Artificial Intelligence Homework 2
Due date: 2/27 by 11:59pm

Search applied to adversarial games gives rise to the Minimax algorithm. In this assignment, you will use Minimax to find the optimal strategy for the game of Connect Four. This assignment is more substantial than the previous one so give yourself enough time to work through it.

This is a programming assignment and you may work with a partner if you choose. Click here to download a directory with the following Java packages:

- **graphics** - This package contains the Java classes responsible for graphically representing the game and calling your computer player.

  The `ConnectFour` class contains the main() method that runs the game. It is this class that you should compile and run. This class takes one command line argument: the number of ply the computer player should look ahead (default is 1-ply).

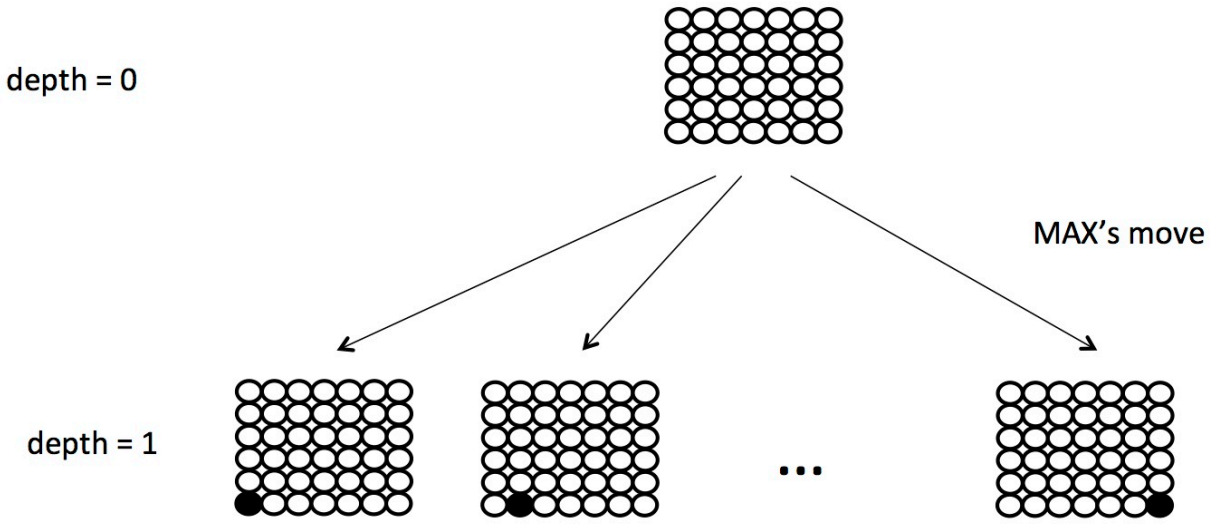
- **players** - This package contains two players for the game: a human player (which simply returns back the move made by the human), and a primitive computer player (which always picks the leftmost open column).

  You are responsible for making this computer player more intelligent by implementing the Minimax algorithm. In particular, the method `getNextPlay()` should take in the board game, and use Minimax to determine the column where the computer should play.

Although all of the code you will write is inside of `ComputerConnectFourPlayer` class, it might be helpful to skim through the `ConnectFour` class so you have a sense of how the game is setup and to cannabilize some of the more useful logic.

Since it is not computationally feasible to run the unmodified Minimax algorithm, you should instead implement the heuristic Minimax algorithm (Section 5.4). This replaces the goal test with a cutoff test and replaces the utility function with an evaluation function.

Cutoff test: The constructor of the ComputerConnectFourPlayer will be passed a depth. Your cutoff test should return true for any node at this depth. For example, suppose we're at the very beginning of the game and the depth is 1. Then the game tree would look like:



If the depth was 2, the game tree would include the next level which are all the possible moves that MIN could make in response. Your cutoff method should also return true for actual terminal states that occur before the cutoff depth.

Evaluation Function: The evaluation function should take in the board and look at all horizontal, vertical, and diagonal spans of 4 contiguous slots. (A standard game board has 24 horizontal spans, 21 vertical ones, and 24 diagonal ones). For each span of 4 contiguous spots,

- If it has 4 tokens of the AI's color and none of the opponent's, it is worth an infinite amount of points
- If it has 3 tokens of the AI's color and none of the opponent's, it is worth 100 points.
- If it has 2 tokens of the AI's color and none of the opponent's, it is worth 10 points.
- If it has 1 token of the AI's color and none of the opponent's, it is worth 1 points.
- If it has no tokens, or it has a mix of the AI's and the opponent's tokens, it is worth 0 points.
- If it has some of the opponent's tokens but none of the AI's, then it is worth the same point total just negated. For example, 3 of the opponent's tokens but none of the AI's should be worth -100 points.

The evaluation function should iterate over all such spans and, for each span, accumulate the values given above. The total sum is then returned.

Additional Information:

- Wikipedia has a description of Connect Four if you've never played or it's been awhile.
- The game board is represented as a 2-dimensional array of bytes. A byte is an integer type that has a very limited range (from about -127 to 127). The number 0 is used to denote an empty slot in the board. The numbers 1 and -1 are used to represent the tokens for the two players.
- This code can get messy fast so I recommend testing each method as you write it. In particular, if you write any smaller helper methods, test them inside a main method. That way you'll only have to worry about the correctness of the actual minimax methods themselves and you'll have confidence that all of your helper methods are correct.
- You do not need to implement Alpha-Beta pruning
- Watch out for overflow or underflow if you use Integer.MAX_VALUE and Integer.MIN_VALUE

Running Your Code
I recommend adding a main() method to your ComputerConnectFourPlayer to test your implementation. To play the actual game, compile and run the ConnectFour class. Again, this takes a single command line argument which is the number of ply to look ahead (default of 1).

Grading and Submission
Your assignment will be graded based upon correctness. Most likely, I will either play against your AI player using a pre-determined set of moves or I'll compare the actions taken by your AI player against the actions taken by mine. Your code will also be based on efficiency (i.e. the time/space complexity of your implementation) as well as its adherence to the Java style guide.

To submit, rename your directory hw2_FirstName_LastName. Then zip your directory and upload it to Moodle. Please remember to rename your directory *before* you zip it.

Last modified: Tue Jul 15 13:34:17 PDT 2014