CS 182, PROBLEM SET 3

Due: November 1, 2021 11:59pm

This problem set covers Lectures 8, 9, 10, 11. The topics include game theory, AI game playing, Stackelberg security games, and social choice theory.

1. (15 points) *Comprehension.*

(1) (5 points) *Game Theory.* The following payoff matrix shows a game between Paul and Fiona. There are three cards on a table: a red card, a green card, and a blue card. Each player picks one card, and both players pick their card without knowing the other player's choice. Both players also know the entire payoff matrix.

	Paul: red	Paul: green	Paul: blue
Fiona: red	F = 7, P = 1	F = 9, P = 2	F = 6, P = 3
Fiona: green	F = 8, P = 10	F = 7, P = 5	F = 7, P = 1
Fiona: blue	F = 7, P = 2	F = 8, P = 2	F = 8, P = -3

Find a pure Nash equilibrium for this game, and then find a mixed-strategy Nash equilibrium where both players have strictly positive probability of playing each action.

(2) (5 points) *AI-game Playing.* Paul and Fiona decide to play one final game, in which Paul is attempting to maximize his score, while Fiona attempts to minimize Paul's score. Both Paul and Fiona play optimally, and the game tree is shown in the following figure. Perform the minimax algorithm with alpha-beta pruning to find the alpha and beta values passed to each node when the node is first called. Include an image with the alpha beta values at each node, as well as an indication of which branches are pruned. Assume that the algorithm visits each child from left to right. What are the actions that each player takes?

Date: October 18, 2021.

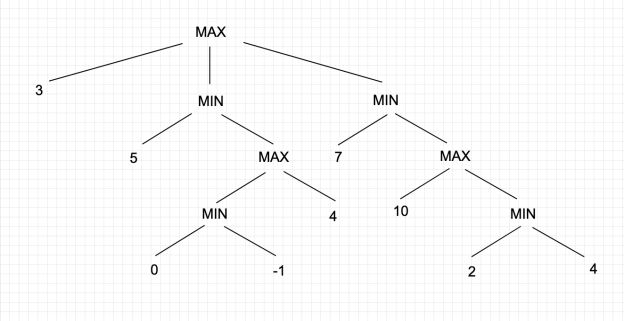

FIGURE 1. Game Tree for Problem 1.3

(3) (5 points) *Social Choice Theory.* Paul and Fiona are joined by Marissa, and each of them desire to run for student body president. Construct an example of a preference profile with 3 alternatives $A = \{\text{Paul, Fiona, Marissa}\}$ in which the plurality voting rule and the STV voting rule winner is Paul but the Borda count voting rule winner is Fiona.

2. (30 points) Alice and Bob are playing a sequential, turn-based game on a tree. The game starts with some initial tree T. The two players alternate turns, with Alice making the first move. On each turn, a player selects a node of the tree and removes that node, along with the subtree rooted at that node. See 2 for an example.

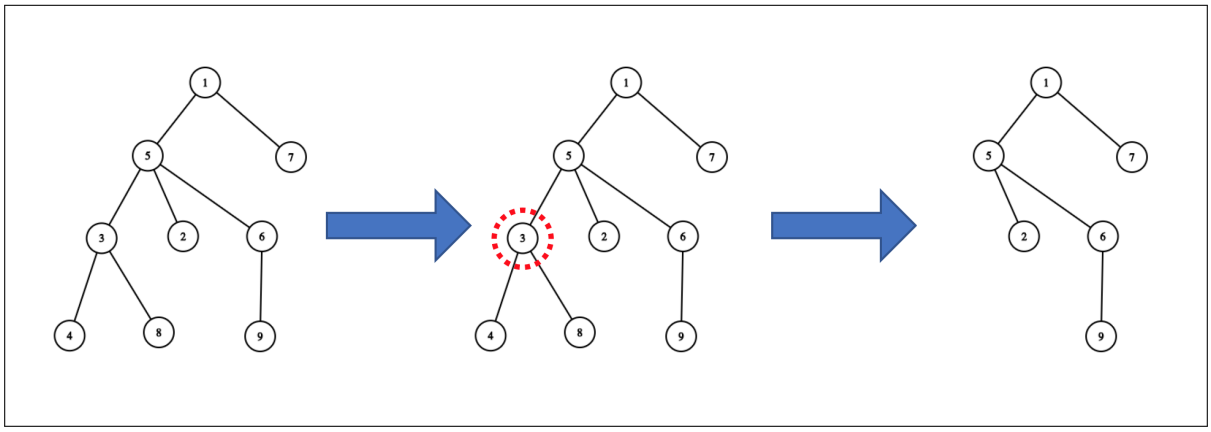

FIGURE 2. Example turn of the subtree-pruning game. Here, the player removes node 3.

The game ends when the root node is taken, and the player who was forced to take that node loses. This is a zero-sum game, so payoffs are $(+1, -1)$ if Alice wins and $(-1, +1)$ if Bob wins.

(1) (10 points) The *perfect binary tree* of order k, denoted B_k, is the tree defined recursively as follows:
- If $k = 1$, then B_k is a single node.
- If $k \geq 2$, then B_k consists of a root node, with two copies of B_{k-1} as children.

Suppose that Alice and Bob are playing the tree-pruning game on $T = B_k$, for some $k \geq 1$. Assuming *perfect play* (i.e., subgame-perfect Nash equilibrium), show that Bob always wins the game, and describe a winning strategy.

(2) (10 points) The *perfect ternary tree* of order k, denoted C_k, is also defined recursively:
- If $k = 1$, then C_k is a single node.
- If $k \geq 2$, then C_k consists of a root node, with three copies of C_{k-1} as children.

This time, Alice and Bob are playing the pruning game on $T = C_k$ for some $k \geq 1$. For each possible value of k, which player wins the game, assuming perfect play? Describe a winning strategy in each case.

(3) (10 points) Generalize the game from binary (2-ary) and ternary (3-ary) trees to perfect n-ary trees, for some positive integer n. For each value of $n \geq 2$ and $k \geq 1$, who wins, and why?

3. (20 points) Recall that a voting rule is Condorcet consistent if the rule selects an alternative that is a Condorcet winner whenever there exists a Condorcet winner in the given preference profile. As in class we denote the set of voters by $N = \{1, \ldots, n\}$ and the set of alternatives by A, where $|A| = m$.

(1) (10 points) The minimax voting rule (not to be confused with the minimax algorithm) picks an alternative that minimizes the maximum number of voters by which another alternative beats this one, i.e.,

$$\operatorname*{argmin}_{x \in A} \max_{y \in A \setminus \{x\}} \text{Score}(y, x),$$

where $\text{Score}(y, x) = \sum_{i=1}^{n} \mathbb{1}[y \succ_i x]$ counts the number of voters who prefer y over x, with $\succ_i$ denoting preference according to the ranking of voter i. For example, in the profile on slide 5 of Lecture 11, $\text{Score}(b, c) = 4$, as there are 4 voters who prefer b to c. The minimax rule would choose b, as $\max_{y \in A \setminus \{b\}} \text{Score}(y, b) = \text{Score}(a, b) = 2$.
Show that the minimax rule is Condorcet consistent. You may assume that the number of voters n is odd.

(2) (10 points) A *positional scoring rule* using a score vector $(s_1, \ldots, s_m)$, where $s_1 \geq \cdots \geq s_m$, is one that determines a winner by picking an alternative with maximum score, where scores are determined as follows:

- Each voter i gives s_j points to the alternative that is the jth highest on their ranking (e.g., the highest alternative in σ_i receives s_1 points and the lowest alternative σ_i receives s_m points).
- Each alternative's score is the sum of the points given over all voters.

Answer the following questions about positional scoring rules.

(a) (4 points) Explain why both Plurality voting and Borda voting are examples of positional scoring rules.

(b) (6 points) Under the assumption that $s_1 > s_2 > \cdots > s_m$, prove that no positional scoring rule is Condorcet consistent.

Hint: Construct a preference profile with three alternatives and a smallish number of voters such that there is a Condorcet winner but any positional scoring rule with $s_1 > s_2 > s_3$ would select a different alternative.

4. (35 points) *AI Game playing in Ghost.* In this problem, you will solve the game of *Ghost*[1] using alpha-beta pruning. This game is played with two players, as follows. The first player chooses an English letter, the second player adds another letter to the end of the first player's letter, the first player then adds another letter to the end of that, and so on. The key is that, at any given point in time, the growing string must have the ability to form a word, but the first player who is forced to make the string into a word loses. For example, an example game might play out as follows:

First Player: W *// Growing string: "W"*

Second Player: A *// Growing string: "WA"*

First Player: T *// Growing string: "WAT"*

Second Player: E *// Growing string: "WATE"*

First Player: R *// Game ends, "WATER" forms a word*

Note that the first player played an R at the end, so since "WATER" is a word, the first player loses.

We have modified the game in the following way: the value of a player's is the reciprocal of the length of the word in the terminal state, flipped with the appropriate sign depending on which player's turn it is. This has the effect that players aren't merely interested in winning, but they are also interested in winning quickly whenever they can guarantee their victory, and they want to make the game last as long as possible if their opponent can guarantee victory instead,

The code file is located in `pset3.py`. We have included a text file of English words in `dictionary.txt`. Note that you only have to implement the alpha-beta pruning agent, and not any of the Ghost game details. Our autograder will run your agent against ours and check that your agent wins in all the situations in which it should win. Note that the Gradescope Autograder corresponds to the point values in parts (1), (2), and (4) of this problem.

(1) (5 points) Implement the `MinimaxAgent`.

(2) (10 points) Implement the `AlphaBetaAgent`. You should find that the number of calls to `getSuccessor` is smaller than before (which would indicate that pruning has improved our search).

(3) (5 points) Recall that the minimax algorithm helps you find the optimal value against an opponent who is also playing optimally. Now suppose that you know that your opponent is not necessarily playing smart; instead, your opponent simply just picks a

[1]The authors of this problem are unclear whether this is the universal name for this game, but we shall go with it.

random letter that is a valid move. Describe the strategy that maximizes or minimizes (depending on whether you are the max or the min agent) the expected value of the terminal game state.

(4) (10 points) Implement the agent in part (3) in `OptimizedAgainstRandomAgent`. You should implement both cases, whether the agent wants to maximize or minimize the terminal value of the game, depending on the agent's index.

(5) (5 points) In `simulate_versus_random`, we compare the performance of the agent from part (4) against a random agent with the performance of a minimax agent against a random agent. We take the random agent to be the minimizer. We compute the average value that each agent wins over `k=10000` trials. By about how much does your agent from (4) beat the minimax agent on the initial game state prefixes [`beh`, `feb`, `gw`]? Explain your intuition for why the `OptimizedAgainstRandomAgent` beats the `MinimaxAgent` on average against a random agent (there is more than one possible answer to this question).