Lecture 6: Introduction to Planning
MAE 345/549

Anirudha Majumdar

Princeton

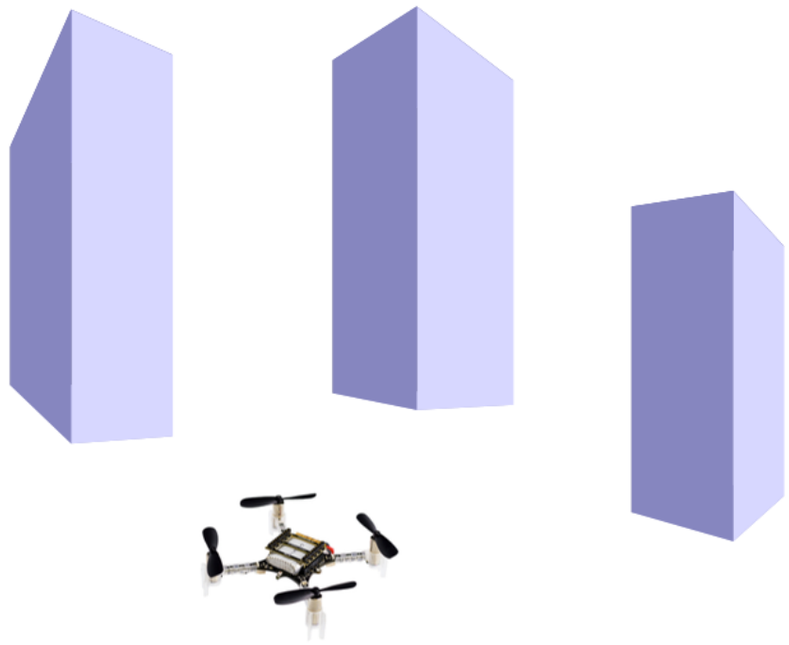
Sept. 22, 2022

PRINCETON
UNIVERSITY

Mechanical and
Aerospace
Engineering

Motion Planning

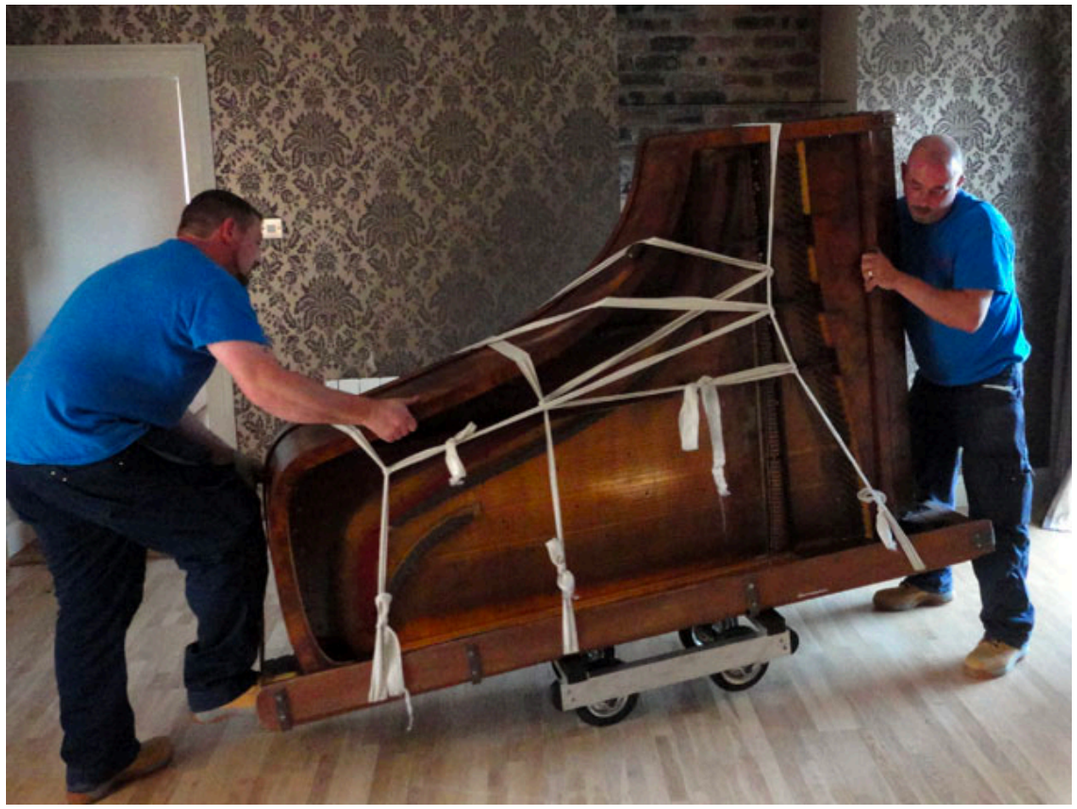
- Recall our overall goal for the course: Make quadrotor navigate through obstacle environments



- Next 5 lectures: focus on particular aspect of this
  - **Motion Planning**

Piano Mover's Problem

- Suppose we have a geometric model (e.g., CAD model) of piano and apartment

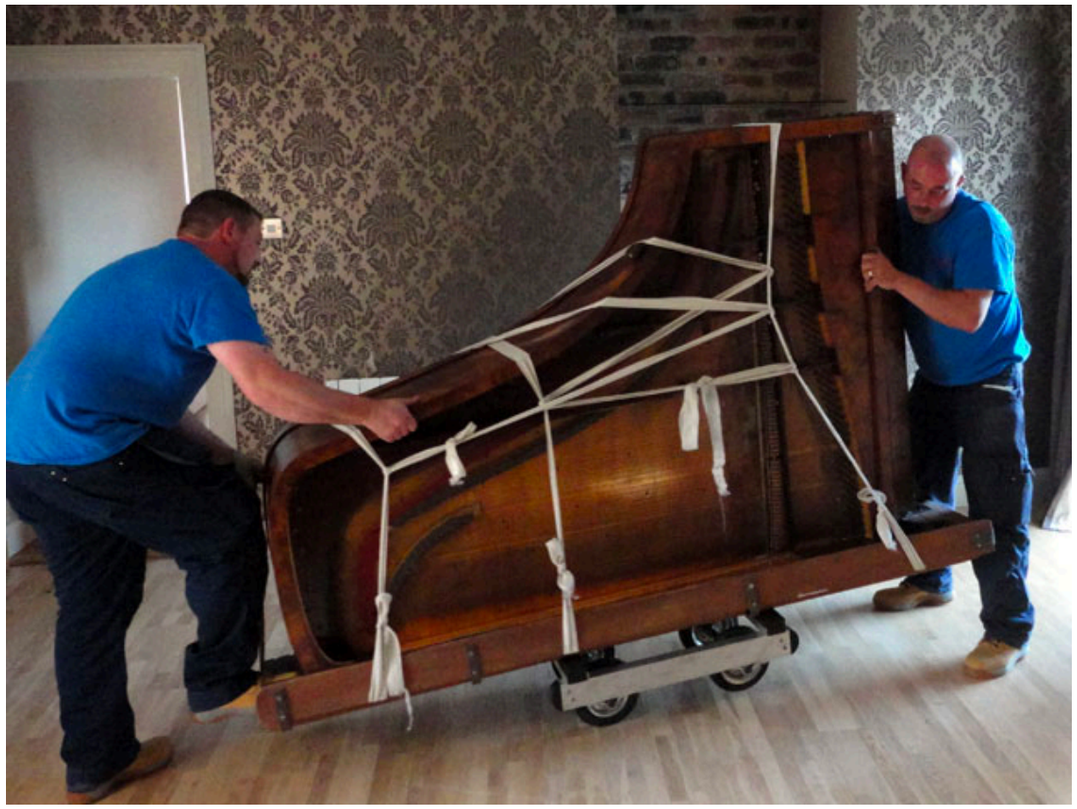


- Question: how can we get piano from point A to point B without hitting anything?

- Complex geometric problem!

Piano Mover's Problem

- Suppose we have a geometric model (e.g., CAD model) of piano and apartment



- Question: how can we get piano from point A to point B without hitting anything?

- Complex geometric problem!

Piano Mover's Problem

https://www.youtube.com/watch?v=cXm3WW-geD8

Piano Mover's Problem

- Piano mover's problem: long history in robotics (since ~1980s)

- Lots of early work on understanding computational complexity of problem (PSPACE-complete)

- Today, the term "piano mover's problem" is synonymous with "motion planning"

Piano Mover's Problem

Motion Planning: problem of finding path from configuration A to configuration B (without collisions)

(Assumes that geometric model of environment is provided. We will think about how to obtain such a model later in course.)

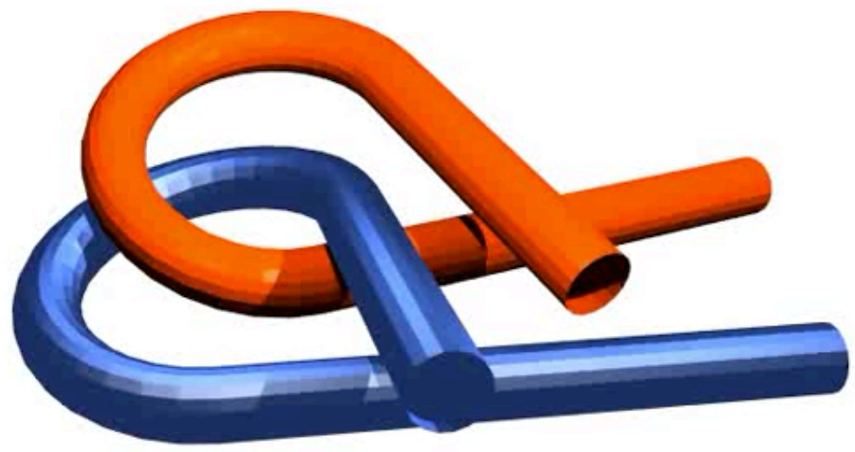
Examples of Planning

- Video from Real-Time robotics: http://rtr.ai/autonomous-vehicles/

Examples of Planning

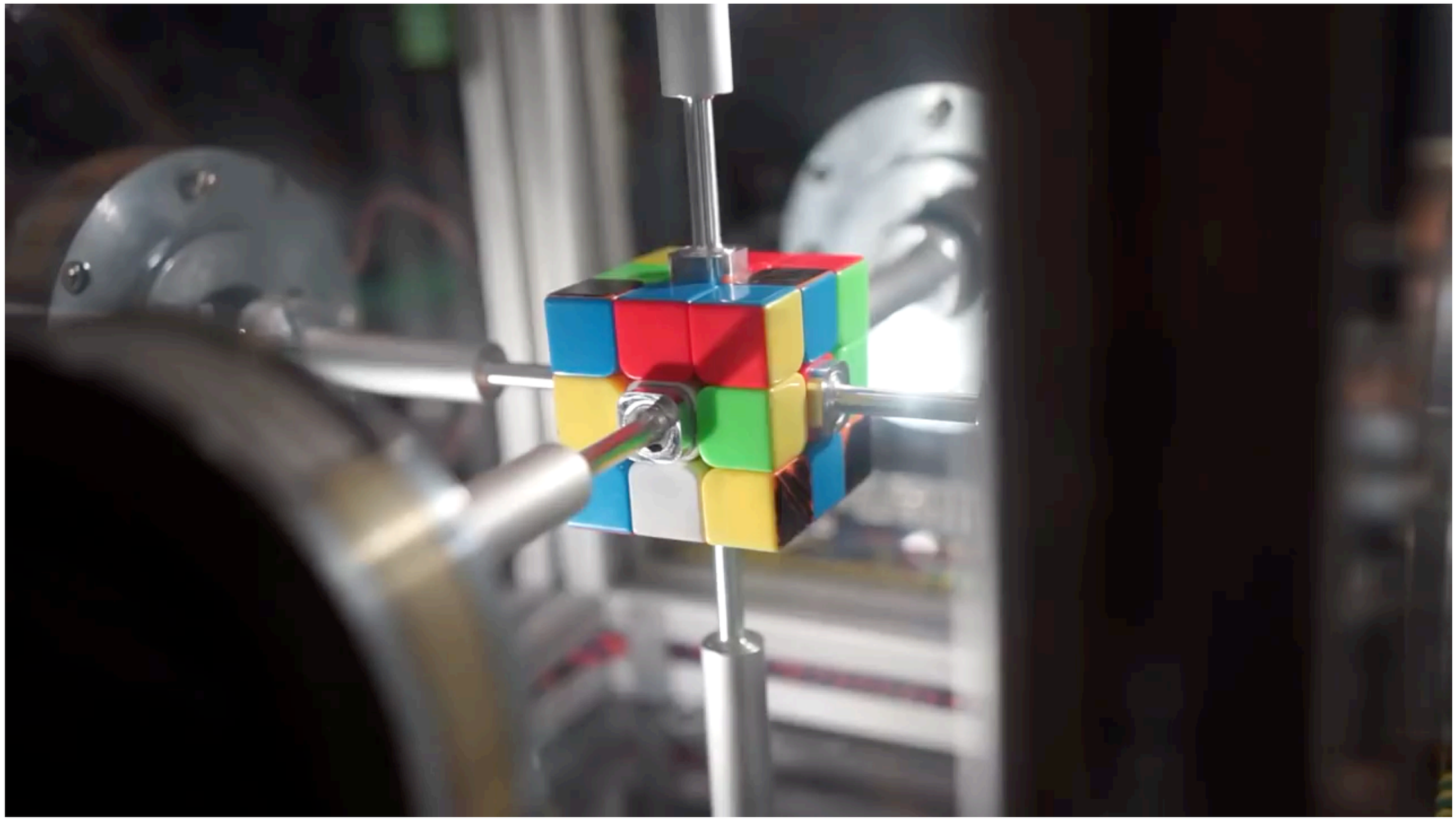
Intelligent and Mobile Robotics Group
http://imr.felk.cvut.cz

Alpha Puzzle 1.0



https://www.youtube.com/watch?v=UTbiAu8IXas

Examples of Planning



https://www.youtube.com/watch?v=OZu9gjQJUQs

Feasibility and Optimality

- As with control, there are two considerations with motion planning:

  - Feasibility: find me *any* path that gets from A to B without collisions

    Challenging by itself!

  - Optimality: find me *best* path (according to some performance criterion, e.g., distance)

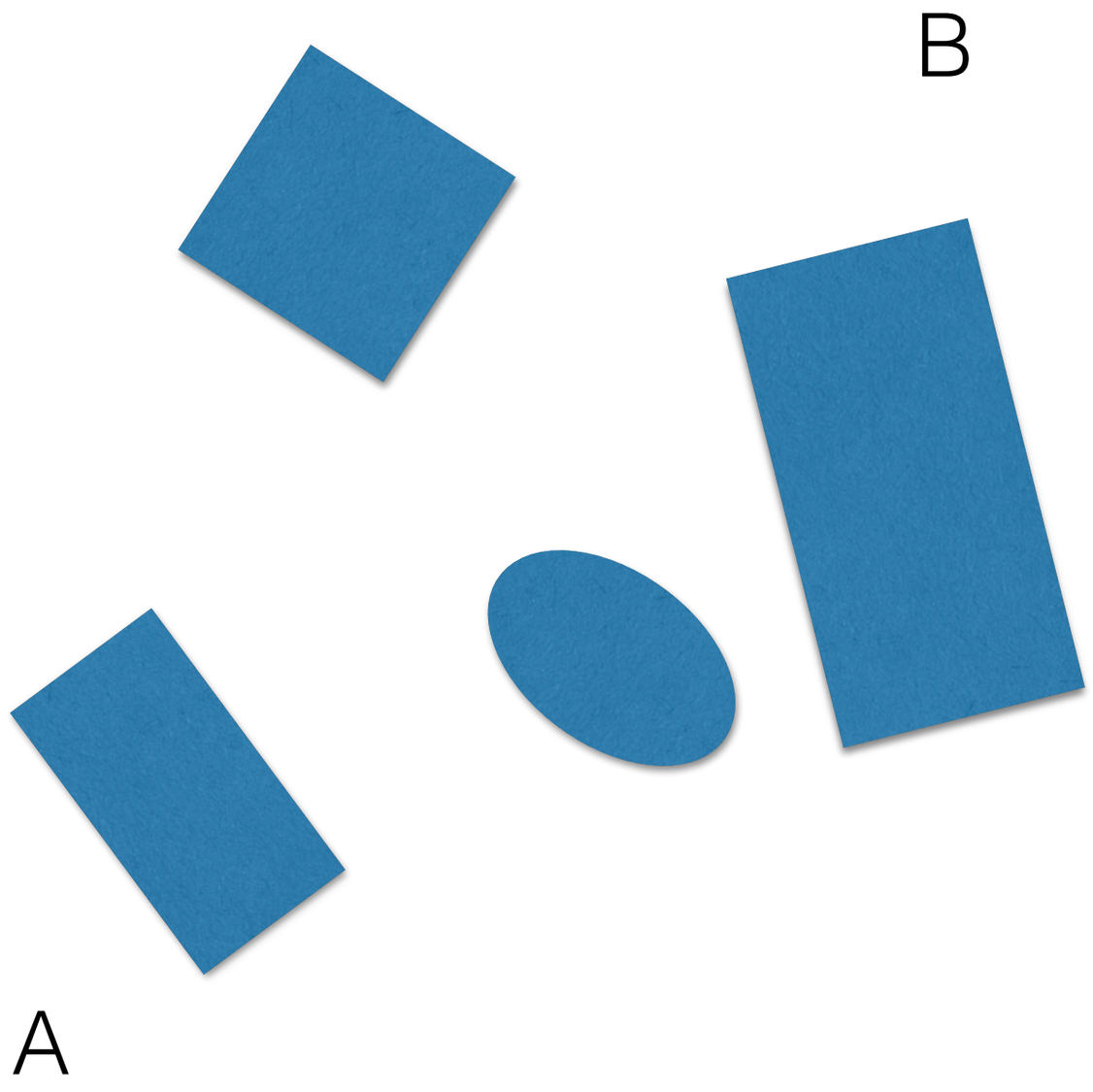
    Nice to have

Assumptions

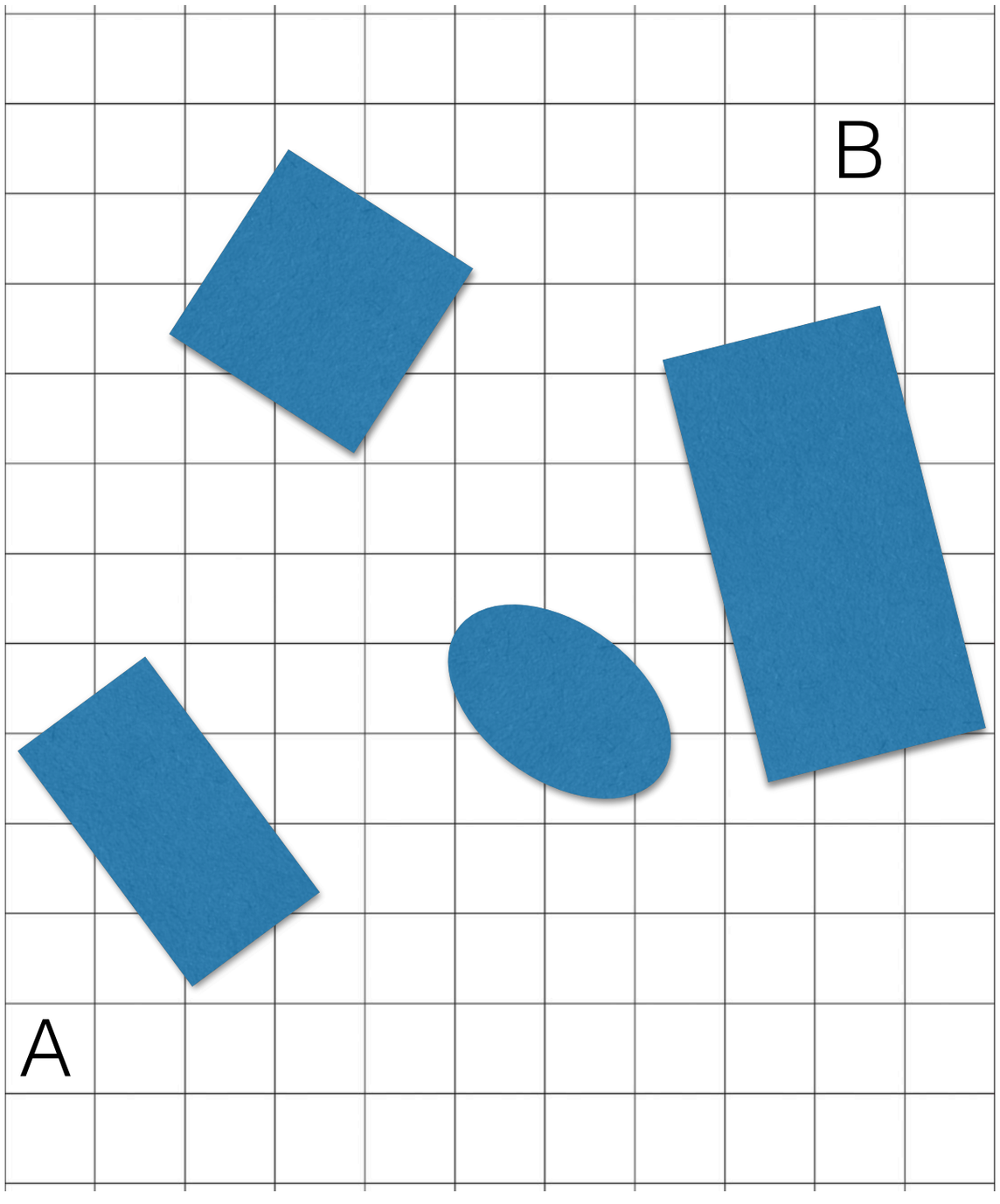
- Assumptions we will make for now:
    - Geometry of environment and robot is given to us
    - Any path can actually be executed by robot (big assumption; not true in general)
    - Any path can be perfectly followed by robot (not true, but feedback control helps with this)

Discrete Planning

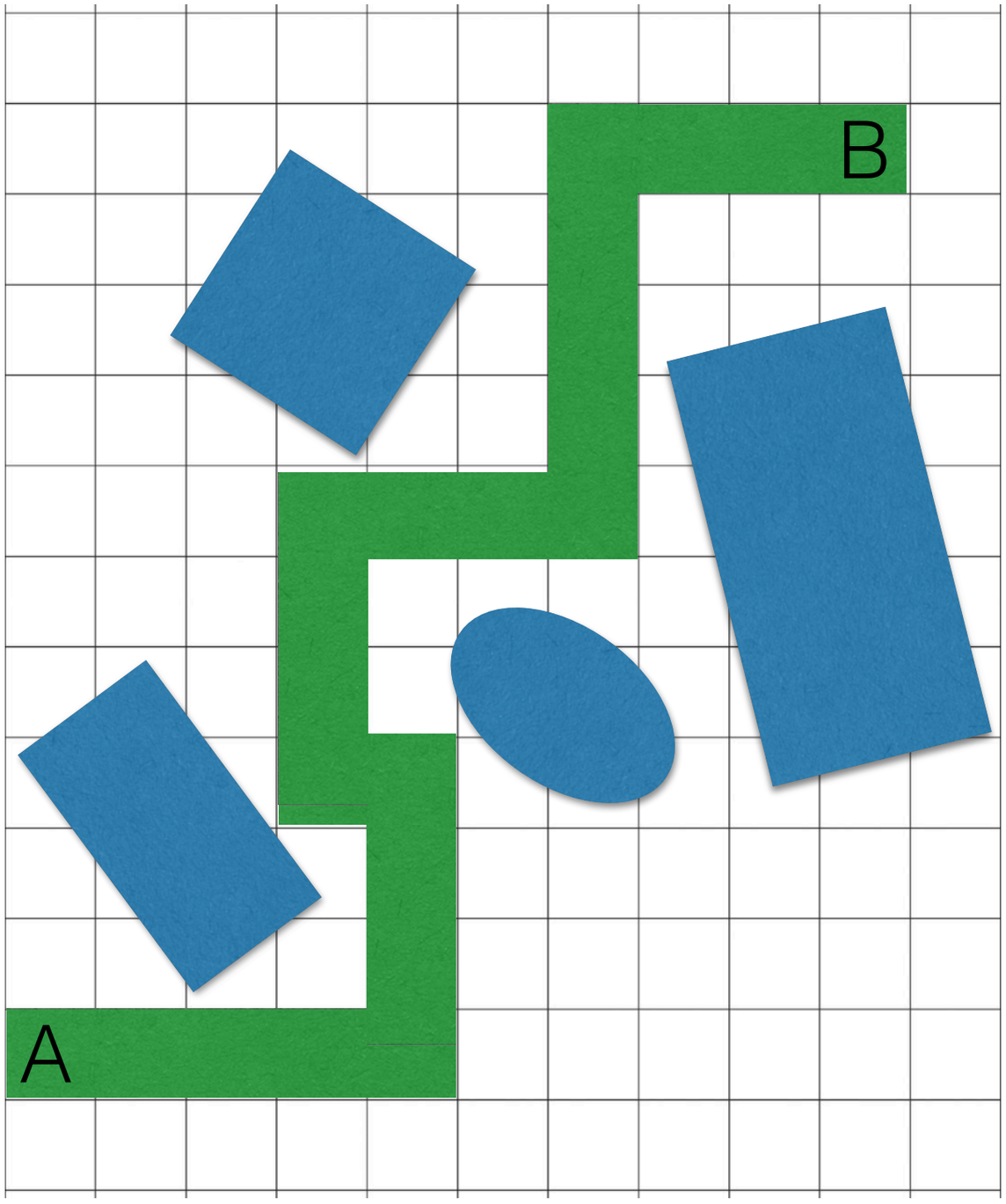
Discretization



Discretization

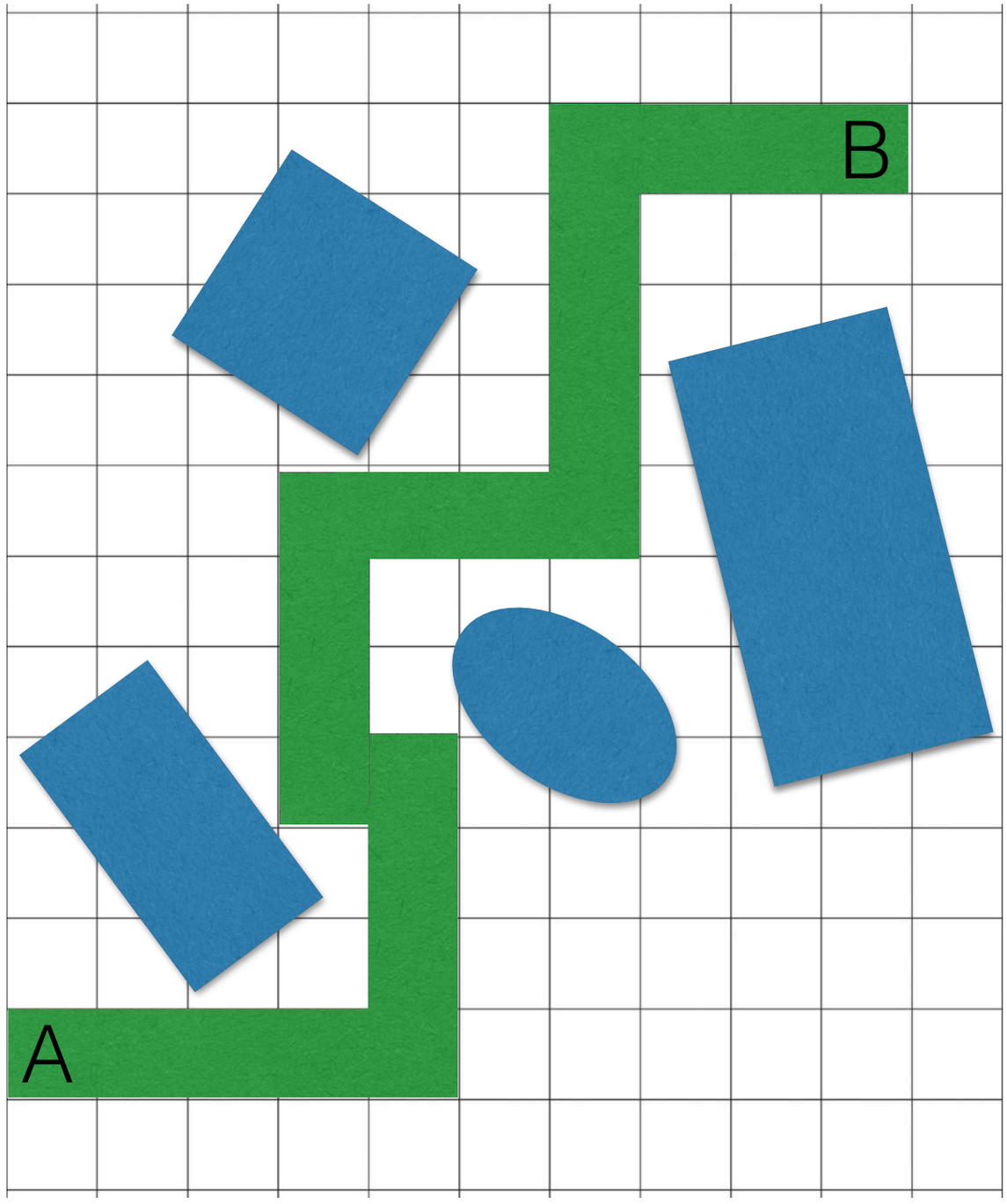


Discretization



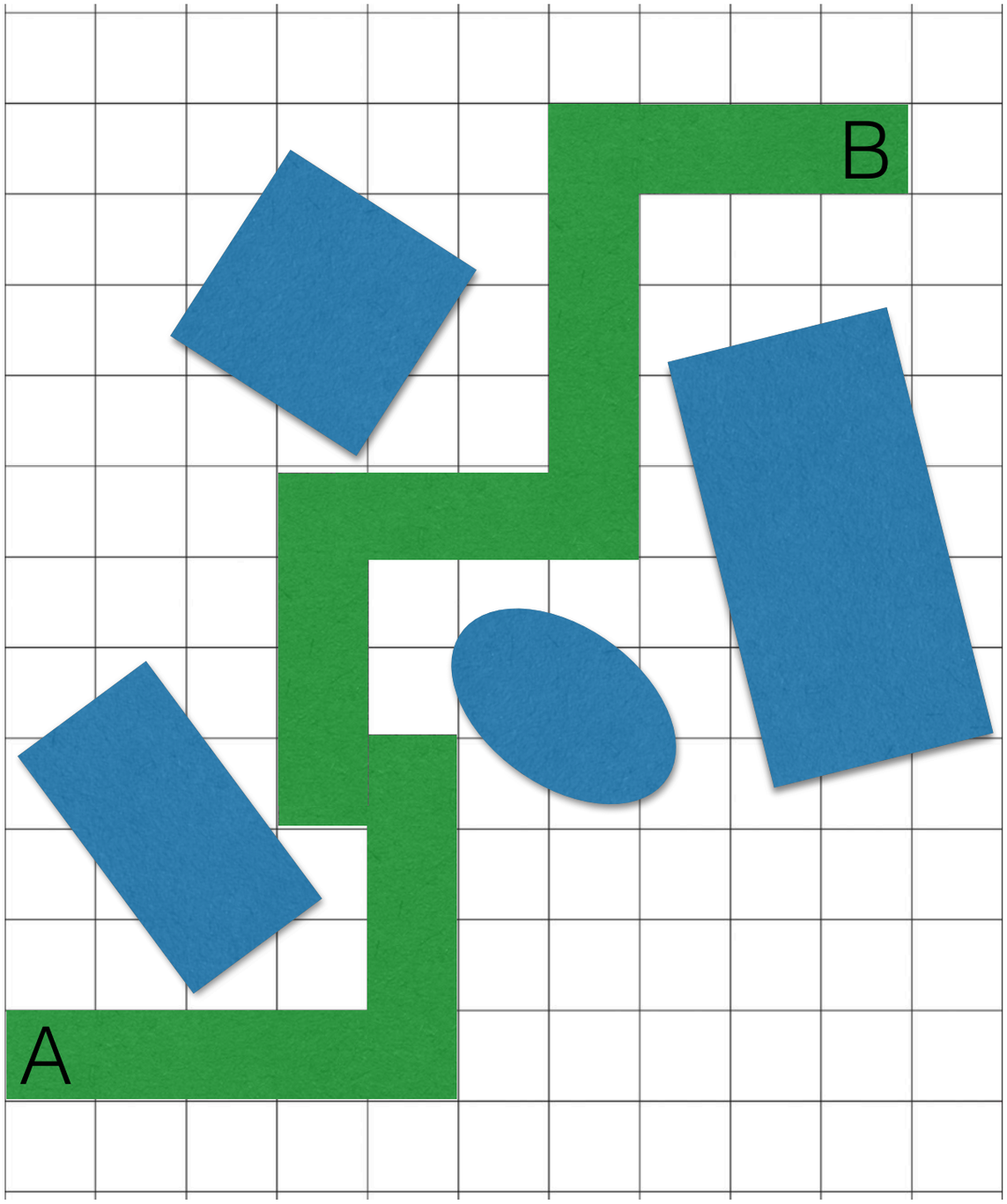
Discretization

- Discretization allows us to apply powerful algorithms for graph search to solve planning problem



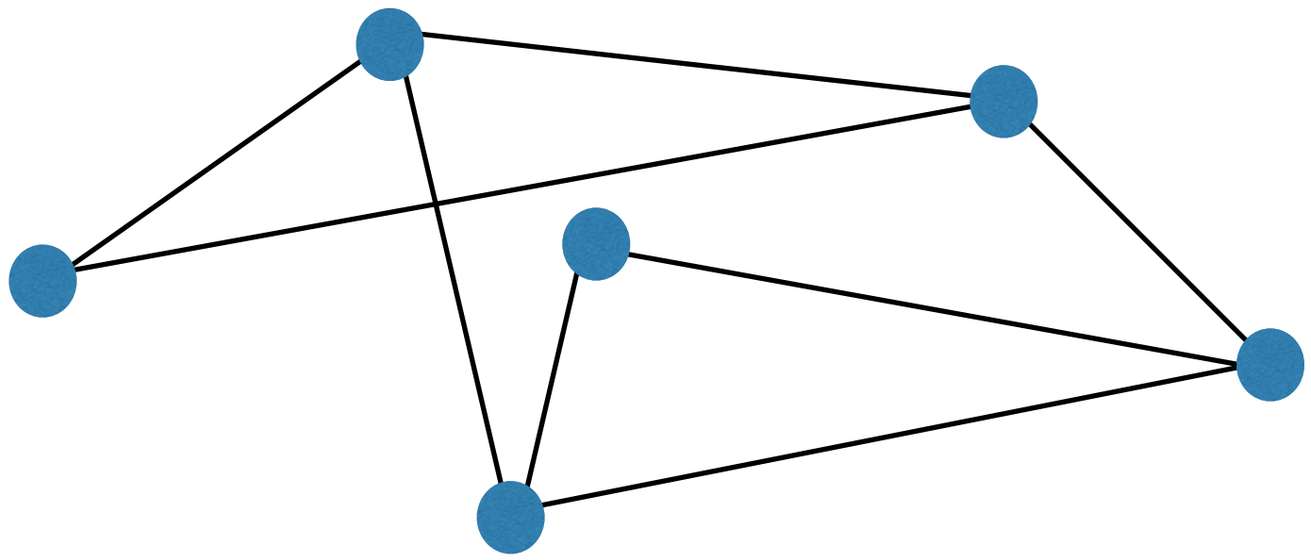
Discretization

- Choices we made:
- Uniform grid
- 4-connected (e.g., instead of allowing diagonal motion, i.e., 8-connected)
- Resolution of grid (finer grid is better approximation, but makes planning harder)



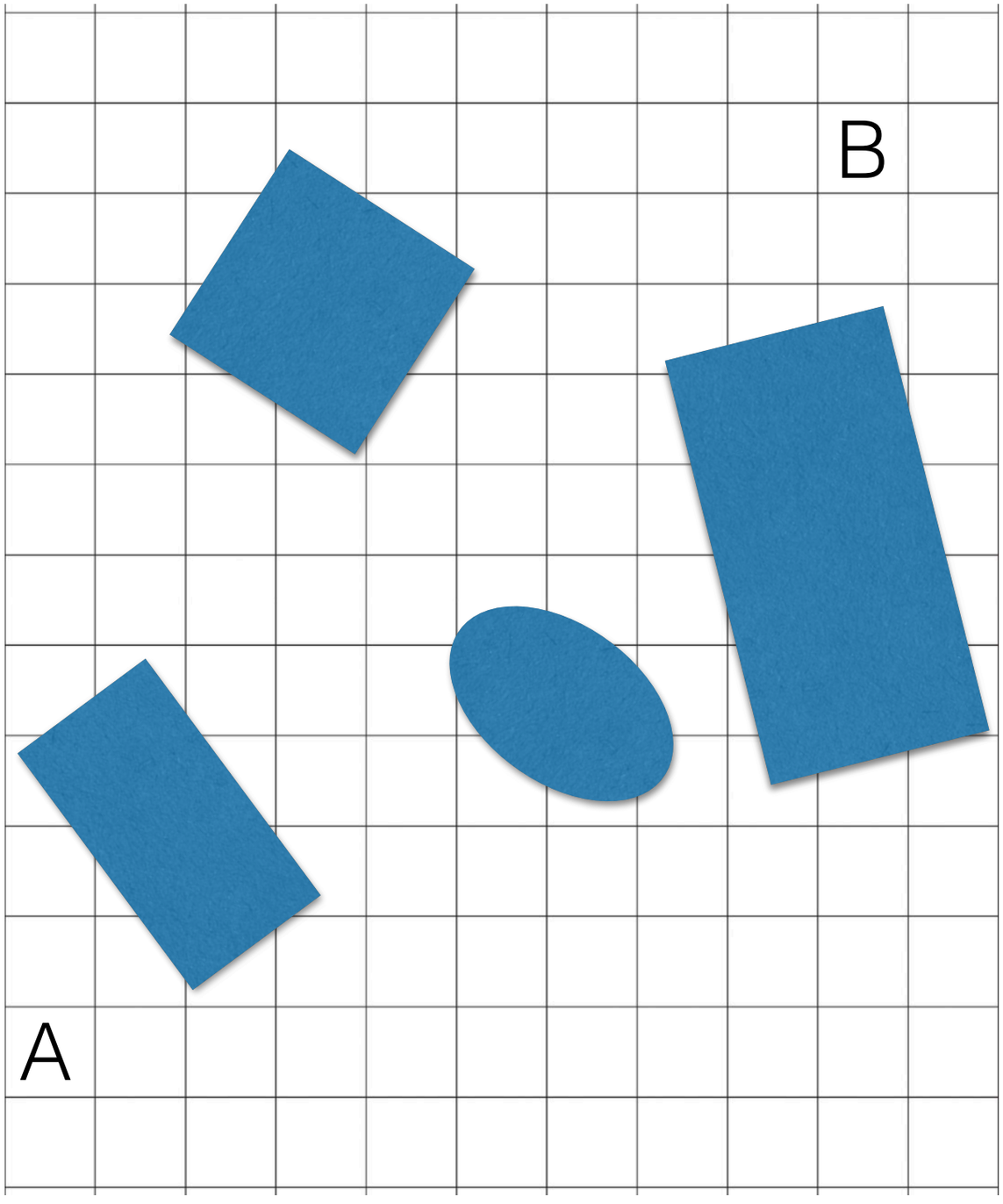
Graph Search

- Graph: collection of vertices (nodes) and edges
  - Edges define connectivity



Planning as Graph Search

- Associate a vertex with each cell in grid
- Connect vertices with edges based on 4-connectivity (or 8-connectivity)
- Delete any vertices (and corresponding edges) that have obstacle



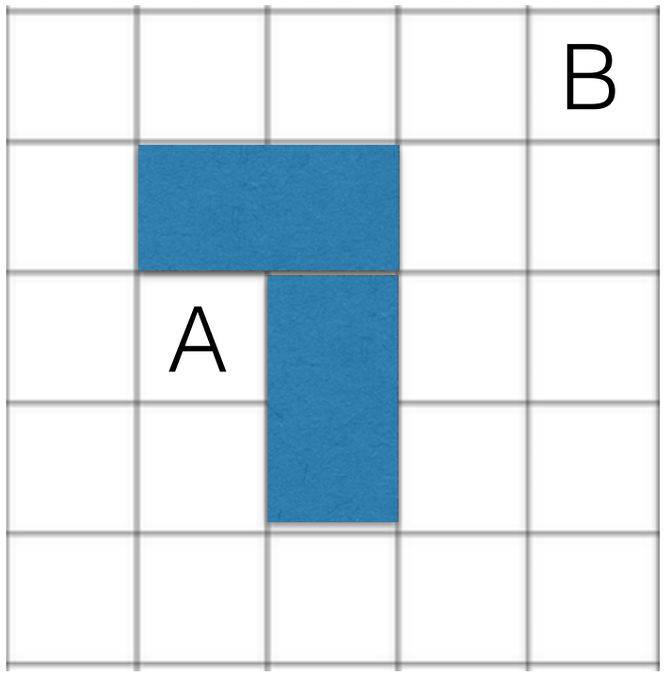
Planning as graph search: Example

Planning as graph search: Example

References: "Planning Algorithms" by S. LaValle

- Ch. 1.1, 1.2, 1.3
- Ch 2.1, 2.2

Example

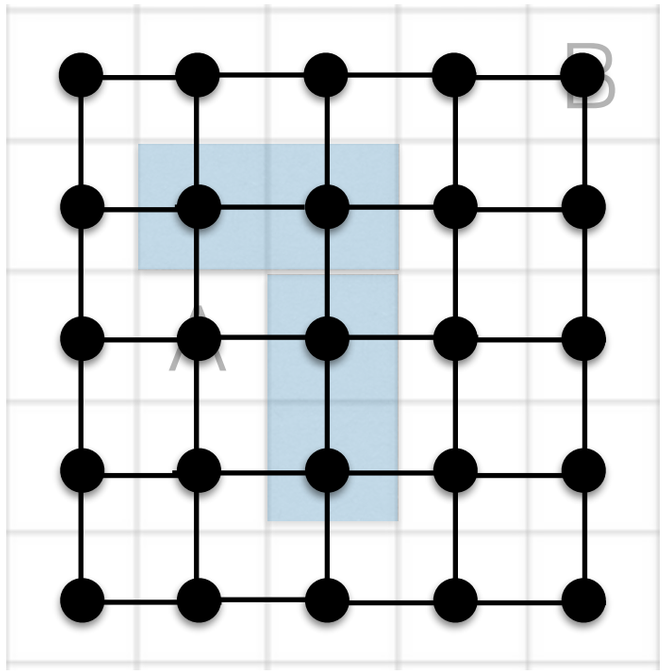


1. Associate a vertex with each cell

2. Edges connect vertices that we can travel between in one step

3. Remove vertices (and associated edges) that are occupied by obstacles

Example

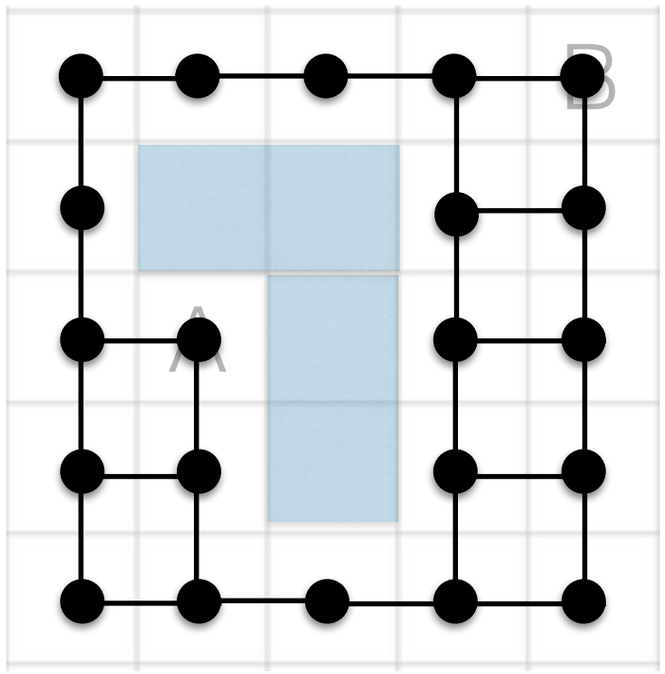


1. Associate a vertex with each cell

2. Edges connect vertices that we can travel between in one step

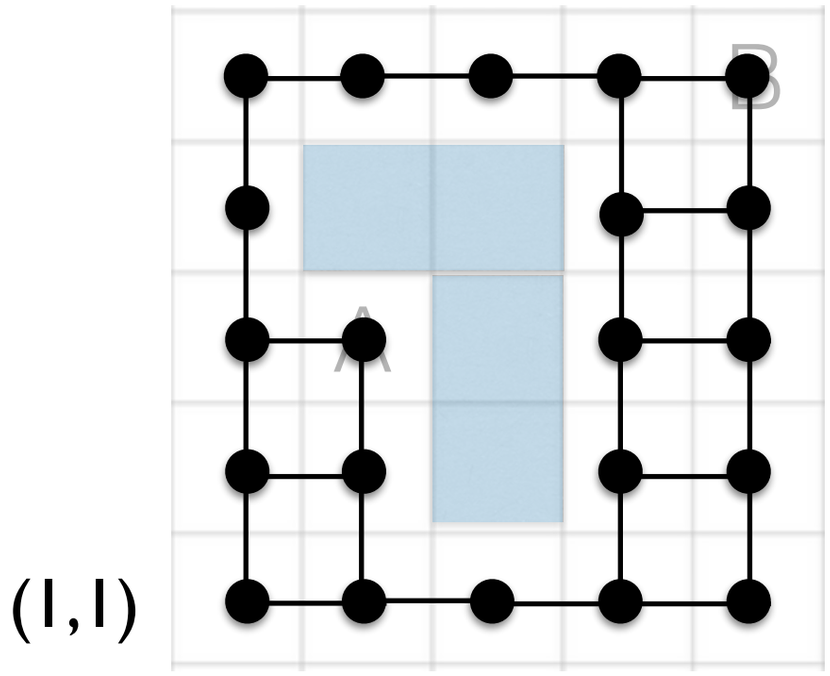
3. Remove vertices (and associated edges) that are occupied by obstacles

Example



1. Associate a vertex with each cell
2. Edges connect vertices that we can travel between in one step
3. Remove vertices (and associated edges) that are occupied by obstacles

Example



We will label vertices: (i,j) = (x, y) = (column, row)

Examples: A : (2,3)

B : (5,5)

Graph Search

- We will start at A, and incrementally explore the graph until we have arrived at B

- Today: focus only on finding feasible plans

  - We will discuss optimal plans in the next lecture

Algorithm: Forward Search

Algorithm 1: Forward Search

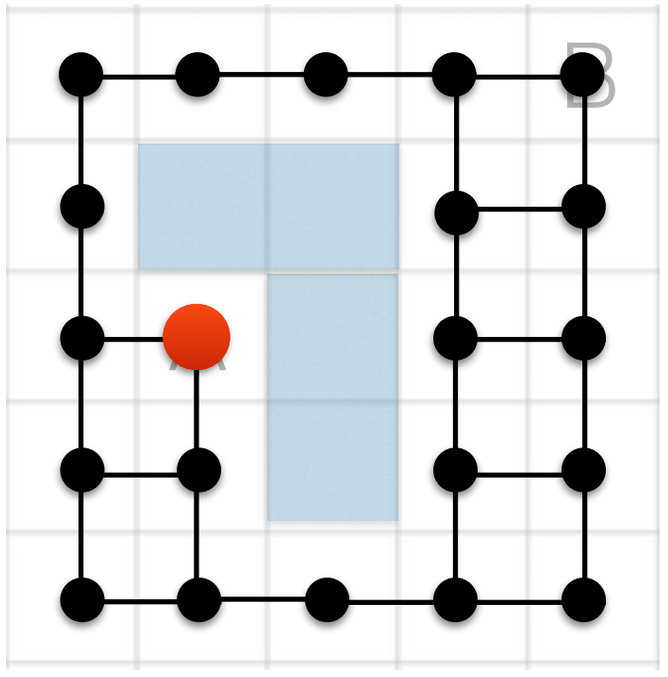
```
Q.Insert(A) and mark A as visited;
while Q not empty do
    x ← Q.GetVertex() ;
    if x ∈ x_G then
        // x_G is goal set, e.g., {B} ;
        Return SUCCESS;
    end
    for all neighbors x' of x do
        if x' not visited then
            Parent(x') ← x ;
            Mark x' as visited ;
            Q.Insert(x') ;
        end
    end
end
Return FAILURE ;
```

Example

Q = {(2,3)} = {A}

Mark A as visited

(visited vertices will be marked in red)

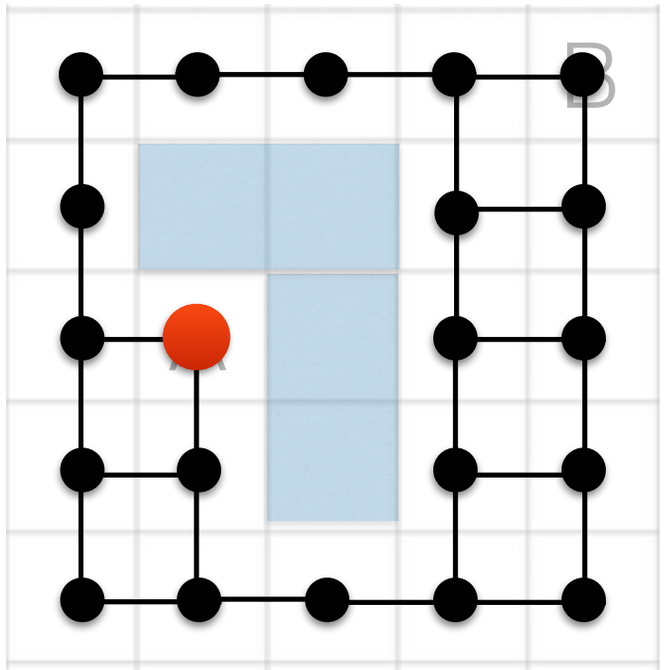


Example

Iteration 1

Q = {(2,3)} = {A}

$x \leftarrow Q.\mathrm{GetVertex}()$



Example

Iteration 1

Q = {(2,3)} = {A}

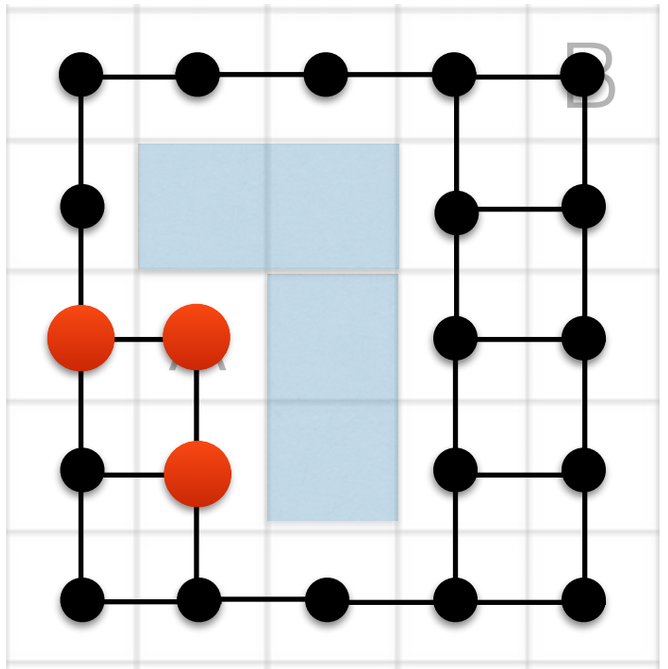
$x \leftarrow Q.\mathrm{GetVertex}()$

x = (2,3)

Neighbors of x: {(1,3), (2,2)}

Mark these as visited and add to Q

Q = {(1,3), (2,2)}

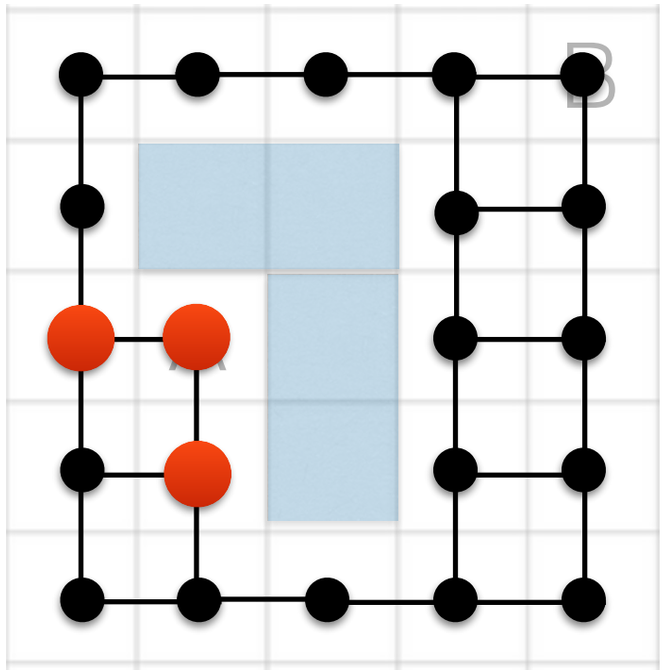


Example

Iteration 2

Q = {(1,3), (2,2)}

$x \leftarrow Q.\mathrm{GetVertex}()$



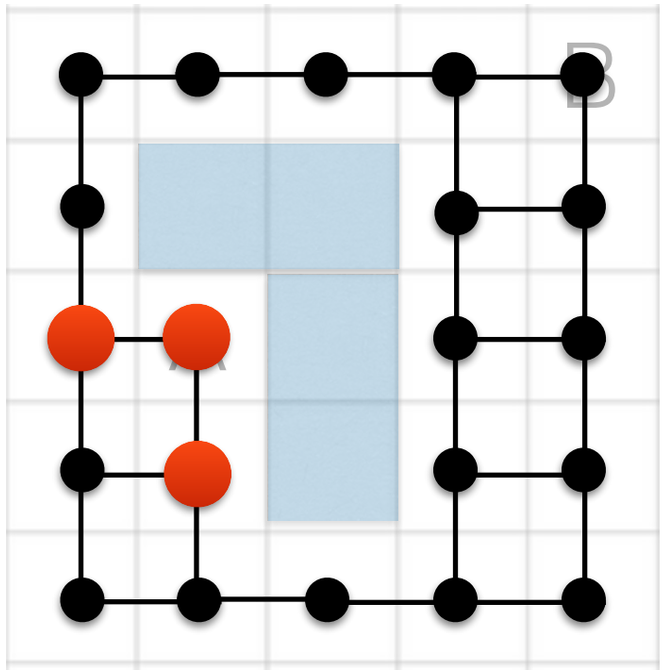
Example

Iteration 2

Q = {(1,3), (2,2)}

$x \leftarrow Q.\mathrm{GetVertex}()$

x = (1,3) [Here, we implement "First-in-first-out"]



Example

Iteration 2

Q = {(1,3), (2,2)}

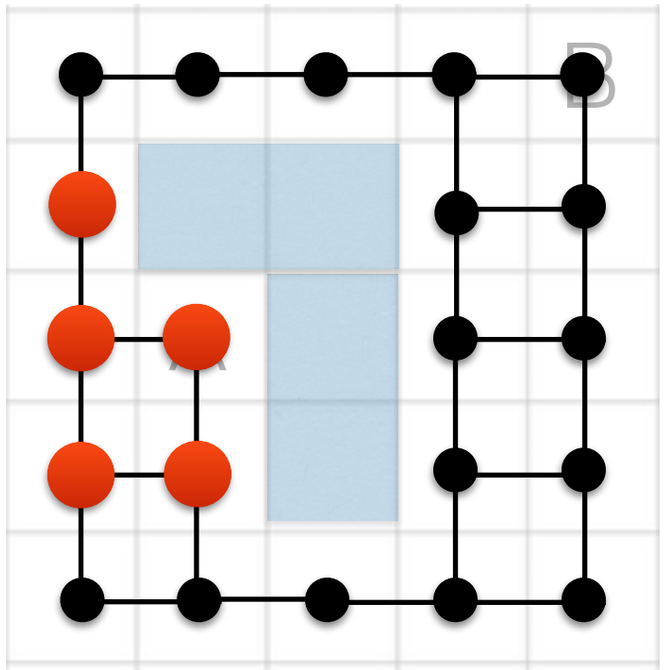
$x \leftarrow Q.\mathrm{GetVertex}()$

x = (1,3) [Here, we implement "First-in-first-out"]

Neighbors of x: {(1,4), (1,2), ~~(2,3)~~}

Mark unvisited as visited and add to Q

Q = {(2,2), (1,4), (1,2)}



Example

Iteration 3

Q = {(2,2), (1,4), (1,2)}

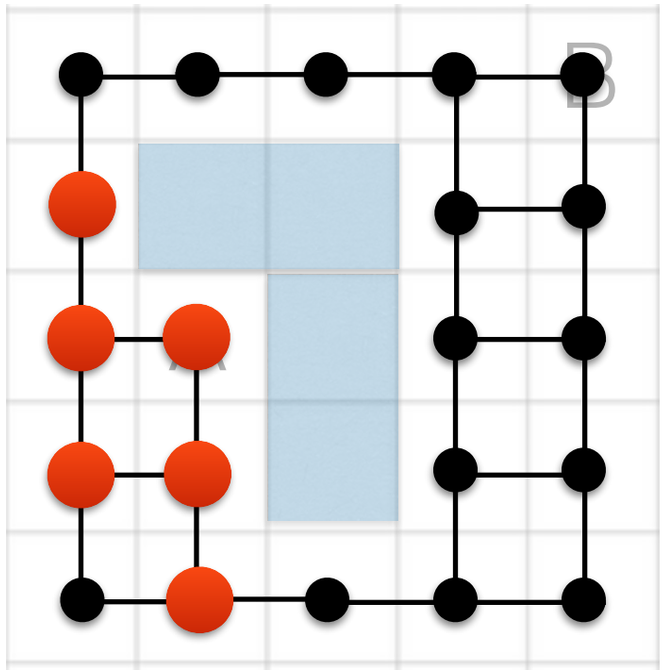
$x \leftarrow Q.\mathrm{GetVertex}()$

x = (2,2) ["First-in-first-out"]

Neighbors of x: {~~(1,2)~~, (2,1), ~~(2,3)~~}

Mark unvisited as visited and add to Q

Q = {(1,4), (1,2), (2,1)}



(5,5)

(1,1)

Example

Iteration 4

Q = {(1,4), (1,2), (2,1)}

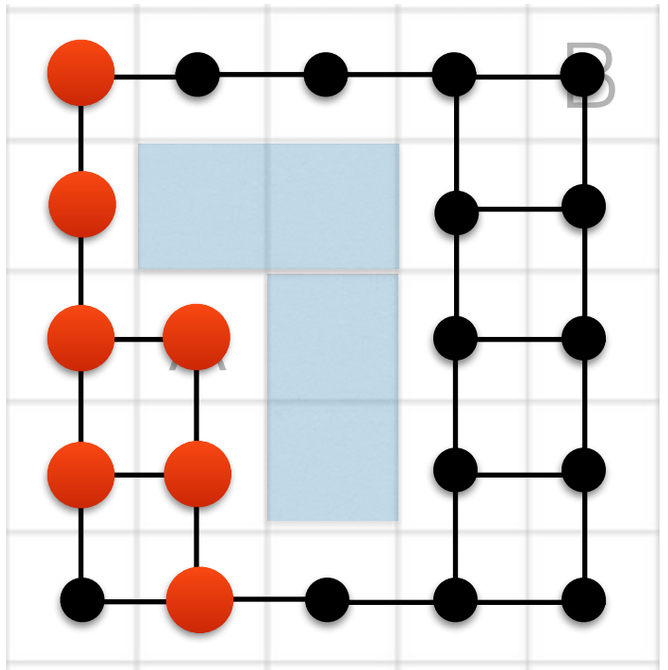
$x \leftarrow Q.\mathrm{GetVertex}()$

x = (1,4) ["First-in-first-out"]

Neighbors of x: {(1,5), ~~(1,3)~~}

Mark unvisited as visited and add to Q

Q = {(1,2), (2,1), (1,5)}



(5,5)

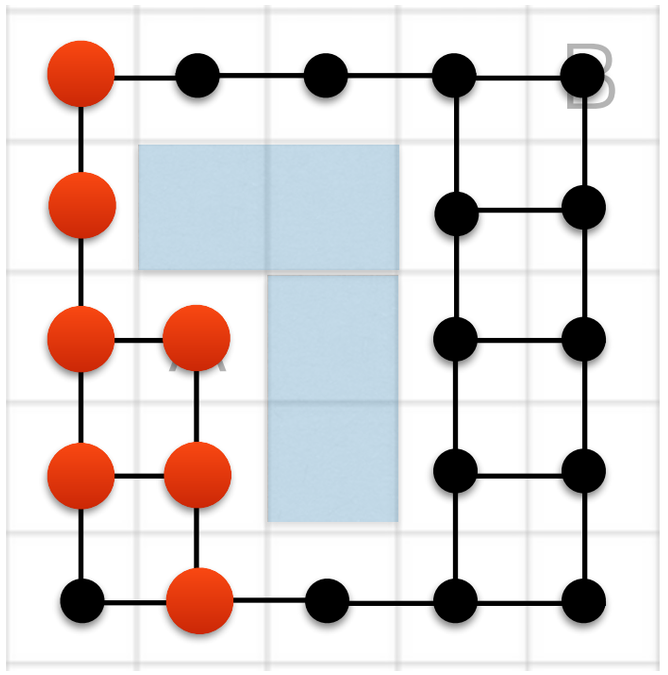
(1,1)

Example

Iteration 5

… and so on …

(until x hits goal vertex B)

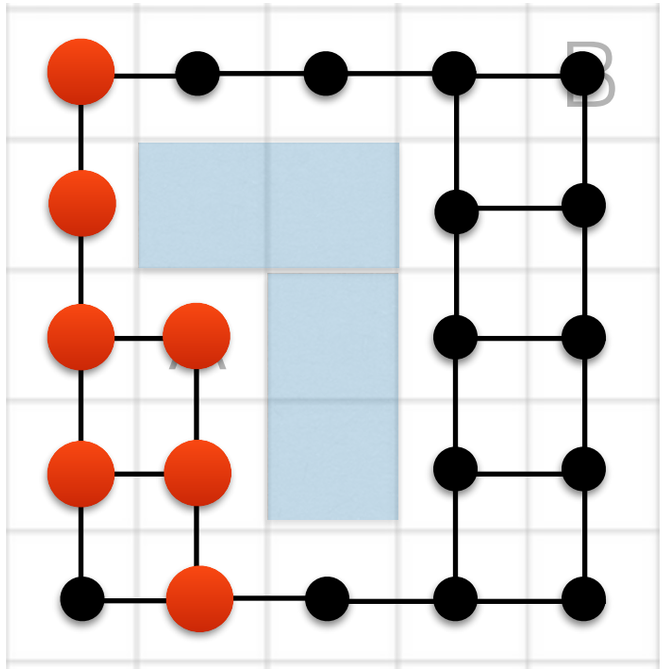


(5,5)

(1,1)

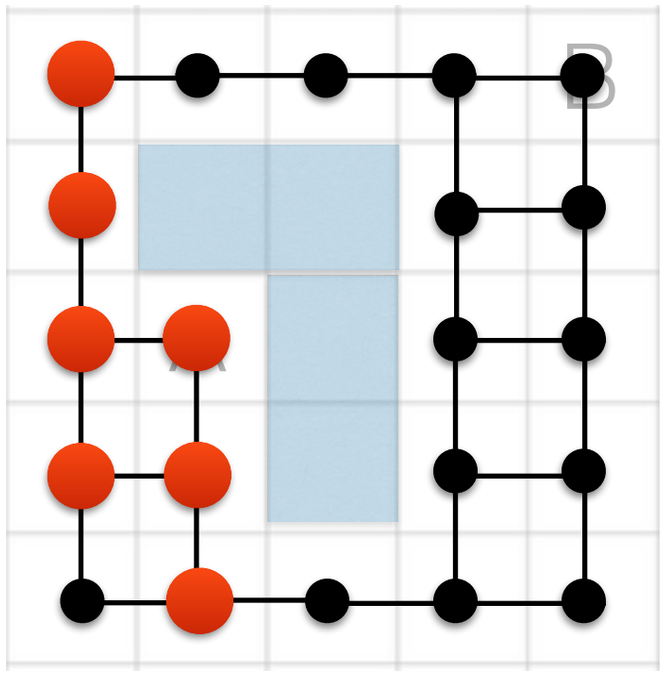
Example

- Note:
  - We can back out path from A to B by storing the "parent" of x' (i.e., the vertex of which x' was a neighbor). We can start by looking at the parent of the goal, then looking at the parent of the parent, and so on… until we hit A.



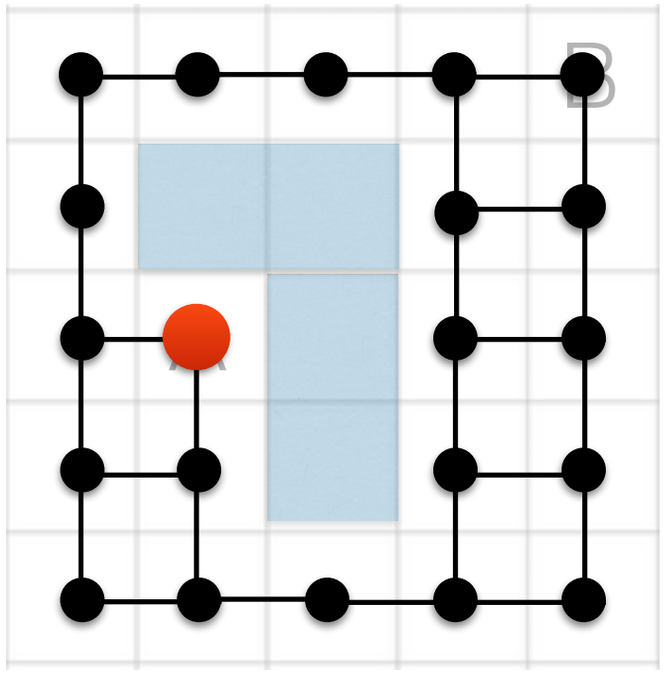
Example

- Note:
  - We implemented Q.GetVertex() using "first-in-first-out (FIFO)"
  - This corresponds to Breadth First Search (BFS)
  - BFS searches all paths of length k before searching paths of length k+1



Depth First Search (DFS)

Let's explore another way to implement Q.GetVertex():"Last-in-first-out" (LIFO)



Depth First Search (DFS)

Iteration 1

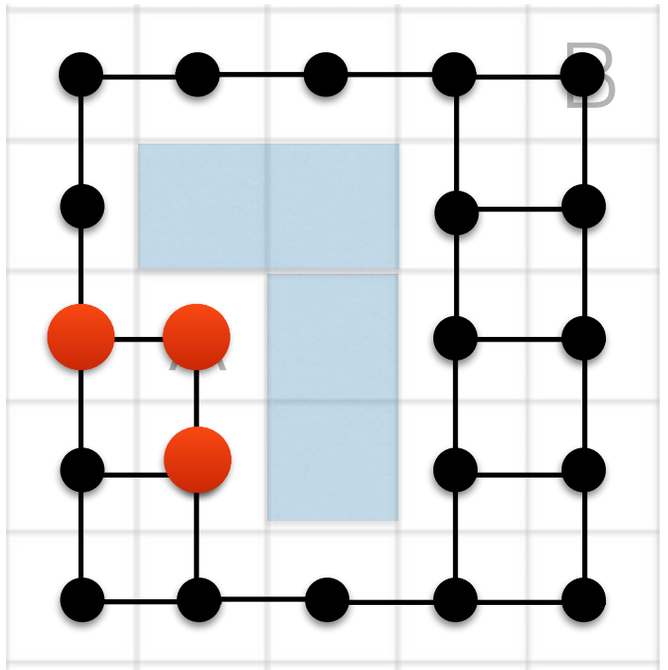
Q = {(2,3)} = {A}

$x \leftarrow Q.GetVertex()$

x = (2,3)

Neighbors: {(2,2), (1,3)}

Q = {(2,2), (1,3)}



Depth First Search (DFS)

Iteration 2

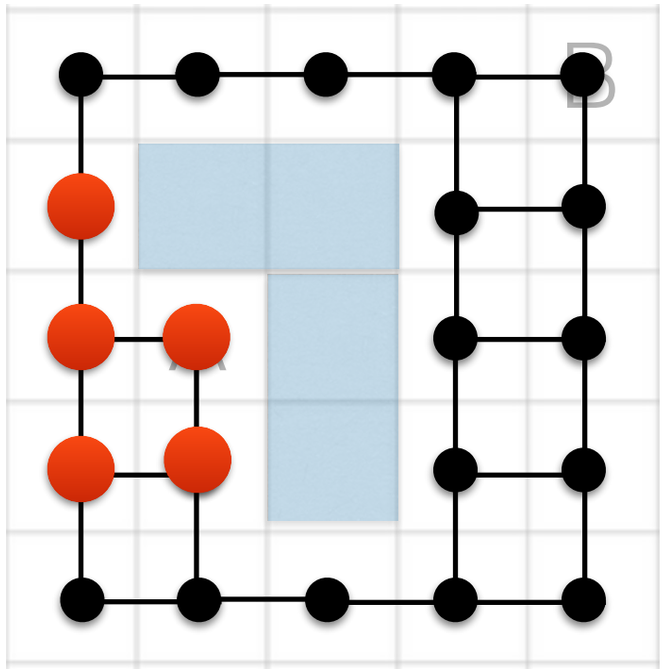
Q = {(2,2), (1,3)}

$x \leftarrow Q.\mathrm{GetVertex}()$

x = (1,3)

Neighbors: {~~(2,3)~~, (1,2), (1,4)}

Q = {(2,2), (1,2), (1,4)}



Depth First Search (DFS)

Iteration 3

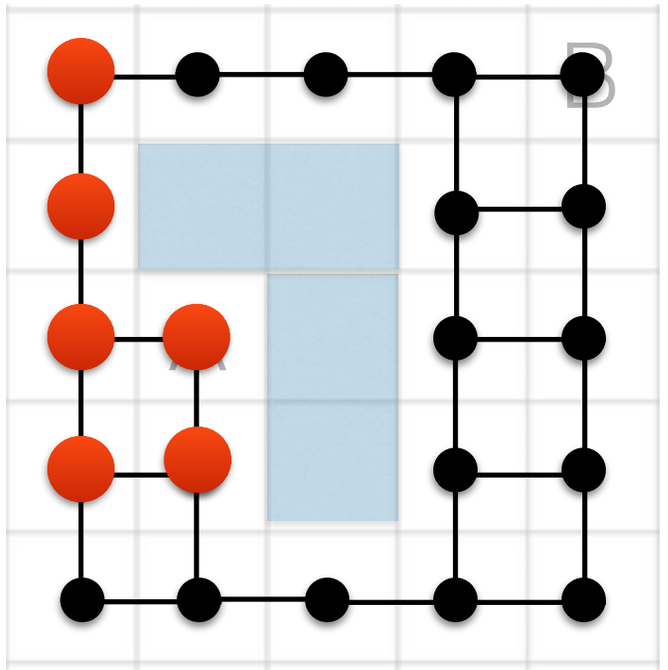
Q = {(2,2), (1,2), (1,4)}

$x \leftarrow Q.\mathrm{GetVertex}()$

x = (1,4)

Neighbors: {~~(1,3)~~, (1,5)}

Q = {(2,2), (1,2), (1,5)}



Depth First Search (DFS)

Iteration 4

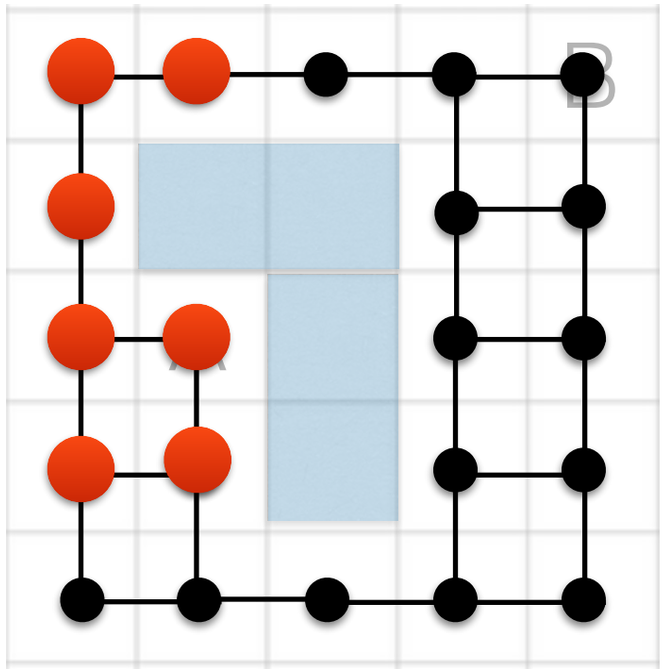
Q = {(2,2), (1,2), (1,5)}

$x \leftarrow Q.\mathrm{GetVertex}()$

x = (1,5)

Neighbors: {~~(1,4)~~, (2,5)}

Q = {(2,2), (1,2), (2,5)}



Depth First Search (DFS)

Iteration 5

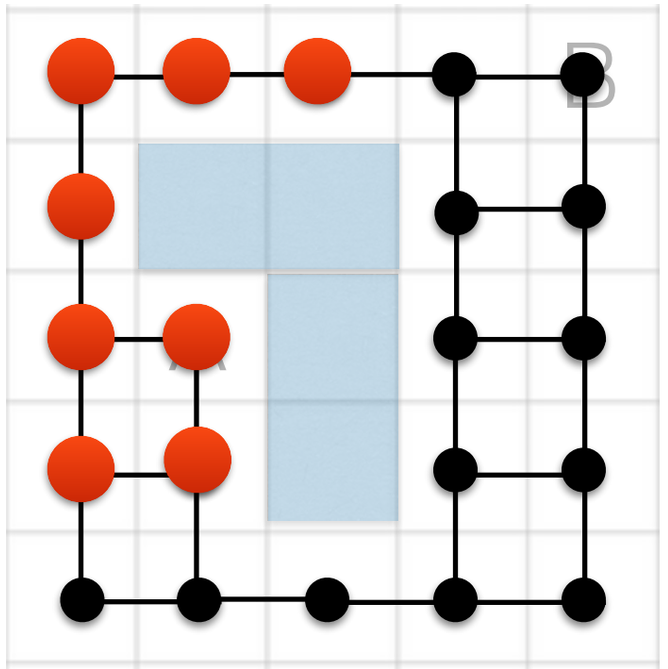
Q = {(2,2), (1,2), (2,5)}

$x \leftarrow Q.\mathrm{GetVertex}()$

x = (2,5)

Neighbors: {~~(1,5)~~, (3,5)}

Q = {(2,2), (1,2), (3,5)}



Depth First Search (DFS)

Iteration 6

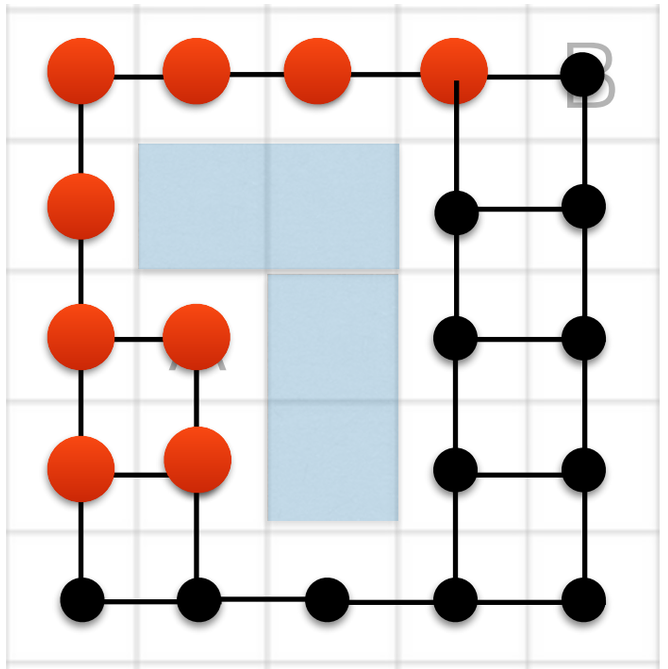
Q = {(2,2), (1,2), (3,5)}

$x \leftarrow Q.\mathrm{GetVertex}()$

x = (3,5)

Neighbors: {~~(2,5)~~, (4,5)}

Q = {(2,2), (1,2), (4,5)}



Depth First Search (DFS)

Iteration 7

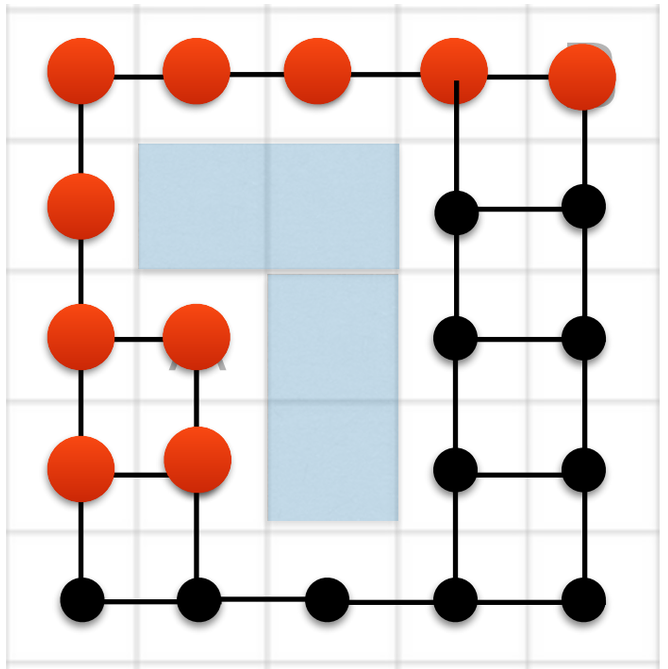
Q = {(2,2), (1,2), (4,5)}

$x \leftarrow Q.\mathrm{GetVertex}()$

x = (4,5)

Neighbors: {~~(3,5)~~, (5,5)}

Q = {(2,2), (1,2), (5,5)}



Depth First Search (DFS)

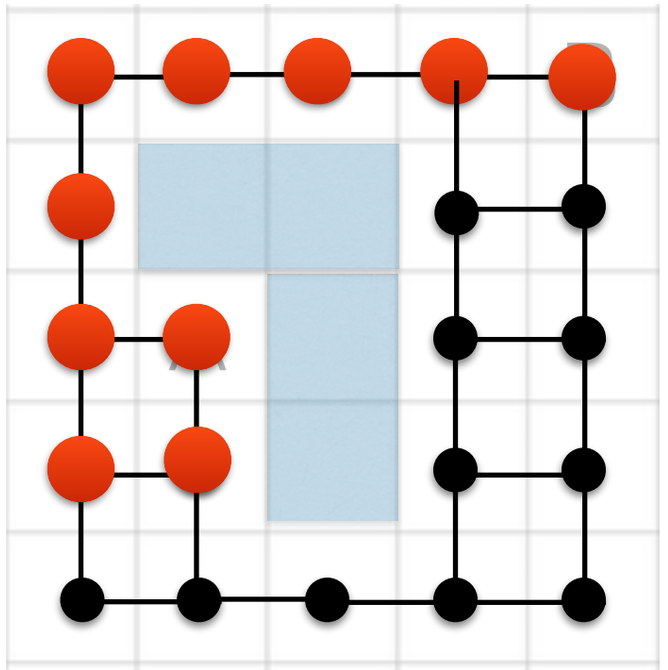
Iteration 8

Q = {(2,2), (1,2), (5,5)}

$x \leftarrow Q.\mathrm{GetVertex}()$

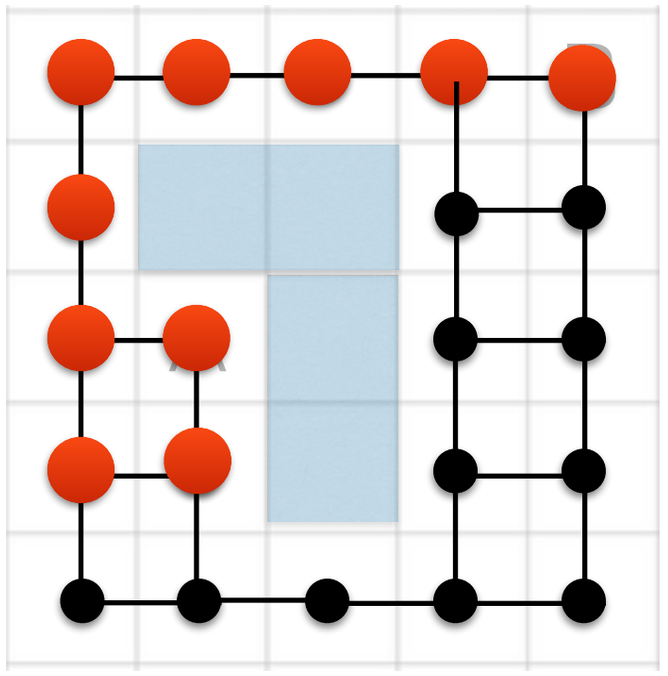
x = (5,5)

Done!



Depth First Search (DFS)

- In general, DFS will keep exploring some path deeply until it hits a dead-end or the goal
- If it hits a dead-end, it will pick some other avenue and explore until the end, and so on…
- DFS may work well in certain cases (as compared to BFS)
- e.g., when the path is long and there are only few ways to get to the goal



Concluding thoughts

- Can you think of other ways to implement Q.GetVertex()?
- Maybe we should bias our search towards the goal? (i.e., choose the element of Q that has the closest Euclidean distance to the goal)
  - We will see this in the next lecture
- We also haven't said much about optimality (we will discuss this in the next lecture too)