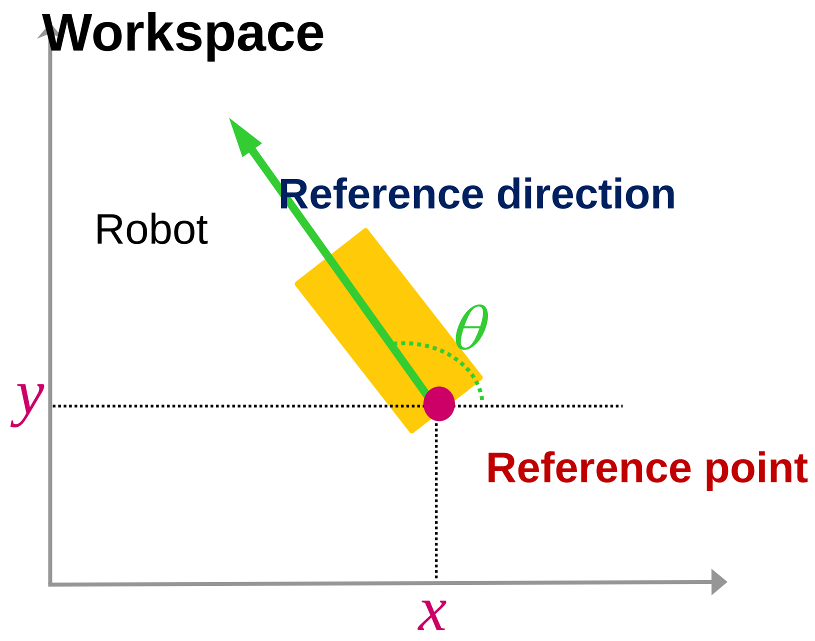
Discrete planning

Jane Li

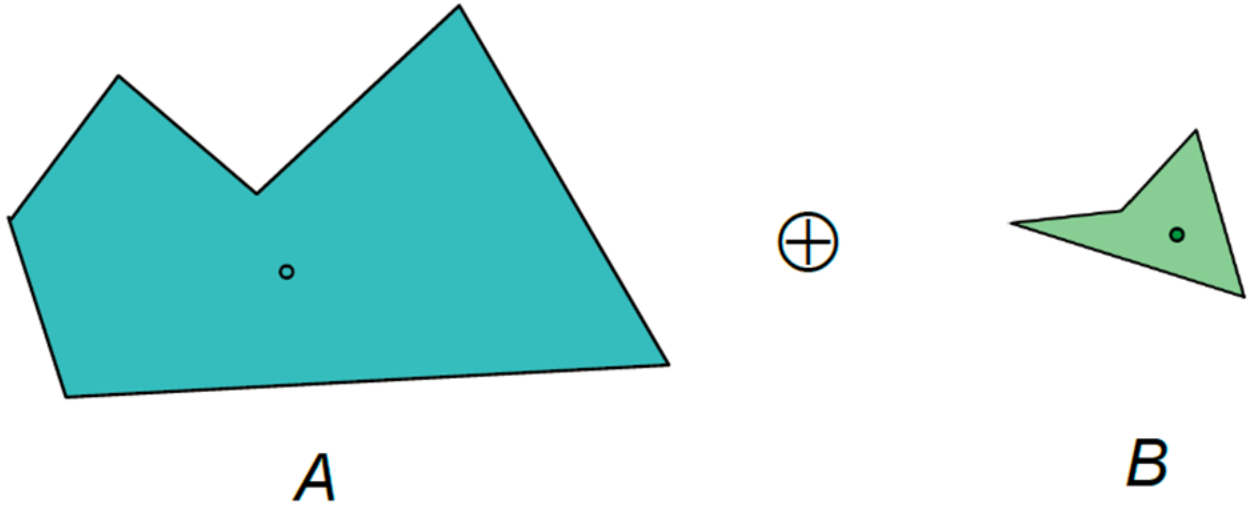
Assistant Professor
Mechanical Engineering Department, Robotic Engineering Program
Worcester Polytechnic Institute

Quiz (10 pts)

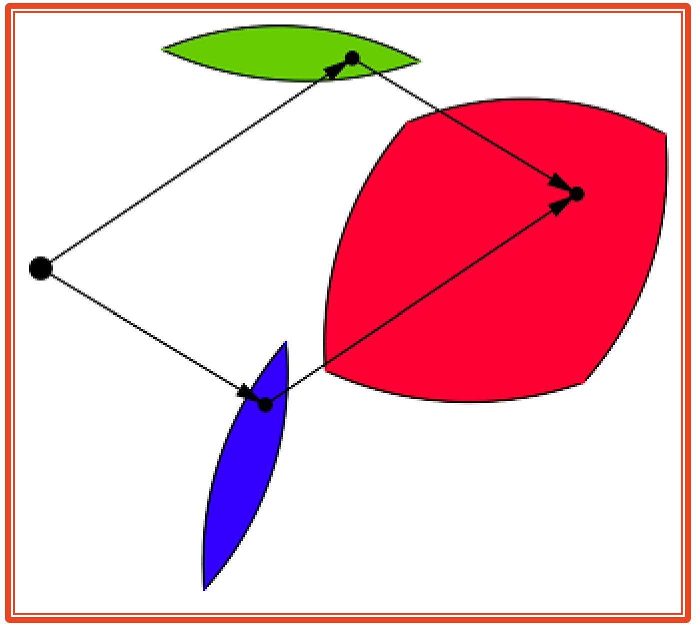
- (3 pts) How to compute Minkowski Sum?
- (3 pts) How to compute the configuration space obstacle for a 2D rigid robot?
- (4 pts) What are homotopic paths?



Minkowski Sum

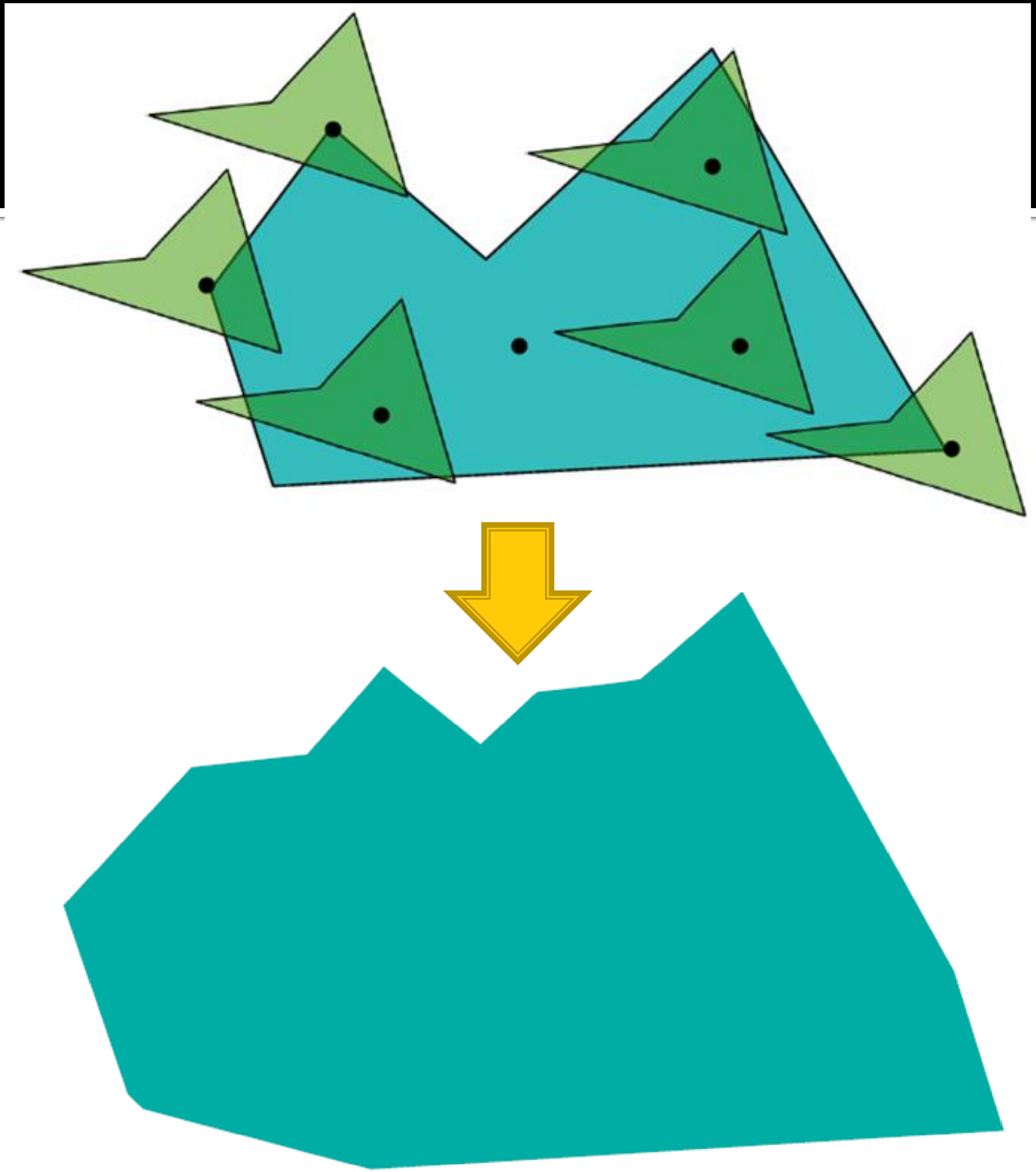


$$A \oplus B = \{a + b \mid a \in A, b \in B\}$$

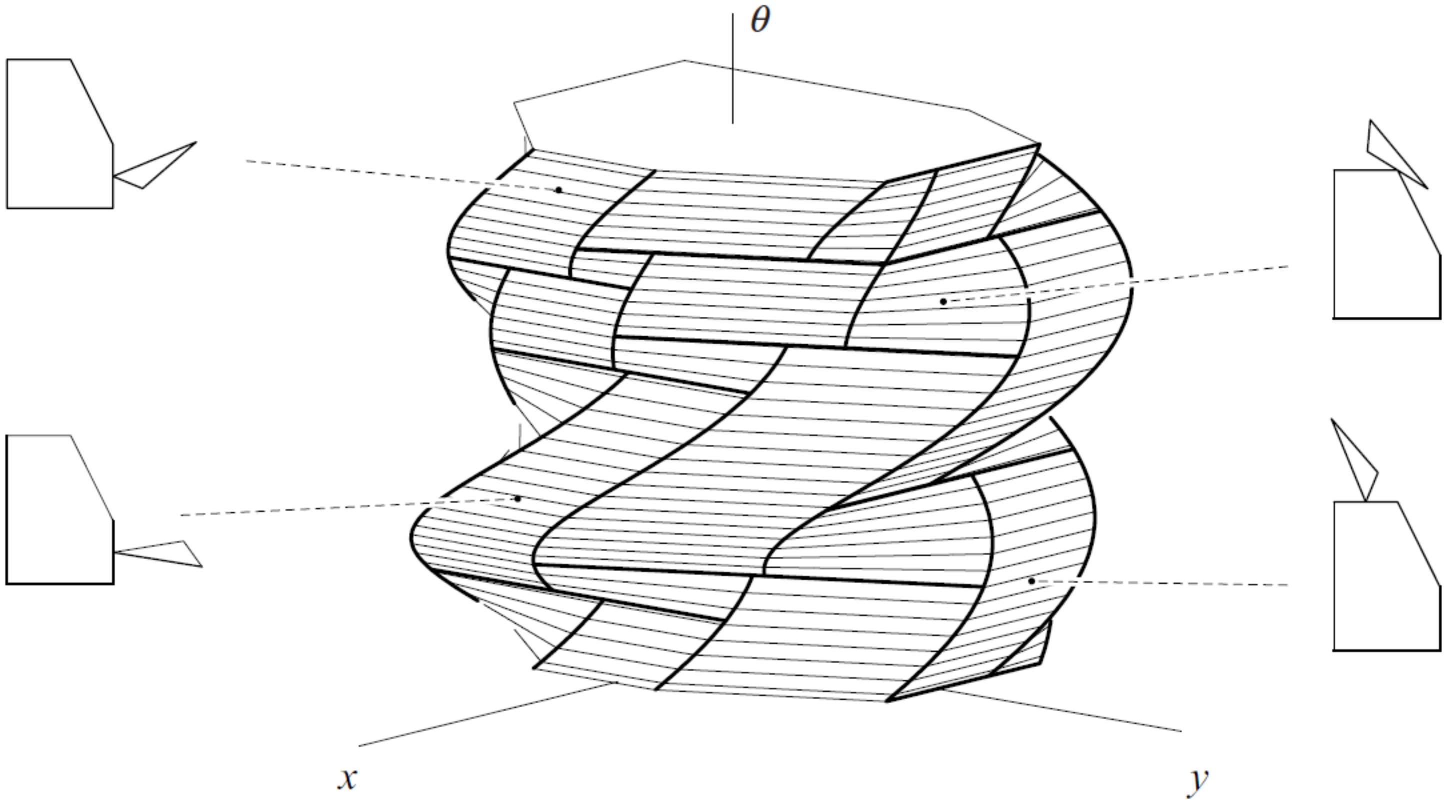


Minkowski Sum

- Dip B into paint
- Put B's origin on A's border
- Translate it along A's edge
- Sum = the painted area

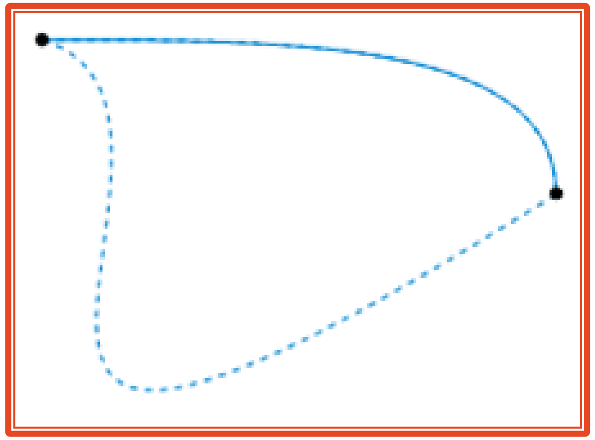


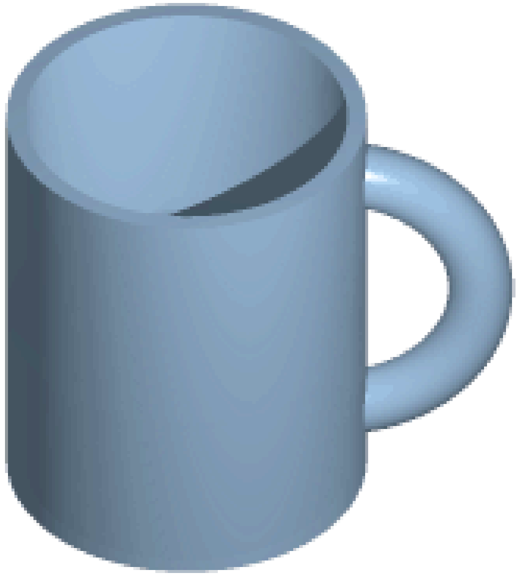
Example – 2D Robot with Rotation



Homotopic paths

- Two paths with the same endpoints is **homotopic** if one path can be deformed into continuously deformed into the other





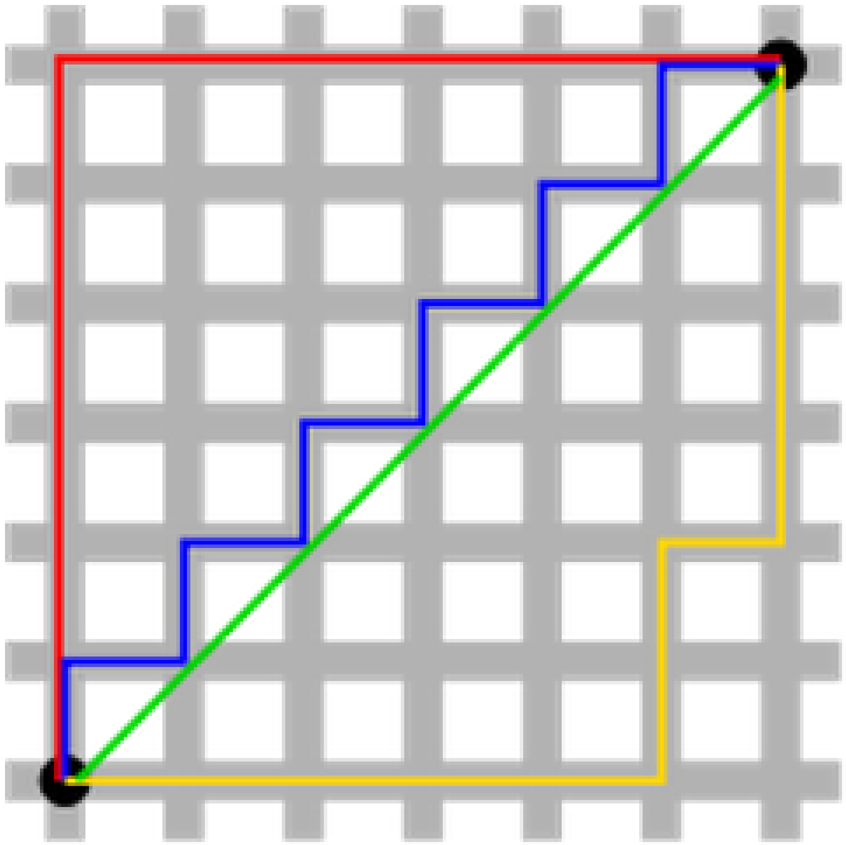
Distance in C-space

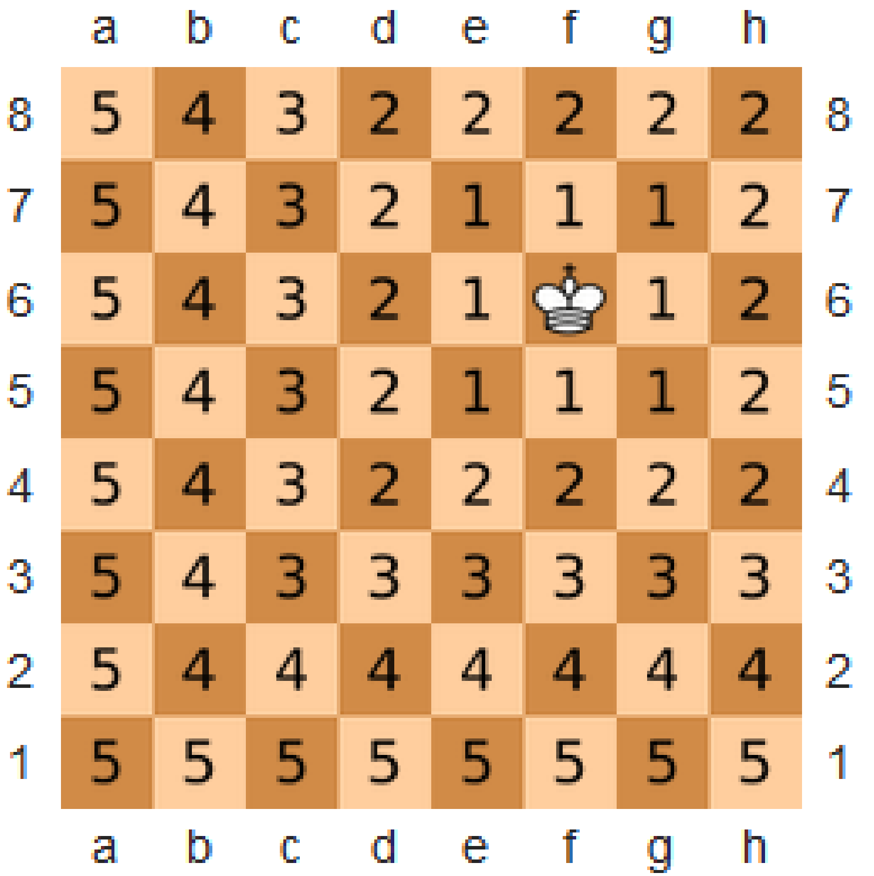
Discussion

- Do we need a specialized distance metric in C-space to do path planning?

- Metrics for distance?
  - Euclidian distance
  - Other metrics?

Distance in C-space





Distance metrics

- L1-norm (Manhattan distance)

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

- L2-norm (Euclidian distance)

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

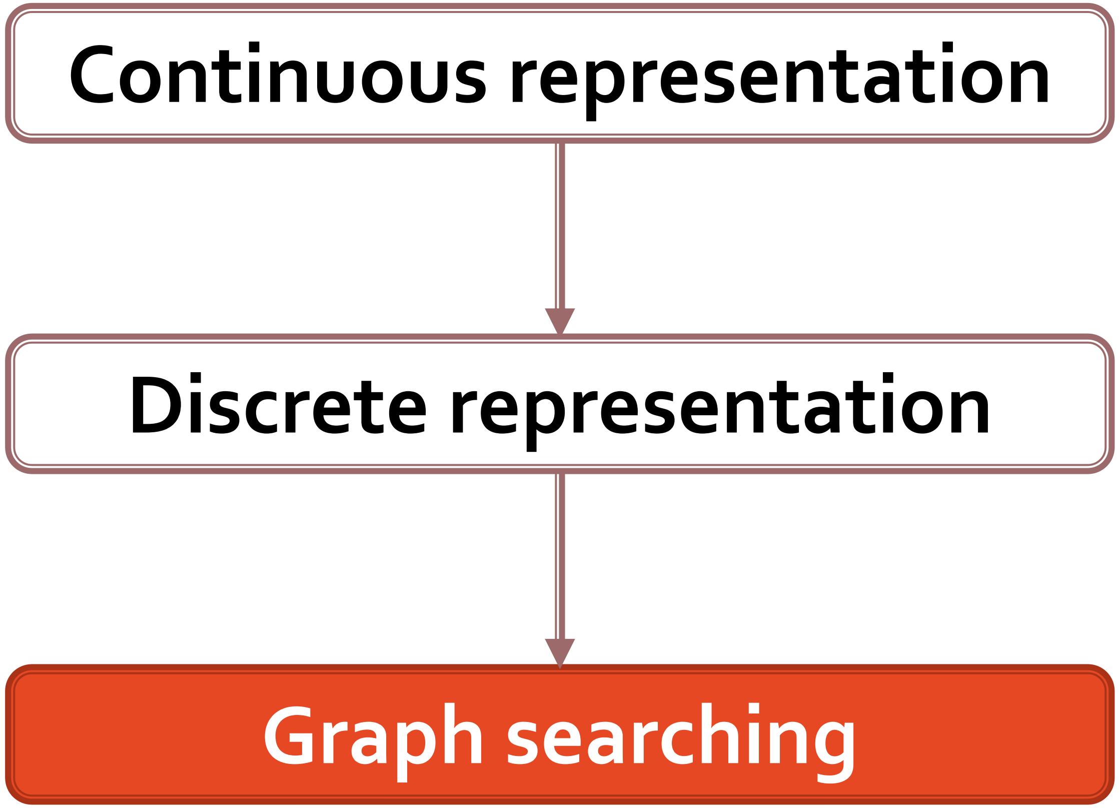
- L_∞-norm (chessboard distance)

$$D_{\mathrm{Chebyshev}}(p, q) := \max_i(|p_i - q_i|).$$

Discrete planning

Nothing but search

- What to search?
- How to search?



What to search

- Your map
  - Data structure matters
- Your goal
  - How improve the chance to hit your goal quickly?
- Your path
  - A feasible, optimal solution?

How to search

- Search algorithms – you name it
  - Breadth-first search
  - Depth-first search
  - Dijkstra's algorithm
  - Best-first Search
  - A* search
  - …

How to search

- More issues
  - Do you need to search again, and again?
  - What if you search within limited amount of time?
  - What if your search may terminate all of sudden?

Discrete Search

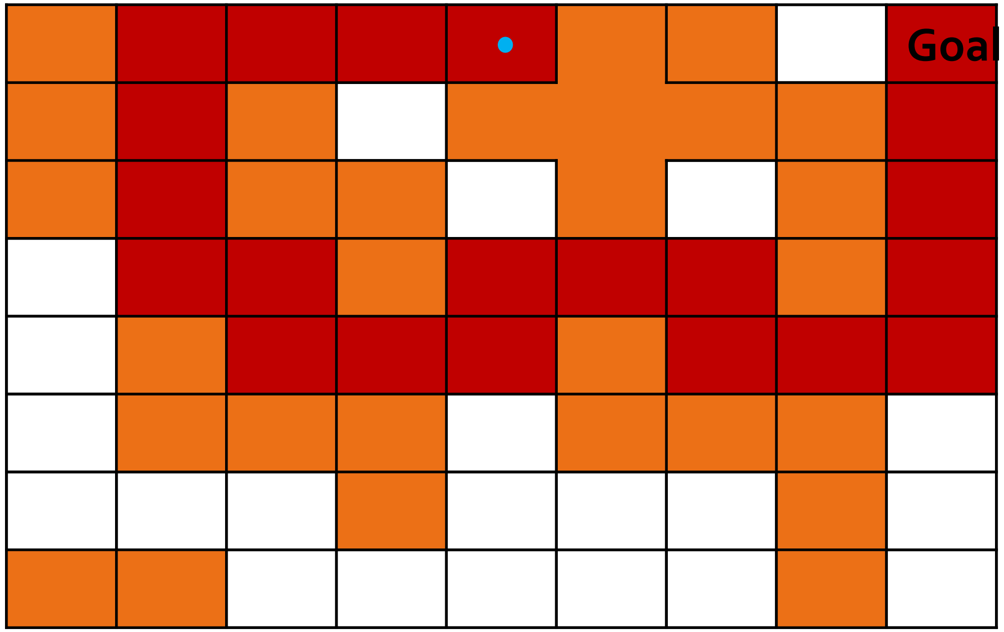
- Discrete search
  - Find a finite sequence of discrete actions from start to goal
- CAUTION
  - Discrete search can be sensitive to **dimensionality** of state space

Problem formulation

- State Space
  - The whole world to search in
- Action
  - What action(s) to take at a state
- Successor
  - Given my current state, where to search next?
- Action cost
  - Cost of performing action a at the state s
- Goal Test
  - Condition for termination

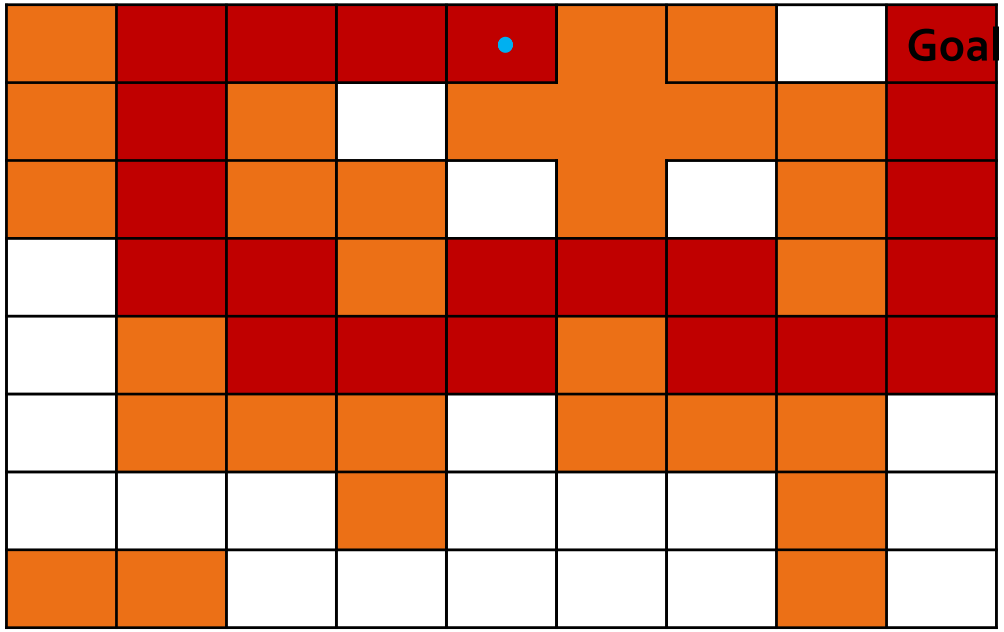
Example

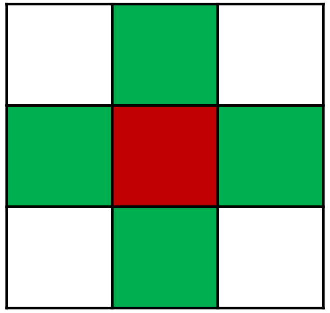
- State Space?
  - The space of cells (x-y coordinates)



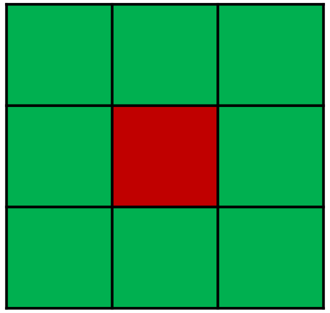
Example

- Successor?
  - Neighbor cells
  - 4 connected vs. 8 connected?





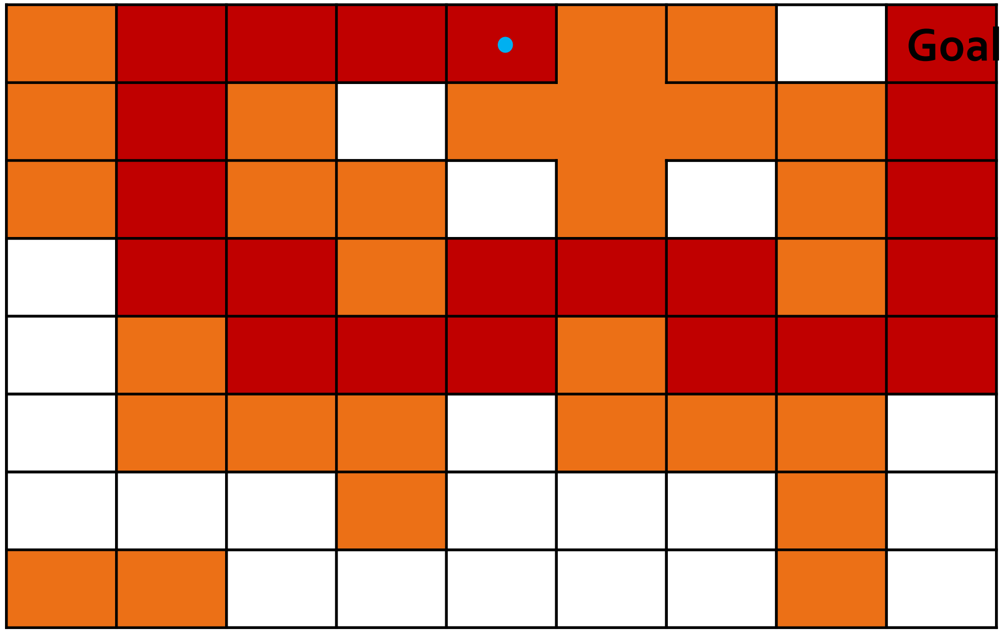
4-connected



8-connected

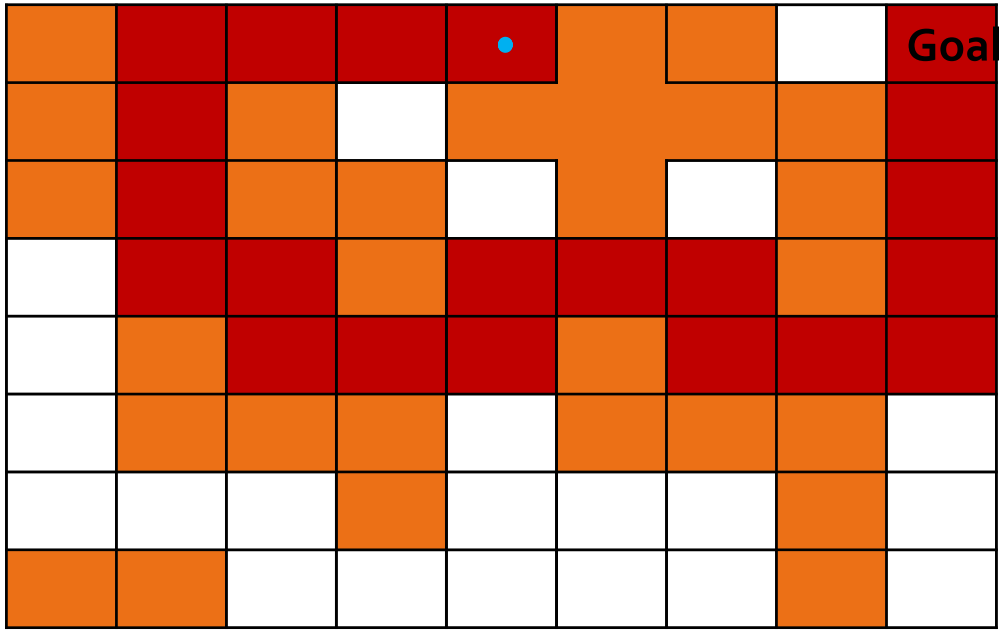
Example

- Actions?
  - Move to a neighboring cell



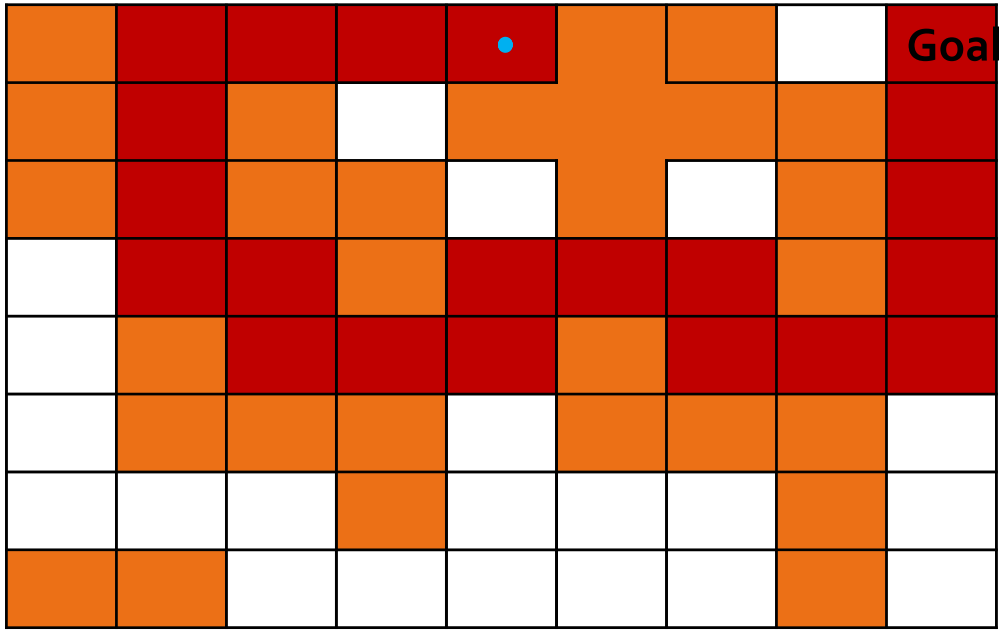
Example

- Action Cost
  - Distance between cells traversed
  - Same for 4 vs 8 connected?



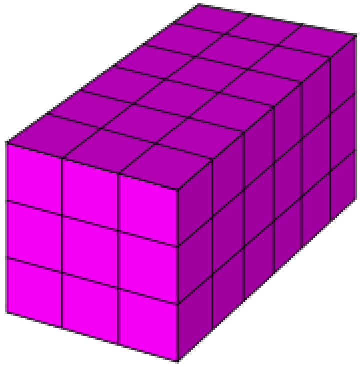
Example

- Goal Test
  - Check if at goal cell
  - Multiple cells can be marked as goals?
  - Example?

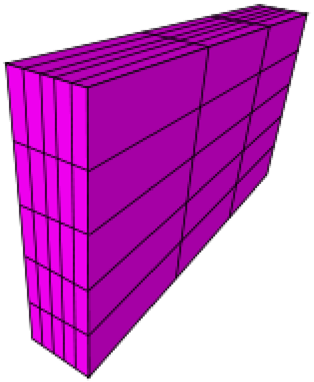


State space

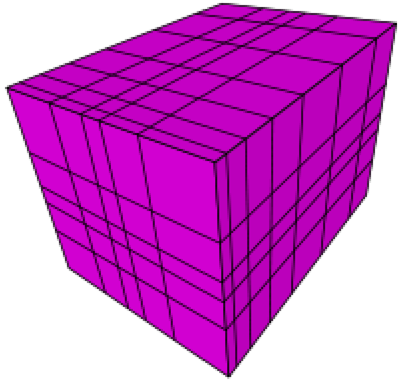
- For motion planning, state space is usually a **grid**
- There are many kinds of grids!



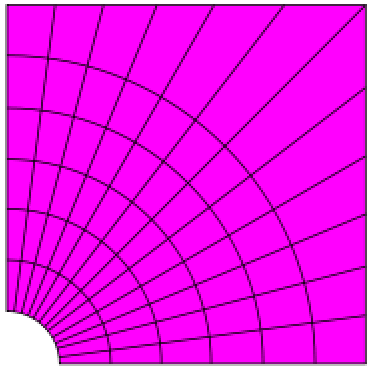
Cartesian Grid



Regular Grid



Rectilinear Grid



Curvilinear Grid

State space

- The choice of grid (i.e. state space) is **crucial**
  - Performance
  - Accuracy

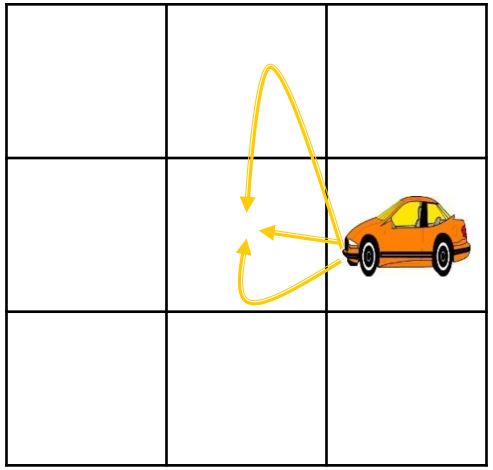
- The world is really continuous; these are all approximations

Actions

- Actions in motion planning are also often **continuous**

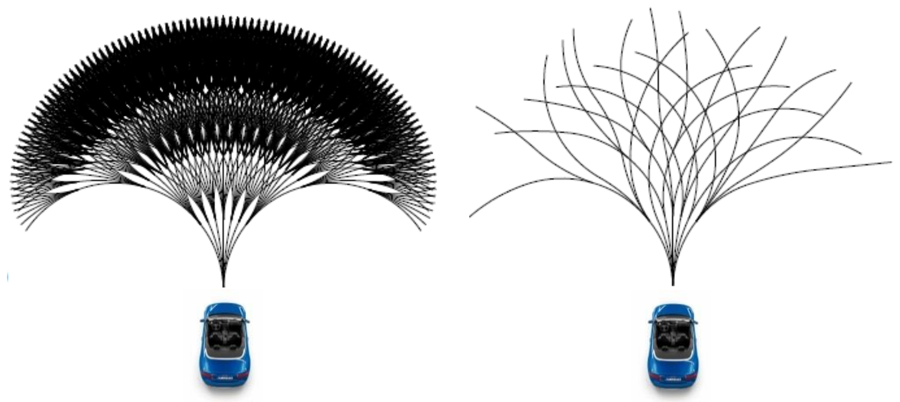
Why?

- There are **many** ways to move between neighboring cells



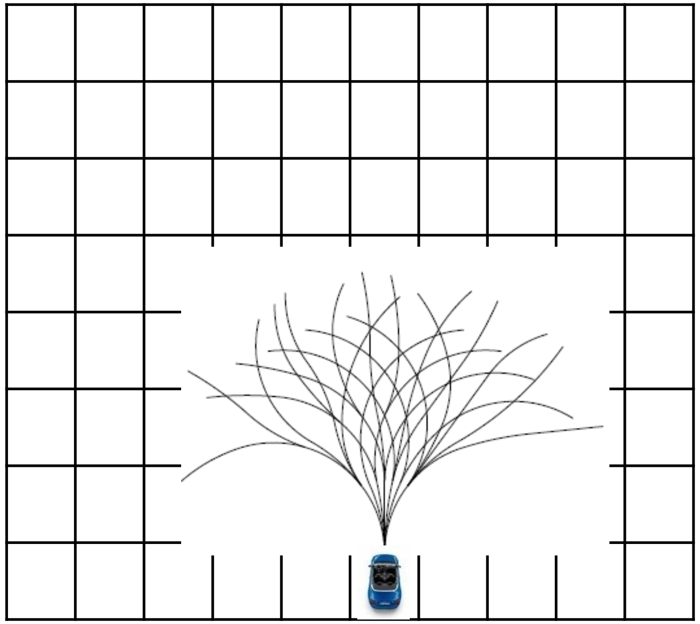
Action

- Usually, pick a discrete action set a **priori**
- What are the tradeoffs in picking action sets?
  - A major issue in in non-holonomic motion planning



Successors

- These are largely determined by the **action set**
- Successors may not be known a priori
  - Try each action to see which cell you end in



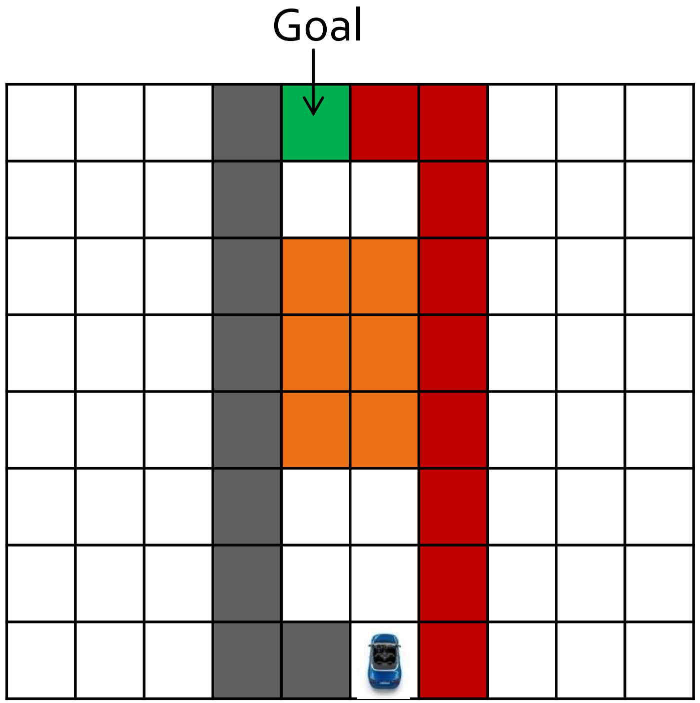
Action Cost

- Depends on what you're trying to **optimize**
  - Cost for Minimum Path Length?
  - Cost for motion smoothness?
  - Other cost?

- Sometimes we consider more than one criterion
  - Weighted cost → coefficient?

Goal Test

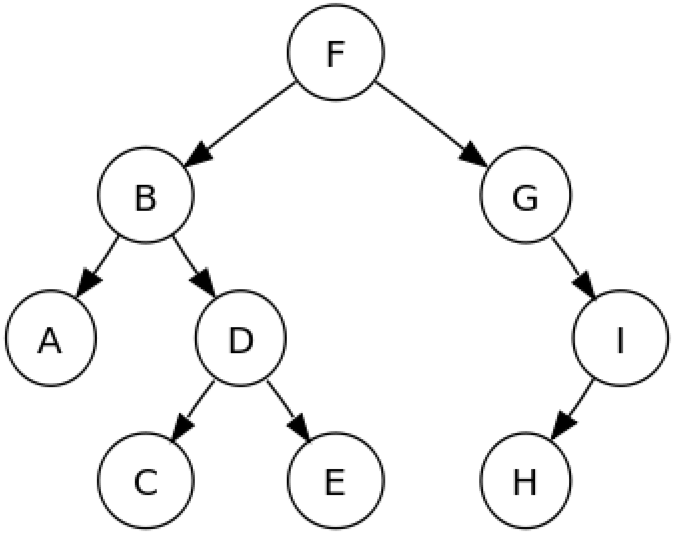
- Goals are most commonly specific cells you want to get to
- But they can be more abstract, too!
- Example Goals?
  - A state where X is visible
  - A state where the robot is contacting X
  - Topological goals



Tree Search Algorithms

```
function Tree-Search(problem, strategy)
    Root of search tree <- Initial state of the problem
    While 1
        If no nodes to expand
                return failure
        Choose a node n to expand according to strategy
        If n is a goal state
                return solution path  //back-track from goal to
                                      //start in the tree to get path
        Else
                NewNodes <- expand n
                Add NewNodes as children of n in the search tree
```

The Trees you know



Tree Search Algorithms

- All you can choose is the strategy
  - Which node to expand next

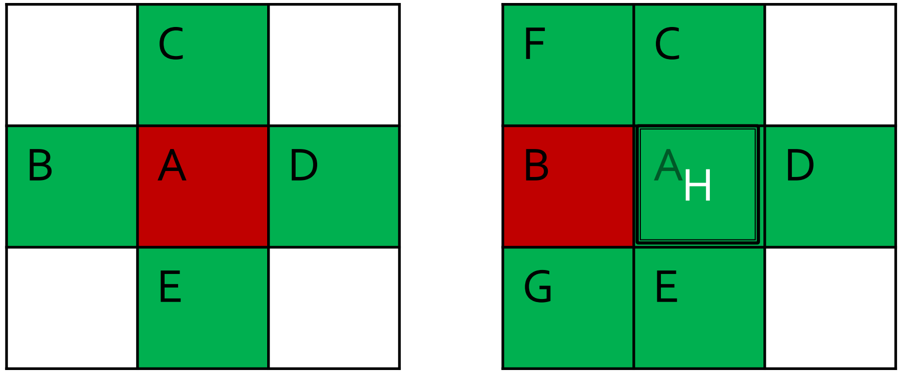
- What does the strategy choice affects?
  - Completeness – Does the algorithm find a solution if one exists?
  - Optimality – Does it find the least-cost path?
  - Run Time
  - Memory usage

Tree Search Algorithms

- What may affect run time and memory usage?
  - Branching Factor – how many successors a node has
  - Solution Depth – How many levels down the solution is
  - Space Depth – Maximum depth of the space

Tree Search Algorithms

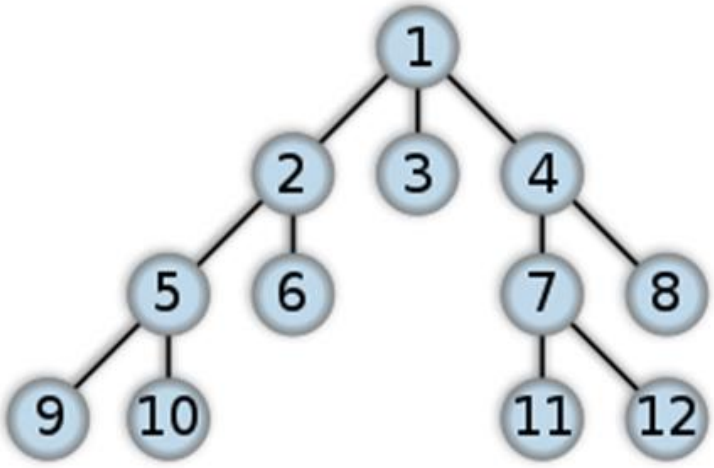
- Need to **avoid re-expanding** the same state



- Solution?
  - An ***open list*** to track which nodes are unexpanded
  - E.g., a queue (First-in-first-out)

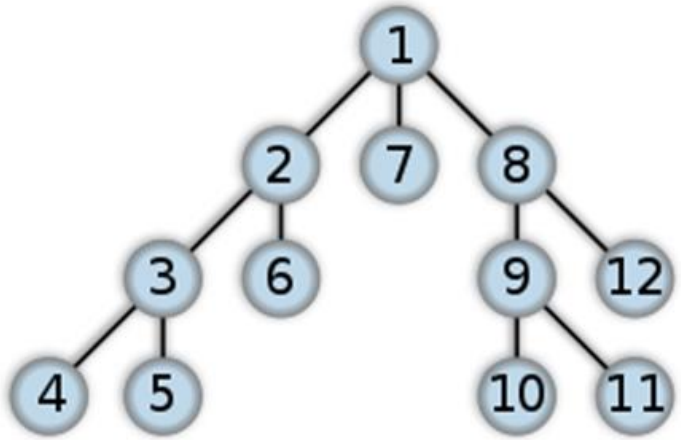
Breadth-first Search (BFS)

- Main idea
  - Build search tree in layers
- Open list?
  - Queue – Insert new nodes to the **back**
- Expansion strategy?
  - "Oldest" nodes are expanded first
- Optimality?
  - Yes, BFS can find the **shortest** path to the goal



Depth-first Search (DFS)

- Main idea
  - Go as deep as possible as fast as possible
- Open list?
  - Stack – Insert new nodes to the **front**
- Expansion strategy?
  - "Newest" nodes are expanded first
- Optimality?
  - DFS does **NOT** necessarily find the **shortest** path to the goal



Efficiency

- BFS v.s. DFS?
  - Depending on the data structure and what you are looking for, either DFS or BFS could be advantageous
- When would **BFS** be very inefficient?
- When would **DFS** be very inefficient?

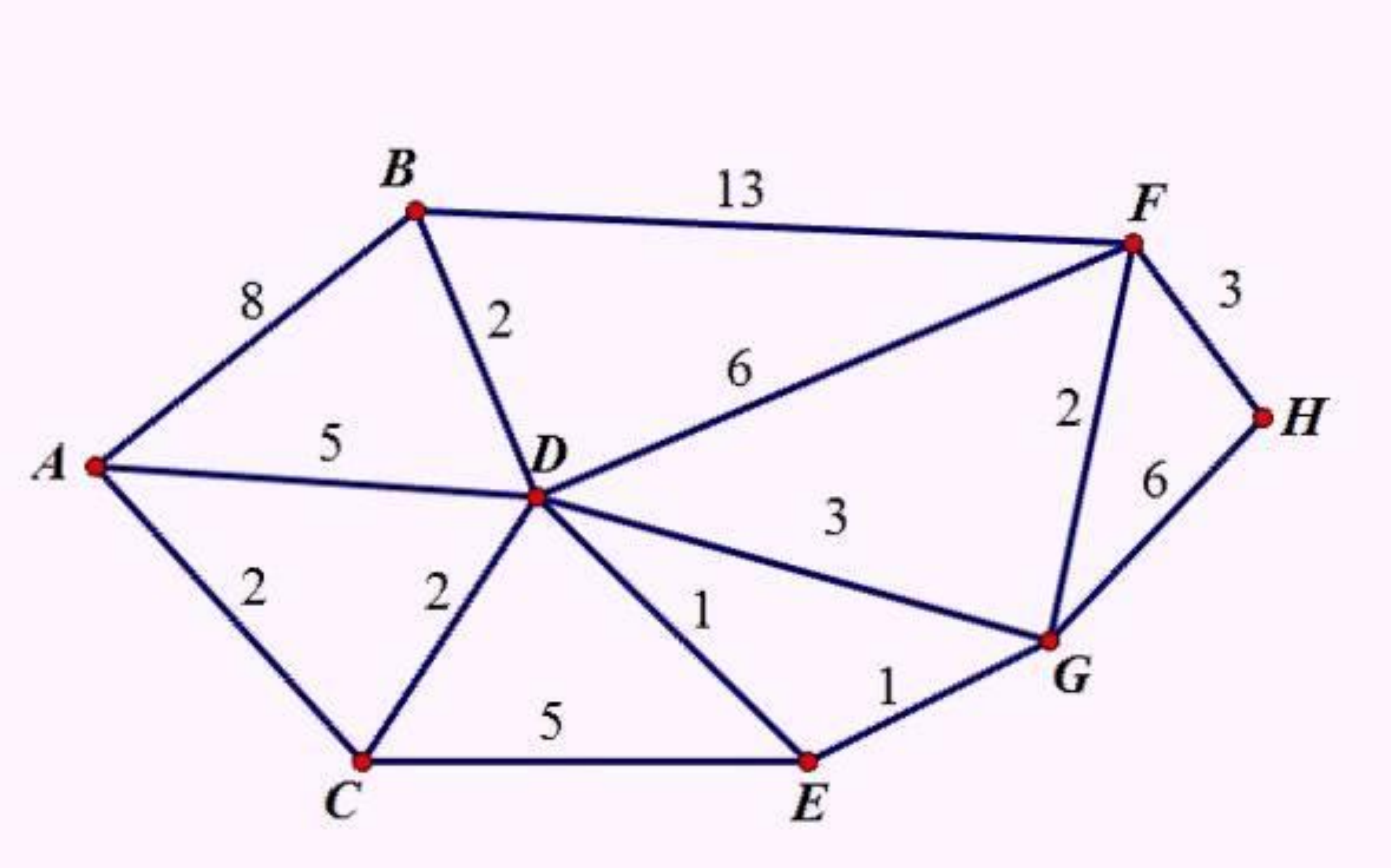
Dijkstra's algorithm

- Main Idea
  - Like BFS, but edges can have **different costs**

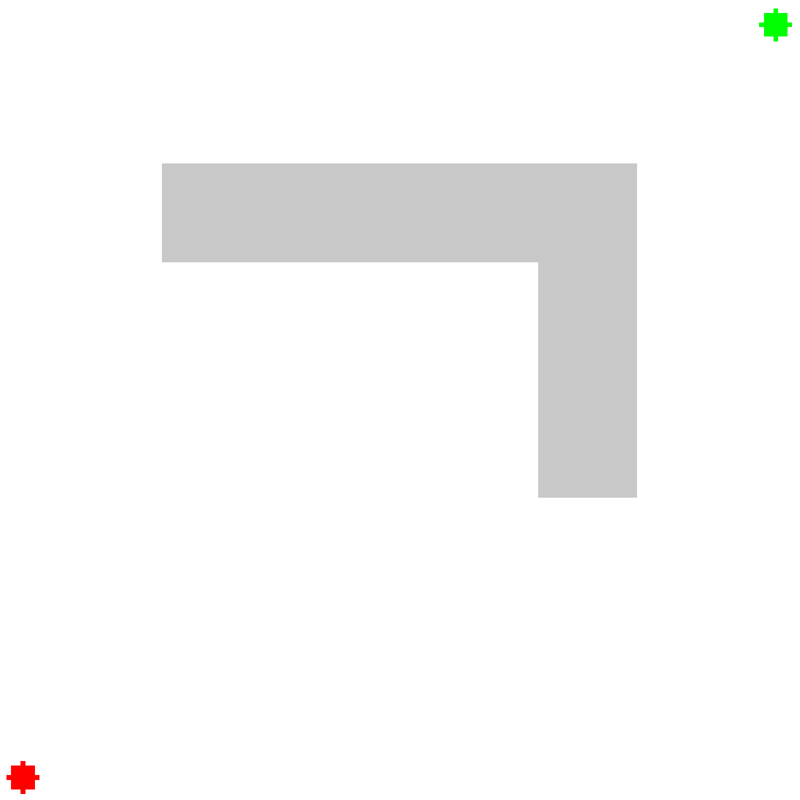
- Open list
  - ***priority queue*** → Nodes are sorted according to g(x), the minimum current cost-to-come to x

- g(x) for each node is updated during the search
  - keep a list lowest current cost-to-come to all nodes

Dijkstra's algorithm



Dijkstra's algorithm

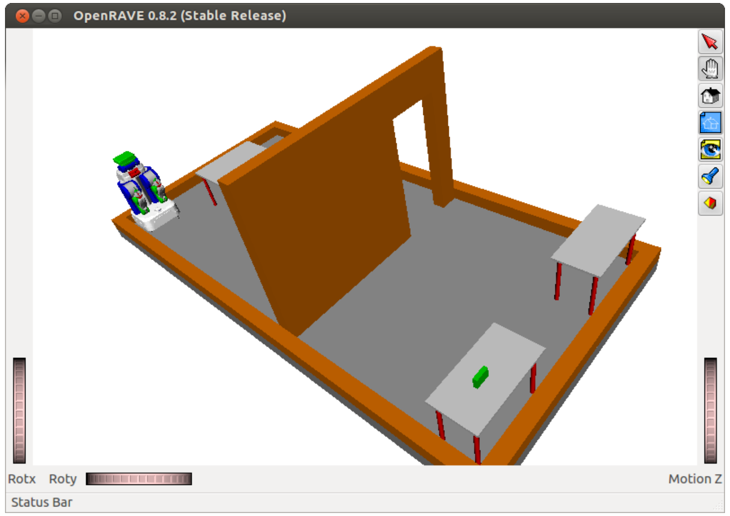


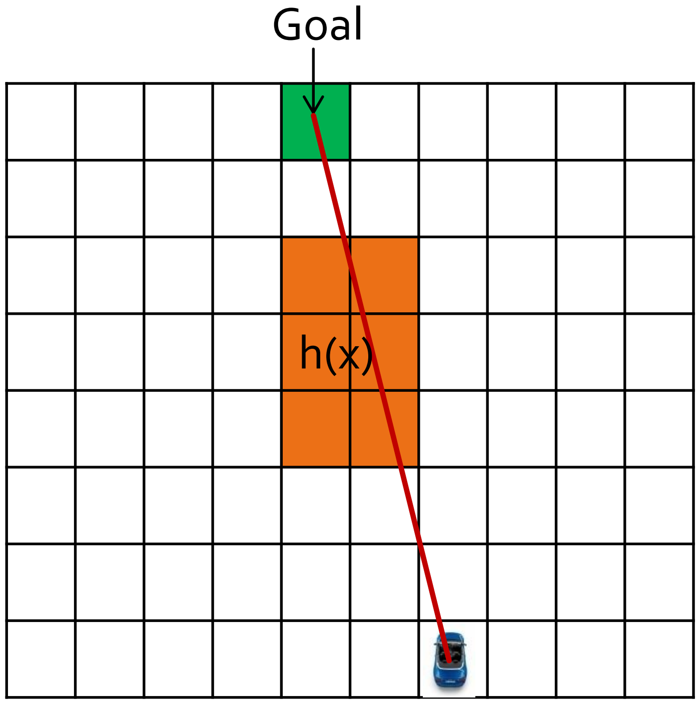
Best-first Search

- Main idea
  - Heuristic function h(x) to estimate each node's distance to goal
  - Expand node with minimum h(x)
- Open list
  - ***priority queue*** → Nodes are sorted according to h(x)

Best-first Search

- Result
  - Works great if **heuristic is a good** estimate
- Guarantee to find the least-cost path?
  - No



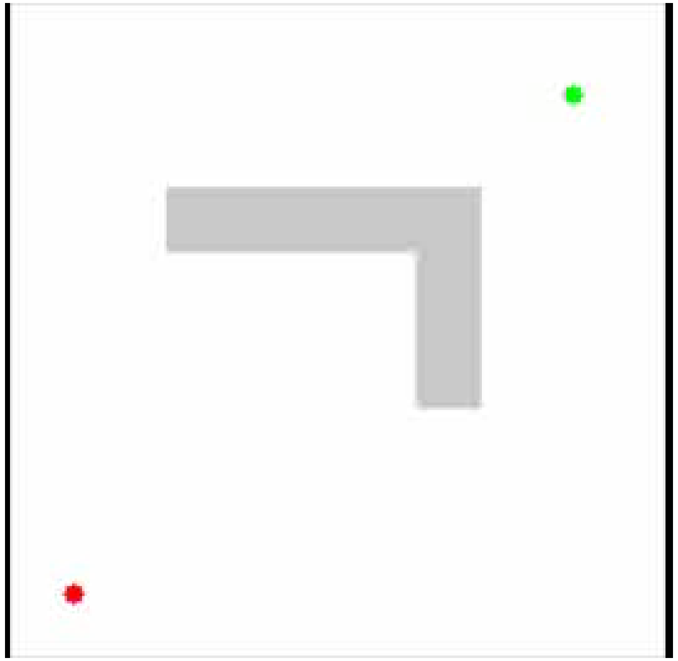


Example of Best-first Search – A*

- Main idea:
  - Select nodes based on cost-to-come and heuristic:

    $f(x) = g(x) + h(x)$

- Open list?
  - *priority queue*
  - Nodes are sorted according to f(x)



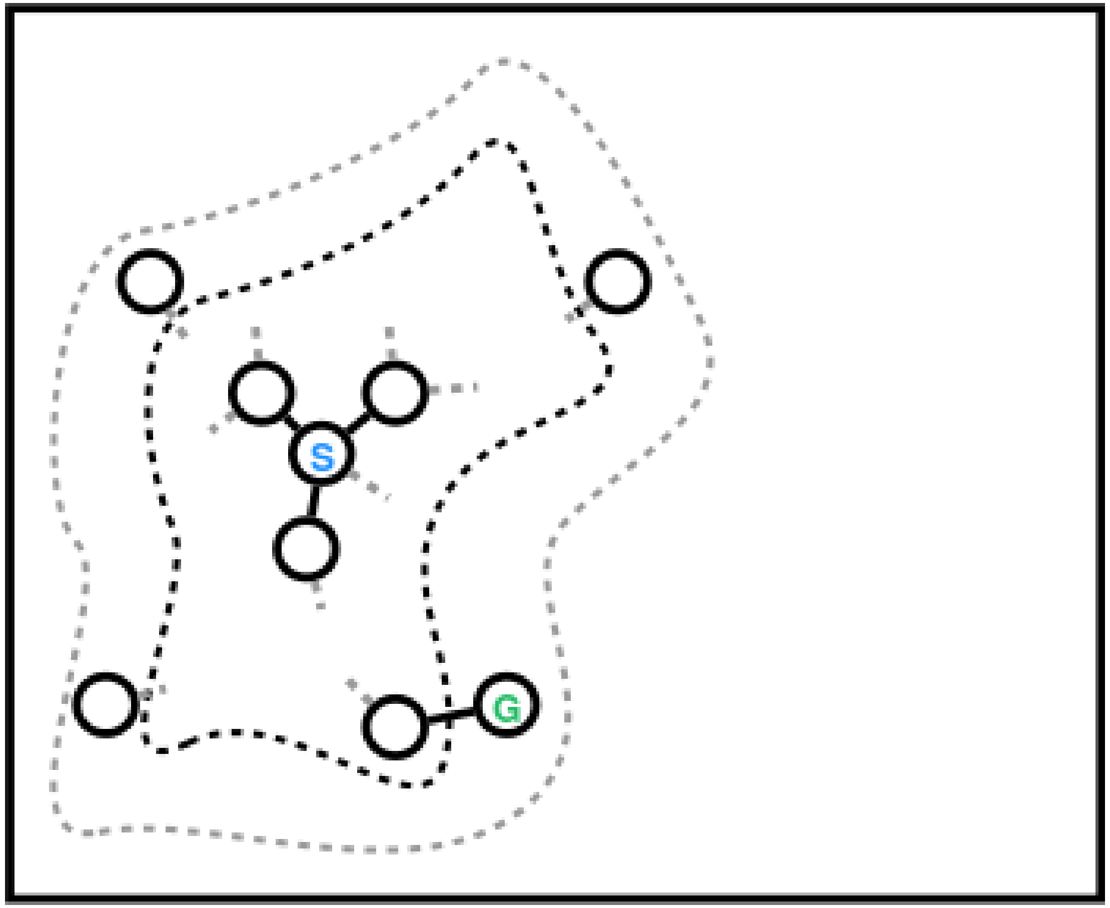
Admissibility

- h(x) must *never overestimate* the true cost-to-come
  - h(x) < h*(x), where h*(x) is the true cost
  - h(x) > 0 (so h(G) = 0 for goals G)
- If h(x) is *admissible*, A* will find the least-cost path!
- "Inflating" the heuristic
  - Faster search
  - Least-cost path is not guaranteed

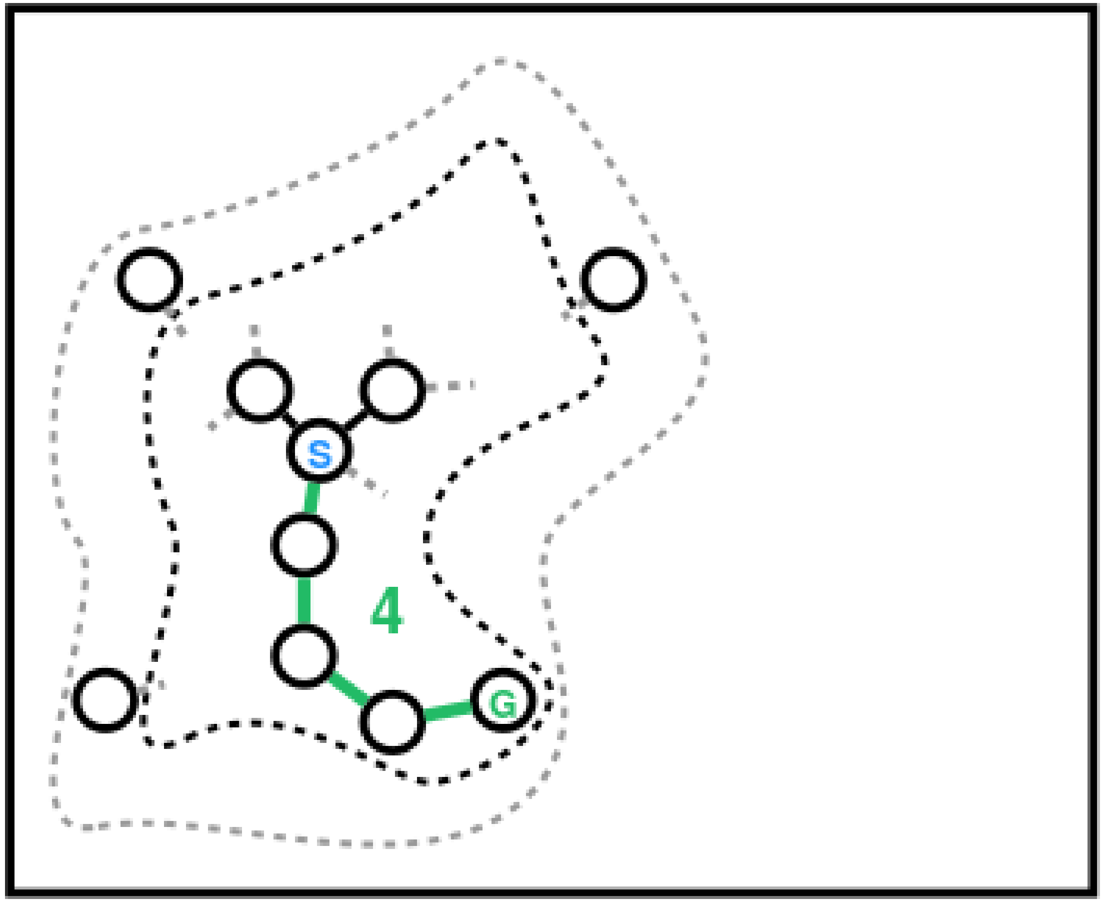
A* Optimality Proof

- Assumption
  - Heuristic is admissible
  - The path found by A* is sub-optimal
- How to prove?
  - Proof by contradiction

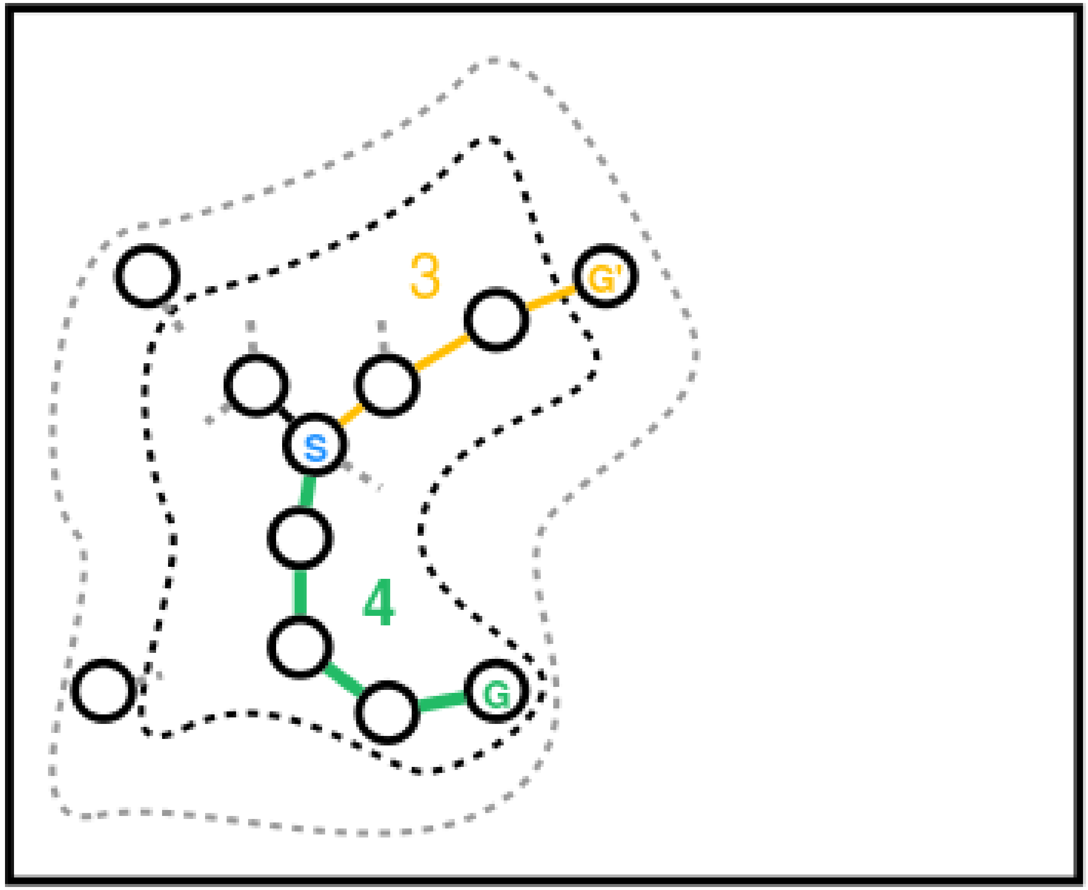
A* Optimality Proof



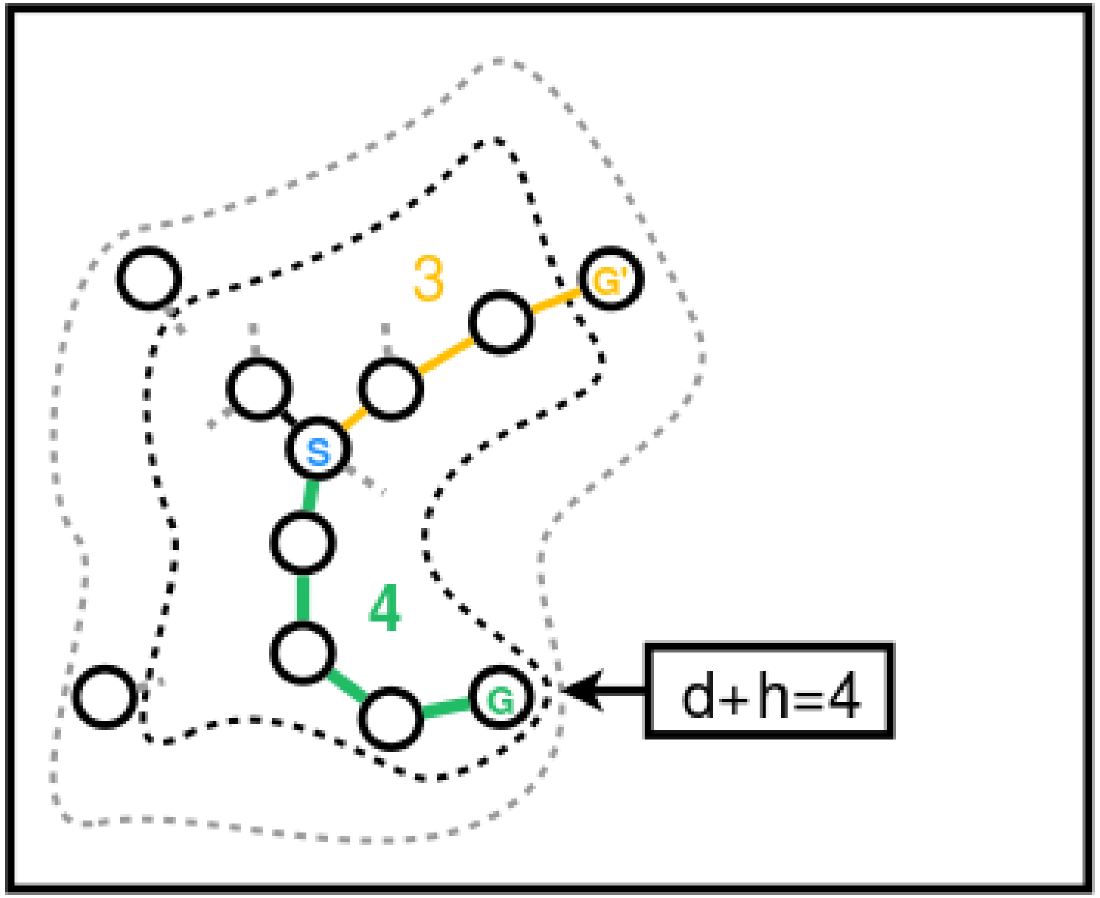
A* Optimality Proof



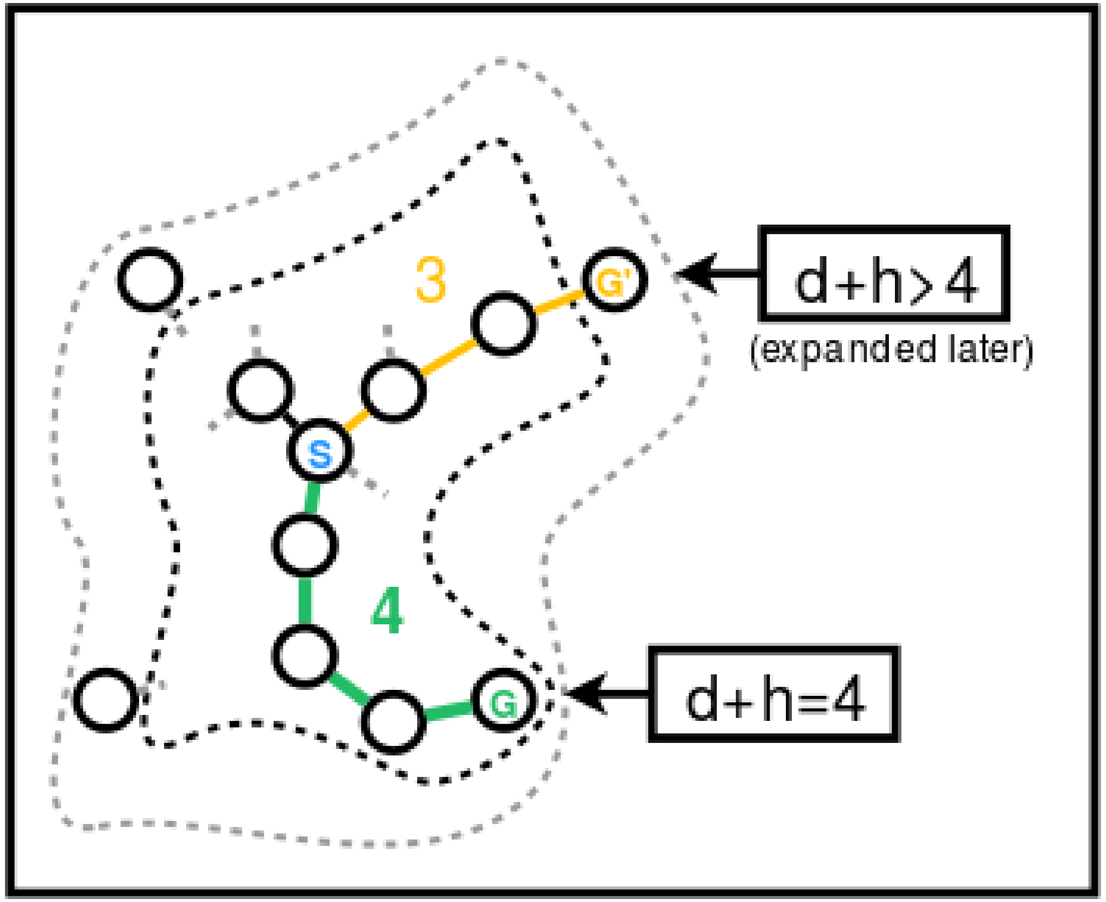
A* Optimality Proof



A* Optimality Proof

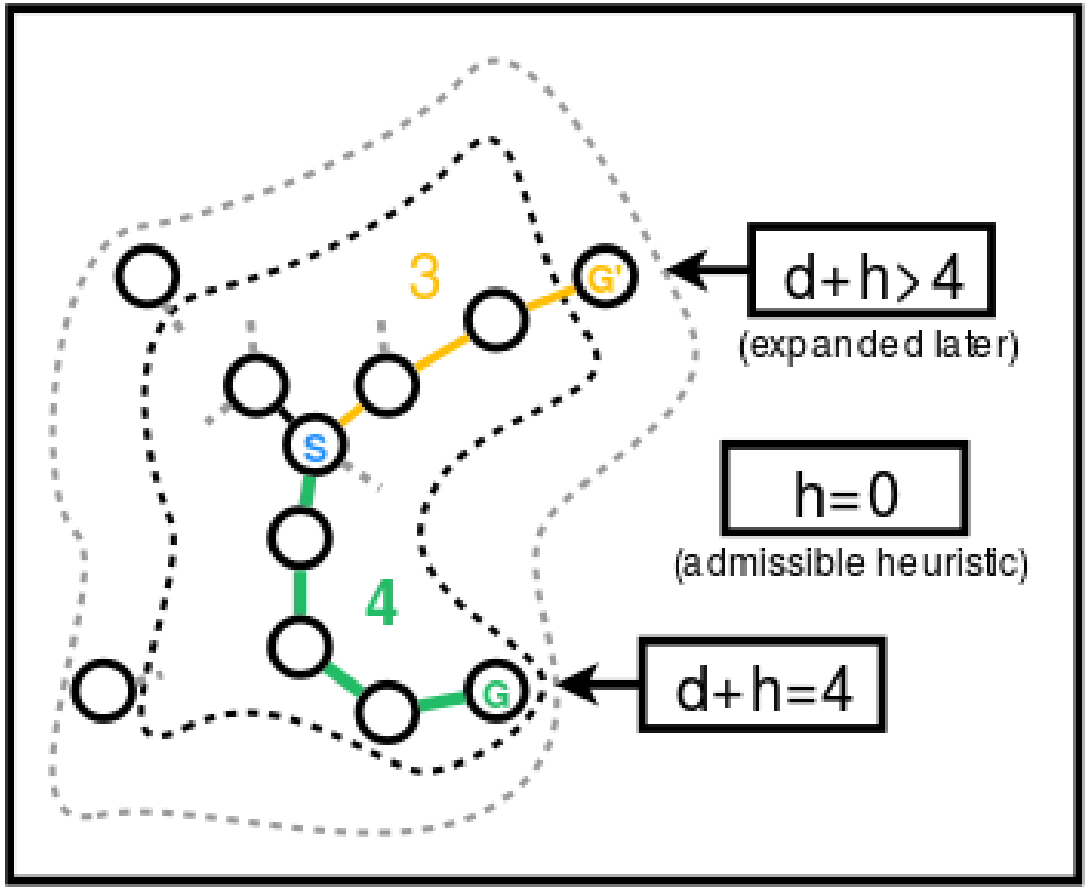


A* Optimality Proof

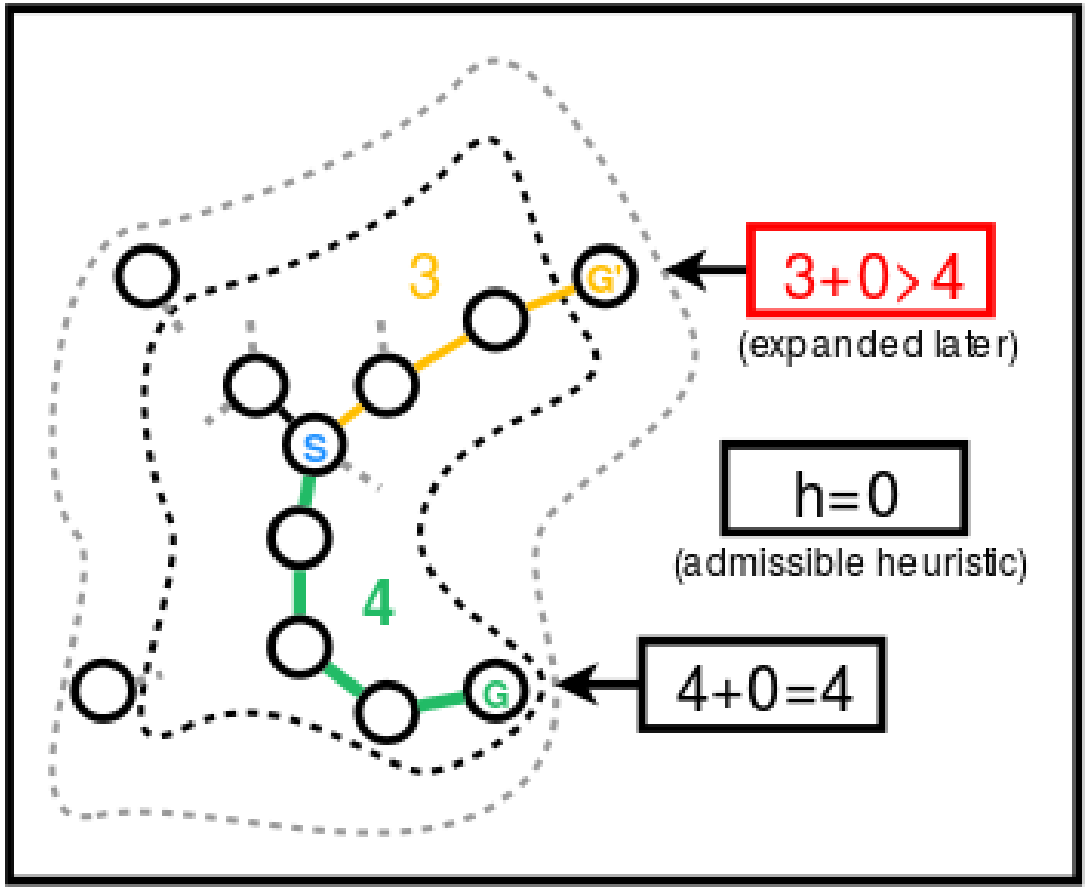


First expand nodes with smaller heuristics

A* Optimality Proof



A* Optimality Proof



Discussion

- If you set h(x) = 0 for all x, A* is equivalent to … ?
- If you set g(x) = 0 for all x, and h(x) = depth of x, A* is equivalent to …?
- In the worst case, what percentage of nodes will A* explore?

A* variants

- Other search
  - D*, ARA, ANA*, R*, ...
- Applications
  - Applications to Search-based motion planning

Assignment – Due Feb 9 by noon

- Individual literature review on discrete planning
  - An advanced search algorithm
  - Applications to motion planning

- Next lecture – Select multiple student talk
  - 10 min talk + 5 min interactive discussion
  - About 10 slides, with notes
  - Double rewards

Reference

- [1] Choudhury, Shushman, Christopher M. Dellin, and Siddhartha S. Srinivasa. "Pareto-optimal search over configuration space beliefs for anytime motion planning." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2016.

- [2] Dantam, Neil T., Zachary K. Kingston, Swarat Chaudhuri, and Lydia E. Kavraki. "Incremental Task and Motion Planning: A Constraint-Based Approach." In Robotics: Science and Systems, pp. 1-6. 2016.

- [3] González, David, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. "A review of motion planning techniques for automated vehicles." IEEE Transactions on Intelligent Transportation Systems 17, no. 4 (2016): 1135-1145.

End