Lecture 8: Applications of Trees and Graphs

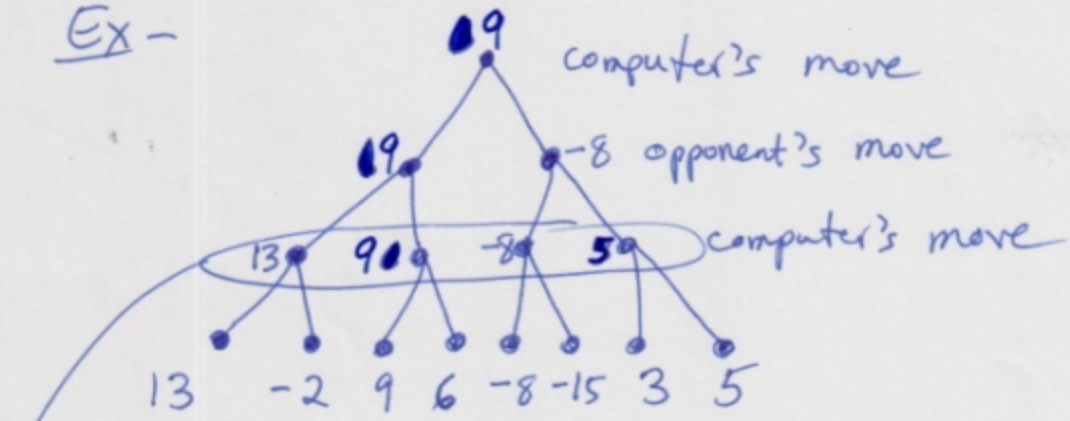
Game Tree Search

Basic algorithm for deciding moves in a 2-player game is called the <u>MiniMax Algorithm</u> → this is at the core of every chess program or other board game program

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)

positive score ⇒ position favorable to computer
negative score ⇒ position favorable to opponent

Ex -



→ 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 (<u>maximum</u> 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 <u>minimum</u> value from the next level down

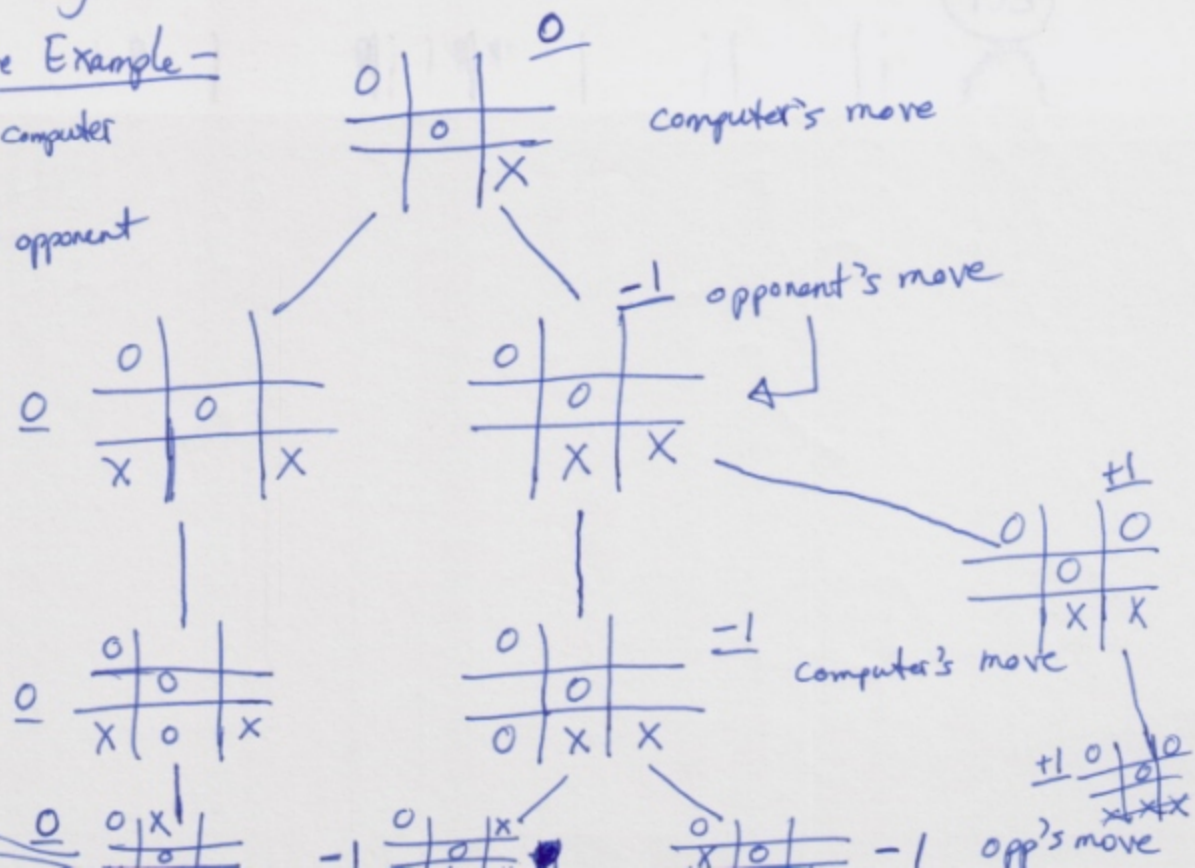
at top level, it is computer's choice again, so pick the maximum

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 Mini Max Algorithm

<u>Tic Tac Toe Example</u> –

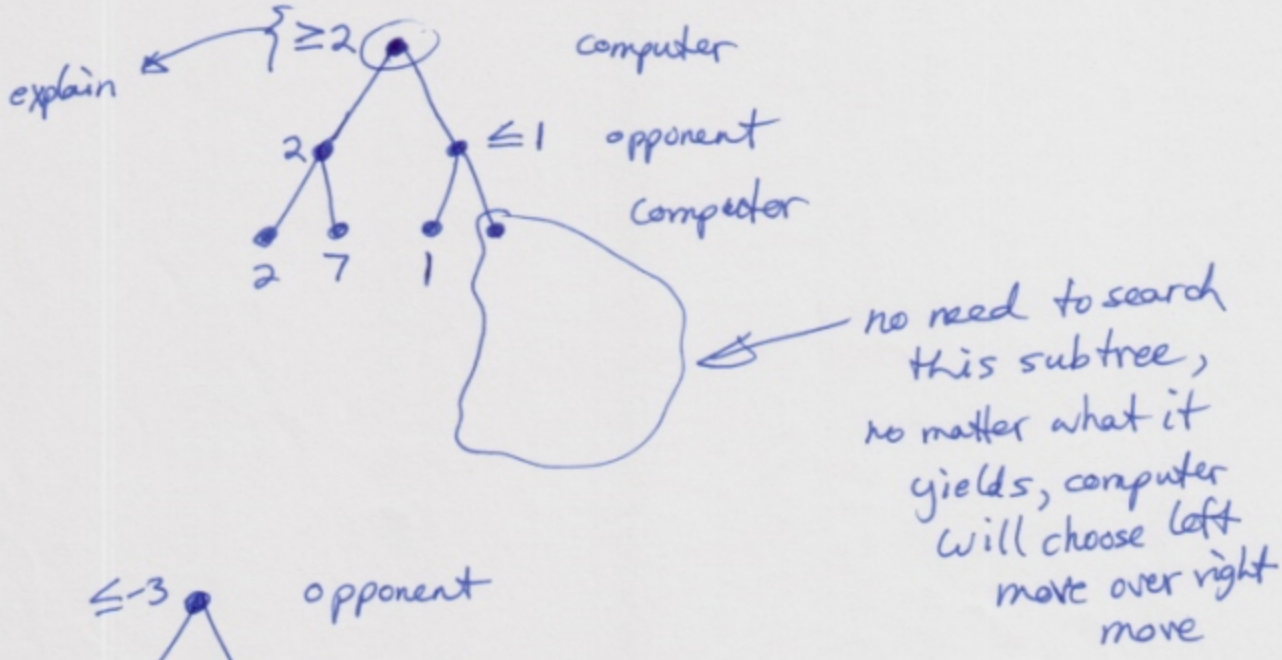
+1 = win for computer
0 = draw
−1 = win for opponent

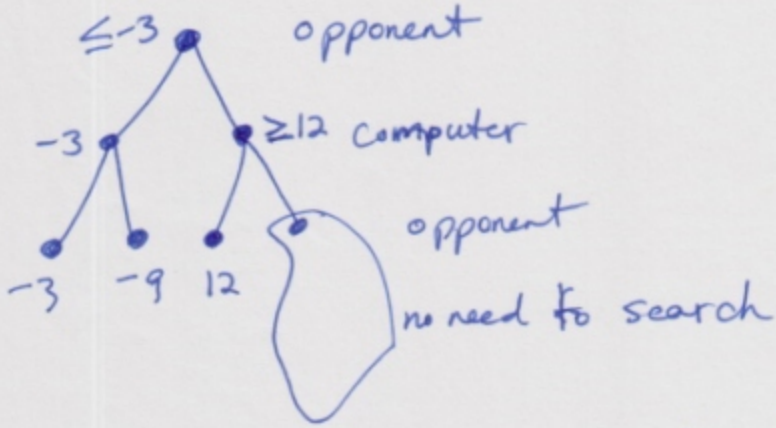


we can modify basic MiniMax to avoid searching some ~~branches~~ subtrees when they cannot change the outcome

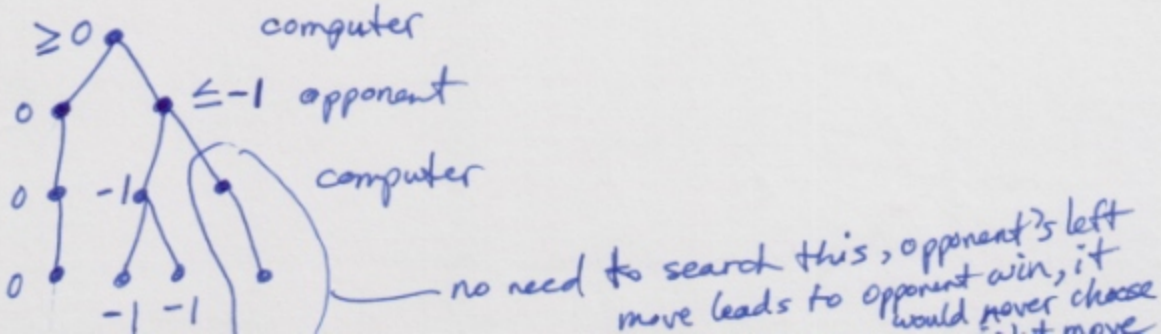
this is called the <u>Alpha-Beta Pruning Algorithm</u>

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 approach than basic MiniMax (exhaustive search down to a certain level)

additional enhancements:

- timed moves ⇒ use <u>progressive deepening</u>

  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 situations 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)

<u>Note about tic-tac-toe:</u> total no. of possible games $< 9! = 362,880$
($<$, not $=$, because some games end before 9 moves are made)

w/ this no., a MiniMax Alg. can easily search the entire space of games and play "perfect" tic-tac-toe, i.e. best opponent could hope for is a draw