Artificial Intelligence Project#2

Connect-Four Game
Report

Rıdvan Gülcü – 150117508
Berre Ergün – 150117507

Connect-Four Game

In our Connect-Four Game we did used 3 different heuristic methods and a ply number that can be change for AI player. We used minimax algorithm with alpha-beta pruning.

How to run:

java demoConnect4 -g “gameType” -p1 “heuristicFunction” “depth" -p2 “heuristicFunction” “depth”

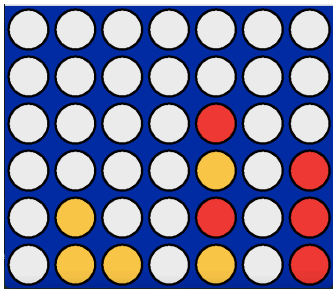
gameType: 1 for Human vs Human , 2 for Human vs Ai and 3 for Ai vs Ai
heuristicFunction : h1 for heuristic1 , h2 for heuristic3 and h3 for heuristic3
depth: Integer between 1 and 8

Example Runs:
Human vs Human -> java demoConnect4 -g 1
Human vs AI ->java demoConnect4 -g 2 -p2 h2 8
AI vs AI -> java demoConnect4 -g 3 -p1 h3 8 -p2 h2 8

P.S : You should press enter to perform AI movement.

Heuristic Function 1

Our first heuristic method is the simplest and less complex one. It simply does two checks: horizontal and vertical. First it looks for horizontal. In every row it counts the number of pieces that connected. State gets 500 points for every connected 3 pieces and 50 points for 2 connected pieces. And then function counts for vertically connected pieces. If 3 pieces are connected and if that column is not full(means 4 pieces can be connected) state gets 500 points, and for every connected 2 pieces it gets 50 points(again with 2 empty place above them). An example state and its point is given below.



For the red player, state gets 500 points from column 7, but cant get any points from column 5 since these 2 vertically connected pieces can not be fulled to 4.

Heuristic Function 2

Our second heuristic is quite simple but powerful. For every place in the board we have a point according to value of that places. Values are the number of possible 4-connected. The values are the numbers of possible 4-connected pieces from that place. This heuristic counts every pieces value. Every place's value is given below.

$$\begin{bmatrix} 3 & 4 & 5 & 7 & 5 & 4 & 3 \\ 4 & 6 & 8 & 10 & 8 & 6 & 4 \\ 5 & 8 & 11 & 13 & 11 & 8 & 5 \\ 5 & 8 & 11 & 13 & 11 & 8 & 5 \\ 4 & 6 & 8 & 10 & 8 & 6 & 4 \\ 3 & 4 & 5 & 7 & 5 & 4 & 3 \end{bmatrix}$$

Heuristic Function 3

For calculate score related to heuristic3 we consider we check each possible 4-connected cells with 4 sized blocks and increment one if cell values is equal to our value. For check all possible 4-connected we used 4 for lop. One is perform vertical checks , other one is perform horizontal and two of them check diagonal. Diagonal check need 2 different for because their directions may be different.

Actually this will be give us same score with heuristic2 so far. But when we checks possible 4-connected cells we also count the neighbor have same colors. After determine neighbor of colors we add some specific points related to neighborhood.

Some possible case represent below:

3 neighbor color and other neighbor is free : +50 points
3 neighbor color and other neighbor is free : + 20 points

2 neighbor color and other two neighbors are free : +10 points
2 neighbor color and one of neighbor is free : +3 points
2 neighbor color and there is no free neighbor: +1 points

Our program contains 6 classes, their names and methods are given below.

Node Class

Node class is for the states. It has a string 2D array which name is state. In this class we calculate states score according to the player and the heuristic function.

isGameOver: This method gets 2 parameters : depth and score. And if the score is equal to the winning score(100000) or the opposite(if the opponent player is winning), if board is full or depth is equal to zero, method returns true. Else it returns false.
calculateScore: This method gets player, row and column index. Checks four place after that indexes and controls if the player has 4 connected pieces vertical, horizontal or vertically. If player has 4-connected pieces that method returns win score(100000). And at the same time it counts for opponent player, IF opponent player has 4-connected pieces method returns –win score.
getScoreHeuristic1: This methods checks if player wins or loses in that state, else it calculates the score according to the heuristic 1 with using checkHorizontal and checkVertical methods.
getScoreHeuristic2: This methods checks if player wins or loses in that state, else it calculates the score according to the heuristic 2.
getScoreHeuristic3: This methods calculates the score according to the heuristic 3 and checks if player wins or loses at the same time.
isBoardFull: Looks every place in that state and if state is full returns true, else false.

Utils Class
Utils class contains some methods to print board, deep copy for state and getting the reverse player.

Player Class
Player class is to create player object. Class keeps label of the player, heuristic function, depth and name of the player.

Constant Class
This class keeps the final values like row size, column size, winning score(10000000) and the initial board.

Connect4 Class

generateAiDecision: generate and perform adi movement with minimax algorithm.
maxPlay: maximum Player for minimax tree algorithm.
minPlay: minimum player for minimax tree algorithm.
move: take state player and colChoice and perform movement related to colChoice. Return the row index of added piece. If column is full return -1.