ADVERSARIAL SEARCH

Today

- Introduce adversarial games
- Minimax as an optimal strategy
- Alpha-beta pruning
- Real-time decision making

Adversarial Games

- People like games!
- Games are fun, engaging, and hard-to-solve
- Games are amenable to study: precise, easy-to-represent state space

Game pieces found in a burial site in Southeast Turkey. Dated about 3000 BC

"Game of Twenty squares" discovered in a burial site in Ur. Dated about 2550-2400 BC

Backgammon is also among one of the oldest games still played today

Adversarial Games

- Two-player games have been a focus of AI as long as computers have been around

Checkers

Solved: state space was completely mapped out!

Backgammon and Chess

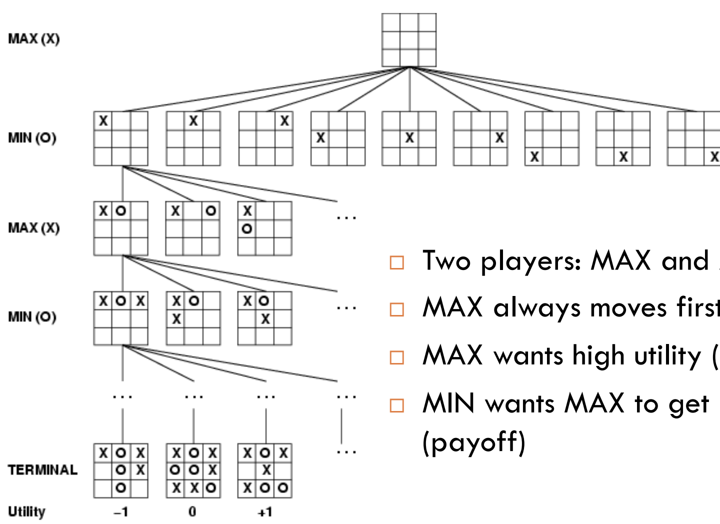
Computers can compete at a championship level

Go

Can play at professional level (2015)

Terminology

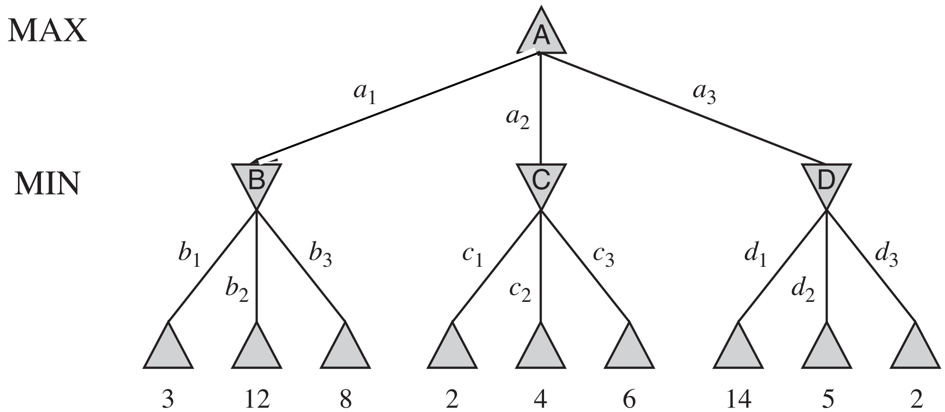
Game Tree



1-ply

- Two players: MAX and MIN
- MAX always moves first
- MAX wants high utility (payoff)
- MIN wants MAX to get low utility (payoff)

Minimax: an optimal policy

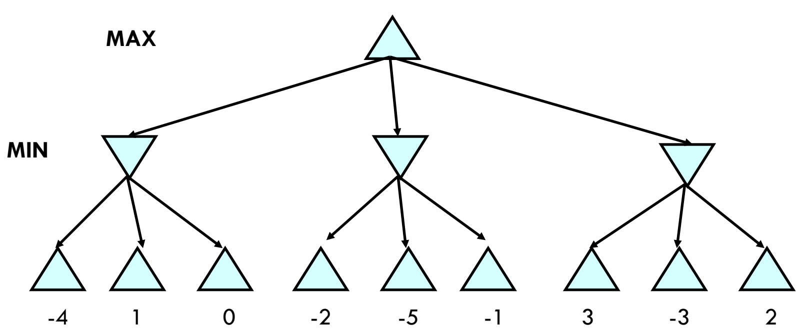


What action should MAX take?

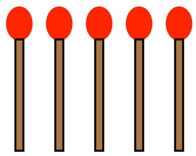
Minimax: an optimal policy

Minimax Algorithm

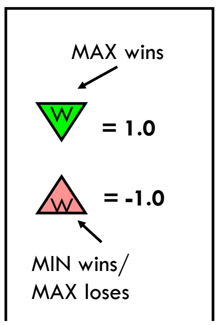
Minimax Example



Minimax Example: Baby Nim



Take 1 or 2 at each turn
Goal: take the last match

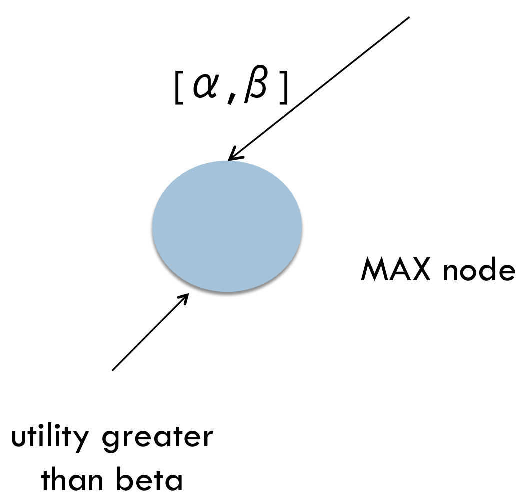


Properties of Minimax

- Minimax performs depth-first exploration of game tree.
  - Recall time complexity for DFS is $O(b^m)$
- For chess, $b \approx 35$, $d \approx 100$ for "reasonable" games
  - exact solution completely infeasible
- How can we find the exact solution faster?

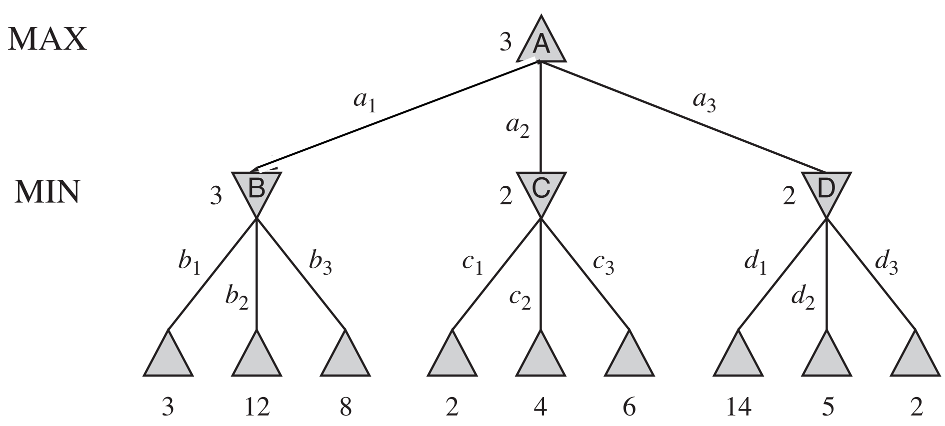
Alpha-beta Pruning

When can we prune?



Alpha-beta Pruning Algorithm

Alpha-beta Example



Properties of α - β

- Pruning *does not* affect final result
- Effectiveness affected by order in which we examine successors
- Exponent reduces to m/2 or 3m/4
- What do you do if you don't get to the bottom of the tree on time?

Real-time decision making

- "Programming a computer for playing chess"
- Claude Shannon, 1950
- Truncate (apply cutoff test) and estimate utility
- Called an evaluation function

Real-time decision making

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

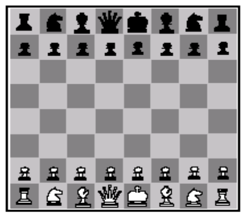
$$\text{H-MINIMAX(s,d)} = \begin{cases} \textbf{EVAL}(s) & \text{if } \textbf{CUTOFF-TEST}(s,d) \\ \max_a \text{H-MINIMAX(RESULT}(s,a), \mathbf{d+1}) & \text{if PLAYER}(s) = \text{MAX} \\ \min_a \text{H-MINIMAX(RESULT}(s,a), \mathbf{d+1}) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

Evaluation function

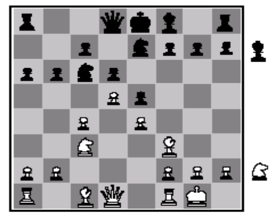
- Estimates utility of game from truncated position
  - Order *terminal* states in same manner
  - Fast to compute
  - For non-terminal states, correlated with the truth
- Weighted linear combination of features
  - independence assumption

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

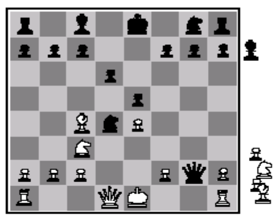
Heuristic EVAL example



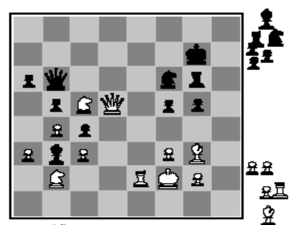
(a) White to move
Fairly even



(b) Black to move
White slightly better

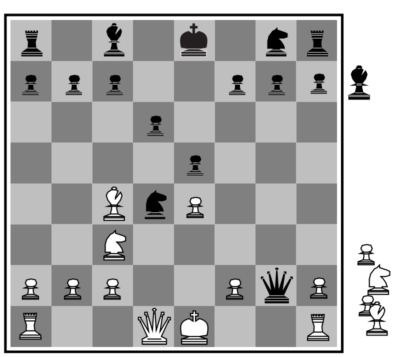


(c) White to move
Black winning

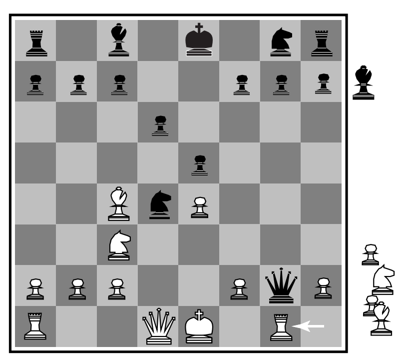


(d) Black to move
White about to lose

Heuristic difficulties



(a) White to move



(b) White to move

Summary

- Minimax is an optimal strategy but requires full traversal of game tree
- Alpha-beta pruning
  - Effectively reduces branching factor
- In reality, probably need to use an evaluation function