Book solutions

Q1:

Question 1 Minimax

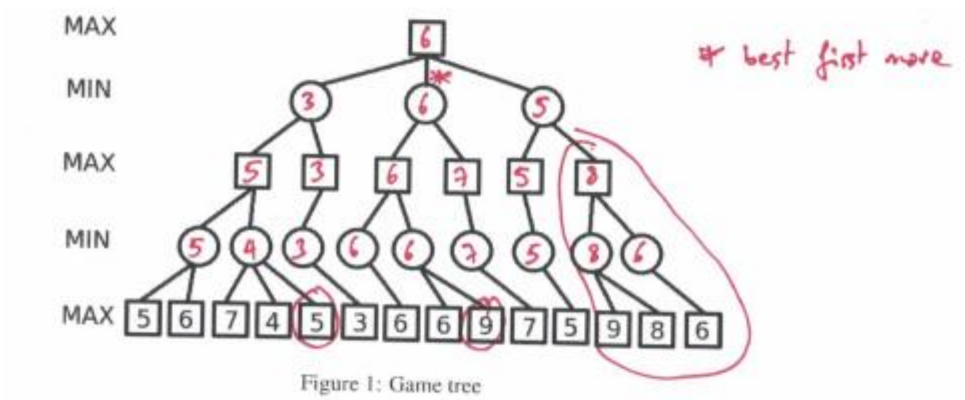

Figure 1: Game tree

No need to search the circled nodes. For the first one (5), we know that using the depth first evaluation, MAX has the option of 5 from the left subtree. As soon as (4) is encountered, we know that MIN will choose this, meaning it is impossible to do any better than the alpha value (5) already encountered irrespectove of the value in this node. So no need to search it.

Likewise for the (6) alpha value found in the middle subtree. No need to search the (9) node because MIN will always choose 6, so MAX can't do any better than they already have.

On the right-hand side note that MAX knows MIN will choose 5 from left subtree. This is already worse than the 6 found from the middle subtree. So there is no need to search any of the nodes on the far right-hand side.

Q2:

6.1 Figure S6.1 shows the game tree, with the evaluation function values below the terminal nodes and the backed-up values to the right of the non-terminal nodes. The values imply that the best starting move for X is to take the center. The terminal nodes with a bold outline are the ones that do not need to be evaluated, assuming the optimal ordering.

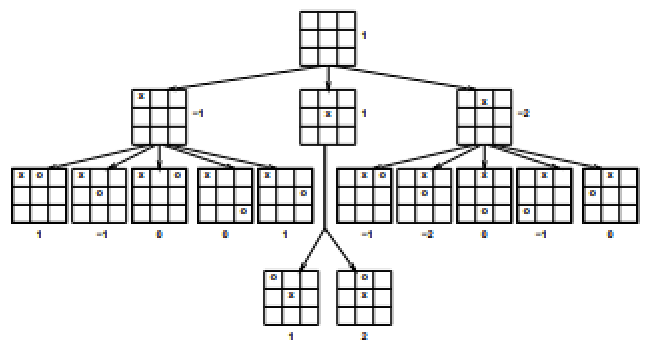

Figure S6.1 Part of the game tree for tic-tac-toe, for Exercise 6.1.

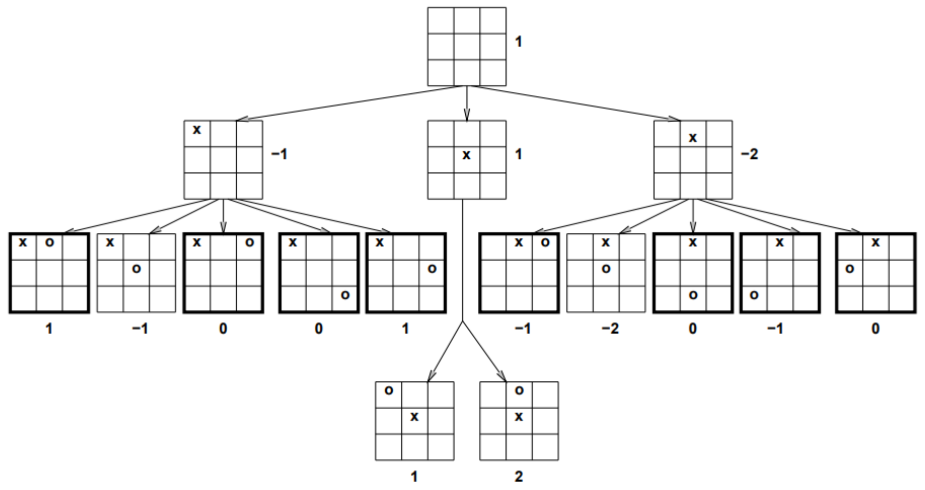

Figure S6.1 Part of the game tree for tic-tac-toe, for Exercise 6.1.

Q3

6.3

a. (5) The game tree, complete with annotations of all minimax values, is shown in Figure S6.3.

b. (5) The "?" values are handled by assuming that an agent with a choice between winning the game and entering a "?" state will always choose the win. That is, min(–1,?) is –1 and max(+1,?) is +1. If all successors are "?", the backed-up value is "?".

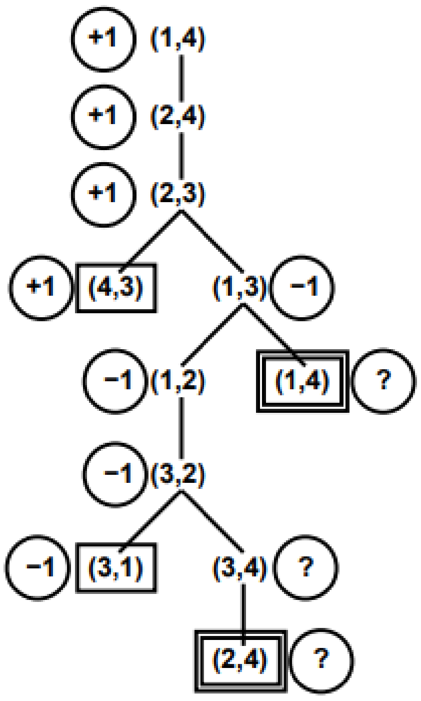

Figure S6.3 The game tree for the four-square game in Exercise 6.3. Terminal states are in single boxes, loop states in double boxes. Each state is annotated with its minimax value in a circle.

c. (5) Standard minimax is depth-first and would go into an infinite loop. It can be fixed by comparing the current state against the stack; and if the state is repeated, then return a "?" value. Propagation of "?" values is handled as above. Although it works in this case, it does not *always* work because it is not clear how to compare "?" with a drawn position; nor is it clear how to handle the comparison when there are wins of different degrees (as in backgammon). Finally, in games with chance nodes, it is unclear how to compute the average of a number and a "?". Note that it is *not* correct to treat repeated states automatically as drawn positions; in this example, both (1,4) and (2,4) repeat in the tree but they are won positions.

What is really happening is that each state has a well-defined but initially unknown value. These unknown values are related by the minimax equation at the bottom of page 163. If the game tree is acyclic, then the minimax algorithm solves these equations by propagating from the leaves. If the game tree has cycles, then a dynamic programming method must be used, as explained in Chapter 17. (Exercise 17.8 studies this problem in particular.) These algorithms can determine whether each node has a well-determined value (as in this example) or is really an infinite loop in that both players prefer to stay in the loop (or have no choice). In such a case, the rules of the game will need to define the value (otherwise the game will never end). In chess, for eaxmple, a state that occurs 3 times (and hence is assumed to be desirable for both players) is a draw.

d. This question is a little tricky. One approach is a proof by induction on the size of the game. Clearly, the base case $n = 3$ is a loss for A and the base case $n = 4$ is a win for A. For any $n > 4$, the initial moves are the same: A and B both move one step towards each other. Now, we can see that they are engaged in a subgame of size $n - 2$ on the squares $[2, \ldots, n-1]$, *except* that there is an extra choice of moves on squares 2 and $n - 1$. Ignoring this for a moment, it is clear that if the "$n - 2$" is won for A, then A gets to the square $n - 1$ before B gets to square 2 (by the definition of winning) and therefore gets to n before B gets to 1, hence the "n" game is won for A. By the same line of reasoning, if "$n - 2$" is won for B then "n" is won for B. Now, the presence of the extra moves complicates the issue, but not too much. First, the player who is slated to win the subgame $[2, \ldots, n-1]$ never moves back to his home square. If the player slated to lose the subgame does so, then it is easy to show that he is bound to lose the game itself—the other player simply moves forward and a subgame of size $n - 2k$ is played one step closer to the loser's home square.