Minimax

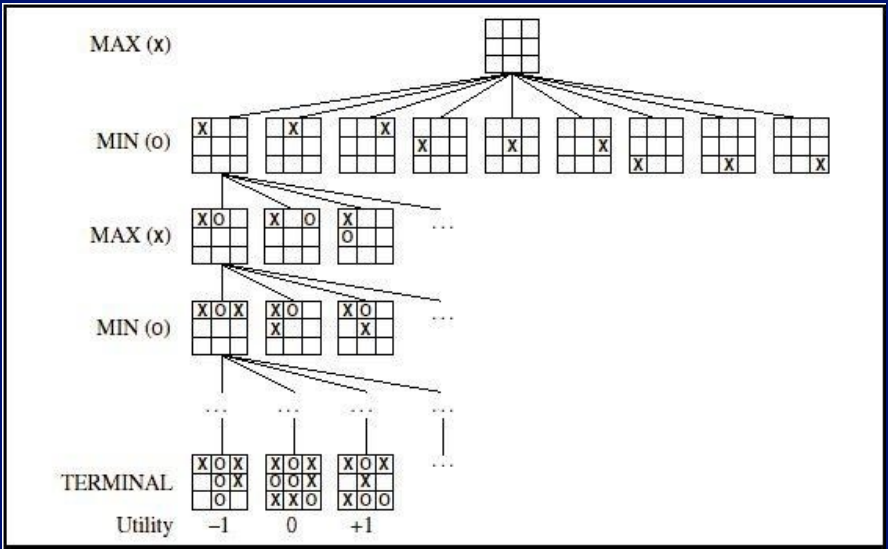


Figure 6.1 A (partial) search tree for the game of tic-tac-toe. The top node is the initial state, and MAX moves first, placing an X in an empty square. We show part of the search tree, giving alternating moves by MIN (O) and MAX, until we eventually reach terminal states, which can be assigned utilities according to the rules of the game.

What data do we need to play?

Initial State

- How does the game start?

Successor Function

- A list of legal (move, state) pairs for each state

Terminal Test

- Determines when game is over

Utility Function

- Provides numeric value for all terminal states

Minimax Strategy

Optimal Stragtegy

- Leads to outcomes at least as good as any other strategy when playing an infallible opponent
- Pick the option that most (max) minimizes the damage your opponent can do
  - maximize the worst-case outcome
  - because your skillful opponent will certainly find the most damaging move

Minimax

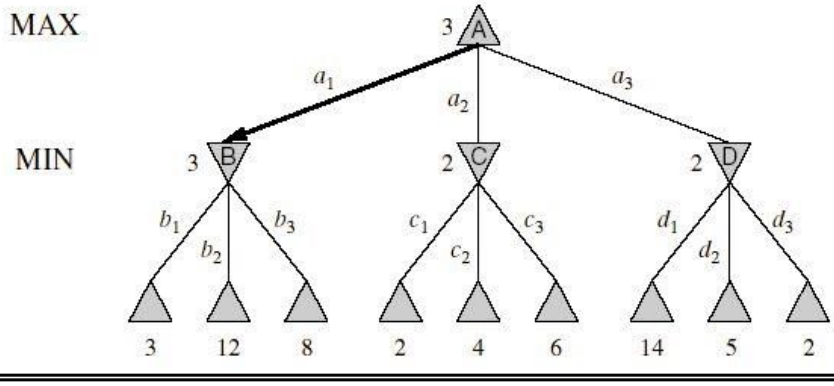


Figure 6.2 A two-ply game tree. The △ nodes are "MAX nodes," in which it is MAX's turn to move, and the ▽ nodes are "MIN nodes." The terminal nodes show the utility values for MAX; the other nodes are labeled with their minimax values. MAX's best move at the root is a_1, because it leads to the successor with the highest minimax value, and MIN's best reply is b_1, because it leads to the successor with the lowest minimax value.

Minimax Algorithm

We wish to identify minimax decision at the root

- Recursive evaluation of all nodes in game tree

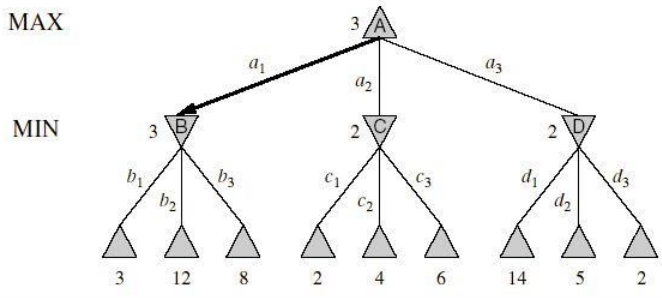


Figure 6.2 A two-ply game tree. The △ nodes are "MAX nodes," in which it is MAX's turn to move, and the ▽ nodes are "MIN nodes." The terminal nodes show the utility values for MAX; the other nodes are labeled with their minimax values. MAX's best move at the root is a_1, because it leads to the successor with the highest minimax value, and MIN's best reply is b_1, because it leads to the successor with the lowest minimax value.

- Time complexity = $O(b^m)$

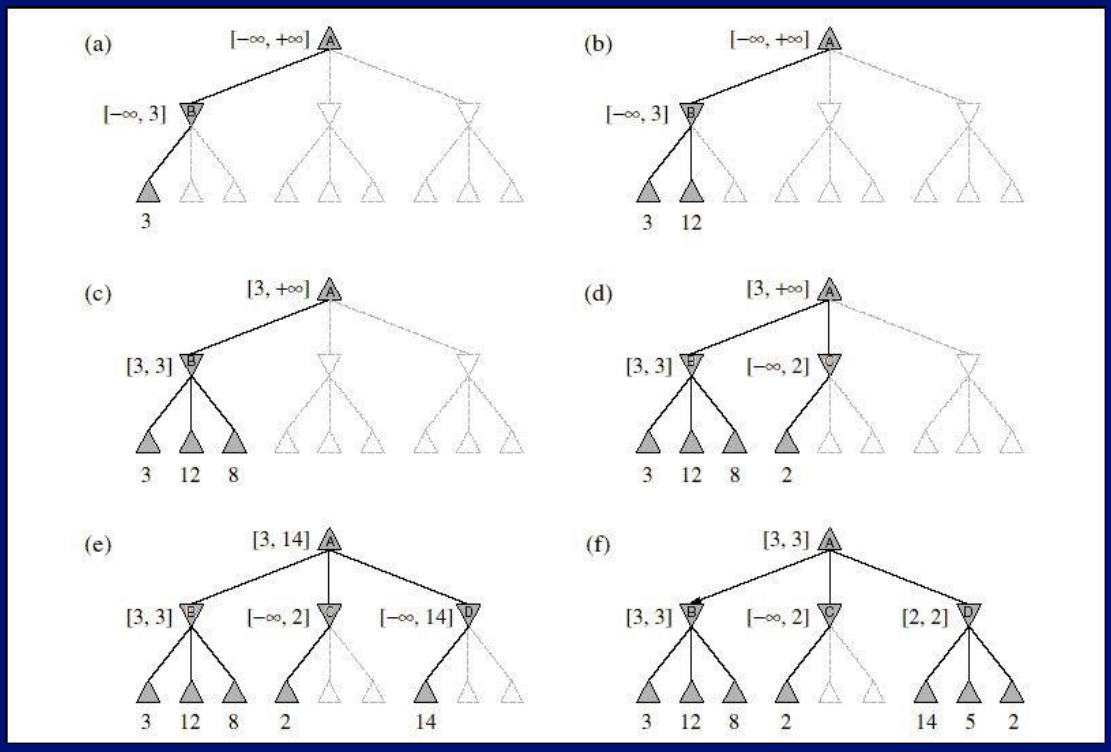
Feasibility of minimax?

How about a nice game of chess?

- Avg branching = 35 and avg # moves = 50 for each player
  - $O(35^{100})$ time complexity = 10^{154} nodes
    - 10^{40} distinct nodes

Minimax is impractical if directly applied to chess

Alpha-beta pruning



Alpha-beta pruning

- α
  - the value of the best (highest) choice so far in search of MAX
- β
  - the value of the best (lowest) choice so far in search of MIN
- Order of considering successors matters (look at step f in previous slide)
  - If possible, consider best successors first

Realtime decisions

What if you don't have enough time to explore entire search tree?

- We cannot search all the way down to terminal state for all decision sequences
- Use a heuristic to approximate (guess) eventual terminal state

Evaluation Function

The heuristic that estimates expected utility

- Cannot take too long (otherwise recurse to get answer)
- It should preserve the ordering among terminal states
  - otherwise it can cause bad decision making
- Define features of game state that assist in evaluation
  - what are features of chess?

Mini Max Example