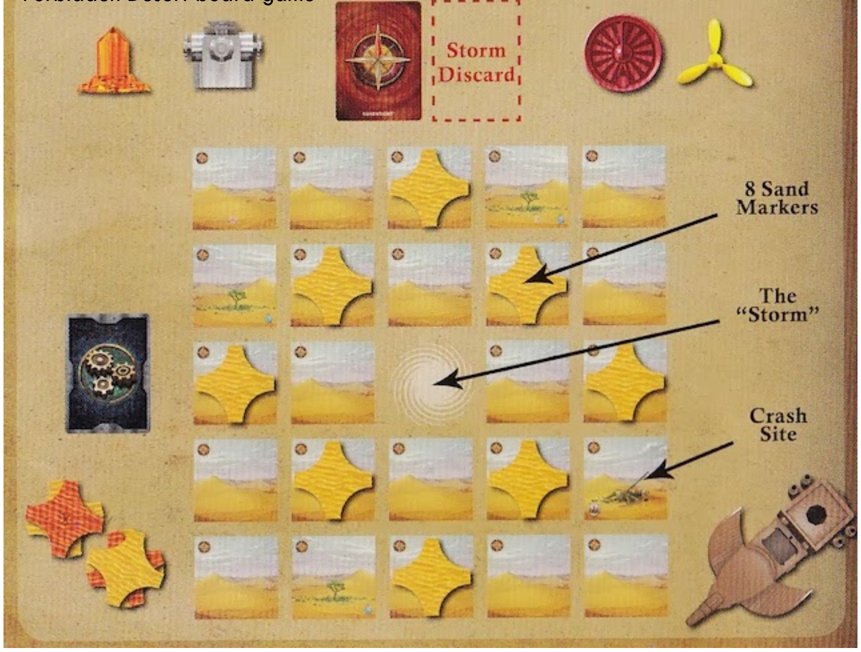
ADVERSARIAL SEARCH

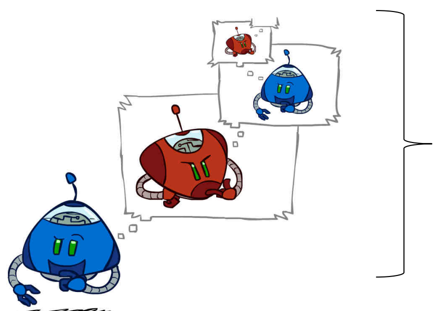
Forbidden Desert board game



Today

- Reading
  - AIMA Chapter 5.1-5.5, 5.7,5.8
- Goals
  - alpha-beta pruning
  - Finish real-time decisions
  - Stochastic games

Minimax: an optimal strategy



If I did this, then he would do that, but then I would do that, and then he would do this…

$$\text{MINIMAX(s)} = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_a \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_a \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

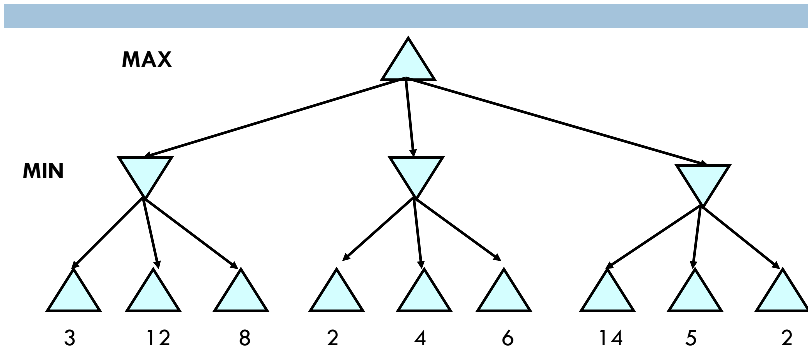
Minimax: An Optimal Strategy

function MINIMAX-DECISION(*state*) **returns** *an action*
  $v \leftarrow$ MAX-VALUE(*state*)
  return the *action* in SUCCESSORS(*state*) with value v

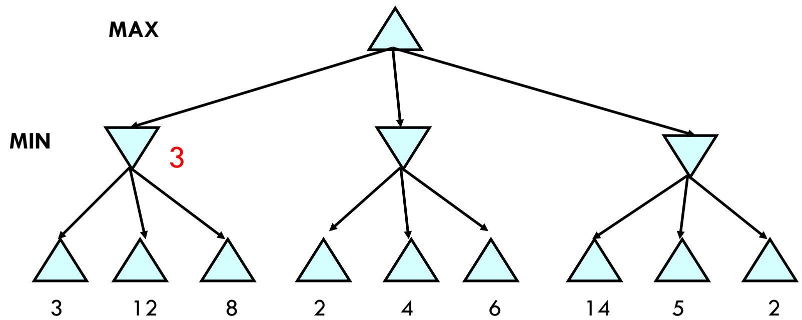
function MAX-VALUE(*state*) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s))
  return v

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**
    $v \leftarrow$ MIN(v, MAX-VALUE(s))
  return v

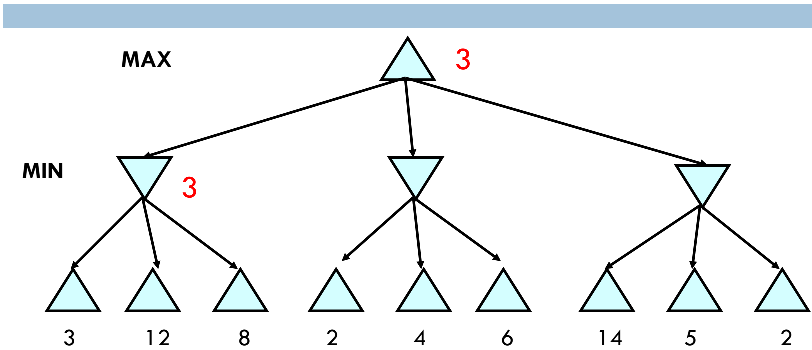
Minimax example



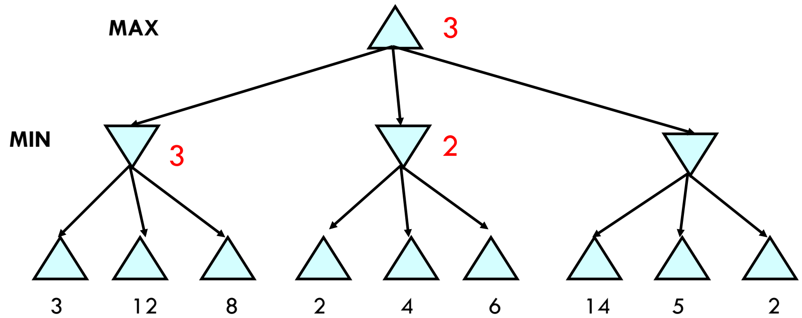
Minimax example



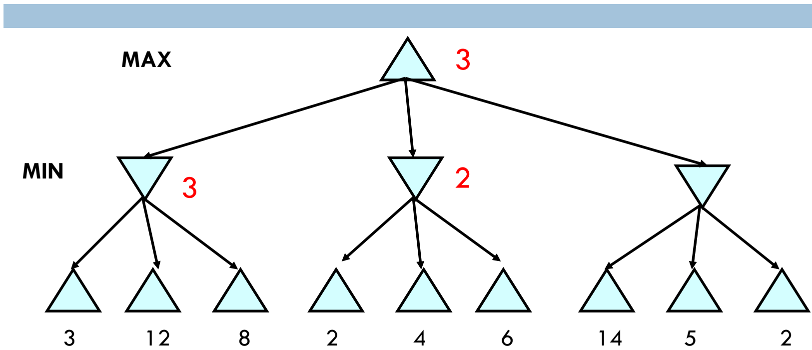
Minimax example



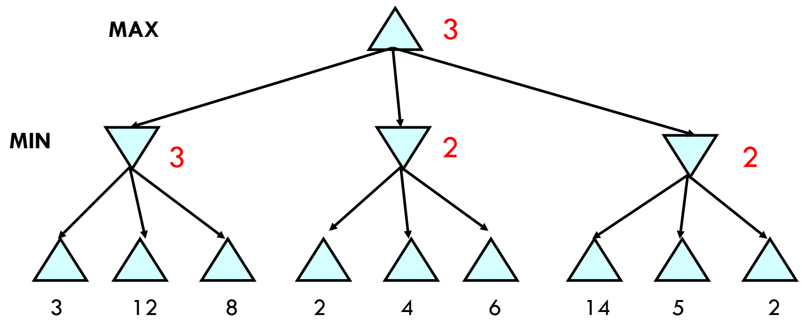
Minimax example



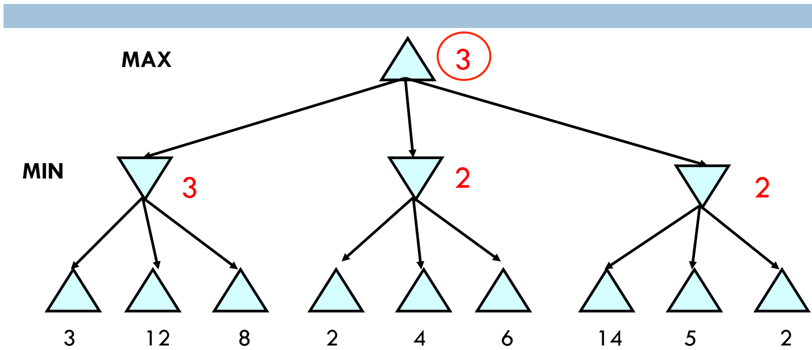
Minimax example



Minimax example



Minimax example



Complexity of Minimax

- Minimax performs DFS of search tree
  - Time $O(b^m)$
  - Tic-tac-toe: ~5 legal moves, 9 moves/game
    - $5^9 = 1{,}953{,}125$ states
  - Chess: ~35 legal moves, ~100 moves/game
    - 35^{100} states to search
- Common games produce enormous search trees

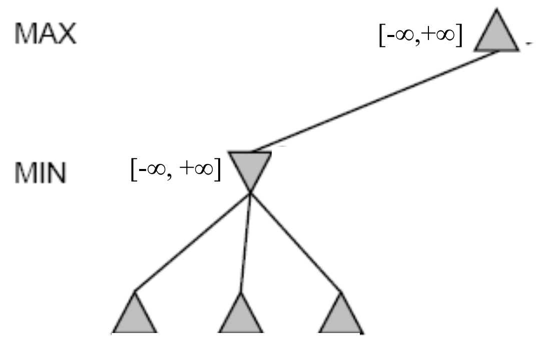
Alpha-Beta pruning

- alpha is the best scenario MAX has found so far
  - MAX can always achieve a utility of alpha (and hopes for higher)
- beta is the best scenario MIN has found so far
  - MIN can always achieve a utility of beta (and hopes for lower)

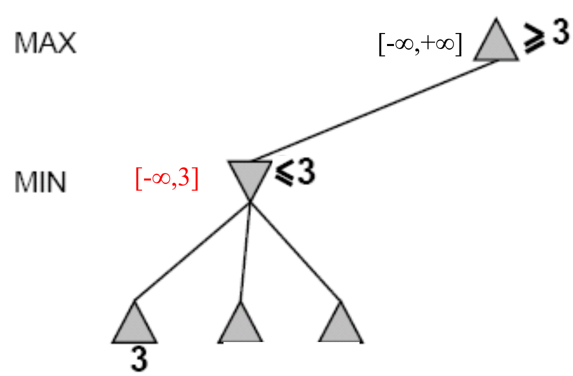
$$[\alpha, \beta]$$

Alpha-Beta Example

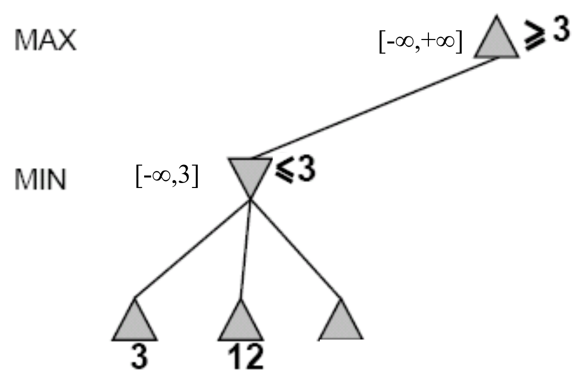
Do depth-first search until first leaf



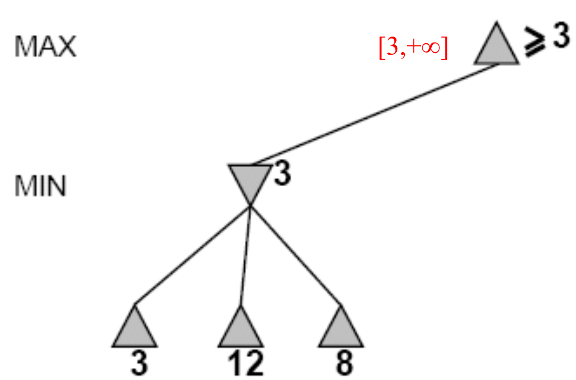
Alpha-Beta Example



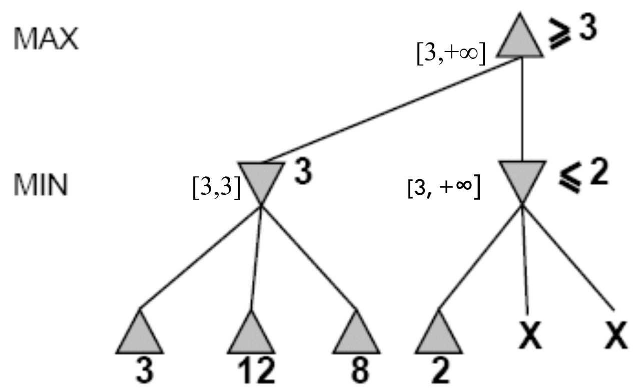
Alpha-Beta Example



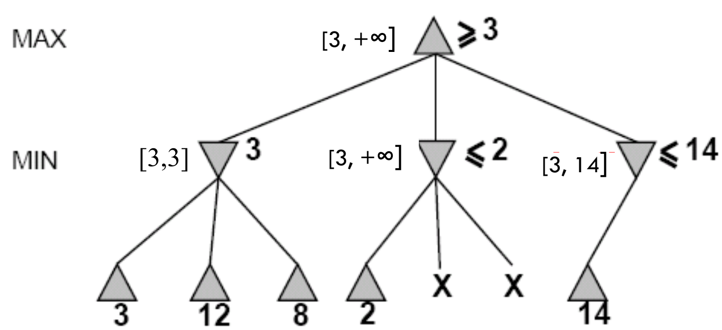
Alpha-Beta Example



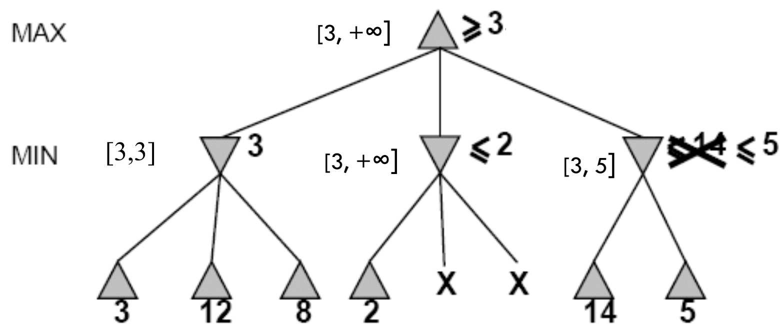
Alpha-Beta Example



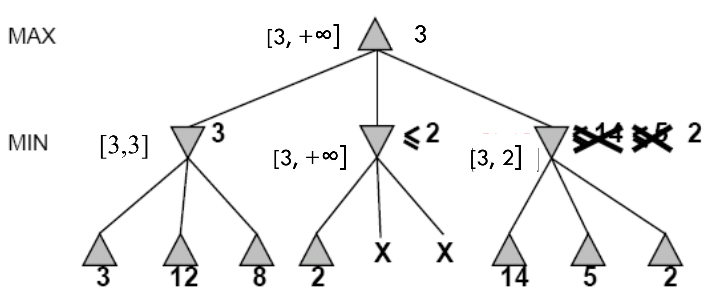
Alpha-Beta Example



Alpha-Beta Example



Alpha-Beta Example



Alpha-Beta pruning

```
function ALPHA-BETA-SEARCH(state) returns an action
  inputs: state, current state in game

  v ← MAX-VALUE(state, −∞, +∞)
  return the action in SUCCESSORS(state) with value v
```

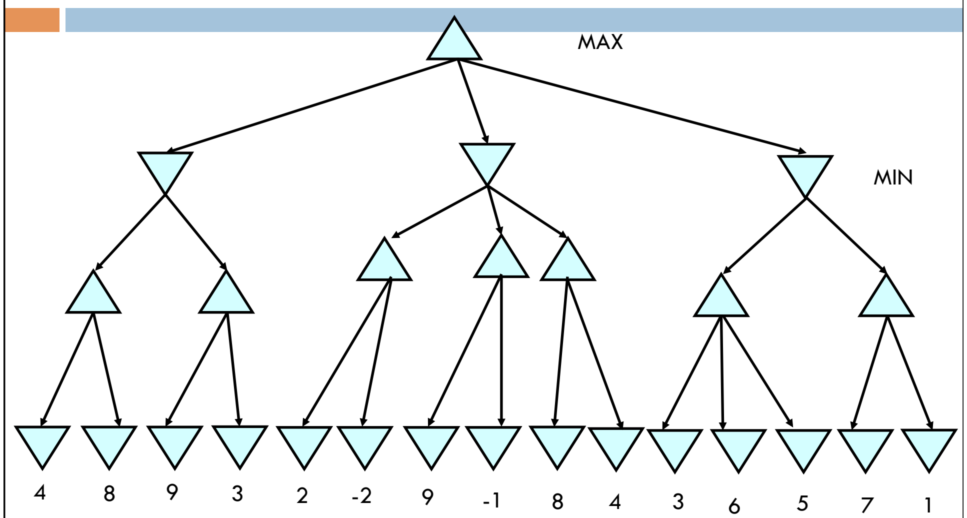
```
function MAX-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  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) 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 ← +∞
  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
```

Alpha-Beta pruning example

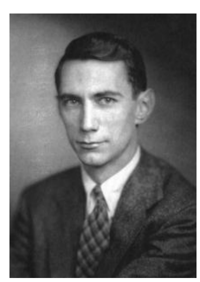


Effectiveness of Alpha-Beta pruning

- Highly-dependent on the order in which the states are examined
- Try to examine those states that are likely to be best
- In practice, the running time for alpha-beta pruning is $O(b^{m/2})$ as opposed to $O(b^m)$
  - Effective branching factor is square root of b
  - Can search twice as deep as minimax

Real-time decision making

- Alpha-beta pruning still has to search down to the leaf nodes (for part of the search tree)
- Standard approach (Shannon, 1950):
  - apply a cutoff test (turn non-leaf nodes into leaves)
  - replace utility function by an evaluation function that estimates "desirability" of position



Claude Shannon

Pioneered the field of information theory, introduced the major problems in information theory, and simultaneously solved all the problems in one fantastic publication.

Real-time decision making

$$\text{MINIMAX(s)} = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_a \text{MINIMAX(RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_a \text{MINIMAX(RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

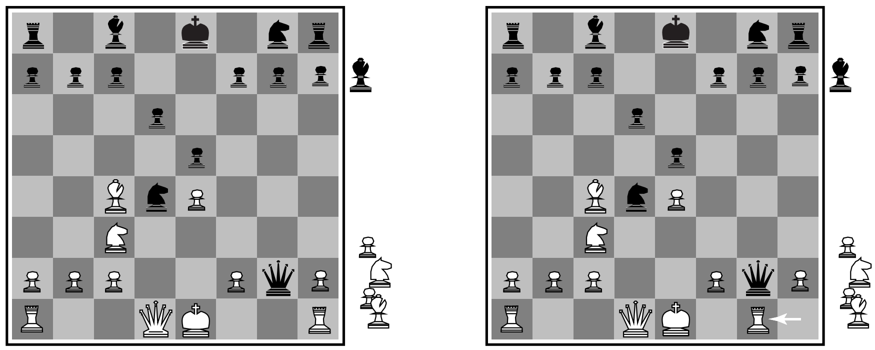
$$\text{H-MINIMAX(s,d)} = \begin{cases} \textbf{EVAL}(s) & \text{if } \textbf{CUTOFF-TEST}(s,d) \\ \max_a \text{H-MINIMAX(RESULT}(s,a), \mathbf{d+1}) & \text{if PLAYER}(s) = \text{MAX} \\ \min_a \text{H-MINIMAX(RESULT}(s,a), \mathbf{d+1}) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

Evaluation function

- Estimates utility of game from truncated position
  - Order *terminal* states in same manner
  - Fast to compute
  - For non-terminal states, correlated with the truth
- Weighted linear combination of features
  - independence assumption

$$\text{EVAL}(s) = w_1 f_1(s) + \ldots + w_n f_n(s) = \sum_{i=1}^{n} w_i f_i(s)$$

Heuristic difficulties



(a) White to move

(b) White to move

Cutoff tests – when to stop?

- At a fixed depth
- Iterative deepening
  - Report the result of the last IDS search that was fully completed
- Horizon effect
  - Pushing off the inevitable
- Quiescence search
  - Stop at quiescent (quiet) positions
  - Focus on non-quiescent positions

Stochastic Games

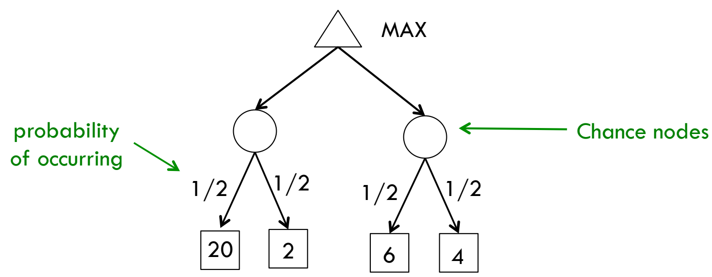
Stochastic games

- Stochastic games include an element of chance
  - e.g. dice, unpredictable or random opponents
- Example 1-player and 2-player stochastic games
  - solitaire, minesweeper, backgammon, pacman

How do we find the optimal strategy in the presence of uncertainty?

Stochastic games

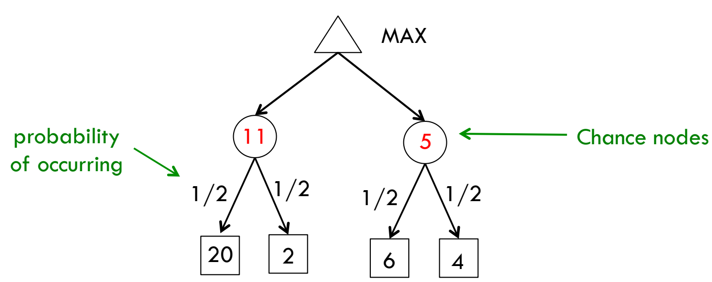
- How do we find the optimal strategy in this case?
  - Change minimax tree to include chance nodes



- Compute average (expected) utility
  - e.g. ½(20) + ½(2) = 11

Stochastic games

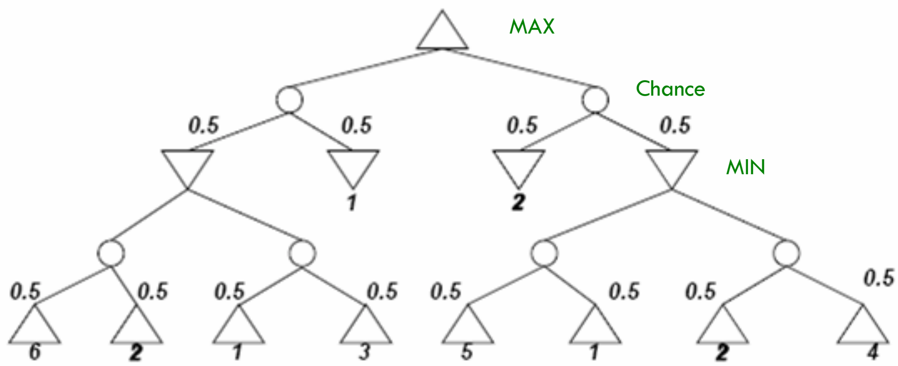
- How do we find the optimal strategy in this case?
  - Update minimax tree to include chance nodes



- Compute average (expected) utility
  - e.g. ½(20) + ½(2) = 11

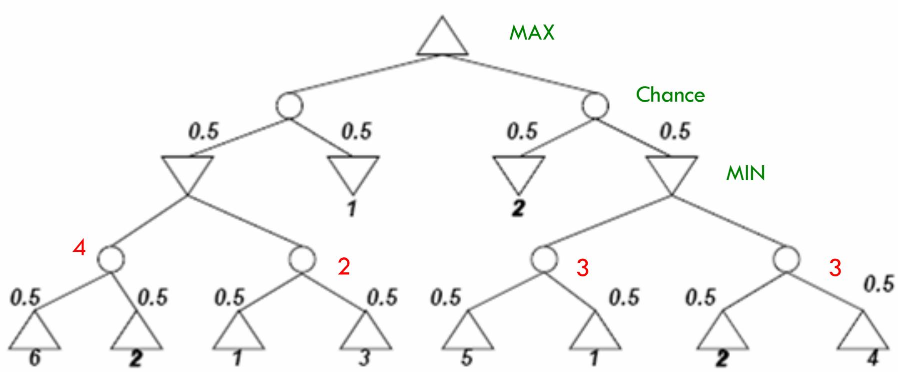
Stochastic games

- 2-person stochastic game: MAX, MIN, chance nodes



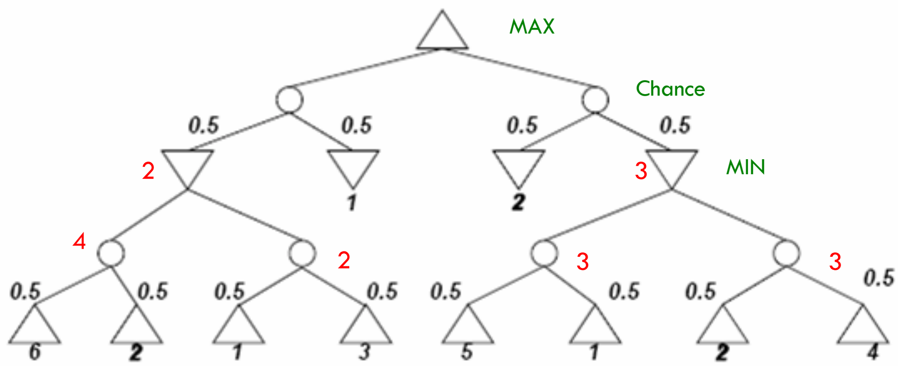
Stochastic games

- For a 2-person stochastic game: MAX, MIN, chance



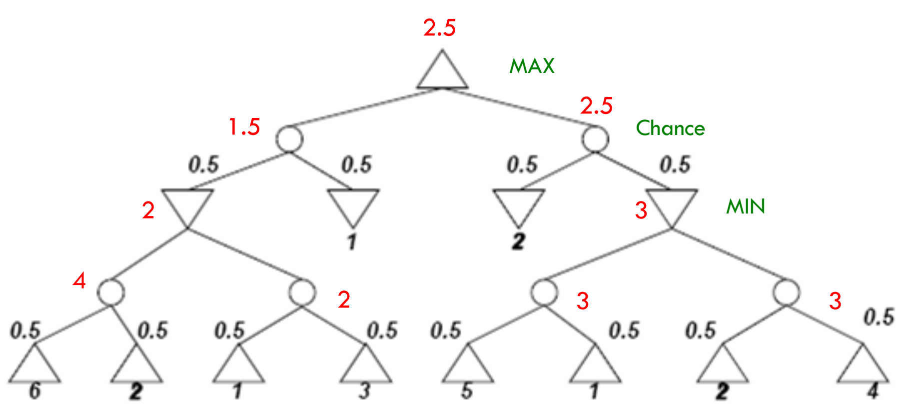
Stochastic games

- For a 2-person stochastic game: MAX, MIN, chance



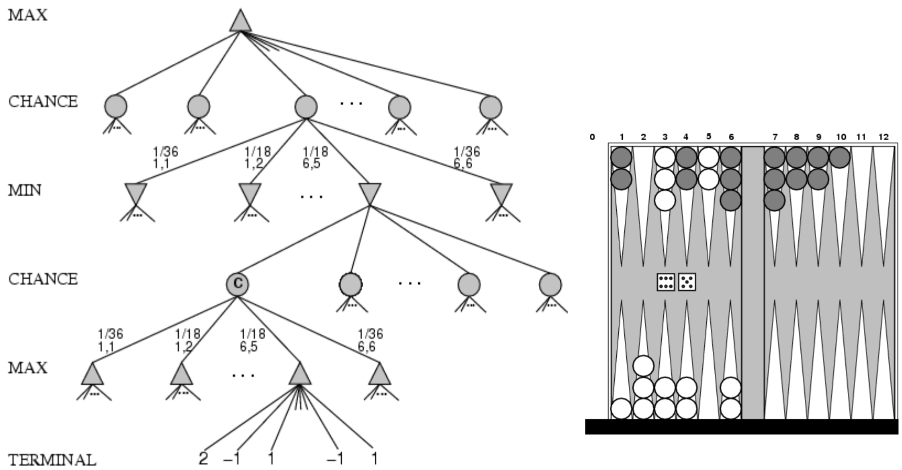
Stochastic games

- For a 2-person stochastic game: MAX, MIN, chance



Example: Backgammon

- White rolls 6-5: (5-10, 5-11), (5-11,19-24), …



ExpectiMinimax

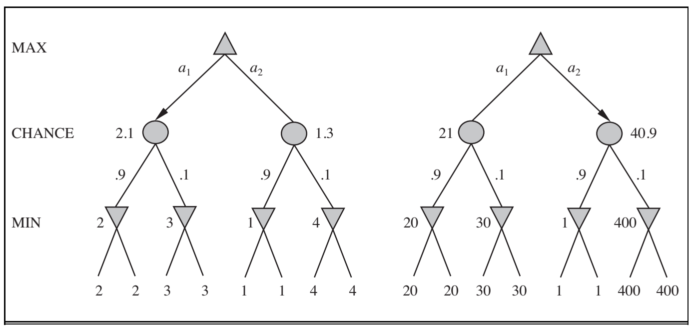
- Update minimax strategy to compute weighted average (expected value) at chance nodes
- Gives the expected value of a position/move

$$\text{EXPECTIMINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \\ \sum_r P(r) \cdot \text{EXPECTIMINIMAX}(\text{RESULT}(s,r)) & \text{if PLAYER}(s) = \text{CHANCE} \end{cases}$$

- $O(b^m n^m)$ where n is number of distinct outcomes

ExpectiMinimax in real-time

- Recall, real-time games need to make fast decisions
- For minimax, scale of the evaluation function doesn't matter
- For expectiMinimax, the scale is important



Summary of adversarial games

- Talked mostly about zero-sum games
  - If I win then you lose
- Minimax is optimal strategy but often too slow
  - Alpha-beta pruning can increase max depth by factor of 2
  - Implement evaluation function and cutoff-tests for early stop and guided search
- Expectiminimax used for games with an element of chance/randomness