COSC76/276 Artificial Intelligence
Fall 2022
Adversarial search

Soroush Vosoughi
Computer Science
Dartmouth College
Soroush@Dartmouth.edu

Reminders

- SA-2 due today at 11:59pm ET (completion based)
- PA-2 due Oct 10th at 11:59pm ET
- All assignment due dates have been postedon Canvas. I may give extensions as needed but will never move due dates forward
- Missing Wednesday's class for Yom Kippur

Grading

- SAs will be graded based on completion, but you will see what you did wrong.
- The source code for PAs will not be released, but I will go over them with you during office hours if you want.

Recap: Search problem

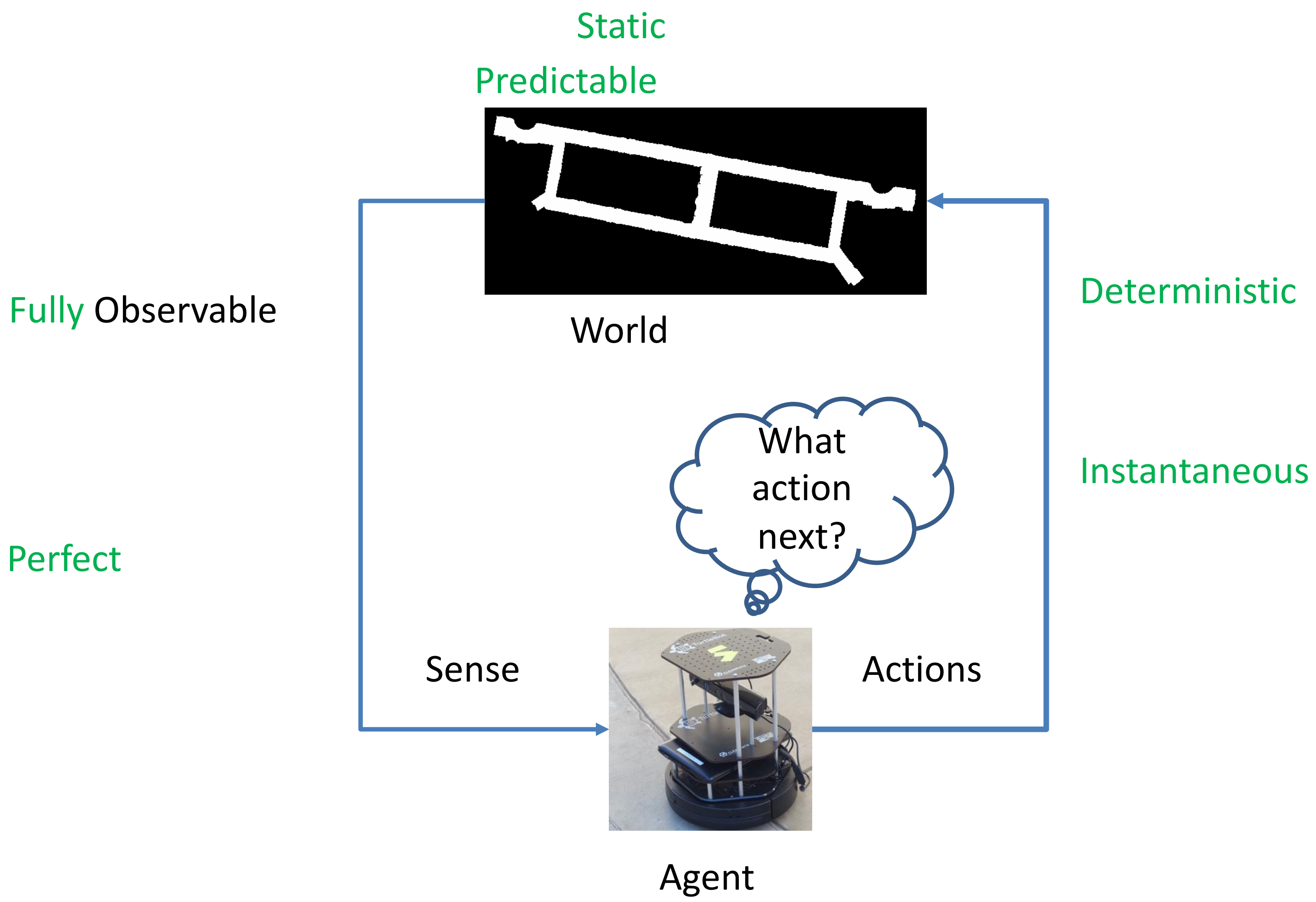
- Uninformed search with cost: UCS
- Informed search
  - Heuristic
  - Greedy and A*

Deriving heuristics

- Most of the work in solving hard search problems optimally is in coming up with admissible heuristics
- Often, admissible heuristics are solutions to *relaxed problems,* where the problem has fewer restrictions, i.e., new actions are available
  - The relaxed problem may have better solutions as there might be shortcuts

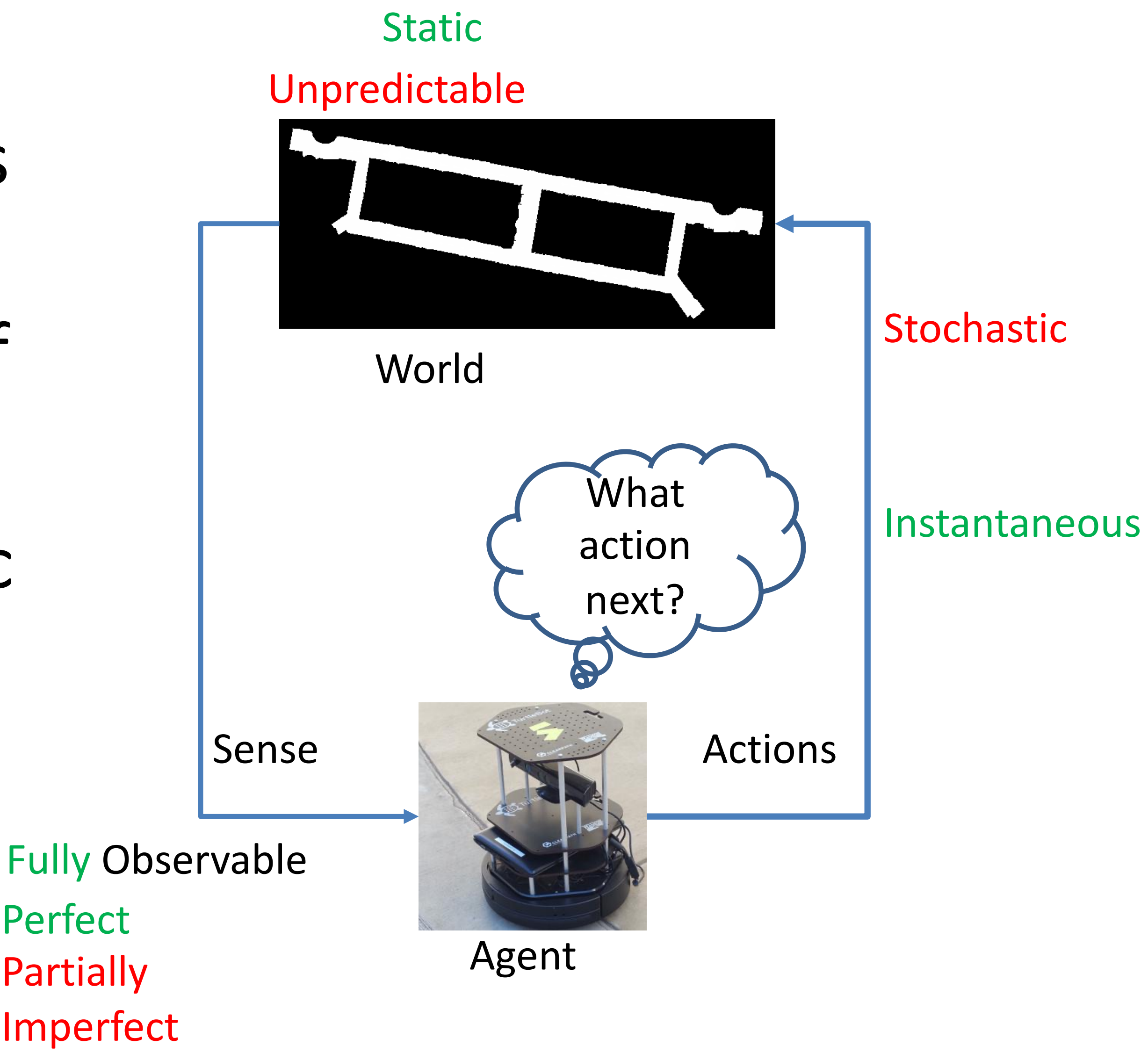
Non-deterministic actions and partial observations

Classical Planning view



Planning views

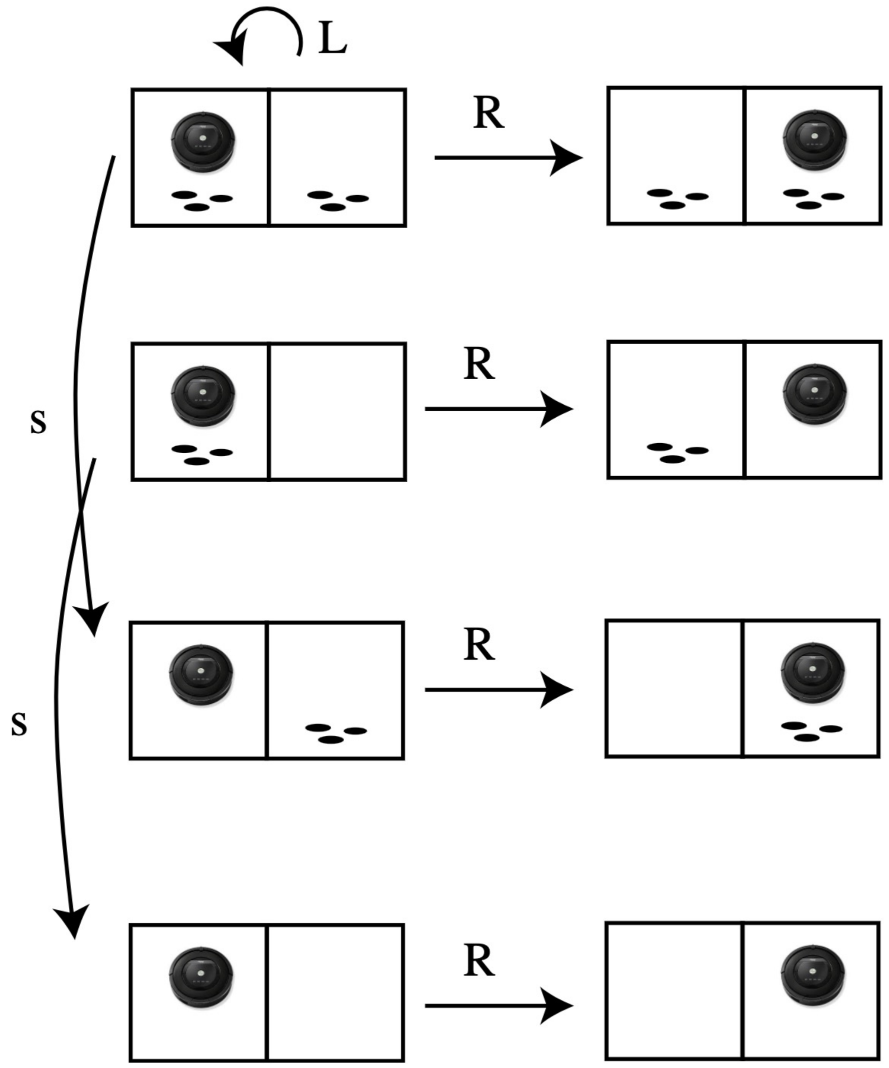
- Different planning views which involve different set of techniques
- E.g., Stochastic planning



Vacuum world: State space

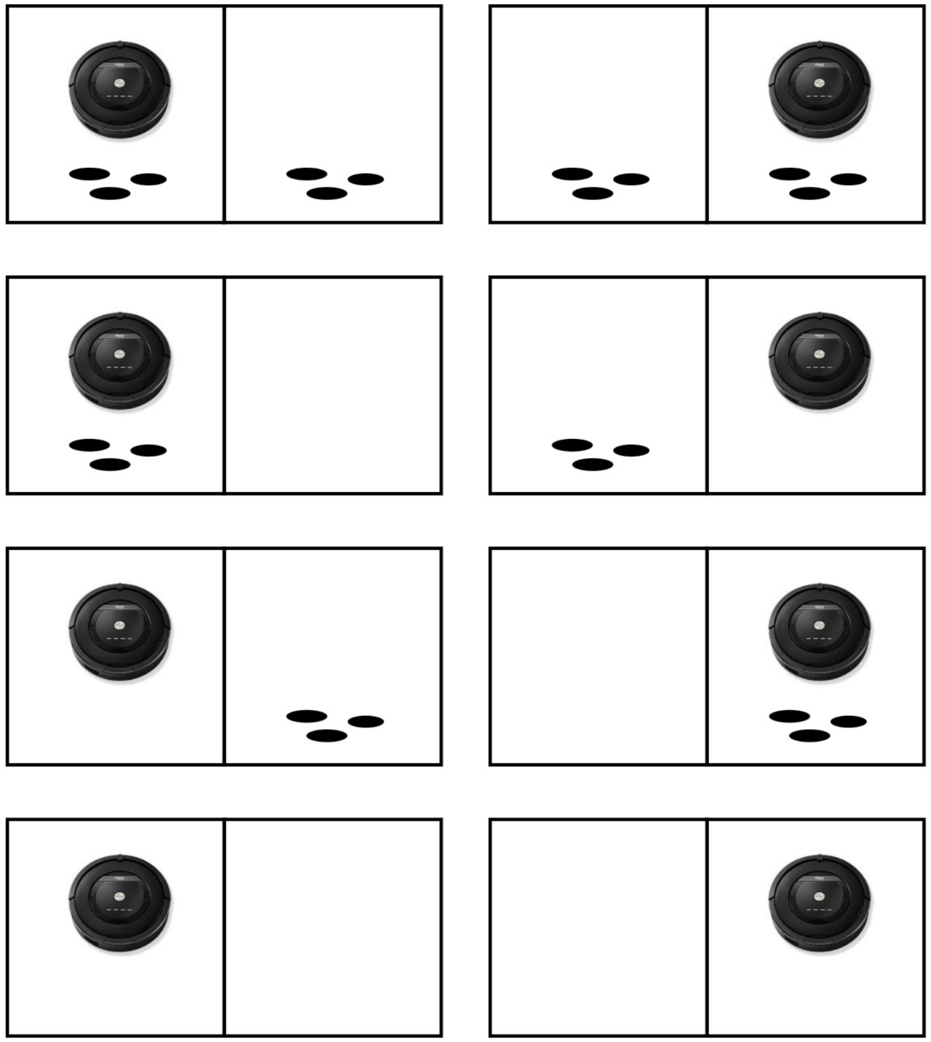
- L go left
- R go right
- S sweep

(not all edges are marked)



Vacuum world

- Clean 2 rooms

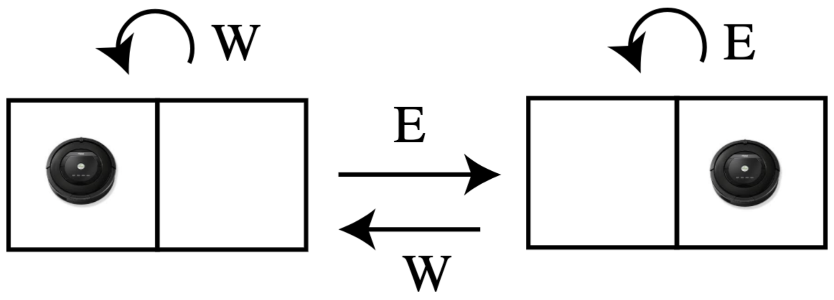


Non-deterministic actions

Sweep action:

- If dirty square, then the action cleans the square and sometimes cleans up dirt in an adjacent square
- If applied to a clean square, it might deposit dirt

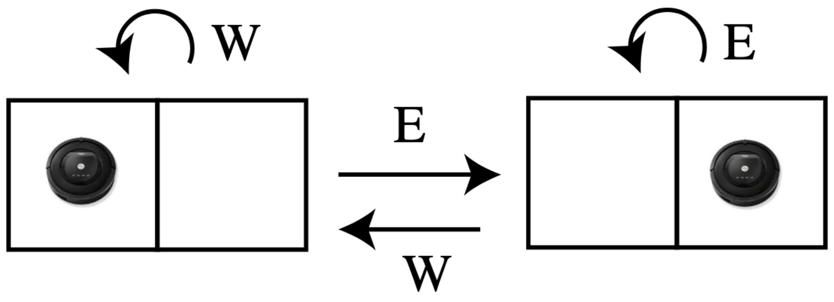
Simplified vacuum world: Partial observations



- Simplified vacuum world:
- The robot is kidnapped (does not know start location).
- The robot is blind (cannot detect location).
- The robot has a compass (can take action reliably).

Simplified vacuum world: Partial observations

What is the sequence of actions that will take the robot to square 0?



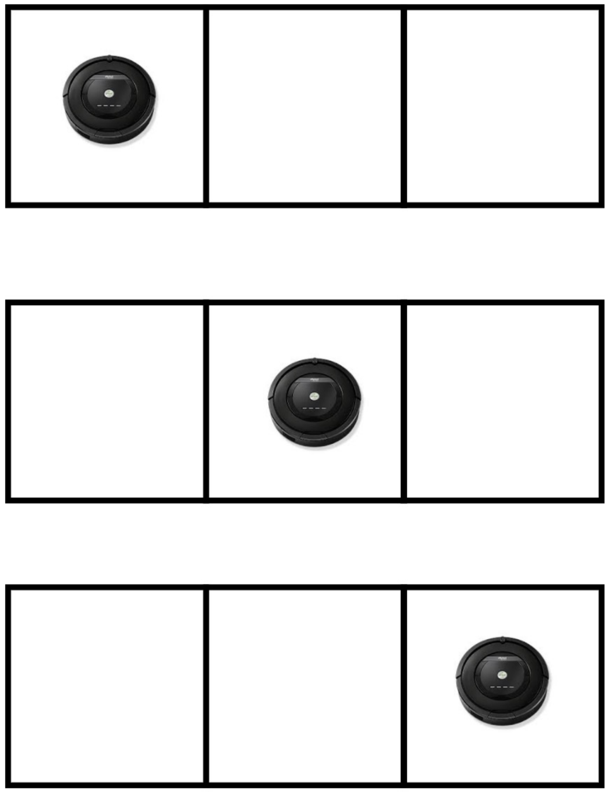
Reasoning:

- Either I'm home (square 0) or I'm not (square 1).
- If I'm in square 1, go West.
- If I'm in square 0, I still have to act, go West.

The sequence of actions **W** brings the robot home.

Simplified vacuum world: Partial observations

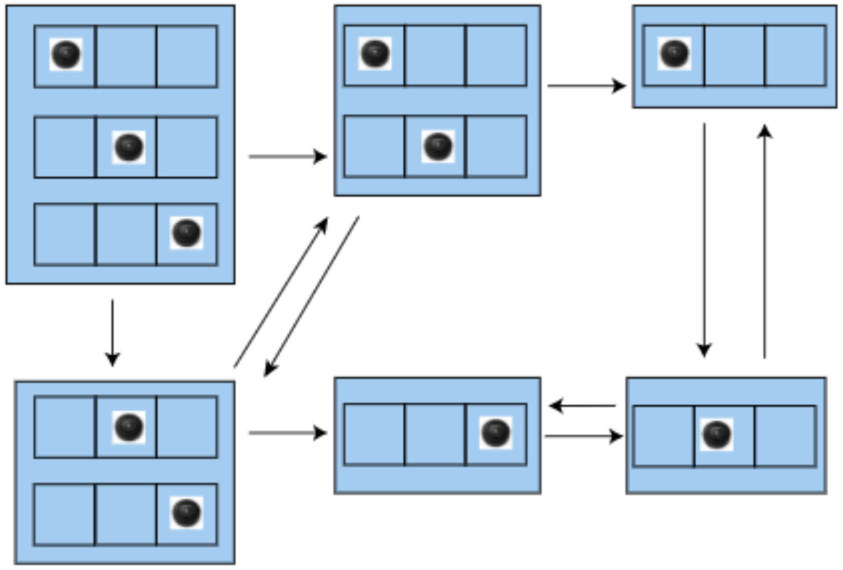
What is the sequence of actions that will take the robot to square 0?



The sequence of actions **WW** brings the robot home.

Partial observations

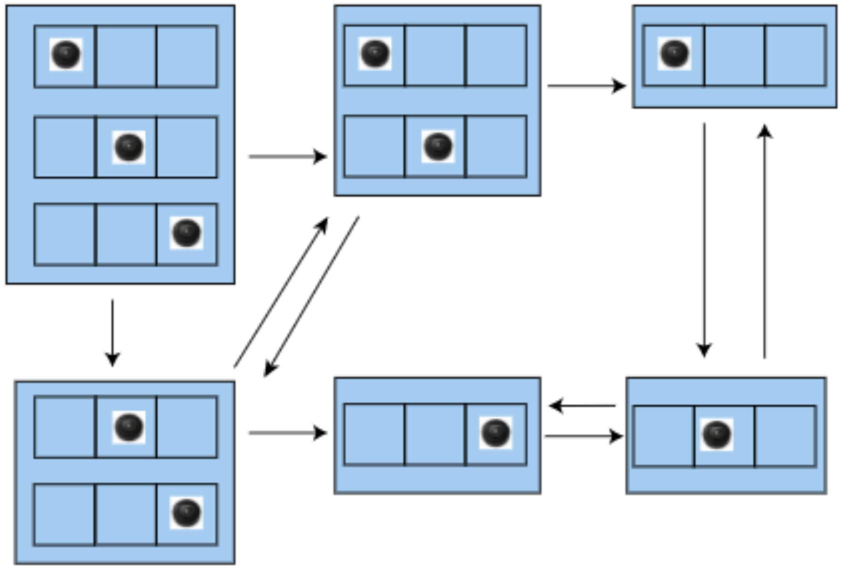
- Belief states: where the agent thinks it might be



What is the upper bound for the size of belief state space?

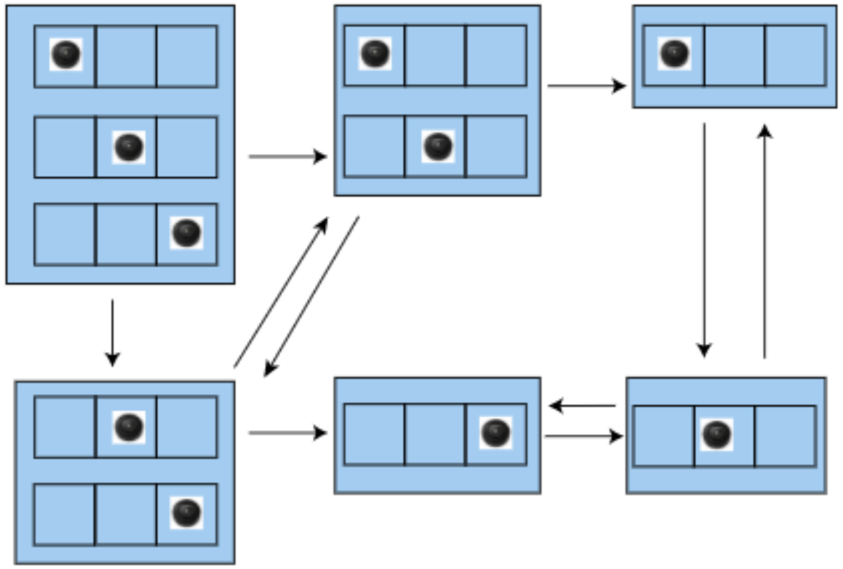
Partial observations

- Belief states: where the agent thinks it might be



Partial observations

- Belief states: where the agent thinks it might be

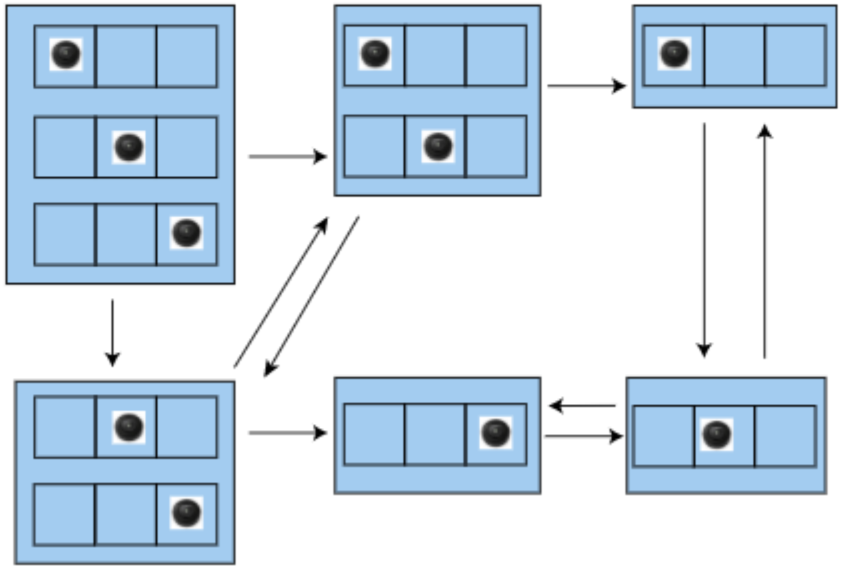


- Upper bound of belief state space size: 2^3-1

What can be a heuristic? Is that optimistic / admissible??

Partial observations

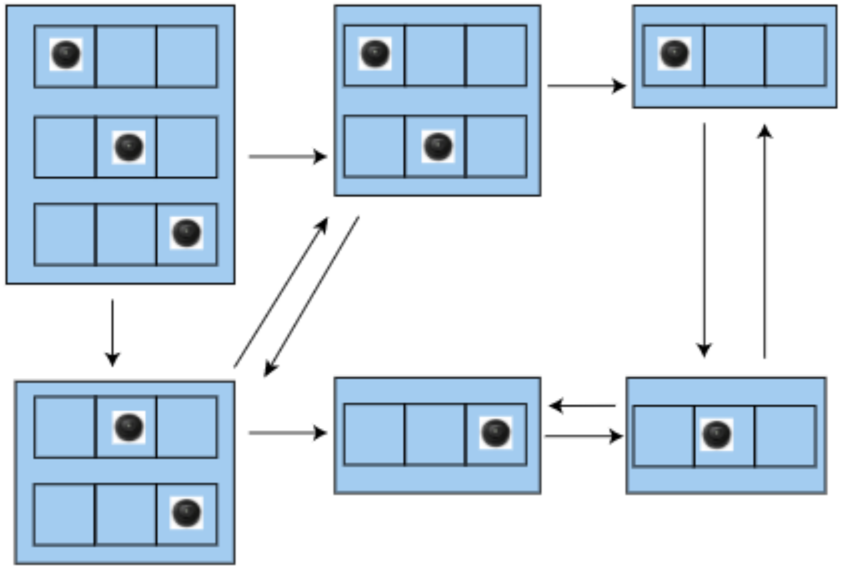
- Belief states: where the agent thinks it might be



- Upper bound of belief state space size: 2^3-1

Partial observations

- Belief states: where the agent thinks it might be



- Heuristic: number of hypotheses in a belief state

Think about and write on paper. Email me your responses!

Discussion

- What's a universal plan for a large empty square room of size 20x20?
- What is a reasonable heuristic for our blind robot? Write down a state and compute the heuristic. Is the heuristic admissible?

Summary

- Non-deterministic actions generate multiple states the agent can be in
- Partial observations generate belief states

Adversarial Search

Objectives

- Define adversarial search and model problems as search tree
- Implement algorithms to solve such a problem
  - minimax algorithm

Outline

- Types of problems
- Adversarial search model
- Minimax

Coordination between agents

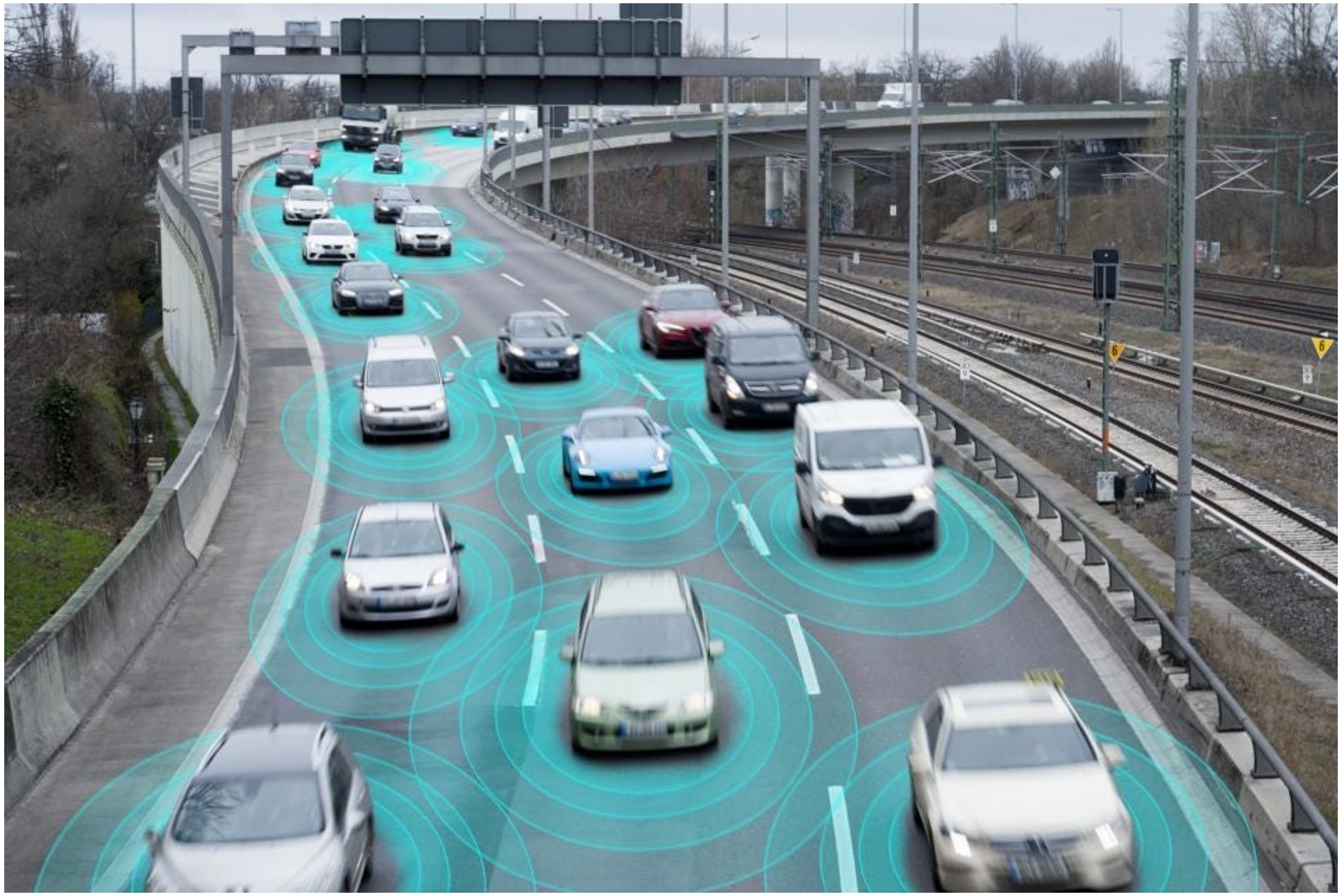


Image: forbes.com

Coordination with humans

Image: Mingi Jeong

Types of Games

- Classified over different axes:
  - Number of players
  - Zero sum
  - Deterministic or stochastic
  - Perfect or imperfect information

- We will focus for now on 2-player zero-sum, deterministic games with perfect information

Zero-Sum Game Games: computer vs. humans

- **Checkers:** 1950: First computer player. 1994: First computer champion: Chinook ended 40-year-reign of human champion Marion Tinsley using complete 8-piece endgame. 2007: Checkers solved!
- **Chess:** 1997: Deep Blue defeats human champion Gary Kasparov in a six-game match. Deep Blue examined 200M positions per second, used very sophisticated evaluation and undisclosed methods for extending some lines of search up to 40 ply. Current programs are even better, if less historic.
- **Go:** Human champions are now starting to be challenged by machines, though the best humans still beat the best machines. In go, b > 300! Classic programs use pattern knowledge bases, but big recent advances use Monte Carlo (randomized) expansion methods.

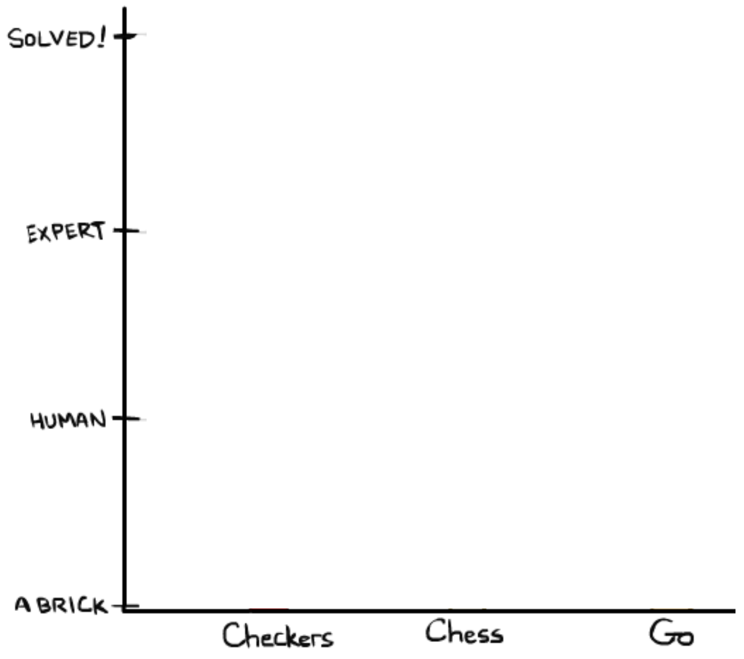


Image: AI Berkeley

Zero-Sum Game Games: computer vs. humans

- **Checkers:** 1950: First computer player. 1994: First computer champion: Chinook ended 40-year-reign of human champion Marion Tinsley using complete 8-piece endgame. 2007: Checkers solved!
- **Chess:** 1997: Deep Blue defeats human champion Gary Kasparov in a six-game match. Deep Blue examined 200M positions per second, used very sophisticated evaluation and undisclosed methods for extending some lines of search up to 40 ply. Current programs are even better, if less historic.
- **Go :2016: Alpha GO defeats human champion. Uses Monte Carlo Tree Search, learned evaluation function.**

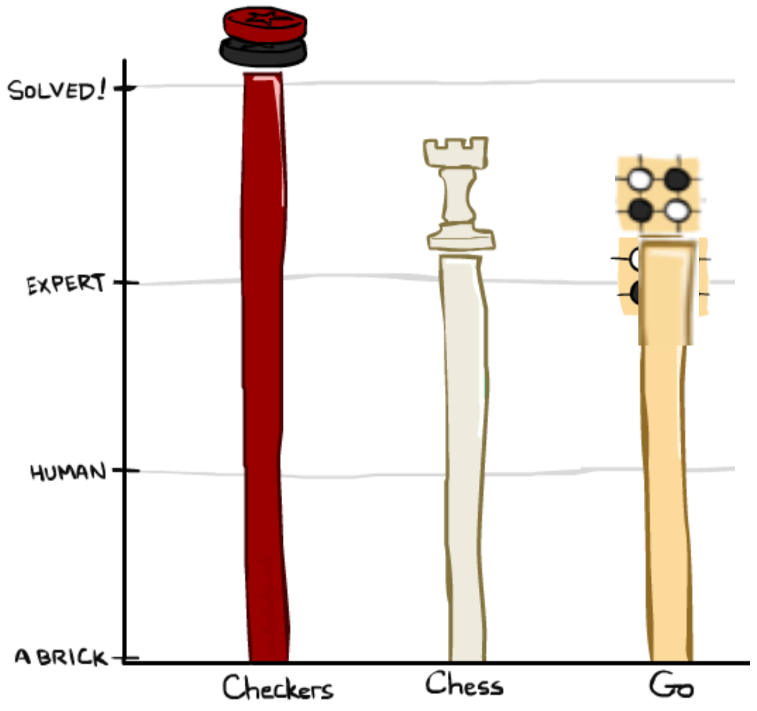


Image: AI Berkeley

Image: google.com

Adversarial search

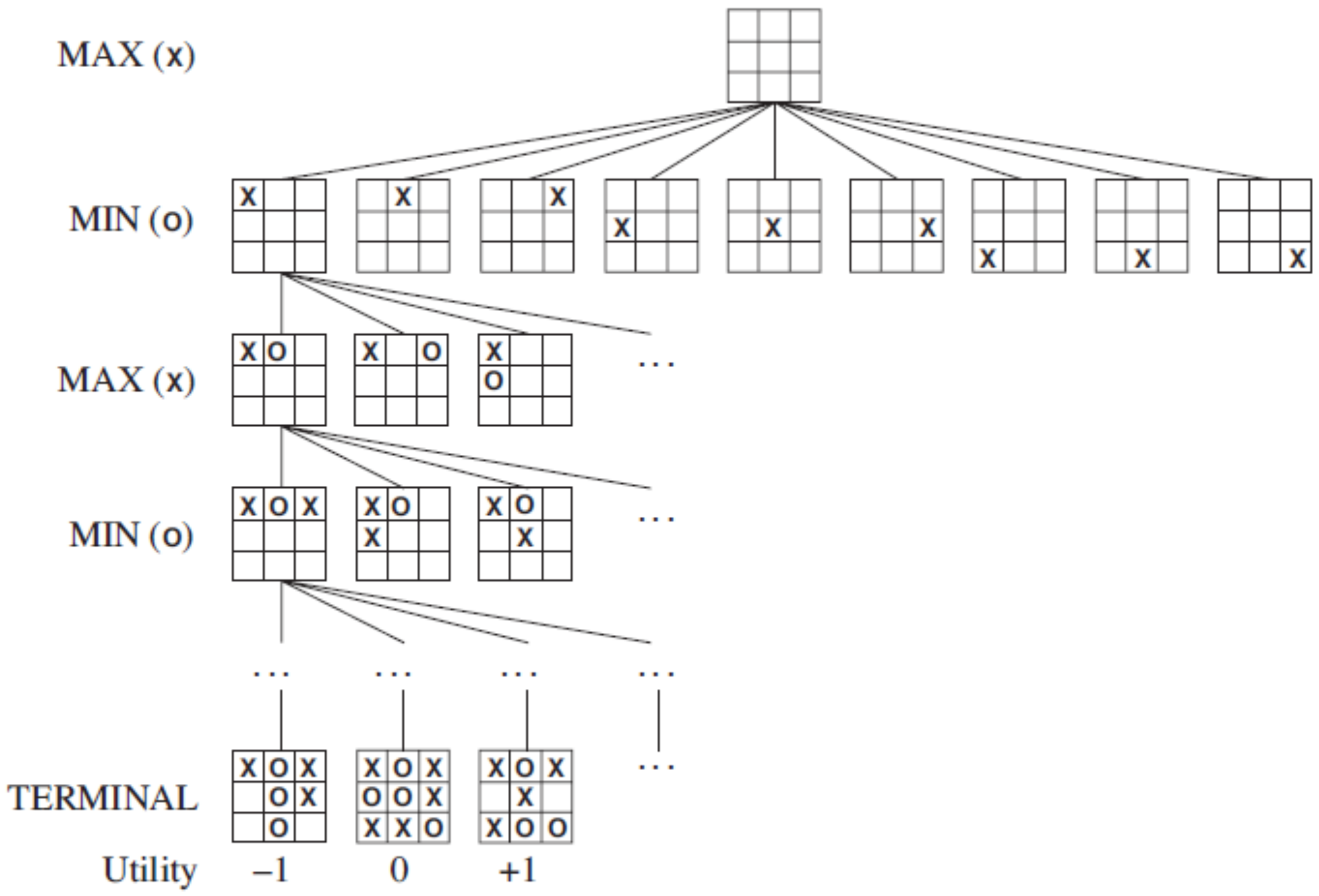
Game as a search problem:

- S_0: **initial state**
- PLAYER(s): defines which player has the move in a state
- ACTIONS(s): Returns the set of legal moves in a state
- RESULT(s, a): **transition model**
- TERMINAL-TEST(s): **terminal test**, which is true when the game is over and false otherwise
- UTILITY(s, p): A **utility function** for the numeric value for a game in terminal state s for a player p.

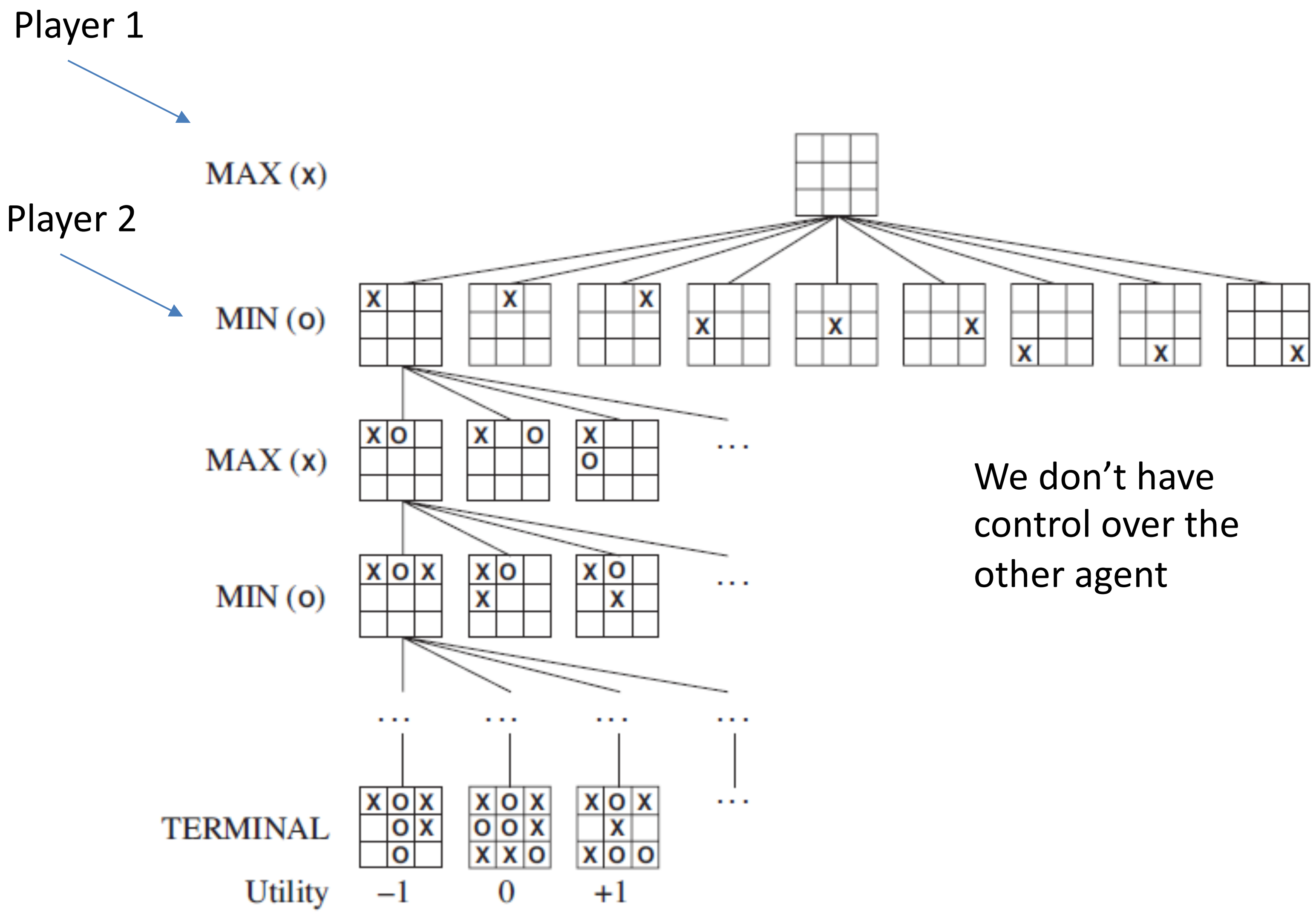
Solution for a player is a policy: S->A

We want to maximize the utility rather than minimizing the cost

Game tree vs search tree



Game tree vs search tree



We don't have control over the other agent

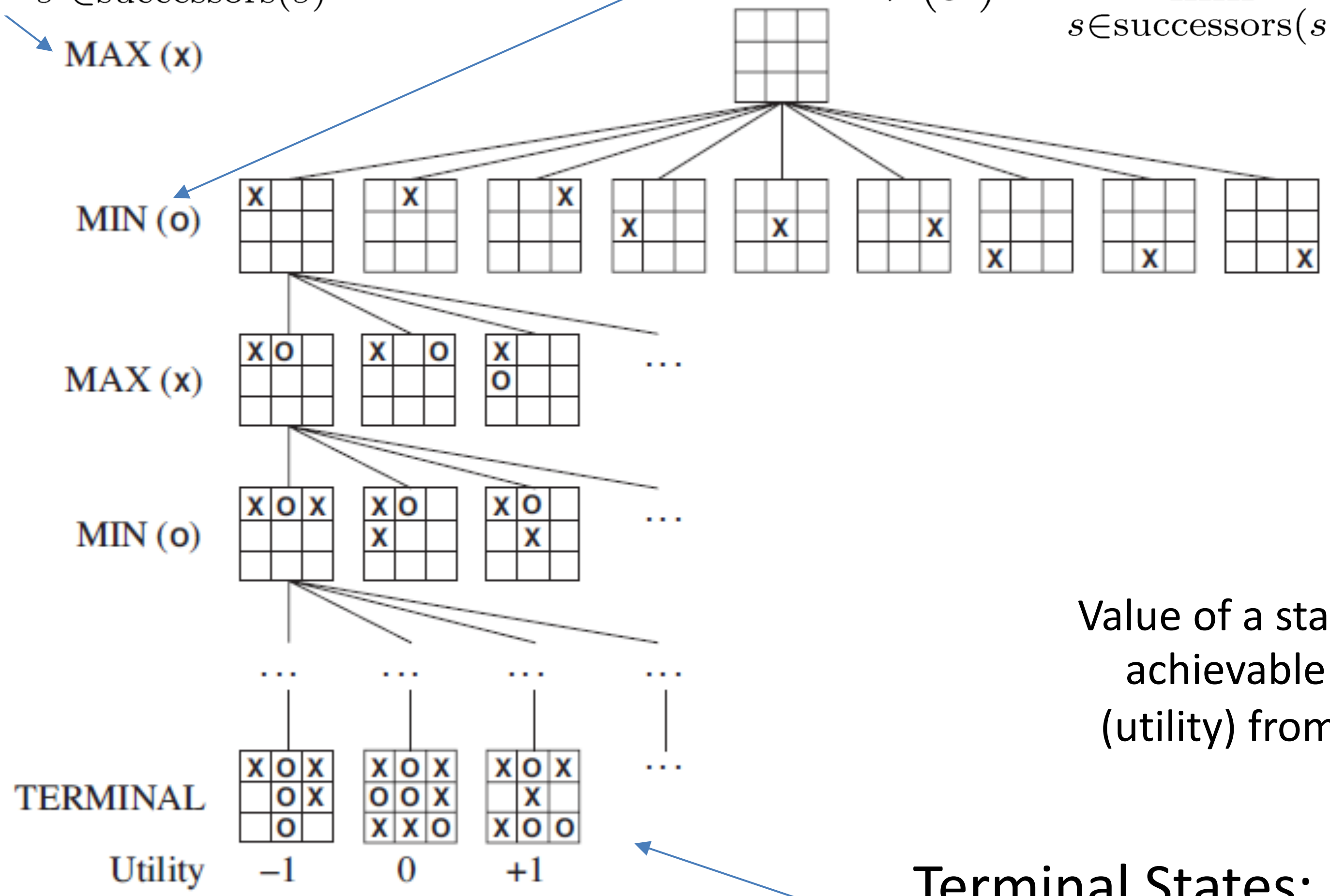
Game tree vs search tree

States Under Agent's Control:

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

States Under Opponent's Control:

$$V(s') = \min_{s \in \text{successors}(s')} V(s)$$

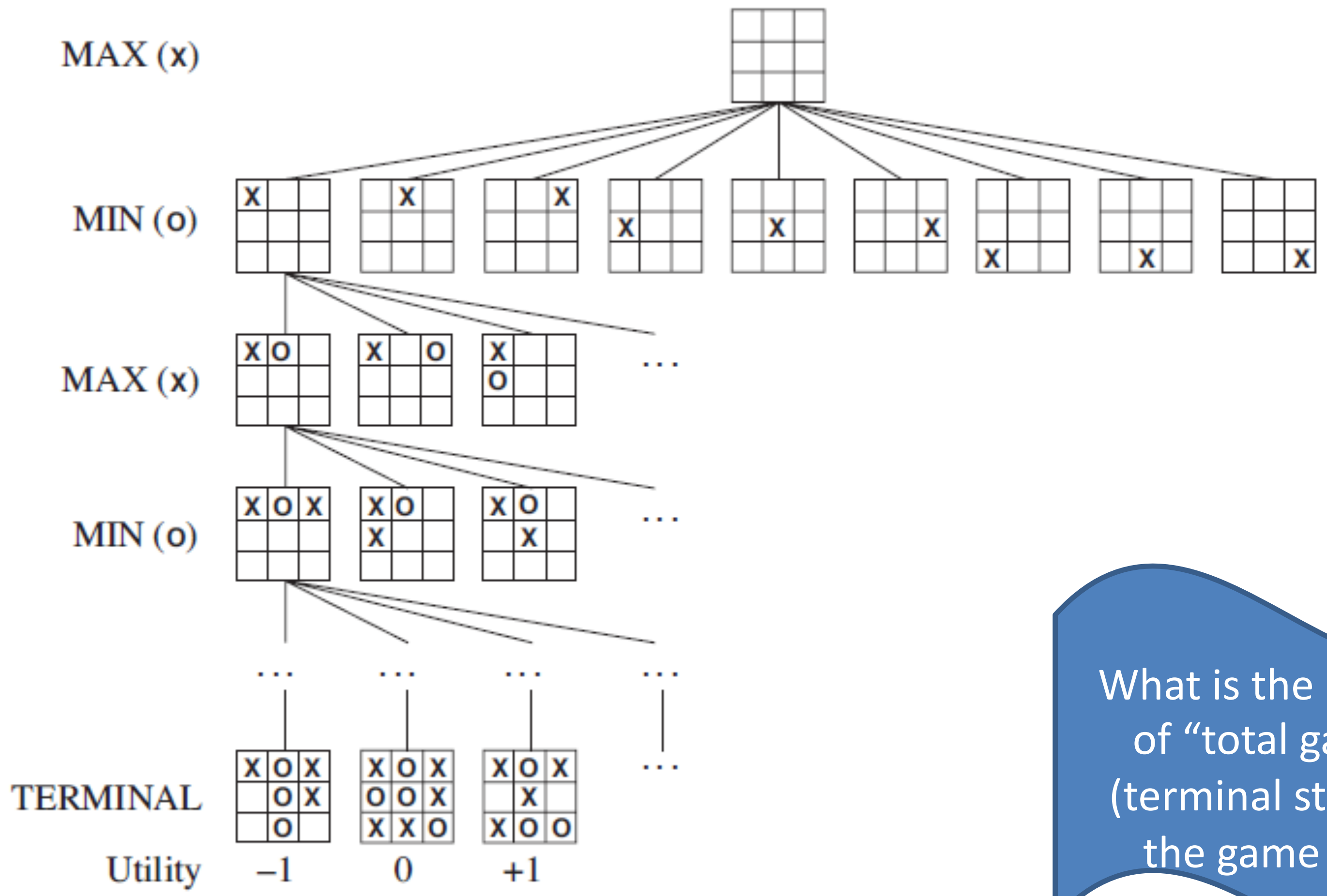


Value of a state: The best achievable outcome (utility) from that state

Terminal States:

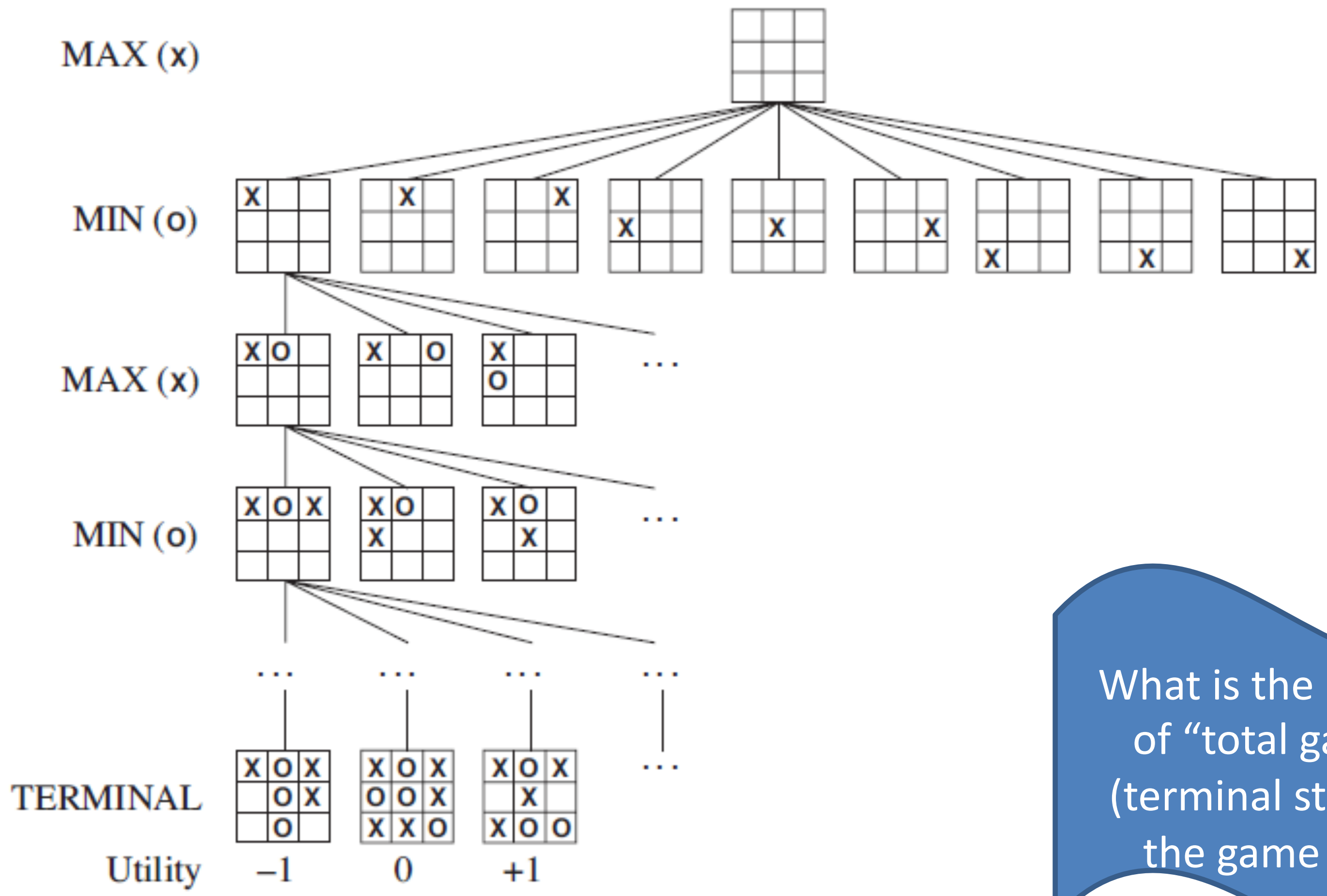
$$V(s) = \text{known}$$

Game tree vs search tree



What is the number of "total games" (terminal states) in the game tree?

Game tree vs search tree



What is the number of "total games" (terminal states) in the game tree?

9*8* … * 1 = 9!

Minimax search

Minimax search:

- A state-space search tree
- Players alternate turns
- Compute each node's minimax value: the best achievable utility against a rational (optimal) adversary

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

Minimax search

Minimax search:

- A state-space search tree
- Players alternate turns
- Compute each node's minimax value: the best achievable utility against a rational (optimal) adversary

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

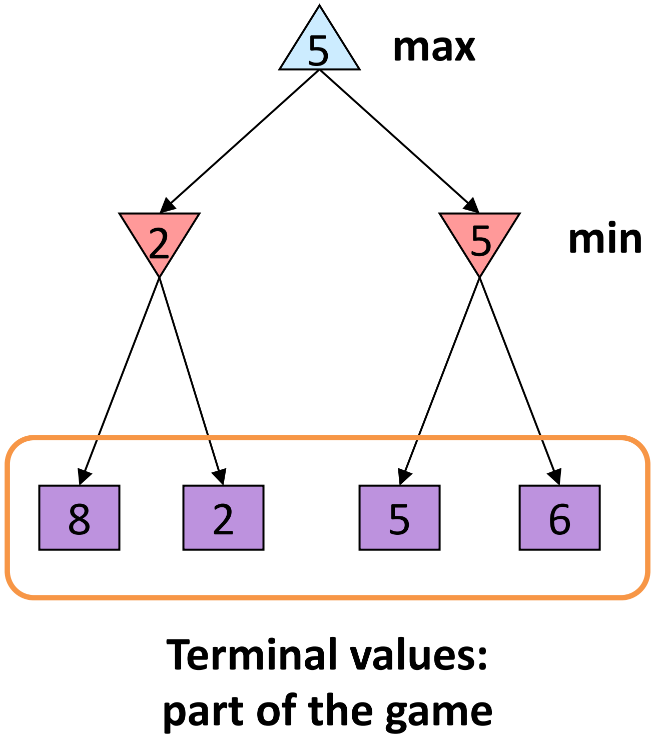
What type of implementation can be used for finding the value?

Minimax search

Minimax search:

- A state-space search tree
- Players alternate turns
- Compute each node's minimax value: the best achievable utility against a rational (optimal) adversary

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$



What type of implementation can be used for finding the value?

Recursive implementation

Minimax Implementation

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)
```

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

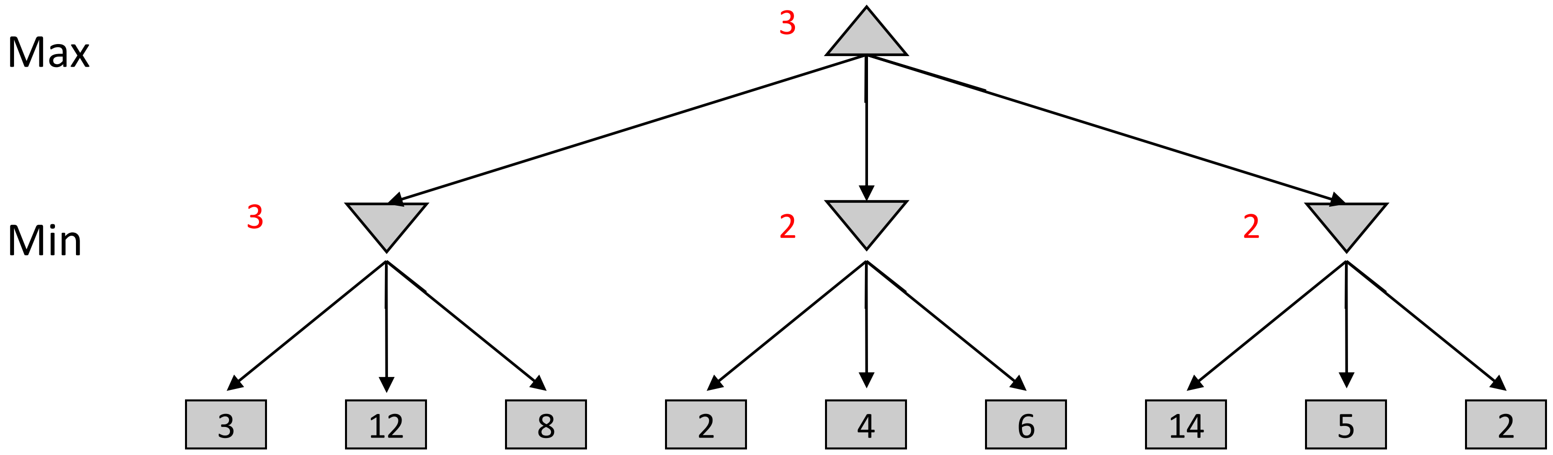
```
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```

Minimax Example

```
def value(state):
        if the state is a terminal state: return the state's utility
        if the next agent is MAX: return max-value(state)
        if the next agent is MIN: return min-value(state)
```

```
def max-value(state):
        initialize v = -∞
        for each successor of state:
                v = max(v, value(successor))
        return v
```

```
def min-value(state):
        initialize v = +∞
        for each successor of state:
                v = min(v, value(successor))
        return v
```



Minimax properties

- Complete: yes, if tree is finite
- Time: $O(b^m)$
- Space: $O(bm)$
- Optimal: assuming perfect players

Minimax properties

- Complete: yes, if tree is finite
- Time: $O(b^m)$
- Space: $O(bm)$
- Optimal: against perfect players

Do time and space complexities look similar to another algorithm we have seen?

Minimax properties

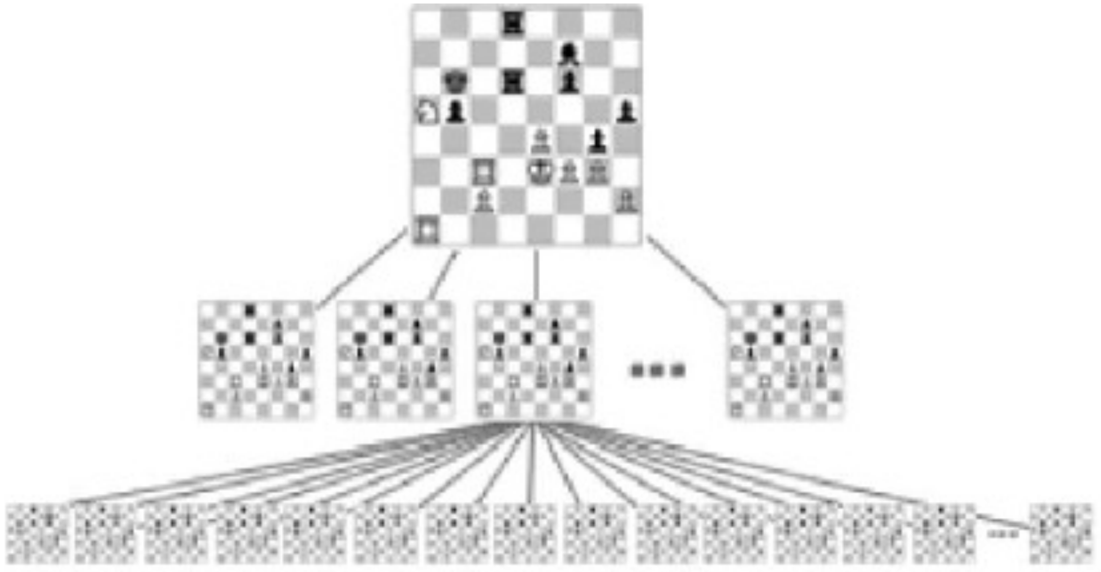
- Complete: yes, if tree is finite
- Time: $O(b^m)$
- Space: $O(bm)$
- Optimal: against perfect players

Do time and space complexities look similar to another algorithm we have seen?

Similar to DFS, though exhaustive

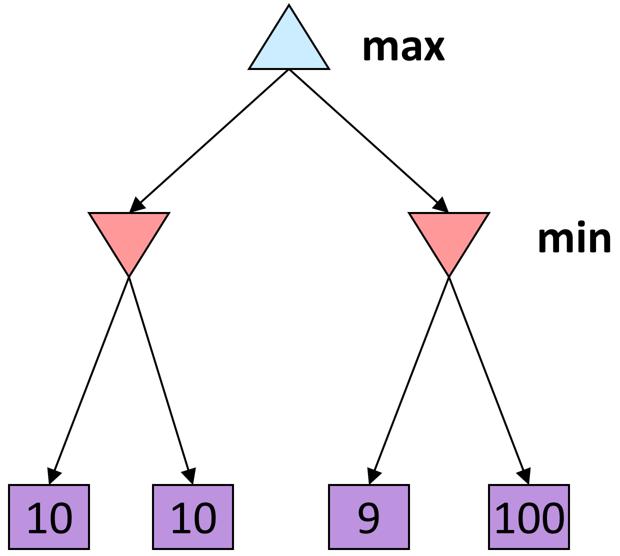
Example for chess

- Example: For chess, $b \approx 35$, $m \approx 100$
  - Exact solution is completely infeasible



Minimax properties

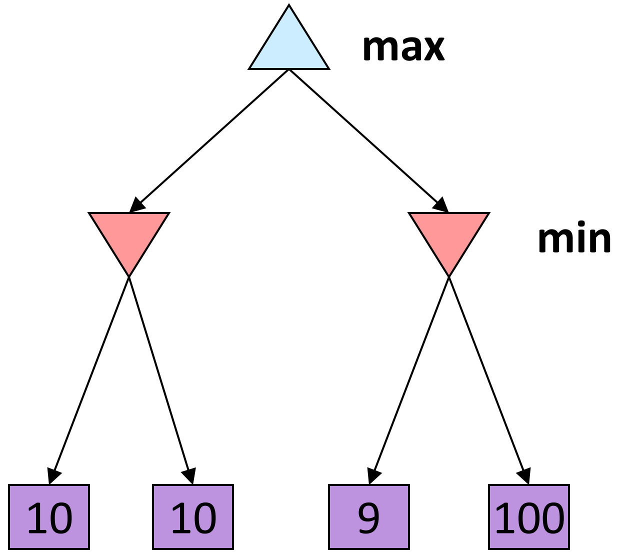
- Complete: yes, if tree is finite
- Time: $O(b^m)$
- Space: $O(bm)$
- Optimal: against perfect players. Otherwise?



What if the other player is not rational?

Minimax properties

- Complete: yes, if tree is finite
- Time: $O(b^m)$
- Space: $O(bm)$
- Optimal: against perfect players. Otherwise?



Minimax optimizes the worst-case outcome

You could have better outcomes if the other player not rational, but the strategy will perform worse against optimal players

What if the other player is not rational?