CSC 242 Project 1 Writeup

Part 1: 3 x 3 board

Part 2: 9 x 9 board

main() makes an instance of the problem class and then initializes the game using startGame(). startGame() asks the user if they want to start first. If their response is ‘y’ then the following functions are executed sequentially until goal state is found: humanMove(), aiMove() and displayBoard(). A response of ‘n’ would run the sequence aiMove(), humanMove() and displayBoard().

humanMove() first checks if the human agent has a choice as to what board she should play on. If board has the value of 0 in humanMove(), then the human agent is free to choose the board position and is therefore prompted to choose one. If board has any other value, then that value stored in board refers to the board where the human agent is forced to play, in which case the user is not prompted to choose a board.

aiMove() first sets the depth of the activeState to 0. This is because we are implementing a depth limited search (depth is 10). So each time hMinimaxDecision() is called from aiMove(), we make sure that the depth of the state we’re starting hMinimaxDecision() on is 0. hMinimimaxDecision() returns the best action, so we set the new activeState to result(activeState, hMinimaxDecision(activeState)).

The result function does four things:

1. Adds the move on the board.
2. Changes the activePlayer from “O” to “X”, or from “X” to “O”
3. Sets the cost of the successor state one more than the cost of previous state
4. Sets the depth of the successor state one more than the cost of previous state

Effects of Alpha beta pruning