Properties of minimax

• Complete?
  – Yes (assuming tree is finite)

• Optimal?
  – Yes (assuming opponent is also optimal)

• Time complexity: $O(b^m)$

• Space complexity: $O(bm)$ (like DFS)

• But for chess, $b \approx 35$, $m \approx 100$, so this time is completely infeasible!
• Problem: minimax takes too long.
• Solution: improve algorithm to ignore parts of the tree that will definitely not be used (assuming both players play optimally).

• New algorithm: \textit{minimax with alpha-beta pruning}.
• Idea: for each node, keep track of the range of possible values that minimax could produce for that node.
Alpha-beta pruning

• Each node in the game tree needs two extra variables, called alpha and beta.
• Alpha and beta are inherited from parent nodes.
• If at a max node, we see a sub-node that has a value bigger than beta, short-circuit.
• If at a min node, we see a sub-node that has a value smaller than alpha, short-circuit.
Alpha-beta code

• For programming, use code in the book.
• For offline use, use this idea:

alpha-beta(node):
  inherit alpha & beta from parents
  let v be each child value in turn:
  
  If at a MAX node:
    if v >= beta, then short-circuit and return v
    else if v > alpha, then alpha = v (and continue)

  If at a MIN node:
    if v <= alpha, then short-circuit and return v
    else if v < beta, then beta = v (and continue)

  if MAX, return alpha; if MIN, return beta (to parent)
• The results of alpha-beta depend on the order in which moves are considered among the children of a node.

• If possible, consider better moves first!