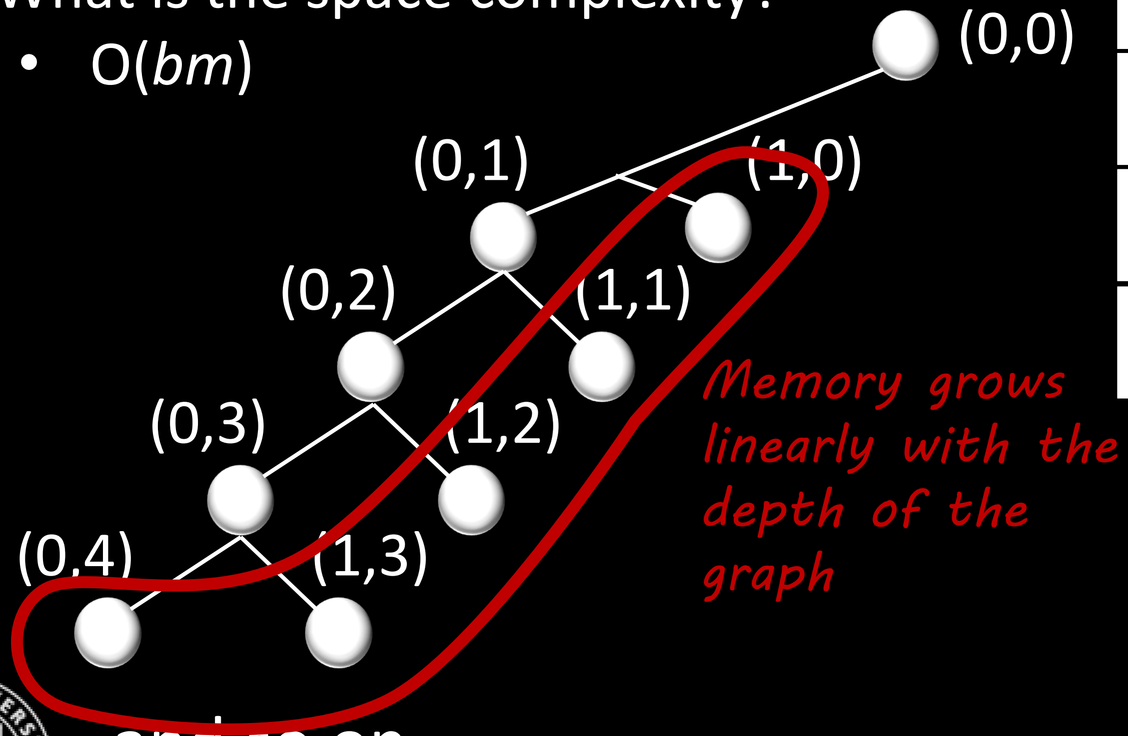


Frontier Buffer?

- ***Last-In First-Out (LIFO) Buffer***

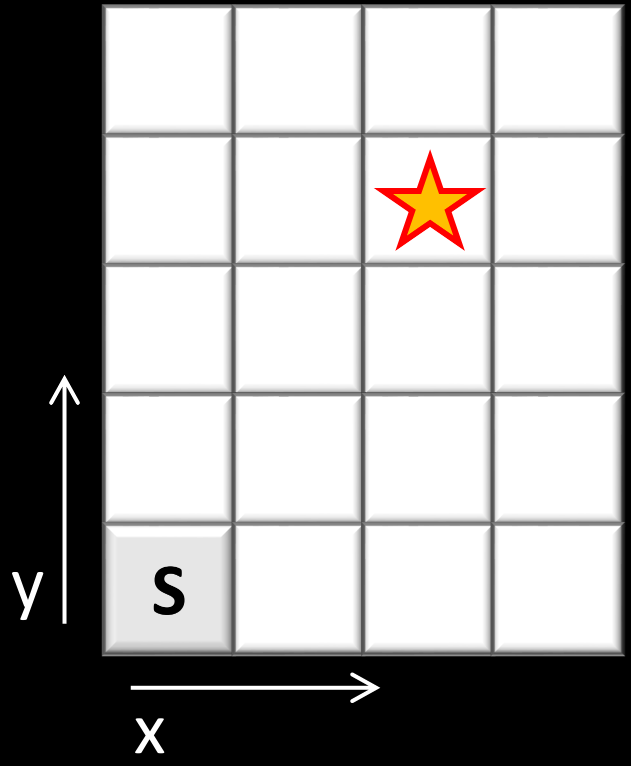
Depth First Search (DFS)

- Is it complete?
  - Yes, but only on finite graphs
- What is the time complexity?
  - $O(b^m)$
- What is the space complexity?
  - $O(bm)$



and so on...

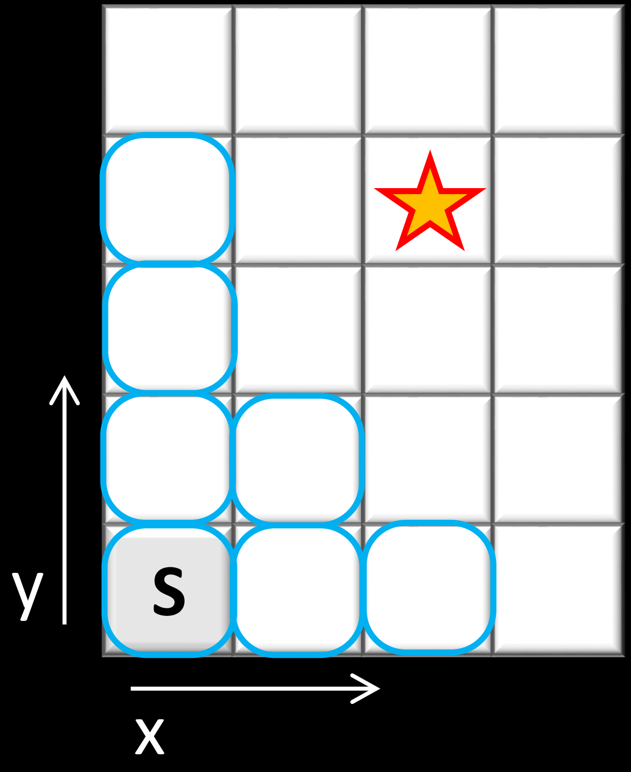
frontier	visited
0,4	0,0
1,3	0,1
1,2	0,2
1,1	0,3
1,0	…
	X*Y



Breadth First Search (BFS)

```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   if n is goal, return solution
   for all actions in n
      n' = a(n)
      if n' not visited
         append n' to frontier
```

frontier	visited
0,0	0,0
0,1	0,1
1,0	1,0
0,2	0,2
1,1	
2,0	…
0,3	

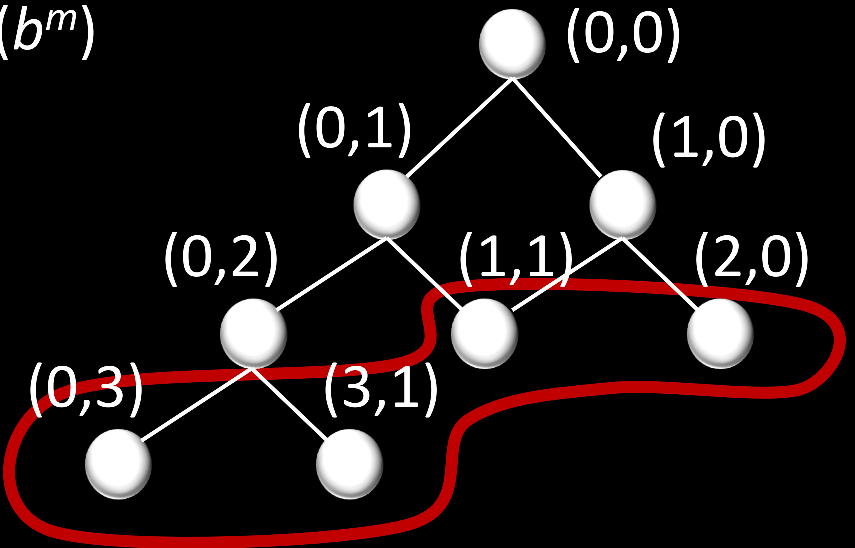


Type of Buffer?

First-In First-Out (FIFO) Buffer

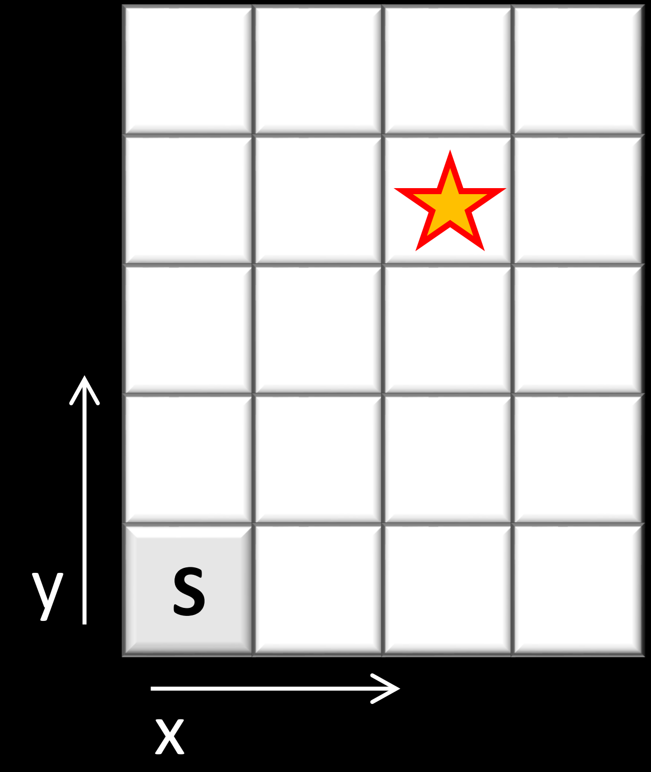
Breadth First Search (BFS)

- Is it complete?
  - Yes, as long as b is finite
- Is it optimal?
  - Yes
- What is the time complexity?
  - $O(b^m)$
- What is the space complexity?
  - $O(b^m)$



Memory grows exponentially with the depth of the graph

frontier	visited
	0,0
	0,1
	1,0
	0,2
1,1	
2,0	…
0,3	



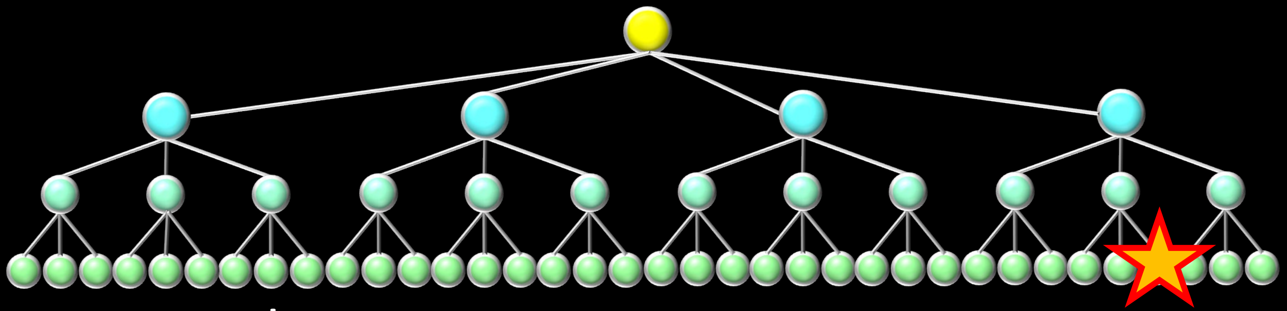
Type of Buffer?

First-In First-Out (FIFO) Buffer

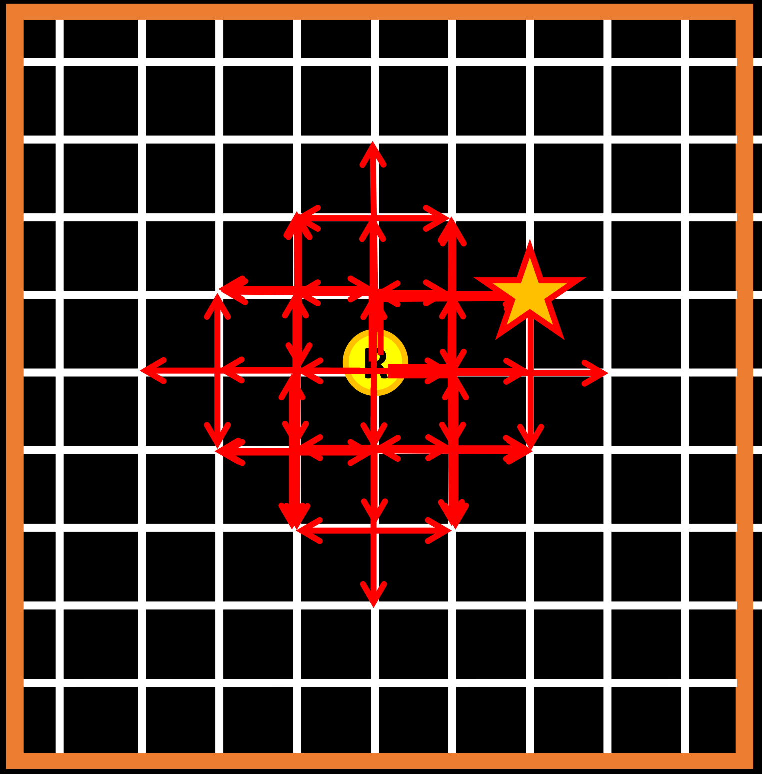
BFS: Memory and Computation

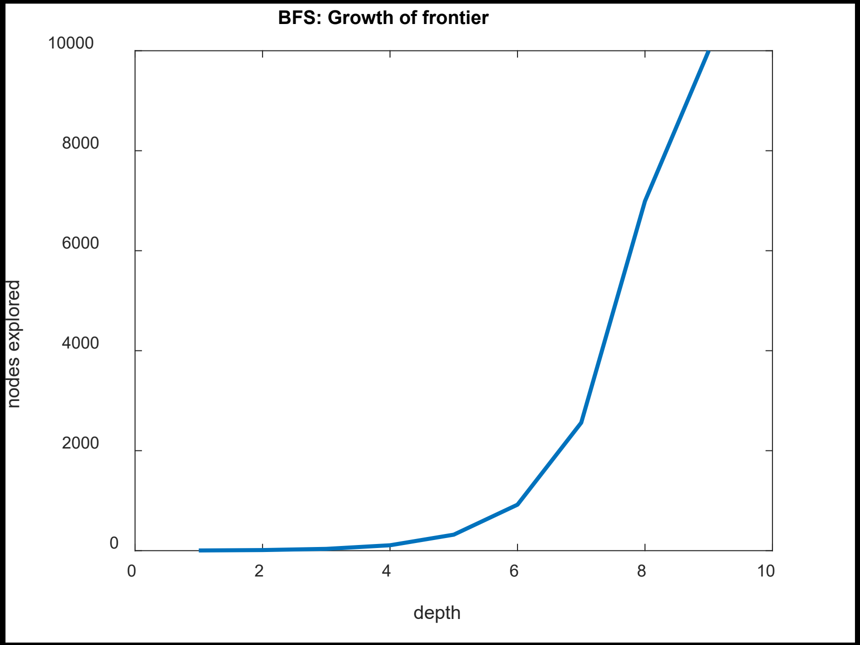
Frontier size:

- 4
- 12
- 36



etc…



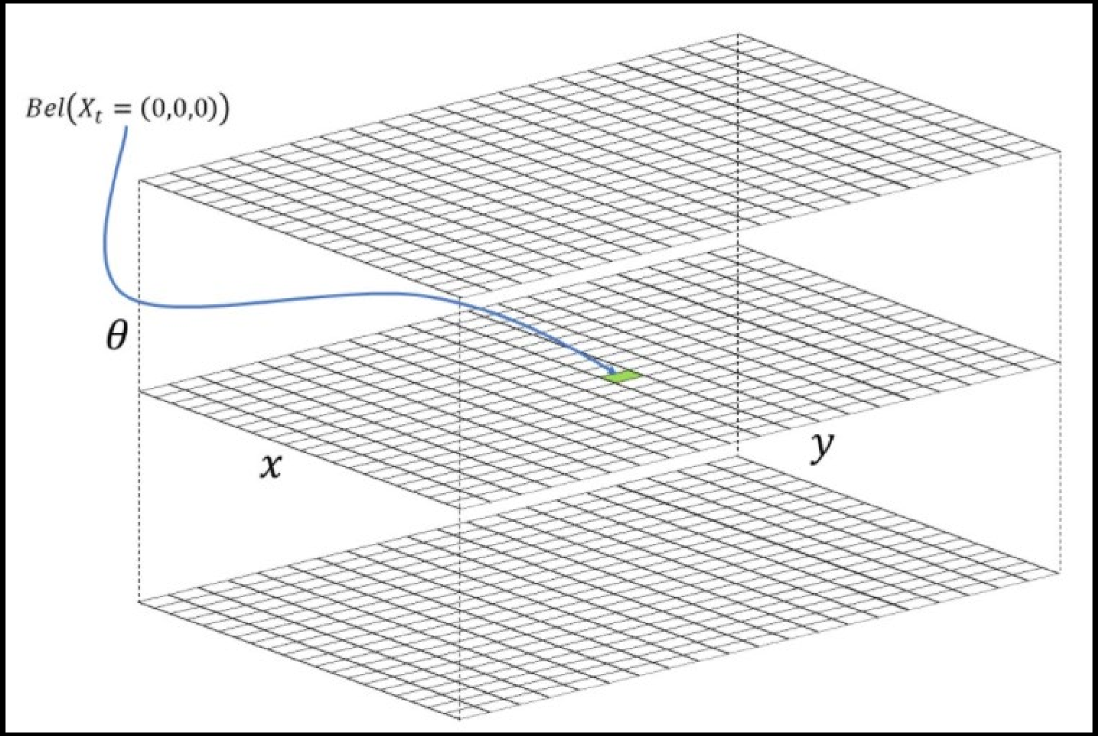


Uninformed Search Algorithms, General

- When is DFS appropriate?
  - If the memory is restricted
  - If solutions tend to occur at the same depth in the tree
- When is DFS inappropriate?
  - If some paths have infinite length / if the graph contains cycles
  - If some solutions are very deep, while others are very shallow
- When is BFS appropriate?
  - If you need to find the shortest path
  - If memory is not a problem
  - If some solutions are shallow
  - If there might be infinite paths
- When is BFS inappropriate?
  - If memory is limited / if the branching factor is very large
  - If solutions tend to be located deep in the tree

ECE4160/5160 – MAE 4910/5910 Fast Robots

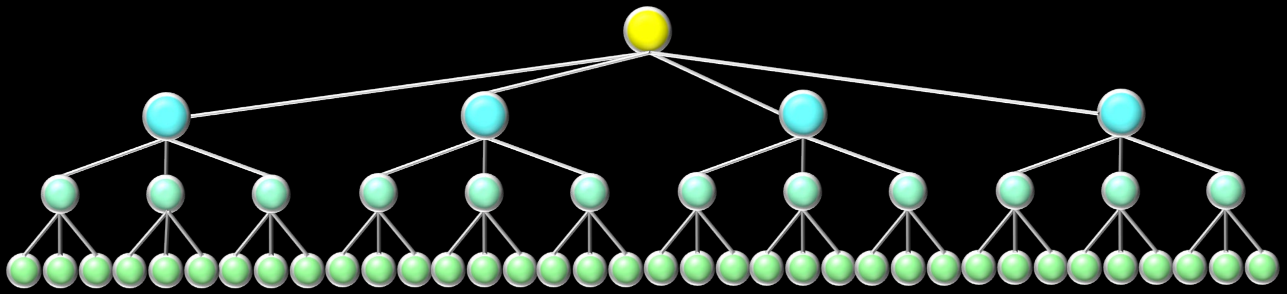
- Is BFS / DFS possible for your task on the Artemis?
  - What is the maximum branching factor?
    - b = 4
  - What is the longest path?
    - m ~ 20*20 = 400
  - Depth First Search
    - Frontier: O(*bm*) = 1,600 nodes
    - Float -> 6.4kB
    - Artemis memory?
      - 1MB flash and 384k RAM
  - Breadth First Search
    - Frontier: O(b^m) = 4^{20*20} = 6.7e240 nodes



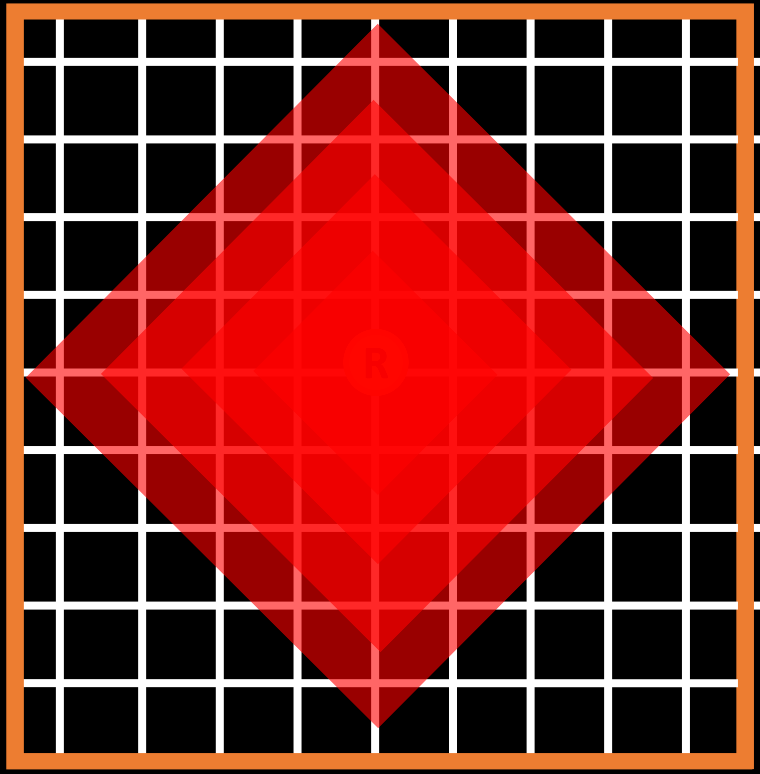
BFS: Memory and Computation

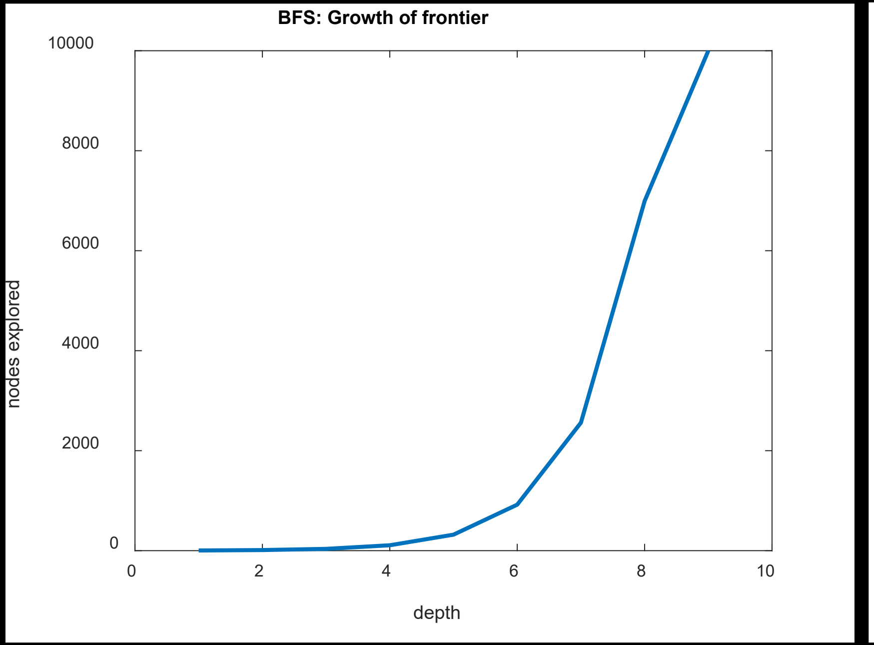
Frontier size:

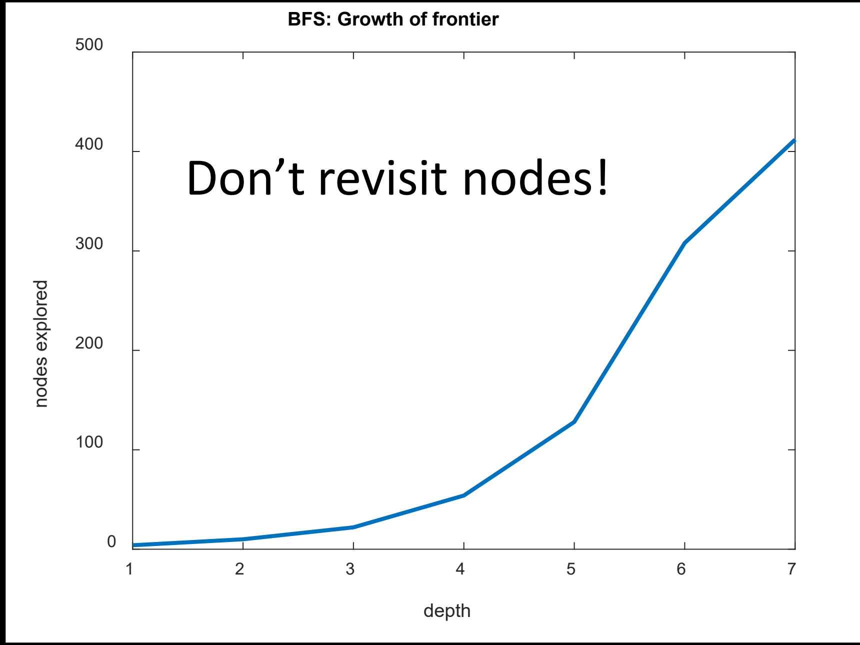
- 4
- 12
- 36



etc…

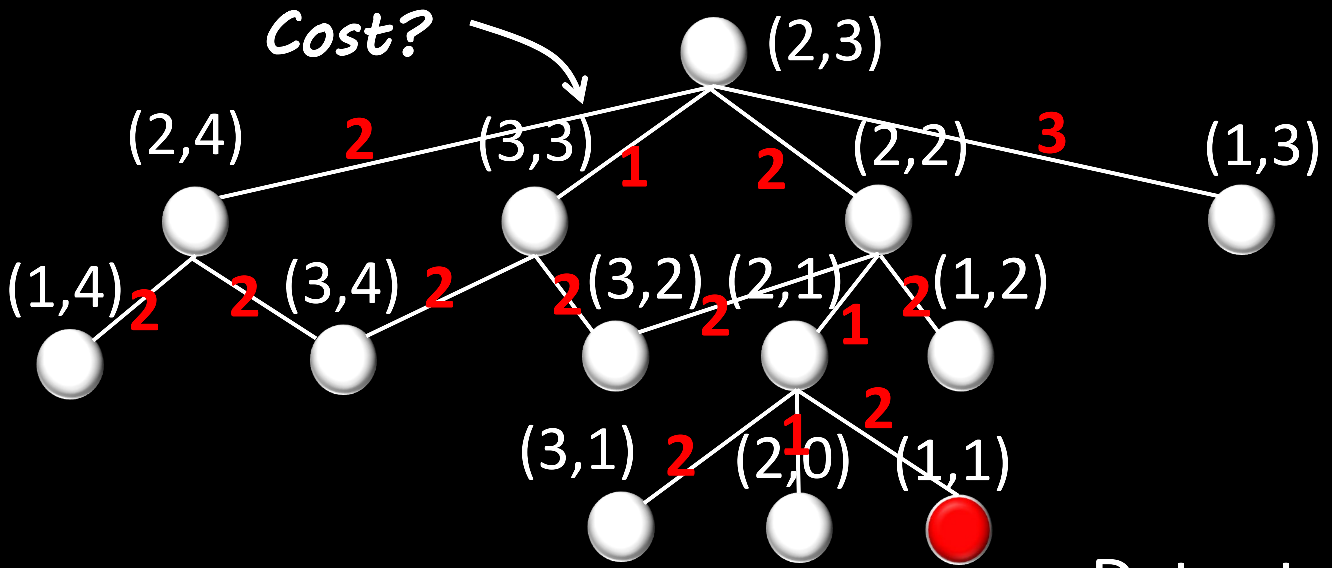






Lowest-Cost First Search

- Consider parent cost!



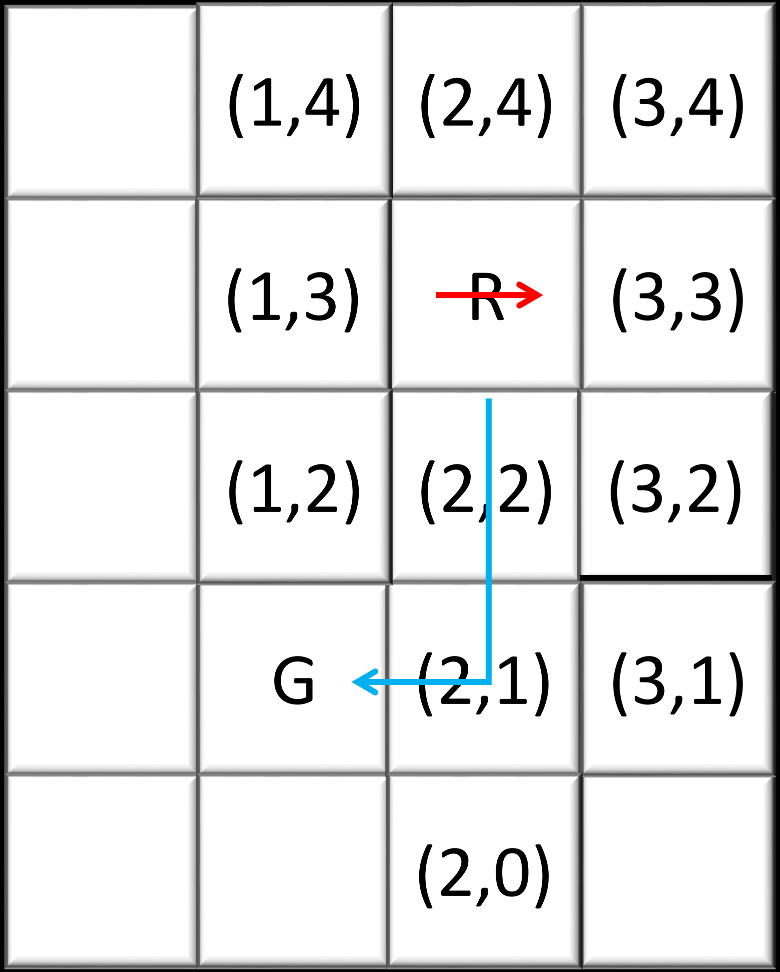
What node to expand next?

Data structure

- n.state
- n.parent
- n.cost
- n.action

What cost heuristic could we add?

- Go straight, cost 1
- Turn one quadrant, cost 1



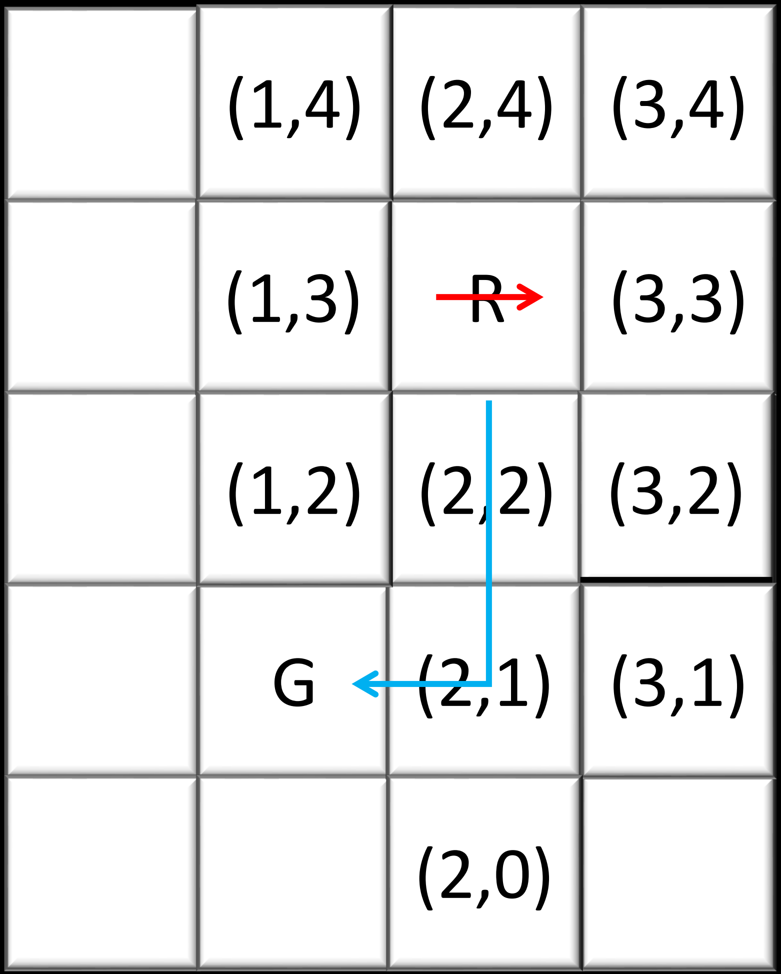
Lowest-Cost First Search

- Consider parent cost!

```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   visited.append(n)
   if n = goal, return solution
   for all actions in n
      n’ = a(n)
      if n’ not visited
            priority = heuristic(goal,n’)
            frontier.append(priority)
```

What cost heuristic could we add?

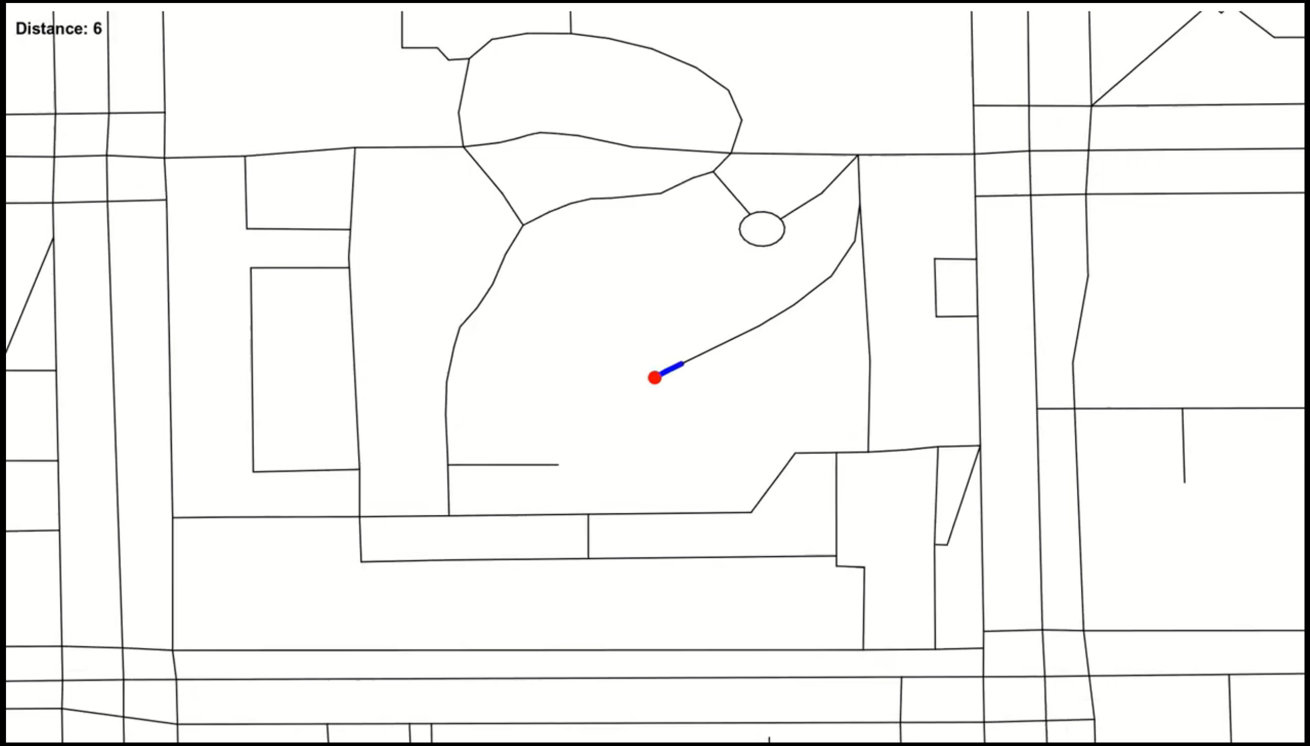
- Go straight, cost 1
- Turn one quadrant, cost 1



Lowest-Cost First Search

- Is it *complete*?
  - Yes, as long as path costs are positive
- What is the time complexity?
  - $O(b^m)$
- What is the space complexity?
  - $O(b^m)$

https://www.youtube.com/watch?v=t7UjtzqIXSA



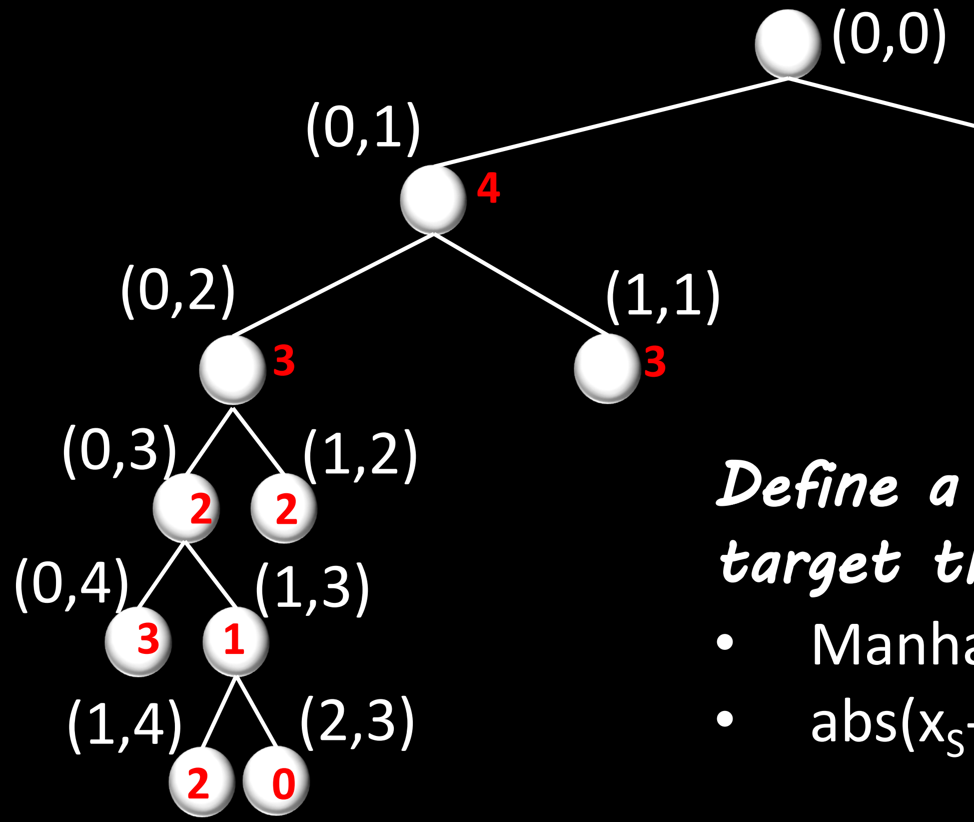
Could we be smarter?

- Sure, *you know the graph and you know the goal is*!
- …Informed search
  - Consider parent cost, and..
  - ..estimate the shortest path to the "goal"
- Assign a value to the frontier
  - Pick frontier closest to the goal (minimize distance)

Informed Search

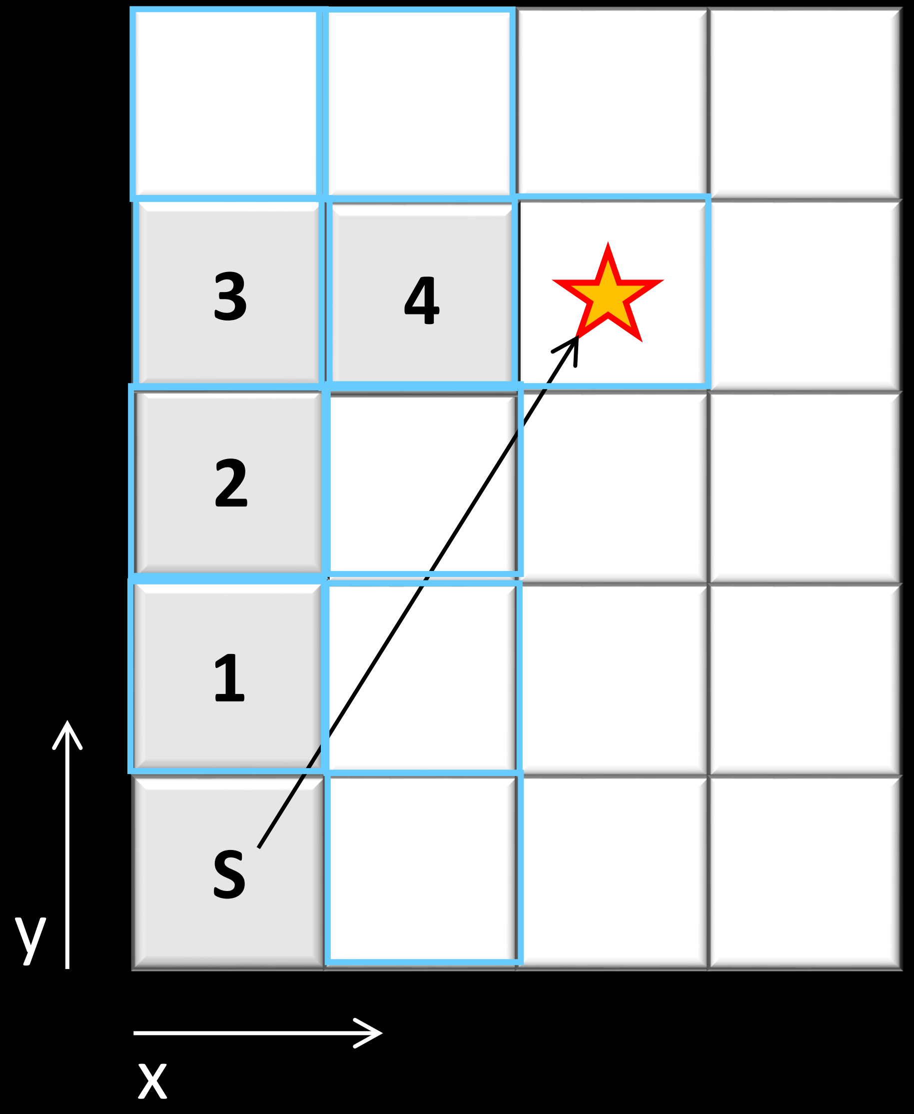
Search order: N, E, S, W

- Greedy Search



Define a heuristic to target the goal

- Manhatten distance
- $abs(x_S - x_G) + abs(y_S - y_G)$



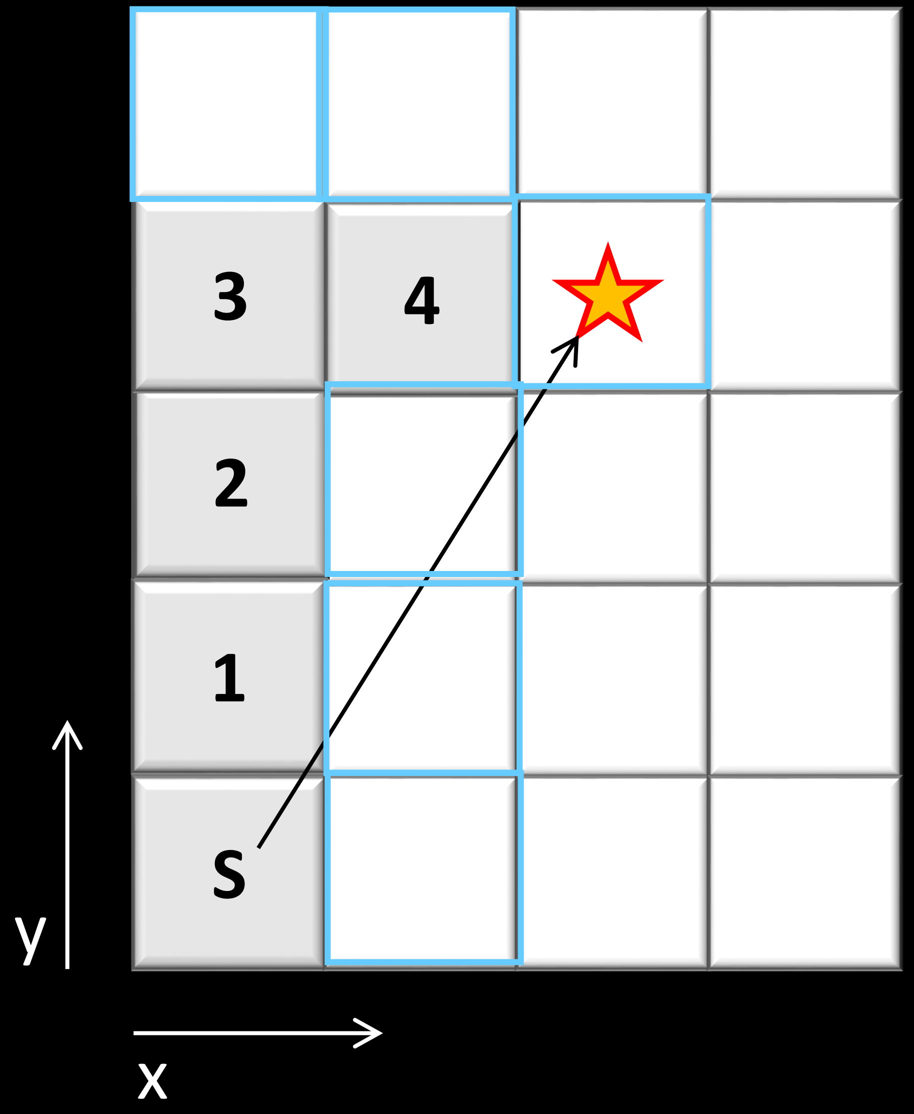
Fast Robots

Search order: N, E, S, W

Informed Search

- Greedy Search

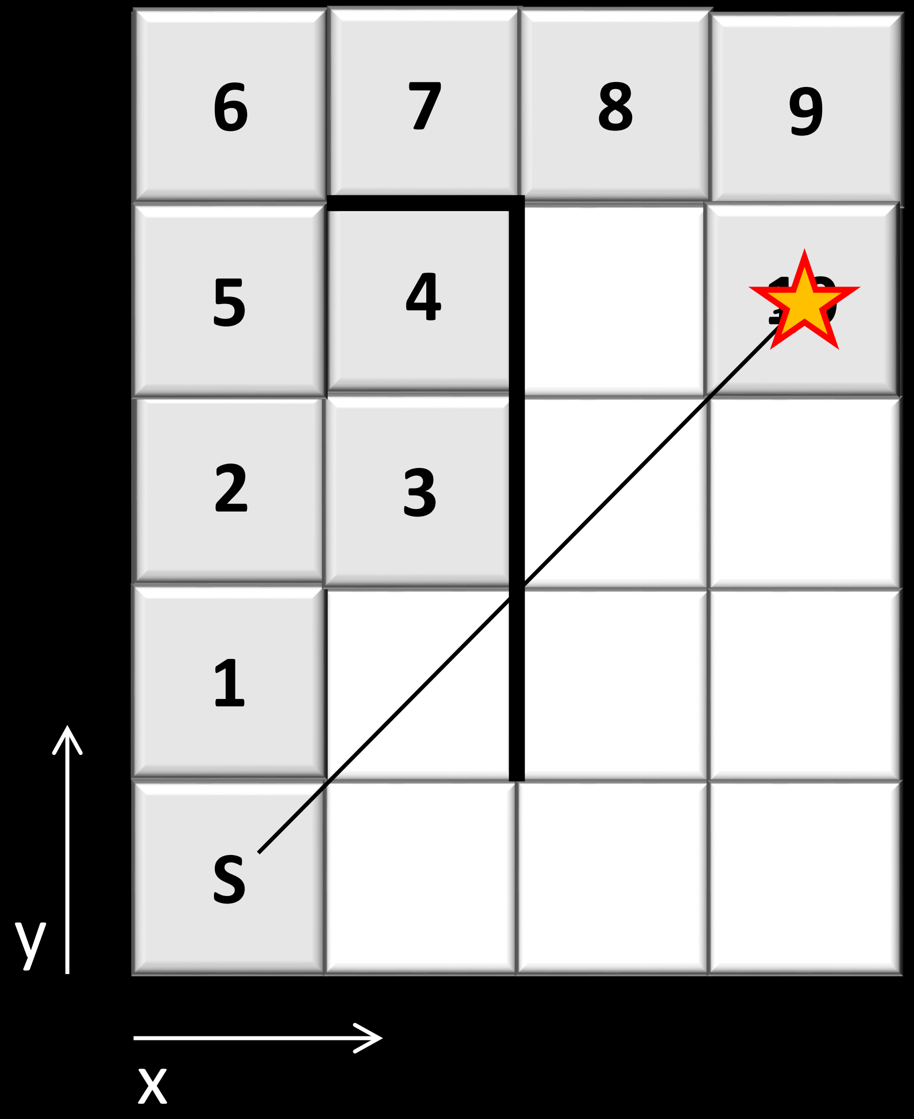
```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   visited.append(n)
   if n = goal, return solution
   for all actions in n
      n' = a(n)
      if n' not visited
            priority = heuristic(goal,n')
            frontier.append(priority)
```



Informed Search

Search order: N, E, S, W

- Greedy Search
  - Complete?
    - No
  - Time complexity?
    - $O(b^m)$
  - Space complexity?
    - $O(b^m)$
  - Optimal?
    - no…



Search Algorithms, General

- Breadth First Search
  - *Complete* and optimal
  - …but searches *everything*
- Lowest-Cost First Algorithm Considers parent cost
  - *Complete and optimal*
  - …but it wastes time exploring in directions that aren't promising
- Greedy Search Considers goal
  - *Complete (in most cases)*
  - …only explores promising directions

A*

Can we do better?

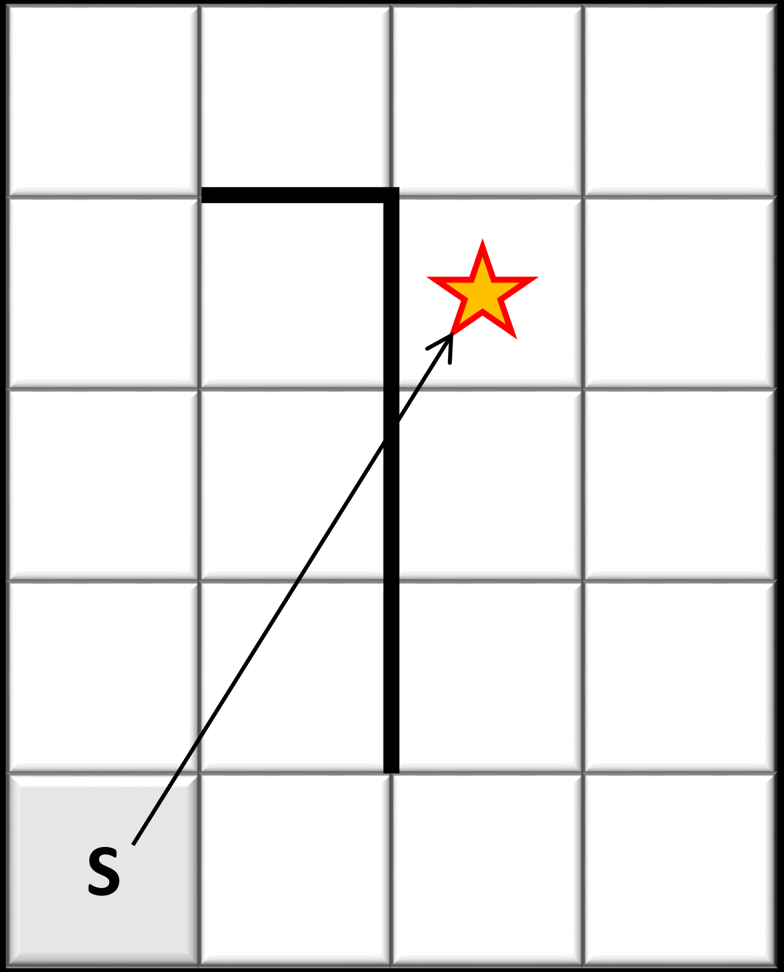
Informed Search

Search order: N, E, S, W

- A* (“A-star”)

```
n = state(init)
frontier.append(n)
while(frontier not empty)
    n = pull state from frontier
    if n = goal, return solution
    for all actions in n
        n’ = a(n)
        if ((n’ not visited or
            (visited and n’.cost < n_old.cost))
                priority = heuristic(goal,n’)+cost
                frontier.append(priority)
                visited.append(n’)
```

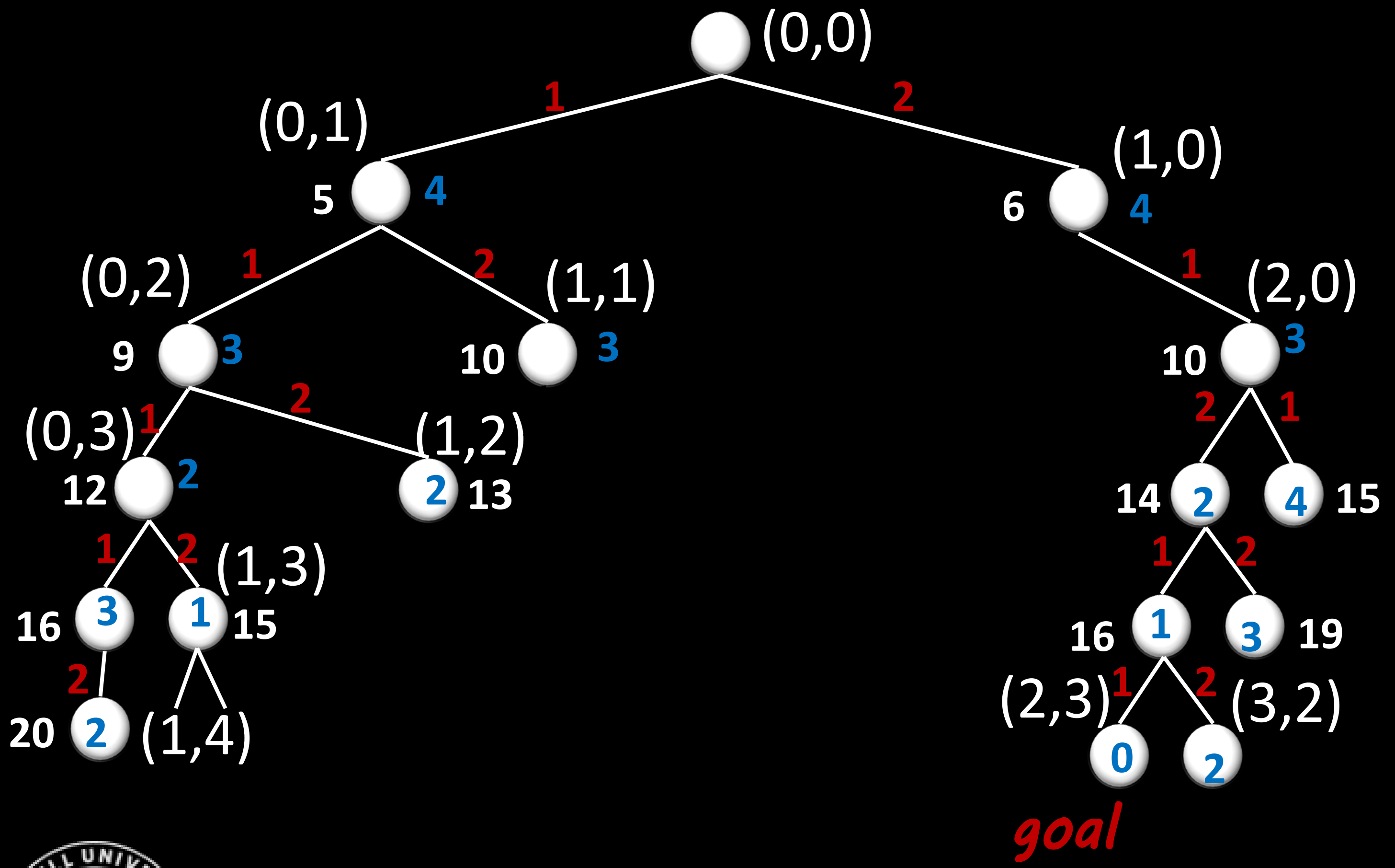
Find a treasure



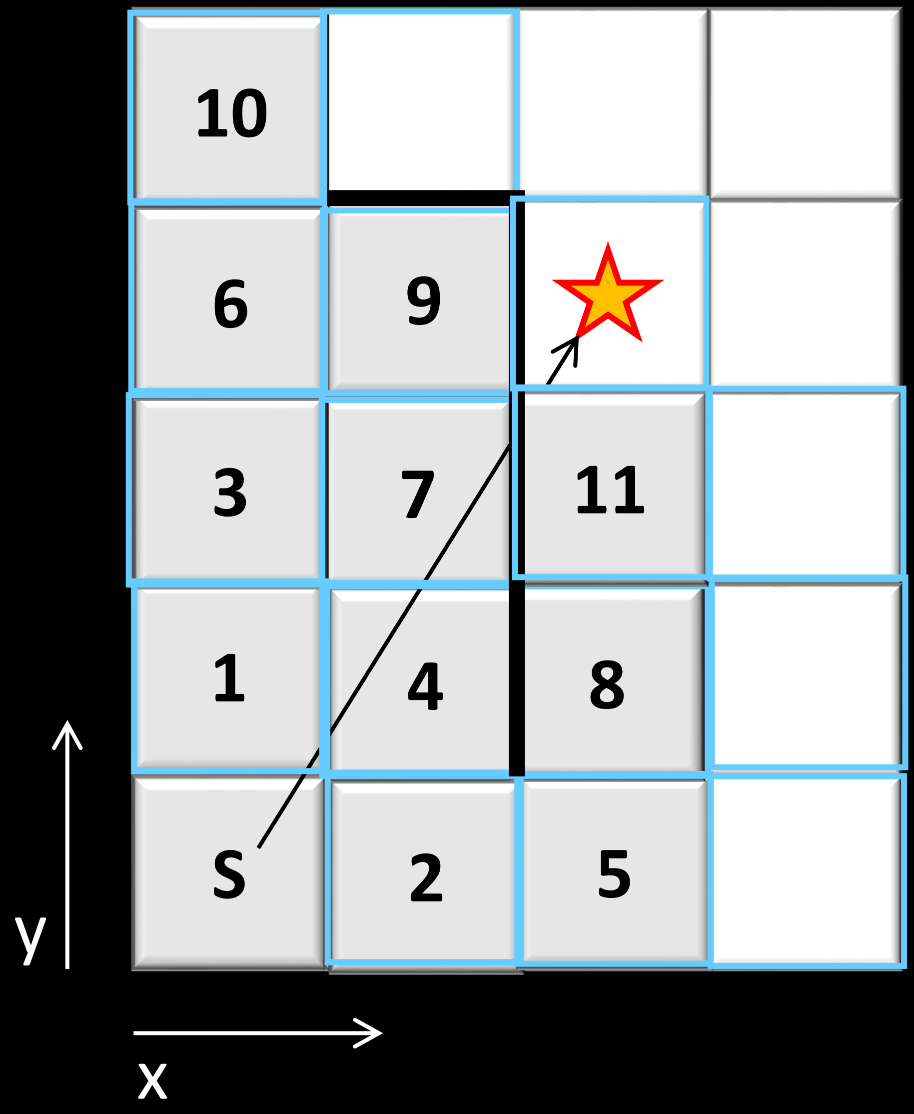
Informed Search

Search order: N, E, S, W

- A* ("A-star")
  - Cost and goal heuristic



Find a goal



A* Search

admissible heuristic

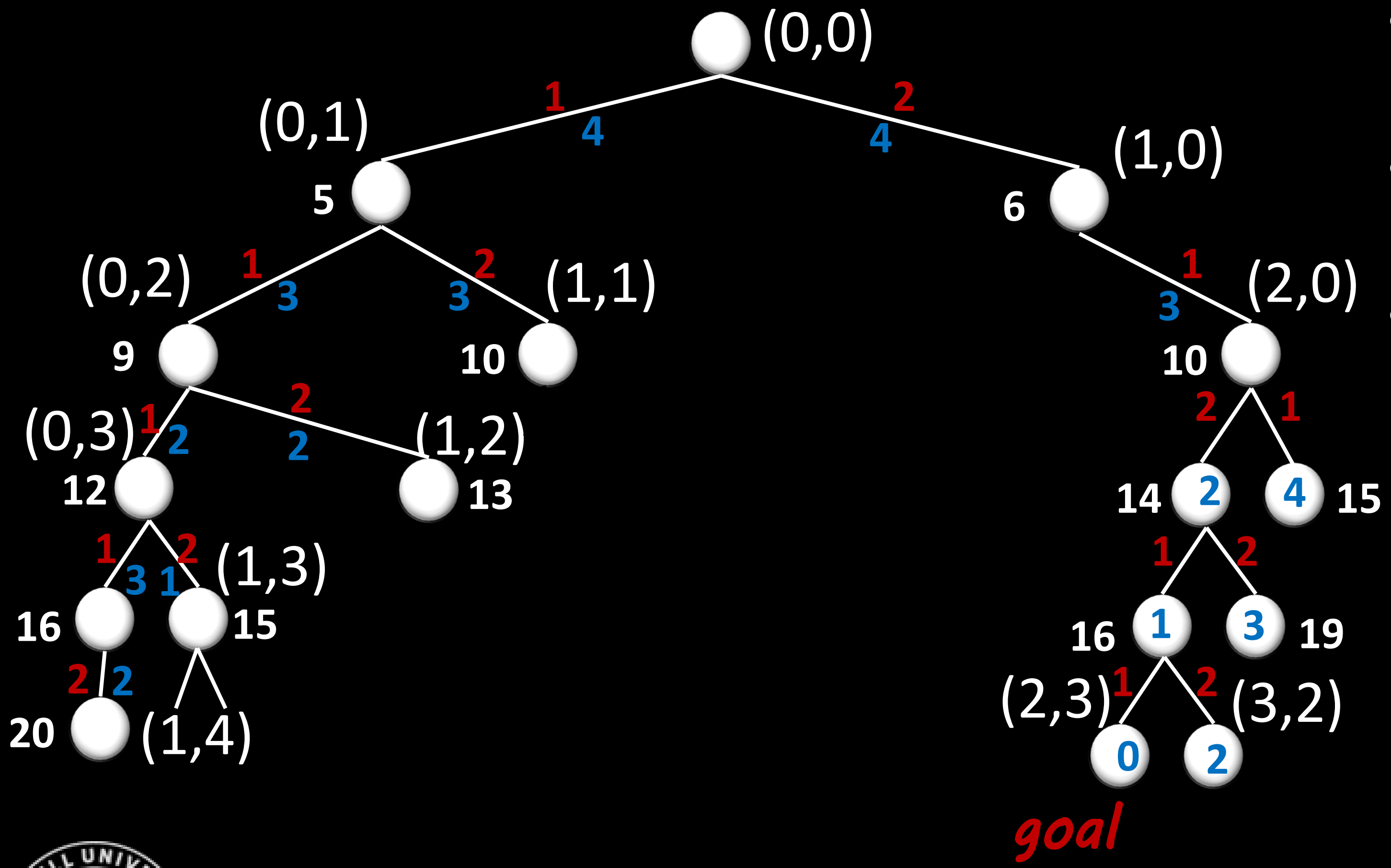
- What if the heuristic is too optimistic?
  - Estimated cost < true cost

inadmissible heuristic

- What if the heuristic is too pessimistic?
  - Estimated cost > true cost
  - No longer guaranteed to be optimal
- What if the heuristic is just right?
  - Pre-compute the cost between all nodes
  - Feasible for you?

Informed Search

- A* ("A-star")
  - Cost and goal heuristic



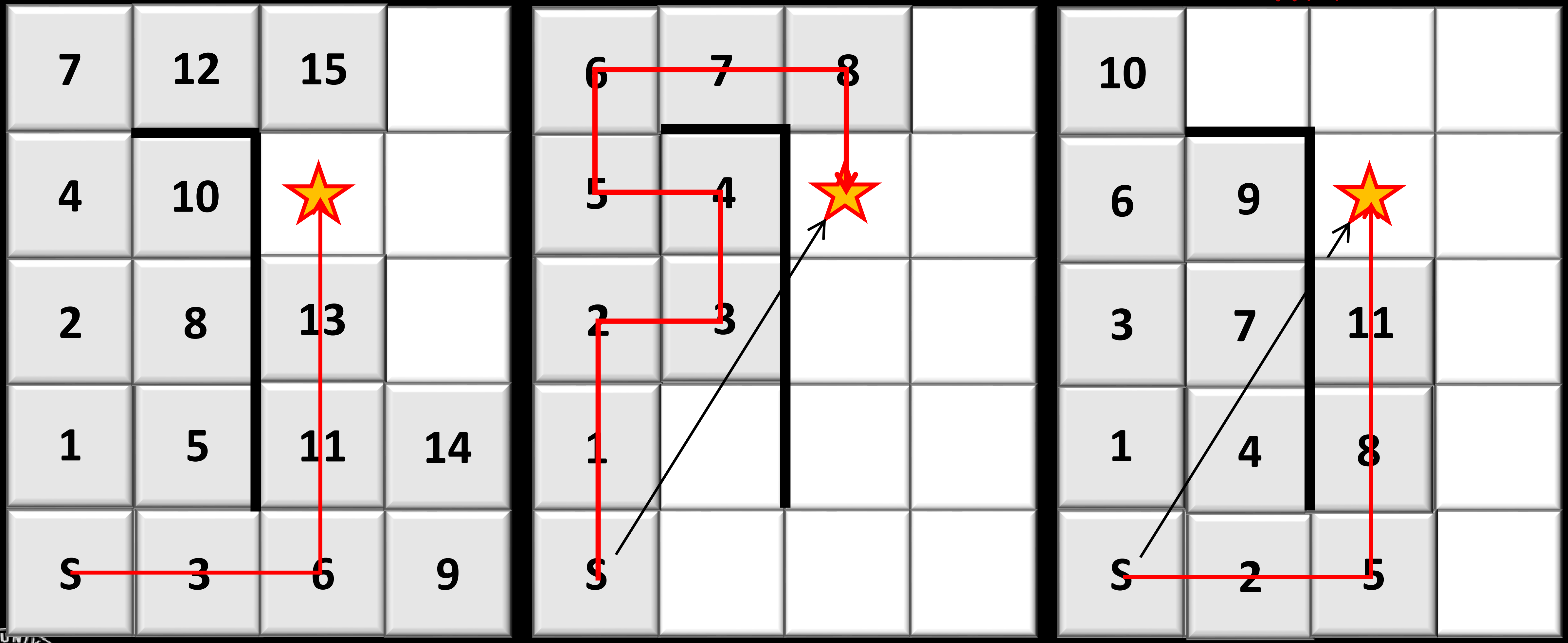
- Complete?
  - Yes!
- Time Complexity
  - $O(b^m)$
- Space Complexity
  - $O(b^m)$
- Optimal?
  - Yes, if the heuristic is admissible!

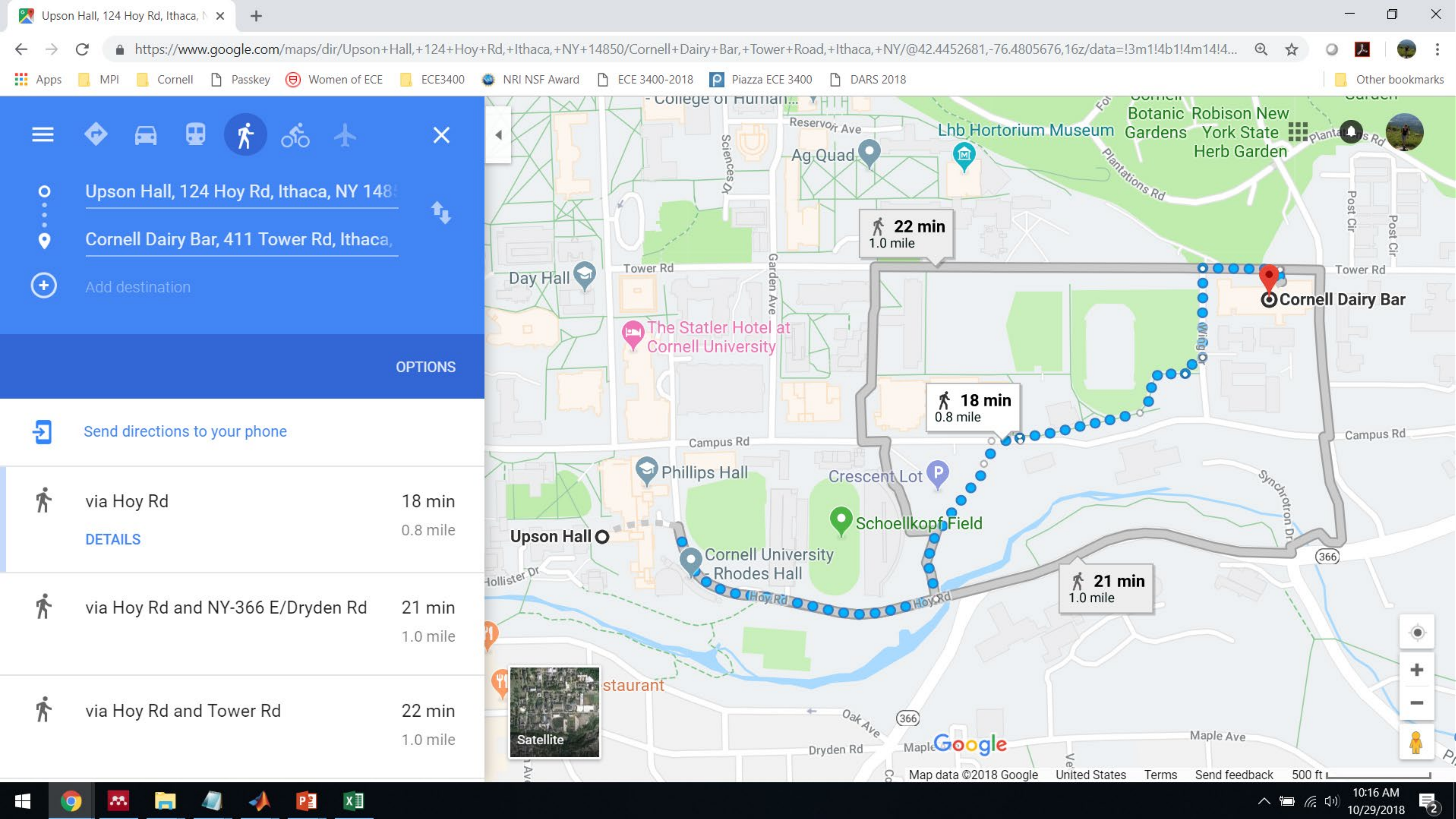
Summary

LCFS *minimum path*

Greedy

A* *minimum path and efficient*





Upson Hall, 124 Hoy Rd, Ithaca, N
https://www.google.com/maps/dir/Upson+Hall,+124+Hoy+Rd,+Ithaca,+NY+14850/Cornell+Dairy+Bar,+Tower+Road,+Ithaca,+NY/@42.4452681,-76.4805676,16z/data=!3m1!4b1!4m14!4...
Apps
MPI
Cornell
Passkey
Women of ECE
ECE3400
NRI NSF Award
ECE 3400-2018
Piazza ECE 3400
DARS 2018
Other bookmarks
Upson Hall, 124 Hoy Rd, Ithaca, NY 148
Cornell Dairy Bar, 411 Tower Rd, Ithaca,
Add destination
OPTIONS
Send directions to your phone
via Hoy Rd
18 min
0.8 mile
DETAILS
via Hoy Rd and NY-366 E/Dryden Rd
21 min
1.0 mile
via Hoy Rd and Tower Rd
22 min
1.0 mile
22 min
1.0 mile
18 min
0.8 mile
21 min
1.0 mile
Upson Hall
Cornell Dairy Bar
Ag Quad
Lhb Hortorium Museum
Botanic Gardens
Robison New York State Herb Garden
Day Hall
The Statler Hotel at Cornell University
Phillips Hall
Crescent Lot
Schoellkopf Field
Cornell University - Rhodes Hall
Tower Rd
Campus Rd
Hoy Rd
Reservoir Ave
Garden Ave
Plantations Rd
Synchrotron Dr
Oak Ave
Dryden Rd
Maple Ave
366
Satellite
Map data ©2018 Google
United States
Terms
Send feedback
500 ft
10:16 AM
10/29/2018