Adversarial Search and Game-Playing

Dr. Sonali Patil

Common Terms in Game Theory

- **Game** — Any interaction between two or more players in which each player's payoff is affected by their decisions & the decisions made by others.
- **Players** — The interdependent agents of the game, which might range from individuals to governments, to companies, etc…
- **Actions** — The strictly-defined behaviors that a player has to choose between, what can players do? Enter a bid? End a strike? Bet on a coin flip?

Common Terms in Game Theory

- **Payoff** — The specific, exact, increases or decreases of "value" within a value system that maps to a player's action.
- **Zero-Sum** — A situation in which one player's gain is equivalent to another's loss; the net change in wealth or benefit to the game as a whole is zero.
- **Non-Zero-Sum** — A situation where there is a net benefit or loss to the system based on out the outcome of the game; winnings & losses of all players do not add up to zero.

Common Terms in Game Theory

- **Simultaneous** — When players are making decisions & taking actions *at approximately the same time*; they don't know the choices of other players when making their choices, such as rock-paper-scissors.
- **Sequential** — When players are making decisions & taking actions in alternating turns such as monopoly or chess.
- **Non-Cooperative** — The more common type of game, this is a strictly-competitive game among individual players.

Common Terms in Game Theory

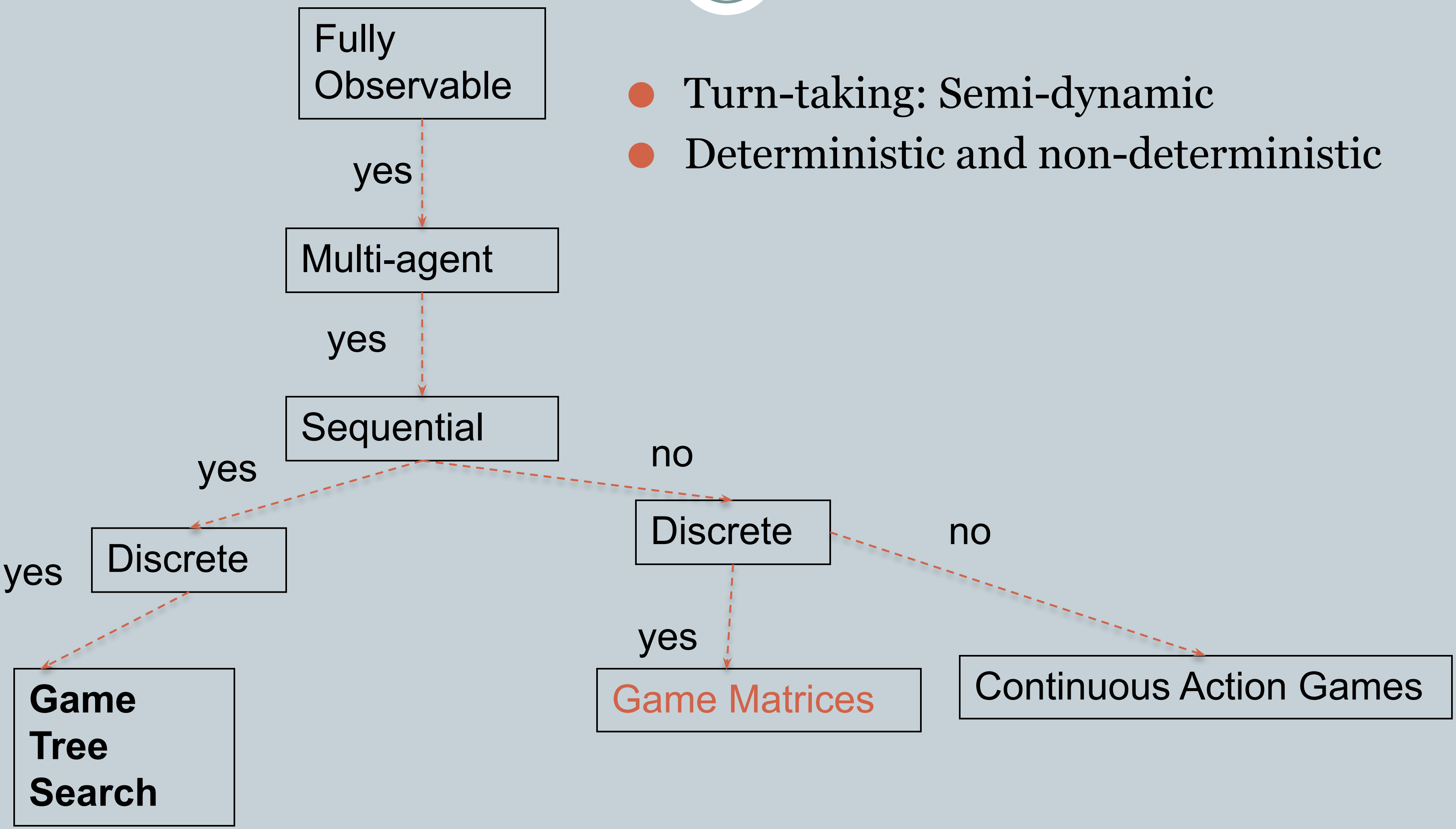
- **Cooperative** — A type of game in which players can forge alliances & respond cooperatively to external credible threats.
- **Complete Information** —A game in which knowledge about other players is available to all participants; the payoff functions, strategies & “types” of players are common knowledge.
- **Incomplete Information** — A game in which players may or may not have information on the game-type, player actions player-type, strategies, payoffs..

Common Terms in Game Theory

- **Imperfect Information** —A game in which players are unaware of the *actions* chosen by other players; however, everything else, player-type, strategies, payoffs, etc…is common knowledge.

Environment Type Discussed In this Lecture

- Turn-taking: Semi-dynamic
- Deterministic and non-deterministic



Game Matrices

Bob

Alice	Paper	Rock	Scissor
Scissor	(1,-1)	(-1,1)	(0,0)
Rock	(-1,1)	(0,0)	(1,-1)
Paper	(0,0)	(1,-1)	(-1,1)

Adversarial Search

- Examine the problems that arise when we try to plan ahead in a world where other agents are planning against us.
- A good example is in board games.
- Adversarial games, while much studied in AI, are a small part of game theory in economics.

Typical AI assumptions

- Two agents whose actions alternate
- Utility values for each agent are the opposite of the other
  - creates the adversarial situation
- Fully observable environments
- In game theory terms: Zero-sum games of perfect information.
- We'll relax these assumptions later.

Search versus Games

- Search – no adversary
  - Solution is (heuristic) method for finding goal
  - Heuristic techniques can find *optimal* solution
  - Evaluation function: estimate of cost from start to goal through given node
  - Examples: path planning, scheduling activities

- Games – adversary
  - Solution is **strategy** (strategy specifies move for every possible opponent reply).
  - **Optimality depends on opponent.**
  - Time limits force an *approximate* solution
  - Evaluation function: evaluate "goodness" of game position
  - Examples: chess, checkers, Othello, backgammon

Types of Games

	deterministic	Chance moves
Perfect information	Chess, checkers, go, othello	Backgammon, monopoly
Imperfect information (Initial Chance Moves)	Bridge, Skat	Poker, scrabble, blackjack

- on-line backgammon
- on-line chess
- tic-tac-toe

- **Theorem of Nobel Laureate Harsanyi:** Every game with chance moves during the game has an equivalent representation with initial chance moves only.
- A deep result, but computationally it is more tractable to consider chance moves as the game goes along.
- This is basically the same as the issue of full observability + nondeterminism vs. partial observability + determinism.

Game Setup

- Games as search:
  - Initial state is initial position: e.g. board configuration of chess
  - Successor function: list of (move,state) pairs specifying legal moves from any position.
  - Terminal test: Is the game over?
  - Utility function: Gives numerical value of terminal states. Numerical outcome for the game. E.g. win (+1), lose (-1) and draw (0) in tic-tac-toe or chess

Game Setup

- Basic Strategy
  - Grow a search tree
  - Only one player can move at each turn
  - Assume we can assign a payoff to each final position-called a utility
  - We can propagate values from the final positions
  - Assume the opponent always makes moves worst for us
  - Pick best moves on own turn

2 Player Games

- Two players
- Zero Sum
- Perfect Information

The two players take turns and try respectively to maximize and minimize a utility function

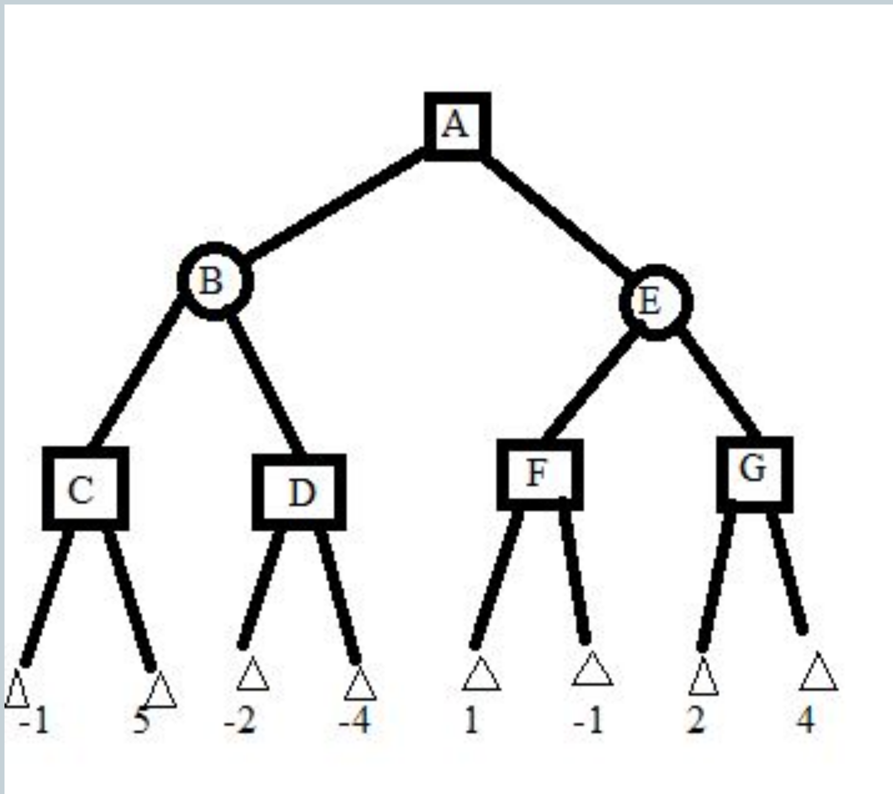
The two players are called respectively as MAX and MIN. We assume that the MAX player makes the first move. They take turns until the game is over. Winner gets award, loser gets penalty

The leaves represent the terminal positions

2 Player Games

- Successive nodes represent positions where different players must move. We call the nodes as MAX or MIN nodes depending of who is the player that must move at that node.
- A game tree could be infinite
- The ply of the node is the number of moves needed to reach that node (i.e. arcs from the root of the tree). The ply of a tree is the maximum of the piles of its nodes.

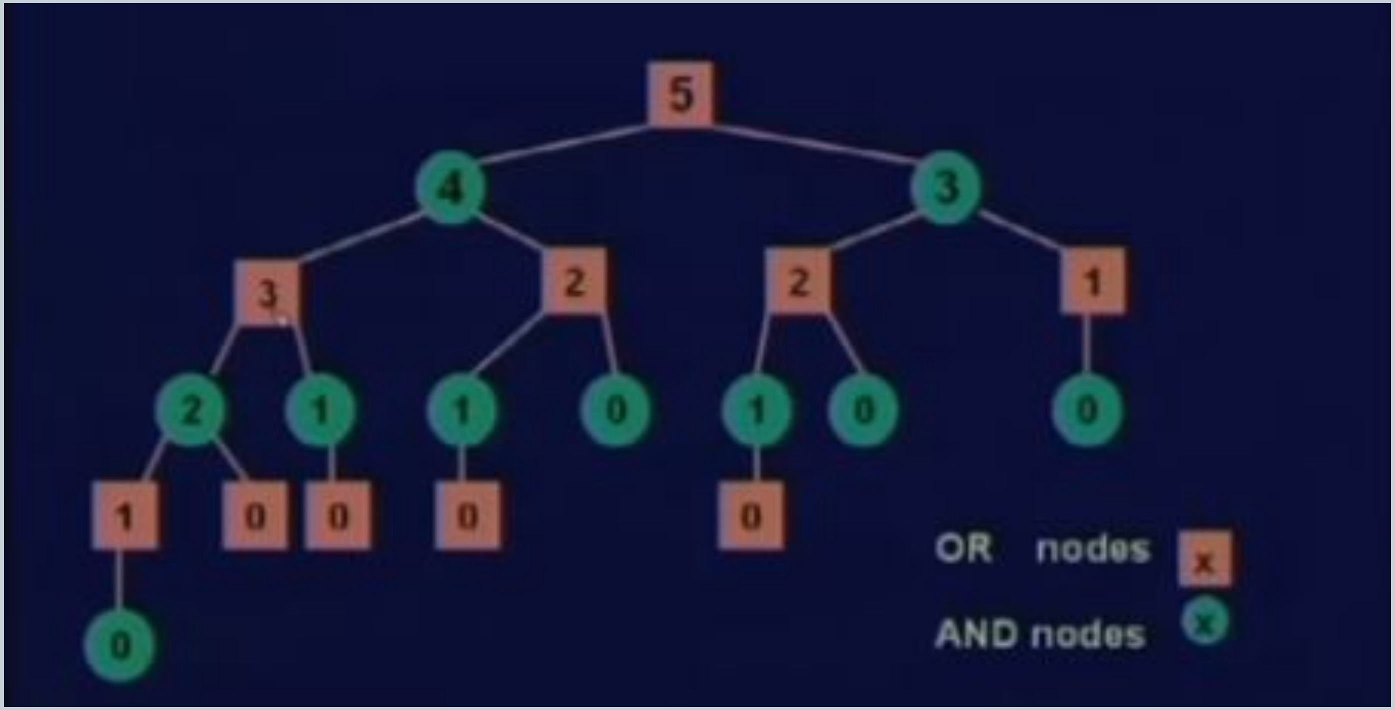
2 Player Games



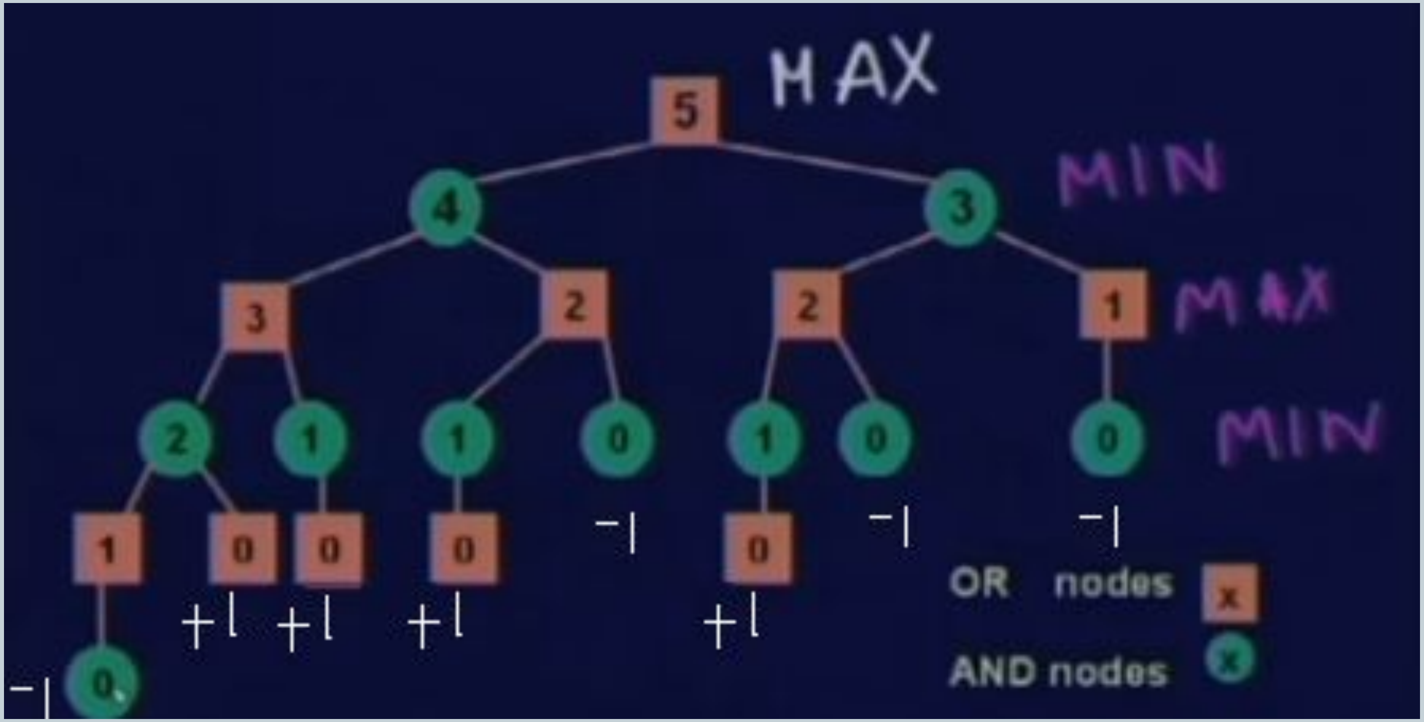
Brute Force Search

- We begin considering a purely brute-force approach for the game playing
- Start at root node and generate entire search tree till the leaf positions assuming that the tree is finite. Feasible for only small games, but provide basis for further discussions (For large games tree could be infinite. Here we need some strategy to define the best move)
- Example of simple 5-Stone Nim
  - Played with two players and pile of stones
  - Each Player removes 1 or 2 stones from the pile
  - Player who removes last stone wins the games

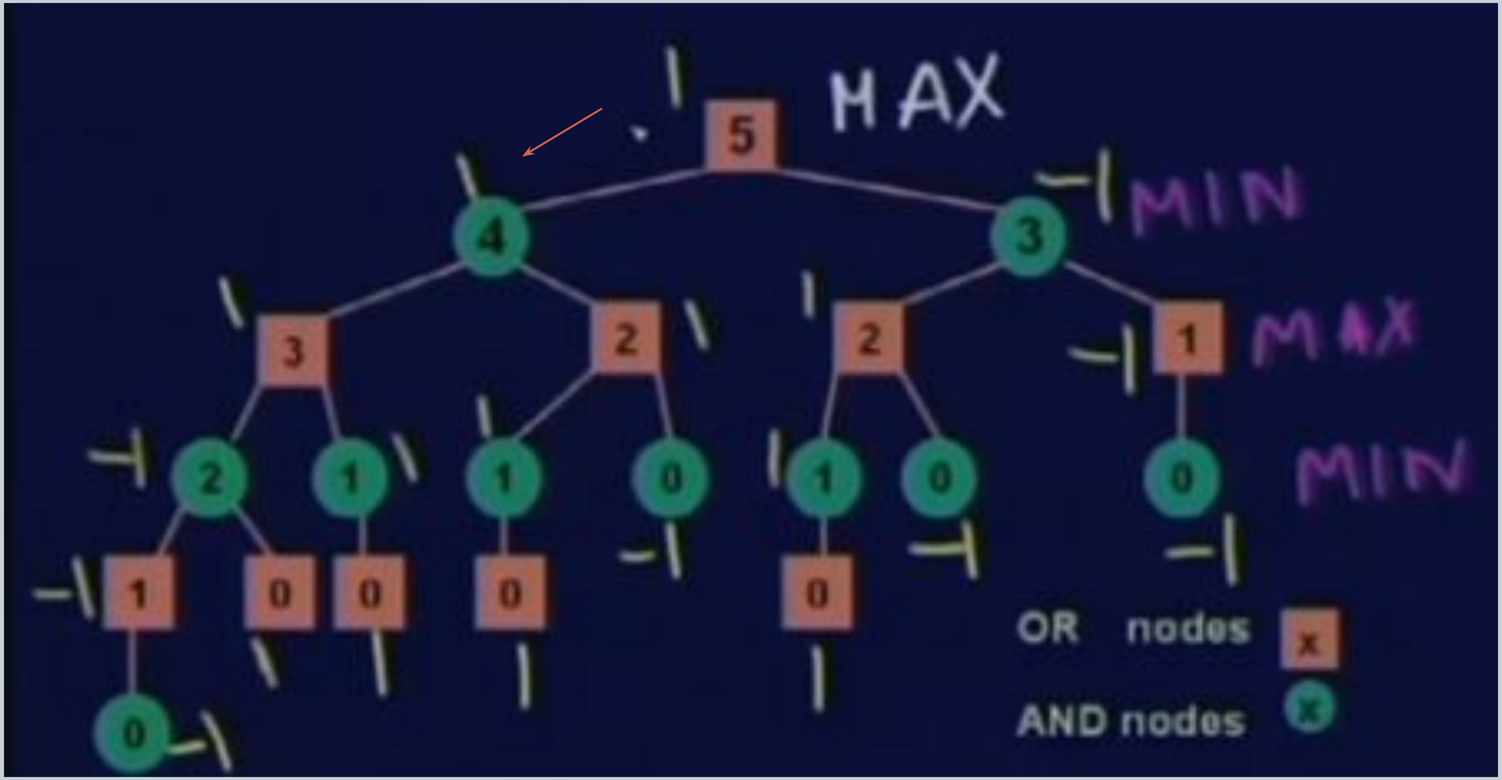
Game Tree for 5-Stone Nim



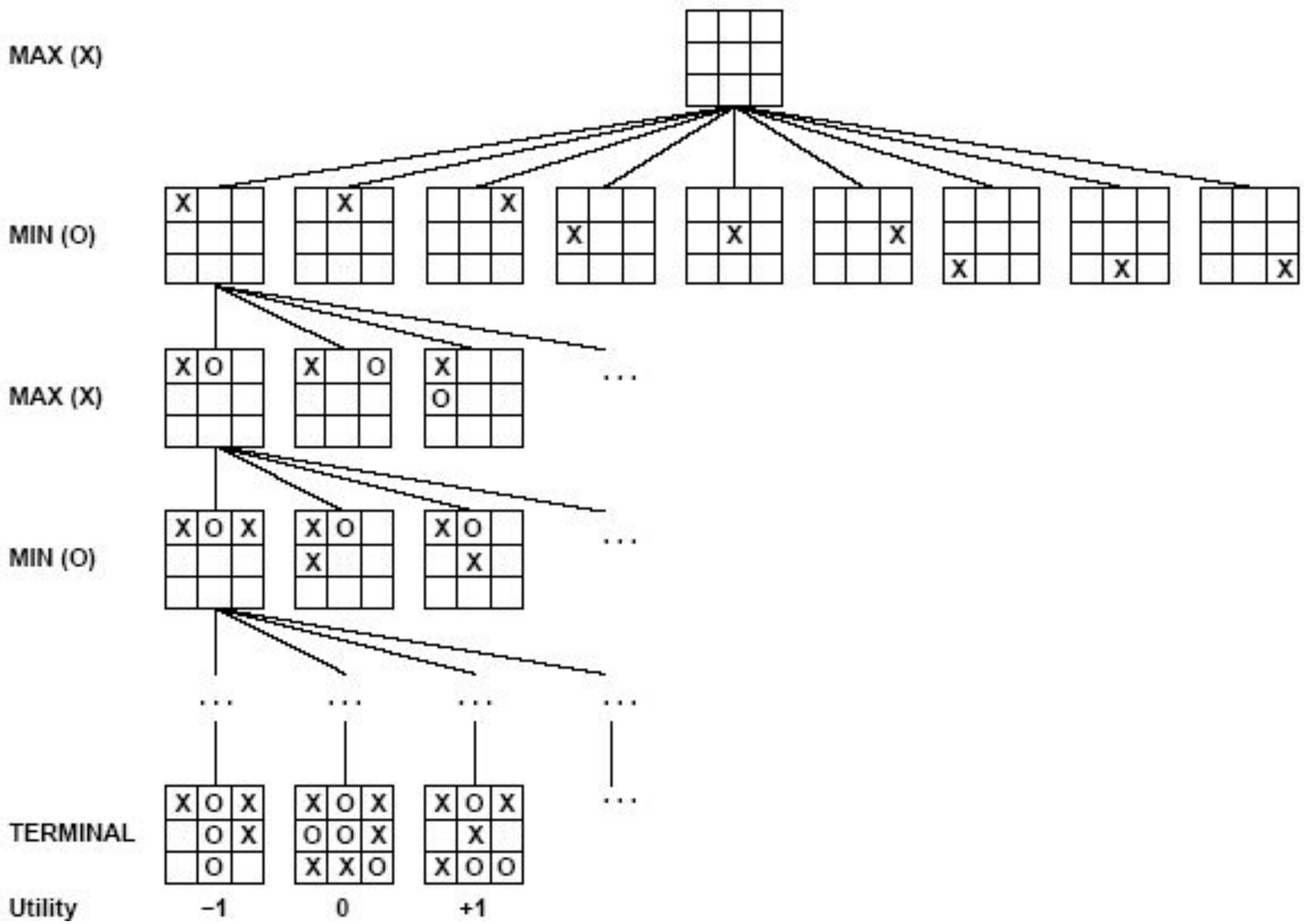
Game Tree for 5-Stone Nim



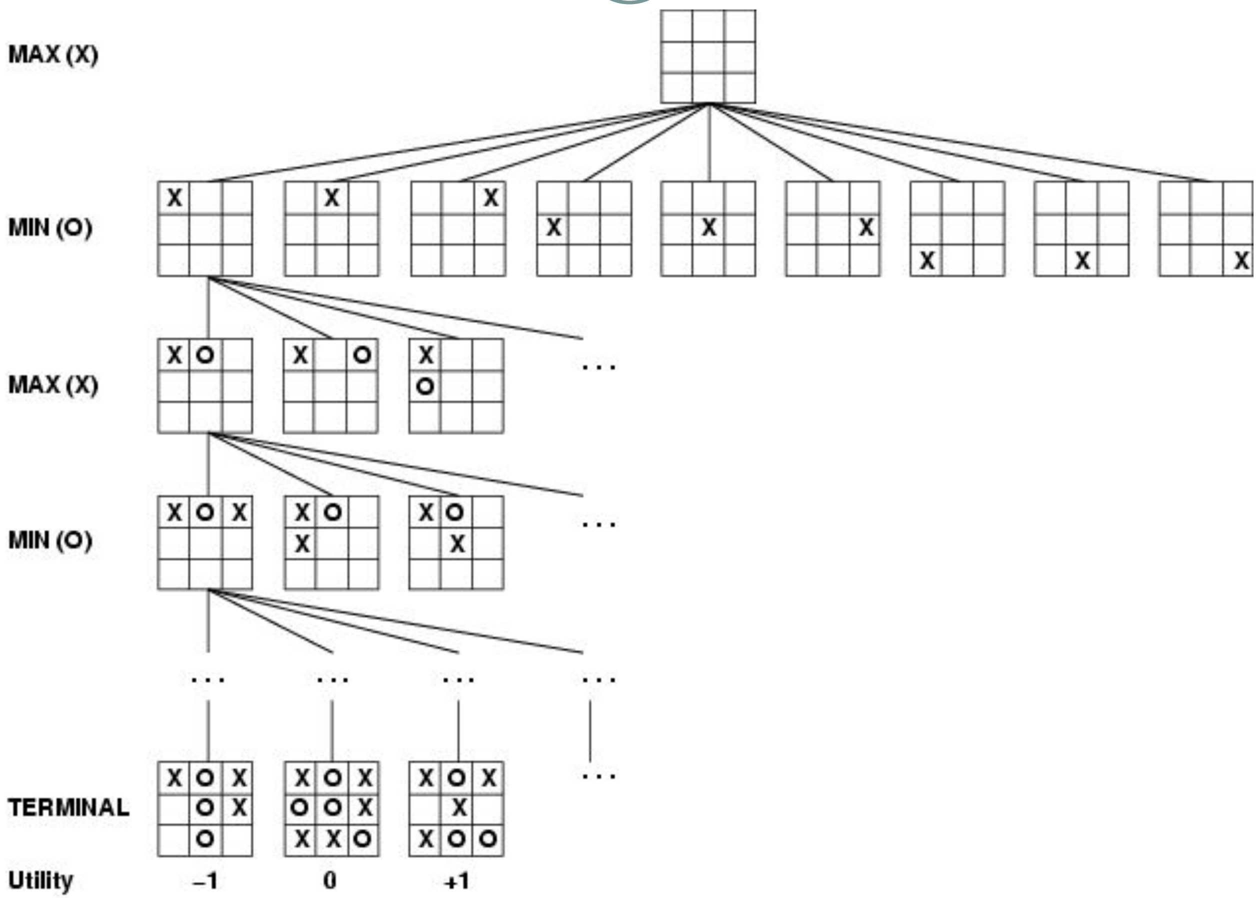
Game Tree for 5-Stone Nim



Partial Game Tree for Tic-Tac-Toe



Game tree (2-player, deterministic, turns)



How do we search this tree to find the optimal move?

Minimax strategy: Look ahead and reason backwards

- Minimax Theorem: Every two-person zero-sum game is a forced win for one player, or a force-draw for either player, in principle, these optimal min-max strategies can be computed. Finding the optimal *strategy* for MAX assuming an infallible MIN opponent
  - Need to compute this all the down the tree
  - If the backed-up value at root is positive, if MAX plays judiciously, MAX can force the win
  - If the value at the root node is negative, no matter how well the MAX plays, if MIN plays extremely intelligently, MIN can force a loss on MAX
  - If the value at the root node is zero and if both the players play their best, the game ends in a draw

Minimax strategy: Look ahead and reason backwards

- Game Tree Search Demo : https://www.yosenspace.com/posts/computer-science-game-trees.html
- Performing this algorithm on Tic-Tac-Toe results in a root being labelled a draw
- Assumption: Both players play optimally!
- Given a game tree, the optimal strategy can be determined by using the **minimax value of each node.**
- Zermelo 1912.

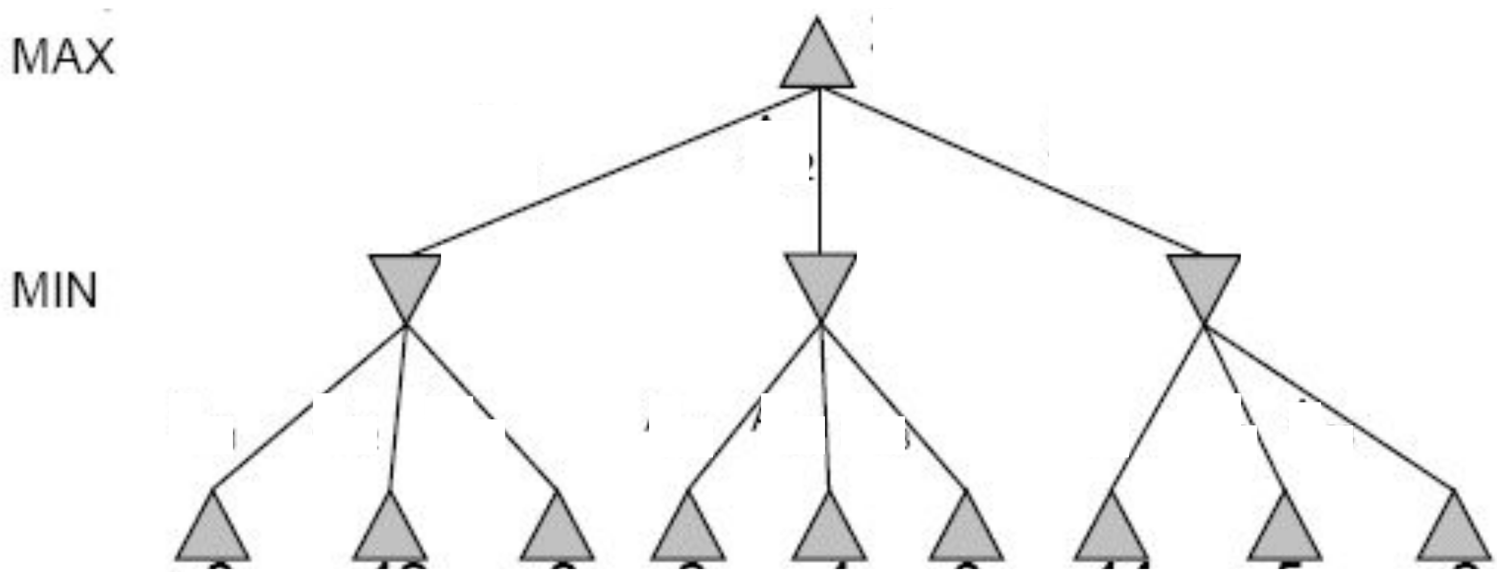
Minimax strategy: Look ahead and reason backwards

- The MAX(MIN) player selects the move that leads to the successor node with highest (lowest) score
- The scores are computed starting from the leaves of the tree and backing up their scores to their predecessors in accordance with the MiniMax strategy
- It explores each node of the tree

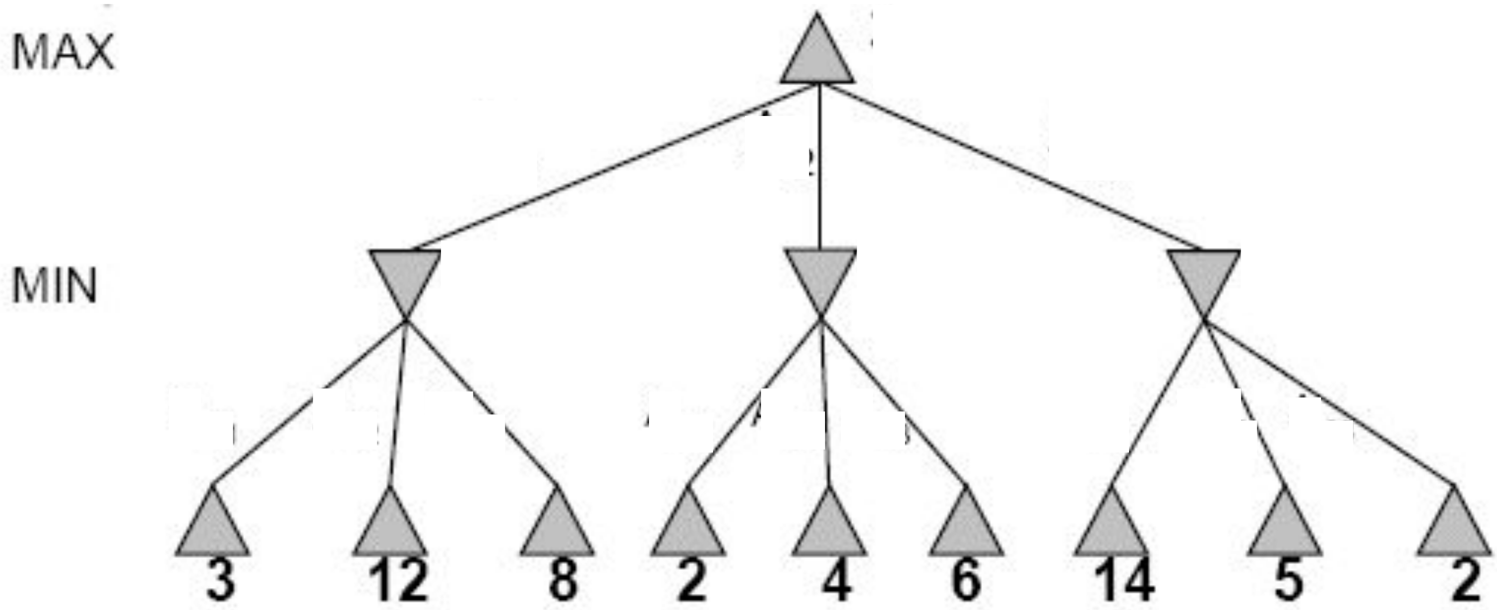
Size of search trees

- b = branching factor
- d = number of moves by both players
- Search tree is $O(b^d)$
- Chess
  - b ~ 35
  - D ~100
    - search tree is ~ 10^{154} (!!)
    - completely impractical to search this
- Game-playing emphasizes being able to make optimal decisions in a finite amount of time
  - Somewhat realistic as a model of a real-world agent
  - Even if games themselves are artificial

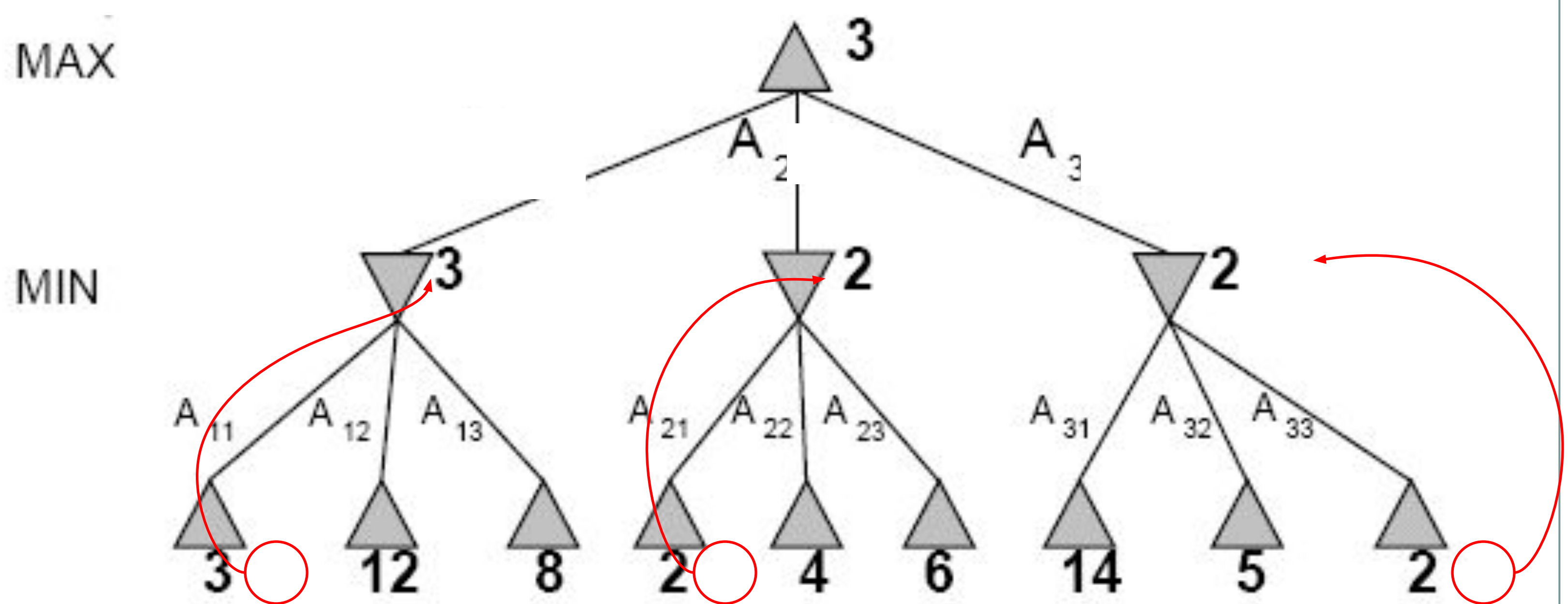
Two-Ply Game Tree



Two-Ply Game Tree

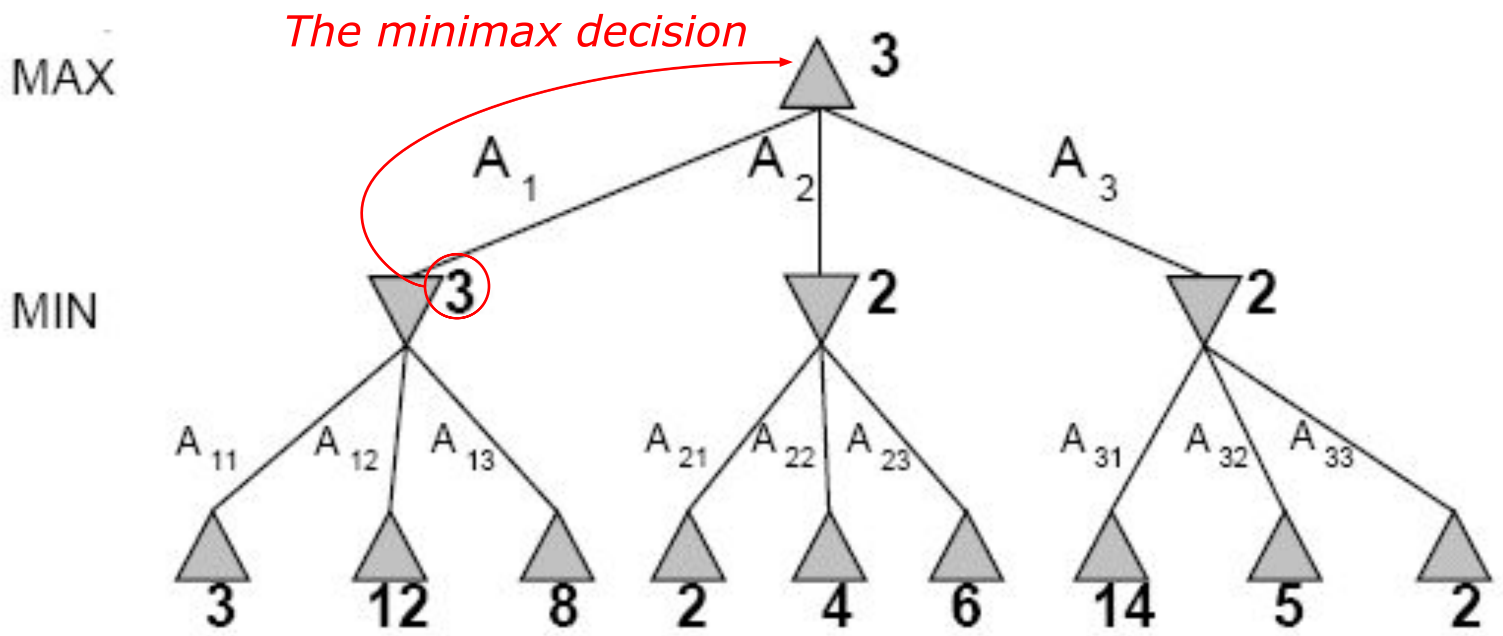


Two-Ply Game Tree



Two-Ply Game Tree

Minimax maximizes the utility for the worst-case outcome for max



Pseudocode for Minimax Algorithm

function MINIMAX-DECISION(*state*) **returns** *an action*
  inputs: *state*, current state in game
  $\boxed{v \leftarrow \text{MAX-VALUE}(state)}$
  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**
    $\boxed{v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s))}$
  return *v*

---

function MIN-VALUE(*state*) **returns** *a utility value*

  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)

  $v \leftarrow \infty$

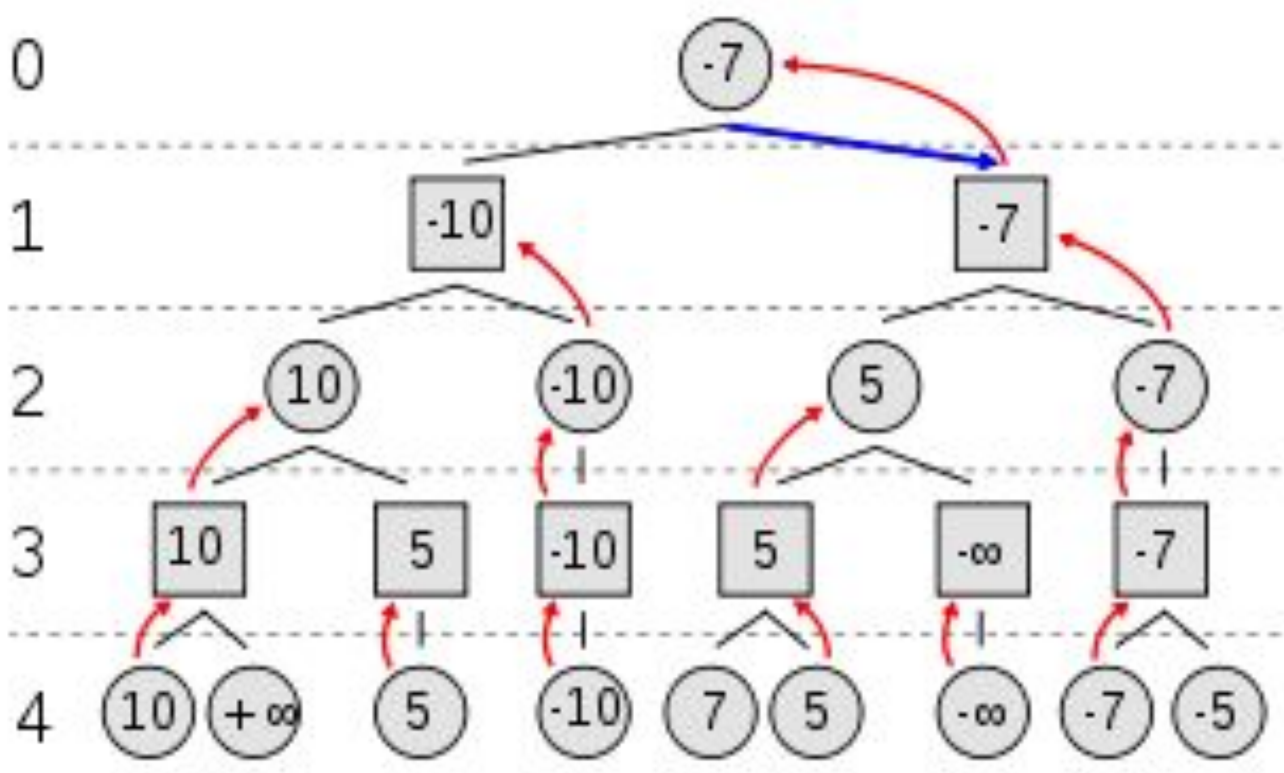
  for *a,s* in SUCCESSORS(*state*) **do**

    $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s))$

  return *v*

Example of Algorithm Execution

MAX to move



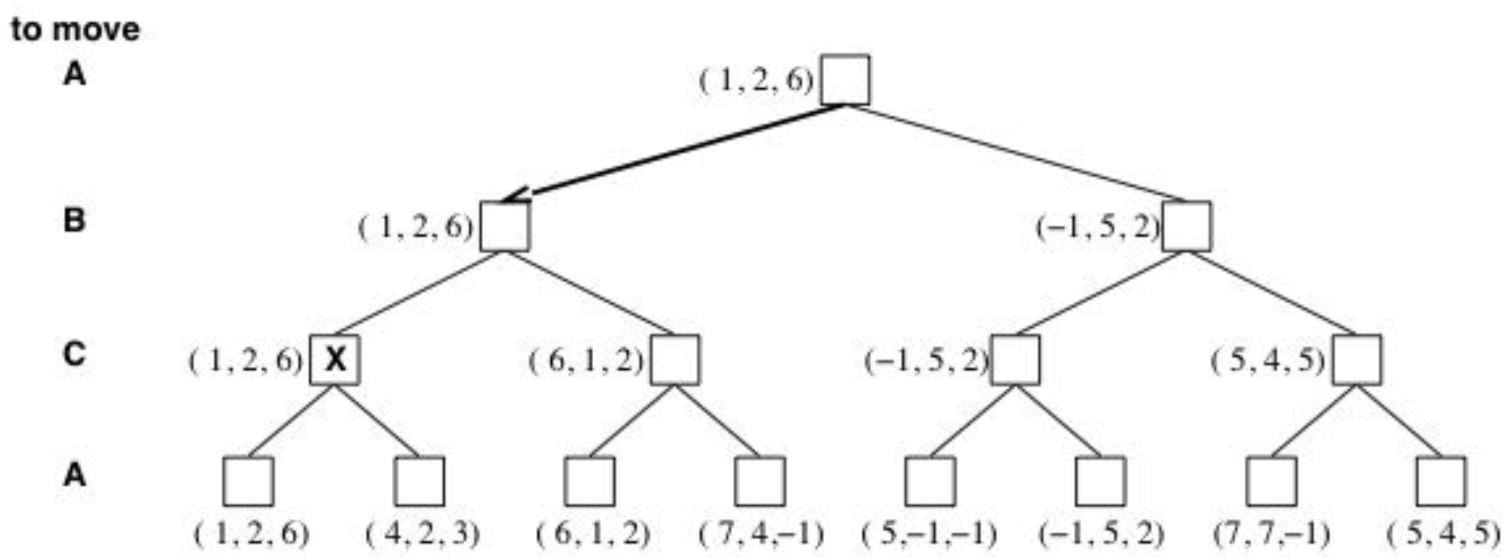
Minimax Algorithm

- Complete depth-first exploration of the game tree
- Assumptions:
  - Max depth = d, b legal moves at each point
  - E.g., Chess: d ~ 100, b ~35

Criterion	Minimax
Time ☹️	$O(b^d)$
Space ☺	$O(bd)$

Multiplayer games

- Games allow more than two players
- Single minimax values become vectors



Aspects of multiplayer games

- Previous slide (standard minimax analysis) assumes that each player operates to maximize only their own utility

- In practice, players make alliances
  - E.g, C strong, A and B both weak
  - May be best for A and B to attack C rather than each other

- If game is not zero-sum (i.e., utility(A) = - utility(B) then alliances can be useful even with 2 players
  - e.g., both cooperate to maximum the sum of the utilities

Practical problem with minimax search

- Number of game states is exponential in the number of moves.
  - Solution: Do not examine every node

    => pruning
    - Remove branches that do not influence final decision

- Revisit example …