G. Biswas

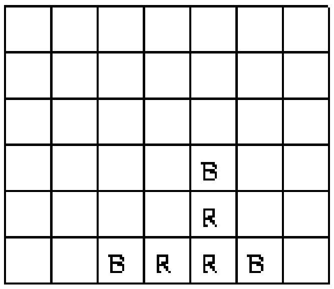
09/17/2014

CS 360: Fall 2014
<u>Programming Assignment 1 – Game Playing</u>

Due: Friday, October 17, 2014 @ midnight.

Background:

The game Connect Four is complex enough to be interesting problem to build an automated player, but simple enough to be well understood. (In fact, Connect Four was solved almost twenty years ago.) It is played on a 6 x 7 grid like the one shown here. You can imagine slots at the top of the grid into which the colored disks are dropped. Disks fall down in a column either until they land on top of another disk, or until they reach the bottom level.



The game configuration shown above would be the result if the following sequence of actions was played:

1. Red drops a disk into the fourth column; it lands at the bottom.
2. Black drops a disk into the third column.
3. Red drops a disk into the fifth column.
4. Black drops a disk into the sixth column.
5. Red drops a disk into the fifth column, landing on the red disk already there.
6. Black drops a disk into the fifth column, landing on the one just dropped by red.

The object of the game is to "connect four" - i.e., to get four of your disks in a row, vertically, horizontally, or diagonally. The game is roughly like generalized tic-tac-toe, with a restriction on the moves that can be made. For instance, you can't choose to start in the center square. That is, the first time a disk is dropped in any column, it must land in the bottom row.

The Assignment:

For this assignment you are to write a program that can play an extended version of **Connect-Four,** let's call it **Connect-Five**. You can assume that you are playing against one other opponent. Your opponent can be either a human or a computer. Your implementation should necessarily have two parts. Part 1 will be an implementation of the minimax algorithm with alpha-beta pruning along with a heuristic evaluation function that you develop. Part 2 will extend part 1 by including strategy and tactical rules to speed up the search process by eliminating undesired moves or by focusing on the more promising moves.

You are given a board consisting of 8 columns and 10 rows. At each turn, one of the two players places his/her piece in a column. It will go into the first free row of that column. You can imagine placing the piece at the top of the column and having it float to the bottom.

The goal of the game is to get five of your pieces in a row (either across a row, down a column, or on a diagonal). The first person to get five in a row wins.

In traditional Connect 4, the board is 7 columns by 6 rows. We deviate from the traditional game in two ways:

1. Our board is larger (10 rows instead of 6; 8 columns instead of 7)
2. To win you connect five of your pieces.
3. We will allow wraparound for the 5 pieces in a row. For example, if player *A* has two pieces in the leftmost columns of row 3 and three pieces in the right most column of row 2, then player *A* wins.

The rules and further details are provided in the two websites below.

Strategy: Strategies and tactics have been developed for playing Connect 4 to the extent that it is a solved game (i.e., the outcome of the game is known if the first player follows a predetermined strategy). The websites linked below have information on strategy and you are welcome to search the web to find other tips on strategies. There is also discussion of a number of evaluation functions that are applied to minimax search algorithms with alpha-beta pruning that you can find on the web. You can study these strategies and evaluation functions and extend them to solve the Connect 5 problem. Note that you can undoubtedly find code online for the traditional version of Connect 4. It is quite likely that code will be more complicated than what you are required to implement for the assignment. While it is OK to look for strategies and evaluation functions online, copying code that you find online will be considered cheating.

General Websites:

Rules of Connect4 and links to strategies

You can play Connect 4 with a computer here.

Additional websites:

Victor Allis' MS Thesis: http://www.informatik.uni-trier.de/~fernau/DSL0607/Masterthesis-Viergewinnt.pdf

Useful Connect Four website: http://gizmodo.com/heres-how-to-win-every-time-at-connect-four-1474572099

What to turn in

The written assignment and programming assignment each will be graded for a maximum of 40 points each. 20 points are reserved for your performance in the tournament. For the final submission, you will use the online submission process to turn in your code and a README file. All code should be working and sufficiently well documented. You will be graded in part based on this documentation.

The README file should include:

1. A description of your approach and any special points that you think we should know about,
2. Details on how to run your code
3. A short paragraph explaining how you implemented each of the following requirements in your program: minimax search with alpha-beta pruning, move generator, and evaluation function.

The written assignment should describe in greater detail

1. The structure of the program
2. A justification of the evaluation function you chose, and an analysis of how well it works.
3. Discussion of the strategy rules you used and justification for these rules.
4. Overall structure of your search algorithm that combines the search algorithm with the strategy rules.

. Be sure to run it on a number of examples to make sure it works correctly. Include at least 3 example runs (i.e., initial configuration and your solution) to demonstrate that your program runs. Once you have all of this done, you can enter your code for the tournament.

Details

Implementation: You are to implement a minimax search using alpha-beta pruning. You should set up your program so that search depth can be a user-provided parameter and you can run your program with different levels of lookahead. You should implement an evaluation function that estimates how good a particular game state is. Consider various strategies, such as the ones given on the web sites, when implementing your evaluation function. You will be graded in part on the creativity you show in implementing your evaluation function. In your program, you will need to keep track of the state space, which should include the layout of the board. You will also need to keep track of pieces in a row; when you make a move that results in 4 pieces in a row, your program should declare that you are the winner. Finally, if you make an illegal move, then you lose the game.

In the second part, you will add strategy and tactical rules that will work in conjunction with the search algorithm to speed up search and also generate better moves.

Tournament: Following the due date, there will be a tournament in which your programs will be pitted against each other. There will be an extra 20 points that will be allocated based on your performance in the tournament. Each program will play every other program twice: once as red (player A), and once as black (player B).

Interface: Since this is a two-player game, you will need to be able to pit your program against a human player (for development and testing) and against other computer players for the tournament. To facilitate this, we are providing a module that you will need to interface with. This module will maintain the state of the game, make your moves, report your opponent's moves to you and provide a GUI for the game.

Details of the tournament will be provided soon.