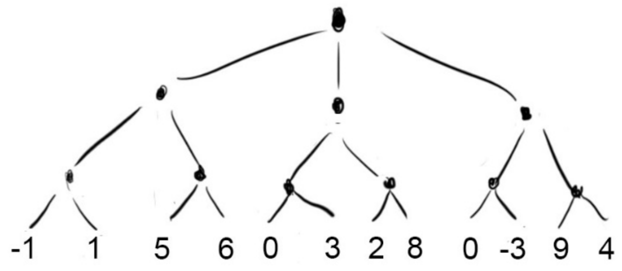
Search Problem

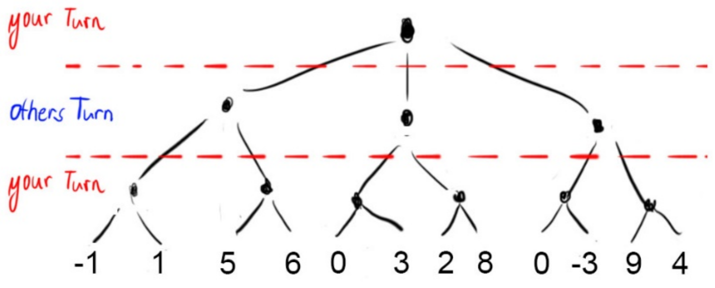
To find the best possible move for the NPC in a round-based game a common solution is to use an algorithm (**Min/Max algorithm**) on a **search tree**.
This Tree stores all possible moves from the beginning until the end of the game. The states where the game ends are called the *terminal states* or *leaves*.
The difficulty in implementing the "Search Problem" is the **enormous size** of most of the trees. For example a small game like *Tic Tac Toe* has a size of $9! = 362880$.

Min/Max Algorithm

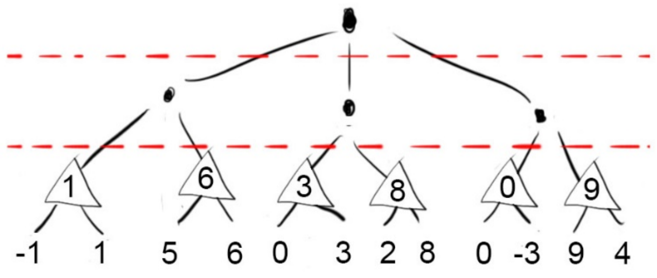
Now we know, that every possible move in a game can be stored in a tree:



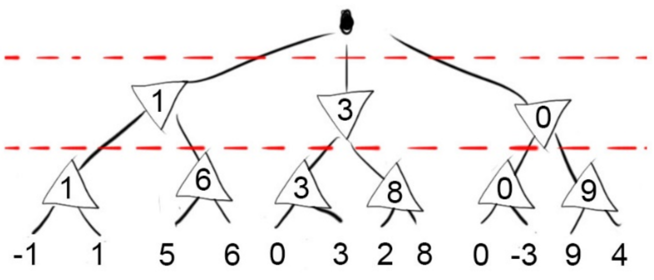
But how are we going to choose the best turn? We do it by using the Min/Max Algorithm. Keep in mind, you want the best for yourself and the enemy wants the worst for you. So we split the tree in sections, which represent your turn and the others turn:



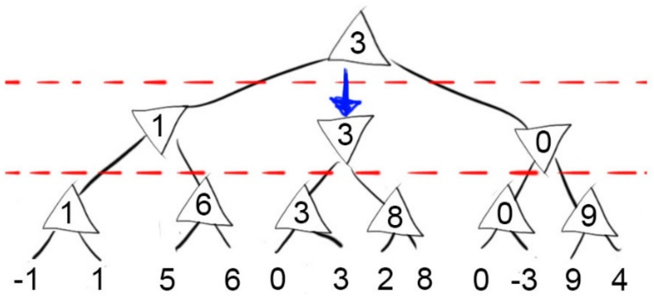
We start from the bottom. Here it's your turn, therefore the maximized result is wanted. We choose the highest number.



On the next stage it's the others turn. Remember, now we think about the move he can make to harm you most. That's why we search the lowest number.

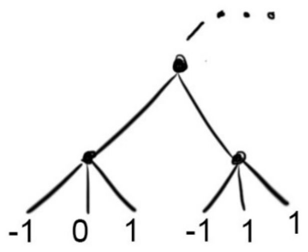


Now we maximise again and the result is the best possible move to make:



Utility Function

The utility function is applied on the leaves of the search tree and returns a value which represents the outcome of the game.
E.g.: Win = 1, draw = 0, lose = -1.



Evaluation Function

The Evaluation Function is rating the situation in a game. The Function cannot only be applied on the terminal stages of the tree. It can be used in any Layer, which means that the size can be reduced. However, applied on the leaves it should correspond with the Utility Function.

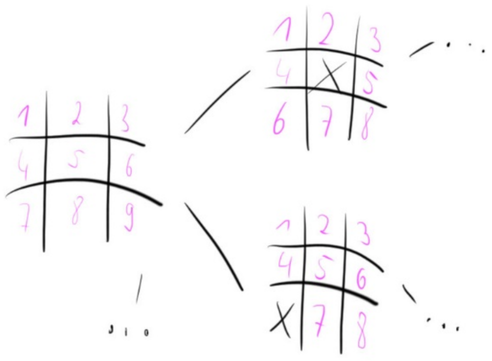
Alpha-Beta-Pruning

The Alpha-Beta-Pruning is another way of optimising the Min/Max-Algorithm, by pruning branches, which doesn't look that promising.

During the search of the best move, two values, Alpha and Beta will be updated. Alpha gives us the Value of *the best already explores option along the maximizer*, while Beta is telling us *the best already explored option of along the path to the root of the minimizer*[1].
If you want to know how this works you can watch the video which is deposited in the Footnote.

Symmetry

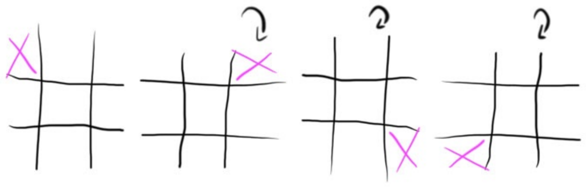
The Tree can be reduced, by finding patterns, too. I'll use *Tic Tac Toe* as an example again. As you see above, the size of a Tic Tac Toe game tree is 9!, because there are 9 possible moves at the start and then 8, 7, 6, 5… and so on.



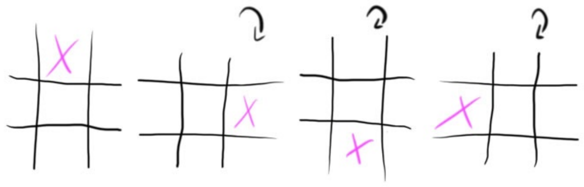
But are there really 9 moves? If we look closer, we can recognize 3 types of moves:

[1] https://www.youtube.com/watch?v=xBXHtz4Gbdo

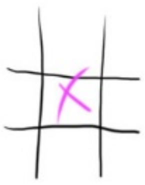
We can make our cross in a corner:



We can choose a side:



Or we can choose the middle:



So we reduced the tree from 9 possible start moves to 3.