CPSC 212: Data Structures and Algorithms

Program 5: Mankalah

In this program you will write parts of a smart Mankalah-playing program by adding minimax search, a better evaluation function, and a time limit cutoff to code that will be given to you. (Source code is linked on the syllabus web page.) On the last day of class, we will have a Mankalah tournament, and the winner will receive a fabulous prize.

You should derive a new player subclass from the given Player abstract class. Your new class should be called myloginPlayer.cs, e.g. abc3Player.cs. It should have an implementation of minimax search to find its next move within the allotted time. The choosemove() function should call the minimax function to find its move. In addition, the class should override the evaluate() function to improve on the evaluate function to consider at least three more factors, such as how many potential go-again moves each side has, captures, numbers of stones on each side, etc. Also override the getImage function to provide an image of you for the tournament. Your player should be complete in one file.

Note that your player should work whether playing on Top or Bottom. Then you can easily modify KalahMatch to use any two players of choice and test them in a match against each other. You can see whether changes to your players improve their play in this way. I will also be able to compile your player with other students' players to run the tournament.

To implement minimax, you can essentially implement the pseudocode in the PowerPoint slide. Note that you will want minimax to return both the best move and the score for that move. You can create a class moveResult for move results with two integer members, move and score, and have Minimax return a moveResult. However, make sure your program doesn't require an additional C# file for the moveResult class or for anything else.

To implement a time limit, do a staged search. That is, choosemove should do a minimax search of depth 1, then depth 2, depth 3, etc. until you are out of time. Modify minimax to return immediately when time expires. Then choosemove can return the best move from the previous, completed minimax search.

Build a DLL by opening the included DLL project, dropping in your player, and building. Rename it something like "JohnSmithPlayer". Also upload a picture for the tournament.

You may submit interesting "cheats" that help your program win by devious means, but also submit an honest program for the tournament.

Alpha-beta pruning (10% extra credit): I'll give you a handout describing this modification to minimax search with move reordering. Under some circumstances, alpha-beta pruning can enable your program to search 50% deeper in a given time, making it a more formidable opponent. This works best if moves that are likely to be very good or very bad are examined first (captures, go-agains).

Turn in your zipped project directory through moodle. Also turn in a DLL for your player and a picture as a second file submission. Also turn in a program grading sheet.

Grading. 50% credit for turning in a program that compiles and runs and implements something of the requirements. An additional 20% for getting minimax search to work correctly, 10% for implementing a better evaluation function, 10% for staged DFS and a 4-second time limit for the search, 10% for alpha-beta pruning, and 10% for good programming style.

CS212 Program 5 – Grading Sheet

Name:_____ Date/time: _____ Is this program late? _____

Parts of the program I didn't get to work correctly:

It took me a long time to get the depth to print properly because on the last move of the game, the depth would increment way beyond where it should have been but the best move and value were not changing.

Comments:

_____[below the line for instructor use only]_____

Submit a program that compiles and runs, including DLL and pic (50%) _____

Minimax search (20%) _____

Staged DFS respecting time limit (10%) _____

Evaluation function measures at least 3 features of the board (10%) _____

Alpha-beta pruning with move reordering (10%) _____

Style (10%) _____

Total: _____