CS686:
Path Planning for Point Robots

Sung-Eui Yoon
(윤성의)

Course URL:
http://sgvr.kaist.ac.kr/~sungeui/MPA

KAIST

Class Objectives

- **Motion planning framework**
  - **Representations of robots and space**
  - **Discretization into a graph**
  - **Search methods**
  - **Ch. 2 of my book**

- **Last time**
  - **Class overview and grading policy w/ HWs: research oriented course**
  - **Half lectures and half presentations from students**

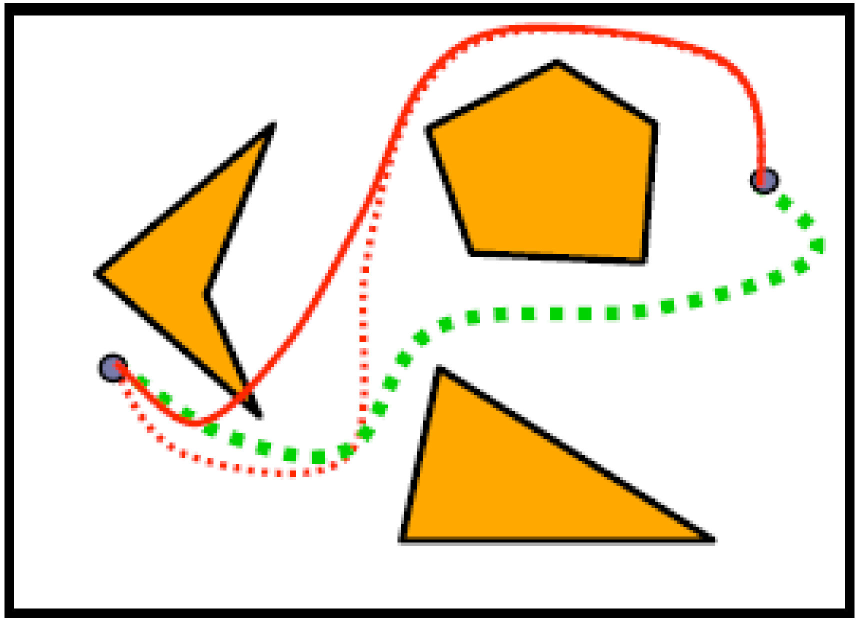
Some Announcement

- **Student stat.**
  - **CS (70%), Robotics (30%), no ME/EE (this year)**
  - **Expect to see diverse topics!!!**
- **Think about possible team mates**
  - **1 member to 3 members for each team**
- **Quiz on the prior homework**
  - **https://forms.gle/9i8sh6eF6hiKVw5JA**

KAIST

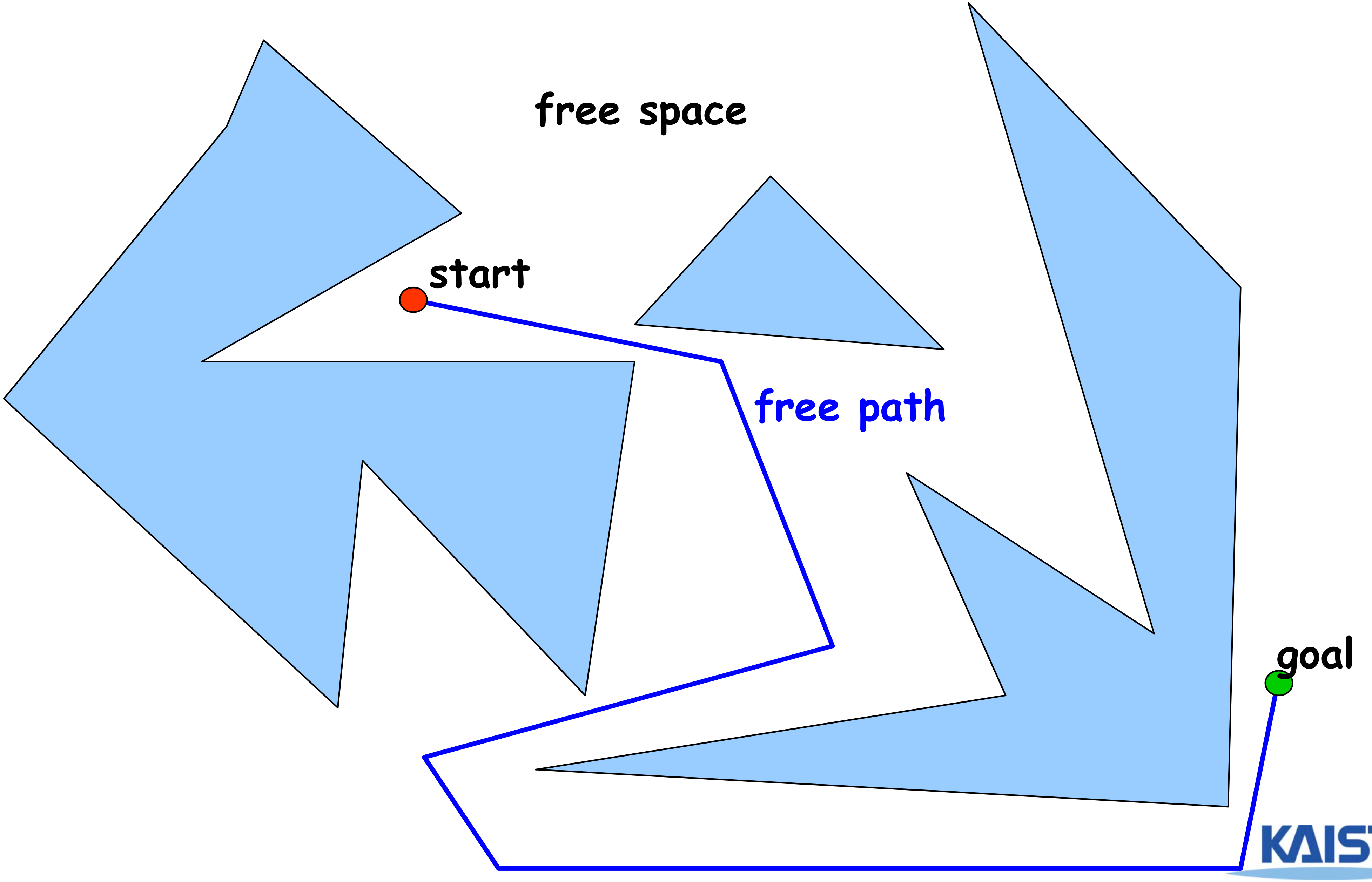
Problem

- Input
  - Robot represented as a **point** in the **plane**
  - Obstacles represented as polygons
  - Initial and goal positions
- Output
  A collision-free path between the initial and goal positions

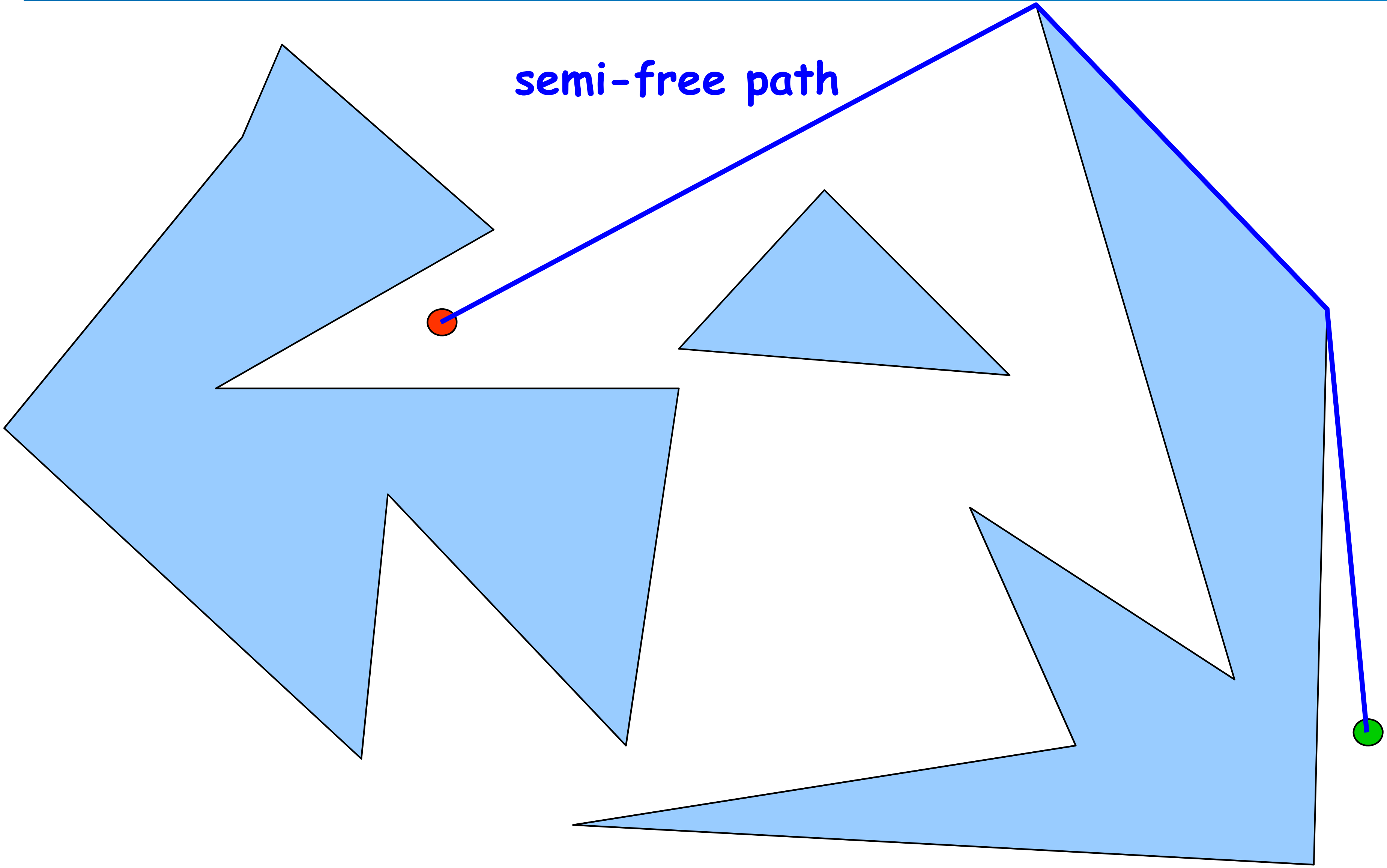


Courtesy of Prof. David Hsu

Problem

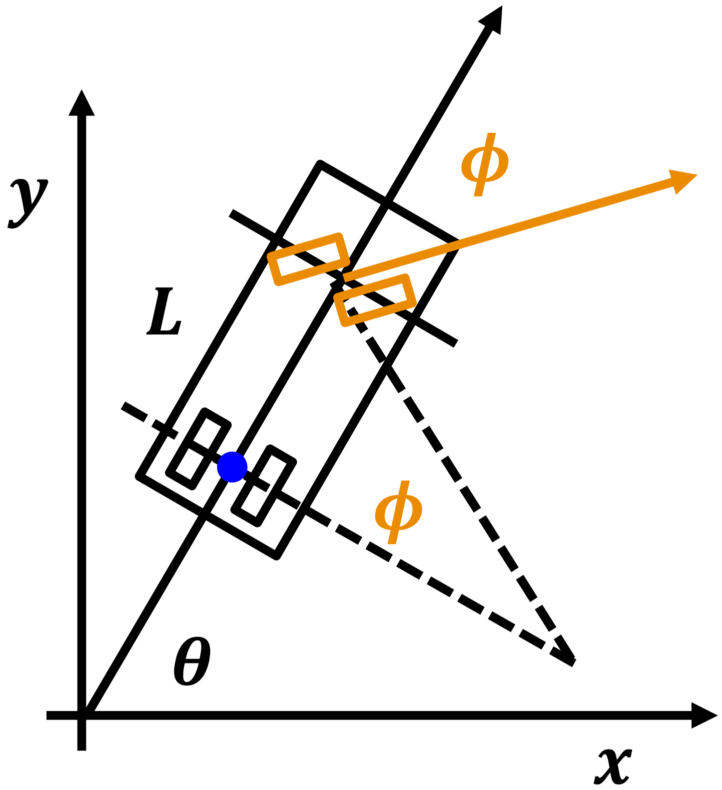


Problem



Types of Path Constraints

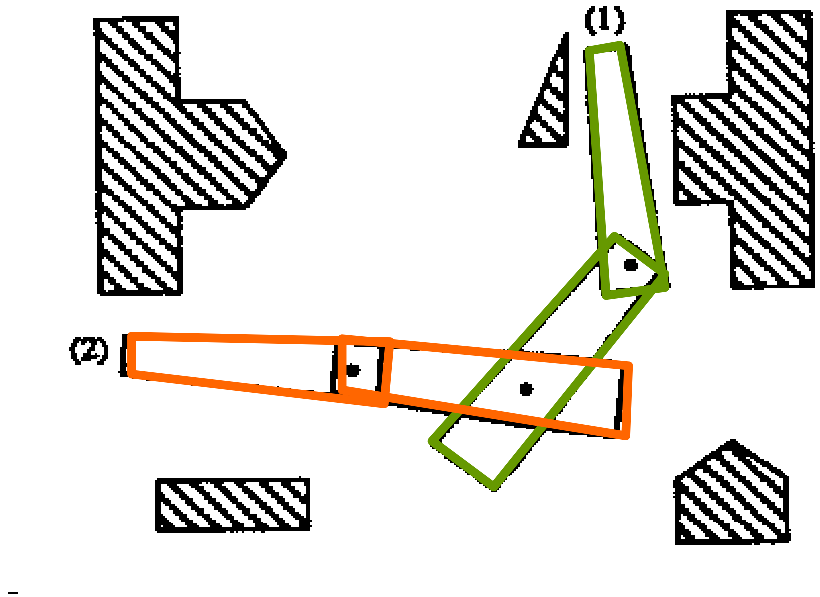
- Local constraints
  - Lie in free space
- Global constraints
  - Have minimal length
- Differential constraints
  - Cannot change the car orientation instantly



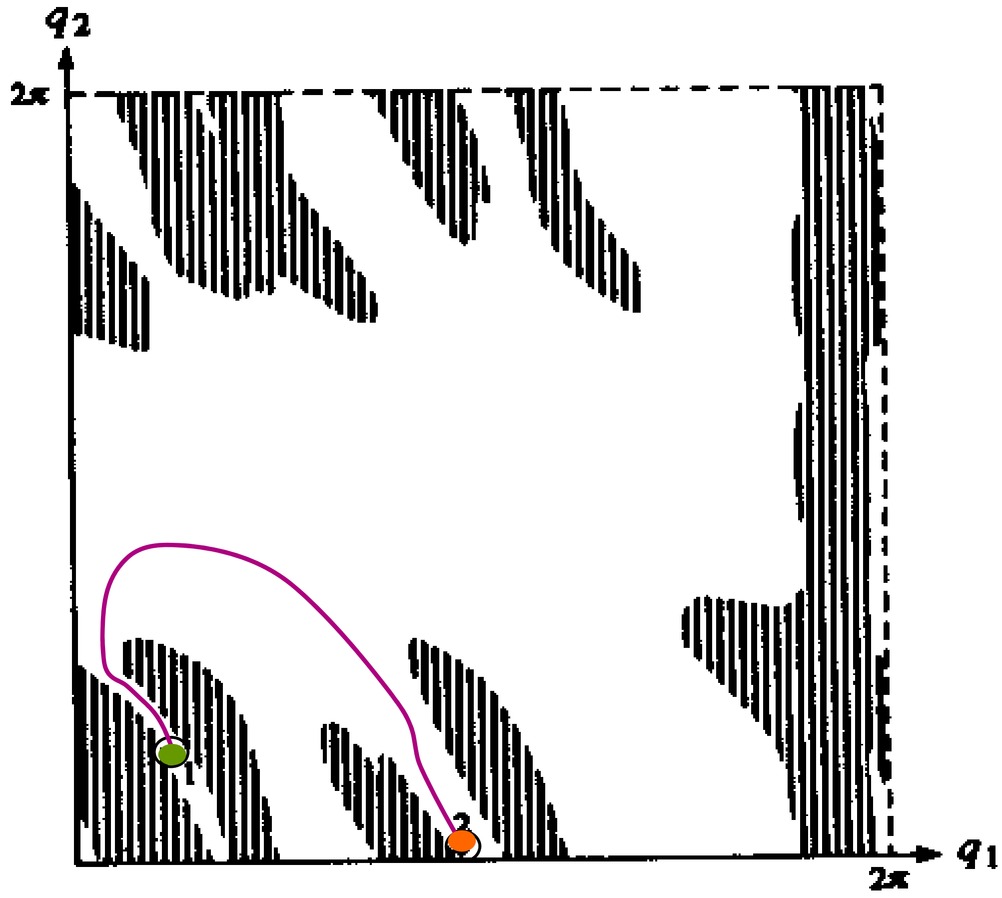
$$\frac{d\theta}{dt} = \frac{v}{L} tan(\phi)$$

See Ch. 4 (Kinematic Car Model) of my draft
http://sgvr.kaist.ac.kr/~sungeui/mp/

Configuration Space: Tool to Map a Robot to a Point



Workspace



Configuration space (C-Space)

Motion-Planning Framework

Continuous representation

(configuration space formulation)

↓

Discretization

(random sampling, processing critical geometric events)

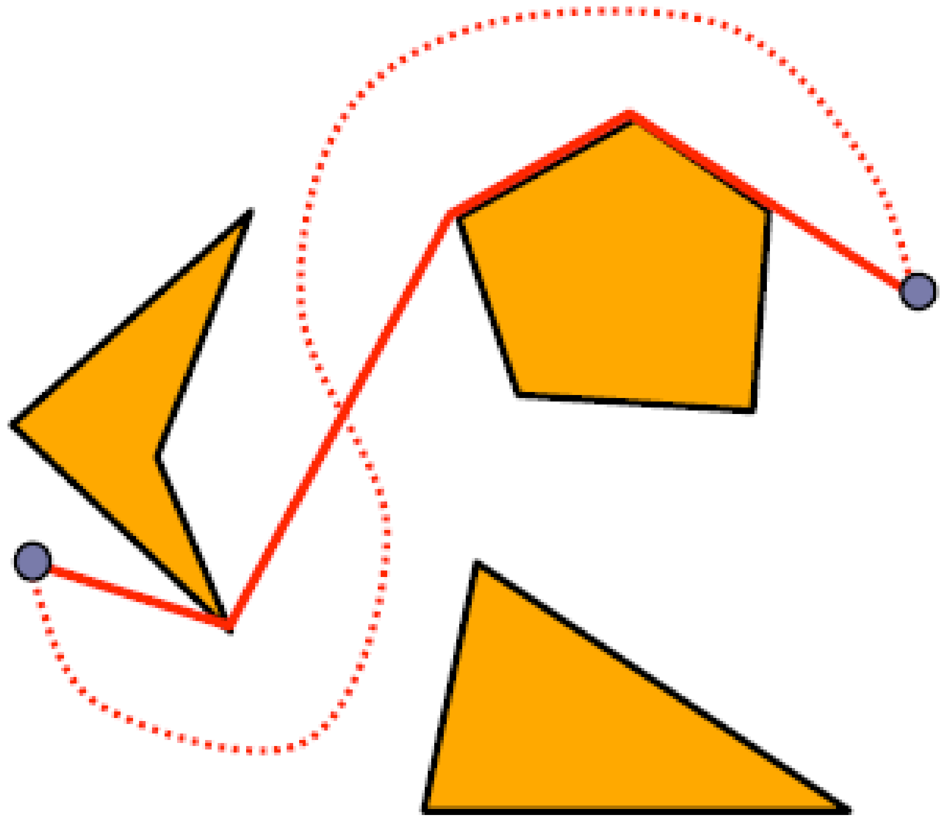
↓

Graph searching

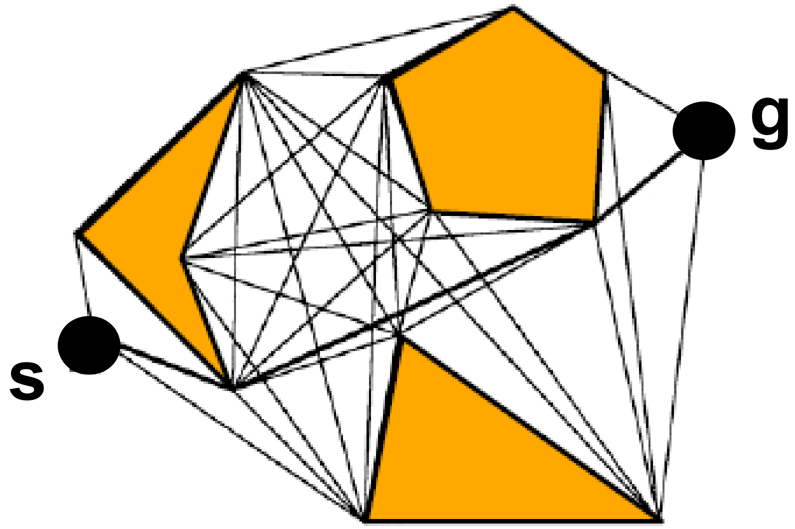
(blind, best-first, A*)

Visibility graph method

- **Observation**: If there is a a collision-free path between two points, then there is a polygonal path that bends only at the obstacles vertices.
- Why?
  Any collision-free path can be transformed into a polygonal path that bends only at the obstacle vertices.
- A **polygonal path** is a piecewise linear curve.

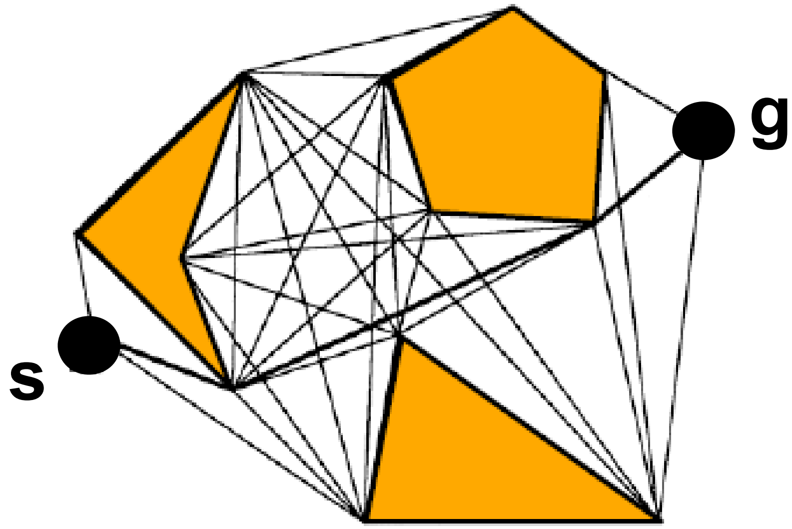


Visibility Graph



- A **visibility graph** is a graph such that
  - Nodes: s, g, or obstacle vertices
  - Edges: An edge exists between nodes u and v if the line segment between u and v is an obstacle edges or it does not intersect the obstacles

Visibility Graph



- A **visibility graph**
  - Introduced in the late 60s
  - Can produce shortest paths in 2-D configuration spaces

Simple Algorithm

- **Input: s, q, polygonal obstacles**
- **Output: visibility graph G**

```
1: for every pair of nodes u, v
2:   if segment (u, v) is an obstacle edge then
3:     insert edge (u, v) into G;
4:   else
5:     for every obstacle edge e
6:       if segment (u, v) intersects e        // check collisions
7:         go to (1);
8:     insert edge (u, v) into G;
9: Search a path with G using A*
```

Computation Efficiency

```
1: for every pair of nodes u, v                        O(n^2)
2:   if segment (u, v) is an obstacle edge then        O(n)
3:     insert edge (u, v) into G;
4:   else
5:     for every obstacle edge e                       O(n)
6:       if segment (u, v) intersects e
7:         go to (1);
8:     insert edge (u, v) into G;
```

- **Simple algorithm: $O(n^3)$ time**
- **More efficient algorithms**
  - **Rotational sweep $O(n^2 \log n)$ time, etc.**
- **$O(n^2)$ space**

Motion-Planning Framework

Continuous representation

(configuration space formulation)

↓

Discretization

(random sampling, processing critical geometric events)

↓

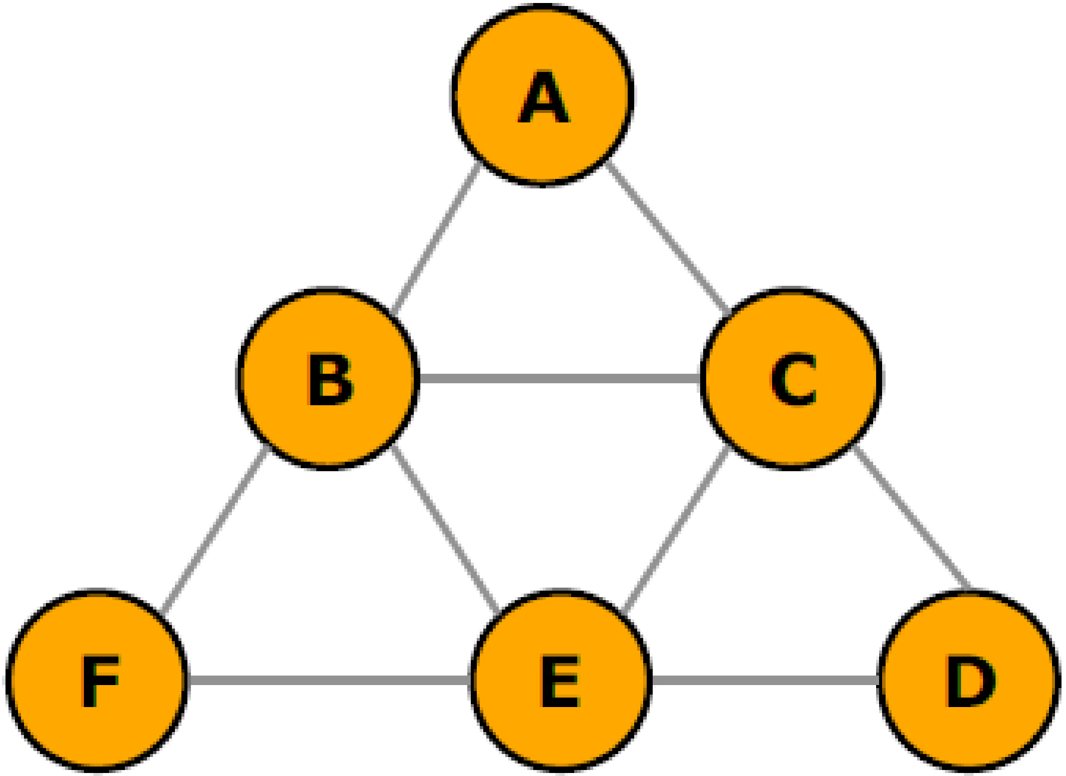
Graph searching

(blind, best-first, A*)

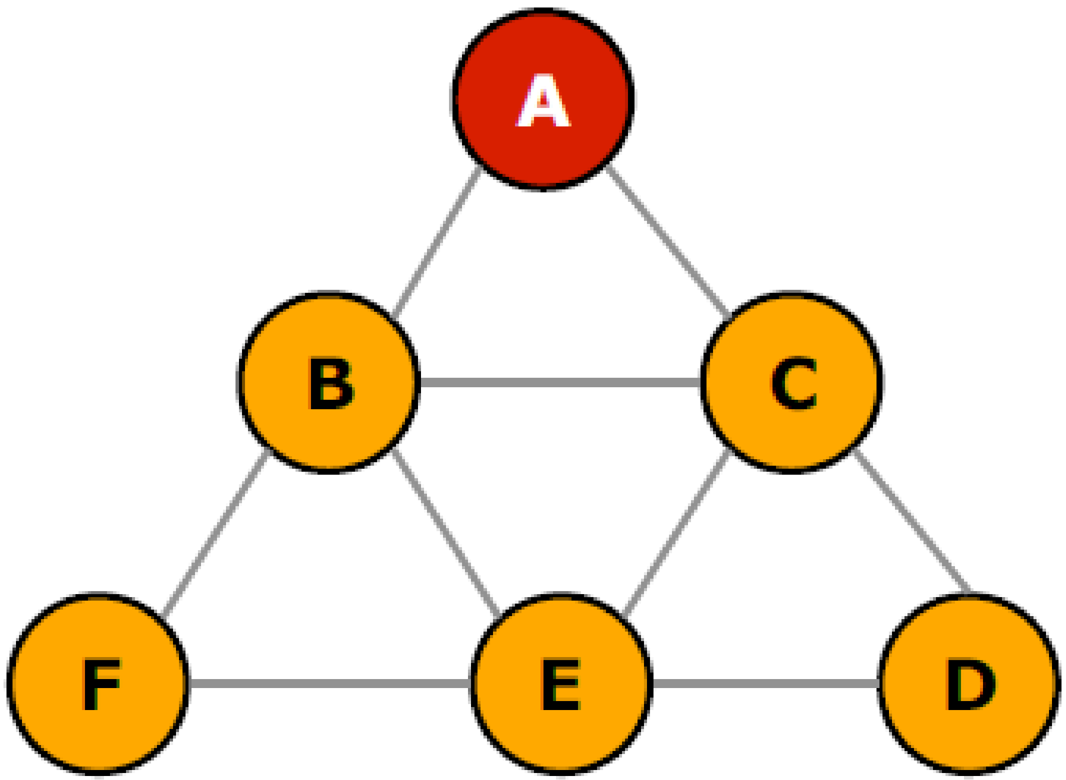
Graph Search Algorithms

- **Breadth, depth-first, best-first**
- **Dijkstra's algorithm**
- **A***

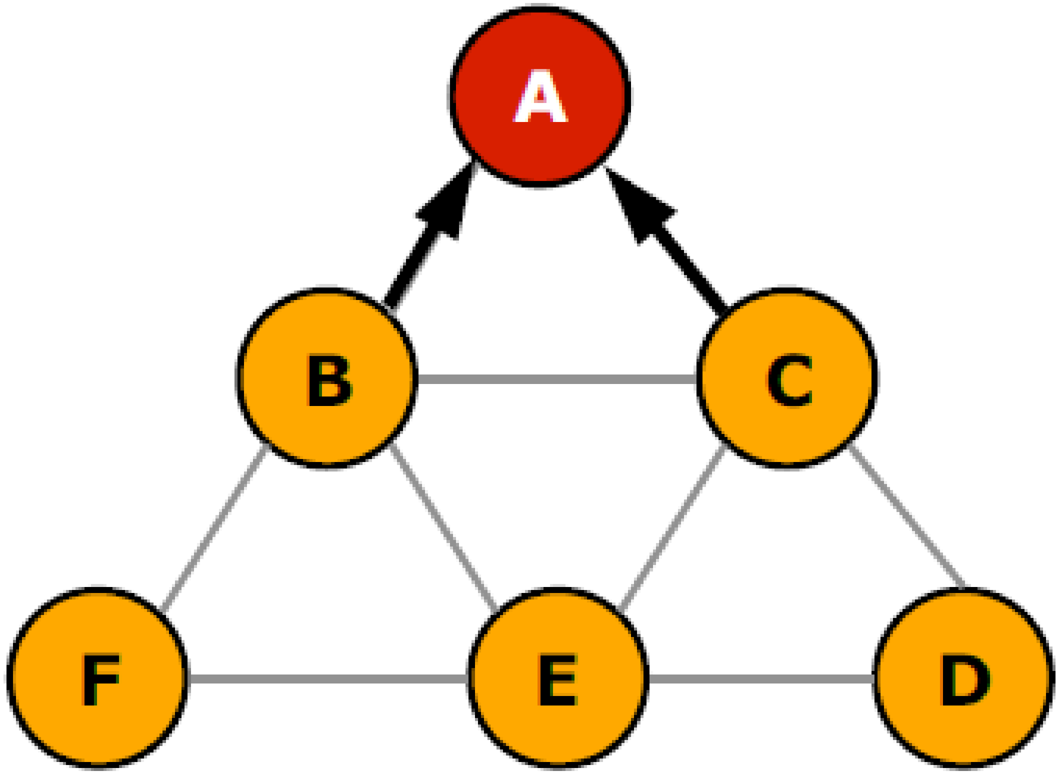
Breadth-first search



Breadth-first search

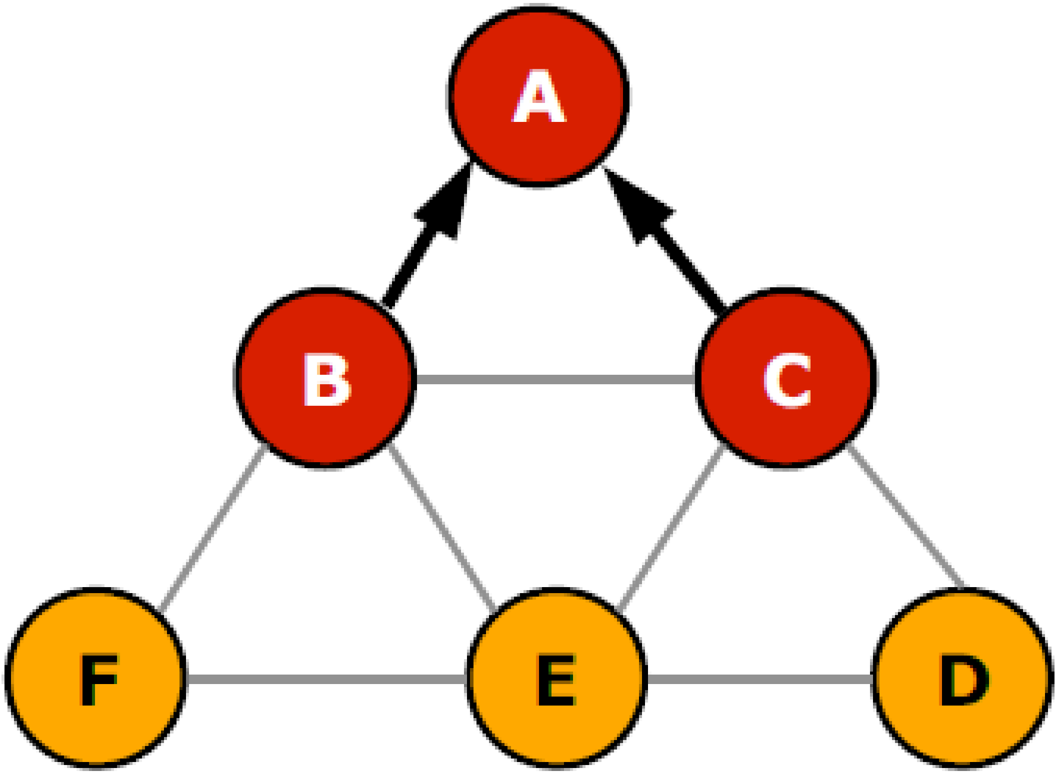


Breadth-first search



Breadth-first search

Traverse the graph by using the queue, resulting in the level-by-level traversal



Dijkstra's Shortest Path Algorithm

- **Given a (non-negative) weighted graph, two vertices, s and g:**
  - **Find a path of minimum total weight between them**
  - **Also, find minimum paths to other vertices**
  - **Has O ($|V|$ lg $|V|$ + $|E|$), where V & E refer vertices & edges**

Dijkstra's Shortest Path Algorithm

- **Set S**
  - **Contains vertices whose final shortest-path cost has been determined**

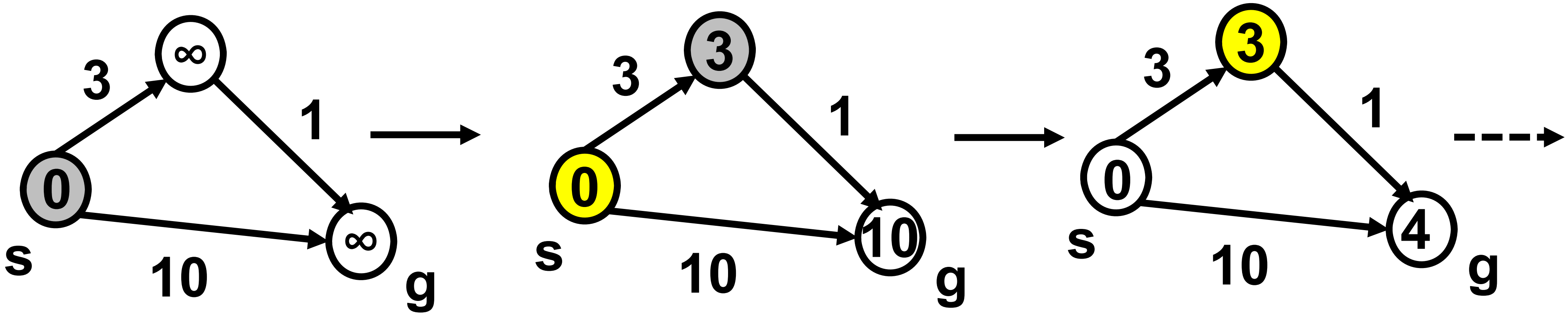
- **DIJKSTRA (G, s):**

Input: G is an input graph, s is the source

```
Initialize-Single-Source (G, s)
S ← empty
Queue ← Vertices of G
While Queue is not empty
    Do u ← min-cost from Queue
        S ← union of S  and {u}
        for each vertex v in Adj [u]
            do RELAX (u, v)
```

Dijkstra's Shortest Path Algorithm

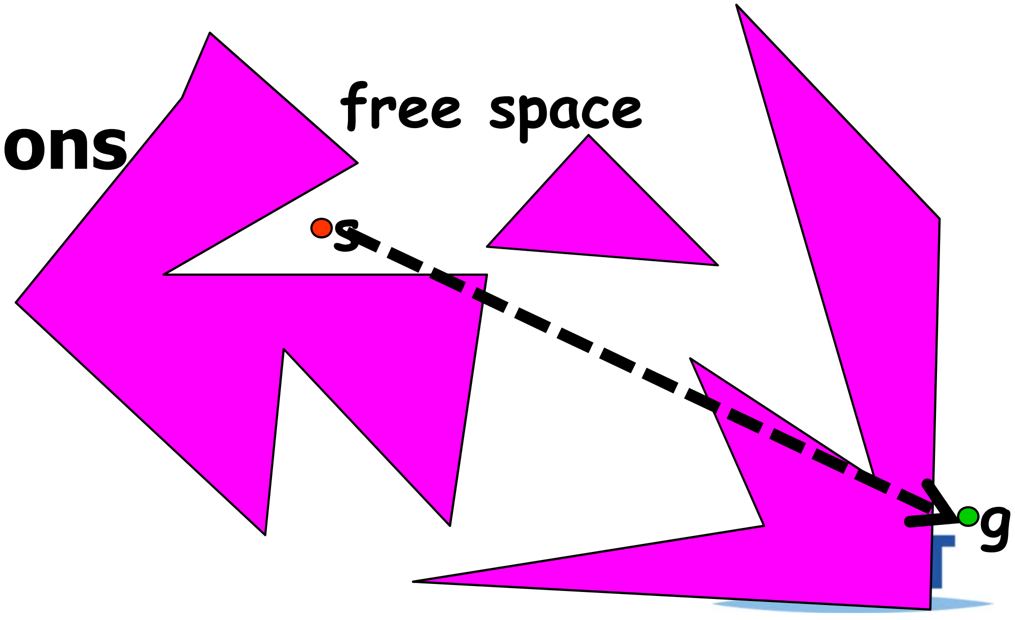
Compute optimal cost-to-come at each iteration



Yellow vertices are in a set with shortest costs
White vertices are in the queue.
Shaded one is chosen for relaxation.

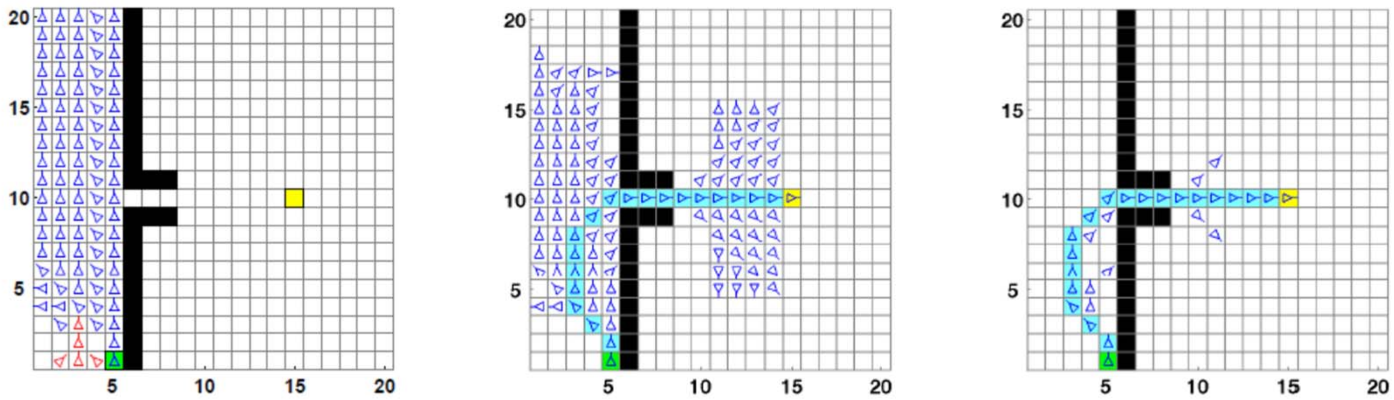
A* Search Algorithm

- An extension of Dijkstra's algorithm based on a heuristic estimate
  - Conservatively estimate the cost-to-go from a vertex to the goal
  - The estimate should not be greater than the optimal cost-to-go
  - Sort vertices based on "cost-to-come + the estimated cost-to-go"
  - Can find optimal solutions with fewer steps



K* Algorithm (Video)

- **Recursive Path Planning Using Reduced States for Car-like Vehicles on Grid Maps**
  - **IEEE Transactions on Intelligent Transportation System**



(a) H_{free} (b) H_{obs} (c) H_{free} & H_{obs}

- **A* and its variants are quite commonly used for its optimality and high performance**

Framework

continuous representation

↓

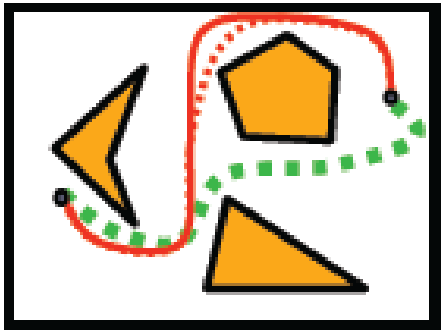
discretization

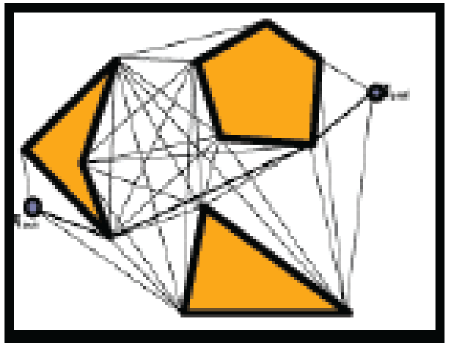
construct visibility graph

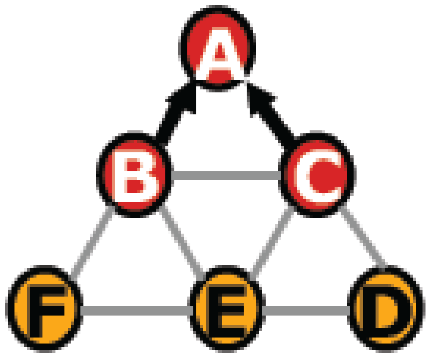
↓

graph searching

breadth-first search







Computational Efficiency

- Running time $O(n^3)$
  - Compute the visibility graph
  - Search the graph
- Space $O(n^2)$

- Can we do better?
  - Lead to classical approaches such as roadmap

Class Objectives were:

- Motion planning framework
  - Representations of robots and space
  - Discretization into a graph
  - Search methods
  - Ch. 2 of my book

Homework

- **Browse 2 ICRA/IROS/RSS/CoRL/WAFR/TRO/IJRR papers**
  - **Submit it online before the Tue. Class**
  - **https://forms.gle/2jdXkgYu5snyAb3s8**

- **Example of a summary (just a paragraph)**

  Title: XXX XXXX XXXX
  Conf./Journal Name: ICRA, 2020
  Summary: this paper is about accelerating the performance of collision detection. To achieve its goal, they design a new technique for reordering nodes, since by doing so, they can improve the coherence and thus improve the overall performance.

Valid Papers for Paper Presentation

- **Related to the course theme**
- **Top-tier conf/journals**
  - **No arxiv paper, unless it has meaningful citation counts (say, 10 per year)**
- **Recent ones**
  - **Published at 2016~2020**

KAIST

Homework for Every Class

- **Go over the next lecture slides**
- **Come up with one question on what we have discussed today and submit at the end of the class**
  - **1 for typical questions**
  - **2 for questions with thoughts or that surprised me**

- **Write a question two times before the mid-term exam**
  - **https://forms.gle/R2ZcS9pZ9me9RzmKA**

Next Time….

- **Classic path planning algorithms**