## CS510 HW Assignment 3

**Christopher Geib**

**Due: Feb.14th 11:59pm**

**Part 1.A: Minimax search (75 points)**
In this assignment you will program an agent that uses Minimax to play the game of Othello. You do not need to implement the game of Othello this time if you don't want. You can use the following Java implementation: java-othello-src.zip, which is a very simple implementation of the game of Othello. You can test it by running the file "Test.java", which should run a game where both players generate moves at random.

The Othello implementation is structured as follows:

- OthelloMove.java: this class contains stores a "move" (which player made the move and the coordinates of the move)
- OthelloState.java: this is the core class, which implements most of the functionality of the game. The functions you should be aware of, for implementing minimax are:
  1. public OthelloState(int a_boardSize): the constructor, you can create boards of whatever size you want, as long as it's 2 or bigger (Othello is typically played in a 8x8 board though).
  2. public boolean gameOver(): determines whether the game has finished or not.
  3. public int score(): returns the score (positive means player O is winning, negative means player X is winning).
  4. public List generateMoves(): returns the list of moves for the next player to move.
  5. public List generateMoves(int player): same as before, but you can specify which player you want to generate moves for.
  6. public OthelloState applyMoveCloning(OthelloMove move): creates a new game state that has the result of applying move ‘move'
  7. OthelloPlayer.java: this is an abstract class defining an agent that player Othello. Your agent should be implemented as a class that extends this one.
- OthelloRandomPlayer.java: an example agent that player Othello, but just by choosing moves at random.
- Test.java: an example of how to use all the above classes to play a game of Othello.

Specifically, what we are asking you to do is the following:

- Create a new class that extends OthelloPlayer.
- Within this new class, implement an agent that plays Othello using the standard minimax algorithm, as we studied it in class.
- As the evaluation function, just use the "score" function that is provided to you in the OthelloState class (make sure that your bot can play both as the first or second player).
- Your agent's constructor should accept the depth up to which we want to search.
- To make sure your agent works, make it play against the OthelloRandomPlayer we provide. Your agent should defeat it easily!

**Part 1.B: Monte Carlo Tree Search (25 points)**
Create an agent that plays Othello using Monte Carlo Tree Search. Use the following algorithm:

```
MonteCarloTreeSearch(board,iterations):
  root = createNode(board);
  for i = 0...iterations:
    node = treePolicy(root);
    if (node!=null)
      node2 = defaultPolicy(node);
      Node2Score = score(node2);
      backup(node,Node2Score);
  return action(bestChild(root))
```

Where the given functions do the following:

*createNode(board):* just creates a game tree node from the given board. As explained in class, each node in the game tree stores (at least): its parent, its children, the action that led to this state, the number of times this node has been visited and the average score found so far for this node.

*bestChild(node)*: if the next player to move in node is PLAYER1, it returns the child with the maximum average score, if the next player to move in the node is PLAYER2, then it returns the child with the minimum average score.

*treePolicy(node):* this function does the following:

- If 'node' still has any children that are not in the tree, then it generates one of those children ('newnode'), it adds 'newnode' as a child to 'node', and returns ‘newnode’.
- If 'node' is a terminal node (no actions can be performed). Then it returns “node”
- If 'node' is not a terminal but all its children are in the tree, then: 90% of the times "nodetmp = bestChild(node)", and 10% of the times "nodetmp = [a child of node at random]" (if you are curious, this is called an epsilon-greedy strategy). Then, the function returns "treePolicy(nodetmp)"

*defautPolicy(node):* this function just uses the random agent to select actions at random for each player, until the game is over, and returns the final state of the game.

*score(node2):* returns the score of the game (you can use the built-in score function in the Othello package, you do NOT need to use the complex evaluation function you created for the previous assignment)

*backup(node,score):* increments in 1 the number of times "node" has been visited, and updates the average score in "node" with the value "score". If "node" has a parent, then it calls "backup(node.parent,score)".

Compare the performance of this agent against the previous agents you have created. Try running the Monte Carlo Tree Search agent with enough iterations so that it can play at a decent level (values like 10000, 50000, or even 100000 or 1000000 if your implementation is fast enough!).

**What to Submit**

All homework for this course must be submitted electronically using Bb Vista. Do not e-mail your assignment to a TA or Instructor! If you are having difficulty with your Bb Vista account, you are responsible for resolving these problems with a TA, an Instructor, or someone from IRT, before the assignment it due. It is suggested you complete your work early so that a TA can help you if you have difficulty with this process.

For this assignment, you must submit: Your C/C++/Java/Lisp,/Python/Javascript/... source code, written documentation for your program, and results of your testing. We strongly recommend using a compression utility so that you can compress your files into a single file (with a .zip extension) and just upload it, rather than go through the tedious and error-prone process of uploading each file separately.

**Academic Honesty**

You must compose all program and written material yourself, including answers to book questions. All material taken from outside sources must be appropriately cited. If you need assistance with this aspect of the assignment, see a consultant during consulting hours.