Games vs. Search problems

- **“Unpredictable” opponent**: Solution is a contingency plan
- **Time limits**: Unlikely to find goal, must approximate
- **Game types**:

	deterministic	**chance**
perfect information	**chess, checkers, go, othello**	**backgammon monopoly**
imperfect information		**bridge, poker, scrab nuclear war**

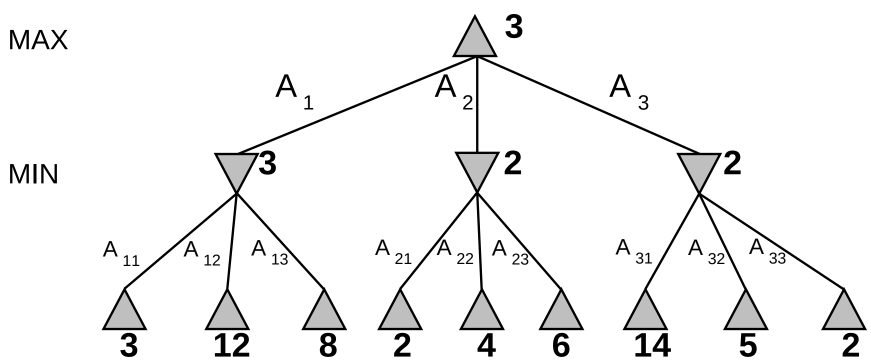
Artificial Intelligence

Adversarial Search

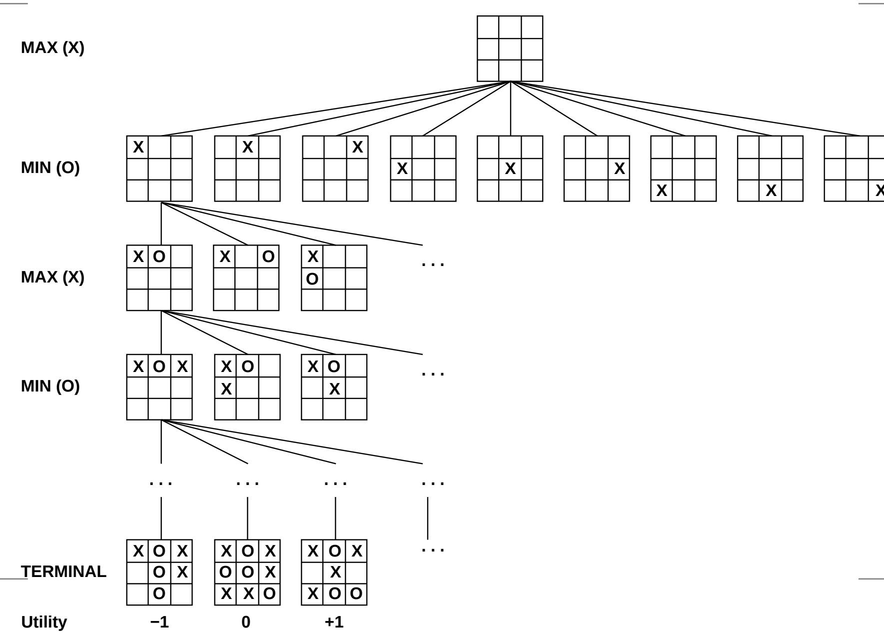
Readings: Chapter 6 of Russell & Norvig.

Minimax

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



Tic-Tac-Toe



Properties of Minimax

- *Complete*: Yes, if tree is finite (chess has specific rules for this)
- *Optimal*: Yes, against an optimal opponent. Otherwise??
- *Time complexity*: $O(b^m)$
- *Space complexity*: $O(bm)$ (depth-first exploration)

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

Minimax Algorithm

function MINIMAX-DECISION(*game*) **returns** *an operator*

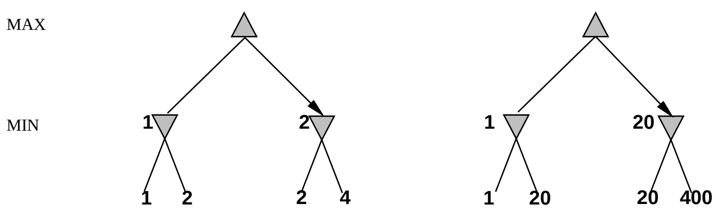
  for each *op* **in** OPERATORS[*game*] **do**
    VALUE[*op*] $\leftarrow$ MINIMAX-VALUE(APPLY(*op*, *game*), *game*)
  end
  return the *op* with the highest VALUE[*op*]

---

function MINIMAX-VALUE(*state*, *game*) **returns** *a utility value*

  if TERMINAL-TEST[*game*](*state*) **then**
    return UTILITY[*game*](*state*)
  else if MAX is to move in *state* **then**
    return the highest MINIMAX-VALUE of SUCCESSORS(*state*)
  else
    return the lowest MINIMAX-VALUE of SUCCESSORS(*state*)

Digression: Exact values don't matter



- Behaviour is preserved under any *monotonic* transformation of EVAL
- Only the order matters: payoff in deterministic games acts as an *ordinal utility* function

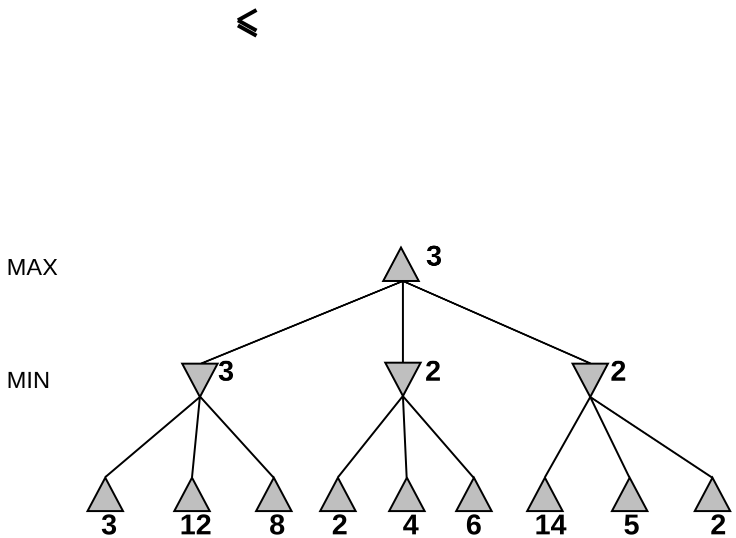
Resource Limits

Suppose we have 100 seconds, explore 10^4 nodes/second
$\Rightarrow$ $\underline{10^6}$ nodes per move

Standard approach:

- *cutoff test*
  e.g., depth limit
- *evaluation function*
  = estimated desirability of position and explore only (hopeful) nodes with certain values

α–β Pruning Example



Cutting Off Search

MINIMAXCUTOFF is identical to MINIMAXVALUE except

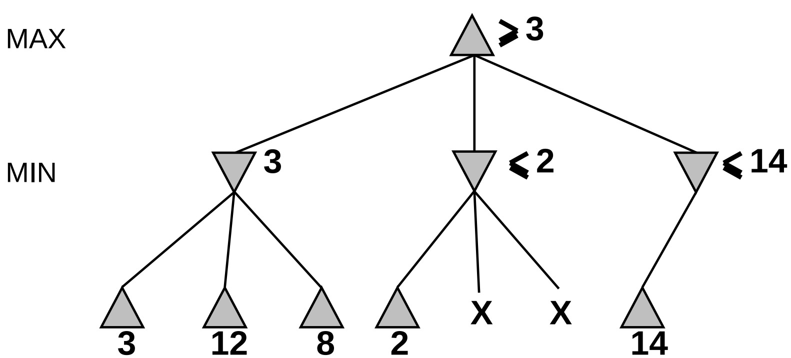
- 1. TERMINAL? is replaced by CUTOFF?
- 2. UTILITY is replaced by EVAL

Does it work in practice?

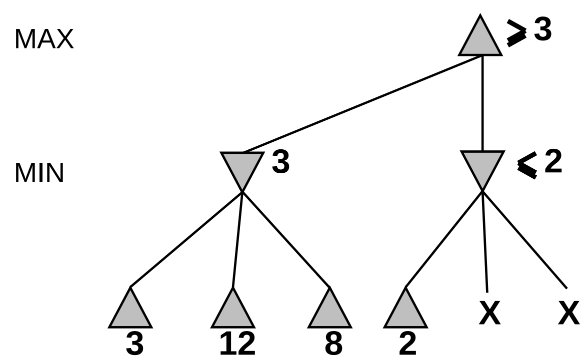
$$b^m = 10^6, \quad b = 35 \quad \Rightarrow \quad m = 4$$

4-ply lookahead is a hopeless chess player!
4-ply $\approx$ human novice
8-ply $\approx$ typical PC, human master
12-ply $\approx$ Deep Blue, Kasparov

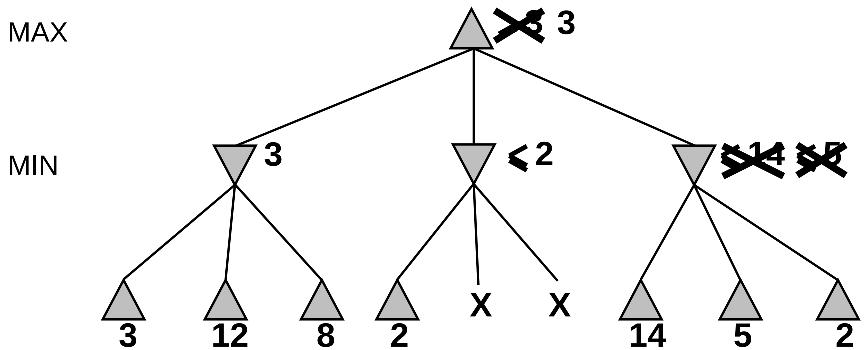
α–β Pruning Example



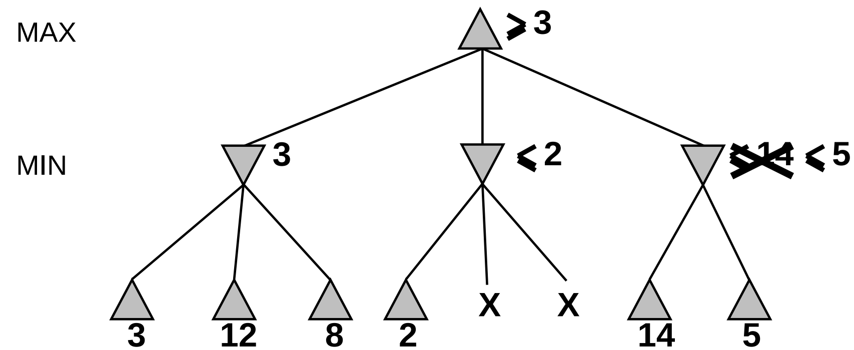
α–β Pruning Example



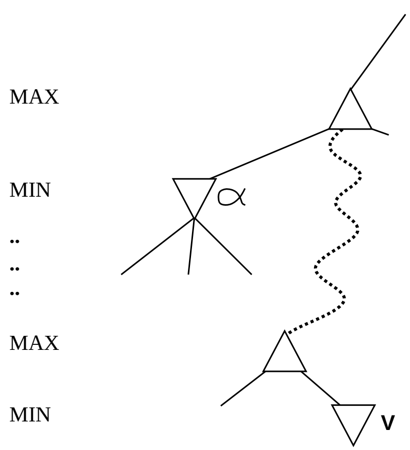
α–β Pruning Example



α–β Pruning Example



Why is it called α–β?



α is the best value (to MAX) found so far off the current path
If V is worse than α, MAX will avoid it $\Rightarrow$ prune that branch
Similarly, β is the best value for MIN.

Properties of α–β

- Pruning *does not* affect final result.
- Good move ordering improves effectiveness of pruning.
- With "perfect ordering," time complexity = $O(b^{m/2})$
  $\Rightarrow$ *doubles* depth of search
  $\Rightarrow$ can easily reach depth 8 and play good chess
- A simple example of the value of reasoning about which computations are relevant (a form of *metareasoning*)

Deterministic Games in Practice

- **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.
- **Chess**: Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.
- **Othello**: human champions refuse to compete against computers, who are too good.
- **Go**: human champions refuse to compete against computers, who are too bad. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves.

The α–β algorithm

function MAX-VALUE(*state*, *game*, α, β) **returns** the minimax value of *state*
  inputs: *state*, current state in game
    game, game description
    α, the best score for MAX along the path to *state*
    β, the best score for MIN along the path to *state*

  if CUTOFF-TEST(*state*) **then return** EVAL(*state*)
  for each *s* **in** SUCCESSORS(*state*) **do**
    $\alpha \leftarrow$ MAX(α, MIN-VALUE(*s*, *game*, α, β))
    if $\alpha \geq \beta$ **then return** β
  end
  return α

function MIN-VALUE(*state*, *game*, α, β) **returns** the minimax value of *state*

  if CUTOFF-TEST(*state*) **then return** EVAL(*state*)
  for each *s* **in** SUCCESSORS(*state*) **do**
    $\beta \leftarrow$ MIN(β, MAX-VALUE(*s*, *game*, α, β))
    if $\beta \leq \alpha$ **then return** α
  end
  return β

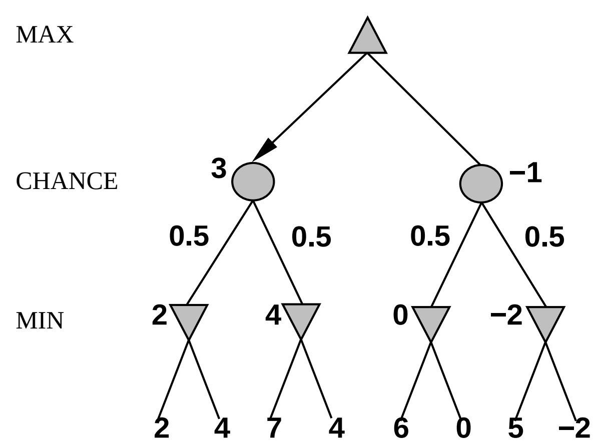
Algorithm for Nondeterministic Games

- EXPECTIMINIMAX gives perfect play
- Just like MINIMAX, except we must also handle chance nodes:

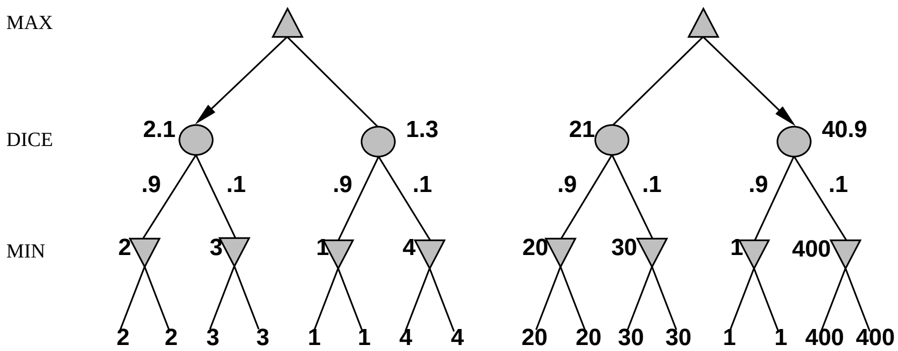
  . . .
  if *state* is a chance node **then**
  return average of EXPECTIMINIMAXVALUE of SUCCESSORS(*state*)
  . . .
- A version of α–β pruning is possible but only if the leaf values are bounded.

Nondeterministic games

In backgammon, the dice rolls determine the legal moves
Simplified example with coin-flipping instead of dice-rolling:



Digression: Exact values DO matter



Behaviour is preserved only by *positive linear* transformation of EVAL
Hence EVAL should be proportional to the expected payoff

Nondeterministic games in practice

- Dice rolls increase b: 21 possible rolls with 2 dice
- Backgammon $\approx$ 20 legal moves (can be 6,000 with 1-1 roll)

$$\text{depth } 4 = 20 \times (21 \times 20)^3 \approx 1.2 \times 10^9$$

- As depth increases, probability of reaching a given node shrinks
  $\Rightarrow$ value of lookahead is diminished
- α–β pruning is much less effective
- TDGAMMON uses depth-2 search + very good EVAL
  $\approx$ world-champion level

Summary

Games are fun to work on! (and dangerous)
They illustrate several important points about AI

- perfection is unattainable $\Rightarrow$ must approximate
- good idea to think about what to think about
- uncertainty constrains the assignment of values to states

Games are to AI as grand prix racing is to automobile design