CSE 473: Artificial Intelligence Spring 2014

Adversarial Search

Hanna Hajishirzi

Based on slides from Dan Klein, Luke Zettlemoyer

Many slides over the course adapted from either Stuart Russell or Andrew Moore

Outline

- Adversarial Search
  - Minimax search
  - α-β search
  - Evaluation functions

Game Playing State-of-the-Art

- **Checkers:** Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions. 2007: Checkers is now solved!
- **Chess:** Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue examined 200 million 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.
- **Othello:** Human champions refuse to compete against computers, which are too good.
- **Go:** Human champions are beginning to be challenged by machines, though the best humans still beat the best machines. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves, along with aggressive pruning.
- **Pacman:** unknown

General Game Playing

General Intelligence in Game-Playing Agents (GIGA'13)

(http://giga13.ru.is)

General Information

Artificial Intelligence (AI) researchers have for decades worked on building game-playing agents capable of matching wits with the strongest humans in the world, resulting in several success stories for games like chess and checkers. The success of such systems has been partly due to years of relentless knowledge-engineering effort on behalf of the program developers, manually adding application-dependent knowledge to their game-playing agents. The various algorithmic enhancements used are often highly tailored towards the game at hand.

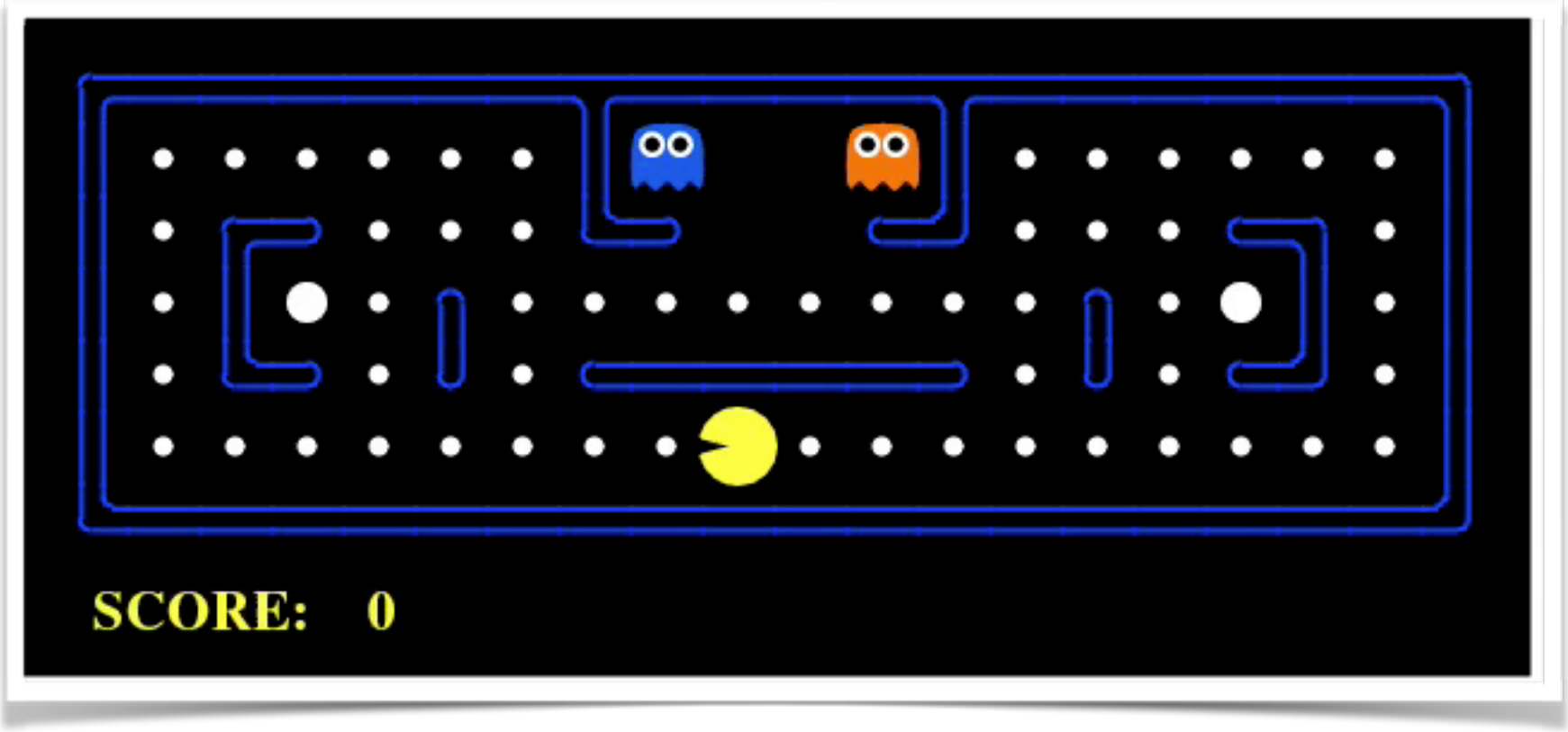
Research into general game playing (GGP) aims at taking this approach to the next level: to build intelligent software agents that can, given the rules of any game, automatically learn a strategy for playing that game at an expert level without any human intervention. In contrast to software systems designed to play one specific game, systems capable of playing arbitrary unseen games cannot be provided with game-specific domain knowledge a priori. Instead, they must be endowed with high-level abilities to learn strategies and perform abstract reasoning. Successful realization of such programs poses many interesting research challenges for a wide variety of artificial-intelligence sub-areas including (but not limited to):

- knowledge representation and reasoning
- heuristic search and automated planning
- computational game theory
- multi-agent systems
- machine learning

The aim of this workshop is to bring together researchers from the above sub-fields of AI to discuss how best to address the challenges of and further advance the state-of-the-art of general game-playing systems and generic artificial intelligence.

The workshop is one-day long and will be held onsite at IJCAI during the scheduled workshop period August 3rd-5th (exact day is to be announced later).

Adversarial Search



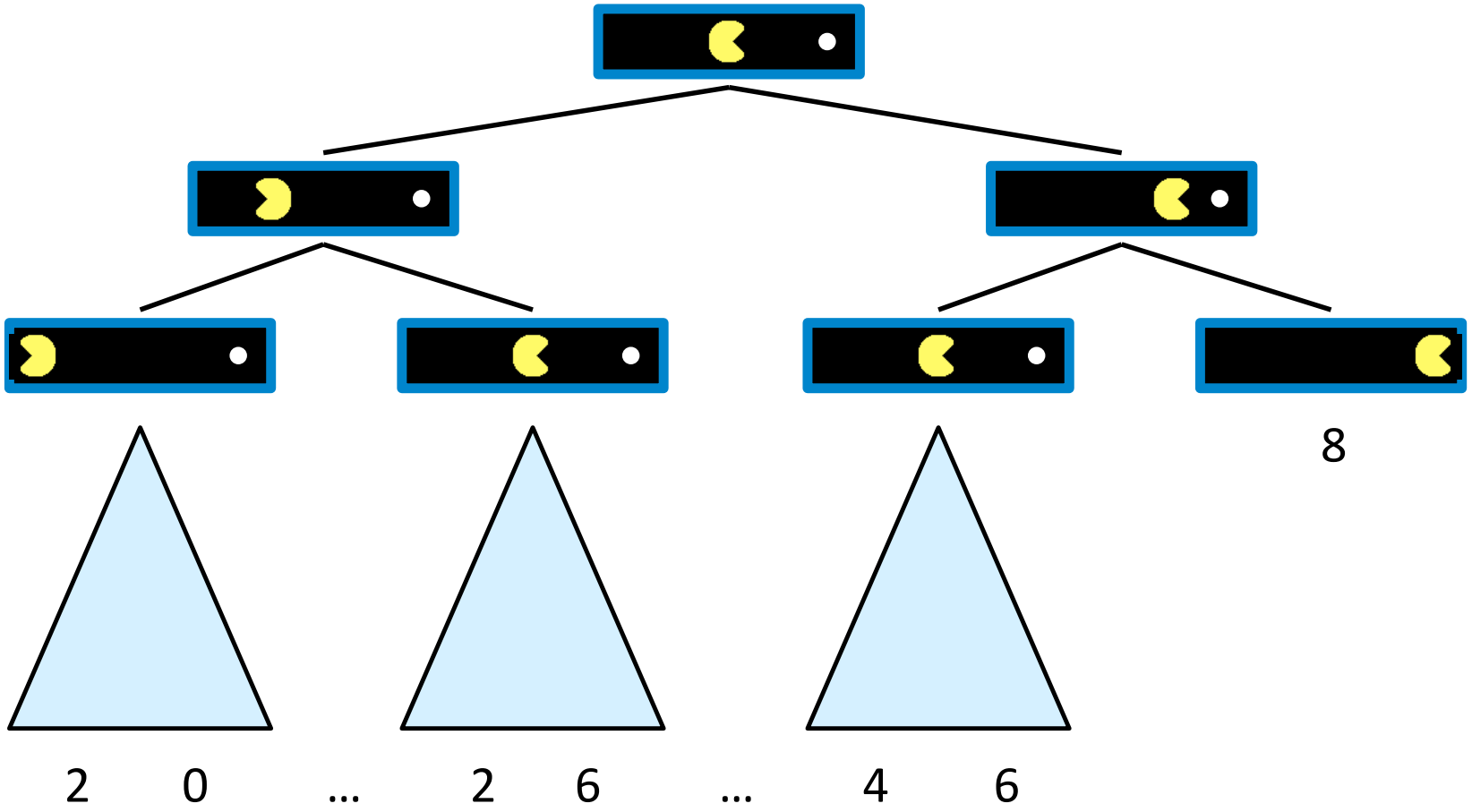
Game Playing

- Many different kinds of games!
- Choices:
  - Deterministic or stochastic?
  - One, two, or more players?
  - Perfect information (can you see the state)?
- Want algorithms for calculating a strategy (policy) which recommends a move in each state

Deterministic Games

- Many possible formalizations, one is:
  - States: S (start at s_0)
  - Players: P={1...N} (usually take turns)
  - Actions: A (may depend on player / state)
  - Transition Function: S x A → S
  - Terminal Test: S → {t,f}
  - Terminal Utilities: S x P → R
- Solution for a player is a policy: S → A

Single-Agent Trees

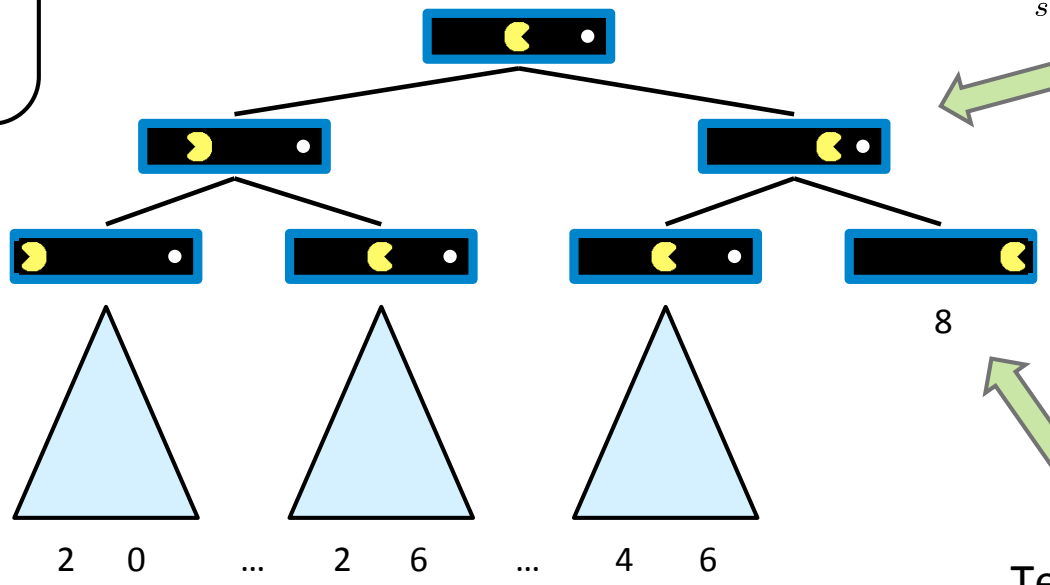


Value of States

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

Non-Terminal States:

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

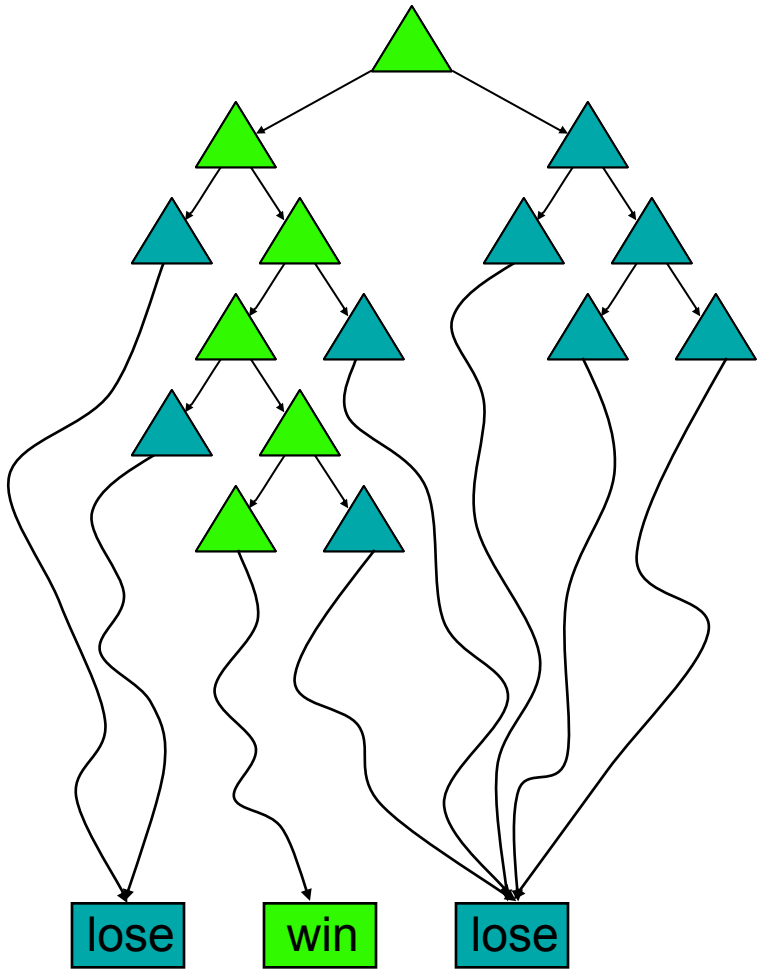


Terminal States:

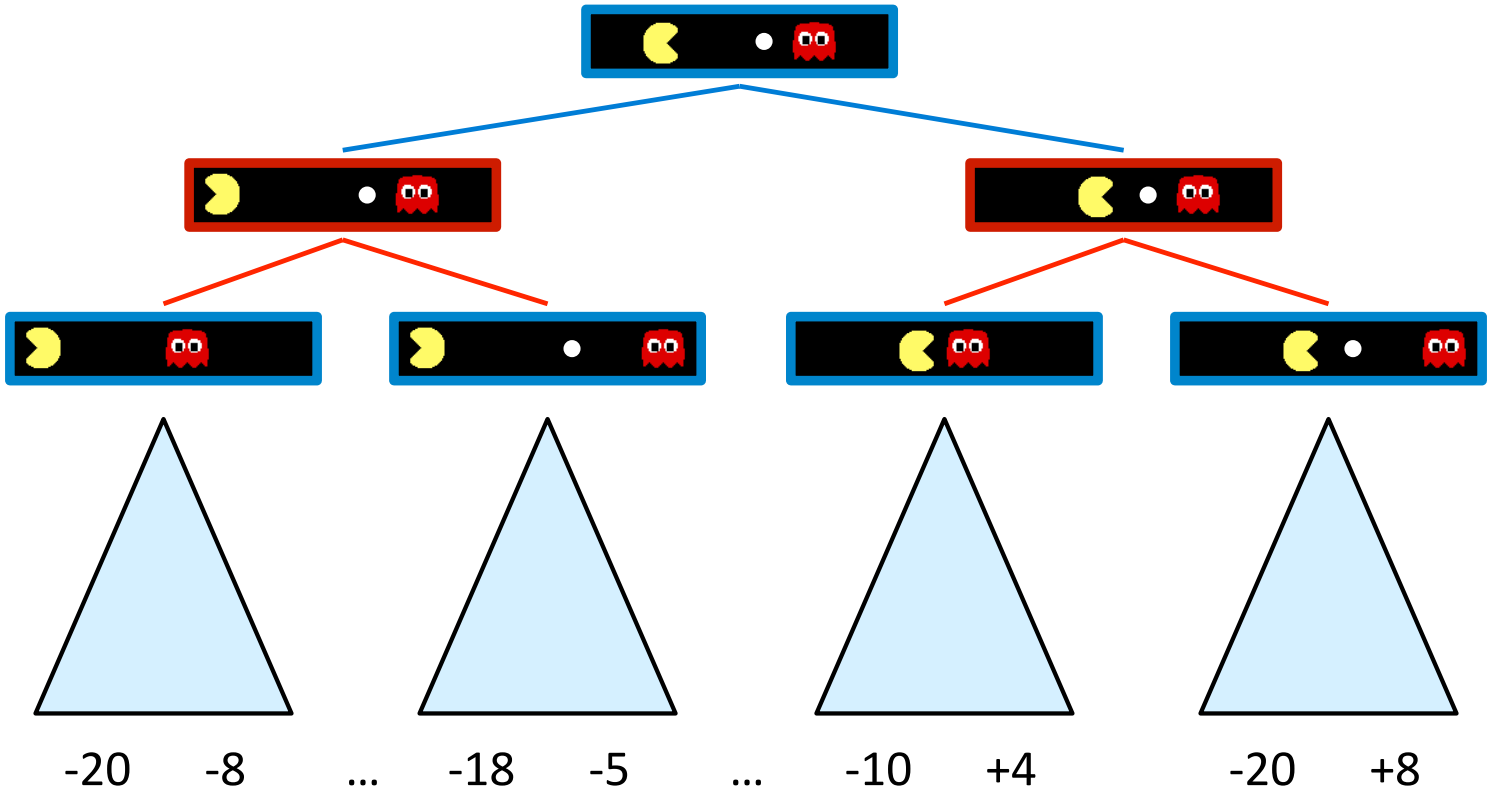
$$V(s) = \text{known}$$

Deterministic Single-Player

- Deterministic, single player, perfect information:
  - Know the rules, action effects, winning states
  - E.g. Freecell, 8-Puzzle, Rubik's cube
- … it's just search!
- Slight reinterpretation:
  - Each node stores a value: the best outcome it can reach
  - This is the maximal outcome of its children (the max value)
  - Note that we don't have path sums as before (utilities at end)
- After search, can pick move that leads to best node



Adversarial Game Trees



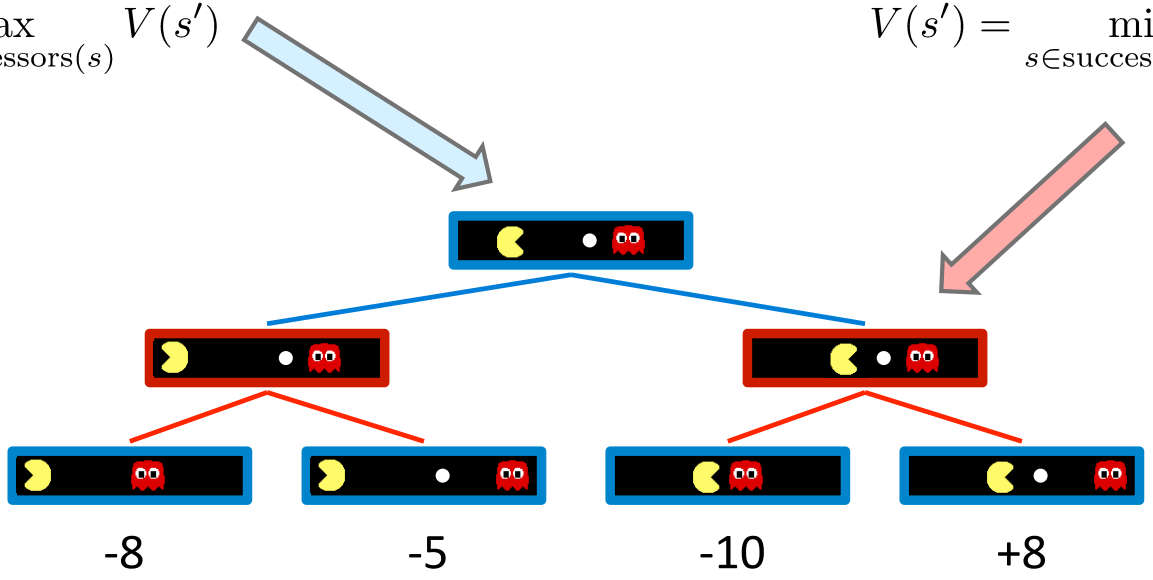
Minimax Values

States Under Agent's Control:

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

States Under Opponent's Control:

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

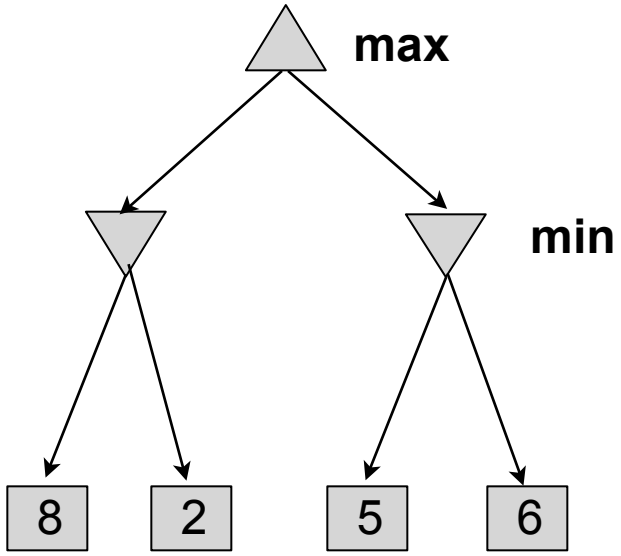


Terminal States:

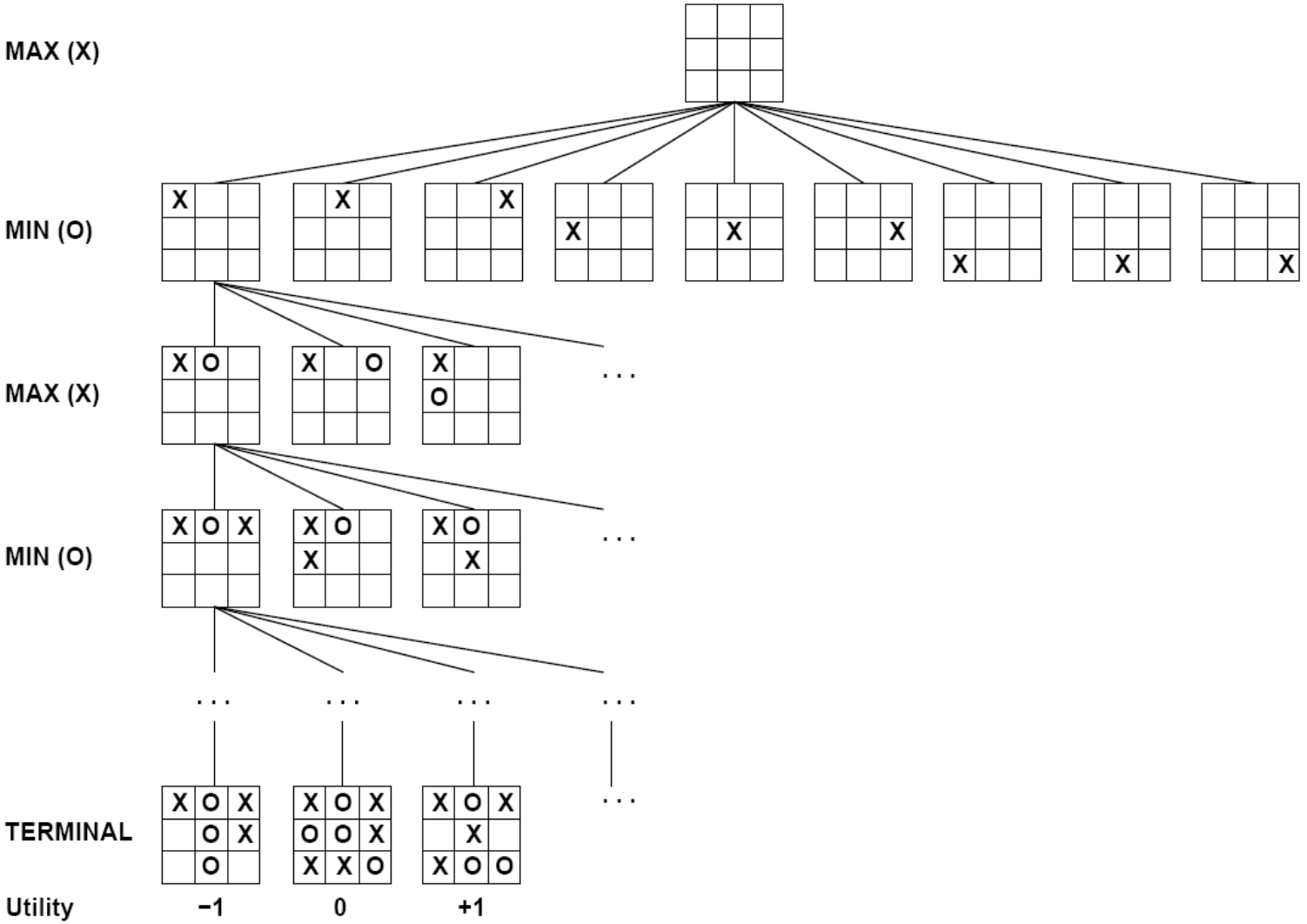
$$V(s) = \text{known}$$

Deterministic Two-Player

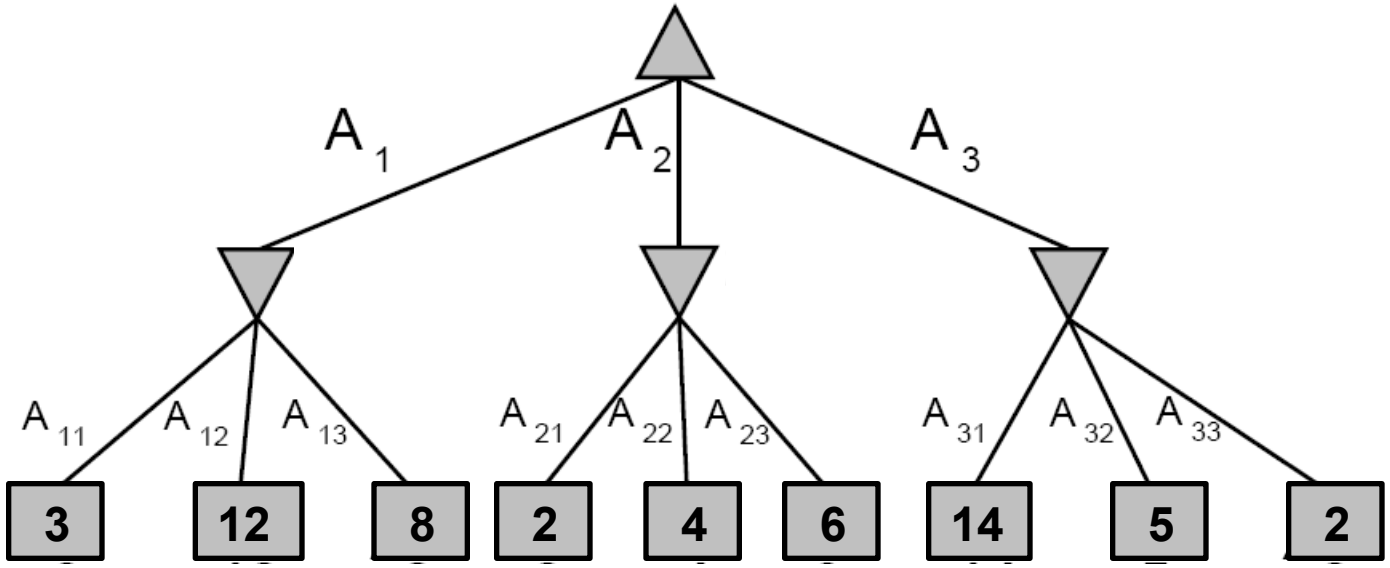
- E.g. tic-tac-toe, chess, checkers
- Zero-sum games
  - Agents have opposite utilities
  - One player maximizes result
  - The other minimizes result
- **Minimax search**
  - A state-space search tree
  - Players alternate
  - Choose move to position with highest minimax value = best achievable utility against best play



Tic-tac-toe Game Tree



Minimax Example



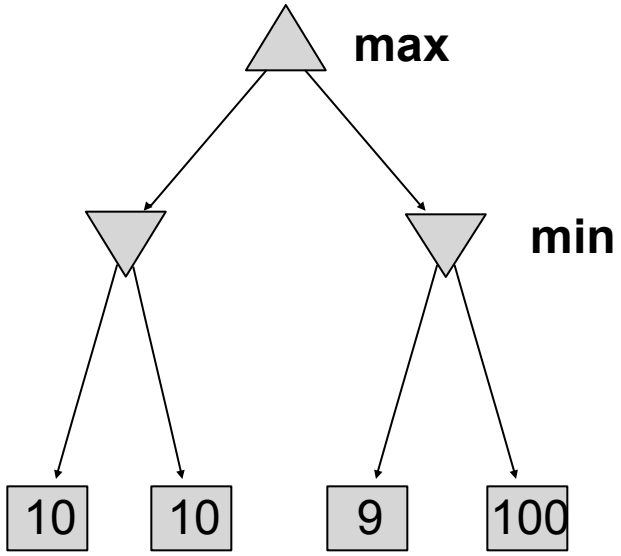
Minimax Search

```
function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do v ← MAX(v, MIN-VALUE(s))
    return v
```

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

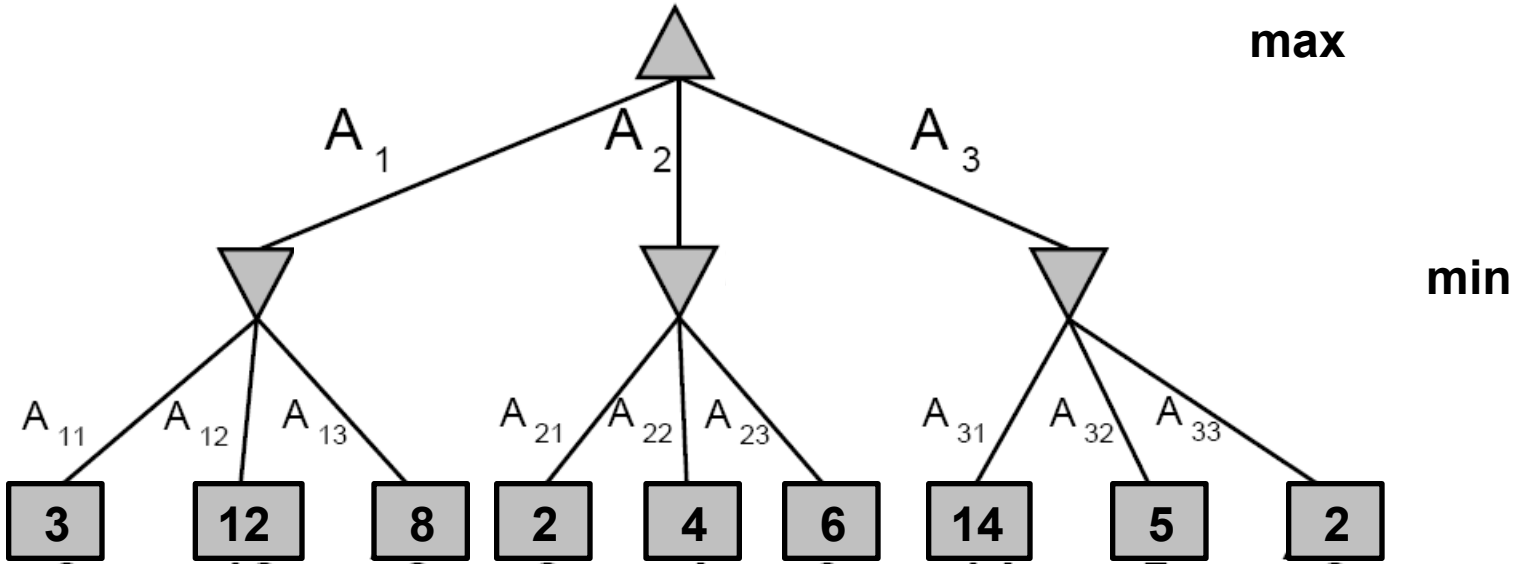
Minimax Properties

- Optimal?
  - Yes, against perfect player. Otherwise?
- Time complexity?
  - $O(b^m)$
- Space complexity?
  - O(bm)

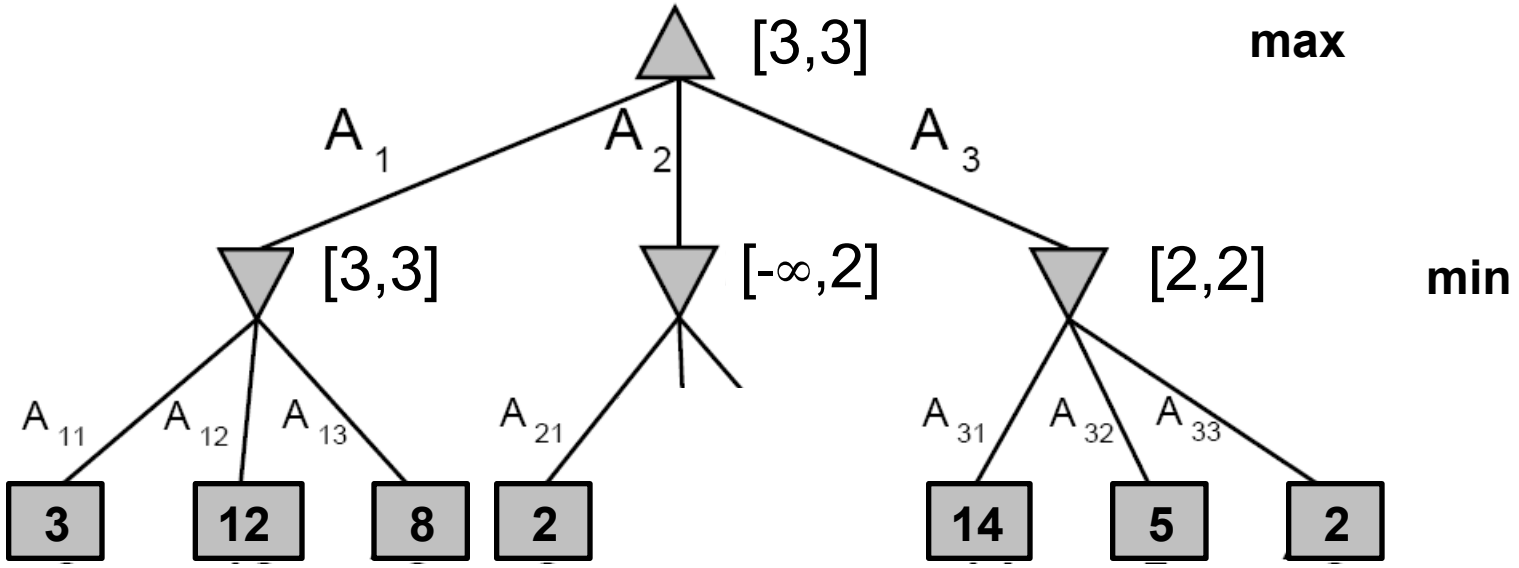


- For chess, $b \approx 35$, $m \approx 100$
  - Exact solution is completely infeasible
  - But, do we need to explore the whole tree?

Can we do better?

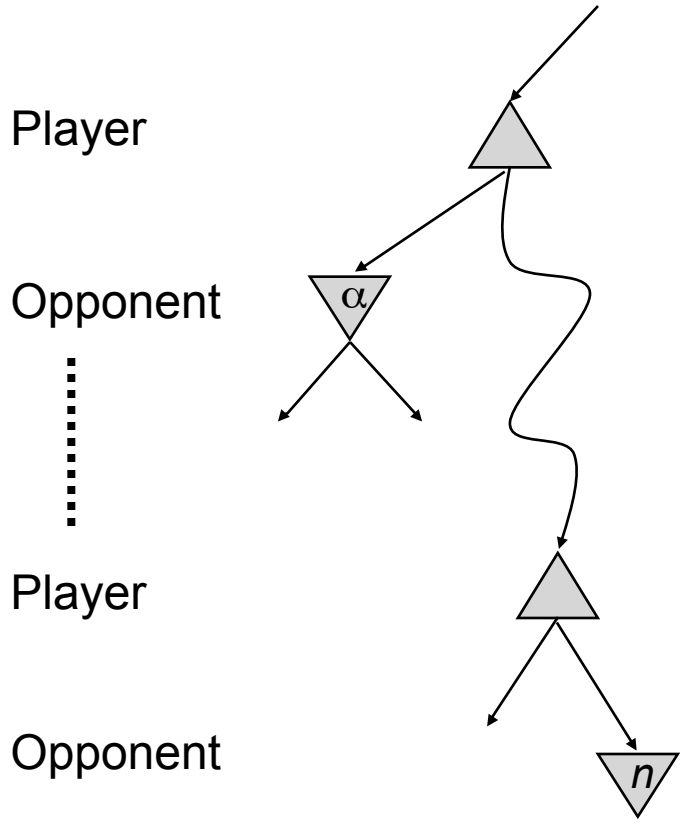


α-β Pruning Example

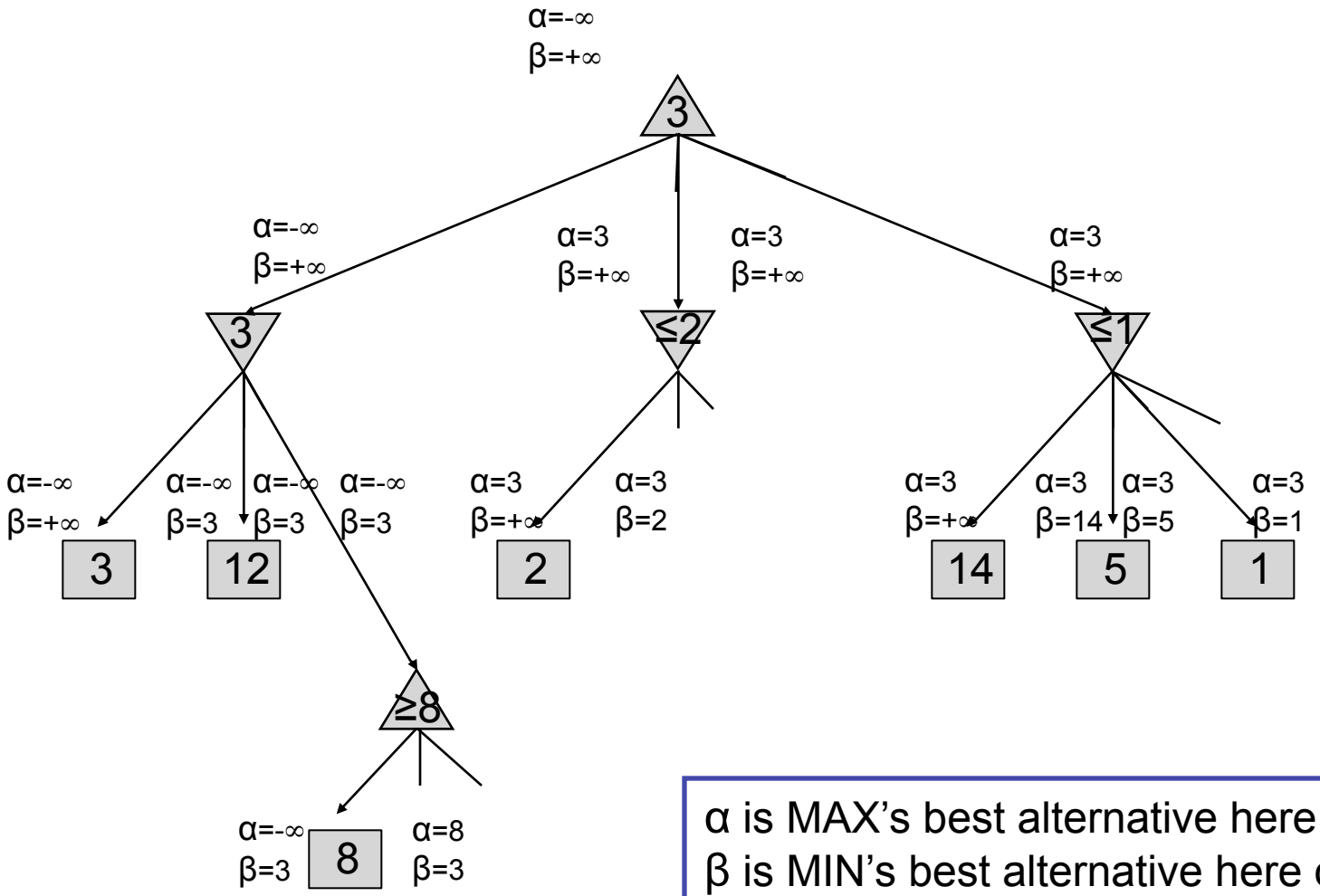


α-β Pruning

- General configuration
  - α is the best value that MAX can get at any choice point along the current path
  - If *n* becomes worse than α, MAX will avoid it, so can stop considering *n*'s other children
  - Define β similarly for MIN



Alpha-Beta Pruning Example



α is MAX's best alternative here or above
β is MIN's best alternative here or above

Alpha-Beta Pseudocode

inputs: *state*, current game state
α, value of best alternative for MAX on path to *state*
β, value of best alternative for MIN on path to *state*
returns: *a utility value*

```
function MAX-VALUE(state,α,β)
    if TERMINAL-TEST(state) then
        return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s,α,β))
        if v ≥ β then return v
        α ← MAX(α,v)
    return v
```

```
function MIN-VALUE(state,α,β)
    if TERMINAL-TEST(state) then
        return UTILITY(state)
    v ← +∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s,α,β))
        if v ≤ α then return v
        β ← MIN(β,v)
    return v
```

Announcements

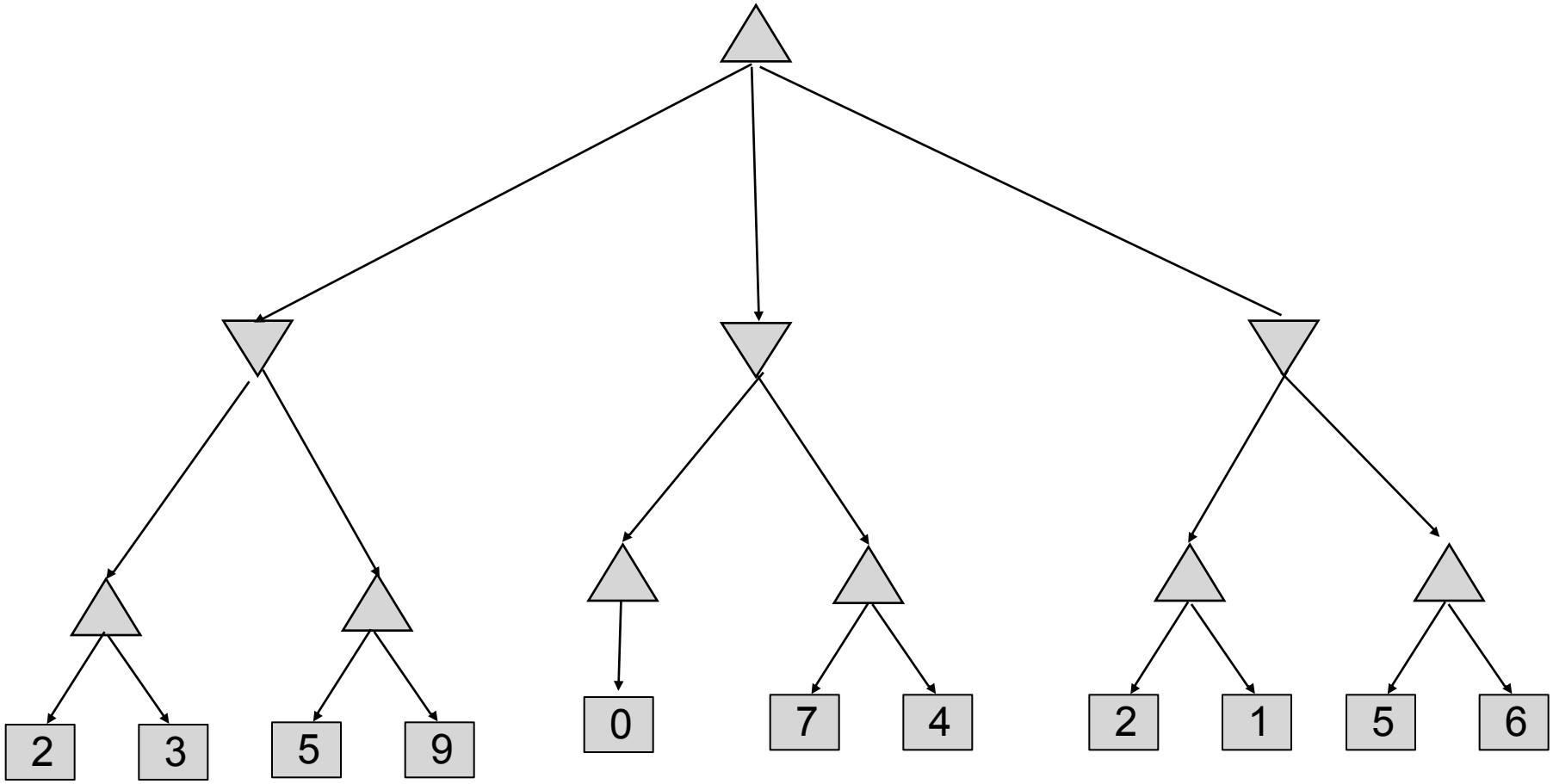
- PS1 is due in a week.
- PS2 is on games, will be released next week.

Adversarial Search (Recap)

- Max tree: 1-player game
- Minimax trees
- Alpha-beta prunning

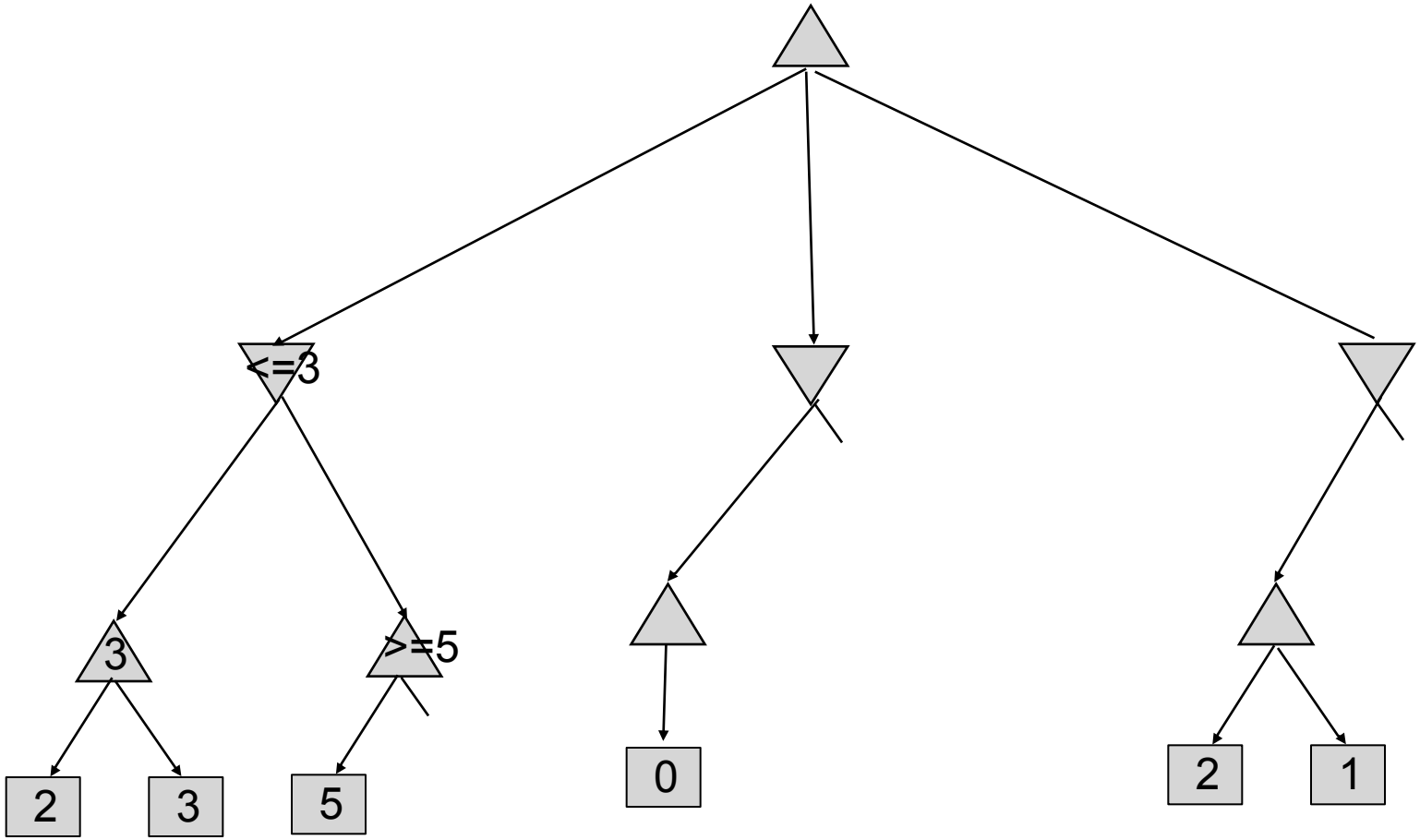
- Today:
  - Evaluation function
  - Expectimax

Alpha-Beta Pruning Example



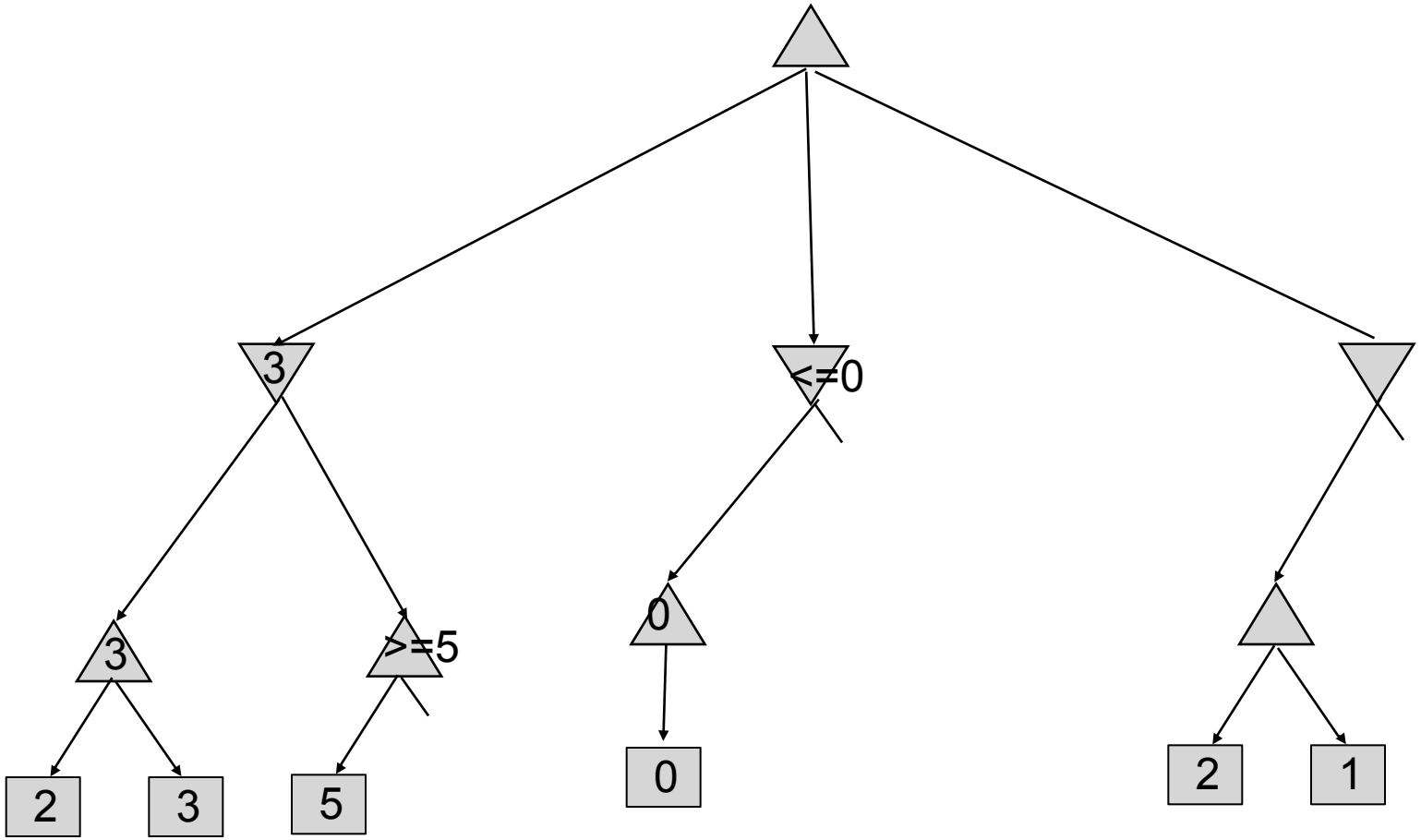
α is MAX's best alternative here or above
β is MIN's best alternative here or above

Alpha-Beta Pruning Example



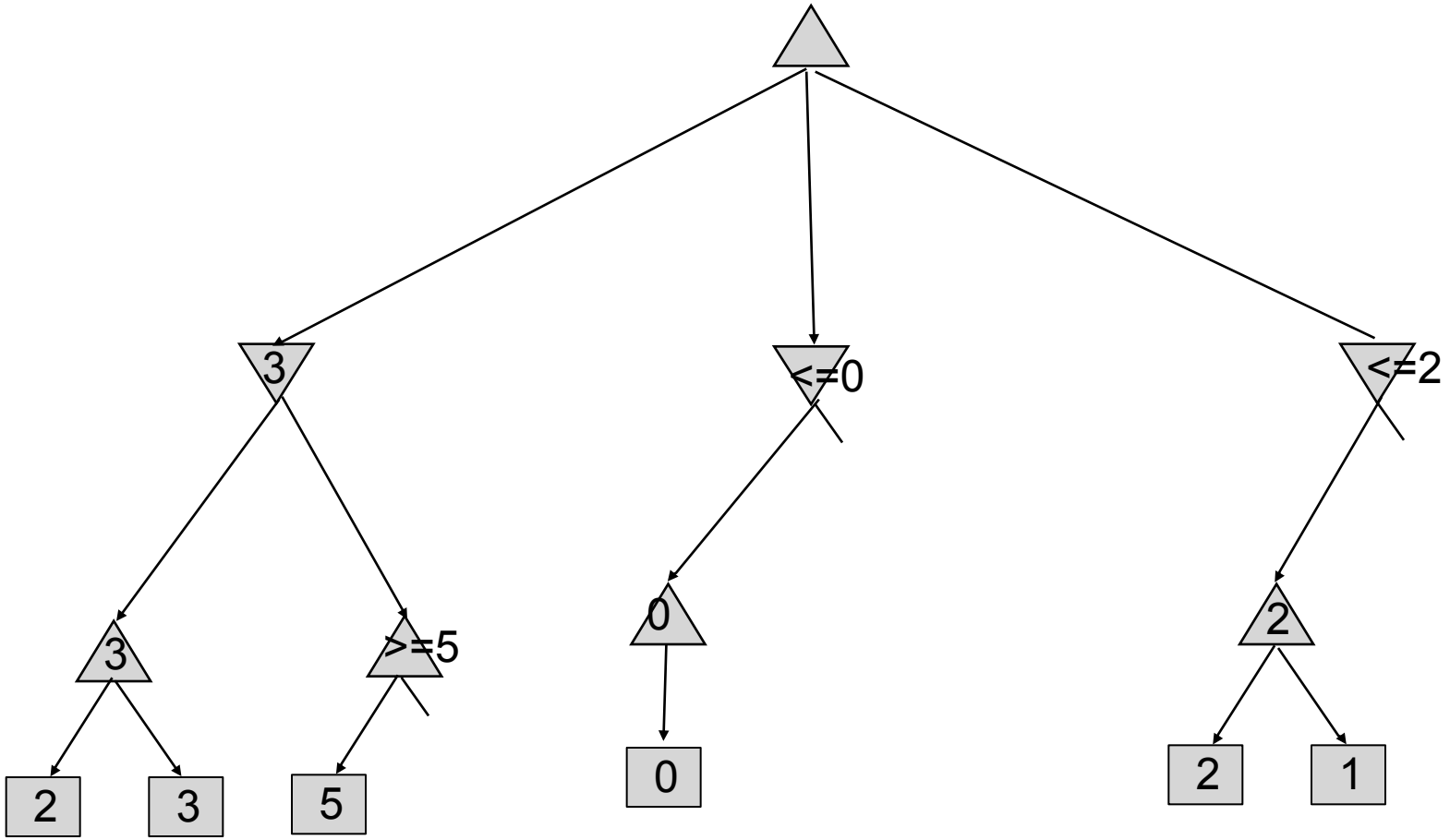
α is MAX's best alternative here or above
β is MIN's best alternative here or above

Alpha-Beta Pruning Example



α is MAX's best alternative here or above
β is MIN's best alternative here or above

Alpha-Beta Pruning Example



α is MAX's best alternative here or above
β is MIN's best alternative here or above

Alpha-Beta Pruning Properties

- This pruning has no effect on final result at the root
- Values of intermediate nodes might be wrong!
  - but, they are bounds
- Good child ordering improves effectiveness of pruning
- With "perfect ordering":
  - Time complexity drops to $O(b^{m/2})$
  - Doubles solvable depth!
  - Full search of, e.g. chess, is still hopeless…