ADVERSARIAL SEARCH

Chapter 5

"... every game of skill is susceptible of being played by an automaton."
from Charles Babbage, *The Life of a Philosopher,* 1832.

Outline

- ◇ Games
- ◇ Perfect play
  - – minimax decisions
  - – α–β pruning
- ◇ Resource limits and approximate evaluation
- ◇ Games of chance
- ◇ Games of imperfect information

Games vs. search problems

"Unpredictable" opponent $\Rightarrow$ solution is a strategy
specifying a move for every possible opponent reply

Time limits $\Rightarrow$ unlikely to find goal, must approximate

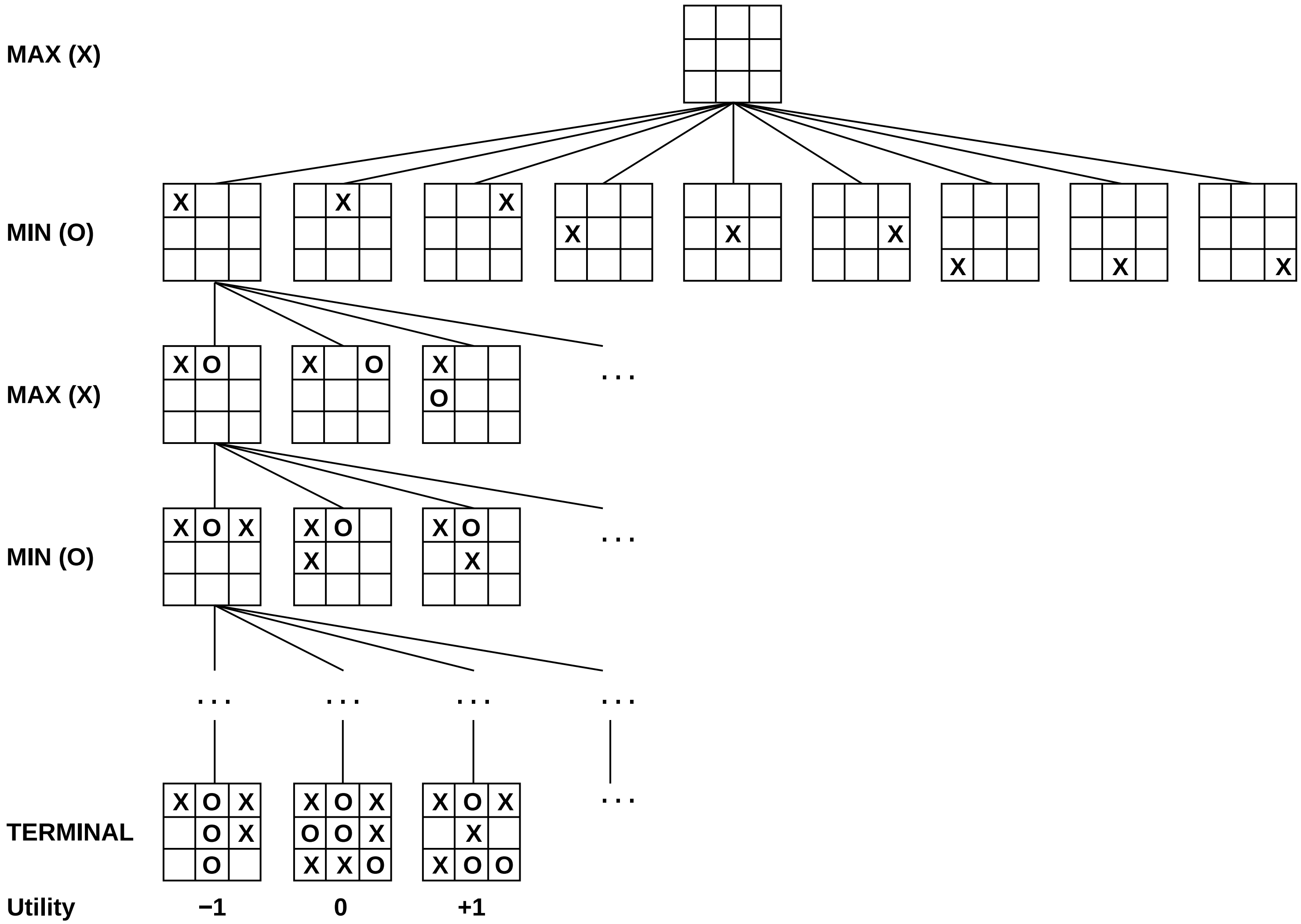
Plan of attack:

- Computer considers possible lines of play (Babbage, 1846)
- Algorithm for perfect play (Zermelo, 1912; Von Neumann, 1944)
- Finite horizon, approximate evaluation (Zuse, 1945; Wiener, 1948; Shannon, 1950)
- First chess program (Turing, 1951)
- Machine learning to improve evaluation accuracy (Samuel, 1952–57)
- Pruning to allow deeper search (McCarthy, 1956)

Types of games

	deterministic	chance
perfect information	chess, checkers, go, othello	backgammon monopoly
imperfect information	battleships, blind tictactoe	bridge, poker, scrabble nuclear war

Game tree (2-player, deterministic, turns)

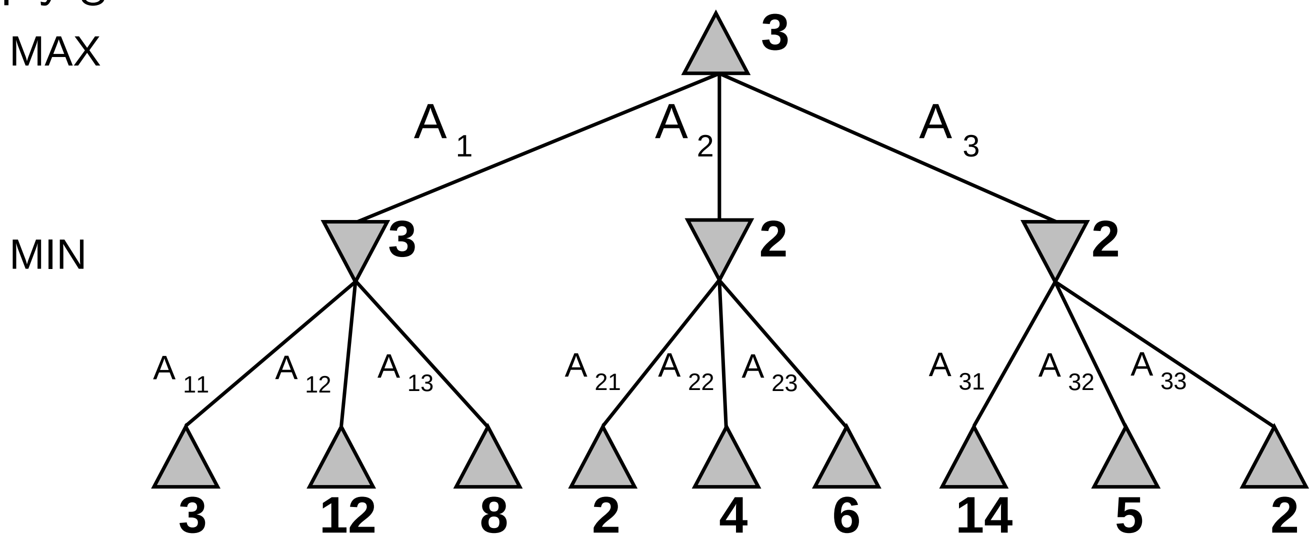


Minimax

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:



Minimax algorithm

function MINIMAX-DECISION(*state*) **returns** *an action*
  inputs: *state*, current state in game

  return the a in ACTIONS(*state*) maximizing MIN-VALUE(RESULT(a, *state*))

---

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

Properties of minimax

Complete??

Properties of minimax

Complete?? Only if tree is finite (chess has specific rules for this).
NB a finite strategy can exist even in an infinite tree!

Optimal??

Properties of minimax

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

Time complexity??

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??

Properties of minimax

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

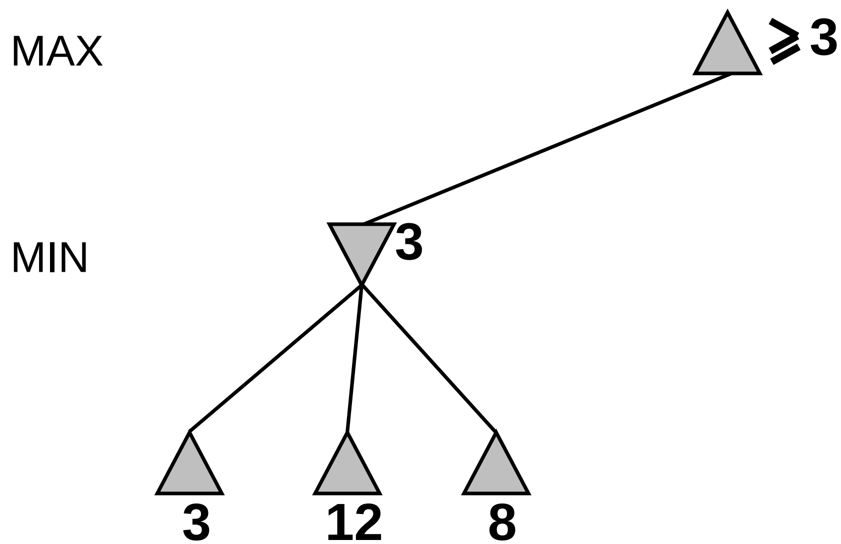
Time complexity?? $O(b^m)$

Space complexity?? $O(bm)$ (depth-first exploration)

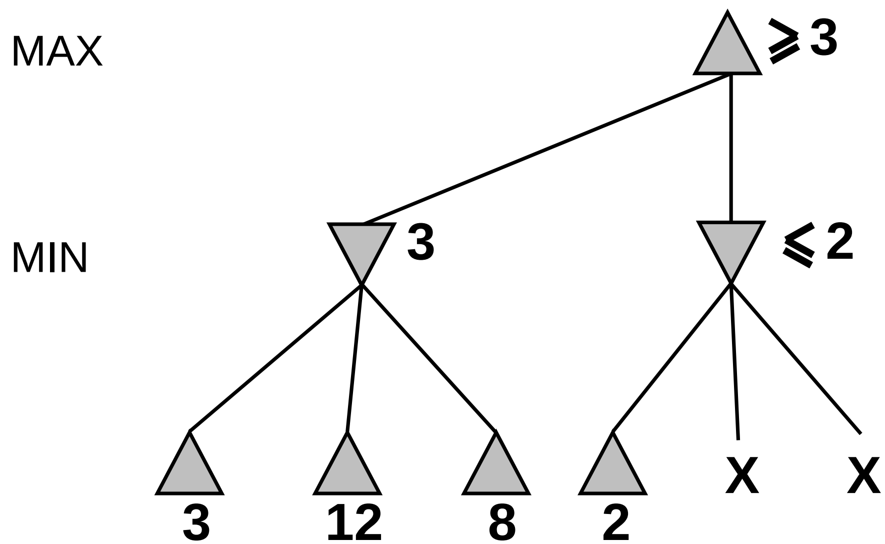
For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
$\Rightarrow$ exact solution completely infeasible

But do we need to explore every path?

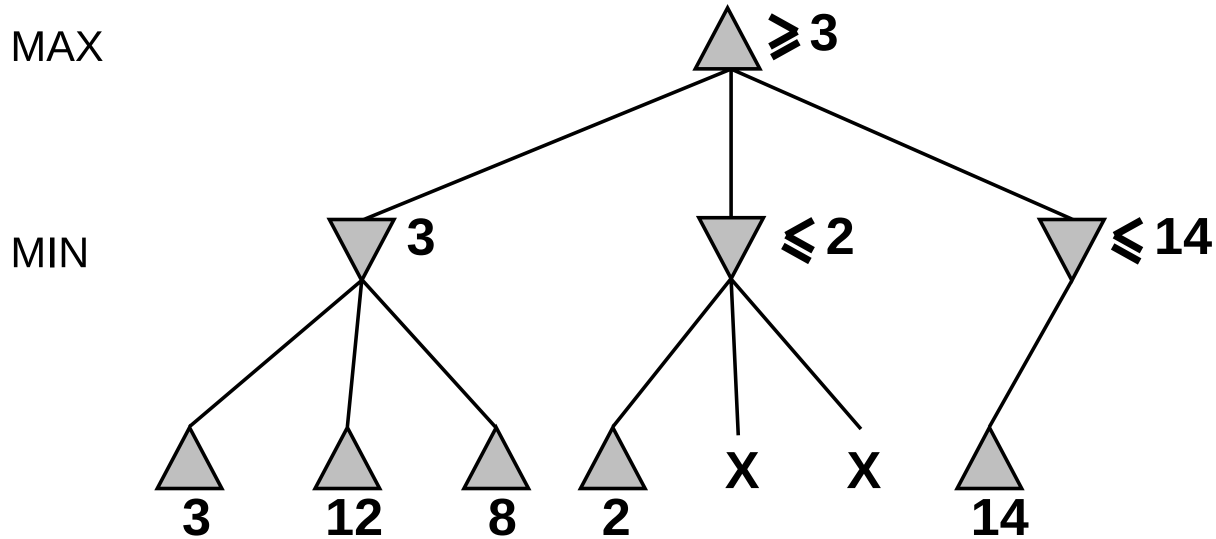
α–β pruning example



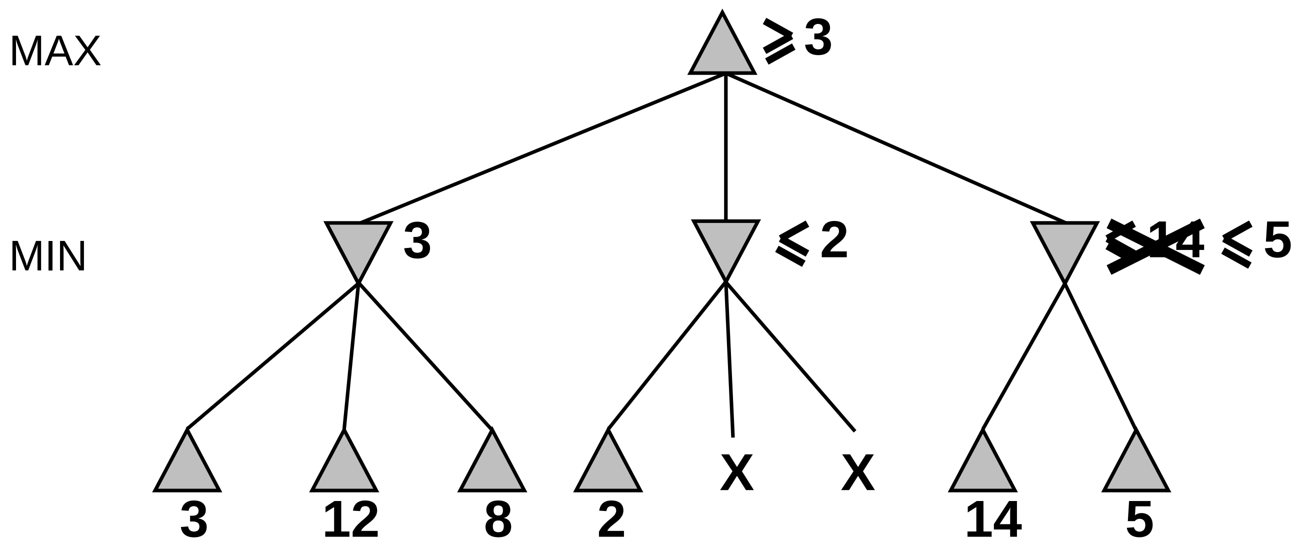
α–β pruning example



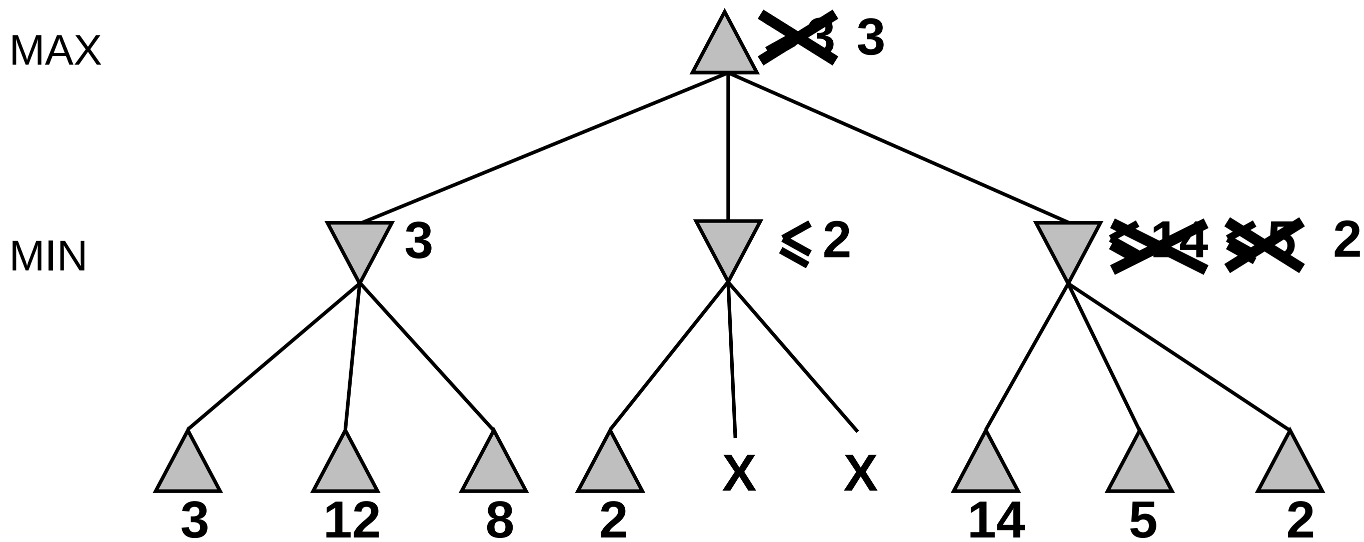
α–β pruning example



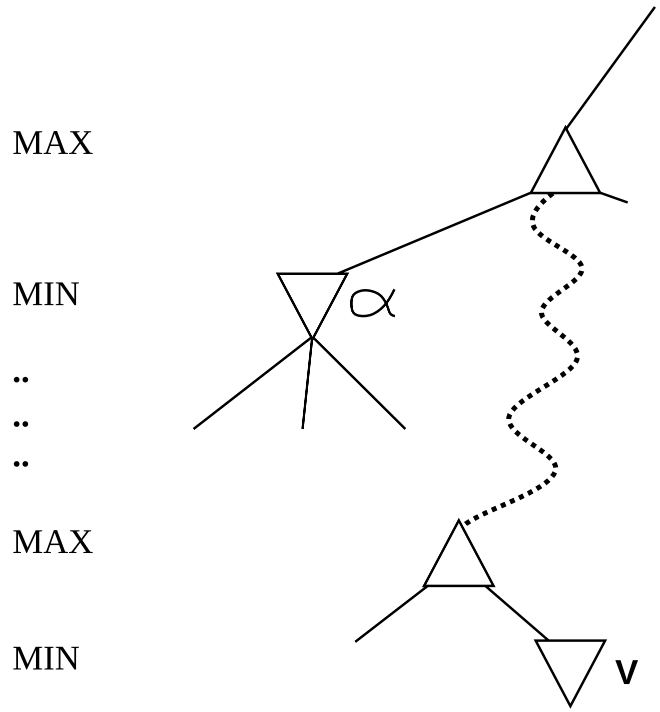
α–β pruning example



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

Define β similarly 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** solvable depth

A simple example of the value of reasoning about which computations are relevant (a form of metareasoning)

Unfortunately, 35^{50} is still impossible!

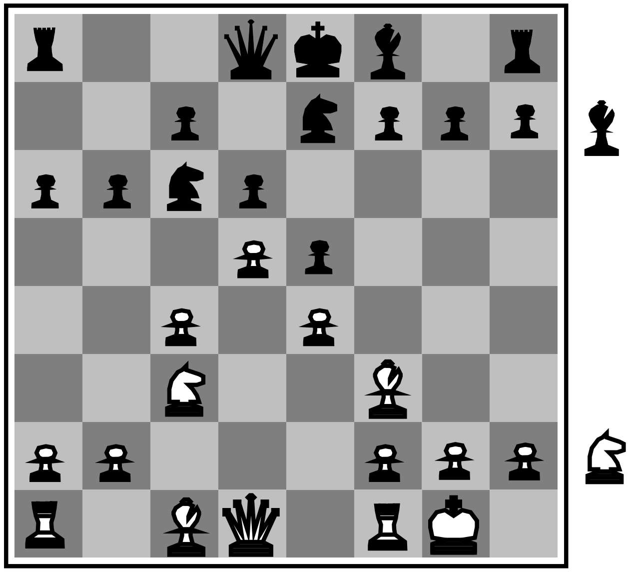
Resource limits

Standard approach:

- Use CUTOFF-TEST instead of TERMINAL-TEST
  e.g., depth limit (perhaps add quiescence search)
- Use EVAL instead of UTILITY
  i.e., evaluation function that estimates desirability of position

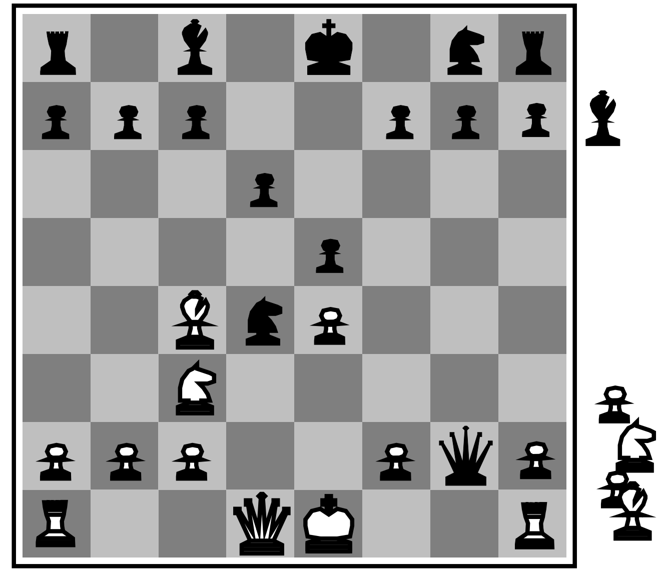
Suppose we have 100 seconds, explore 10^4 nodes/second
$\Rightarrow$ 10^6 nodes per move $\approx 35^{8/2}$
$\Rightarrow$ α–β reaches depth 8 $\Rightarrow$ pretty good chess program

Evaluation functions



Black to move

White slightly better



White to move

Black winning

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.

Exercise – Tic-tac-toe

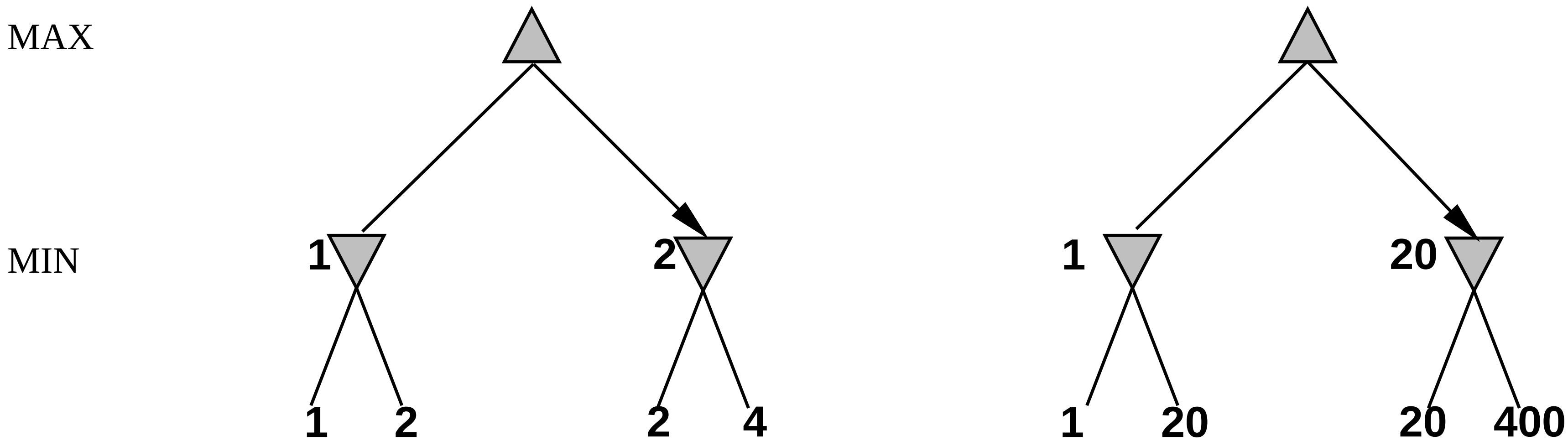
- Define X_n as the number of rows, columns, or diagonals with exactly *n X*'s and no *O*'s. Similarly, O_n is the number of rows, columns, or diagonals with exactly *n O*'s and no *X's*.
- The utility function assigns +1 to any position with $X_3 = 1$ and -1 for any position with $O_3 = 1$. All other terminal positions have utility 0.
- For non-terminal positions, we use a linear evaluation function defined as $\text{Eval}(s) = 3X_2(s) + X_1(s) - (3O_2(s)+O_1(s))$

a) Approximately how many games of tic-tac-toe are there?

Exercise – Tic-tac-toe

b) What does the game tree look like (taking symmetry into account)?

Digression: Exact values don't matter



Behaviour is preserved under any **monotonic** transformation of EVAL

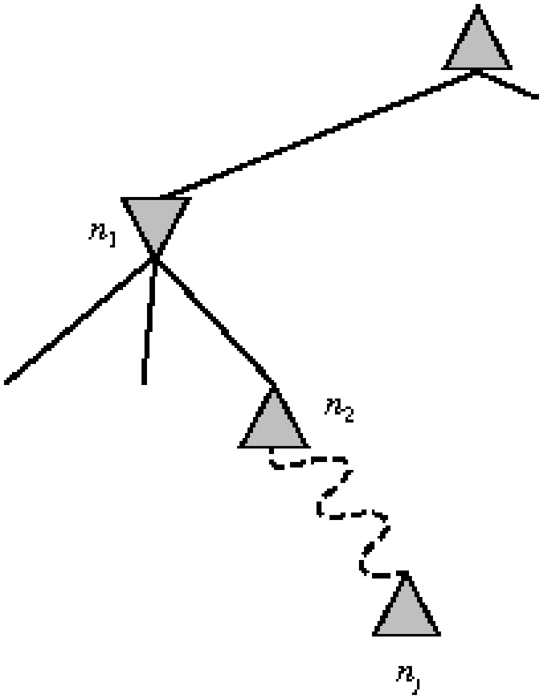
Only the order matters:

payoff in deterministic games acts as an ordinal utility function

How to achieve a good game of chess?

- Extensively tuned evaluation function
- Cutoff test with quiescence search
- Large transposition table [i.e., hash of previously seen positions, saved for re-use]
- Use of alpha-beta, with extra pruning
- Large database of optimal opening and endgame moves
- Fast computer!

Exercise – Prove correctness of α-β



- Question is whether to prune n_j, which is a max-node and descendent of n_1
- Basic idea is to prune it iff the minimax value of n_1 can be shown to be independent of the value of n_j
- Node n_1 takes on the minimum value among its children $n_1 = \min(n_2, n_{21}, \ldots, n_{2_{b2}})$. Find a similar expression for n_2 and hence an expression for n_1 in terms of n_j.