Adversarial Search

Chapter 5

(acknowledgement goes to Gillian Smith(NEU) and Hwee Tou Ng's for slides used in this lecture)

Game Playing

	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

Game Playing

	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

Game Playing

	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

Game Playing

	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

Game Playing

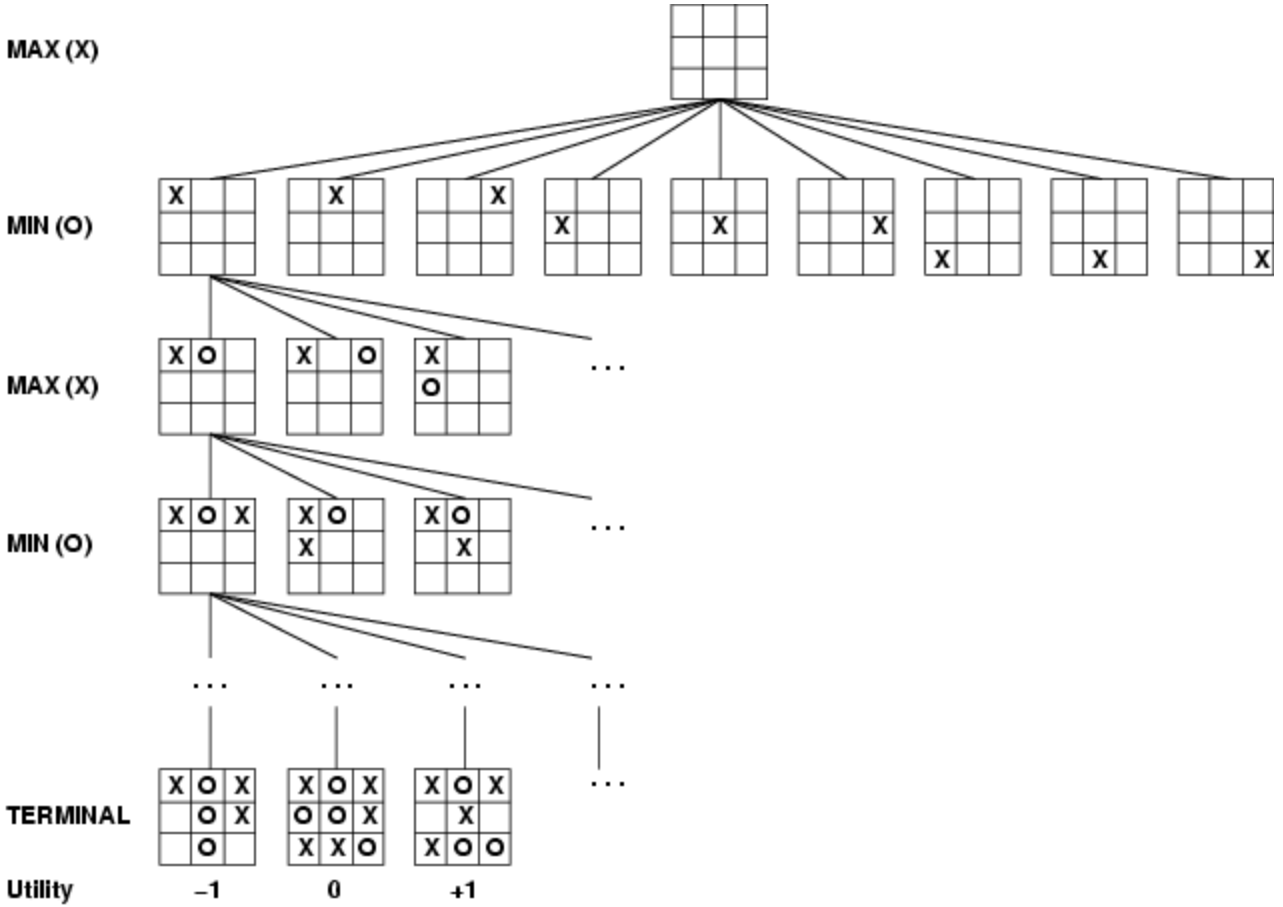
	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

Minimax Algorithm

- Two player, adversarial, perfect information game
  - Tic Tac Toe
  - Nim
  - Chess
  - Go

- Two players: MAX and MIN
  - MAX moves first

Game tree (2-player, deterministic, turns)

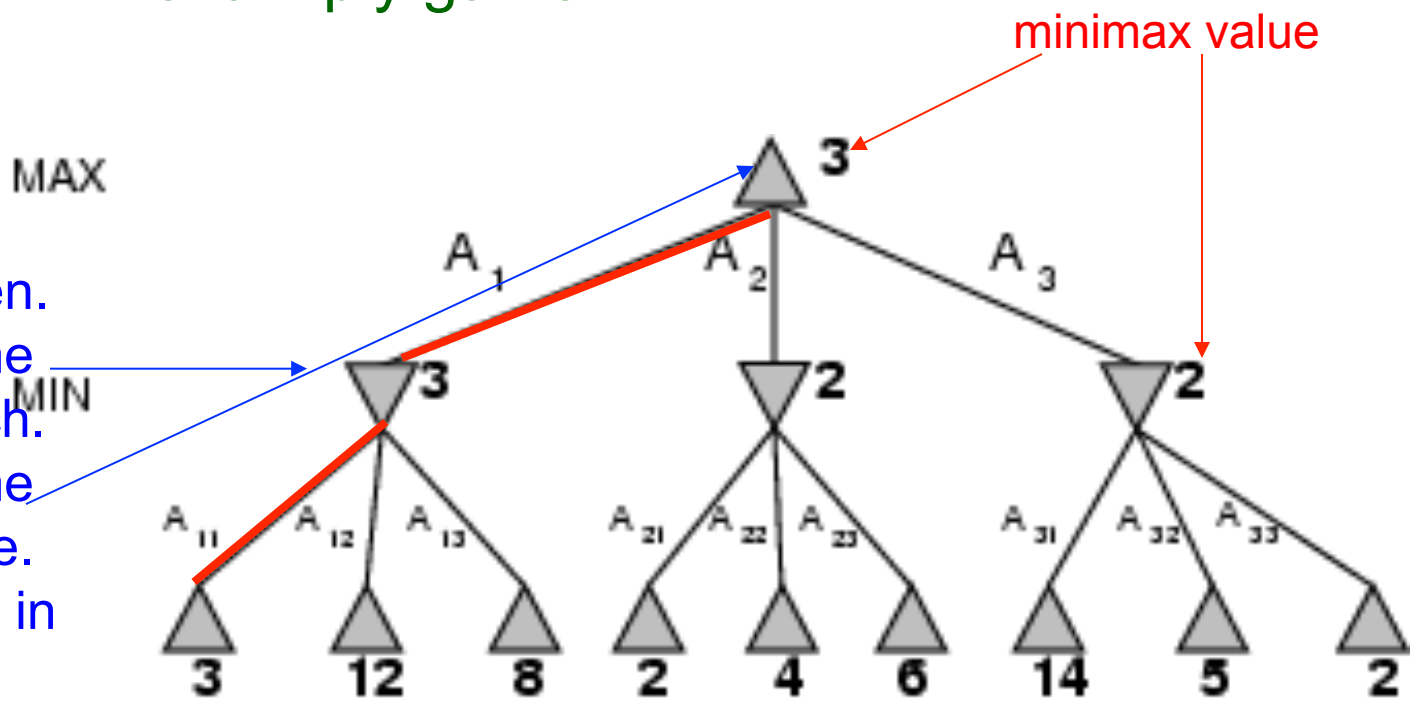


How do we search this tree to find the optimal move?

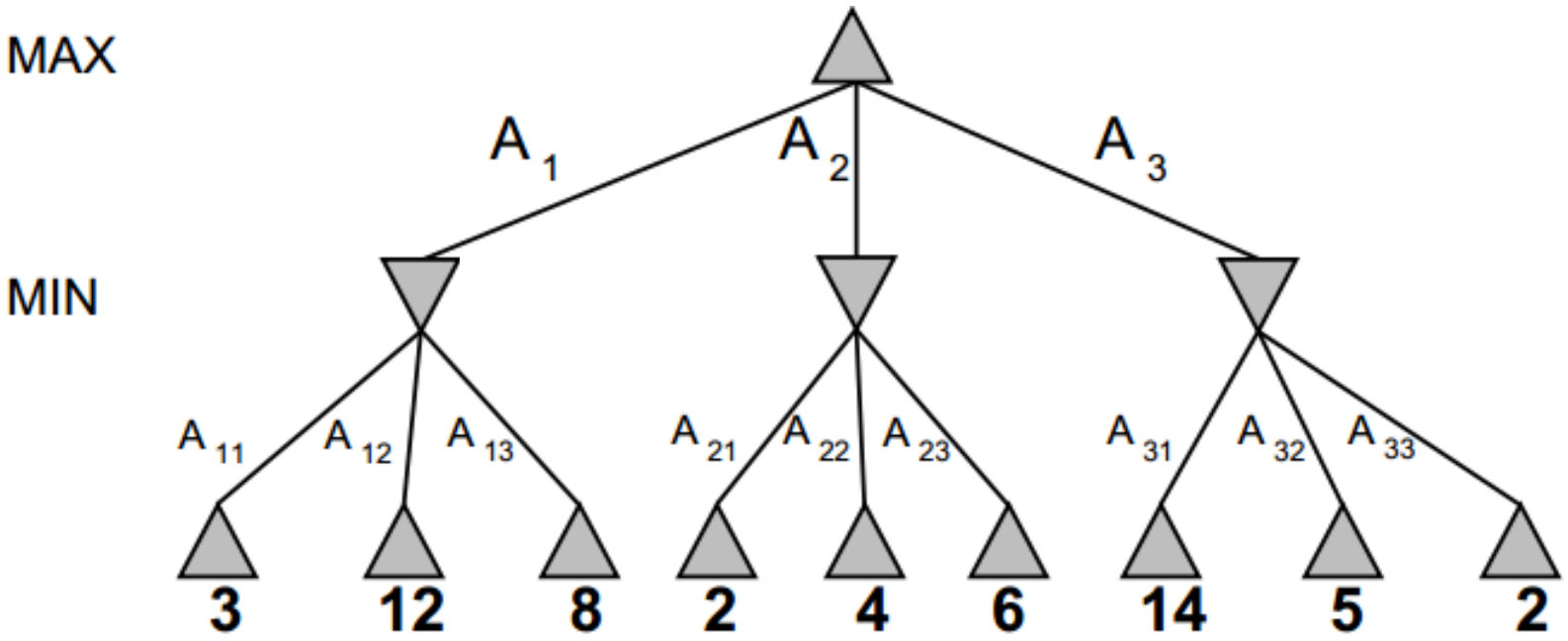
Minimax

- Idea: choose a move to a position with the highest minimax value = best achievable payoff against a rational opponent.
- Example: deterministic 2-ply game:

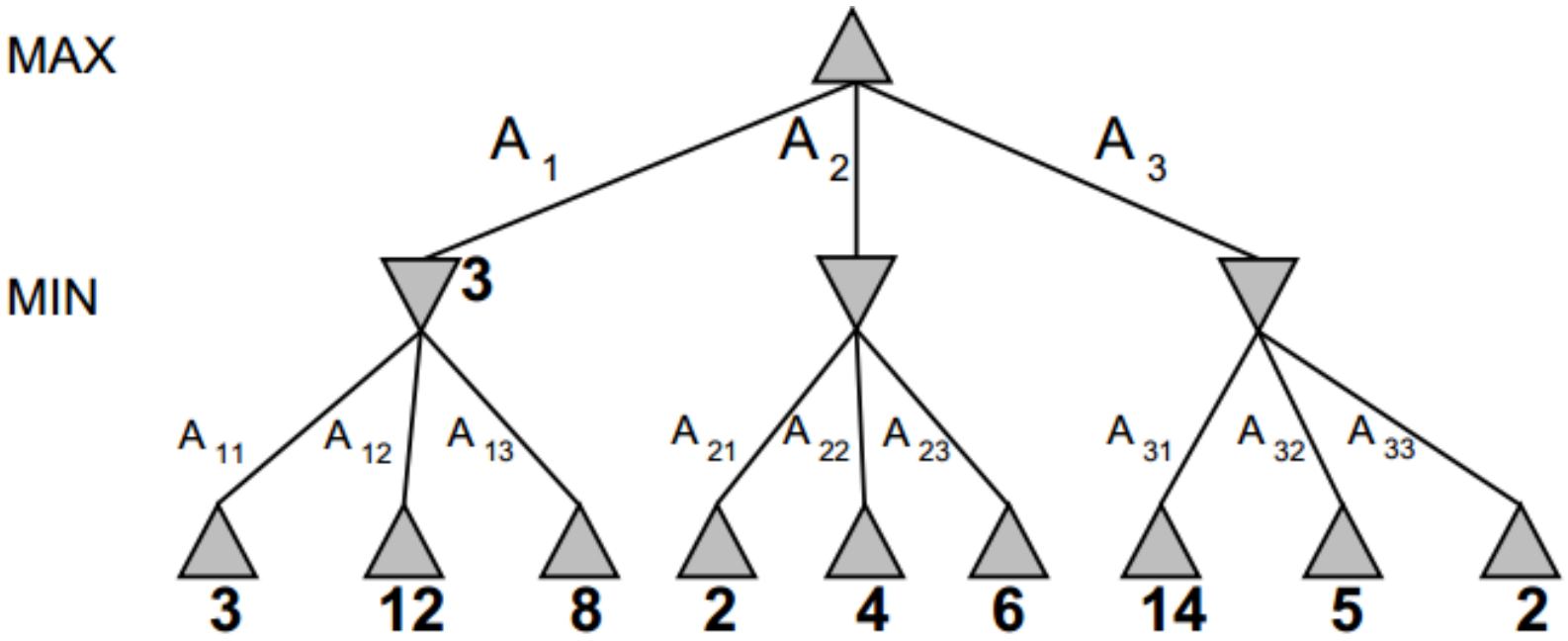
Minimax value is computed bottom up:
-Leaf values are given.
-3 is the best outcome for MIN in this branch.
-3 is the best outcome for MAX in this game.
-We explore this tree in depth-first manner.



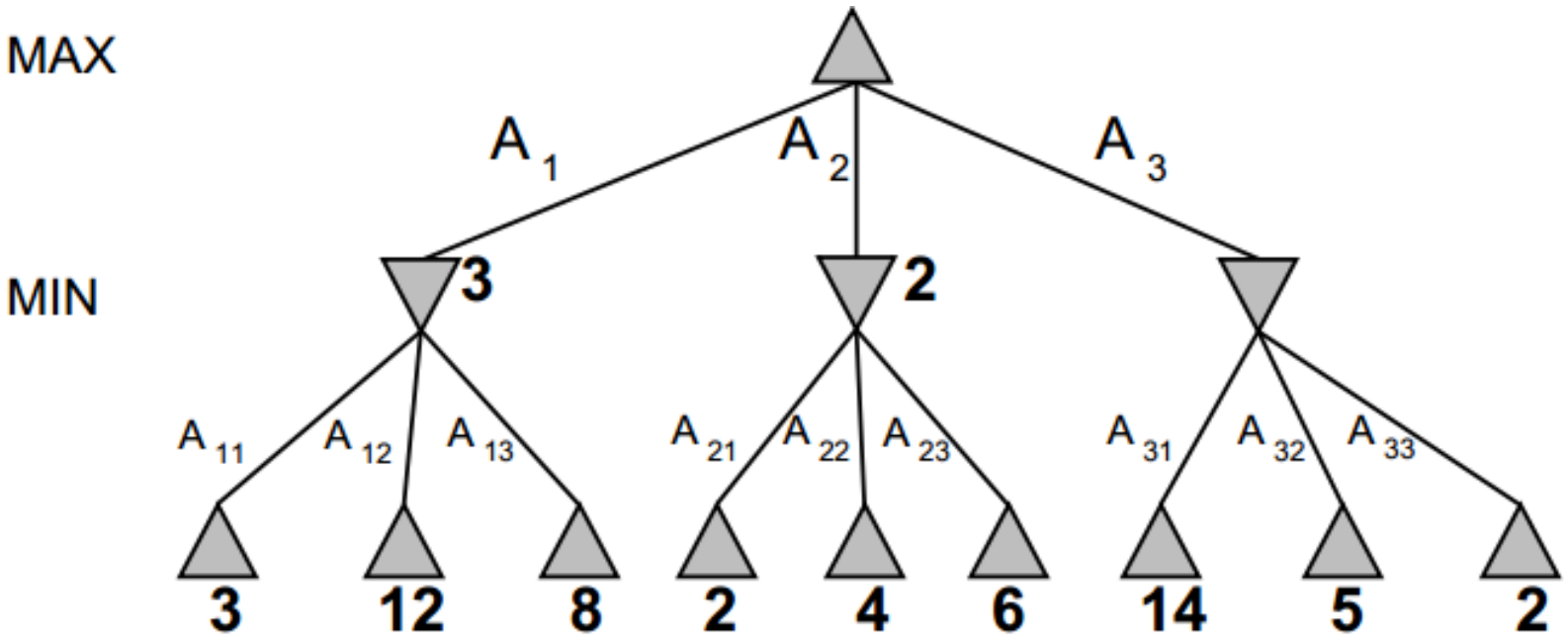
Minimax: Small Example



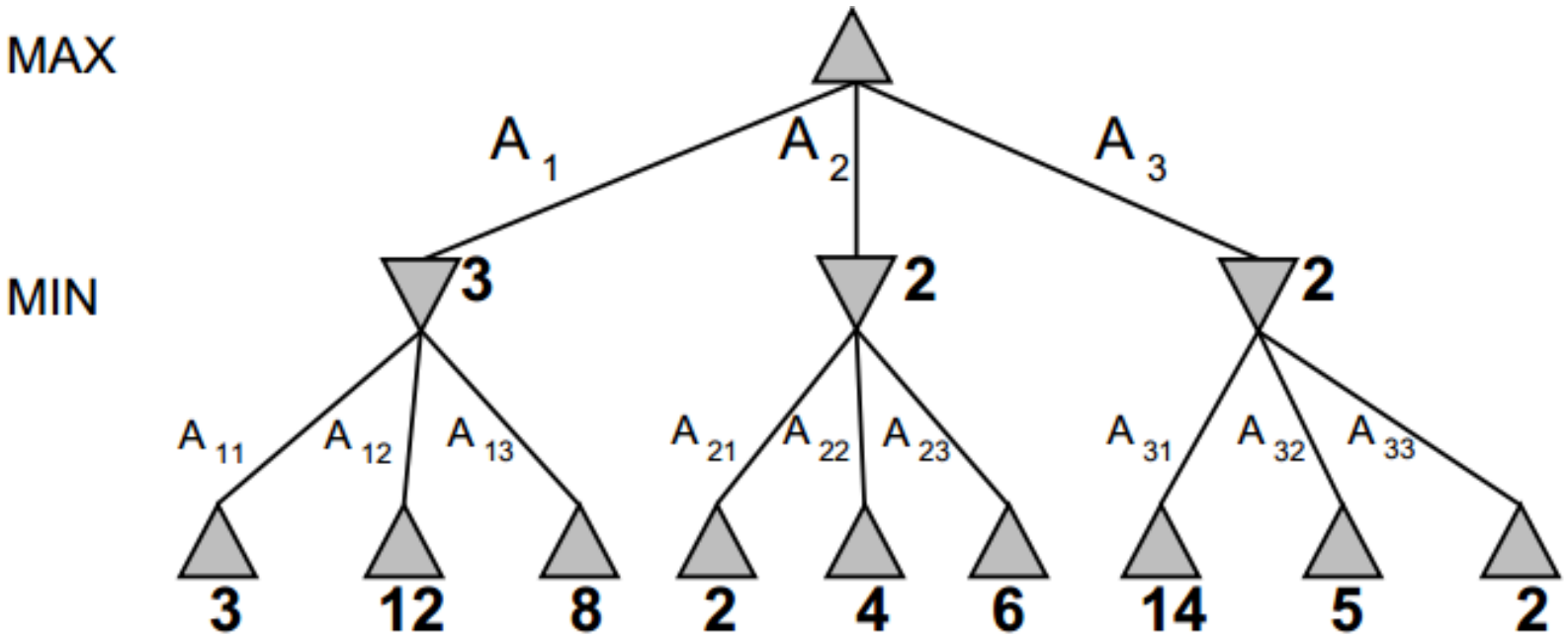
Minimax: Small Example



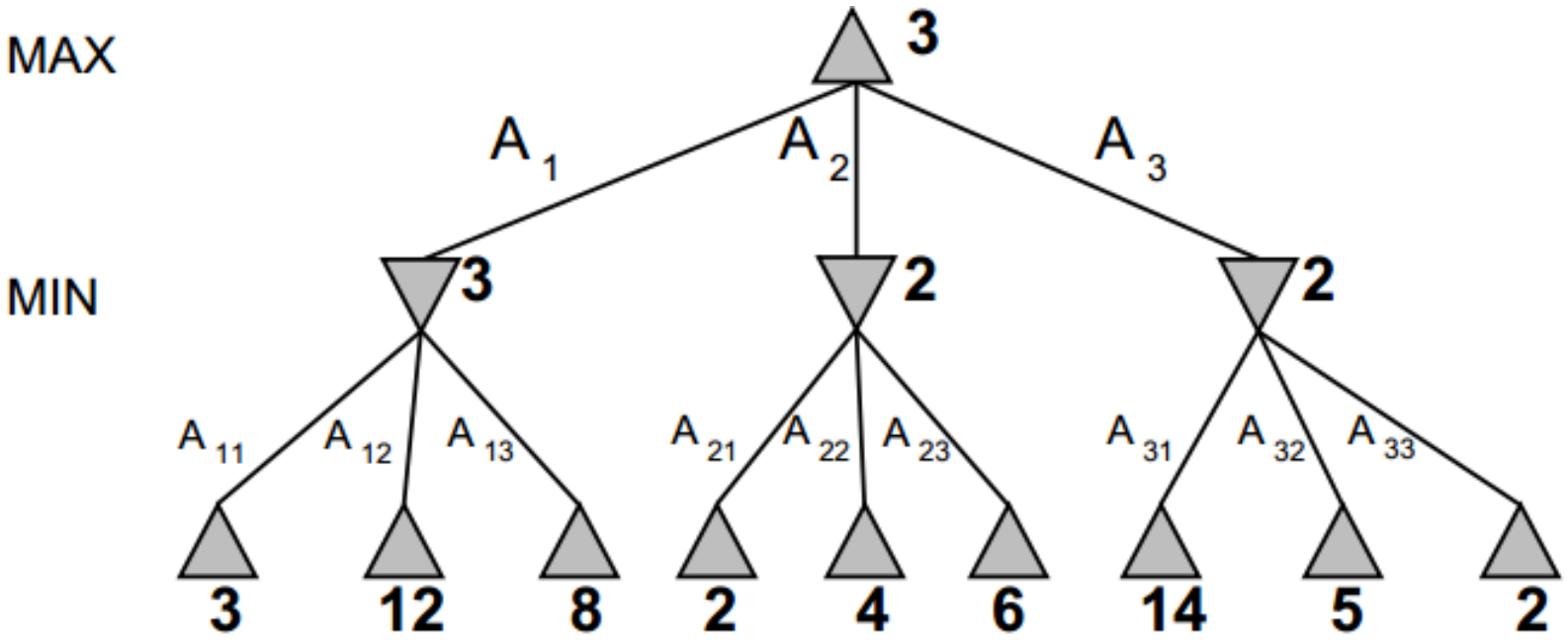
Minimax: Small Example



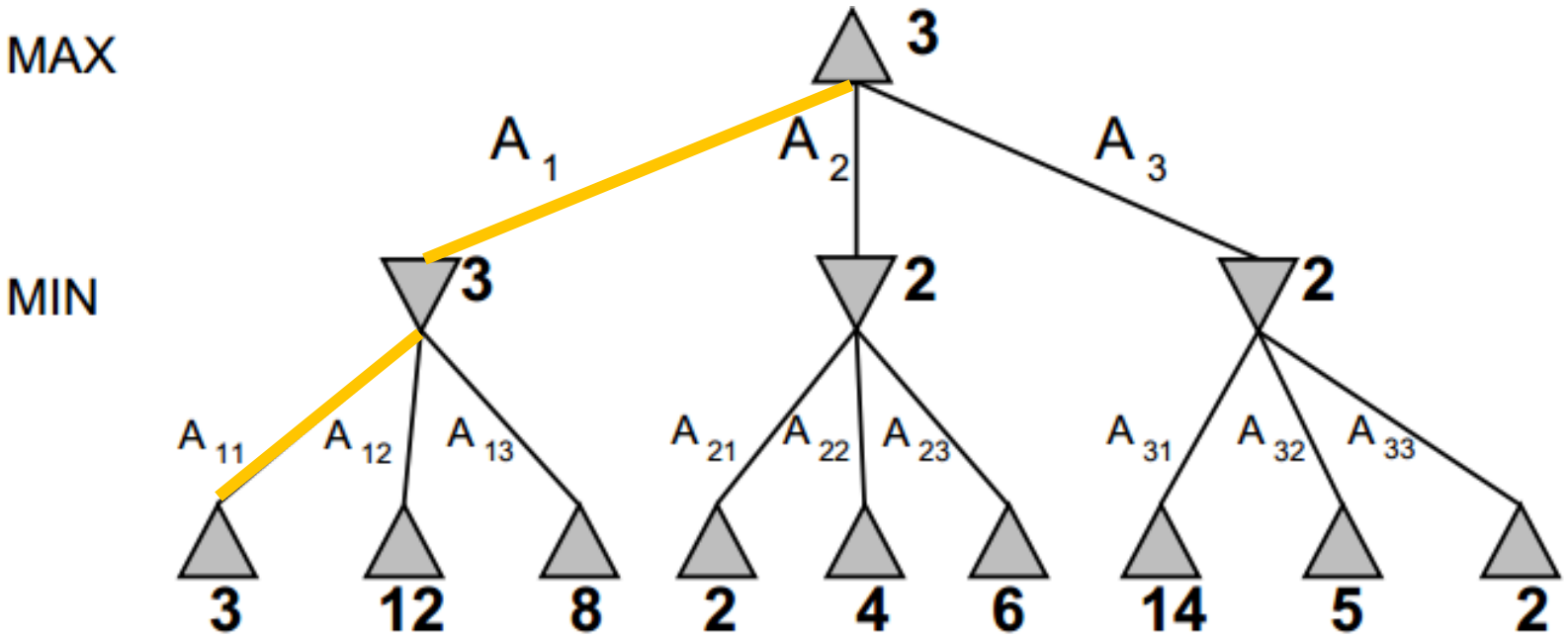
Minimax: Small Example



Minimax: Small Example



Minimax: Small Example



Properties of minimax

- Complete? Yes (if tree is finite)
- Optimal? Yes (against an rational opponent)
- Time complexity? $O(b^m)$
- Space complexity? O(bm) (depth-first exploration)

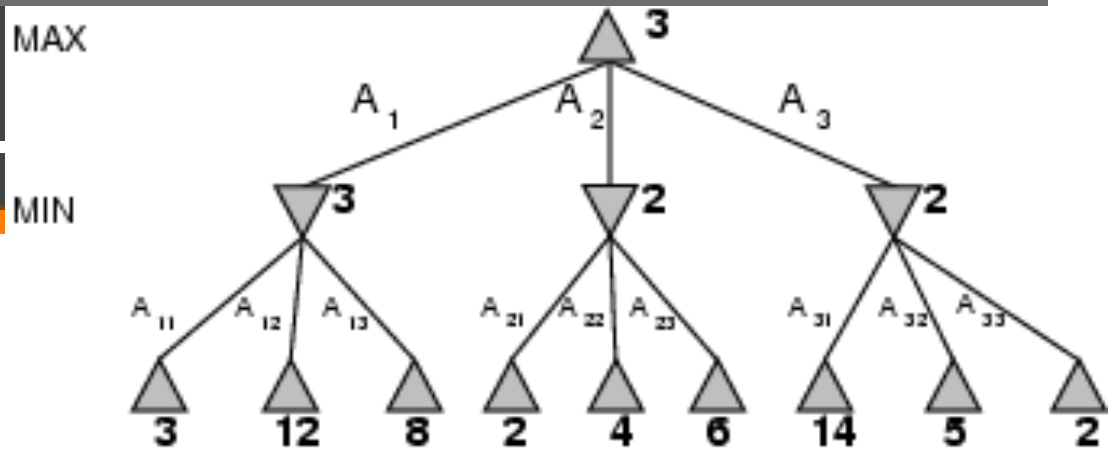
- For chess, b ≈ 35, m ≈100 for "reasonable" games
  → exact solution completely infeasible

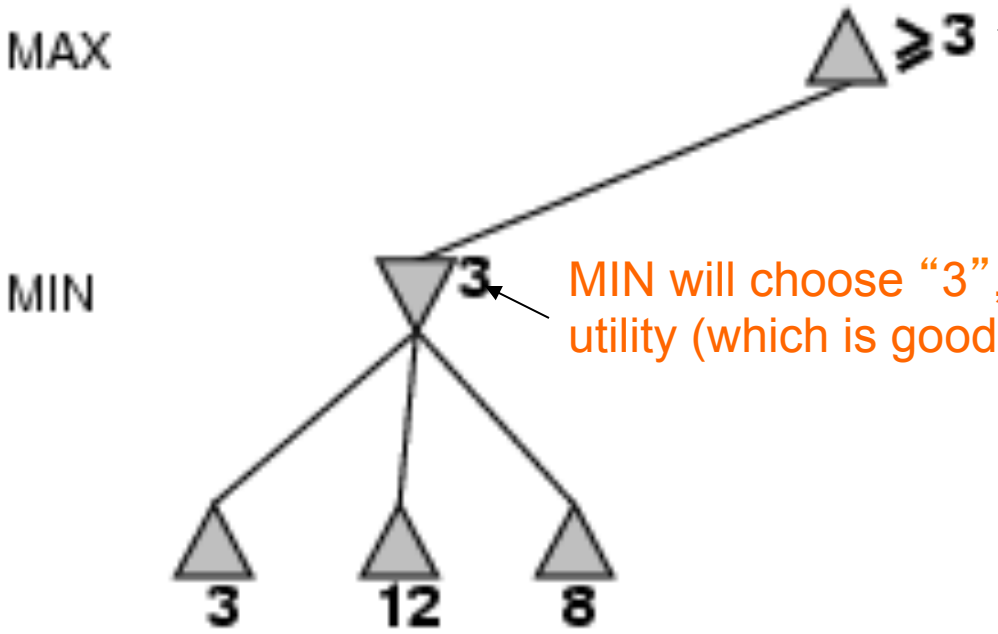
$\alpha - \beta$ Pruning

1. Do we need to expand all nodes?

2. No: We can do better by pruning branches that will not lead to success.

α-β pruning ex.

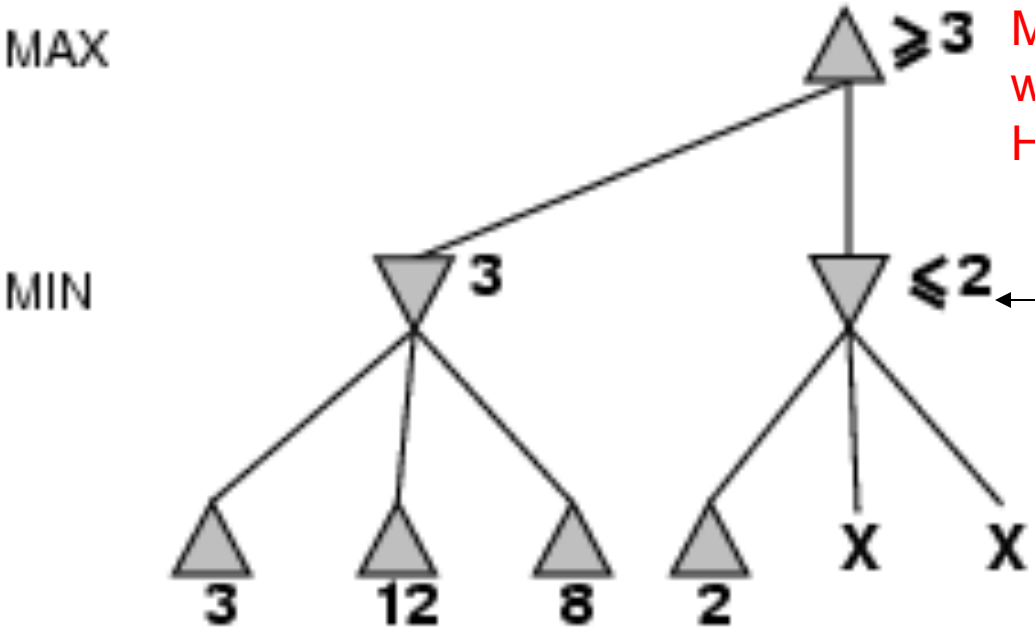




MAX knows that it can at least get "3" by playing this branch

MIN will choose "3", because it minimizes the utility (which is good for MIN)

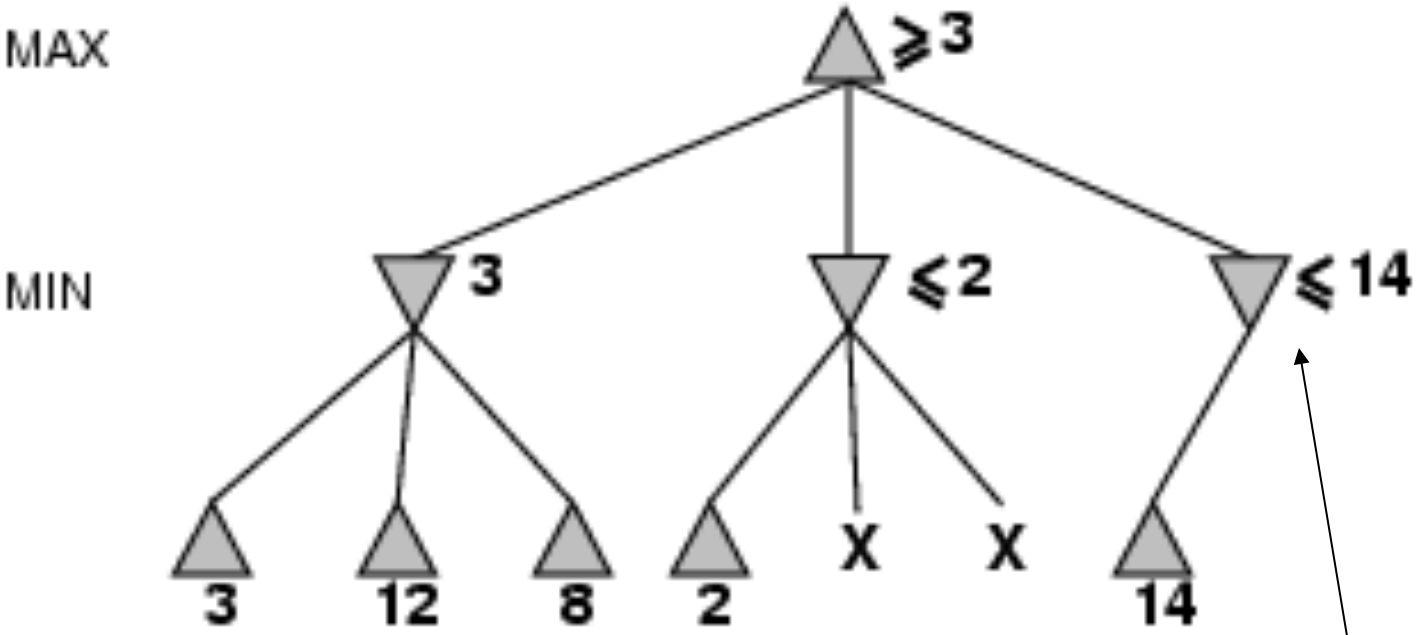
α-β pruning example



MAX knows that the new branch will never be better than 2 for him. He can *ignore* it.

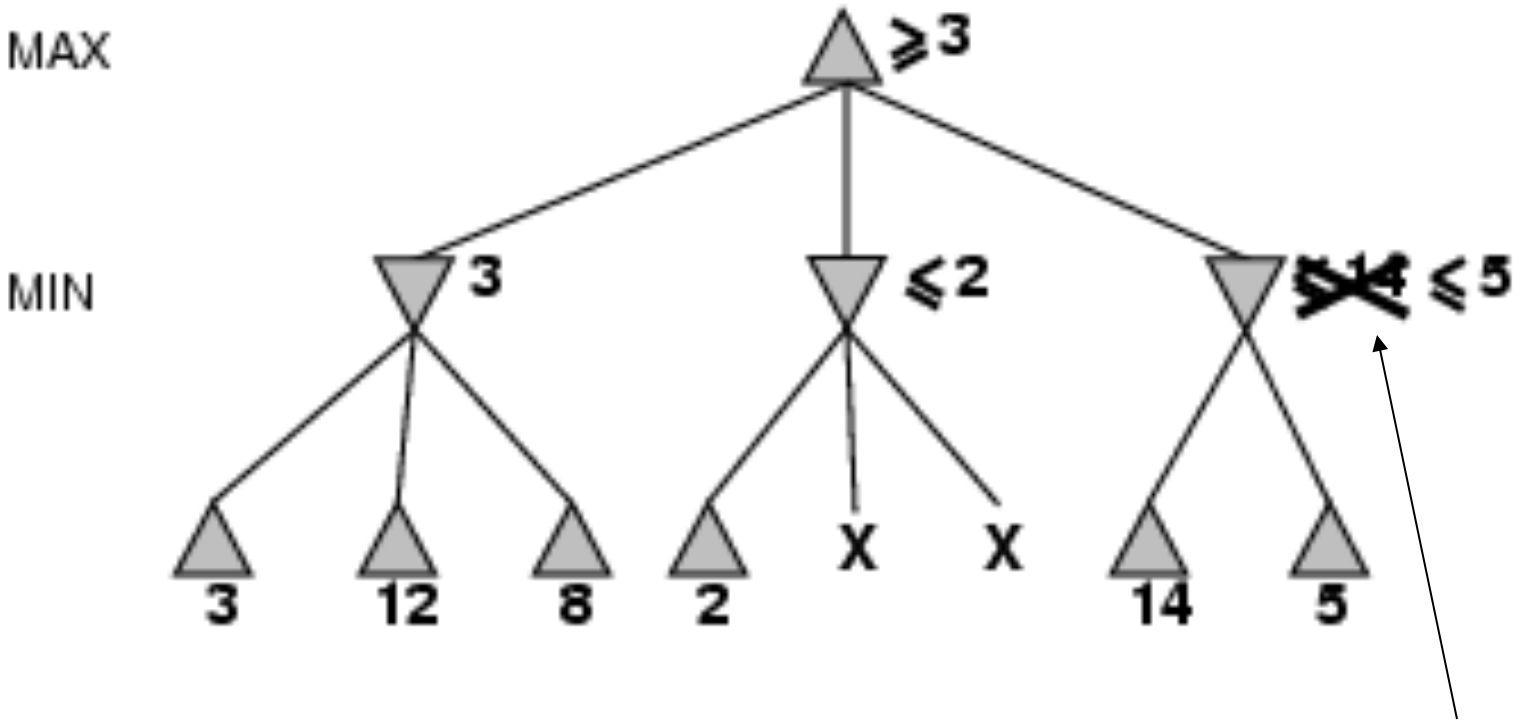
MIN can certainly do as good as 2, but maybe better (= smaller)

α-β pruning example



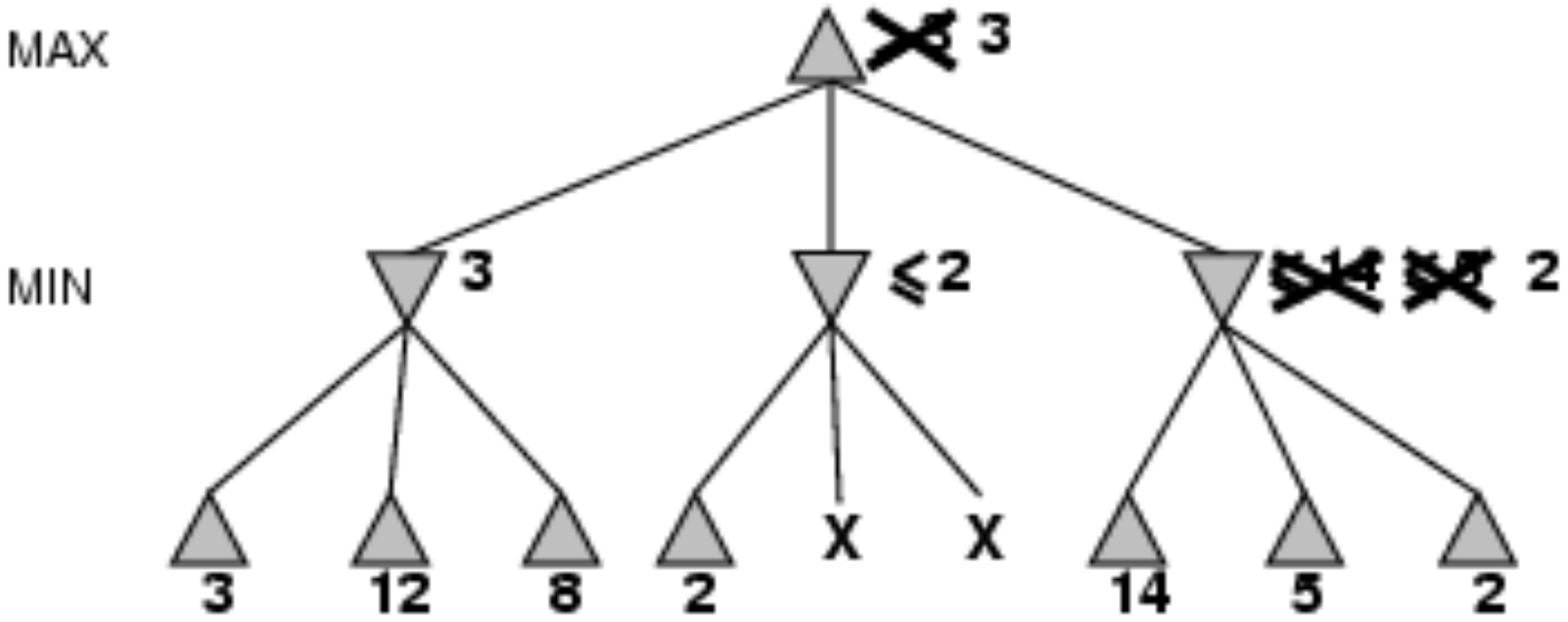
MIN will do at least as good as 14 in this branch (which is very good for MAX!) so MAX will want to explore this branch more.

α-β pruning example



MIN will do at least as good as 5 in this branch (which is still good for MAX) so MAX will want to explore this branch more.

α-β pruning example



Bummer (for MAX): MIN will be able to play this last branch and get 2. This is worse than 3, so MAX will play 3.

Properties of α-β

- Pruning does not affect final result (it is exact).
- Order is important
- With "perfect ordering," time complexity = $O(b^{m/2})$
  → doubles depth of search

The Algorithm

- Visit the nodes in a depth-first manner
- Maintain bounds on nodes.
- A bound may change if one of its children obtains a unique value.
- A bound becomes a unique value when all its children have been checked or pruned.
- When a bound changes into a tighter bound or a unique value, it may become inconsistent with its parent.
- When an inconsistency occurs, prune the sub-tree by cutting the edge between the inconsistent bounds/values.

→ This is like propagating changes bottom-up in the tree.

Practical Implementation

How do we make this practical?

Standard approach:

- cutoff test: (where do we stop descending the tree)
  - depth limit
  - better: iterative deepening
  - cutoff only when no big changes are expected to occur next (quiescence search).

Practical Implementation

- evaluation function/heuristics
  - When the search is cut off, we evaluate the current state by estimating its utility. This estimate is captured by the evaluation function.
  - Run α-β pruning minimax with these estimated values at the leaves instead.

Evaluation functions

- For chess, typically linear weighted sum of features

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

- e.g., $w_1 = 9$ with

  $f_1(s)$ = (number of white queens) – (number of black queens), etc.

Forward Pruning & Lookup

- Humans don't consider all possible moves.
- Can we prune certain branches immediately?
- "ProbCut" estimates (from past experience) the uncertainty in the estimate of the node's value and uses that to decide if a node can be pruned.
- Instead of search one can also store game states.
- Openings in chess are played from a library

In Practice

	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

Deterministic games in practice

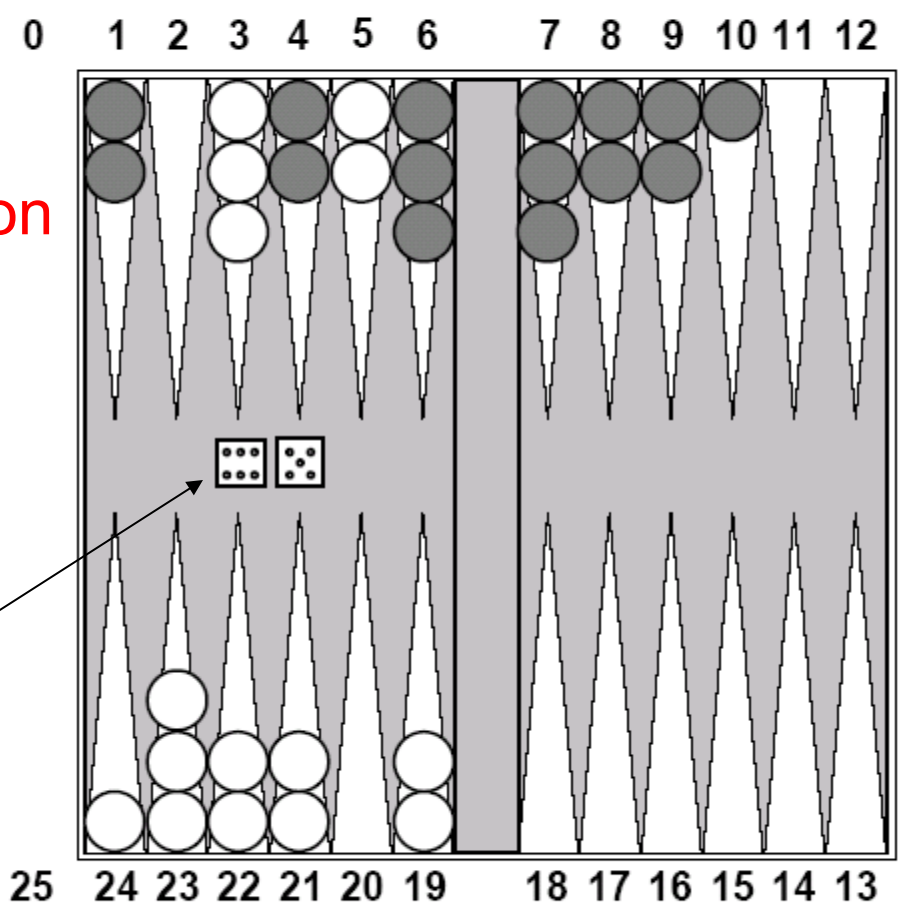
- **Checkers**: Chinook defeated champion in 1990 (Schaeffer)
- **Chess**: Deep Blue defeated human world champion Garry Kasparov in a six-game match in 1997.
- **Othello**: Logistello (Buro 2002) beat champion
- **Poker**: Machine was better than best human poker players in 2008.

What about chance games?

	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

Chance Games

Backgammon

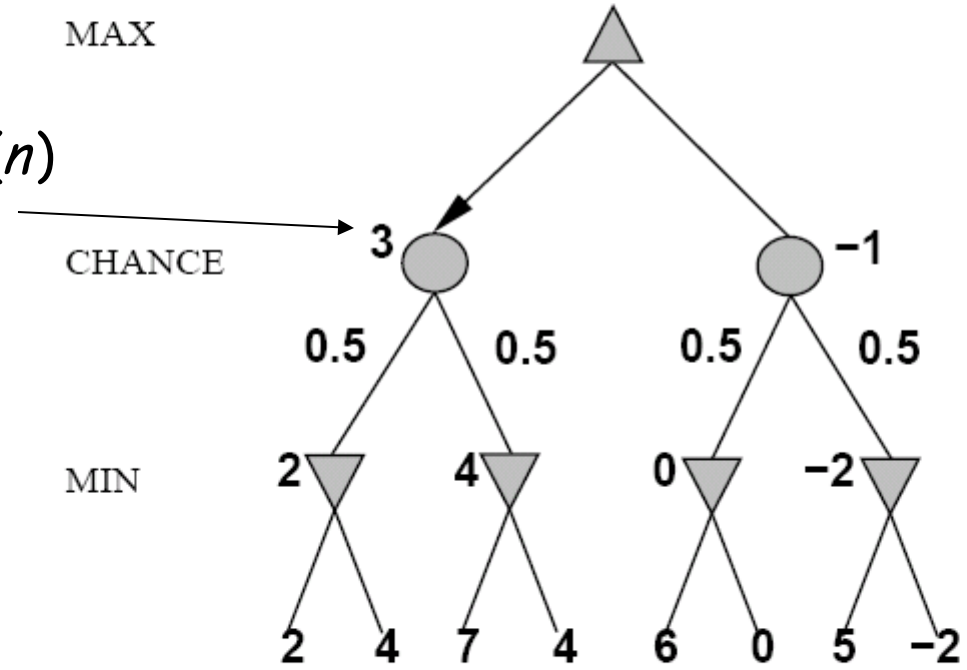


your element of chance

Expected Minimax

$$v = \sum_{\text{chance nodes}} P(n) \times \text{Minimax}(n)$$

$$3 = 0.5 \times 4 + 0.5 \times 2$$



Again, the tree is constructed bottom-up.

Now we have even more nodes to search!

What about imperfect Information?

	Perfect Information	Imperfect Information
Deterministic	Chess Checkers Go Othello	Battleship Mastermind
Stochastic	Backgammon Monopoly Snakes and Ladders	Poker Bridge Scrabble Civilization

Class Assignment

- Blackjack, see http://en.wikipedia.org/wiki/Blackjack
- First identify if the game is deterministic/stochastic, with perfect or imperfect information
- Then figure out an algorithm to develop it
- When you are done, write it up and call me to discuss

Class Assignment

- Read

http://www.eng.ysu.edu/~drwallac/ENGR1550AU10/ENGR%201550%20-%20IE%20Blackjack%20Lab.pdf