

Adversarial Search

Chapter 6

Sections 1 – 3 (and a little of the
rest)

Outline

- Games
- Optimal decisions
- α - β pruning
- Imperfect, real-time decisions (briefly)

Game Search

- Game-playing programs developed by AI researchers since the beginning of the modern AI era (chess, checkers in 1950s)
- **Game Search**
 - Sequences of player's decisions *we control*
 - Decision of other player(s) *we do not control*
- **Contingency problem:** many possible opponent's moves must be “covered” by the solution
 - Introduces uncertainty to the game since we do not know what the opponent will do
- **Rational opponent:** maximizes it's own *utility* function

Types of Game Problems

- **Adversarial**
 - Win of one player is a loss of the other
 - Focus of this course
- Cooperative
 - Players have common interests and utility function
- A spectrum of others in between

Games vs. search problems

- Adversarial search or “games”
 - Multi-agent and competitive environment
 - "Unpredictable" opponent → specifying a move for every possible opponent reply
- Time limits → unlikely to find goal, must approximate

Typical AI “Games:

- Deterministic and Fully Observable Environment
- Two agents with turn-taking for actions
- Zero-sum (adversarial)
- Abstract (robotic soccer notable exception)
 - state easy to represent, few action choices, well-defined goals
 - hard to solve

Game Search

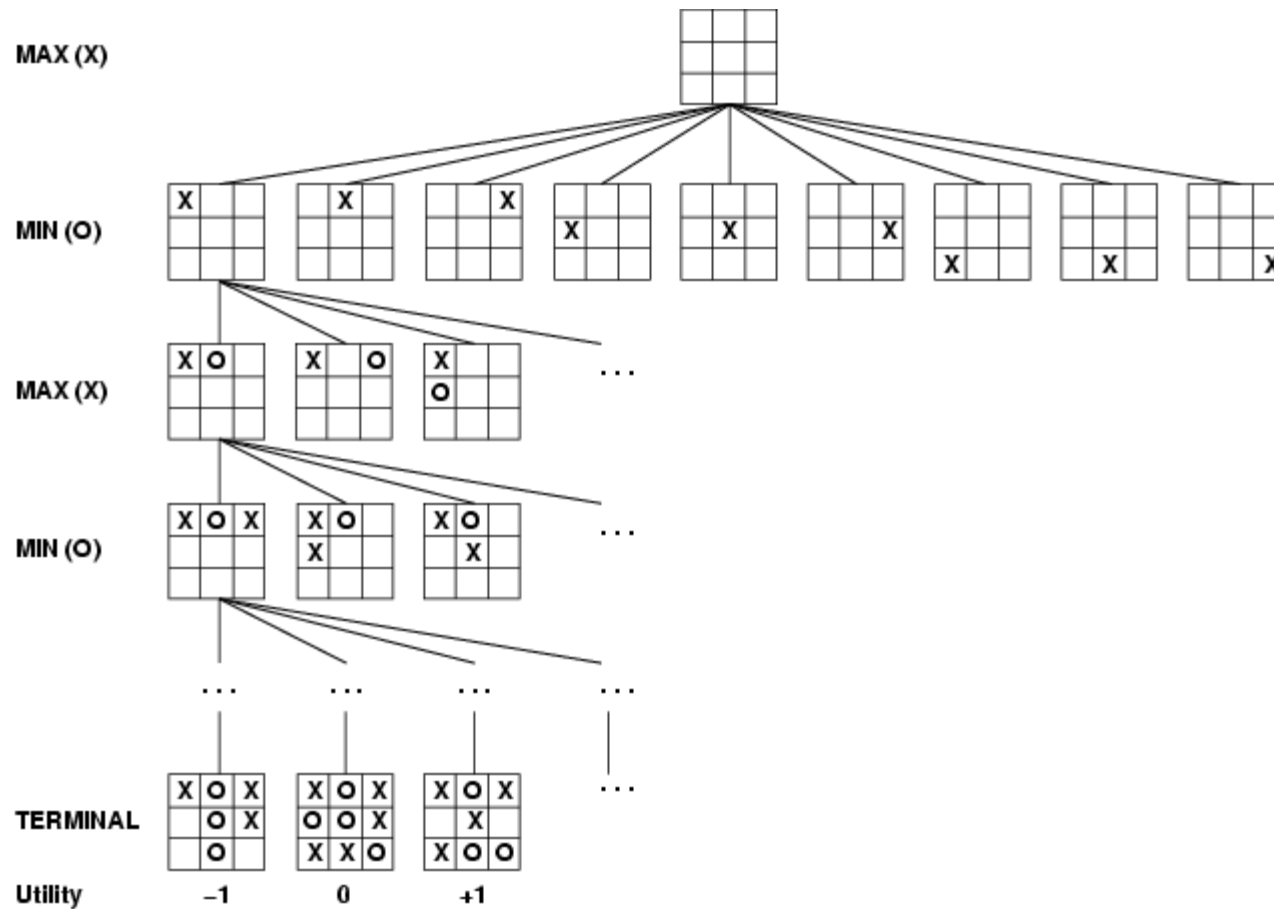
- Problem Formulation
 - **Initial state**: initial board position + information about whose move it is
 - **Successors**: legal moves a player can make
 - **Goal (terminal test)**: determines when the game is over
 - **Utility function**: measures the outcome of the game and its desirability
- Search objective
 - Find the sequence of player's decisions (moves) maximizing its utility
 - Consider the opponent's moves and their utility

Game Tree

- Initial State and Legal Moves for Each Side

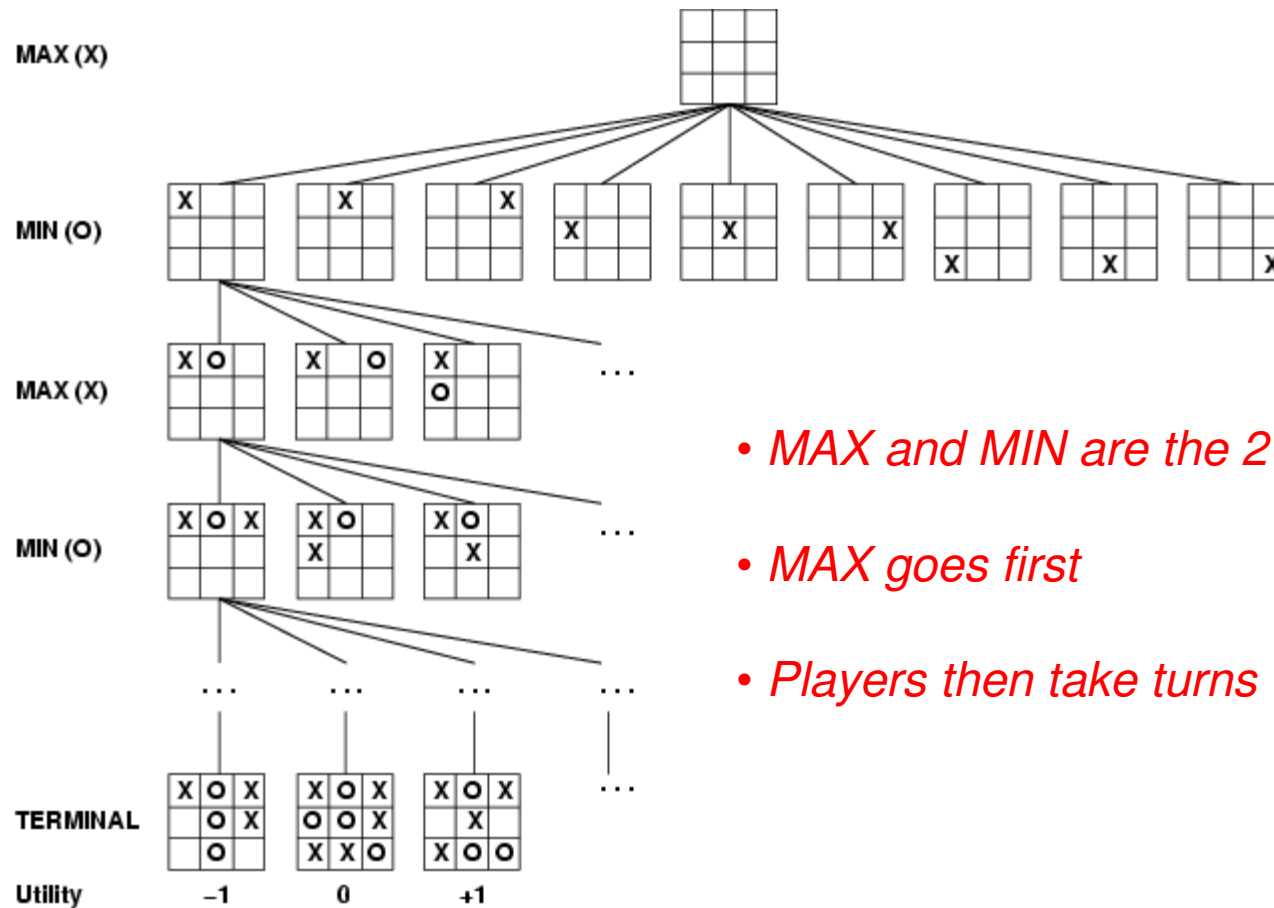
Game Tree

(2-player, deterministic, turns)



Game Tree

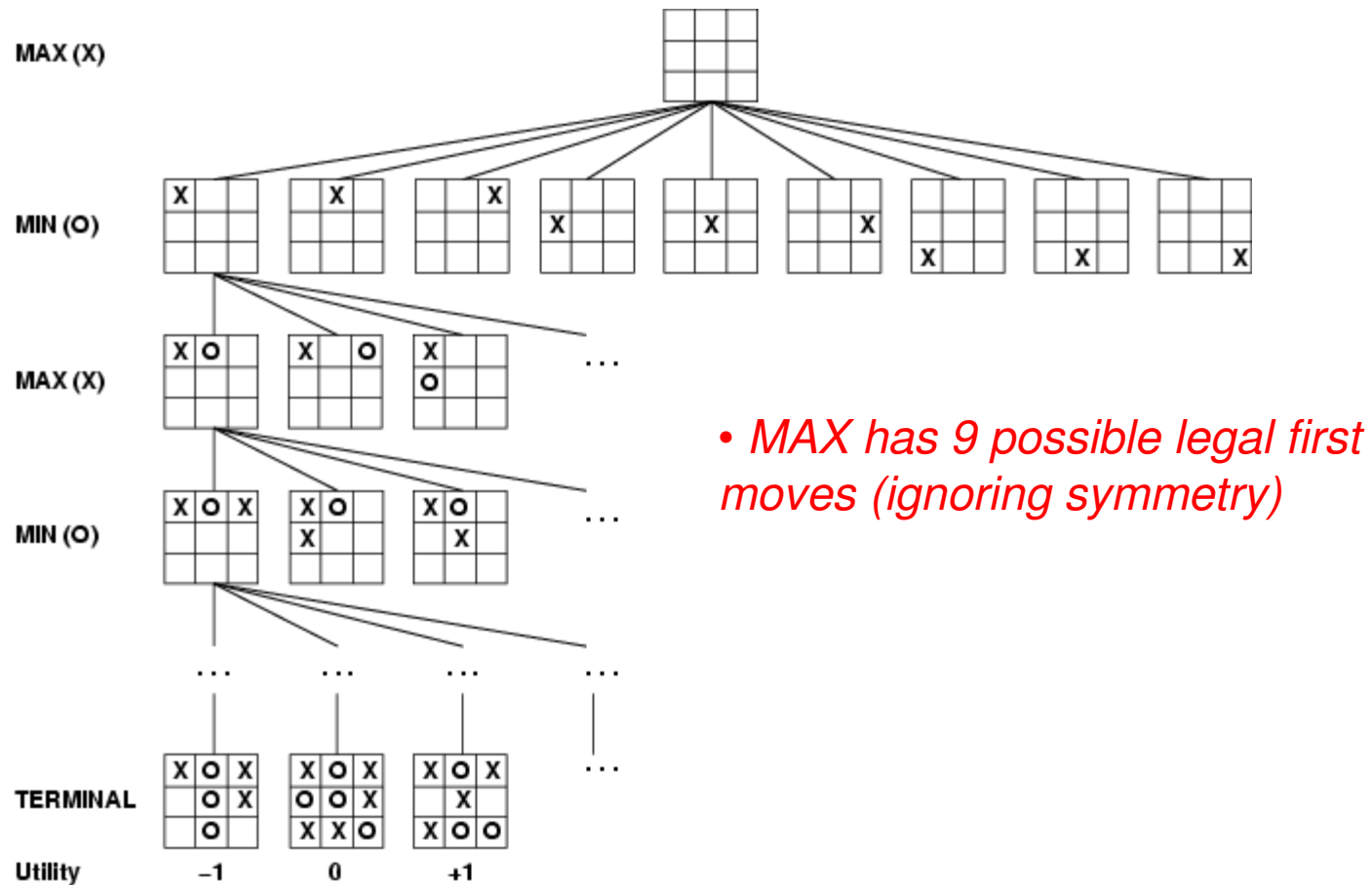
(2-player, deterministic, turns)



- MAX and MIN are the 2 players
- MAX goes first
- Players then take turns

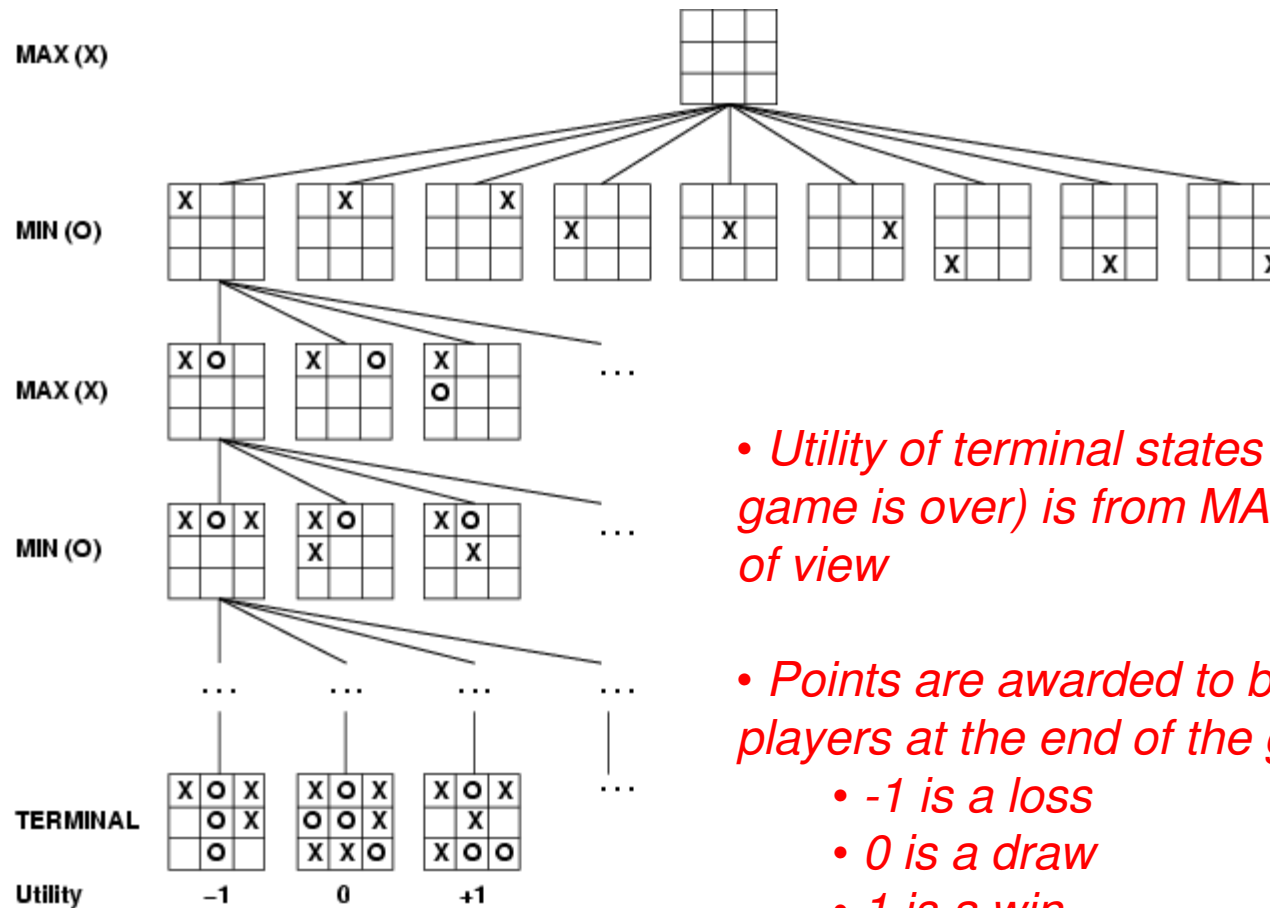
Game Tree

(2-player, deterministic, turns)



Game Tree

(2-player, deterministic, turns)



- *Utility of terminal states (when game is over) is from MAX's point of view*
- *Points are awarded to both players at the end of the game*
 - *-1 is a loss*
 - *0 is a draw*
 - *1 is a win*

Minimax Algorithm

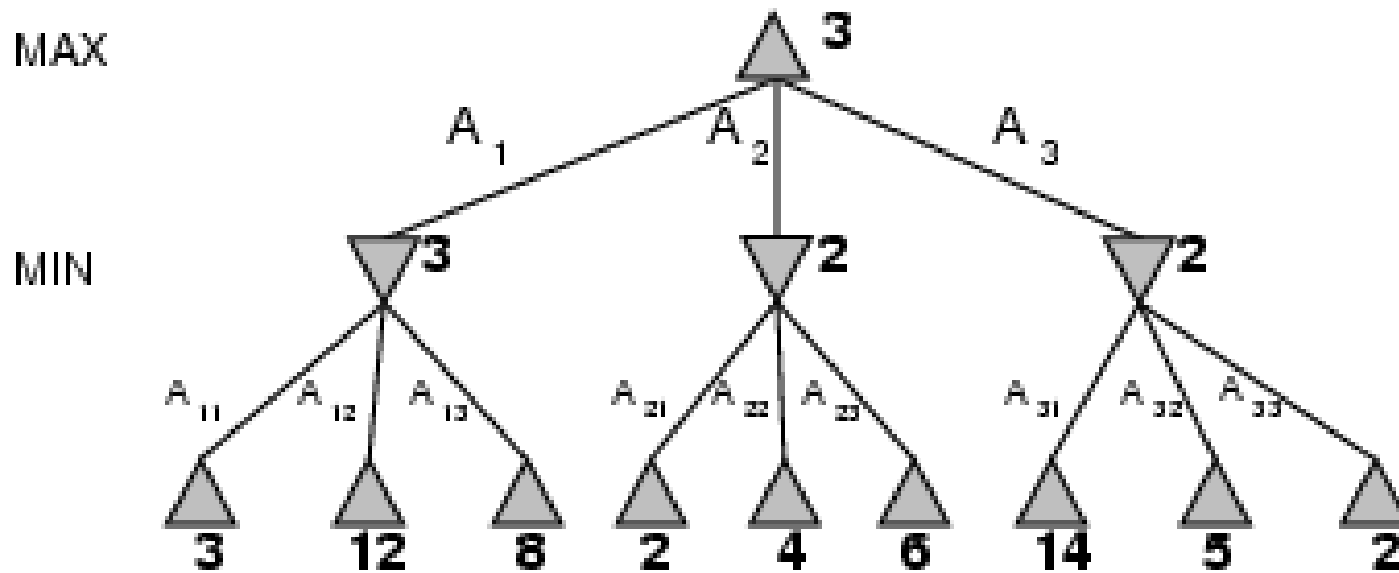
- How do we deal with the contingency problem?
 - Assuming that the opponent is rational and always optimizes its behavior (opposite to us), we consider the opponent's best response
 - Then the minimax algorithm determines the best move

Minimax

- Finds an optimal (contingent) strategy, assuming perfect play for deterministic games
- Idea: choose move to position with highest **MINIMAX VALUE**
= best achievable payoff against best play
- MINIMAX-VALUE (n)
 - UTILITY (n) if n is a terminal state
 - \max_s MINIMAX-VALUE (s) if n is a MAX node
 - \min_s MINIMAX-VALUE (s) if n is a MIN node(where s is an element of the successors of n)

Minimax Example

- E.g., 2-ply game (with utility values at the leaves)



Another Example

- In class

Minimax algorithm

function MINIMAX-DECISION(*state*) *returns an action*

$v \leftarrow \text{MAX-VALUE}(\textit{state})$

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**

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

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*

Notes: recursive (backs up from leaves), depth-first

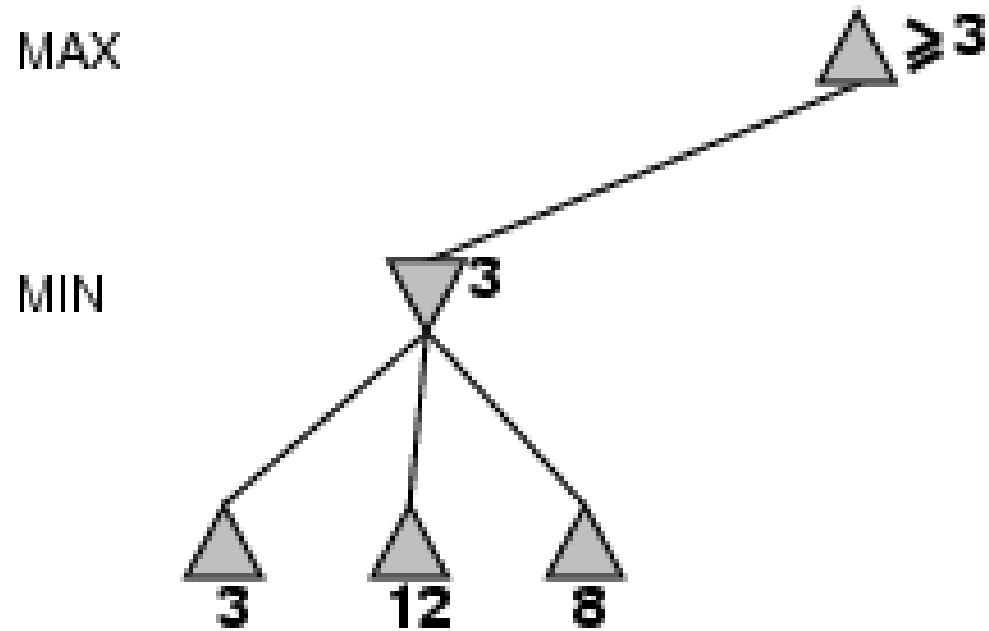
Properties of minimax

- Complete? Yes (if tree is finite)
- Optimal? Yes (against an optimal opponent)
- Time complexity? $O(b^m)$
- Space complexity? $O(bm)$ (depth-first exploration)
- For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
→ exact solution completely infeasible
- Do we really need to explore every path???

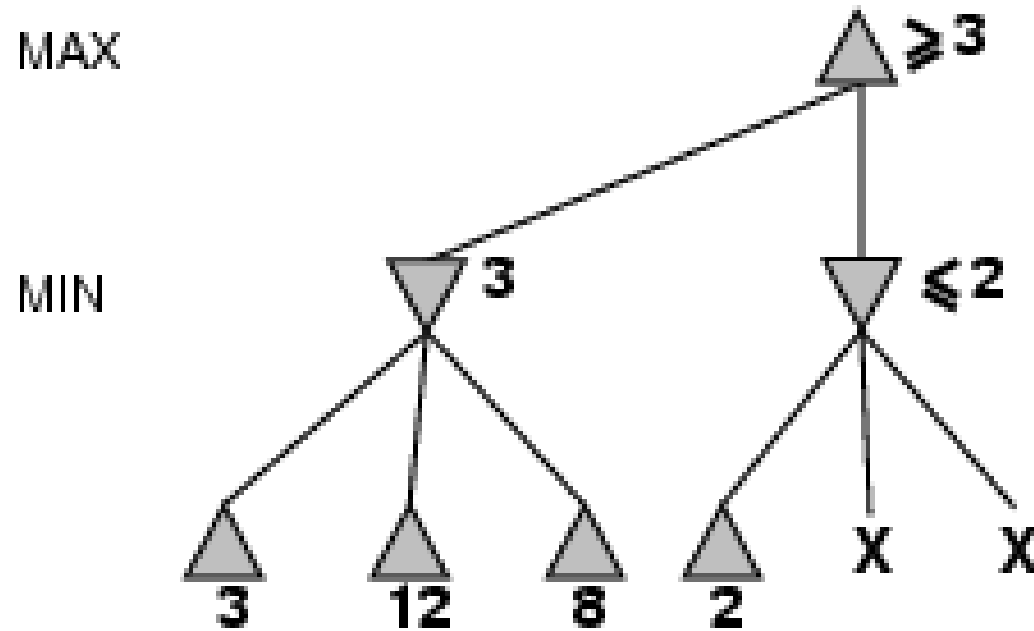
Solutions to the Complexity Problem

- Dynamic pruning of redundant branches of the search tree
 - Some branches will never be played by rational players since they include sub-optimal decisions (for either player)
 - Identify a provably suboptimal branch of the search tree before it is fully explored
 - Eliminate the suboptimal branch
 - Procedure: Alpha-Beta Pruning
- Early cutoff of the search tree
 - Use imperfect minimax value estimate of non-terminal states

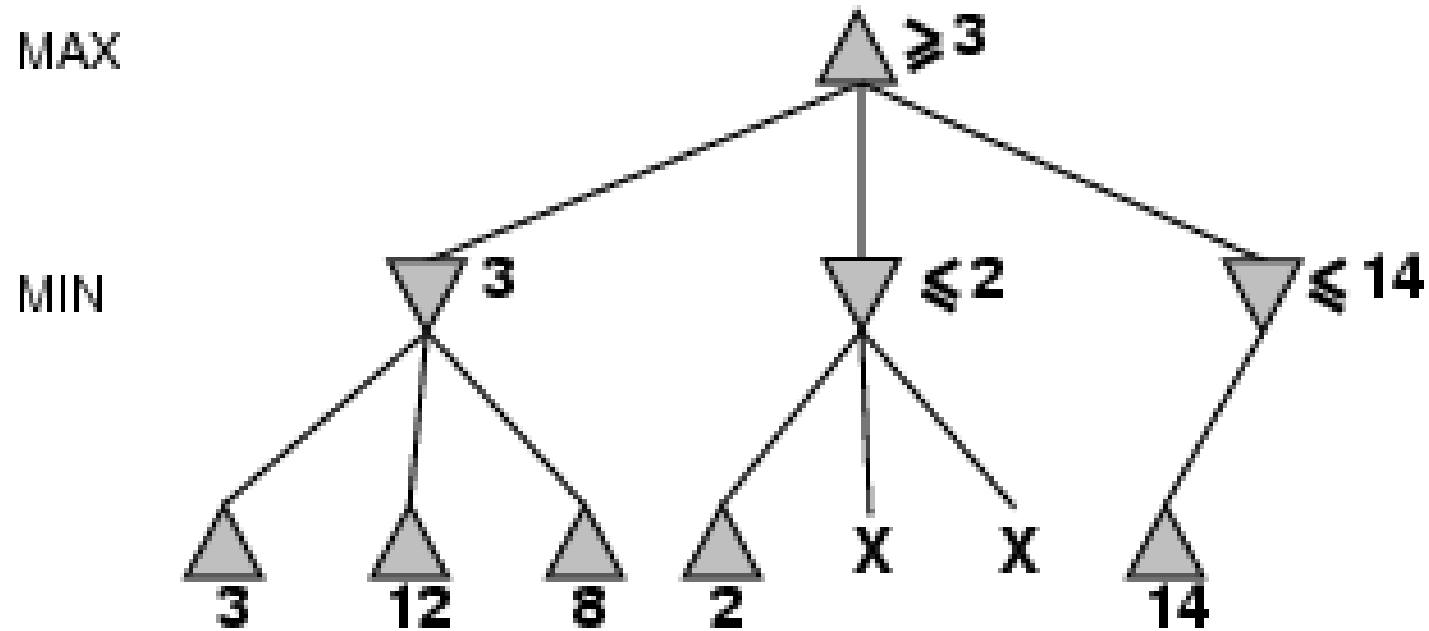
α - β pruning example



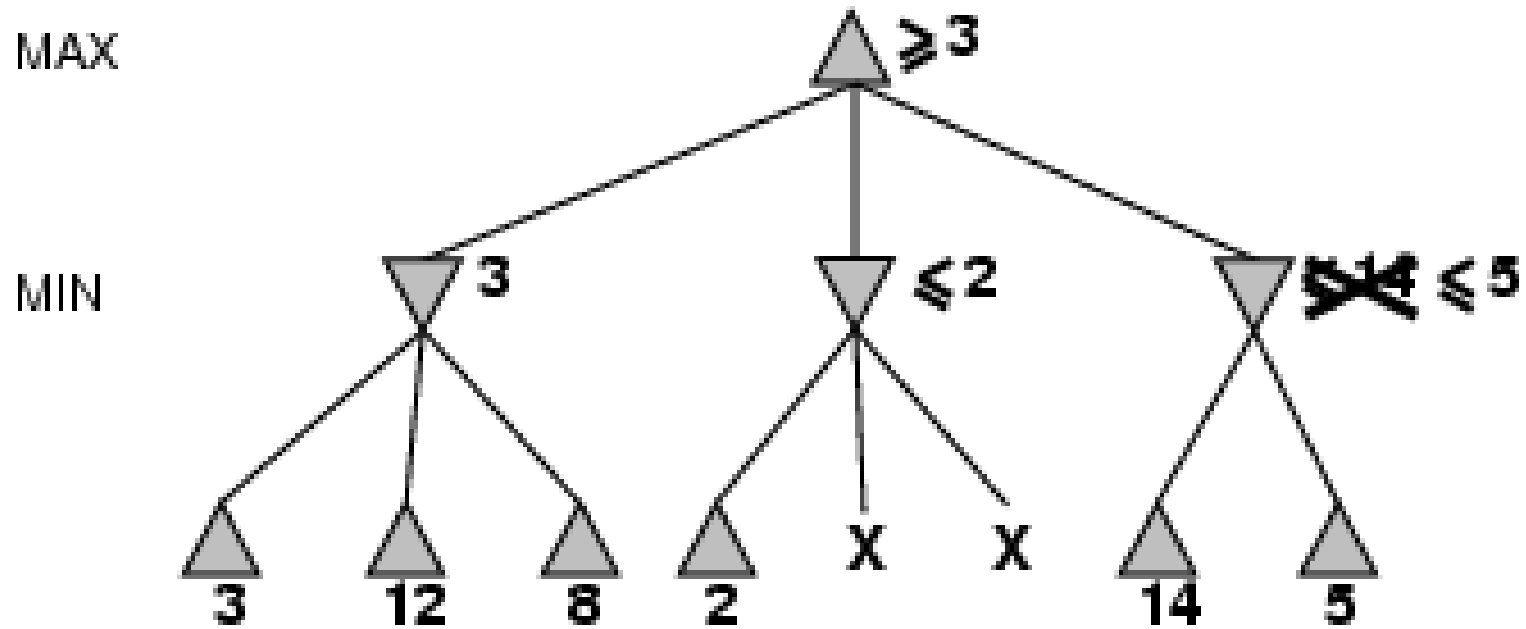
α - β pruning example



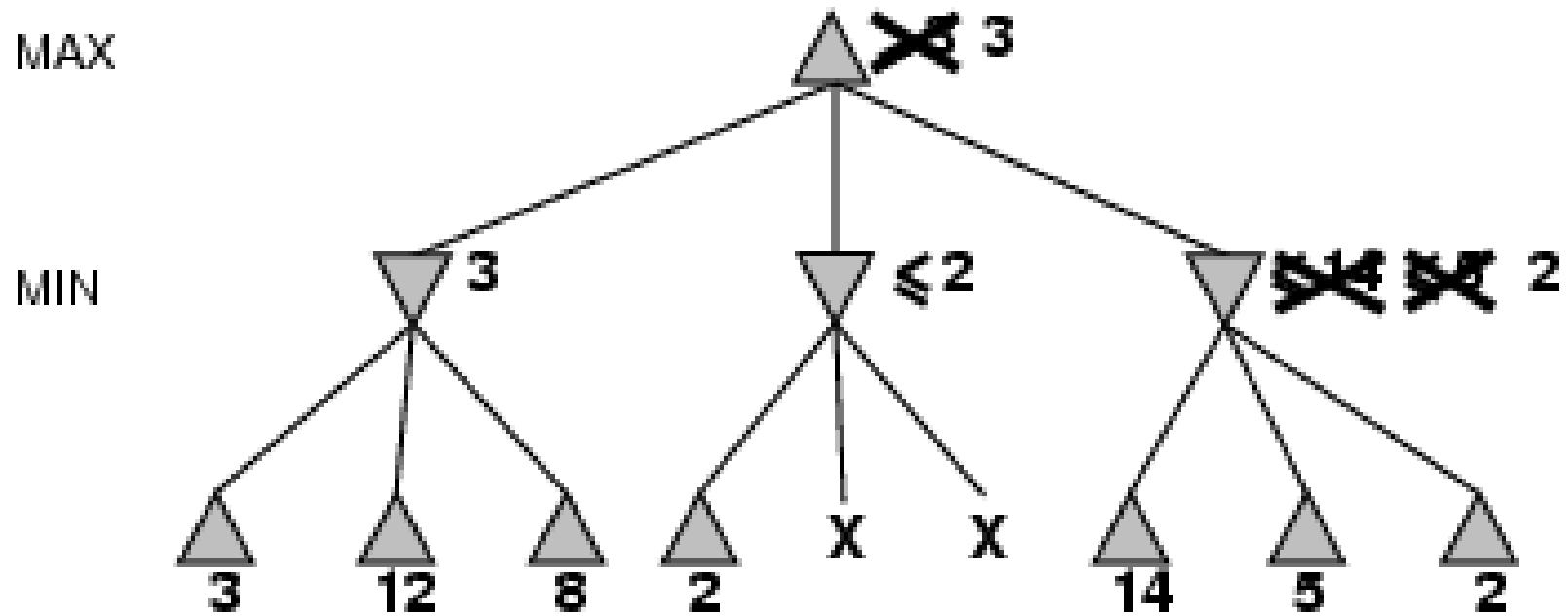
α - β pruning example



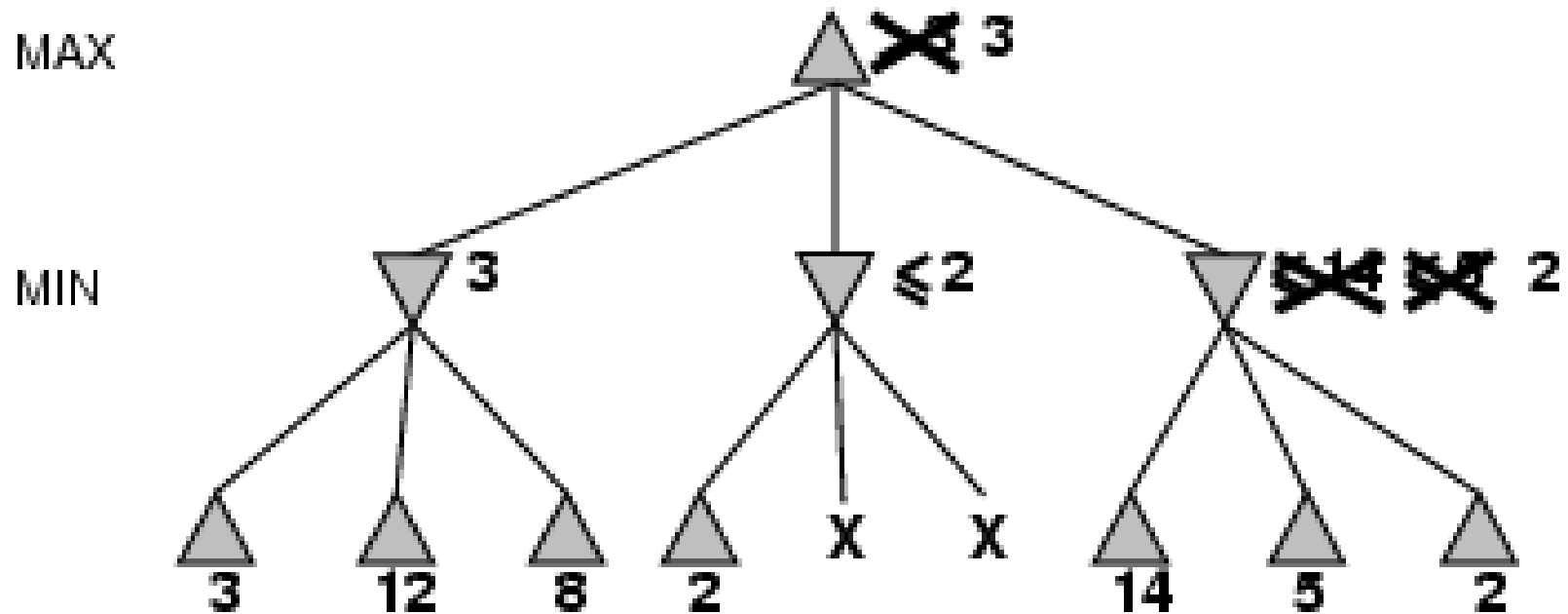
α - β pruning example



α - β pruning example



α - β pruning example



MINIMAX-VALUE(root)
 = $\max(\min(3, 12, 8), \min(2, x, y), \min(14, 5, 2))$
 = $\max(3, \min(2, x, y), 2)$
 = $\max(3, z, 2)$ for $z \leq 2$
 = 3

Properties of α - β

- Pruning **does not** affect final result
- Good move ordering improves effectiveness of pruning
- With "perfect ordering," time complexity = $O(b^{m/2})$
- A simple example of the value of reasoning about which computations are relevant (a form of **metareasoning**)

Why is it called α - β ?

- α is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for *max*
- If v is worse than α , *max* will avoid it
→ prune that branch
- Similarly, β is the value of the best (i.e., lowest-value) choice found so far at any choice point along the path for *min*

MAX

MIN

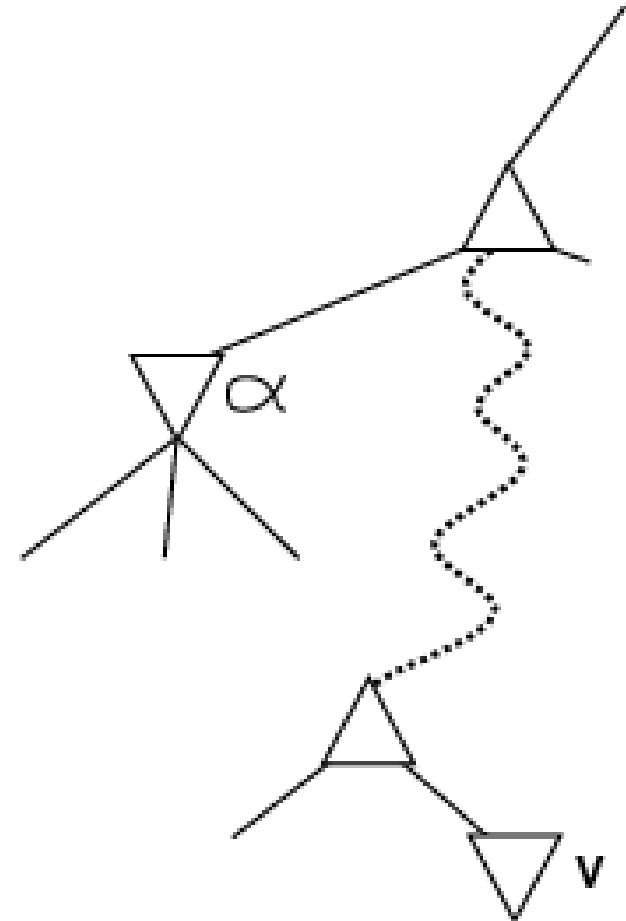
..

..

..

MAX

MIN



Another Example

- In class

The α - β algorithm

function ALPHA-BETA-SEARCH(*state*) *returns an action*

inputs: *state*, current state in game

$v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, +\infty)$

return the *action* in SUCCESSORS(*state*) with value v

function MAX-VALUE(*state*, α , β) *returns a utility value*

inputs: *state*, current state in game

α , the value of the best alternative for MAX along the path to *state*

β , the value of the best alternative for MIN along the path to *state*

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

$v \leftarrow -\infty$

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

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

if $v \geq \beta$ **then return** v

$\alpha \leftarrow \text{MAX}(\alpha, v)$

return v

The α - β algorithm

```
function MIN-VALUE(state,  $\alpha$ ,  $\beta$ ) returns a utility value
  inputs: state, current state in game
            $\alpha$ , the value of the best alternative for MAX along the path to state
            $\beta$ , the value of the best alternative for MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow +\infty$ 
  for  $a, s$  in SUCCESSORS(state) do
     $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s, \alpha, \beta))$ 
    if  $v \leq \alpha$  then return  $v$ 
     $\beta \leftarrow \text{MIN}(\beta, v)$ 
  return  $v$ 
```

Resource limits (Section 6.4)

Recap

- Minimax explores the full search space
- Alpha Beta prunes, but still searches all the way to terminal states for a portion of the search space

Standard approaches to fix resource limits

- **cutoff test:**
e.g., depth limit
- **evaluation function**
= estimated desirability of position

Evaluation functions

- For chess, typically **linear** weighted sum of **features**

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$$

- e.g., $w_1 = 9$ with
 $f_1(s) = (\text{number of white queens}) - (\text{number of black queens})$, etc.

Cutting off search

MinimaxCutoff is identical to *MinimaxValue* except

1. *Terminal?* is replaced by *Cutoff?*
2. *Utility* is replaced by *Eval*

4-ply lookahead is a hopeless chess player!

- 4-ply \approx human novice
- 8-ply \approx typical PC, human master
- 12-ply \approx Deep Blue, Kasparov

Deterministic games in practice

- Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used a precomputed endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 444 billion positions.
- Chess: Deep Blue defeated human world champion Garry Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.
- Othello: human champions refuse to compete against computers, who are too good.
- Go: human champions refuse to compete against computers, who are too bad. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves.
- AAAI conferences now have *general* game-playing competitions, with a \$10K prize!

Summary

- Games are fun to work on!
- They illustrate several important points about AI
- perfection is unattainable → must approximate
- good idea to think about what to think about