# CSE 473: Artificial Intelligence
Spring 2012

Adversarial Search

Dan Weld

Based on slides from

Dan Klein, Stuart Russell, Andrew Moore and Luke Zettlemoyer

## Today

- Adversarial Search
  - Minimax search
  - α-β search
  - Evaluation functions
  - Expectimax
- Reminder:
  - Programming 1 due tonight

## Game Playing State-of-the-Art

- **Checkers:** Chinook ended 40-year-reign of human world champion Marion Tinsley in **1994**. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions. Checkers is now solved!

## Game Playing State-of-the-Art

- **Chess:** Deep Blue defeated human world champion Gary Kasparov in a six-game match in **1997**. Deep Blue examined 200 million positions per second, used very sophisticated evaluation and undisclosed methods for extending some lines of search up to 40 ply. Current programs are even better, if less historic.

## Game Playing State-of-the-Art

- **Othello:** Human champions refuse to compete against computers, which are too good.
- **Go:** Human champions are beginning to be challenged by machines, though the best humans still beat the best machines on the full board. In go, b > 300, so need pattern knowledge bases and monte carlo search (UCT)
- **Pacman:** unknown

## Types of Games

| | deterministic | chance |
|---|---|---|
| perfect information | chess, checkers, go, othello | backgammon, monopoly |
| imperfect information | stratego | bridge, poker, scrabble, nuclear war |

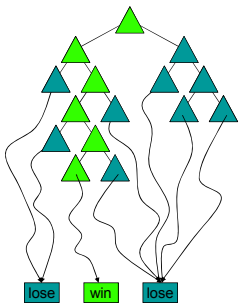
Number of Players? 1, 2, …?

## Deterministic Games

- Many possible formalizations, one is:
  - States: S (start at $s_0$)
  - Players: P={1...N} (usually take turns)
  - Actions: A (may depend on player / state)
  - Transition Function: S x A → S
  - Terminal Test: S → {t,f}
  - Terminal Utilities: S x P→ R
- Solution for a player is a ***policy***: S → A

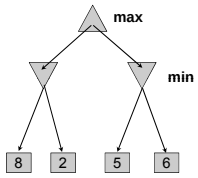
## Deterministic Single-Player

- Deterministic, single player, perfect information:
  - Know the rules, action effects, winning states
  - E.g. Freecell, 8-Puzzle, Rubik's cube
- … it's just search!
- Slight reinterpretation:
  - Each node stores a value: the best outcome it can reach
  - This is the maximal outcome of its children (the max value)
  - Note that we don't have path sums as before (utilities at end)
- After search, can pick move that leads to best node

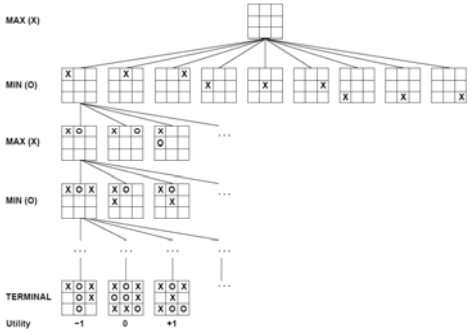



## Deterministic Two-Player

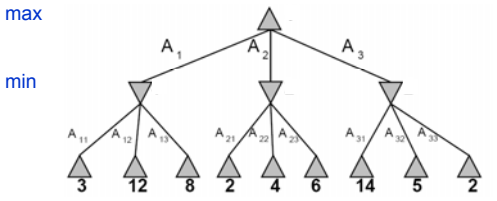
- E.g. tic-tac-toe, chess, checkers
- Zero-sum games
  - One player maximizes result
  - The other minimizes result
- **Minimax search**
  - A state-space search tree
  - Players alternate
  - Choose move to position with highest minimax value = best achievable utility against best play



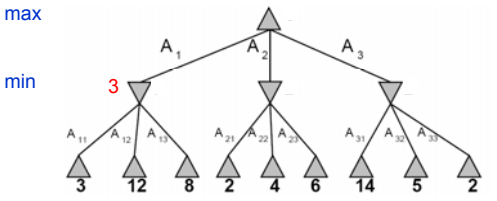

## Tic-tac-toe Game Tree



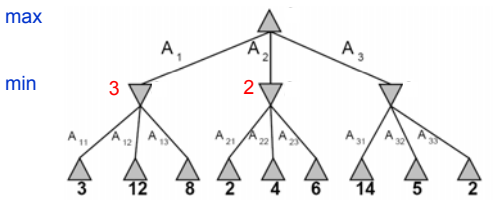

## Minimax Example



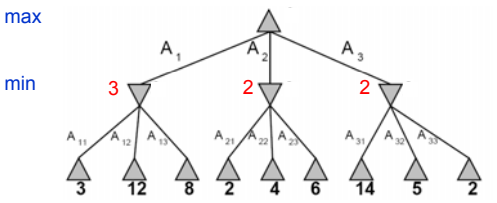

## Minimax Example



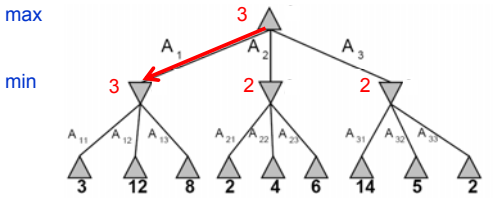
## Minimax Example




## Minimax Example




## Minimax Example




## Minimax Search

```
function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for a, s in SUCCESSORS(state) do v ← MAX(v, MIN-VALUE(s))
  return v

function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for a, s in SUCCESSORS(state) do v ← MIN(v, MAX-VALUE(s))
  return v
```

## Minimax Properties

- Optimal?
  - Yes, against perfect player. Otherwise?
- Time complexity?
  - $O(b^m)$
- Space complexity?
  - O(bm)
- For chess, b ~ 35, m ~ 100
  - Exact solution is completely infeasible
  - But, do we need to explore the whole tree?