

323

ARTIFICIAL INTELLIGENCE & EXPERT SYSTEMS

Adversarial Search

GAMES

Generalizing Search Problems

- ▶ So far: our search problems have assumed agent has complete control of environment.
 - ▶ state does not change unless the agent changes it.
 - ▶ All we need to compute is a single path to a goal state.
- ▶ Assumption not always reasonable
 - ▶ stochastic environment (e.g., the weather, traffic accidents).
 - ▶ other agents whose interests conflict with yours
 - ▶ In both cases you might not traverse the path you are expecting.



Generalizing Search Problems

- ▶ In these cases, we need to generalize our view of search to handle state changes that are not in the control of the agent.
- ▶ One generalization yields game tree search
 - ▶ agent and some other agents.
 - ▶ The other agents are acting to maximize their profits
 - ▶ this might not have a positive effect on your profits.



Generalizing Search Problems

- ▶ **Multiagent environments** – any given agent should consider the actions of other agents and how they affect its own welfare.
 - ▶ **Cooperative multiagent environments**
 - ▶ e.g. in taxi-driving, to avoid collisions, all agents try to maximize the performance measures.
 - ▶ **Competitive multiagent environments**
 - ▶ e.g. in chess, the agent tries to maximize its performance measure while minimizing the other agent's performance measure.



GAMES

- ▶ Competitive environments, lead to **adversarial search** problems.
 - ▶ also known as **games**.
- ▶ In AI, “games” are called **zero-sum games of perfect information**.
 - ▶ deterministic, fully observable environments.
 - ▶ two agents whose actions must alternate.
 - ▶ utility values at the end of the game are always equal and opposite.
 - ▶ e.g. if one player wins a game of chess (+1), the other player necessarily loses (-1).
 - ▶ This opposition between the agents’ utility functions makes the situation **adversarial**.



Two-person Zero-Sum Games

- ▶ chess, checkers, tic-tac-toe, backgammon, go, Doom, “find the last parking space”
- ▶ Your winning means that your opponent loses, and vice-versa.
- ▶ Zero-sum means the sum of your and your opponent's payoff is zero---any thing you gain come at your opponent's cost (and vice-versa). Key insight:
 - ▶ how you act depends on how the other agent acts (or how you think they will act)
 - ▶ and vice versa (if your opponent is a rational player)



More General Games

- ▶ What makes something a game?
 - ▶ there are two (or more) agents influencing state change
 - ▶ each agent has their own interests
 - ▶ e.g., goal states are different; or we assign different values to different paths/states
- ▶ Each agent tries to alter the state so as to best benefit itself.



More General Games

- ▶ What makes games hard?
 - ▶ how you should play depends on how you think the other person will play; but how they play depends on how they think you will play; so how you should play depends on how you think they think you will play; but how they play should depend on how they think you think they think you will play; ...



More General Games

- ▶ Zero-sum games are “fully competitive”
 - ▶ if one player wins, the other player loses
 - ▶ e.g., the amount of money I win (lose) at poker is the amount of money you lose (win)
- ▶ More general games can be “cooperative”
 - ▶ some outcomes are preferred by both of us, or at least our values aren’t diametrically opposed
- ▶ We’ll look in detail at zero-sum games
 - ▶ but first, some examples of simple zero-sum and cooperative games



Game 1: Rock, Paper, Scissors

- ▶ Scissors cut paper, paper covers rock, rock smashes scissors
- ▶ Represented as a matrix:
Player I chooses a row,
Player II chooses a column
- ▶ Payoff to each player in each cell (Pl.I/ Pl.II)
- ▶ 1: win, 0: tie, -1: loss
 - ▶ so it's zero-sum

		Player II		
		R	P	S
Player I	R	0/0	-1/1	1/-1
	P	1/-1	0/0	-1/1
	S	-1/1	1/-1	0/0

Game 2: Prisoner's Dilemma

- ▶ Two prisoners in separate cells, there is not enough evidence to convict them
- ▶ If one confesses, other doesn't:
 - ▶ confessor goes free
 - ▶ other sentenced to 4 years
- ▶ If both confess (both defect)
 - ▶ both sentenced to 3 years
- ▶ Neither confess (both cooperate)
 - ▶ sentenced to 1 year on minor charge
- ▶ Payoff: 4 minus sentence

	Coop	Def
Coop	3/3	0/4
Def	4/0	1/1



Game 3: Battlebots

Two robots: Blue (Craig's), Red (Fahiem's)

- ▶ one cup of coffee, one tea left
- ▶ both C, F prefer coffee (value 10)
- ▶ tea acceptable (value 8)
- ▶ Both robot's go for Cof
 - ▶ collide and get no payoff
- ▶ Both go for tea: same
- ▶ One goes for coffee, other for tea:
 - ▶ coffee robot gets 10
 - ▶ tea robot gets 8

	C	T
C	0/0	10/8
T	8/10	0/0



OPTIMAL DECISIONS in GAMES

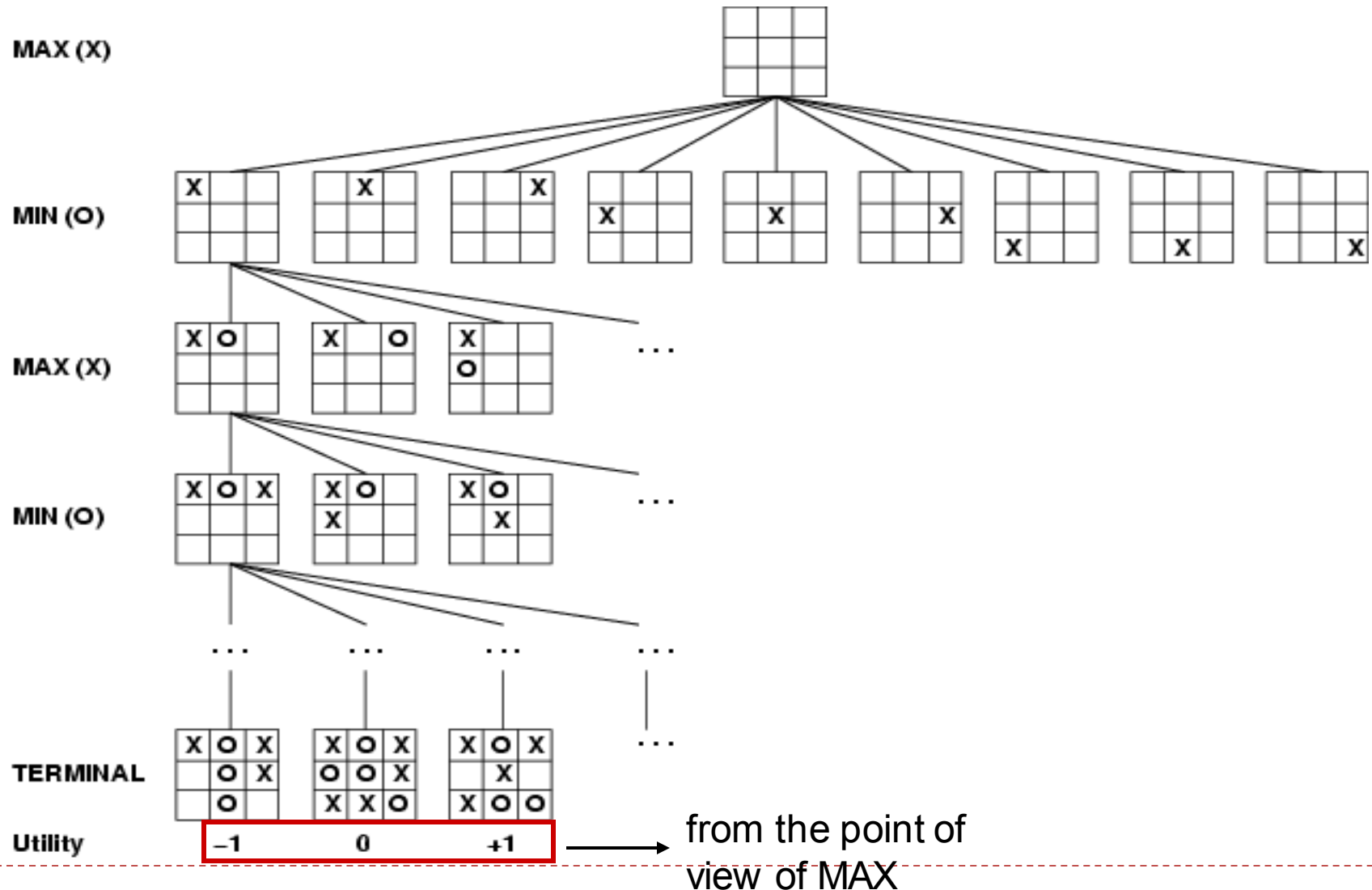
- ▶ Consider games with 2 players: **MAX** and **MIN**
 - ▶ **MAX** moves first and then they take turns moving until the game is over.
 - ▶ At the end, points are given to the winning player as a prize and penalties are given to the loser.
 - ▶ **Game Definition**- as a kind of search problem :
 - ▶ **initial state** – board position and which player to move.
 - ▶ **successor function** – a list of (*move,state*) pairs.
 - ▶ **terminal test** – determines when the game is over.
 - ▶ States where the game has ended are called **terminal states**.
 - ▶ **utility function** – gives a numeric value for terminal states.
 - ▶ e.g. in chess, the outcome is a win, loss or draw with values +1, -1, or 0
-

GAME TREE

- ▶ The initial state and the legal moves for each side define the **game tree**.
 - ▶ e.g. game tree for tic-tac-toe (noughts and crosses) 3x3 board
 - ▶ Initial state → MAX has 9 possible moves
 - ▶ Play alternates between MAX's placing a X and MIN's placing an O until we reach leaf nodes
 - ▶ Terminal states → one player has 3 in a row, column or diagonal OR all the squares are filled.



Game tree (2-player, deterministic, turns)

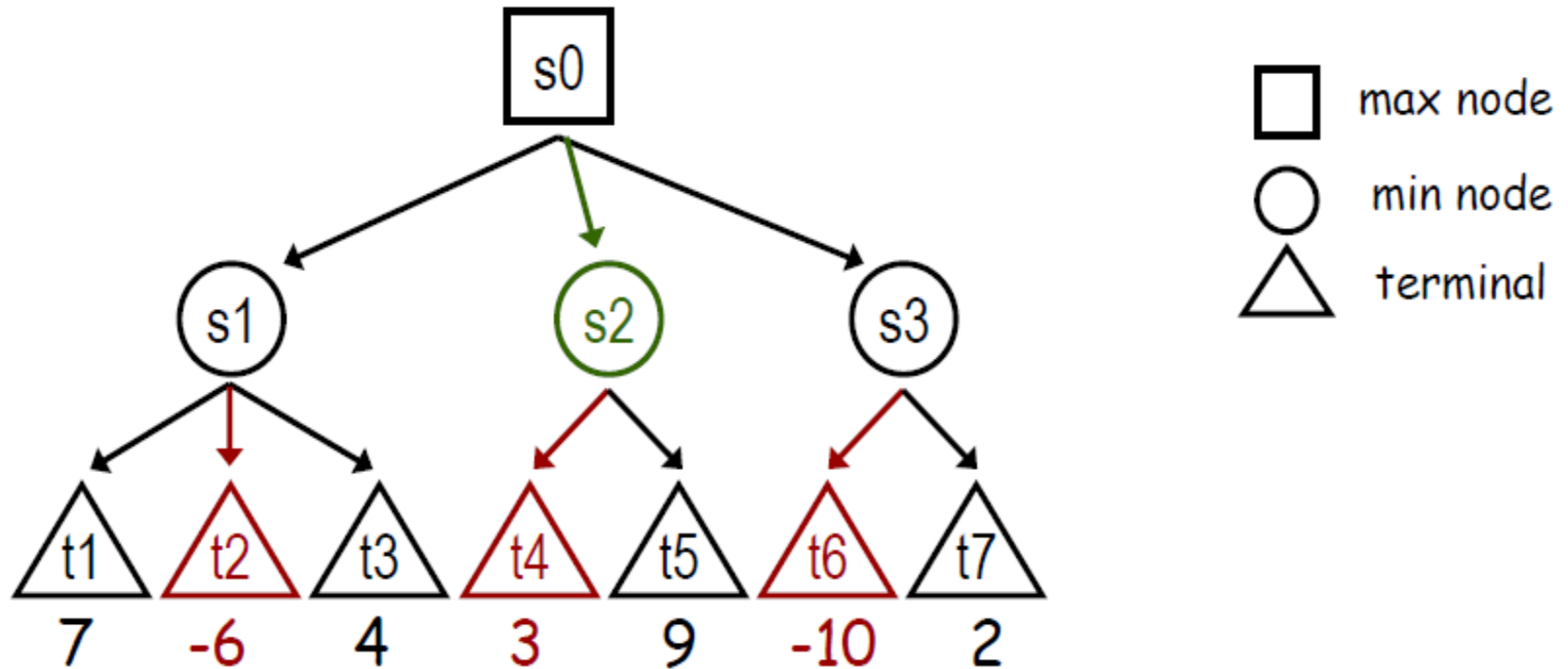


Optimal Strategies

- ▶ In a search problem → optimal solution will be a sequence of moves leading to a goal state
- ▶ In a game → the opponent has something to say about it.
 - ▶ e.g. MAX must find a contingent strategy – specifies MAX's move in the **initial state**
 - ▶ then MAX's moves in the states resulting from every possible response by MIN, and so on..
- ▶ What is a reasonable strategy?

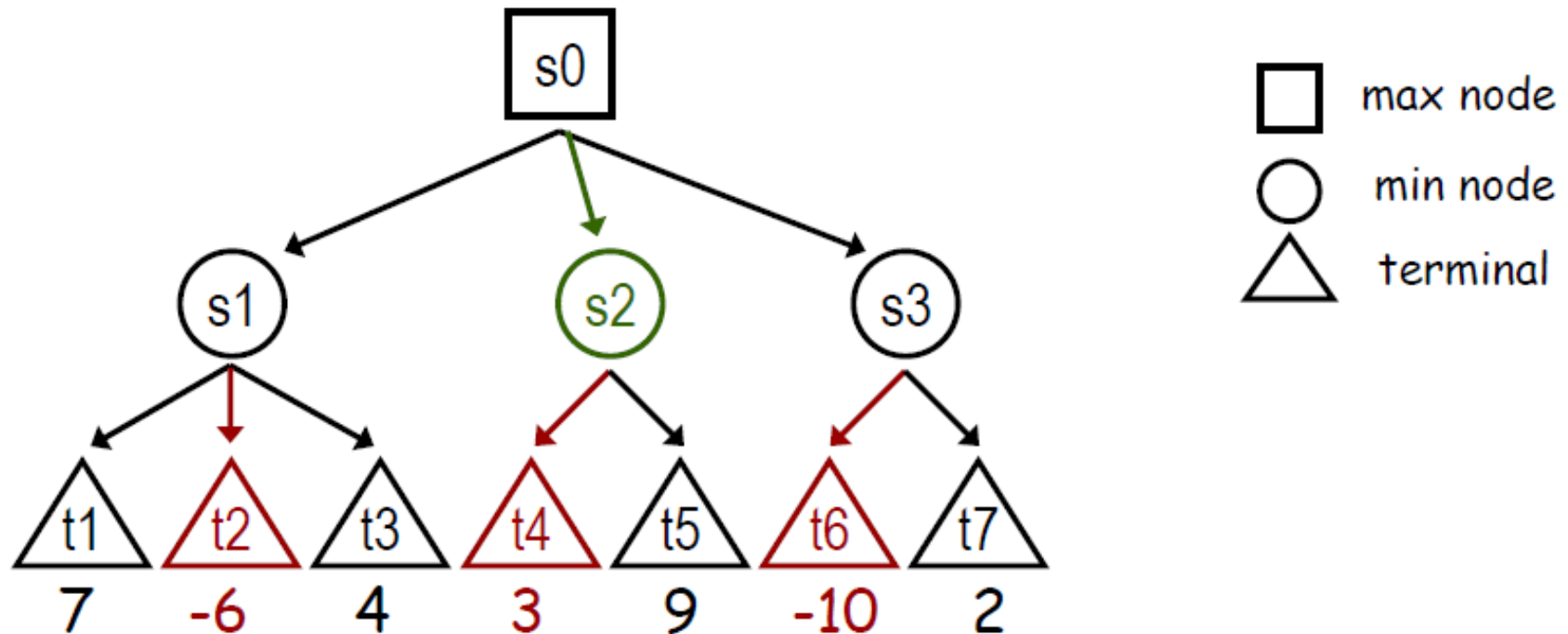


Minimax Strategy



The terminal nodes have utilities.
But we can compute a “utility” for the non-terminal states, by assuming both players always play their best move.

Minimax Strategy

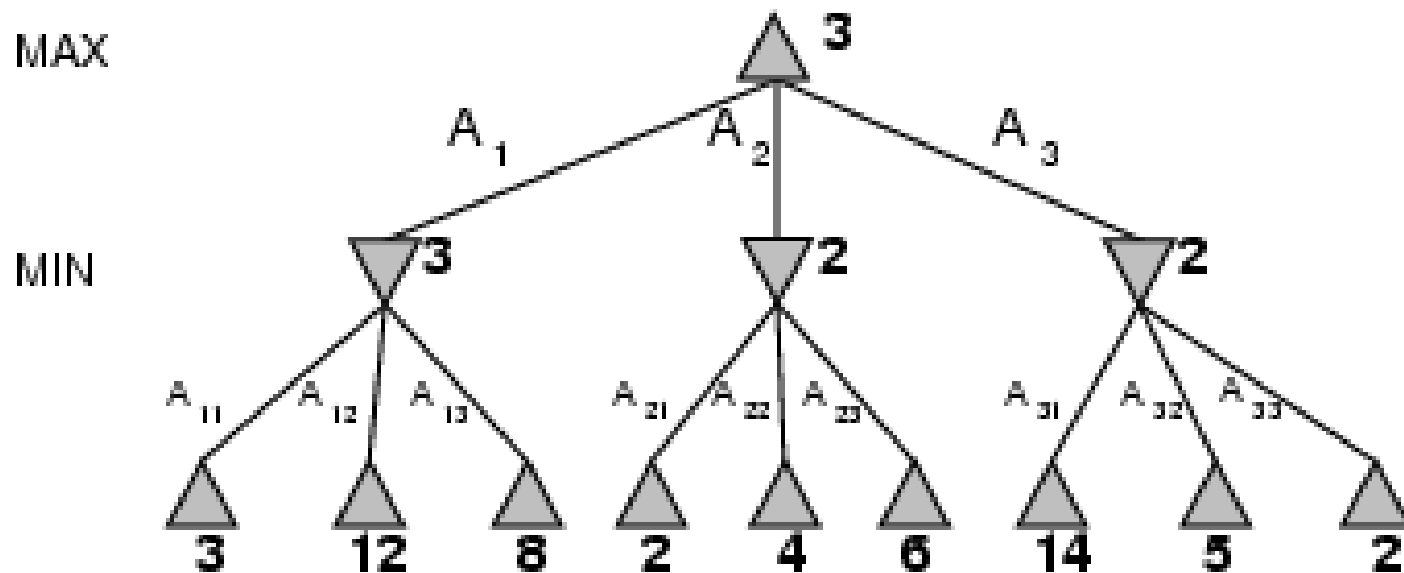


If Max goes to s1, Min goes to t2
* $U(s1) = \min\{U(t1), U(t2), U(t3)\} = -6$
If Max goes to s2, Min goes to t4
* $U(s2) = \min\{U(t4), U(t5)\} = 3$
If Max goes to s3, Min goes to t6
* $U(s3) = \min\{U(t6), U(t7)\} = -10$

So Max goes to s2: so
 $U(s0)$
 $= \max\{U(s1), U(s2), U(s3)\}$
 $= 3$

Minimax

- ▶ Perfect play for **deterministic** games
- ▶ Idea: choose move to position with highest **minimax value**
= best achievable payoff against best play
- ▶ E.g., 2-ply game:



Minimax

- ▶ Given a choice, MAX will prefer to move to a state of maximum value, whereas MIN prefers a state of minimum value:

MINIMAX – VALUE(n) =

$$\begin{cases} \text{UTILITY}(n) & \text{if } n \text{ is a terminal state} \\ \max_{s \in \text{Successors}(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is a MAX node} \\ \min_{s \in \text{Successors}(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is a MIN node} \end{cases}$$



Minimax Strategy

- ▶ Build full game tree (all leaves are terminals)
 - ▶ root is start state, edges are possible moves, etc.
 - ▶ label terminal nodes with utilities
- ▶ Back values *up* the tree
 - ▶ $U(t)$ is defined for all terminals (part of input)
 - ▶ $U(n) = \min \{U(c) : c \text{ a child of } n\}$ if n is a min node
 - ▶ $U(n) = \max \{U(c) : c \text{ a child of } n\}$ if n is a max node



Properties of minimax

- ▶ Minimax algorithm performs a complete **depth-first** exploration of the game tree
 - ▶ $m \rightarrow$ the max depth of the tree
 - ▶ $b \rightarrow$ legal moves of each point
- ▶ Complete? Yes (if tree is finite)
- ▶ Optimal? Yes (against an optimal opponent)
- ▶ Time complexity? $O(b^m)$
- ▶ Space complexity? $O(bm)$

- ▶ For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
 \rightarrow exact solution completely infeasible



Pruning

- ▶ It is not necessary to examine entire tree to make correct minimax decision
- ▶ Assume depth-first generation of tree
 - ▶ After generating value for only *some* of n 's children we can prove that we'll never reach n in a MinMax strategy.
 - ▶ So we needn't generate or evaluate any further children of n !
- ▶ Two types of pruning (cuts):
 - ▶ pruning of max nodes (α -cuts)
 - ▶ pruning of min nodes (β -cuts)



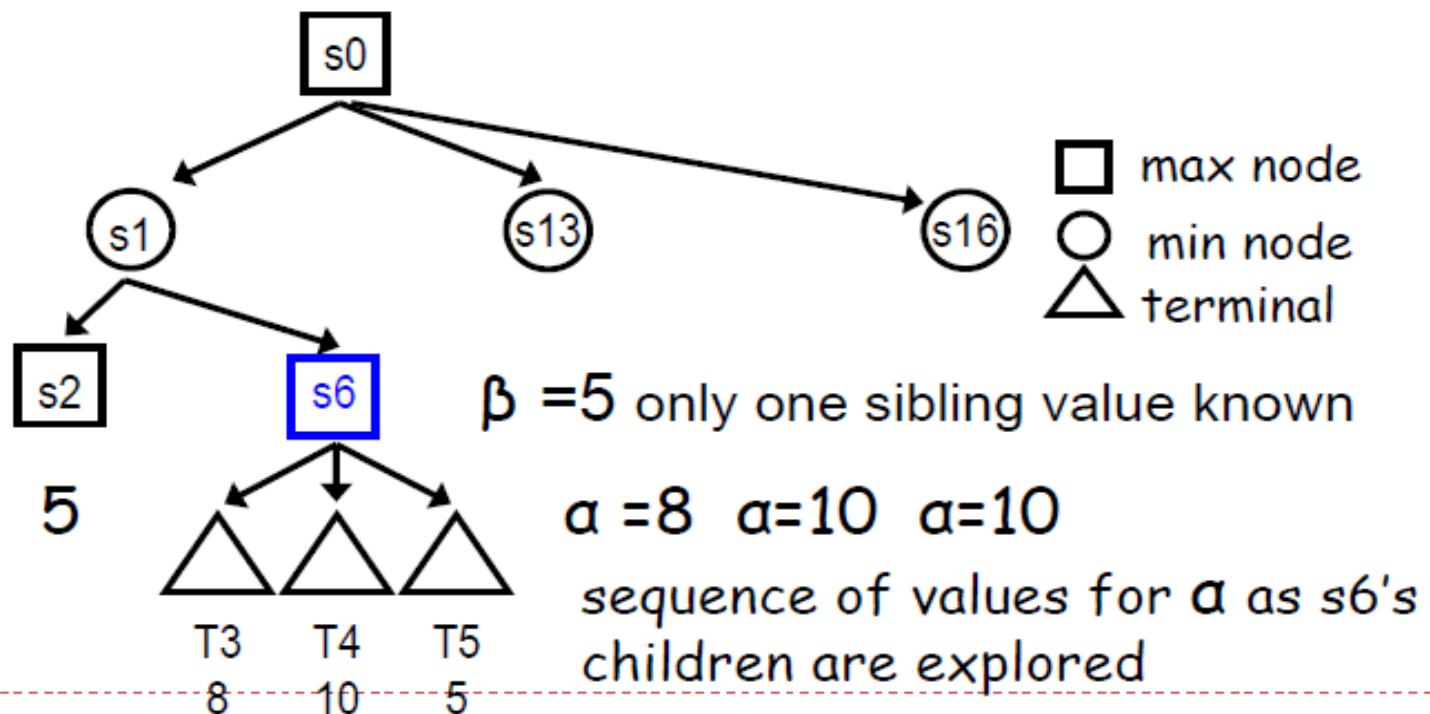
α - β Pruning

- ▶ The problem with minimax search \rightarrow the number of game states is exponential in the number of moves
 - ▶ Solution is to cut it in half \rightarrow compute the correct minimax decision without looking at every node
 - ▶ **Pruning** \rightarrow eliminate large parts of the game tree
 - ▶ This technique is called α - β Pruning
 - ▶ **it returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision.**



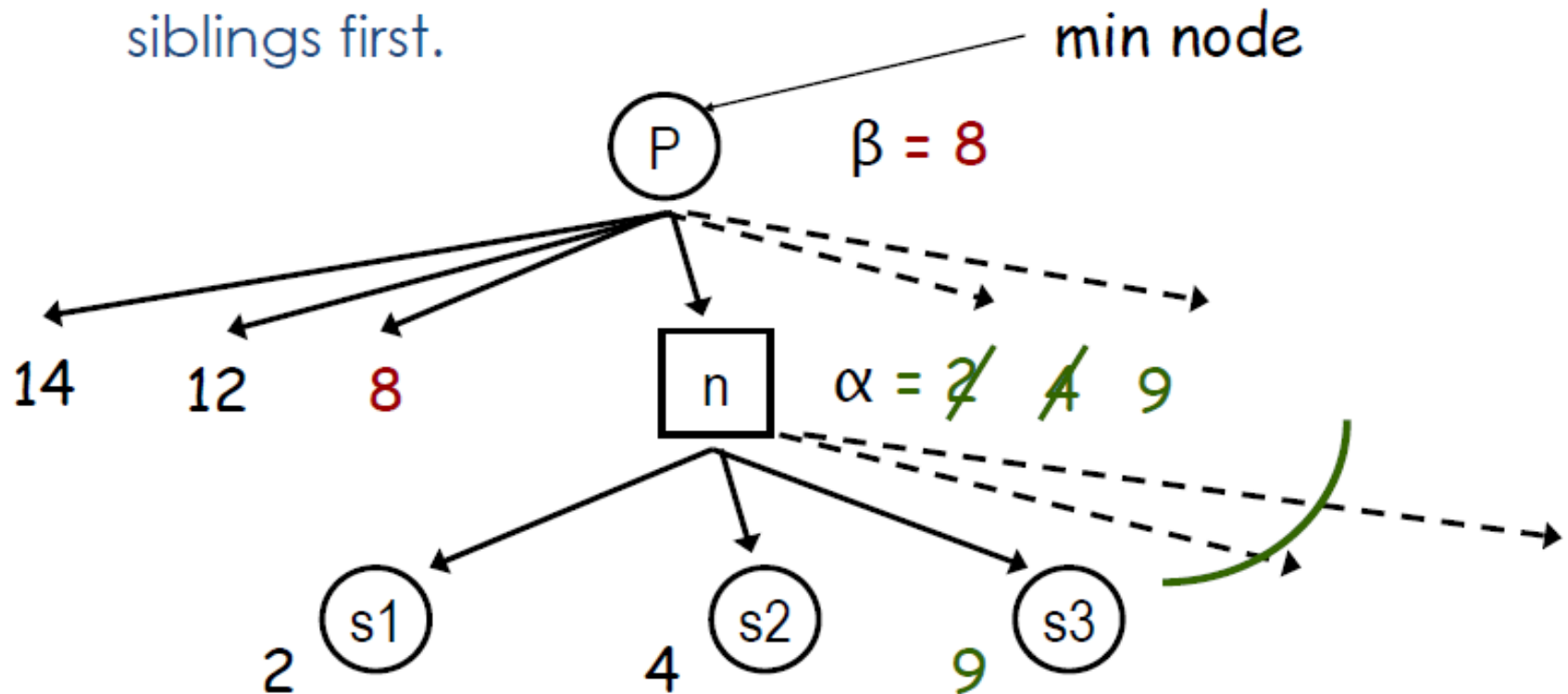
Cutting Max Nodes (Alpha Cuts)

- ▶ At a Max node n :
 - ▶ Let β be the lowest value of n 's siblings examined so far (siblings to the left of n that have already been searched)
 - ▶ Let α be the highest value of n 's children examined so far (changes as children examined)



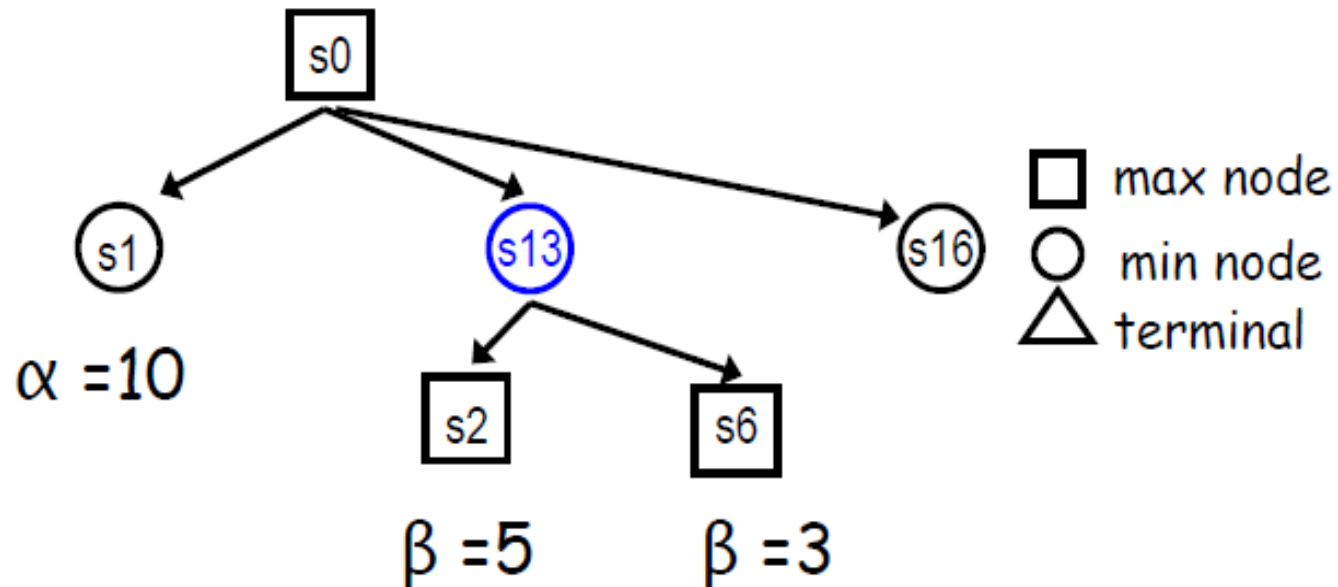
Cutting Max Nodes (Alpha Cuts)

- ▶ If α becomes $\geq \beta$ we can stop expanding the children of n
- ▶ Min will never choose to move from n 's parent to n since it would choose one of n 's lower valued siblings first.



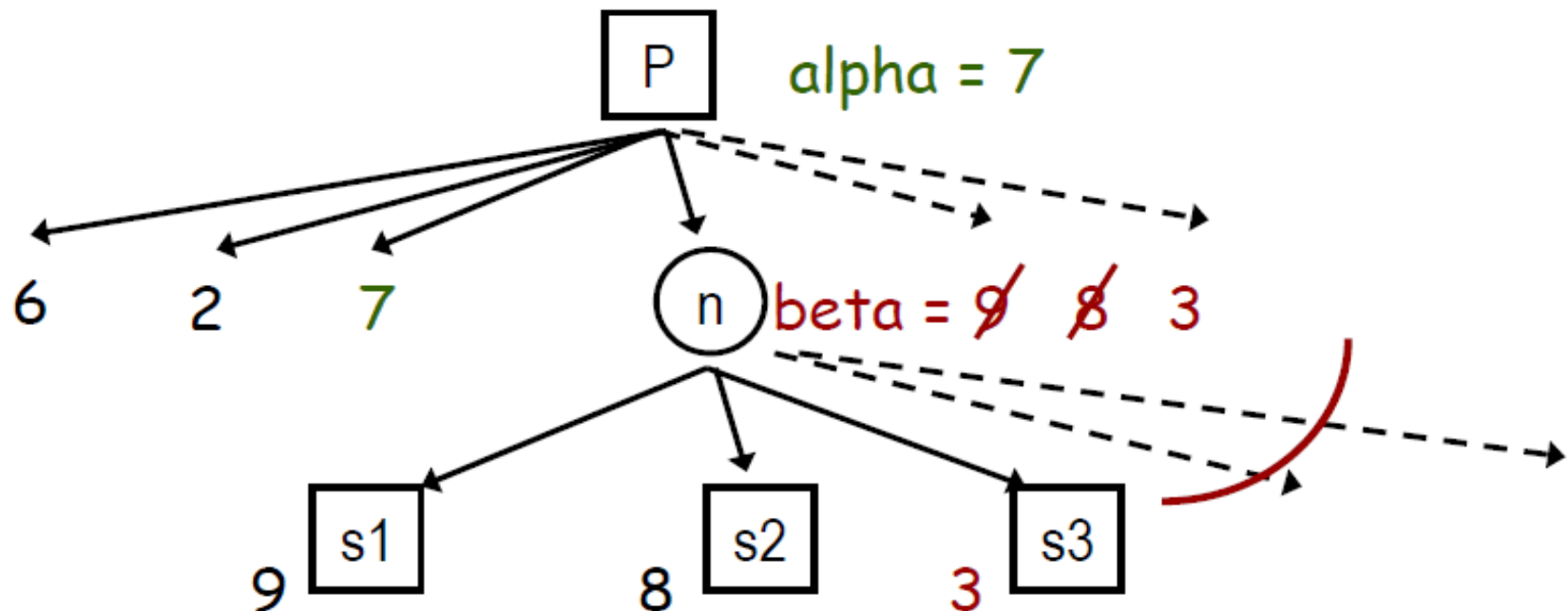
Cutting Min Nodes (Beta Cuts)

- ▶ At a Min node n :
 - ▶ Let β be the lowest value of n 's children examined so far (changes as children examined)
 - ▶ Let α be the highest value of n 's sibling's examined so far (fixed when evaluating n)

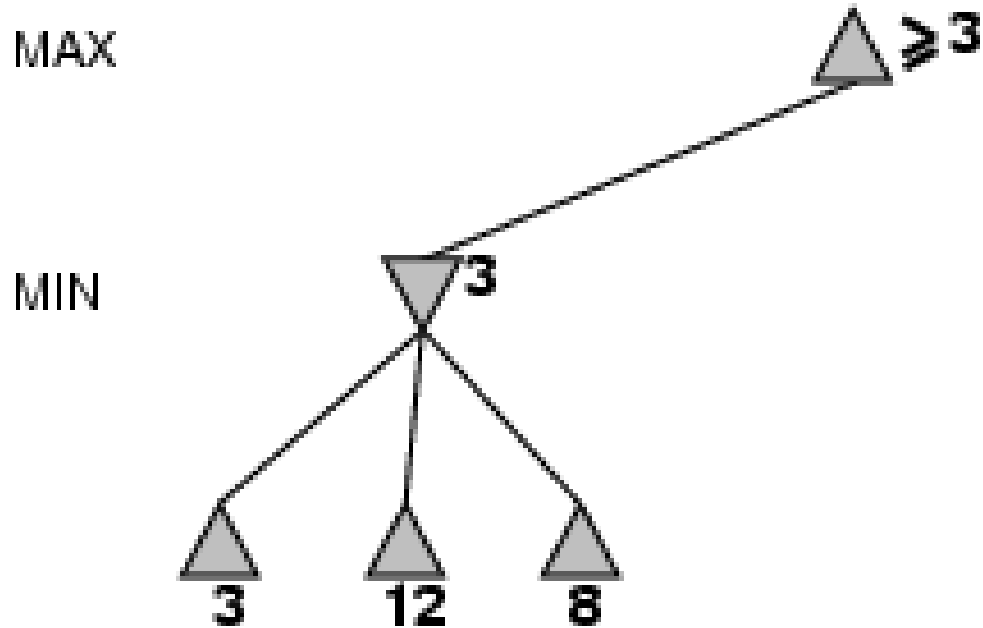


Cutting Min Nodes (Beta Cuts)

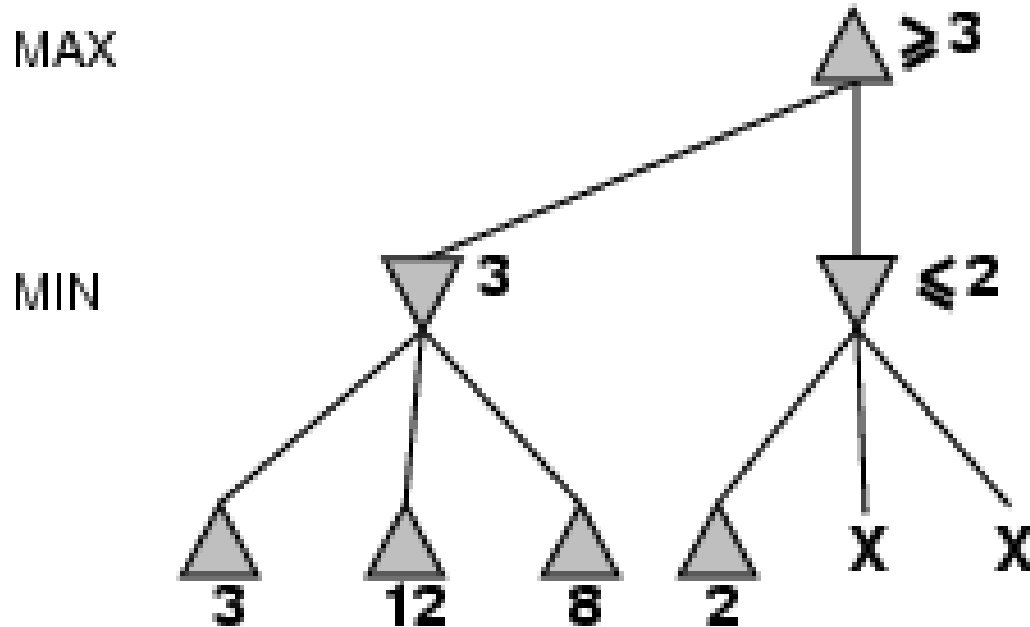
- ▶ If β becomes $\leq \alpha$ we can stop expanding the children of n .
- ▶ Max will never choose to move from n 's parent to n since it would choose one of n 's higher value siblings first.



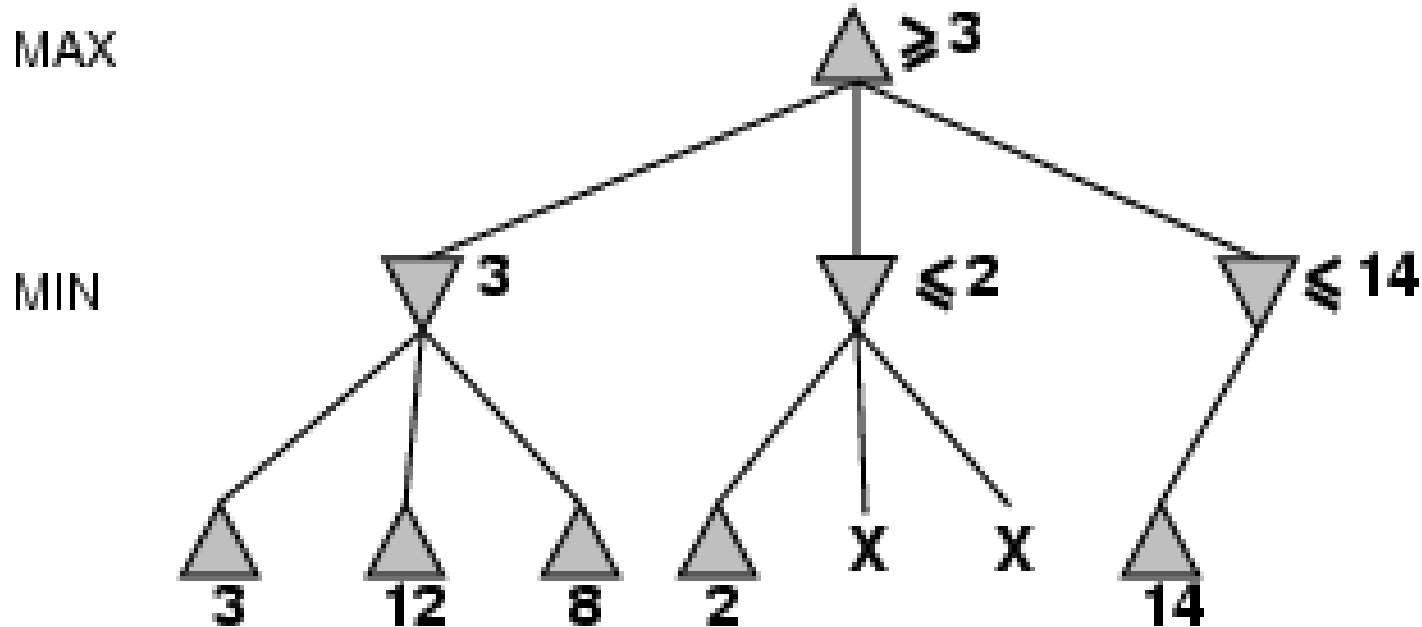
α - β Pruning Example



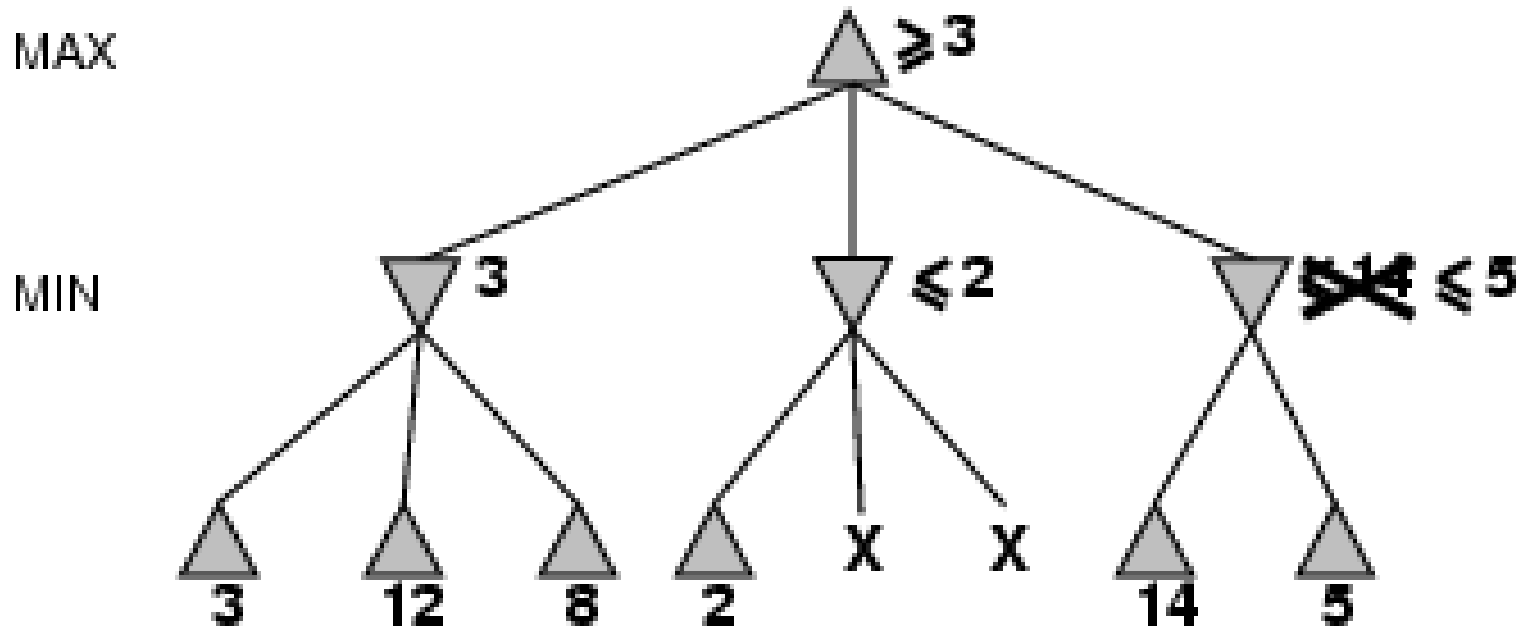
α - β Pruning Example



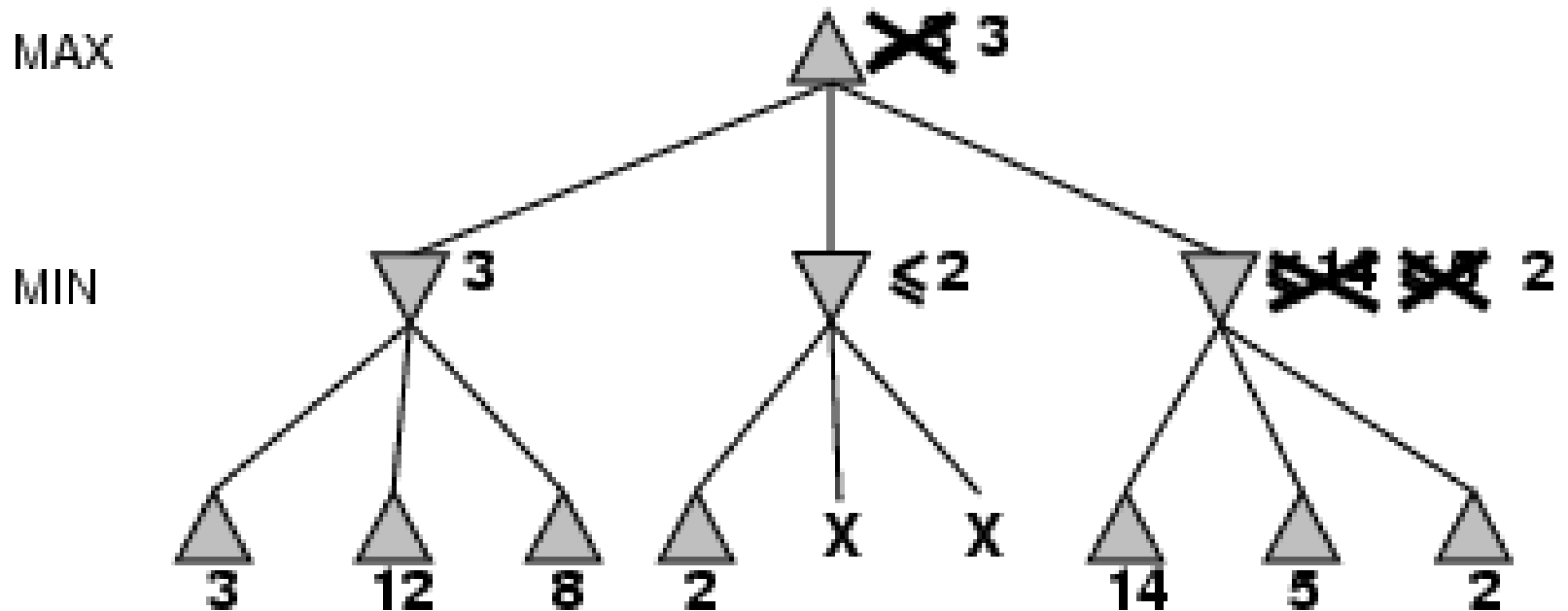
α - β Pruning Example



α - β Pruning Example



α - β Pruning Example



α - β Pruning Example

$$\begin{aligned}\text{MINIMAX-VALUE (root)} &= \max(\min(3, 12, 8), \\ &\quad \min(2, x, y), \min(14, 5, 2)) \\ &= \max(3, \min(2, x, y), 2) \\ &= \max(3, z, 2) \quad z \leq 2 \\ &= 3.\end{aligned}$$



Properties of α - β

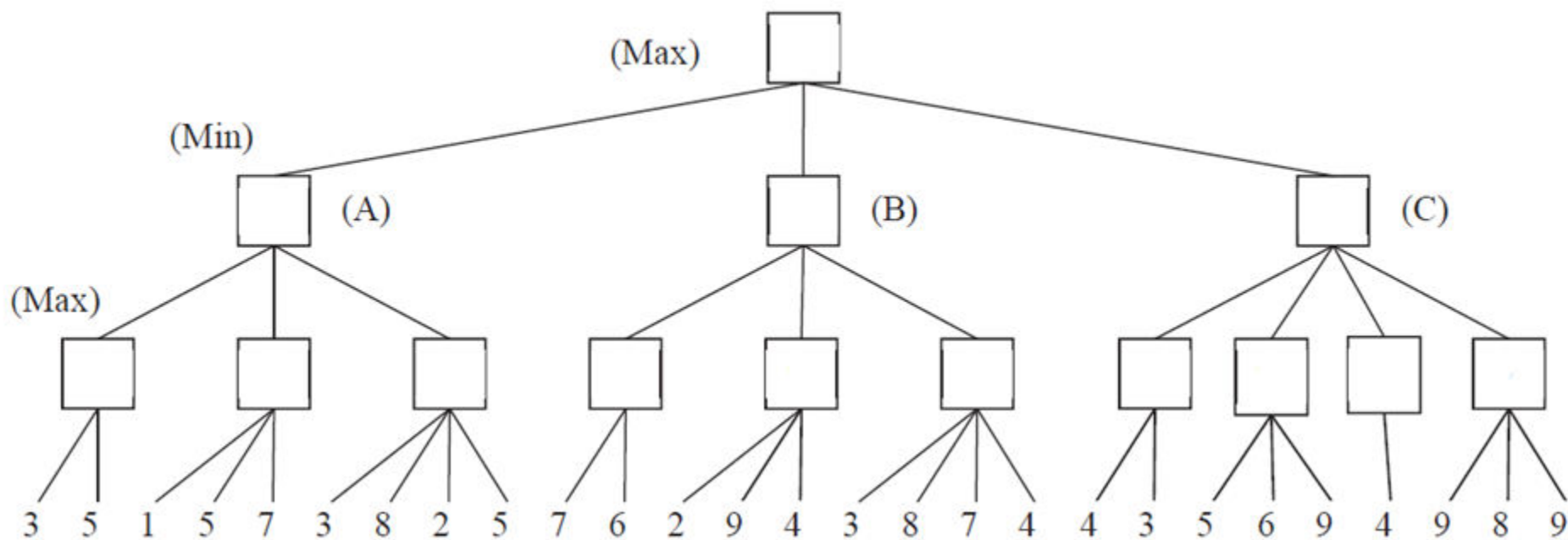
- ▶ Pruning **does not** affect final result
- ▶ Good move ordering improves effectiveness of pruning
- ▶ With "perfect ordering," time complexity = $O(b^{m/2})$
→ **doubles** depth of search
- ▶ What if your opponent doesn't play rationally? Will your stored strategy still work?



Example

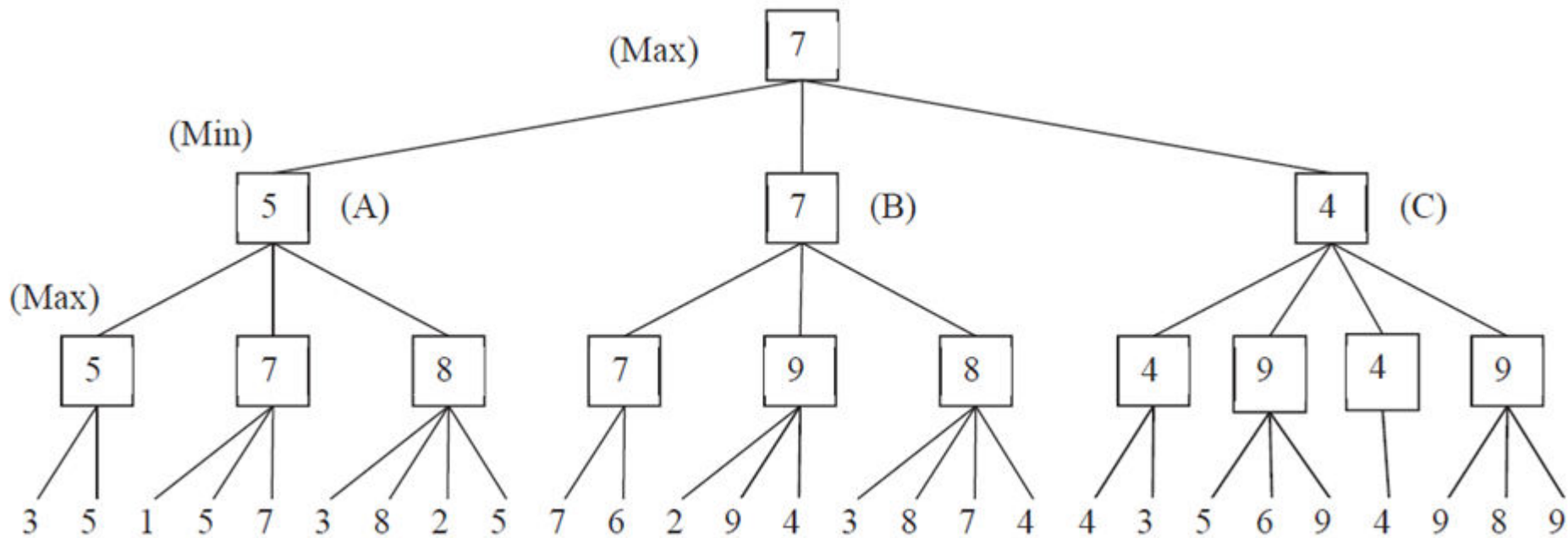
- ▶ The game tree illustrates a position reached in the game.
- ▶ Process the tree left-to-right. It is **Max**'s turn to move.
- ▶ At each leaf node is the estimated score returned by the heuristic static evaluator.





Example cont.

- Fill in each blank square with the proper mini-max search value.



Example cont.

- ▶ **What is the best move for Max?** (write A, B, or C)
- ▶ **What score does Max expect to achieve?**

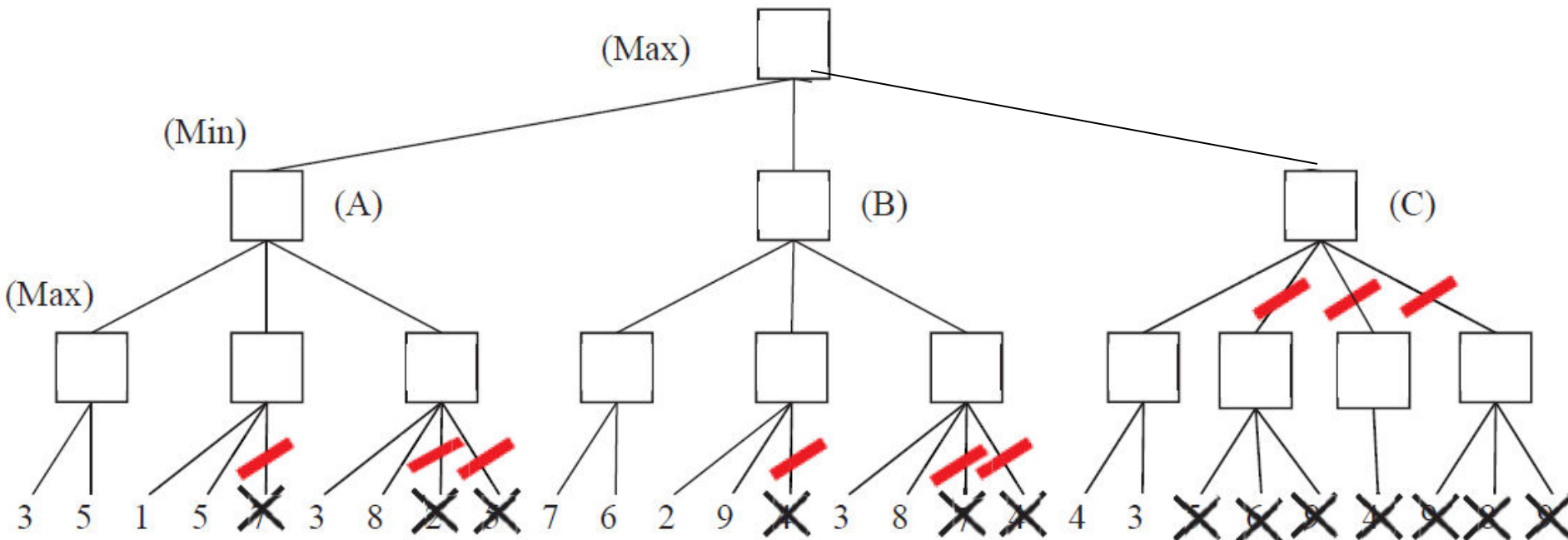


- ▶ **ALPHA-BETA PRUNING.**

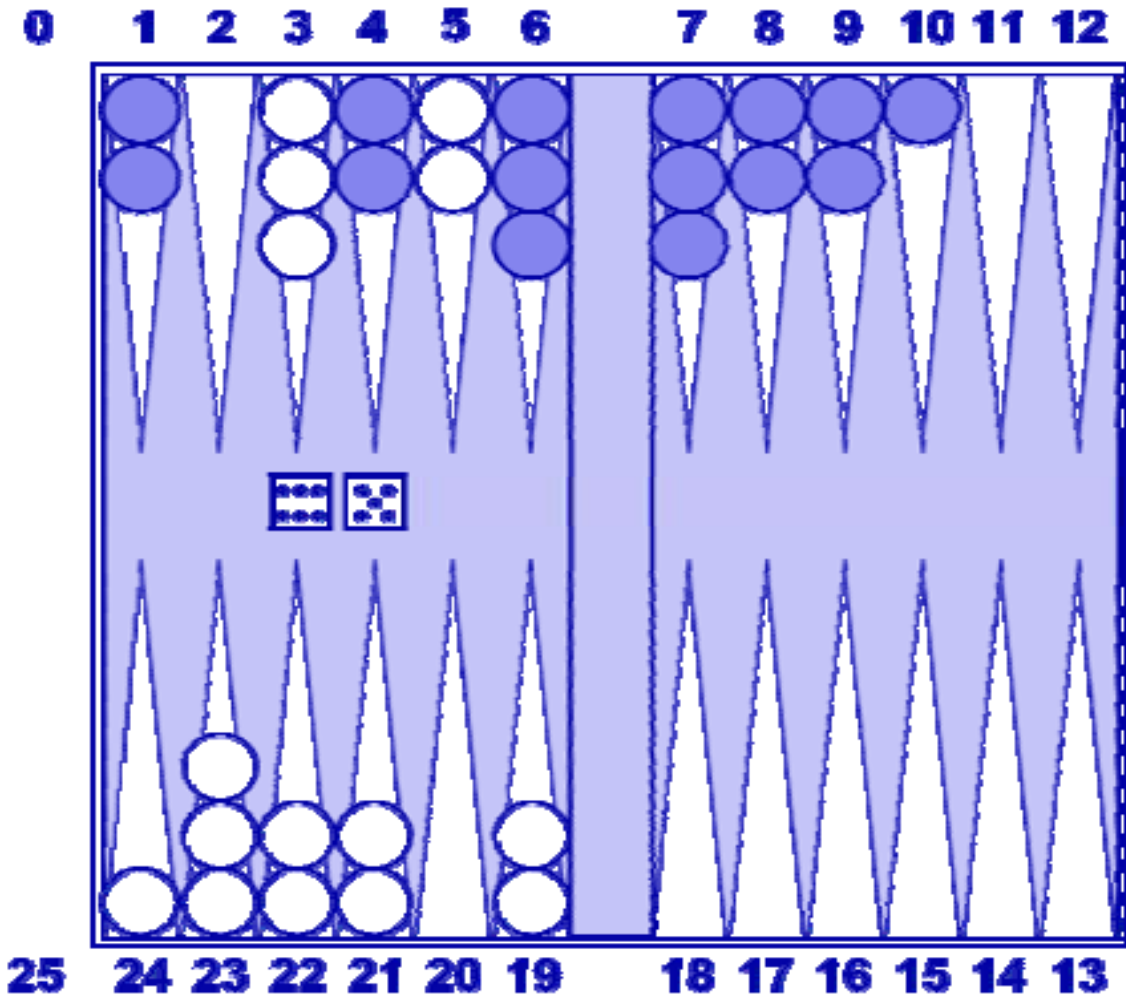
- ▶ Process the tree left-to-right.

- ▶ **Cross out each leaf node that will be pruned by Alpha-Beta Pruning.**





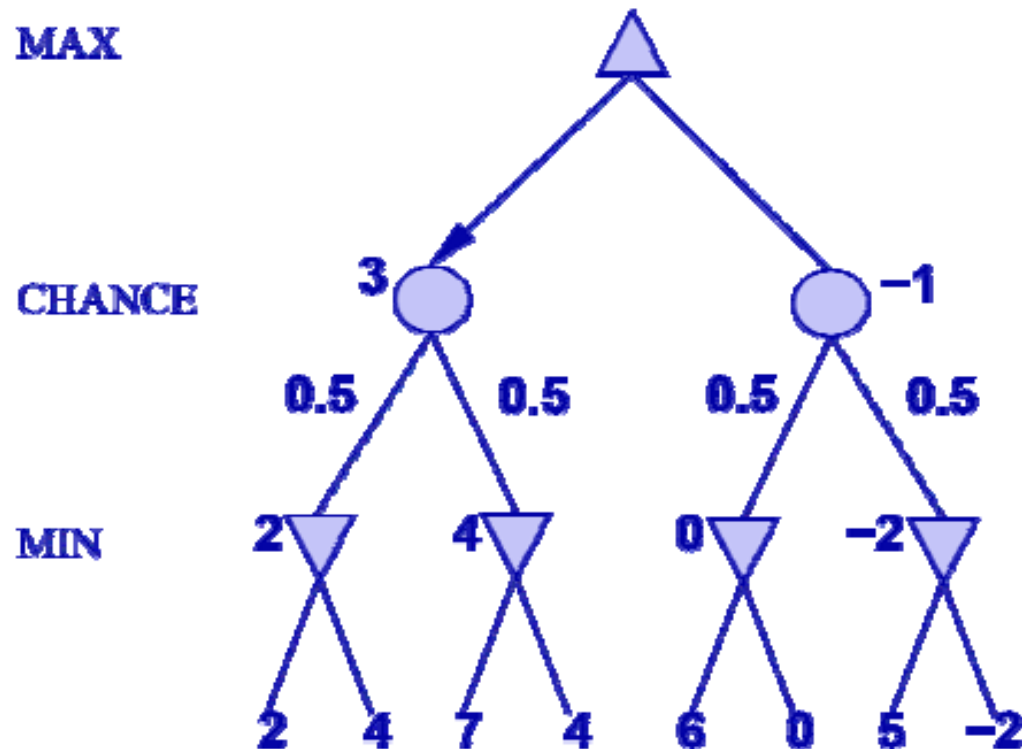
Nondeterministic Games



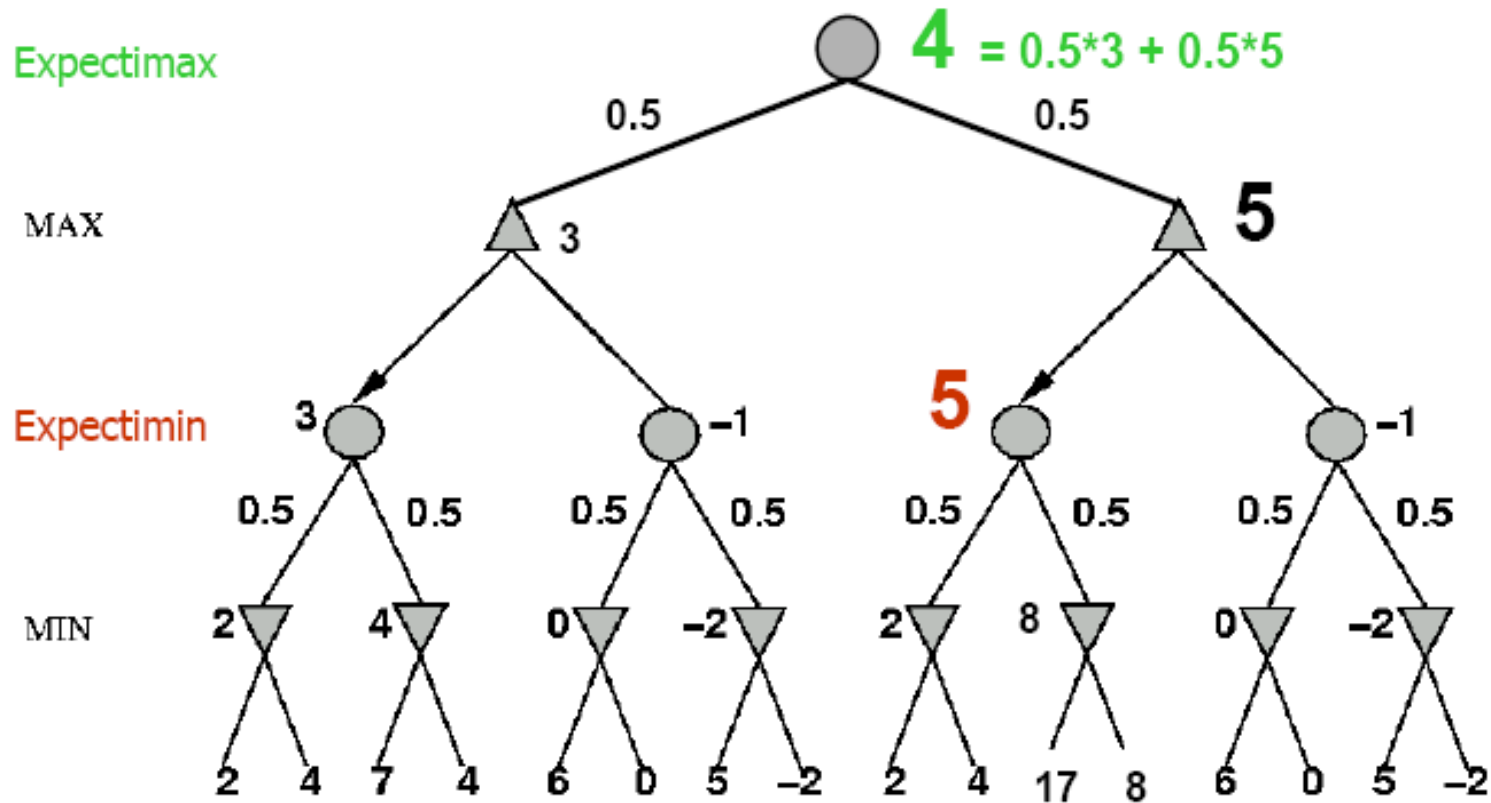
- All “real” games are too large to enumerate tree
- e.g., chess branching factor is roughly 35
 - Depth 10 tree:
2,700,000,000,000,000 nodes
 - Even alpha-beta pruning won't help here!

Nondeterministic games: the element of chance

- ▶ In nondeterministic games, chance introduced by dice, card-shuffling
- ▶ Simplified example with coin-flipping



Nondeterministic games: the element of chance



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

