Game Tree Search

Basic algorithm for deciding moves in a 2-player game is called the MiniMax Algorithm. The idea is to search the game tree down to a certain depth and then assign a numerical score to each board position that results (in chess, use piece count and other factors). A positive score \( \Rightarrow \) position favorable to computer, a negative score \( \Rightarrow \) position favorable to opponent.

Ex -

```
19
  \( \rightarrow \) computer's move

19
  -8
  \( \rightarrow \) opponent's move

13 9 6 -8 5
  \( \rightarrow \) computer's move

13 -2 9 6 -8 -15 3 5
```

> how should we decide values of positions at this level?

Since it is the computer's move and our alg. is operating on behalf of the computer, pick the best move (maximum value from the next level) down.
how do we decide value at next level up?

Since the opponent has the choice at that level, we assume we are playing an intelligent opponent who will pick the best move for them, so pick the minimum value from the next level down.

At top level, it is computer's choice again, so pick the maximum.

So, root node is assigned value '6' and the computer picks the move associated with the left branch.

This idea of maximizing at levels where the computer is moving and minimizing at levels where the opponent is moving is the MiniMax Algorithm.

Tic Tac Toe Example:

+1 = win for computer

0 = draw

-1 = win for opponent

Computer's move

Opponent's move

Computer's move
we can modify basic Minimax to avoid searching some subtrees when they cannot change the outcome. This is called the Alpha-Beta Pruning Algorithm.

Examples where pruning can occur:

We had this situation in our tic-tac-toe example:
So MiniMax Alg. w/ Alpha-Beta Pruning is a slightly more efficient than basic MiniMax (exhaustive search down to a certain level)

additional enhancements:

- timed moves ⇒ use **progressive deepening**
  - 1st search 2 levels
  - then search 3 levels
  - then search 4 levels
  
  when time expires, pick best move from last level finished

since tree grows exponentially, each level takes as much time as all previous levels combined, so we don’t lose much in efficiency by doing this

- heuristic pruning can speed up search by a lot but does not guarantee best move as MiniMax w/ Alpha-Beta does ⇒ idea is to focus on a few moves at every level, danger is that in some very good moves (Queen sacrifice) will not be explored or very bad moves will be selected (because we didn’t explore the opponent’s move that made the choice really bad)

Note about tic-tac-toe: total no. of possible games < 9! = 362,880

(<, not =, because some games end before 9 moves are made)