CSE 4308 / CSE 5360 - *Artificial Intelligence I*

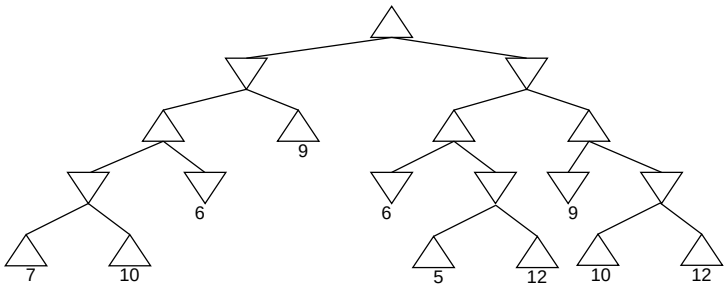
Homework 2- Fall 2013

Sample Solution

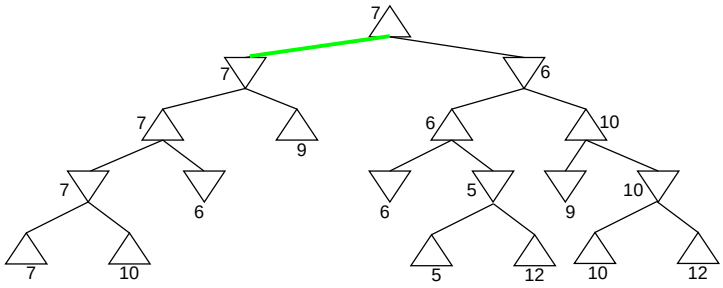
Problems marked with a * are required only for students in the graduate section (CSE 5360). They will be graded for extra credit for students of CSE 4308.

Minimax Search

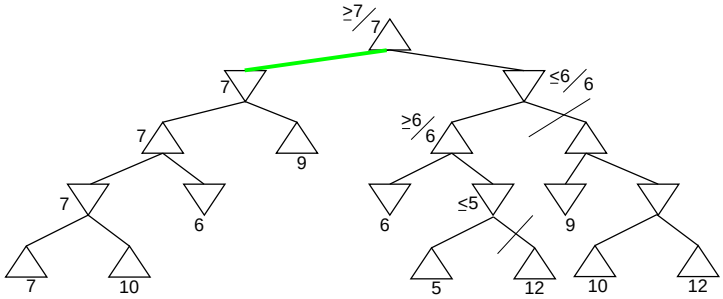
1. Consider the following game tree where the utilities of terminal nodes are indicated below the leaf nodes.



a) Perform standard Minimax search on the tree and indicate the resulting utilities at all the nodes in the tree. Also indicate which of the two choices the Max player should play at the root of the tree.

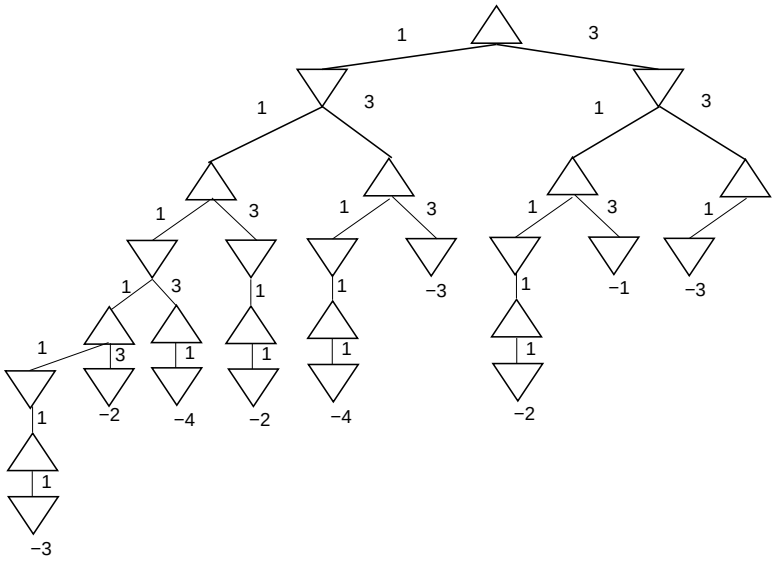


b) Perform Minimax search with alpha-beta pruning on the tree. Indicate which branches are pruned (assuming that the search visits nodes from left to right). Also indicate next to each node either its true value or its α or β value.

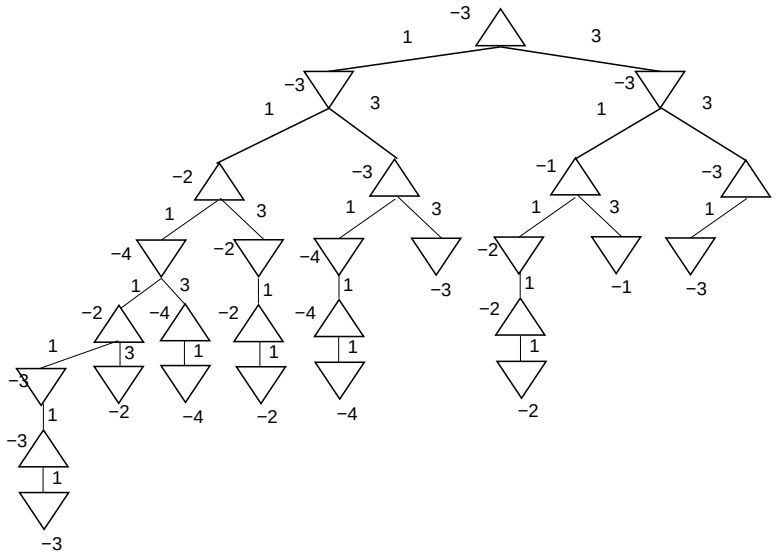


2. Consider the following two-player game. Starting with 7 tokens on the table, each player at each move can either pick up 1 or 3 of them. Whoever picks up the last one looses and has to pay the other player a prize equal to the number of tokens that the winning player has picked up during the game.

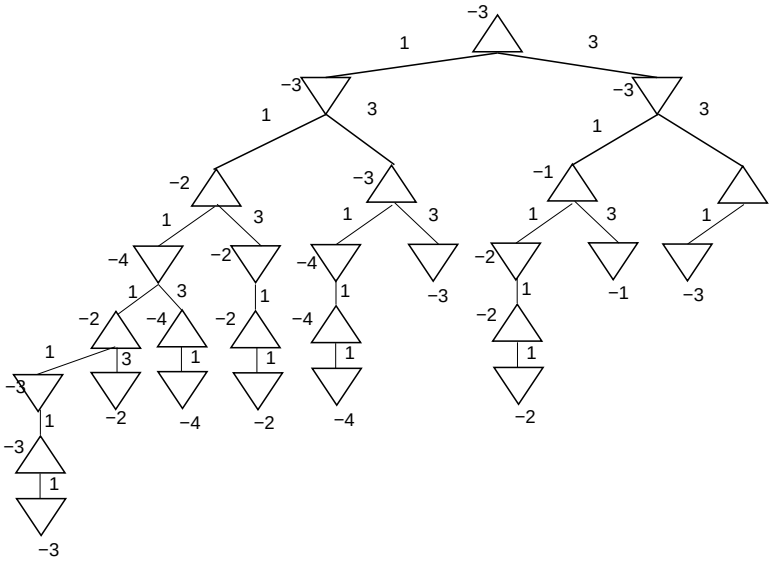
   a) Draw the corresponding Minimax search tree, including the utility values at the leaf nodes.



   b) Perform Minimax search on the tree and indicate all the utilities at the nodes in the tree.



   c) Perform Minimax search with alpha-beta pruning on the tree. Indicate which branches are pruned (assuming that the search visits nodes from left to right). Also indicate next to each node either its true value or its α or β value.



None of the branches can be pruned (note that this is different if actions are ordered differently, i.e. if the left child corresponds to 3 tokens).

Game Playing

3. Consider a generalization of the game in Problem 2 where there are initially n tokens and each player has 3 choices, namely removing 1, 2, or 3 tokens. Scoring (i.e. utility calculation) works the same way as in Problem 2.

   a) Implement a game playing program that (starting with the initial n tokens) plays against a user. The program should first ask if the human or the computer has the first move and then start the game. Whenever it is the user's turn, the program should read a move from the keyboard. For moves by the computer you should implement an appropriate game search (this also requires the capability to identify terminal nodes, i.e. nodes where there are not tokens left). Once the game terminates your program should indicate who won and what the score obtained by the human player is.

   To limit the search time of the computer (for larger n this problem is too complex to search it exhaustively) you have to implement a limit on the number of nodes that are opened before the search terminates. For this assignment your program should open no more than 400 nodes before choosing a move (if you make this parameter flexible you can use it to regulate the strength of your computer player). Not being able to exhaustively construct the search tree implies that you have to implement a heuristic evaluation function which replaces the actual utility in cases where the search stops before reaching terminal nodes (such an evaluation function could look at a variety of different features - e.g. since it is also possible for the opponent to force a round to collectively remove 4 tokens, the remainder of a division by 4 can be indicative of part of what is achievable. (Note: it is very difficult for this problem to come up with a very good evaluation function. While the program will play stronger with a better evaluation function, the goal here is not to come up with an optimal evaluation function).

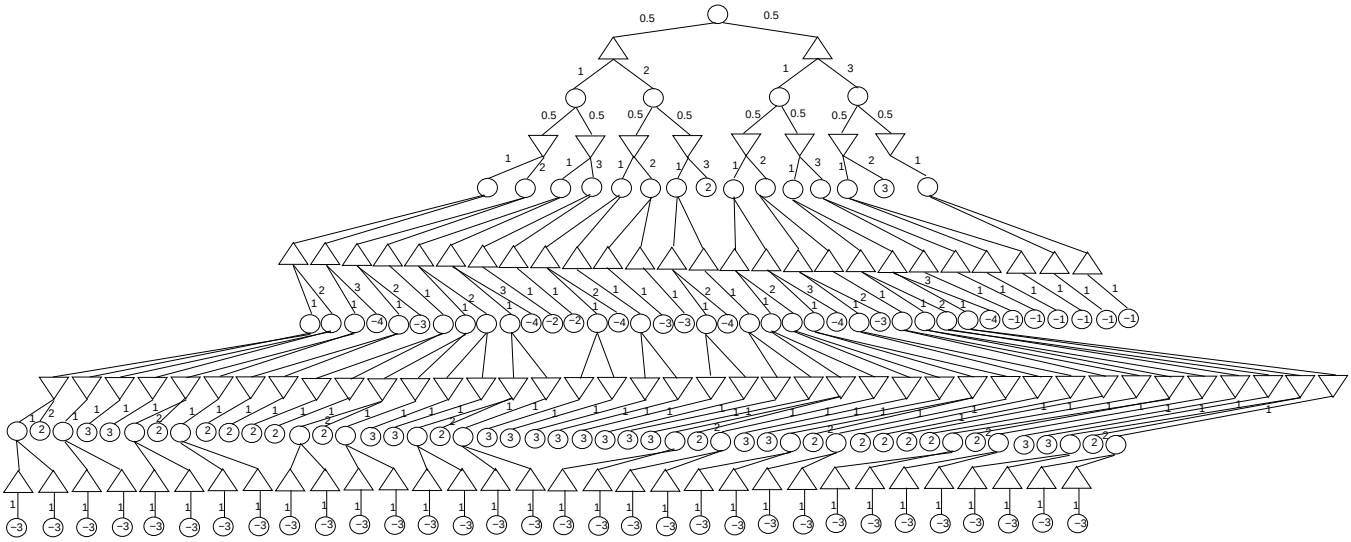
   b)* Implement a second pruning strategy for your computer player to decide which nodes to expand and when to stop (e.g. $\alpha - \beta$ pruning or quiescence). Your agent with this second pruning strategy still has to respect the 400 node limit. Run this agent against your first agent and compare their

performance (e.g. which player was stronger ? How many nodes did each one expand ?, How many plies did each player look for (i.e. what was the depth of the deepest node the player looked at) ?).

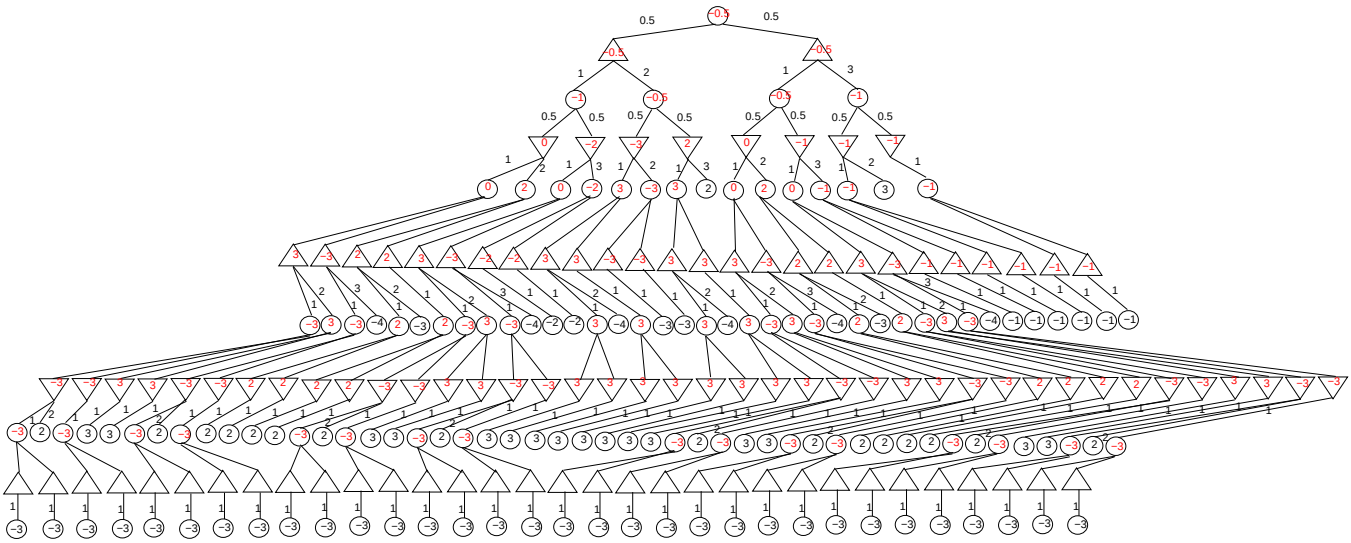
Expectiminimax

4. Consider a modified version of the game from Problem 3 where each player in each round has to first flip a coin and based on the outcome has the choice either between removing 1 or 2 tokens, or between removing 1 or 3 tokens. The scoring is again performed in the same way as before.

   a) Draw the Expectiminimax search tree, including the utility values at the leaf nodes, for a start state in which there are 5 tokens on the table.



   b) Perform Expectiminimax on the tree and indicate all the utilities at the nodes in the tree.



c)* Implement Expectiminimax without pruning (i.e. such that it explicitly visits all the nodes) for this problem with a general start state with n tokens on the table. Note that you are not asked to come up with an evaluation heuristic or pruning function (however, this algorithm will potentially run for a very long time for large values n).