COMP316A. 2015 Artificial Intelligence Techniques and Applications

Assignment 2 - Adversarial Search

This assignment is due by **Thursday April 2, 11.55pm**.

This assignment accounts for one sixth of your internal 316 marks.

1. (40 marks) Minimax search with Alpha beta pruning
   Use the MyTree java program to print a personalized perfect binary game tree. Use your student id number to generate your tree.

   Applying minimax search with alpha-beta pruning, compute, on paper, which move (UP or DOWN) the MAX player should choose. Also record the final value computed for this option, and count how many nodes in the tree the computation could prune (terminal + MIN + MAX nodes). Note that this is a small tree, and that your ID number might give you a bad ordering, such that no pruning is possible.

2. (40/60 marks) Implement a program to compute the next best move for any given Reversi (aka Othello) position based on (alpha-beta pruning) minimax search using iterative deepening search.

   Simply use the difference in stones (black-white) as your heuristic function, plus use +100 for a win of black, and -100 for a win of white. For the rules of the game see Reversi on Wikipedia.

   Your program should read from standard in the specification of a current board plus information on who is next to move, as well as the maximum allowed time in seconds. It should output to standard out the computed move in the format described below (standard in and out are System.in and System.out in Java).

   Expect to read-in a game position plus an indicator for which side is to move and how many seconds you are allowed. ".", "B", "O" are used for "free", "black" player, and "white" player respectively. For instance, the starting position, where "black" is expected to move, and is allowed 60 seconds, would be represented as:

```
- abcdefgh
1 ........
2 ........
3 ........
4 ...OB...
5 ...BO...
6 ........
```

```
7 ........
8 ........
B 60
```

You should output one line with the coordinates of your move, the number of nodes expanded, the max search depth reached, and the minimax estimate, e.g.

```
move c 4 nodes 123345 depth 12 minimax 27
```

One clarification: if a player cannot move, they pass. In that case, return the following:

```
move a -1 nodes 0 depth 0 minimax 0
```

Have a look at TimeOut.java for one way of implementing a time out for your search.
To simplify your move-generation work, here is a slightly inefficient code sample Move.java.

A correct minimax implementation will achieve at most 40 marks. If you also implement alpha-beta pruning correctly on top of the minimax search, than the maximum marks will be 60.

What to submit:

1. written solution for Q1 (show your work)
2. source code for Q2

You may use any of the following languages: Java, C, C++, C#, Python, Ruby, Haskell, Common Lisp, Objective-C, OCaml, Javascript.