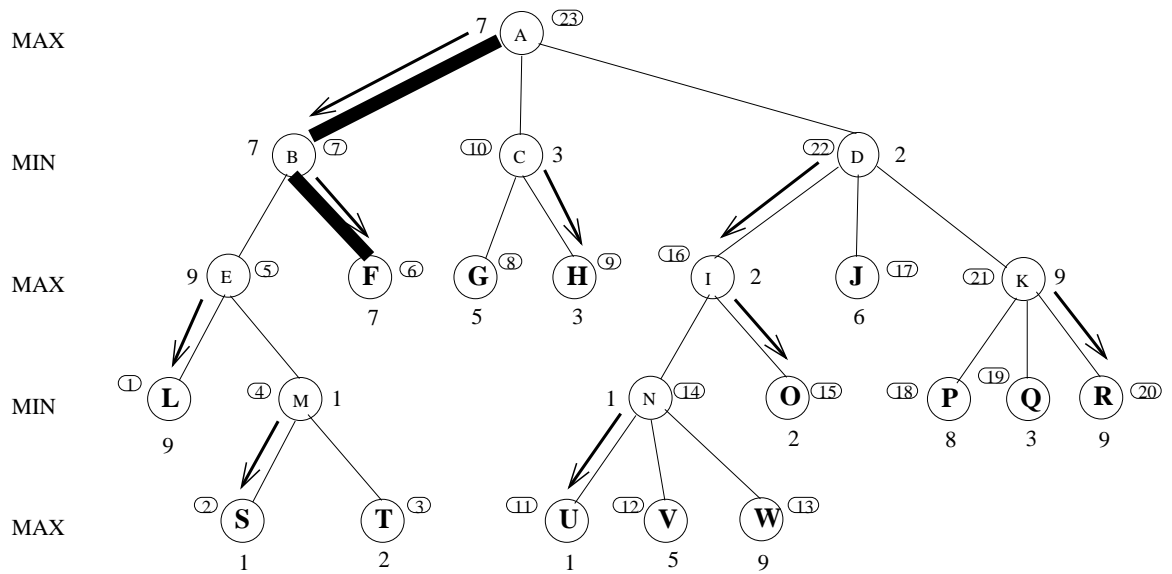


Solutions to Practice Problems on Search in Games

Original date: October 5, 2005; Last revised: November 5, 2006

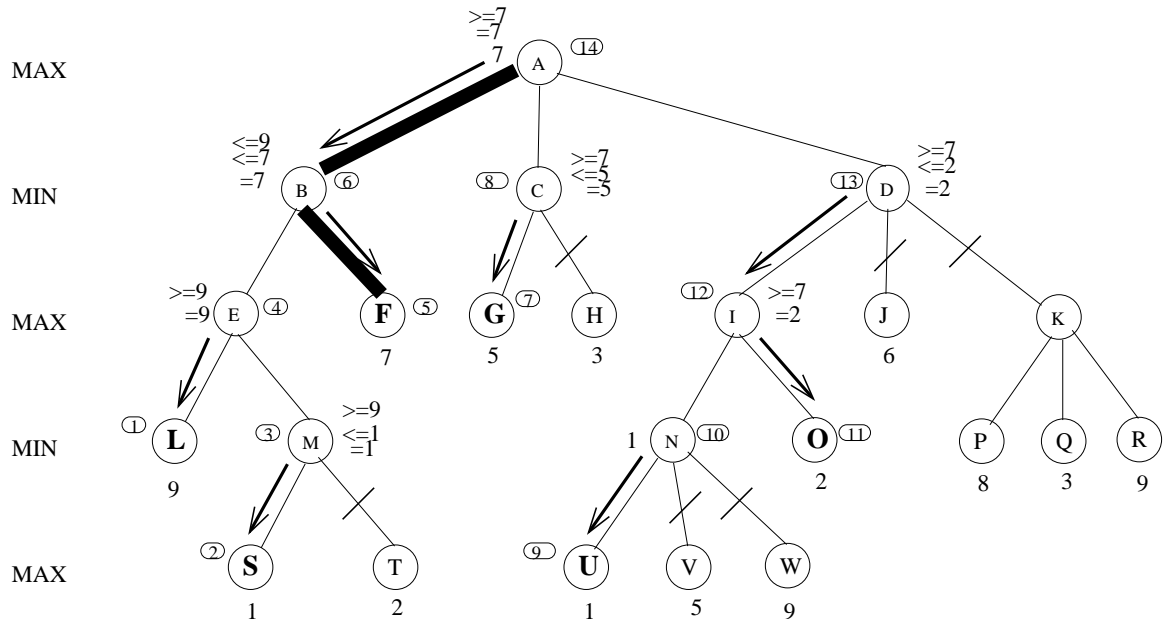
- The following description applies to the figures illustrating the solution to each part of this problem. *Statically* evaluated nodes are indicated by *enlarged and bold labels*. (Recall that only leaves of the depth-cutoff game tree can be statically evaluated.) The order number in which the respective procedure *assigned a value* to each node is given by the number encircled next to each node. (This order is different from the order in which the nodes are visited, i.e., the procedure is *applied* to the node; or in other words, the order given is not the order in which the nodes were *evaluated*.) The best move at *each* node is indicated by an *arrow* pointed at the best-move child. The best *sequence* of moves is indicated by the *edges* with the largest width. The best move for the maximizer at A is to move to B.

(a) The MINIMAX values are given next to each node.

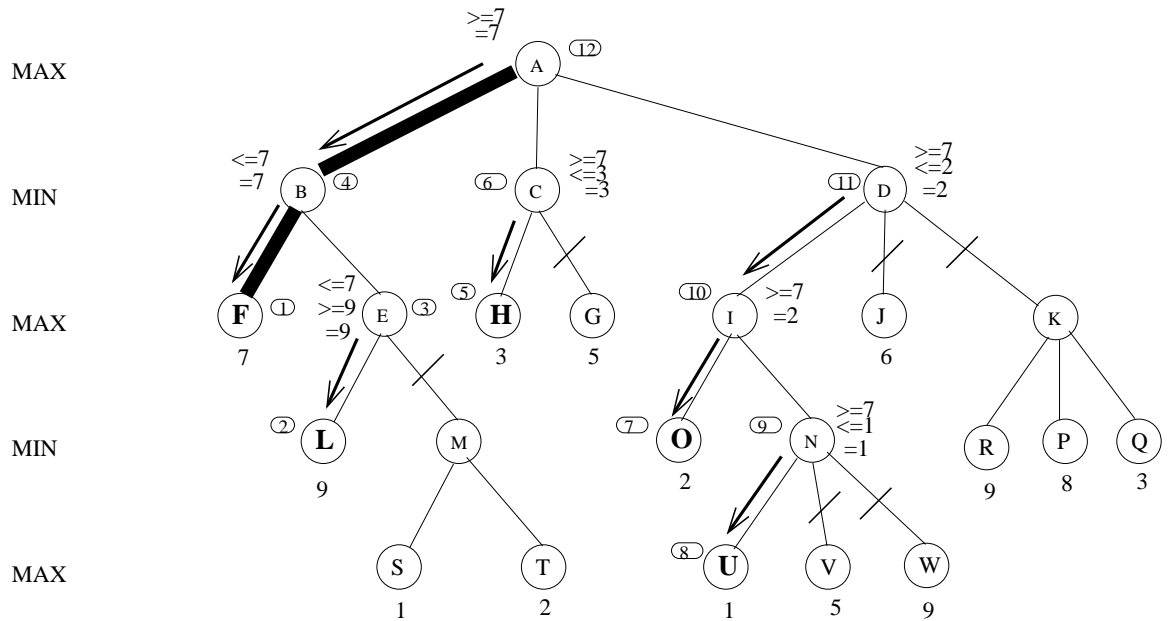


- (b) The following description applies to all the figures below. The α and β values (only when finite), as well as the final node value computed by ALPHA-BETA are given next to each node. Lines across edges in the game tree figure indicate branches that are not evaluated. The arcs pointing to the children of a node indicate the best move found *prior* to ALPHA-BETA returning a value for the node.

In this case, ALPHA-BETA performs a total of 14 evaluations, of which 6 are static evaluations.

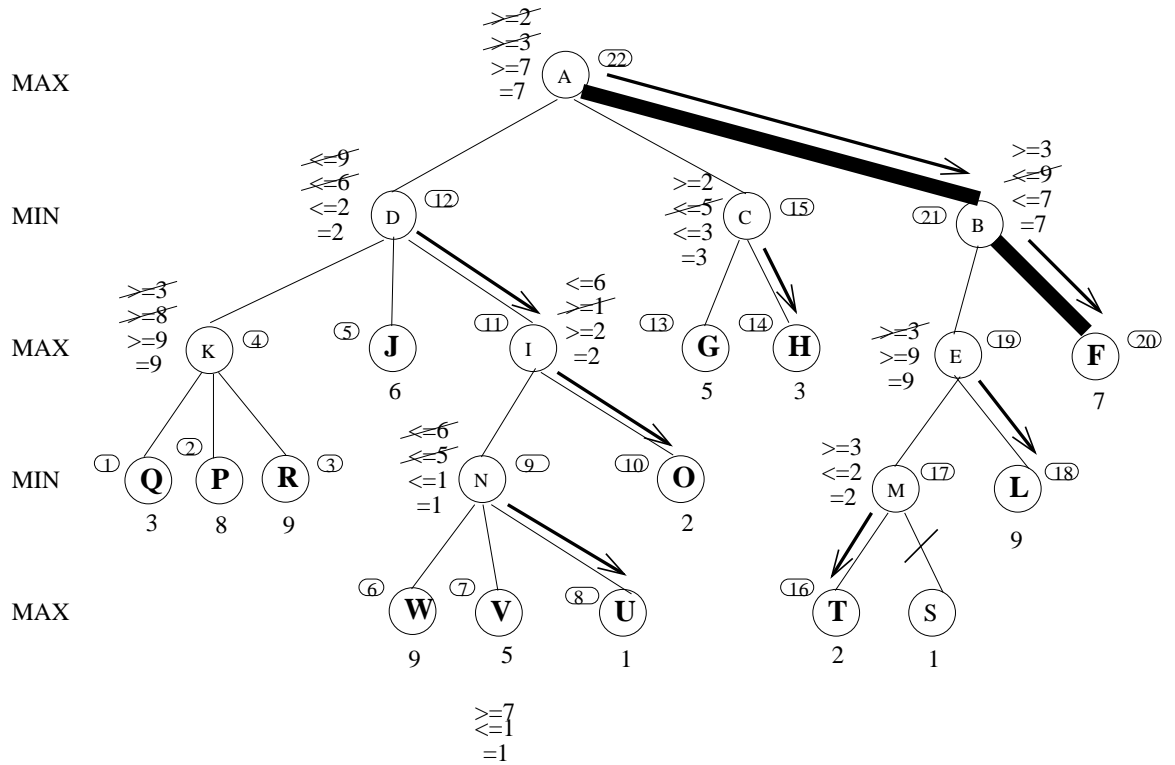


- (c) The following game tree shows the evaluations that ALPHA-BETA performs in the best case. In this case, ALPHA-BETA performs a total of 12 evaluations, of which 5 are static evaluations.



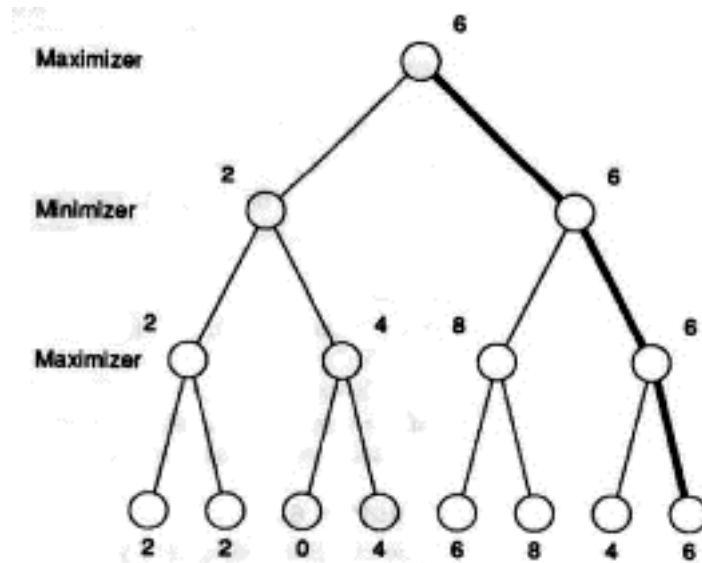
- (d) The following game tree shows the evaluations that ALPHA-BETA performs in the worst case. In this case, ALPHA-BETA performs a total of 22 evaluations, of which 13 are static evaluations.

Solutions to Practice Problems on Search in Games

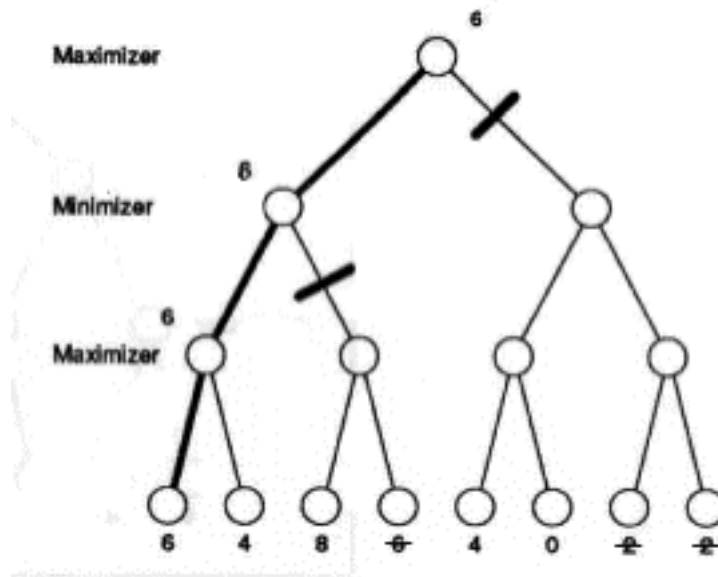


2. (Solution to Exercises 6.3 and 6.4 from the book) ¹

(a)



(b)



(c) If the evaluated nodes are ordered in the manner described below, then you get maximal Alpha-Beta cutoff; the opposite order gets no Alpha-Beta cutoff. For the game trees we've been looking at (*i.e.*, with the bottom row containing the evaluated nodes, and

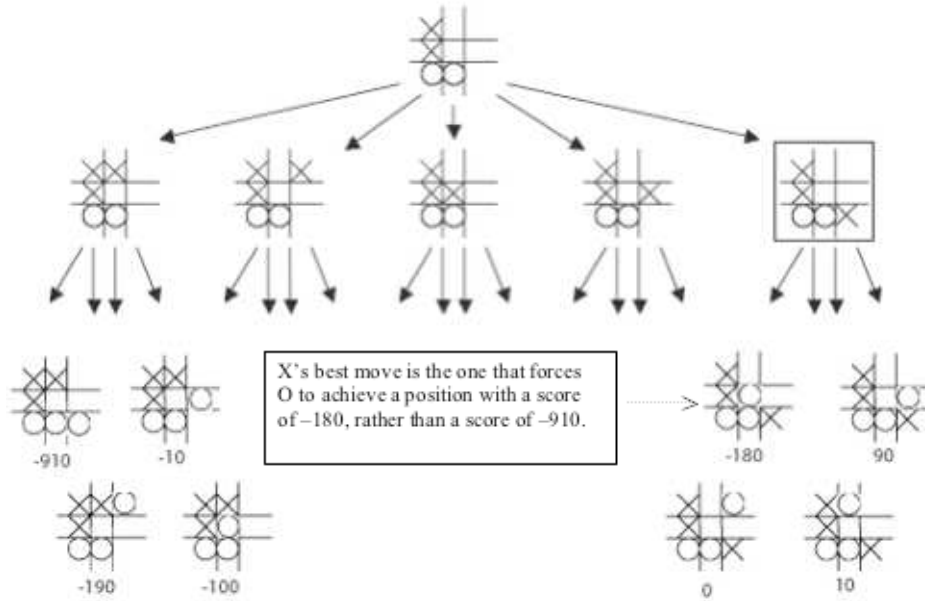
¹The first two parts of this search in games problem solution are from the book *Artificial Intelligence* by Patrick Winston. The last part is due to Kimberle Koile.

successively higher layers of nodes alternating between minimizer and maximizer, or maximizer and minimizer):

If penultimate level of the tree is maximizer level, you get maximal cutoff if descendents of each node in that level are ordered from left to right, max value to min value. If penultimate level of tree is minimizer level, you get maximal cutoff if descendents of each node in that level are ordered from left to right, min value to max value.

Note that since the goal of Alpha-Beta pruning is to cut down on the number of nodes that have to be evaluated, game programs don't actually sort nodes. Two observations that can be made: (1) the performance of Alpha-Beta is variable and can't be counted on to always outperform Minimax; (2) a move generator could attempt to produce new configurations in a sorted order.

3. Tic-Tac-Toe Solution ²



²This search in games problem solution is due to Kimberle Koile.

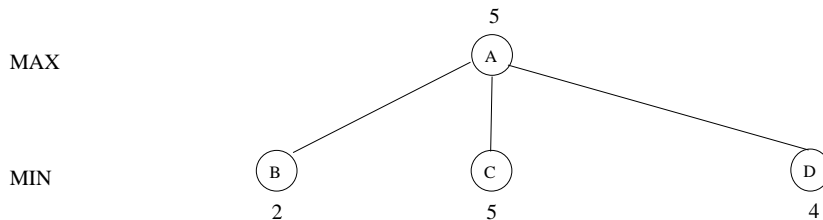
4.

Depth	Best Move	Order Alpha-Beta Applied	Num. Evaluations	
			static	nodes
1	C	A B C D	3	4
2	C	A C G H D I B E	4	8
3	B	A C H G D I N O J K P B E L M F	9	16
4	B	A B F E L D I N U O C H	5	12

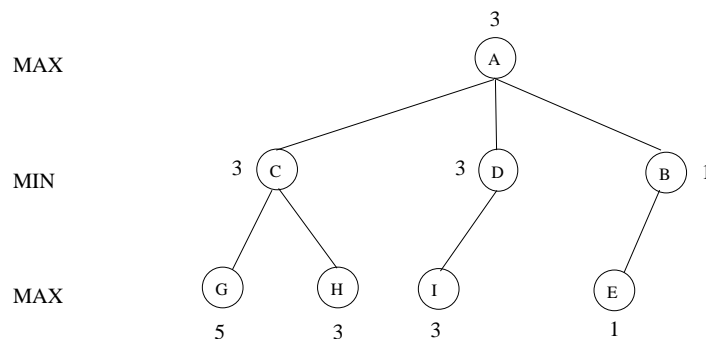
Note how Alpha-Beta did less work at depth 4 than at depth 3, by exploiting information from evaluating up to depth 3 about what the best evaluation order for Alpha-Beta might be (*i.e.*, the node evaluations obtained from Alpha-Beta at depth 3). Using this information does not guarantee an improvement in the performance of Alpha-Beta, but it is very useful as a heuristic.

Below are the trees corresponding to the application of Alpha-Beta at each depth level. The number around a node corresponds to the output of Alpha-Beta after evaluating that node (given also the corresponding values of α and β when Alpha-Beta was called for that node). Only those nodes/states where Alpha-Beta was applied (or, in other words, *visited* by Alpha-Beta) are drawn. The left-to-right order corresponds to the actual order in which Alpha-Beta was applied to the children/successors of the node. Note that, whenever possible, we are using the output evaluations from running Alpha-Beta at the previous depth to order the node evaluations when performing Alpha-Beta up to a particular depth.

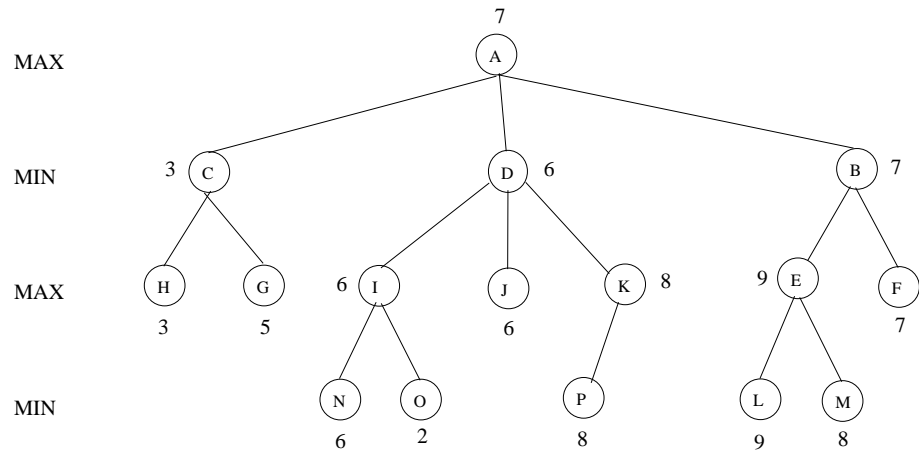
For $d = 1$:



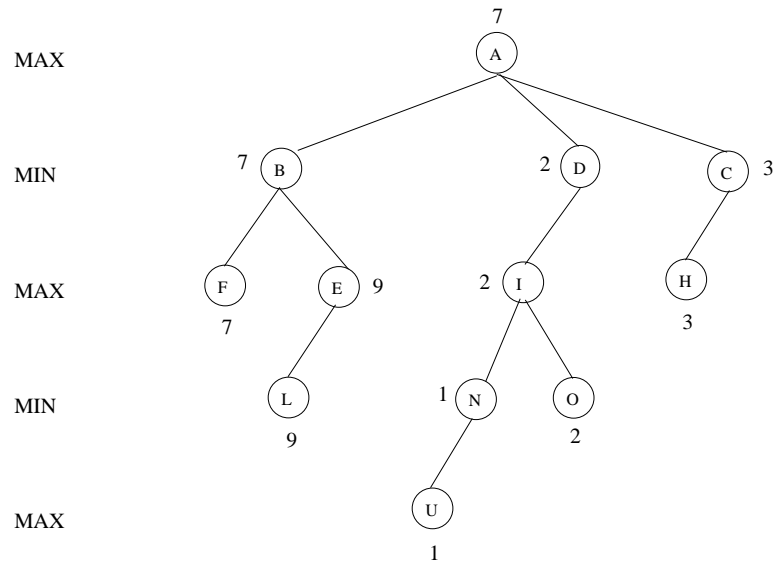
For $d = 2$:



For $d = 3$:



For $d = 4$:



Note that, for each depth, the order in which Alpha-Beta was applied to the nodes is exactly the order in which depth-first search would extend the nodes in the corresponding tree above.