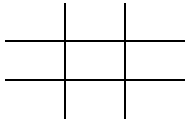
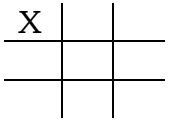
First move

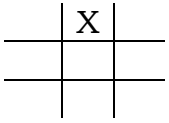


X's turn

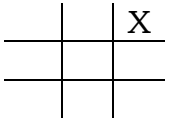
can lead to any one of



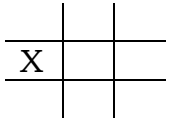
a1



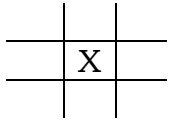
a2



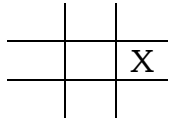
a3



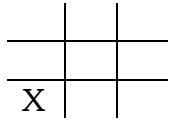
a4



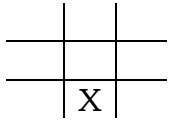
a5



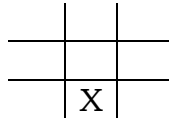
a6



a7



a8

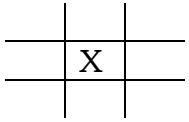


a9

O's turn

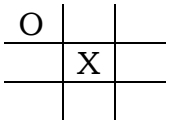
Just looking at a5:

Second move

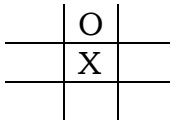


O's turn

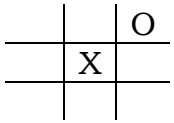
can lead to any one of



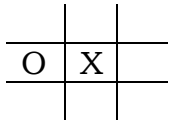
b1



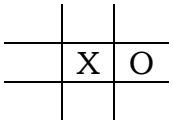
b2



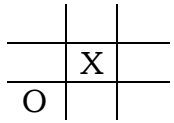
b3



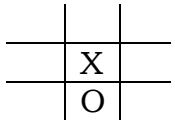
b4



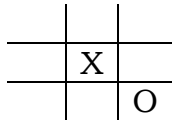
b5



b6



b7

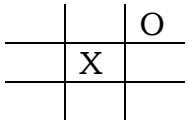


b8

X's turn

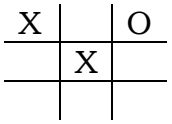
Just looking at b3:

Third move

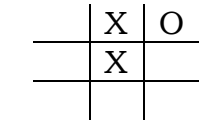


X's turn

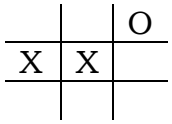
can lead to any one of



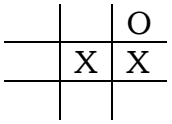
c1



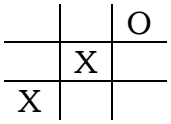
c2



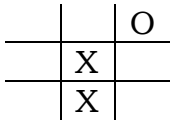
c3



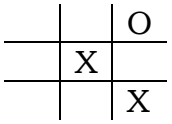
c4



c5



c6

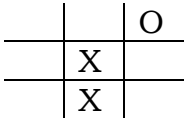


c7

O's turn

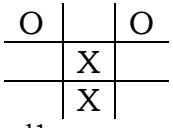
Just looking at c6:

Fourth move

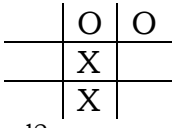


O's turn

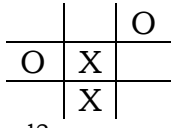
can lead to any one of



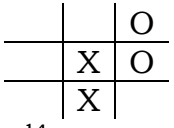
d1



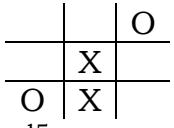
d2



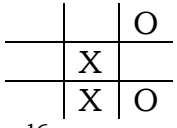
d3



d4



d5

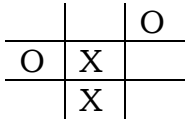


d6

X's turn

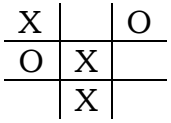
Just looking at d3:

Fifth move

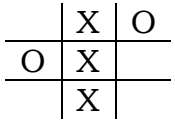


X's turn

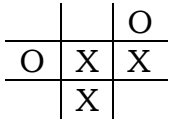
can lead to any one of



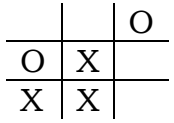
e1



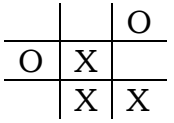
e2



e3



e4



e5

But as this is X's turn, X would obviously pick 2 and the game would be over.
X and O can both look ahead in the same way.
Seeing this outcome, O would not have picked d3 for the fourth move.
O would have picked a d_i that leads to O winning (if there was one)

But knowing that, X would not have picked c6 for the third move.
How would anyone ever pick any move?

Minimax search

`new_board(N)` is `[ [' '] * N for i in range(0, N) ]`

`possible_moves(board)` is list of int pairs `(row, col)` of all empty spaces

`move(board, whose_turn, (row, col))`
whose_turn is `'X'` or `'O'`
just creates new board same as old with that one extra move made

`ended(board) = bool`, no more moves possible: either someone has won or no blank squares left

`utility(board, player) = int`, score for that board from player's point of view, assuming the game is over
`+1` for win, `0` for tie, `-1` for loss.

```
def minimax_strategy(board, persective, whose_turn, other_player):
    if ended(board):
        score = utility(board, whose_turn)
    elif perspective == whose_turn:
        best = -2
        for rowcol in possible_moves(board):
            new_board = move(board, whose_turn, rowcol)
            score = minimax_stratgey(new_board, perspective, other_player, whose_turn)
            if score > best:
                best = score
                the_move = rowcol
    else:
        worst = +2
        for rowcol in possible_moves(board):
            new_board = move(board, whose_turn, rowcol)
            score = minimax_stratgey(new_board, perspective, other_player, whose_turn)
            if score < worst:
                worst = score
                the_move = rowcol
    return score    # Naturally we would want to return the_move too, this is just keeping it simple.
```