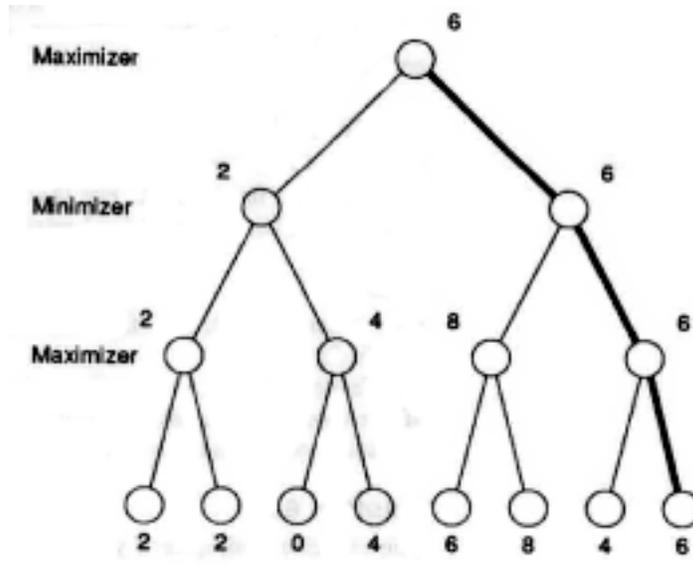
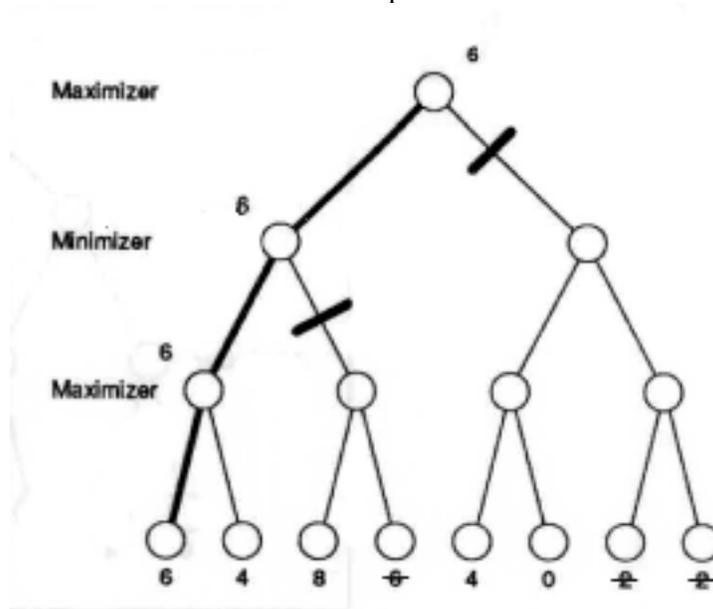


### 6.034 Recitation 3: Game Search Exercise Solutions (10/01/04) LOrtiz (Orig. by KKoile)

1. Consider the game tree shown below. Explore the tree using the alpha-beta procedure. Indicate all parts of the tree that are cut off, and indicate the winning path or paths. Strike out all static evaluation values that do not need to be computed.



2. Now consider the tree shown below, which is a mirror image of the tree shown above. Explore the tree using the alpha-beta procedure. Indicate all parts of the tree that are cut off. Indicate the winning path or paths. Strike out all static evaluation values that do not need to be computed.



3. Compare the amount of cutoff in the above two trees. What do you notice about how the order of static evaluation nodes affects the amount of alpha-beta cutoff?

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

If penultimate level of 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.

#### 4. Tic-Tac-Toe

You are the X player, looking at the board shown below, with five possible moves. You want to look ahead to find your best move and decide to use the following evaluation function for rating board configurations:

```

value V = 0
do over all rows, columns, diagonals R:
  if R contains three Xs, V = V + 1000
  else if R contains three Os, V = V - 1000
  else when R contains only two Xs, V = V + 100
  else when R contains only one X, V = V + 10
  else when R contains only two Os, V = V - 100
  else when R contains only one O, V = V - 10
end do
return V
    
```

Draw the four configurations possible from the leftmost and rightmost board configurations below. Use the above static evaluation function to rate the 8 board configurations and choose X's best move. (A reminder: The board configurations that you draw will show possibilities for O's next move.)

