Level 2

Game Trees

Riccardo Pucella

September 23, 2014

Strategy Games

Last project: puzzle games

- really — round-based one-player games

This project: strategy games

- really — round-based multi-player games
- really really — two-player games

Why are they called strategy games?

Strategy Games

Last project: puzzle games

- really — r

This project:

- really — r
- really reall

Take into account not only the world state, but also the possible actions that the other player might take

Why are they called strategy games?

Classic Example: Tic-Tac-Toe

Two players A (Alice) — plays X
B (Bob) — plays O

We usually take Alice’s perspective

- Alice wins (and therefore Bob loses)
- Alice loses (and therefore Bob wins)
- Alice draws (and therefore Bob draws)

Why Tic-Tac-Toe?

Classic Example: Tic-Tac-Toe

Two players A (Alice) — plays X
B (Bob) — plays O

We usually take Alice's perspective

- Alice wins (and therefore Bob loses)
- Alice loses (and therefore Bob wins)
- Alice draws (and therefore Bob draws)

Why Tic-Tac-Toe? *It is simple*

What does simple mean?

More generally, how do you think about strategy games?

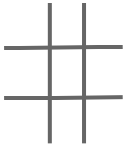
Game trees:

- World states as nodes
- Initial world state is the root
- T is a child of S when
  there is a legal move from S to T

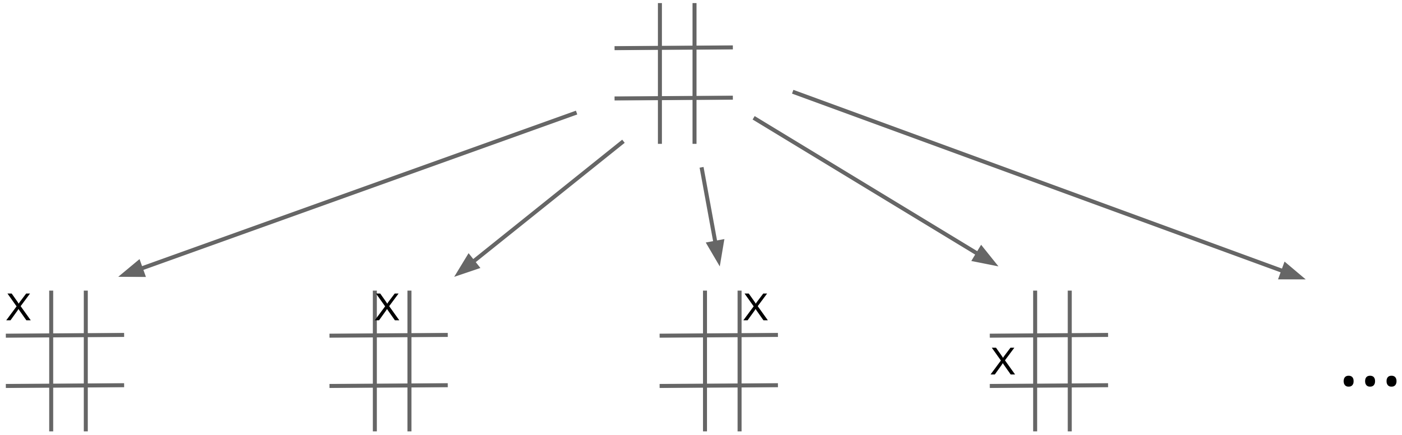
A game tree captures all the possible plays

Game Tree for Tic-Tac-Toe

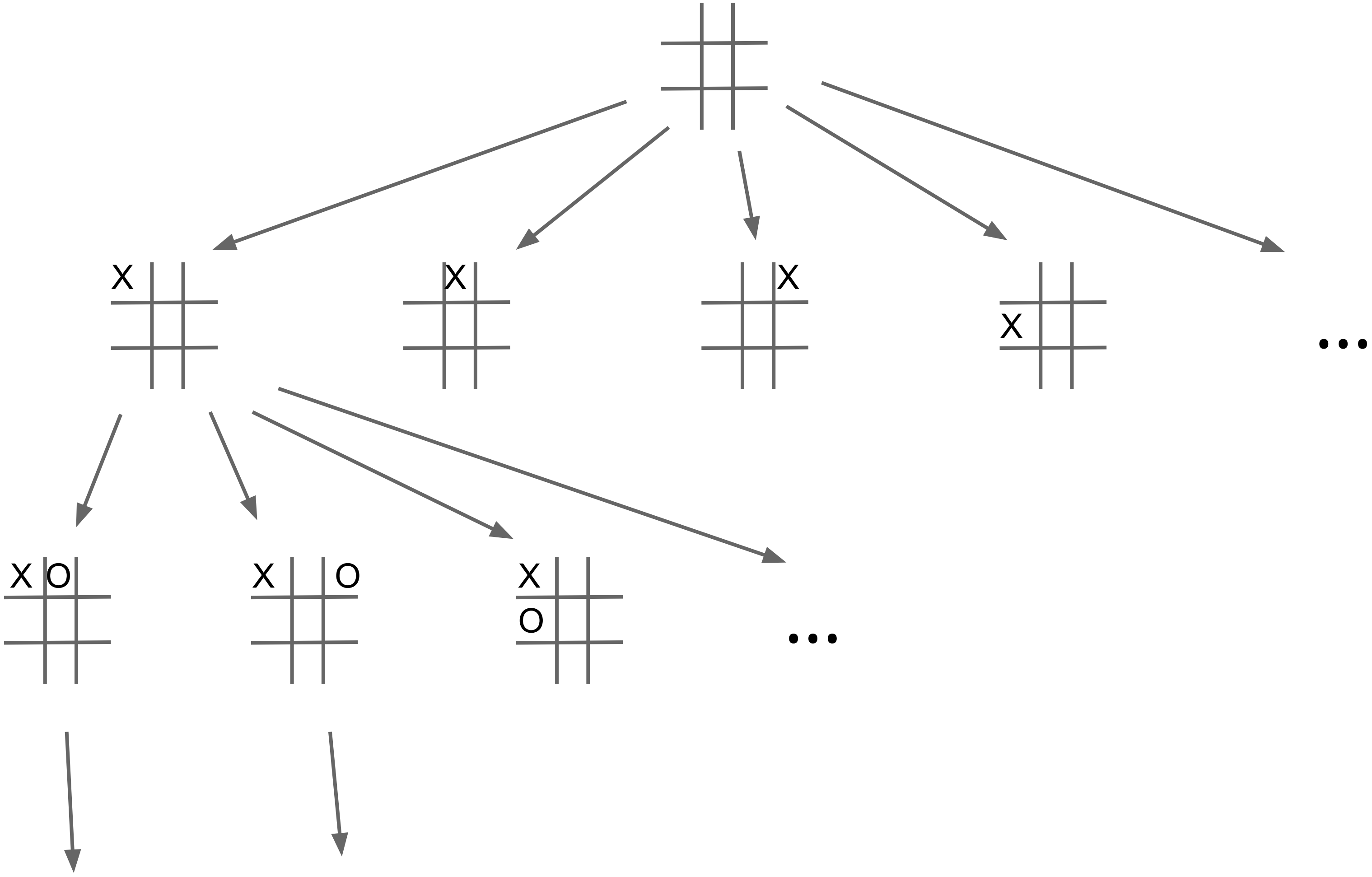
Game Tree for Tic-Tac-Toe



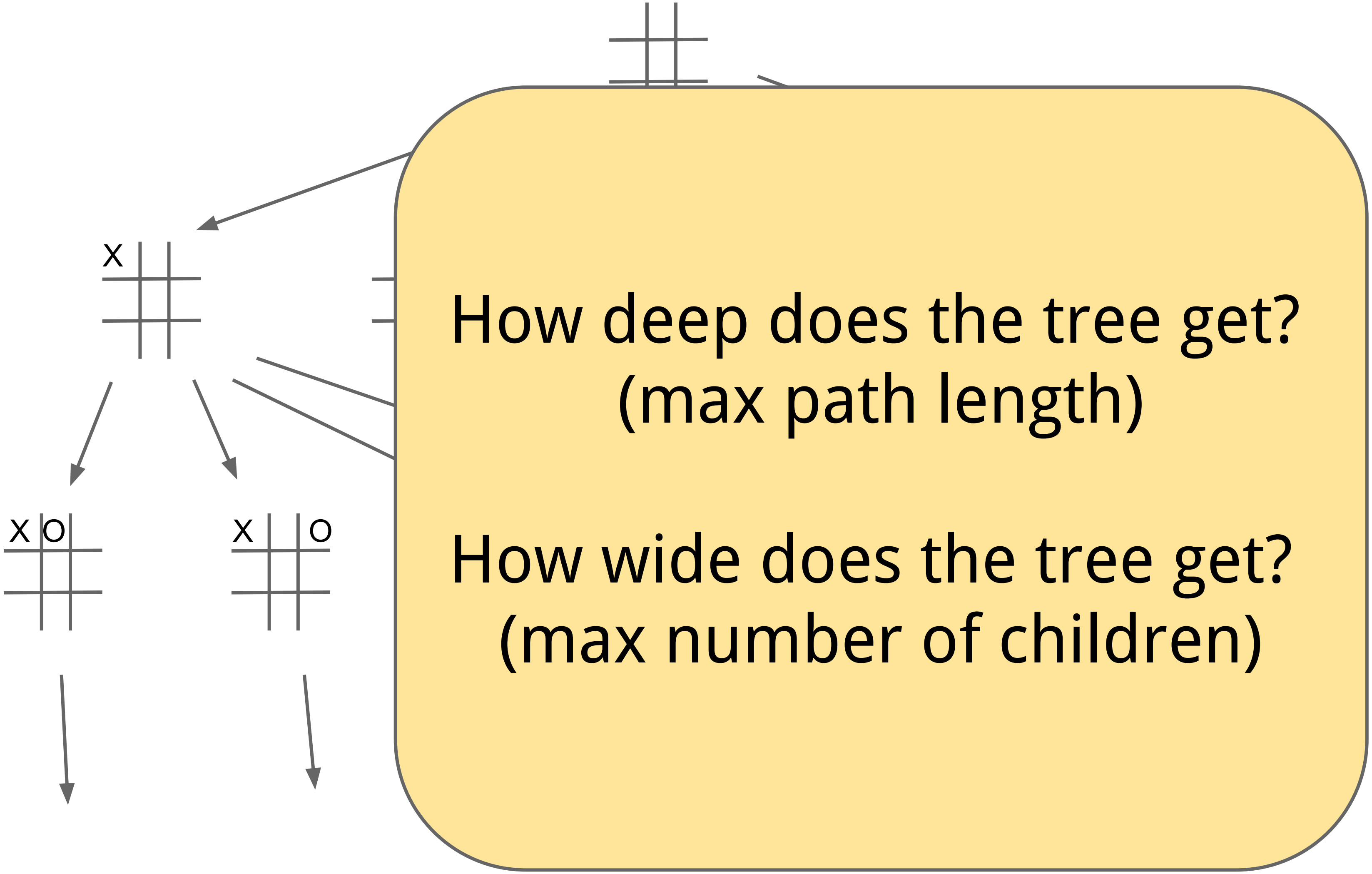
Game Tree for Tic-Tac-Toe



Game Tree for Tic-Tac-Toe



Game Tree for Tic-Tac-Toe



How deep does the tree get?
(max path length)

How wide does the tree get?
(max number of children)

Toe

Compare to chess:

Depth ~ 100 (common)

Branching factor ~ 25

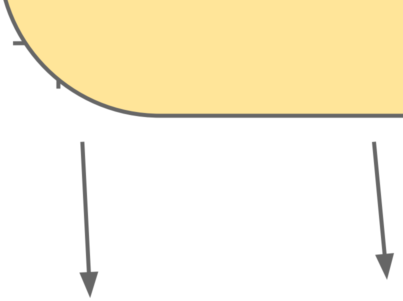
→ $O(25^{100})$ nodes

ree get?

h)

ee get?

(max number of children)



Game Trees for One-Player Games

Game trees not restricted to strategy games

Any round-based game can be thought of in terms of game trees

- Our Rush Hour game has a game tree
- What could it be used for?

Game Trees for One-Player Games

Game trees not restricted to strategy games

Any round-based game can be thought of in terms of game trees

- Our Rush Hour game has a game tree
- What could it be used for?

Determine if a board has a solution

- Construct game tree with that board as root
- Search for a solution (BFS, DFS, …)

Game Trees for Two-Player Games

The game tree should help a player decide which move to make in a given state

- Alice wants to win
- Bob also wants to win — wants Alice to lose

The best move in every state

- Depends on the current state
- Also depends on what the other player does

Assume opponent plays perfectly

Utility of final states

We associate with every final state a utility

- represents how good the state is for Alice
- think of it as a payoff for Alice
  - Alice wants the biggest payoff
  - Bob wants Alice to have the smallest payoff

If you only care about winning/losing:

utility = 1 for a win

utility = 0 for a draw

utility = -1 for a loss

Utility of final states

We associate with every final state a utility

- represents h
- think of it as
  - Alice wants t
  - Bob wants Al

If you only care about ng/losing:

utility = 1 for a win

utility = 0 for a draw

utility = -1 for a loss

Actual numbers not important

The order is

Selecting best moves

The best move for Alice is a move that

- leads to the final state with highest utility
- … given that Bob will do his best to get to a state with lowest utility!

The best move for Bob is a move that

- leads to the final state with lowest utility
- … given that Alice will do her best to get to a state with highest utility!

Minimax values

Compute, for every state in the game tree:

- the utility of the “best” final state reachable for that player assuming best play from opponent

Minimax value of a state (Alice’s turn)

maximum of minimax value of children

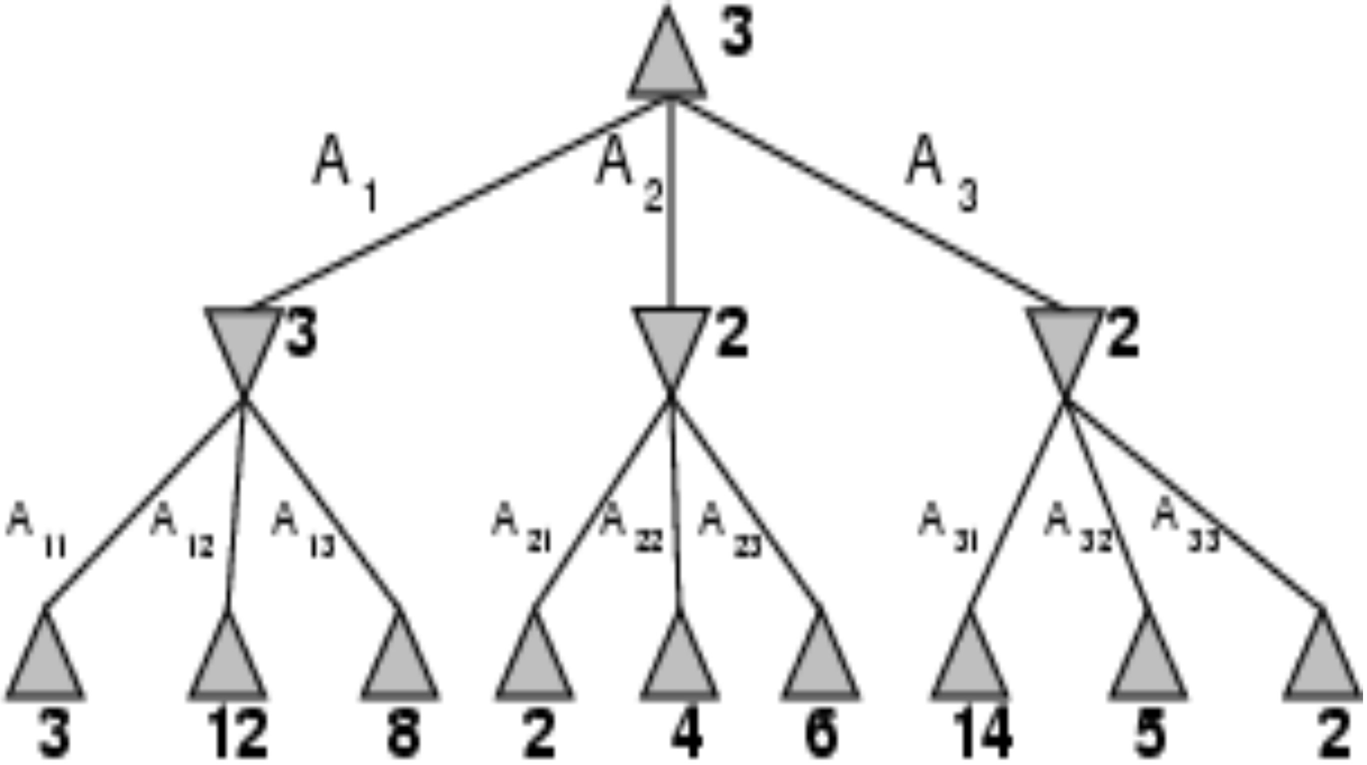
Minimax value of a state (Bob’s turn)

minimum of minimax value of children

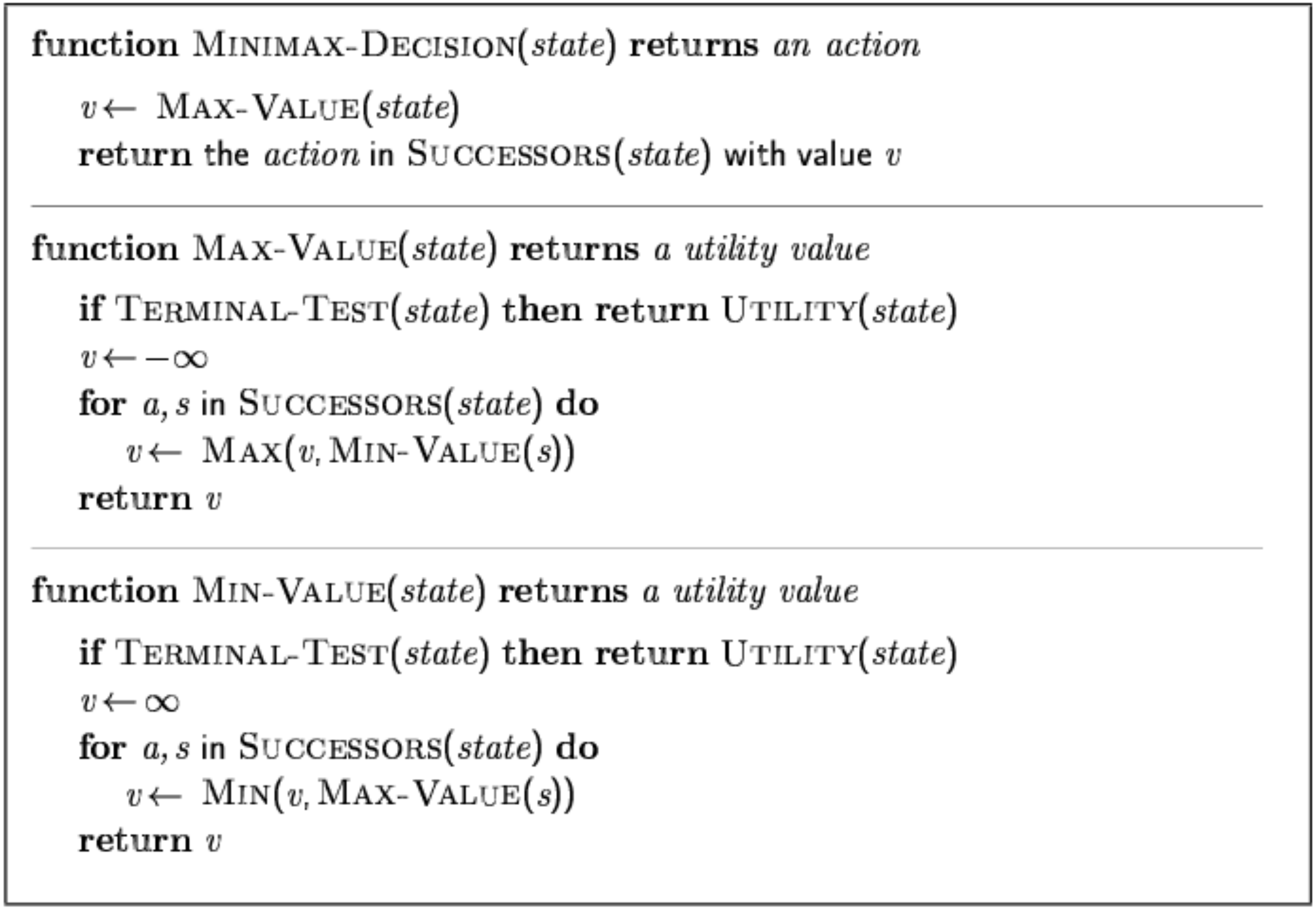
Simple example — 2-round game

Alice
(max)

Bob
(min)



Minimax algorithm



```
function MINIMAX-DECISION(state) returns an action
    $v \leftarrow$ MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value $v$

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    $v \leftarrow -\infty$
    for $a, s$ in SUCCESSORS(state) do
        $v \leftarrow$ MAX($v$, MIN-VALUE($s$))
    return $v$

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    $v \leftarrow \infty$
    for $a, s$ in SUCCESSORS(state) do
        $v \leftarrow$ MIN($v$, MAX-VALUE($s$))
    return $v$
```

Implementation details

You don’t have to construct the game tree explicitly!

You can create it “on the fly”

All you care about in the minimax algorithm is

- what’s the current state?
- what are the successor states you can get to from the current state with legal moves?

Obvious problem

The game tree can get big

- Chess (pretty huge thank you)
- Go (beyond comprehension)

Next time, we’ll look at ways to get around this problem

Exercise: minimax for Tic-Tac-Toe

Some code on the web site to get you started