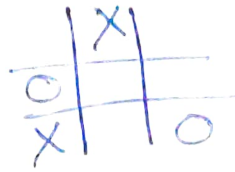
We are given the board

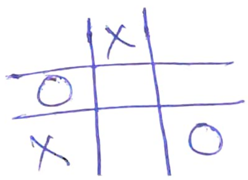


and a scoring function.

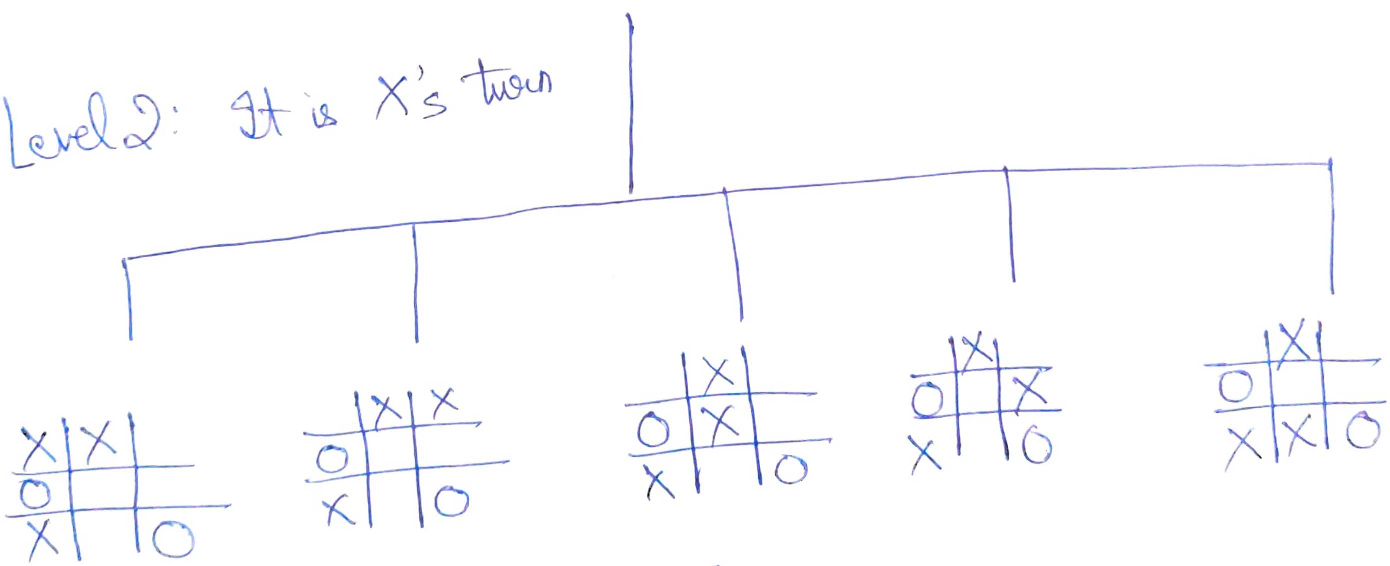
$$8X_3(s) + 3X_2(s) + X_1(s)$$
$$- \left[8O_3(s) + 3O_2(s) + O_1(s)\right]$$

I will first draw the game tree upto 3 levels with 20 leaf nodes and compute the scores at the leafs for the board. With the scores in our hand, we can make the game tree of scores and compute the optimal path to reach our answer.

Level 1:



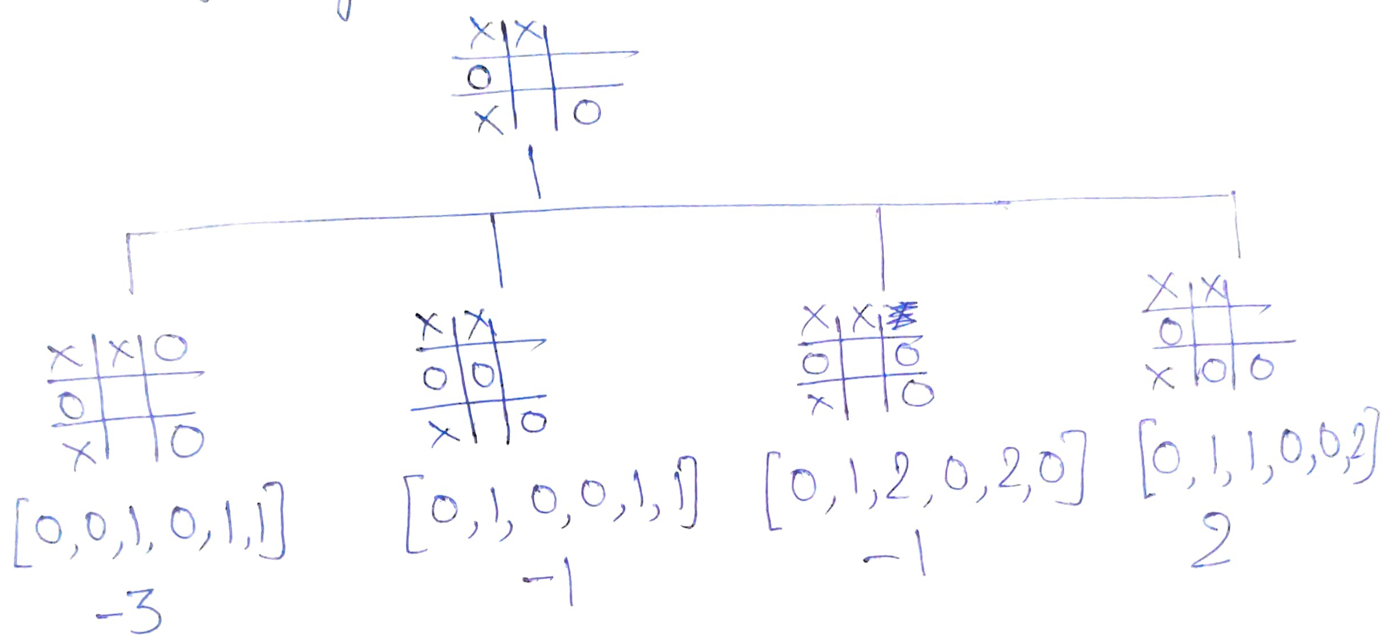
Level 2: It is X's turn

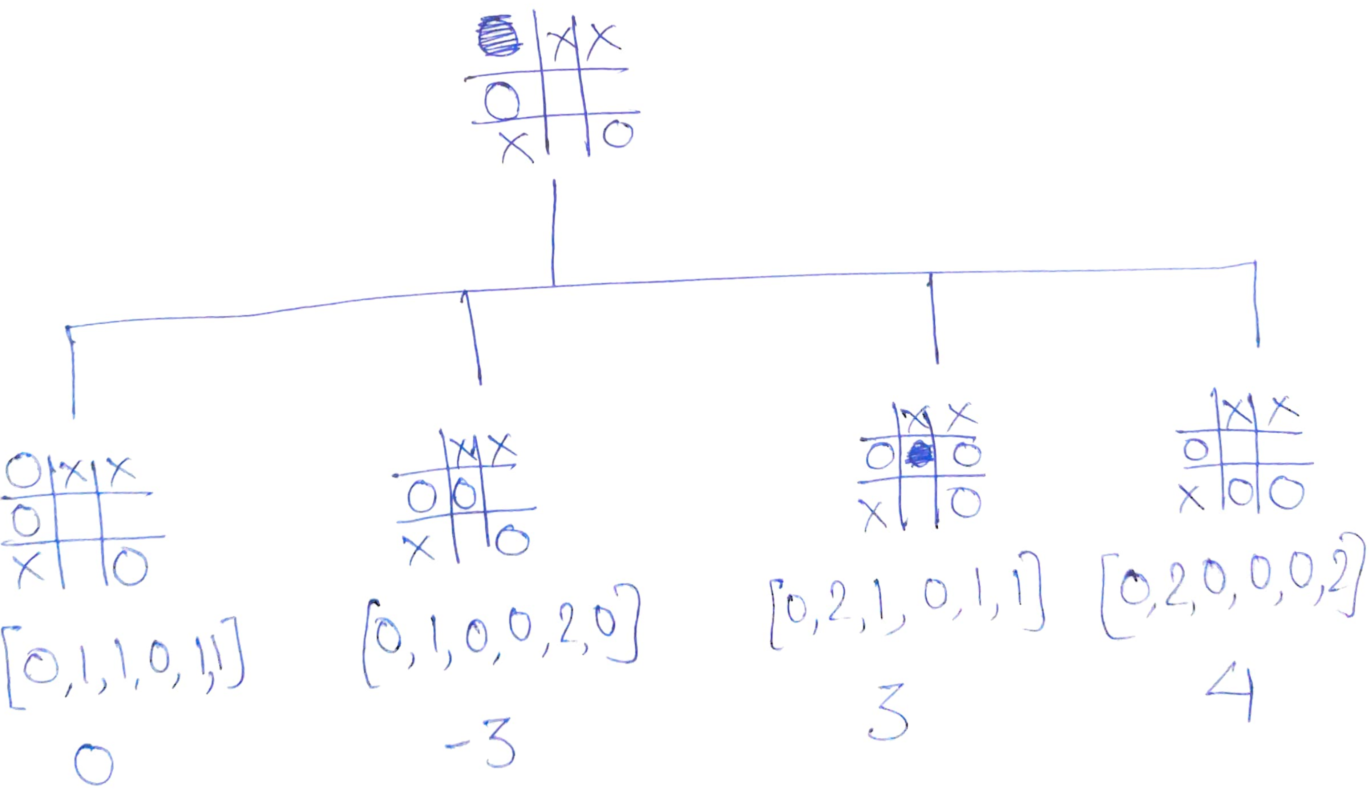


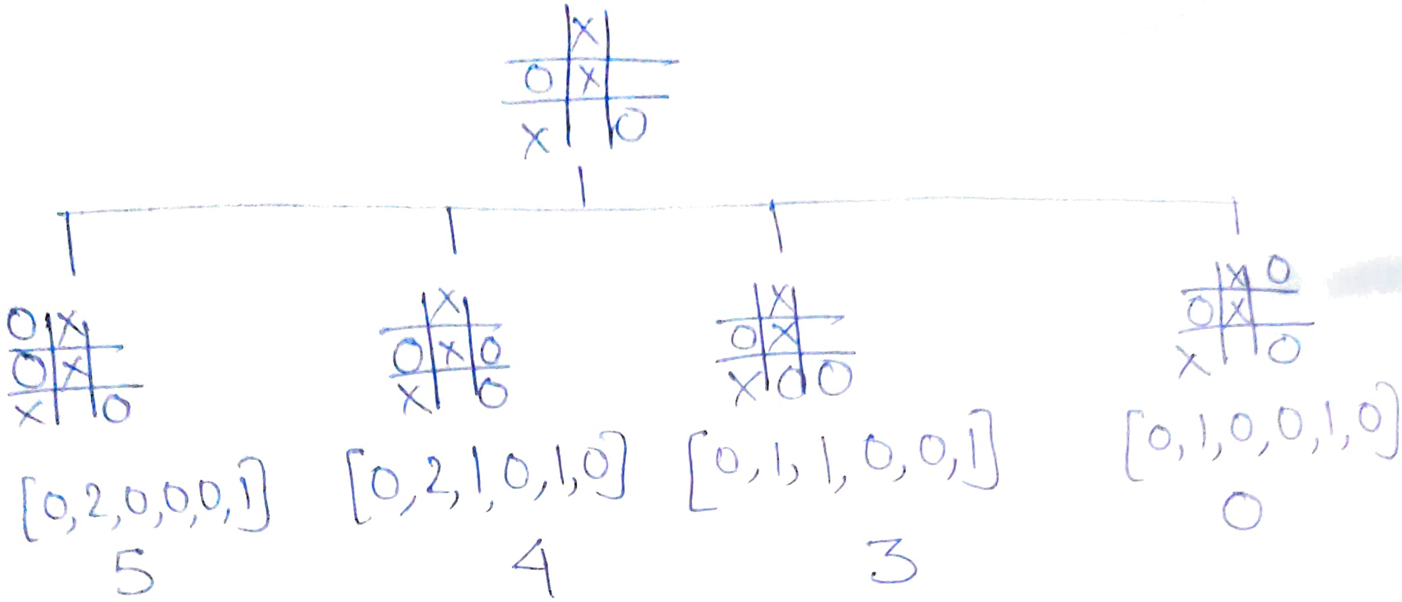
(5 states)

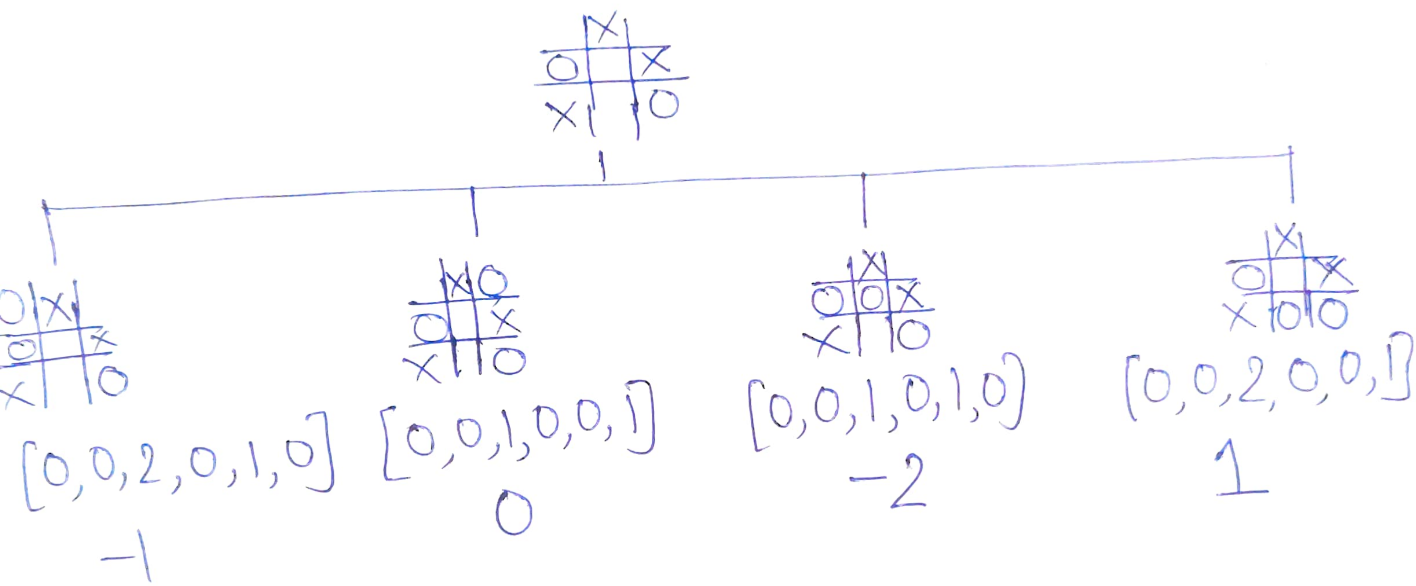
Level 3: It is O's turn

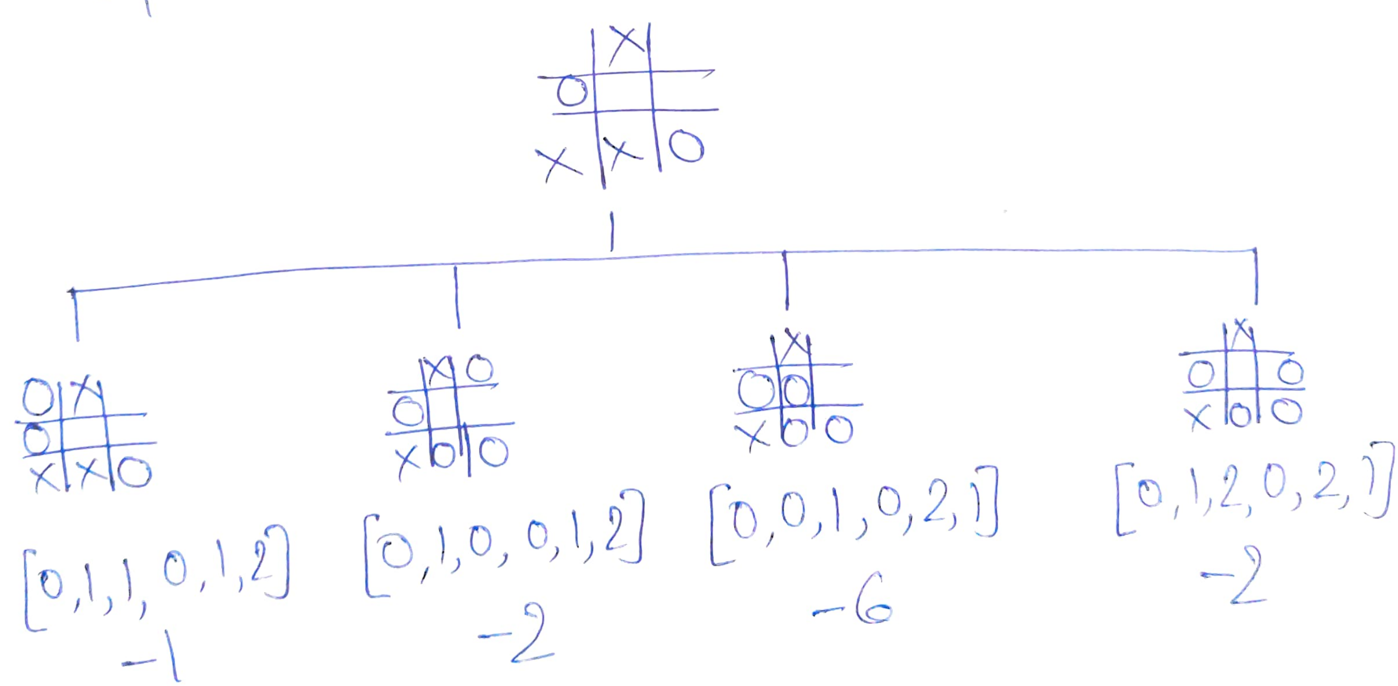
We can have 4 possibilities for each of the 5 states giving us 20 states. I will write the score below each state as they are our leaf nodes.
There will be an array of $[X_3(s), X_2(s), X_1(s), O_3(s), O_2(s), O_1(s)$ computing the score for each matrix, finally followed by score computed by the given evaluation function.





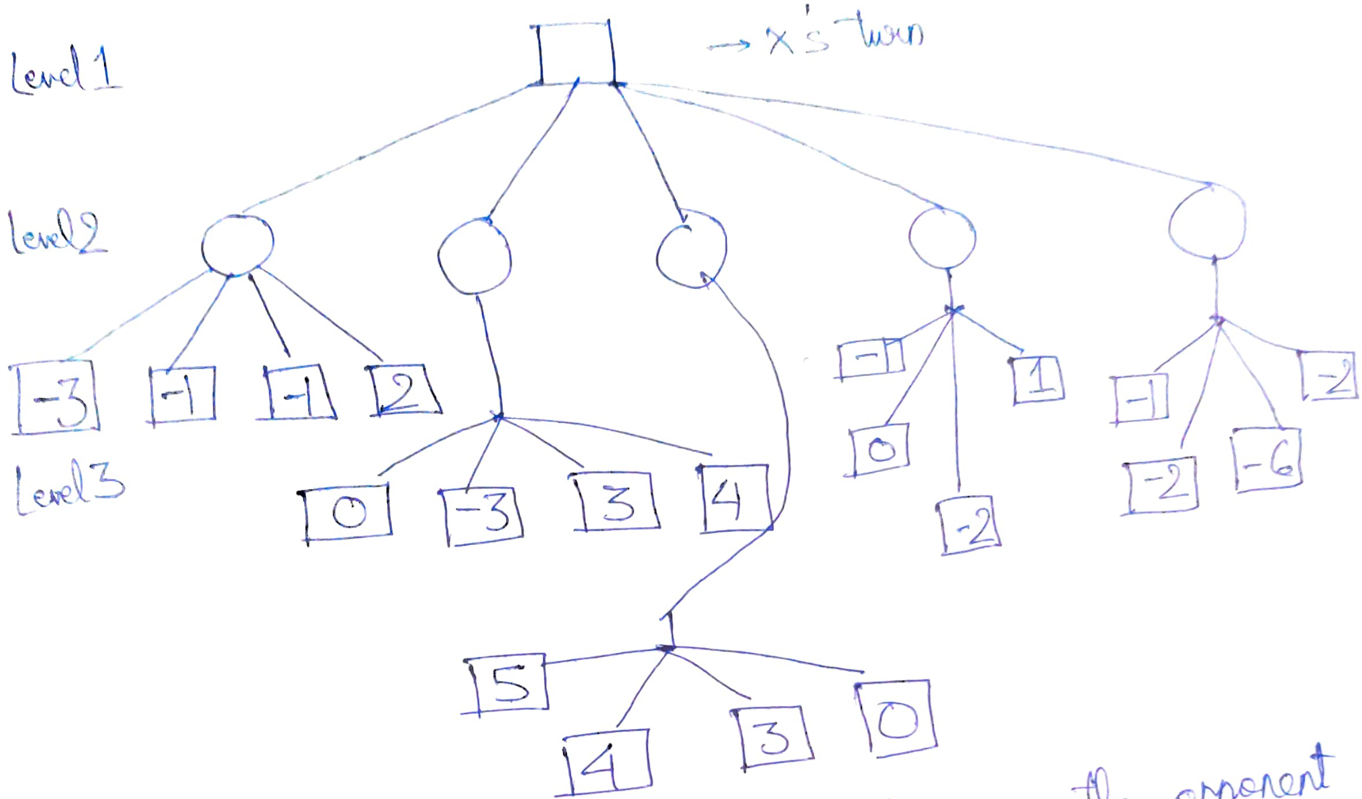

[0,2,0,0,0,1]
5
[0,2,1,0,1,0]
4
[0,1,1,0,0,1]
3
[0,1,0,0,1,0]
0


[0,0,2,0,1,0]
-1
[0,0,1,0,0,1]
0
[0,0,1,0,1,0]
-2
[0,0,2,0,0,1]
1


[0,1,1,0,1,2]
-1
[0,1,0,0,1,2]
-2
[0,0,1,0,2,1]
-6
[0,1,2,0,2,1]
-2

Drawn all states, so our tree with scores

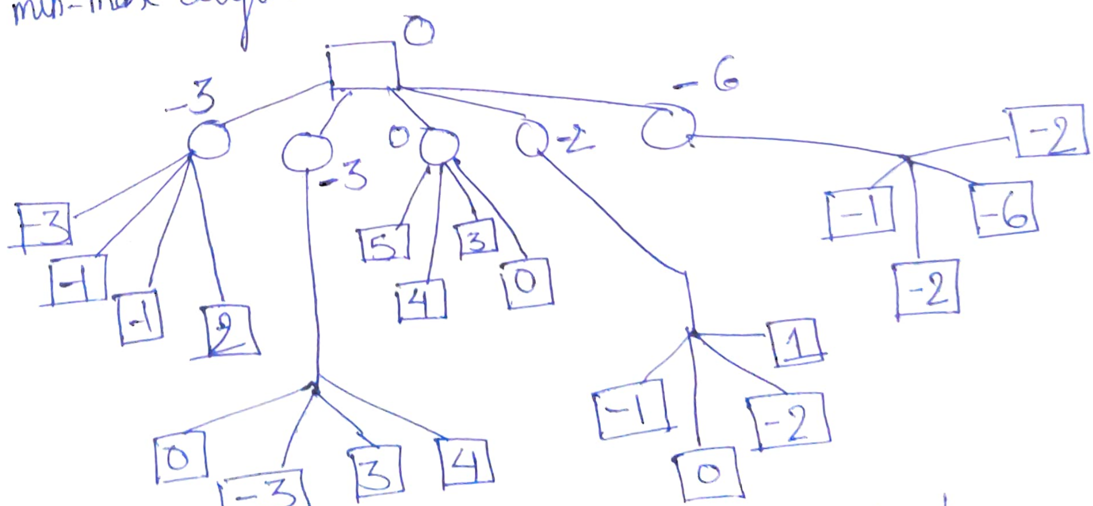
□ → MAX   ○ → MIN



so, we have ○ that take minimum as they are the opponent

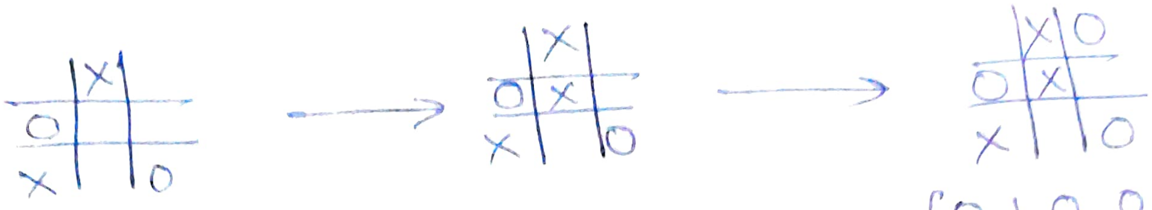
but □ that take maximum

Applying min-max algorithm on the tree



Thus, we get the score of 0 on top, which is when both players play optimally.

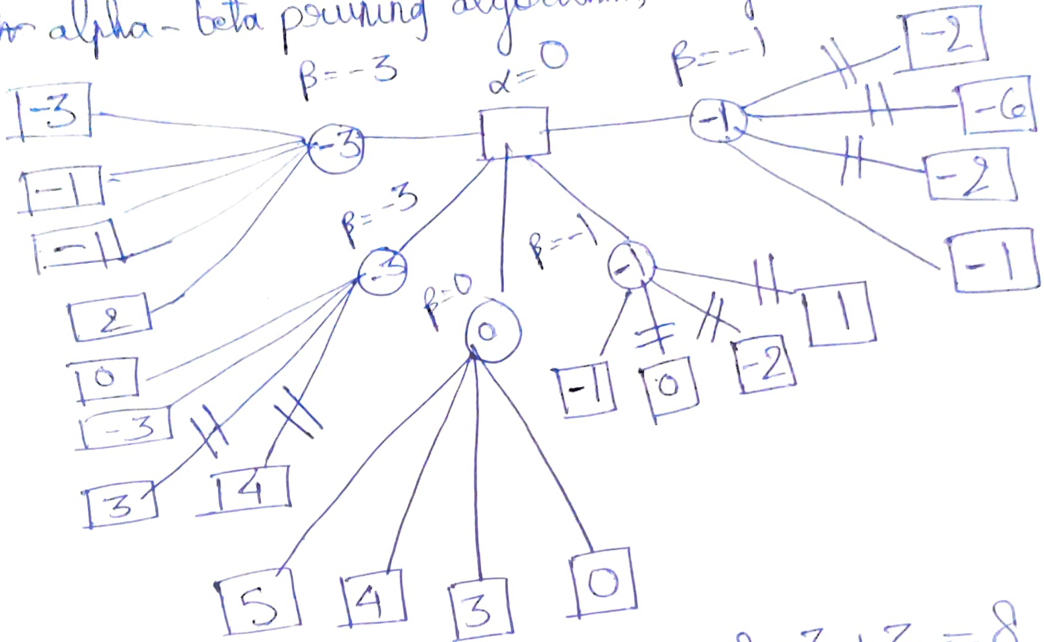
If we find the optimal move (path) from root to leaf, we will map it to the above branching of states, we get



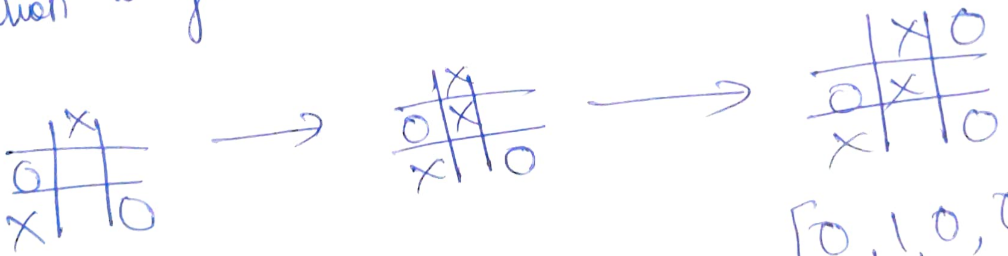
[0, 1, 0, 0, 1, 0]

0.

With ~~alph~~ alpha-beta pruning algorithm, we get



With alpha-beta pruning, we can prune 2+3+3 = 8 nodes from exploration to give us the same optimal path as before



[0, 1, 0, 0, 1, 0]

0

Using both the algorithms, we compute the same optimal path for the game tree, that is proved above.