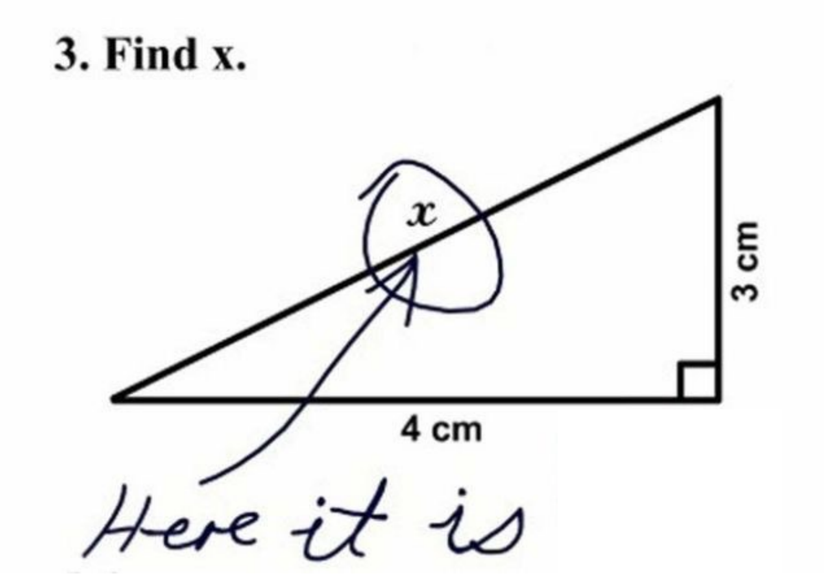


AI as Search

CS B551
Fall 2022

Announcements

- Canvas, Q&A Community, Slack, etc.
- Activity posted on Canvas – due Monday!
- Assignment 0 coming soon!
  - Practice with searching, and with Python.
  - Lots of online resources to learn Python: Google Code, CodeAcademy, many, many tutorials, etc.

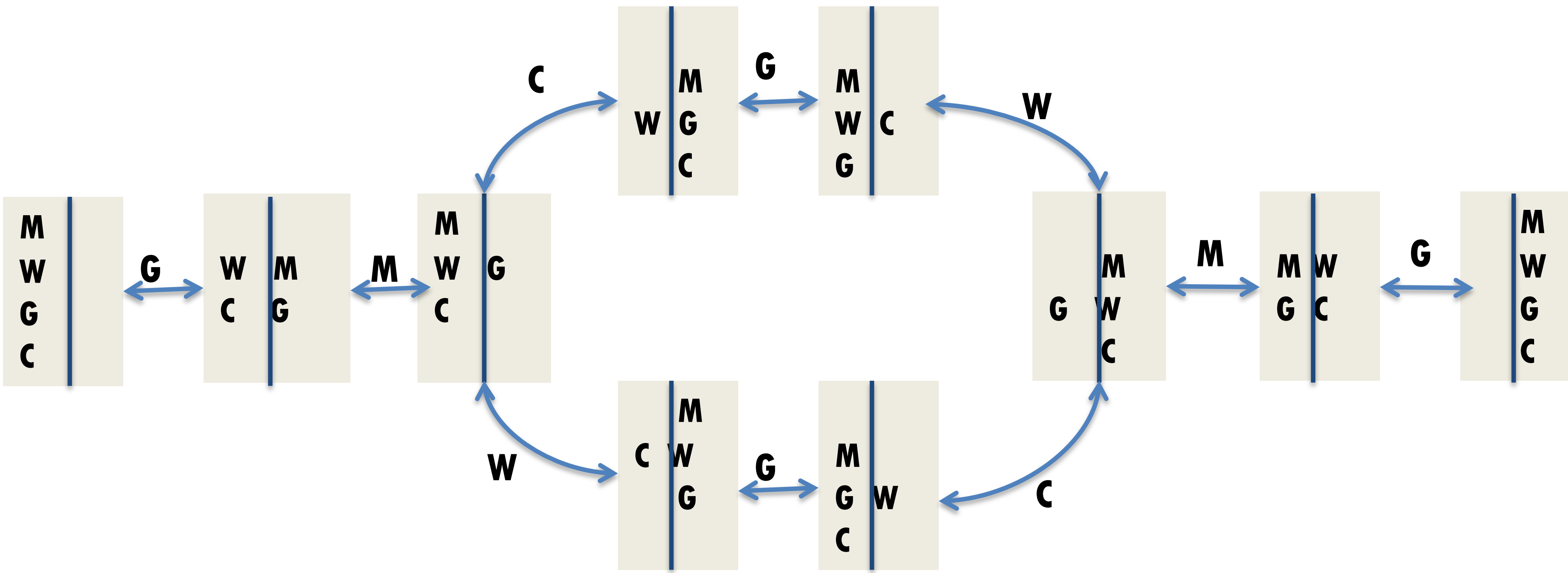
An old puzzle…

Kordemsky: The Moscow Puzzles (Dover)

Puzzles and games have long been considered a challenge for human intelligence:

- Chess in Persia and India ~4000 years ago
- Checkers in 3600-year-old Egyptian paintings
- Go in China over 3000 years ago

A representation of the problem



State space represented as a graph

M: Man goes alone
W: Man takes wolf
G: Man takes goat
C: Man takes cabbage

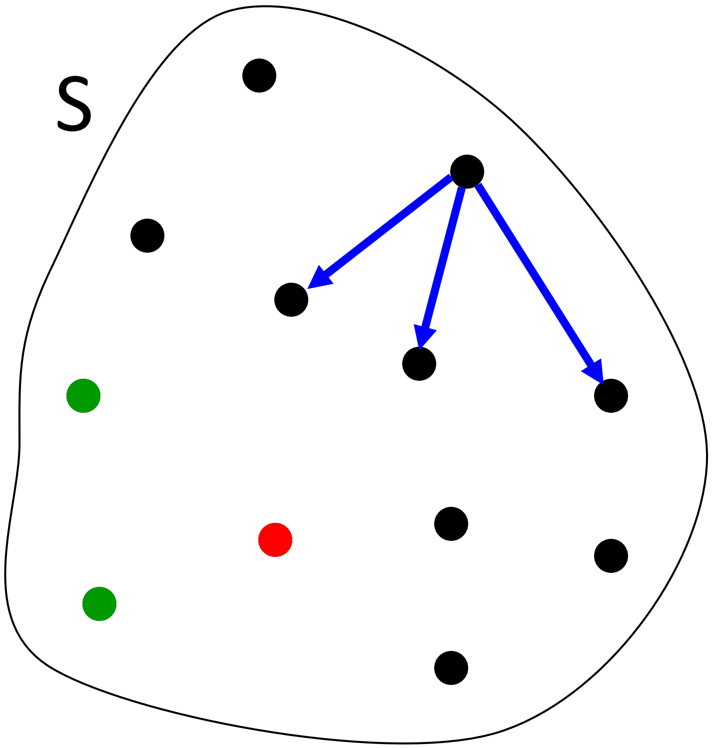
Exploring Choices

- Problems that seem to require intelligence usually require exploring multiple choices.
- Search:
  – A systematic way of exploring choices.
  – The process of looking for a sequence of actions that reaches the goal.

These abstractions have 5 parts:

1. Set of states S
2. Initial state s_0
3. A successor function SUCC: S → 2^S that encodes possible transitions of the system. A successor is any state reachable from a given state by applying a single action.
4. Set of goal states
5. A cost function that calculates how "expensive" a successor is

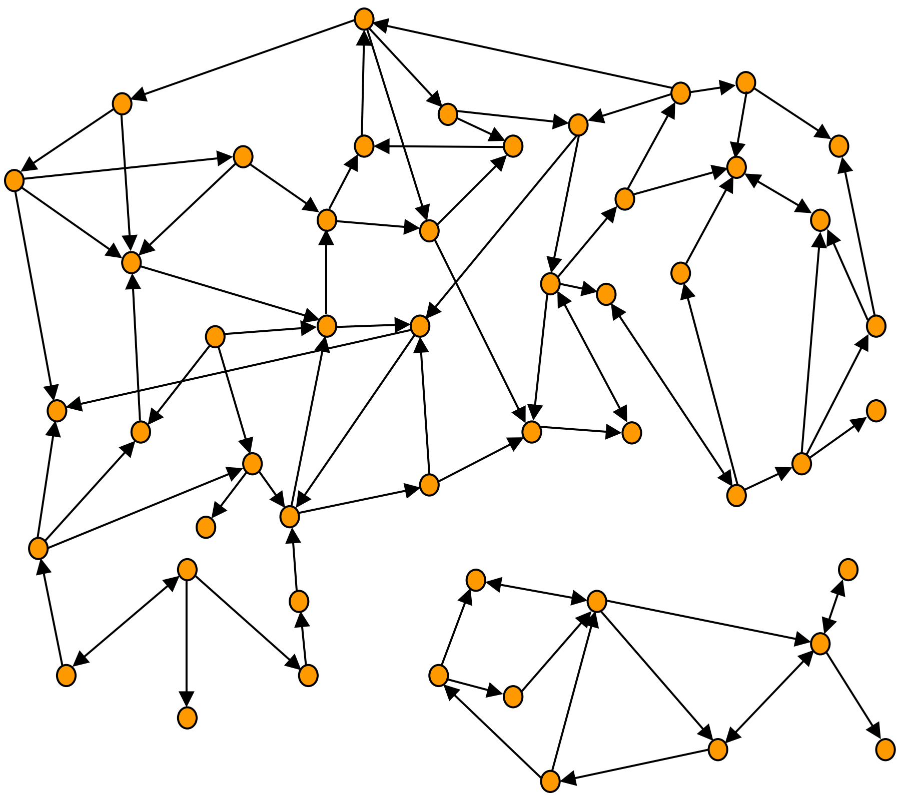
Defining a Search Problem



- State space S
- Successor function:
  $x \in S \rightarrow \text{SUCC}(x) \in 2^S$
- Initial state s_0
- Goal test:
  $x \in S \rightarrow \text{GOAL?}(x) = T \text{ or } F$
- Cost

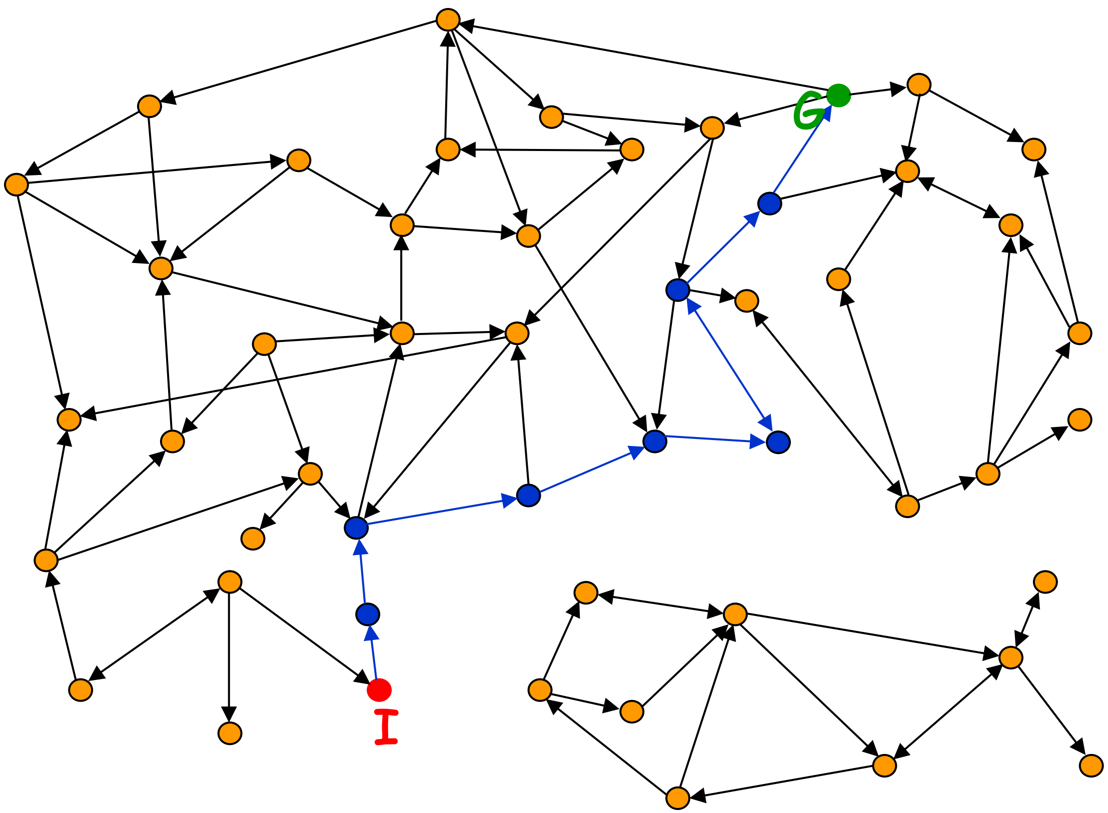
State Graph

- Each state is represented by a distinct node
- An edge connects a node s to a node s' if s' $\in$ SUCC(s)
- The state graph may contain more than one connected component



Solution to the Search Problem

- A solution is a path connecting the initial node to a goal node (any one)
- The cost of a path is the sum of the edge costs along this path
- An optimal solution is a solution path of minimum cost
- There might be no solution !



Example: 8-Puzzle

8	2	
3	4	7
5	1	6

Initial state

1	2	3
4	5	6
7	8	

Goal state

Example: 8-Puzzle

8	2	
3	4	7
5	1	6

Initial state

1	2	3
4	5	6
7	8	

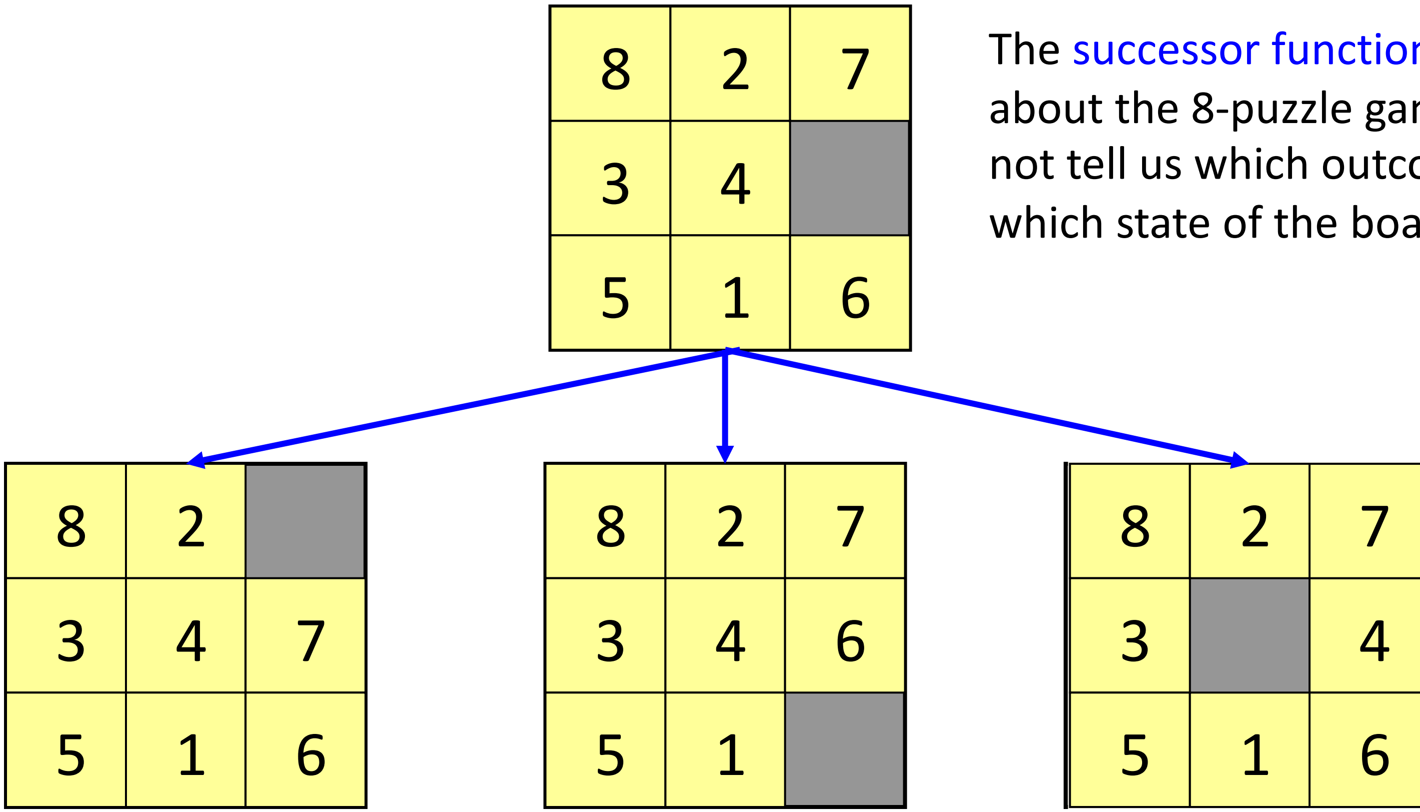
Goal state

State: Any arrangement of 8 numbered tiles and an empty tile on a 3x3 board
Successor function: given by available actions (sliding tiles) L, R, U, D.
Cost: How many moves were performed

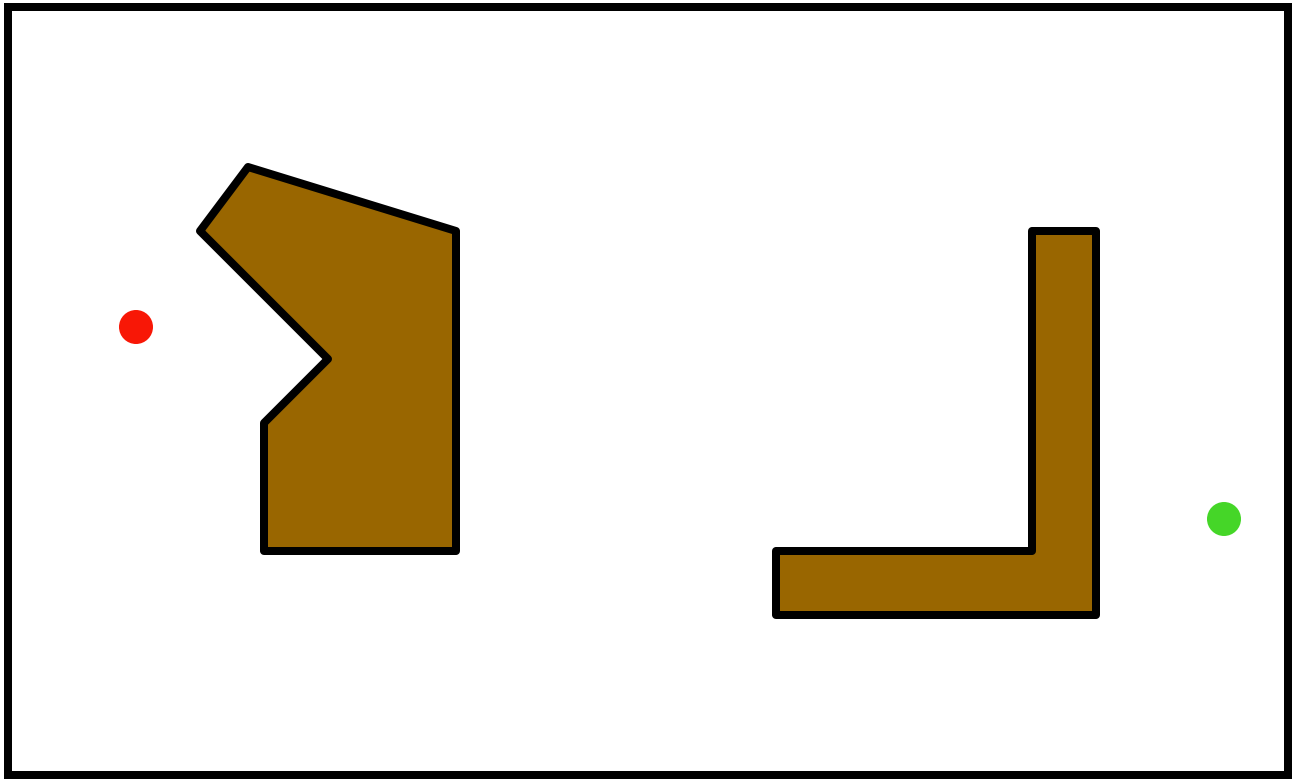
Successor Function: 8-Puzzle

SUCC(state) → subset of states

The successor function is knowledge about the 8-puzzle game, but it does not tell us which outcome to use, nor to which state of the board to apply it.

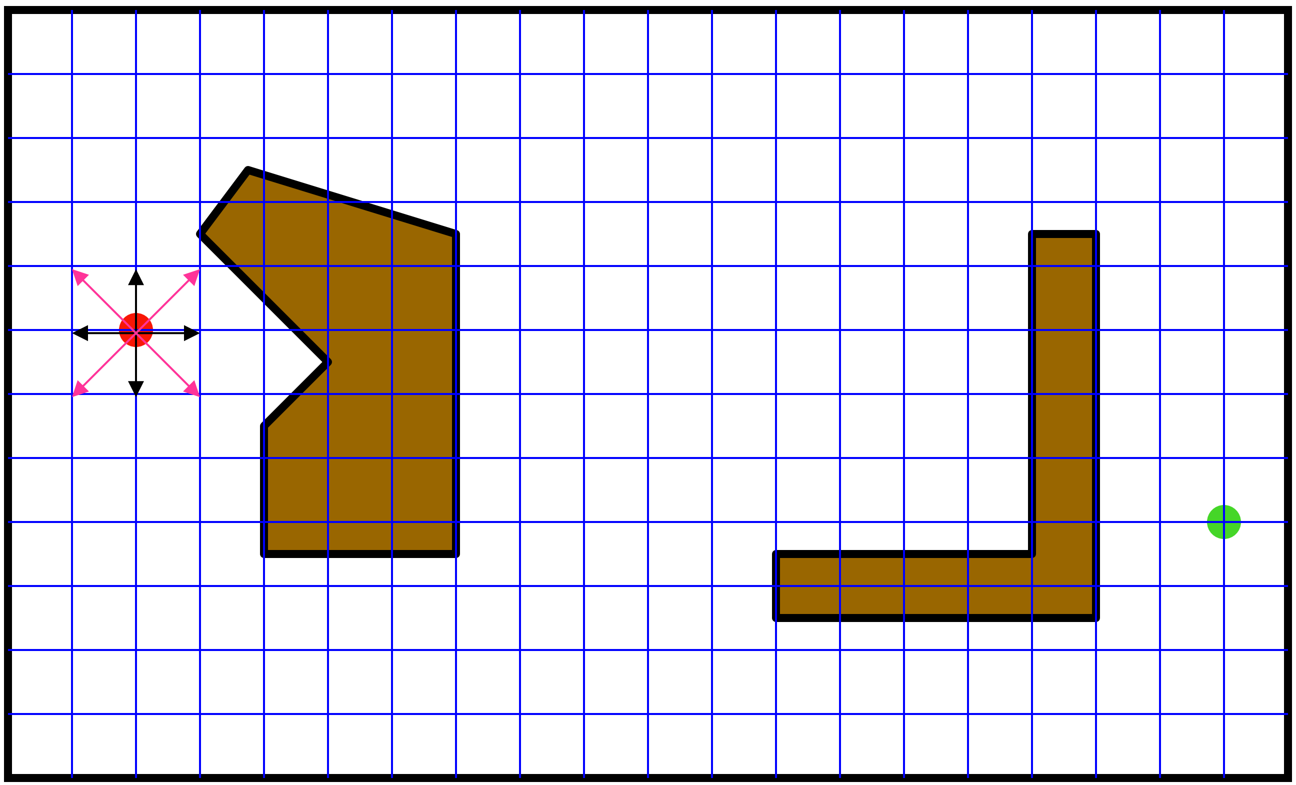


Path Planning



What is the state space?

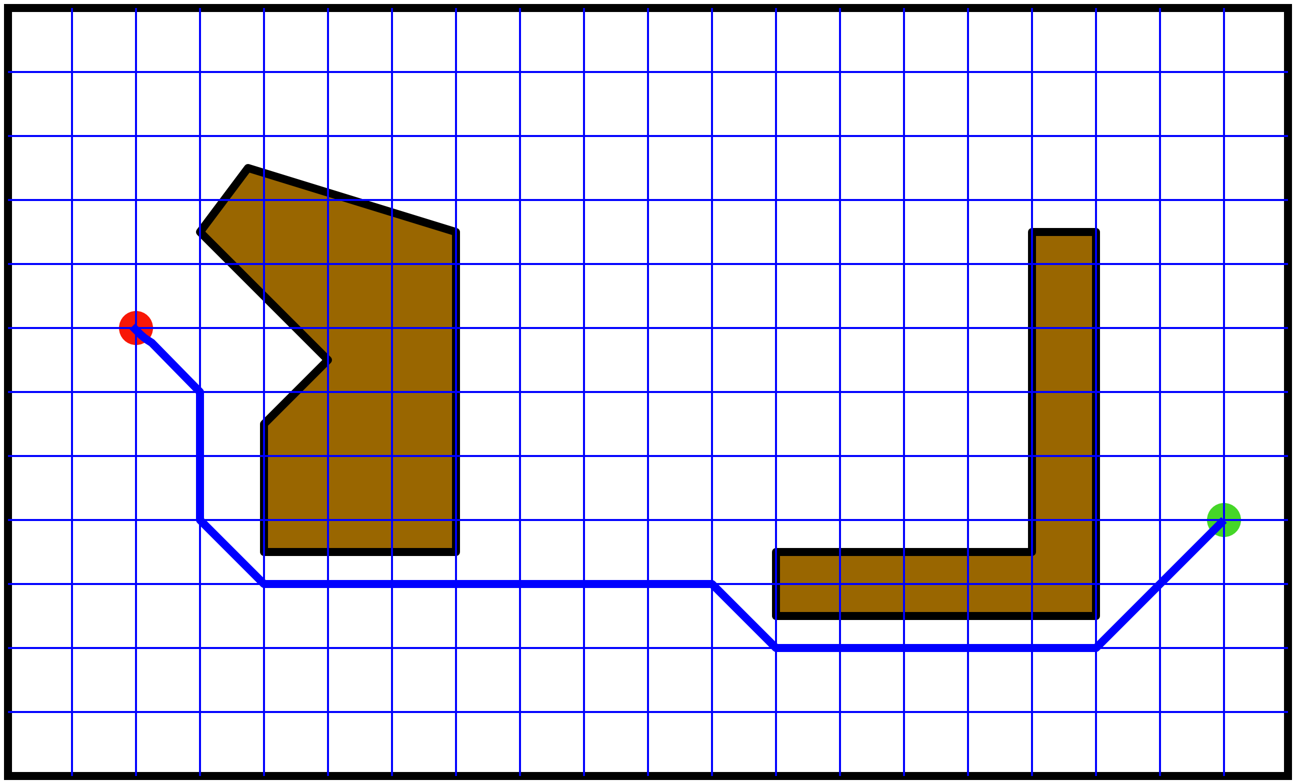
Formulation #1



Cost of one horizontal/vertical step = 1

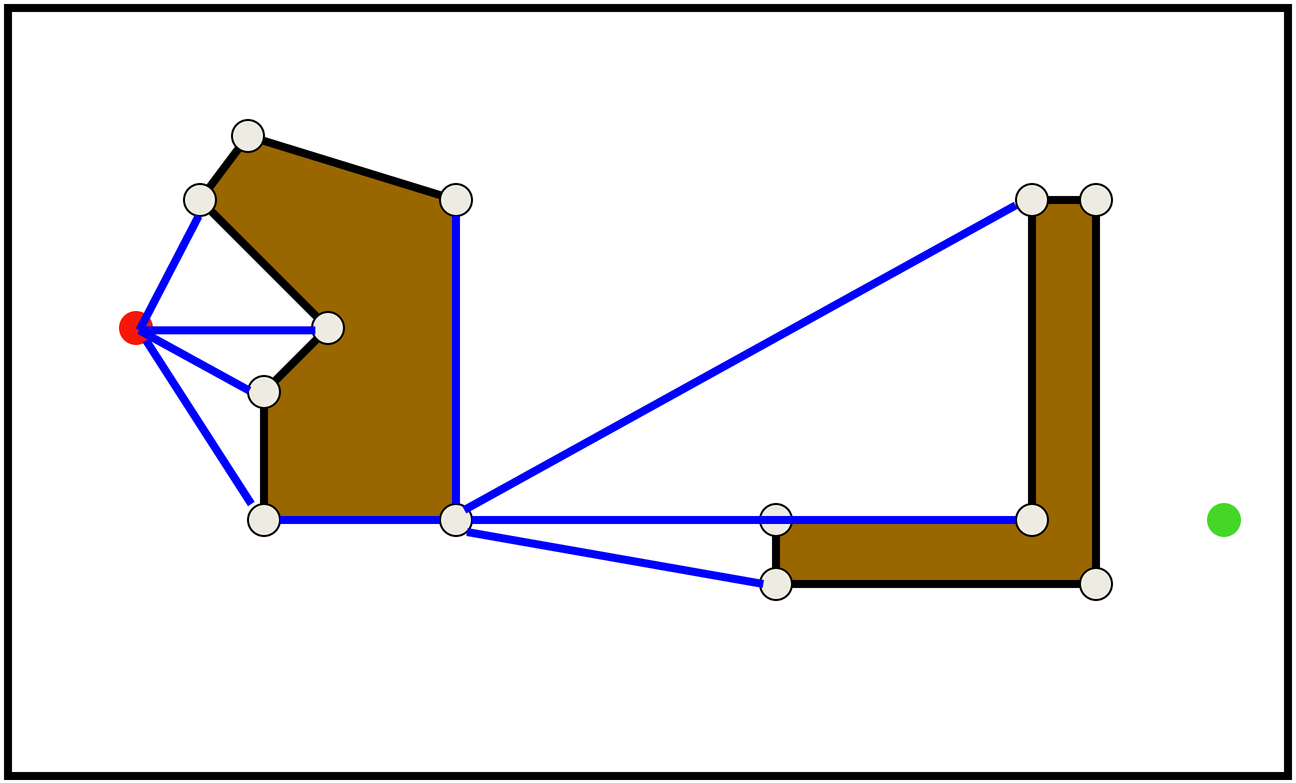
Cost of one diagonal step = $\sqrt{2}$

Optimal Solution



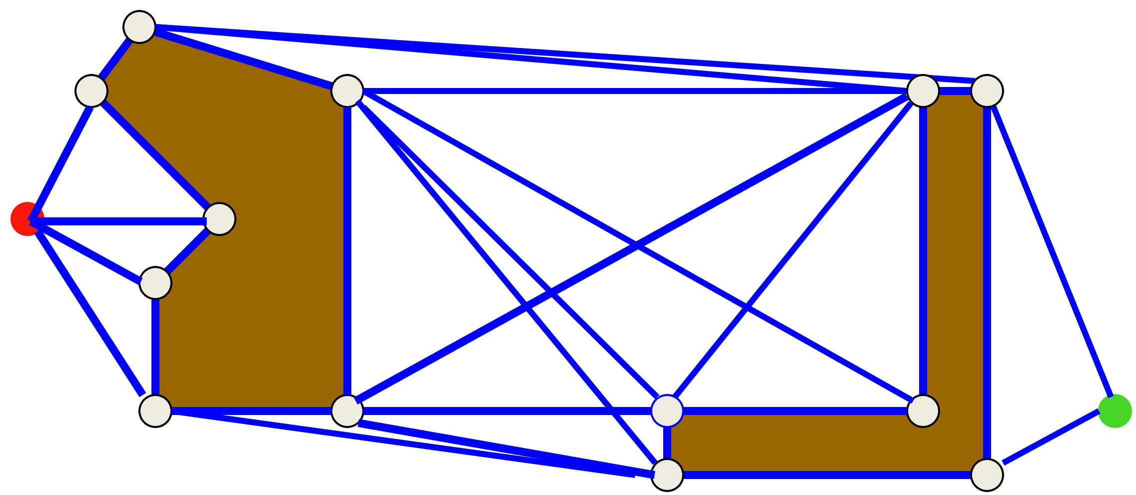
This path is the shortest in the discretized state space, but not in the original continuous space

Formulation #2



Cost of one step: length of segment

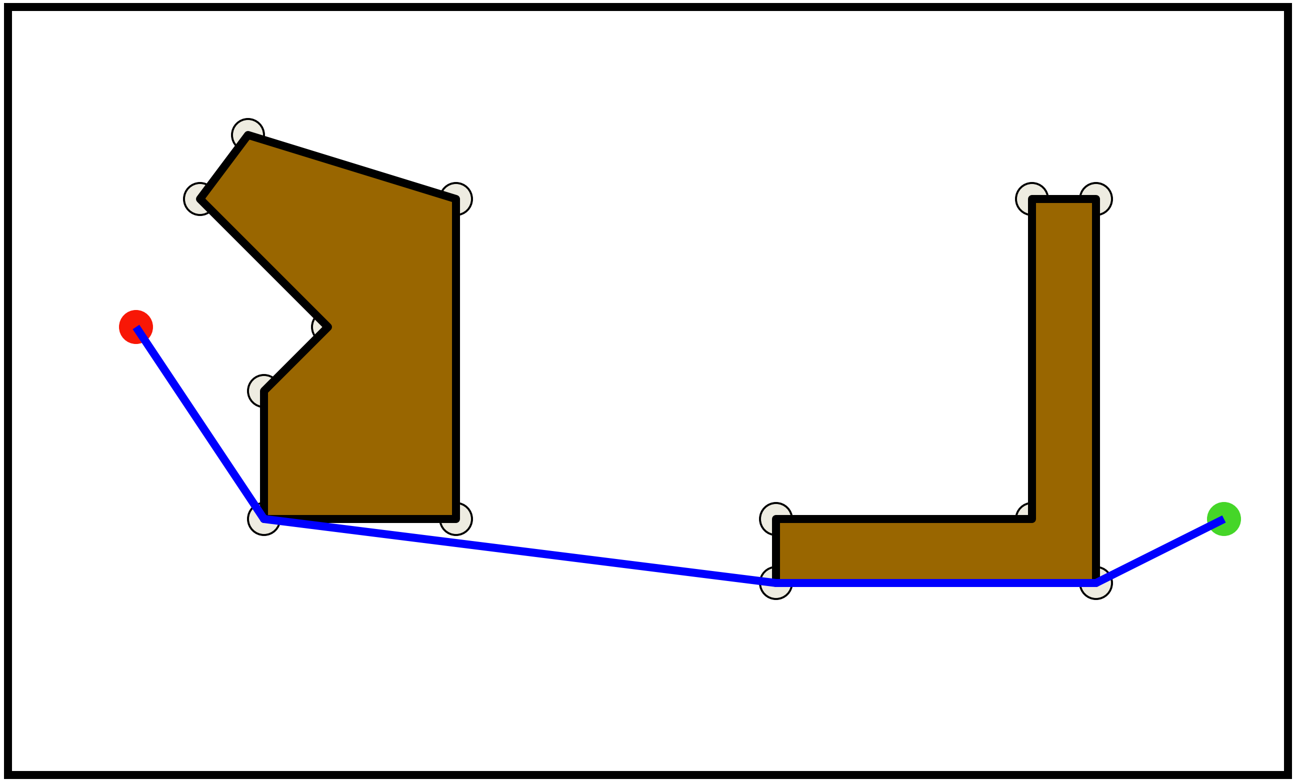
Formulation #2



Visibility graph (any two points “see” each other iff the line segment between them is NOT in the exterior)

Cost of one step: length of segment

Solution Path



The shortest path in this state space is also the shortest in the original continuous space

What is a State?

- **A state does:**
  - Represent all information meaningful to the problem at a given “instant in time” – past, present, or future
  - Exist in an *abstract*, *mathematical* sense
- **A state DOES NOT:**
  - Necessarily exist in the computer’s memory
  - Tell the computer how it arrived at the state
  - Tell the computer how to choose the next state
  - Need to be a unique representation

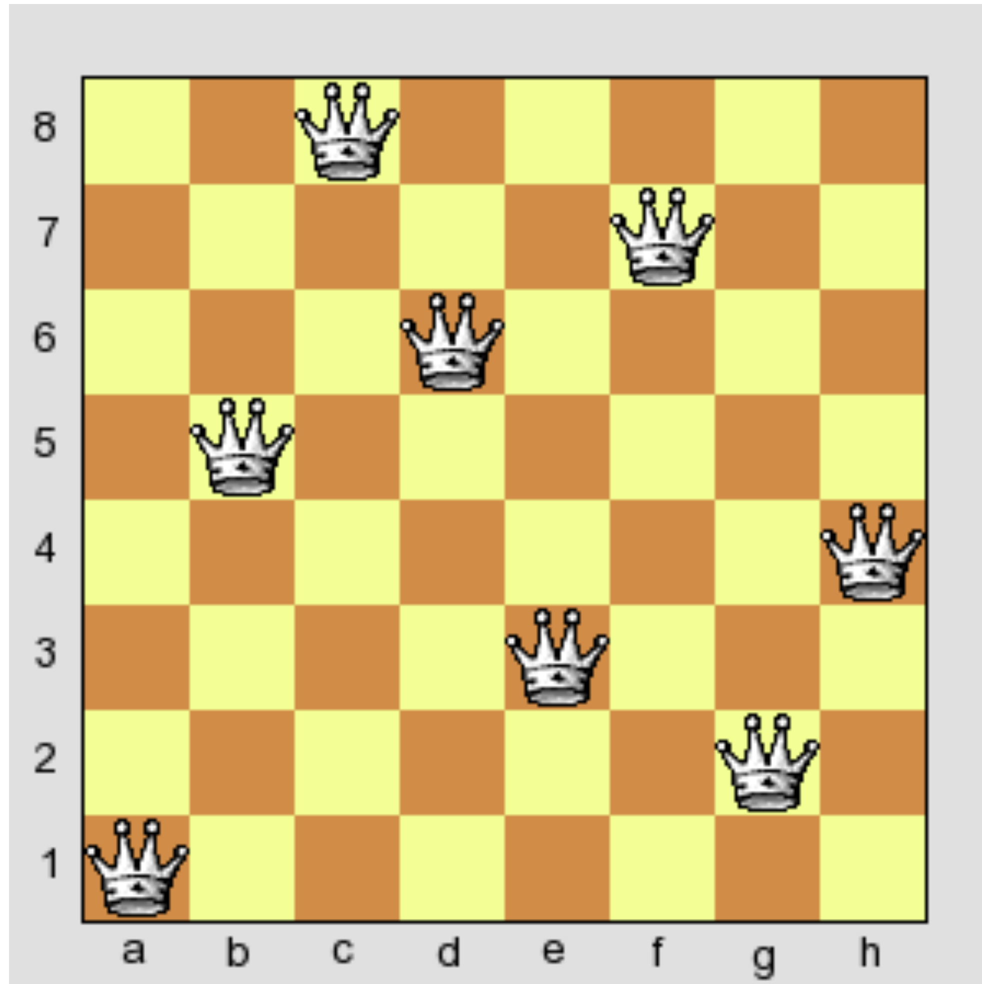
What is a State Space?

- An abstract mathematical object
  - E.g., the set of all permutations of (1,…,8,empty)
- Membership should be trivially testable
  - So S = { s | s is reachable from the start state through transformations of the successor function } is *a bad definition* (not trivially testable).
- Edges should be easily generated

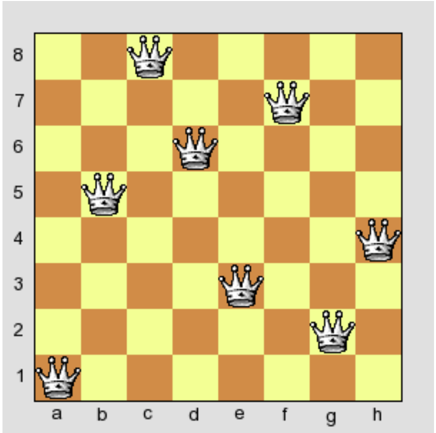
- *Again: the state space does NOT contain information about which edge to take (or not to take) in a given state*

8-Queens Problem

Place 8 queens in a chessboard so that no two queens are in the same row, column, or diagonal.



8-Queens Problem

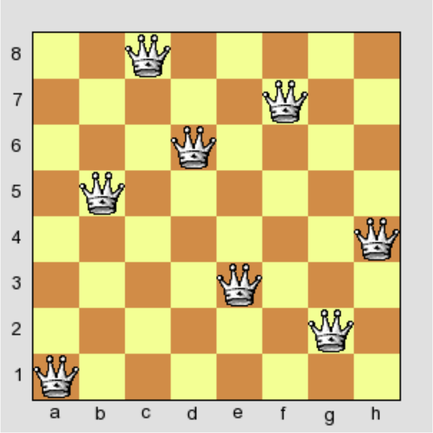


Two main formulations of the problem:

1. **An incremental formulation:**
   - **States**: Any arrangement of 0 to 8 queens on the board is a state.
   - **Initial state**: No queens on the board.
   - **Actions**: Add a queen to any empty square.
   - **Goal test**: 8 queens are on the board, none attacked.

In this formulation, we have $64 \cdot 63 \cdots 57 \approx 1.8 \times 10^{14}$ possible sequences to investigate.

8-Queens Problem



Two main formulations of the problem:
2. Complete-state formulation: prohibit placing a queen in any square that is already attacked

- States: All possible arrangements of n queens ($0 \le n \le 8$), one per column in the leftmost n columns, with no queen attacking another.
- Actions: Add a queen to any square in the leftmost empty column such that it is not attacked by any other queen.
- Goal test: 8 queens are on the board, none attacked.

This formulation reduces the 8-queens state space from 1.8×10^{14} to just 2,057, and solutions are easy to find.

Next class: Search algorithms (ch 3.4 in the book)