The game with minimax algorithm

What is Minimax?

Max is moving first and turns are changing until the game is over. Points are awarded to the winner player and penalties to the lost one. A game can be formally defined as a kind of a search problem with the following elements:

The initial sate – how game condition looks like at the start.

Player – defines which player has the move in a state

Action – returns the set of a legal moves in a state

Result – the transition model that defines the result of a move

Terminal-test – a terminal test, that is true when the game is over and false when it's in process. States where game is ended is called terminal state.

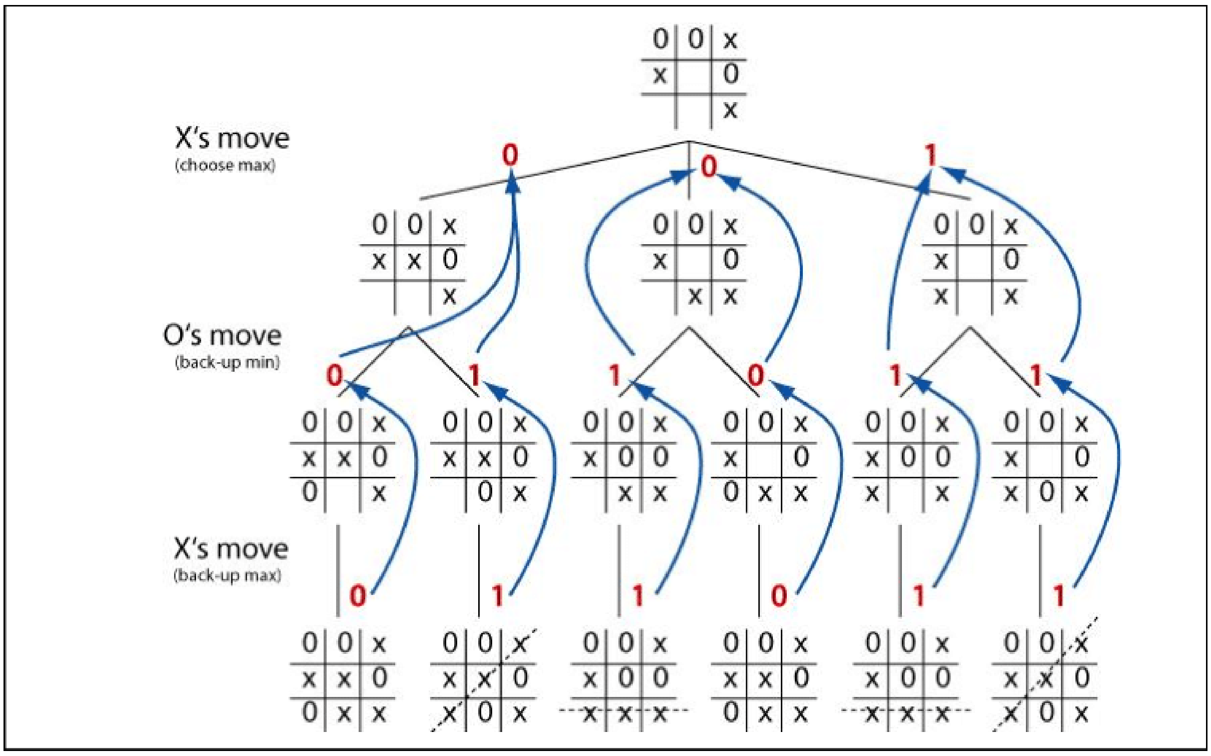
Utility – an utility function that is also called an objective function, defines the final numeric value for a game that ends with a terminal state s for a player p.

The initial state, actions function and result function define the game tree for the game – a tree where the nodes are game states and the edges are moves. In TTT game the initial state, MAX has 9 possible moves. Play alternatives between MAX's placing an X cross and MIN's placing a circle O until we reach leaf nodes corresponding to the terminal states such as one player has 3 in a raw or all of the squares are filled. The number of each leaf node indicates the utility value of the terminal state from the point of view of MAX, high values are assumed to be good for MAX and bad for MIN.

For TTT game tree is 362, 880 terminal nodes. Search tree term is used for a tree that is superimposed on the full game tree, and examines enough nodes to allow a player to determine which move to make.

MAX must find a contingent strategy that specifies MAX's move in the initial state, then MAX's moves in the states resulting from every possible response by MIN, then MAX's moves in the states resulting from every possible response by MIN to those moves and so on. The optimal strategu can be determined from the minimax value of each node, which is written as MINIMAX. The minimax value of a node is the utility for MAX of being in the corresponding state, assuming that both players play optimally from there to the end of the game. The minimax value of a terminal state is just its utility. MAX prefers to move to a state of maximum value and MIN prefers minimum value. This is recursive algorithm.

```
terminal case and the base case - terminal case is when the game is over
3 situations: 1) player 1 wins, 2) player 2 wins, 3) draw //terminal cases
Circle player is minimizing his utilities for the winning (for example).
Cross player is maximizing utilities for winning.
1. Firstly checking whether board is on the terminal case or not. 2. Board not
on a turminal case - looping through the board to get which squares are empty.
3. Minimizing player tries the circle. 4. Checking if the board is full.
Repeating again.
5. Reaching a terminal case(player 1 wins) - negative number is returned.
For maximizing player returns positive number(+1). For Draw it returns 0.
Checking all the scenarios where min or max can win and define cases that are
able to win
```



X's move
(choose max)
O's move
(back-up min)
X's move
(back-up max)
0
0
1
0
1
1
0
1
1
1
0
1
1
0
1