CSE4082 – Artificial Intelligence

Project 2 Connect-Four Game Report

Malik Türkoğlu

Introduction

This document explains how Connect-Four game is implemented. Methods and classes will be laid out concisely.

This document is strictly bonded to the project document. There may be parts without explanation, which are coming directly from the project document. α-β pruning is not implemented. There was a class AI, which is not included here since we have used it for testing purposes.

The maximum ply number is 8, minimax expands 8 level down. The time required for expanding is around 5-5.5 seconds. All are presented with screenshots.

Heuristics

- H1: This heuristic is the simple one. It uses minimax but its approach is chancebased. Nevertheless, it wins rarely.
- H2: This heuristic has an approach such that having a line of pieces brings points. For example, if two pieces are side by side, player earns 20 points; if three, then 50 points; if 4, game-winning condition, then it earns 1000 points. Additionally, blocking other player from winning brings 90 points.
- H3: This heuristic is improved version of H2. It does the same thing but additionally it checks for solution-producibility potential. For example, say there is a double along with fifth row but AI cannot add two more to it to win. Then the AI ignores this line and looks for another place to play since this double does not have solution-producibility potential- it is a dead end. If there is no place that can produce a solution, then H3 acts as H2.

Classes and Methods

1. ConnectFour Class

This is the class where the game is implemented. Methods and attributes are explained below.

Attributes

- static int[][] gameTable: This is the game table. It holds the current status of the game.
- int rowAmount: number of columns. - int columnAmount: number of rows.

Methods

- startGame: It prints the menu and prompts the user to interact with the system.
- playHumanToHuman: It enables two human players play with each other. It works simple; take input and update the game table.
- playHumanToAi: It enables the user to challenge our best AI player which uses H3.
- playAiToAi13: This method lets the AI player with H1 competes with the AI player with H3.

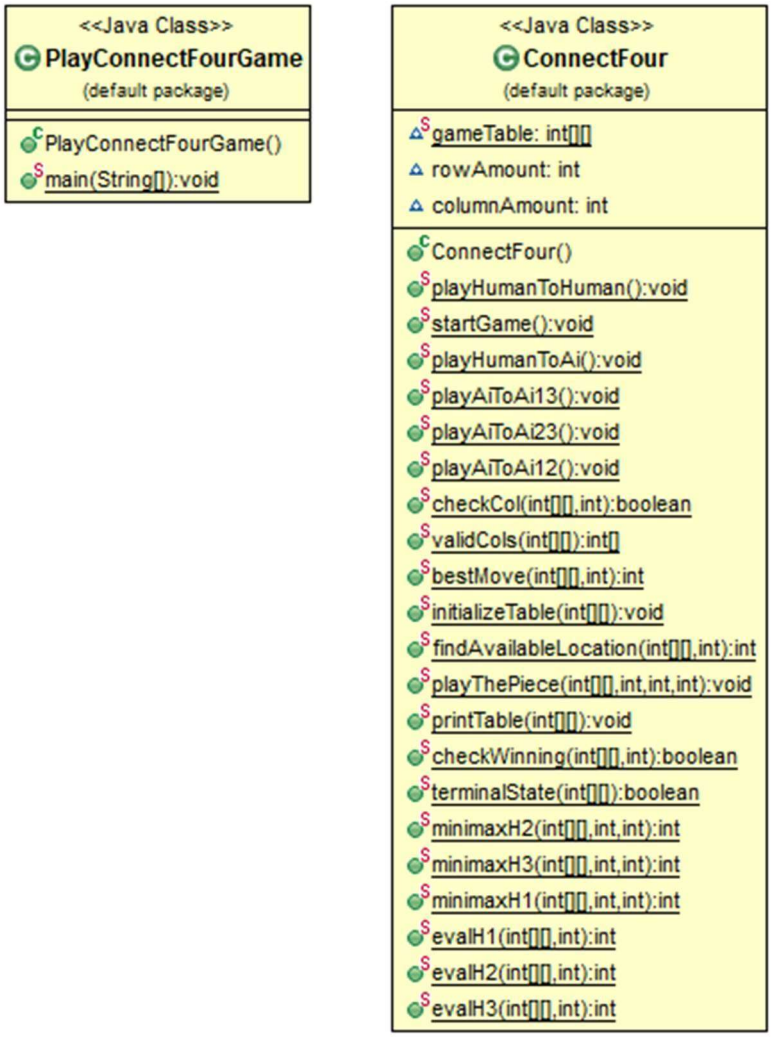
- playAiToAi23: This method lets the AI player with H2 competes with the AI player with H3.
- playAiToAi12: This method lets the AI player with H1 competes with the AI player with H2.
- checkCol: Checks the location if it is valid or not e.g., inside the borders of the game table.
- validCols: Returns all valid locations' column numbers.
- initializeTable: Builds the game table with all empty locations.
- findAvaliableLocation: Returns the empty place at the lowest level of the column.
- playThePiece: Puts the piece on the table.
- printTable: Prints the current status of the table.
- checkWinning: Checks if there is winner.
- evalH1: Implements H1 which is about playing randomly.
- evalH2: Implements H2. The implementation is done by searching. The method traverses whole game table and try to finds followings; 1) Winning condition, 2) the other player's winning condition, 3) line with three/two pieces side by side, on top of each other or diagonal. According
  to the result of searching, it assigns necessary points to each case and returns the score.
- evalH3: It implements H3. As an addition to H2, it also checks whether a move can produce a solution. It finds that by simply checking the rest of the line. For example, if there is a line with two pieces, this method looks further two locations and checks whether they are empty or not. If locations are not empty, then moves to another location to check. If they are empty, then it chooses that location immediately. In the case of not finding any such location, then it simply implements evalH2 to behave as H2.
- minimaxH1: It implements minimax algorithm with H1 as an evaluation function. It punishes by 500000 in the case of losing and rewards with 5000000 in the case of winning.

- minimaxH2: It implements minimax algorithm with H2 as an evaluation function. Punishing and rewarding policies are the same as minimaxH1.
- minimaxH3: It implements minimax algorithm with H3 as an evaluation function. Punishing and rewarding policies are the same as minimaxH1.

2. PlayConnectFourGame Class

It is a very simple class that includes main method to start the game.

The class diagram is given below.

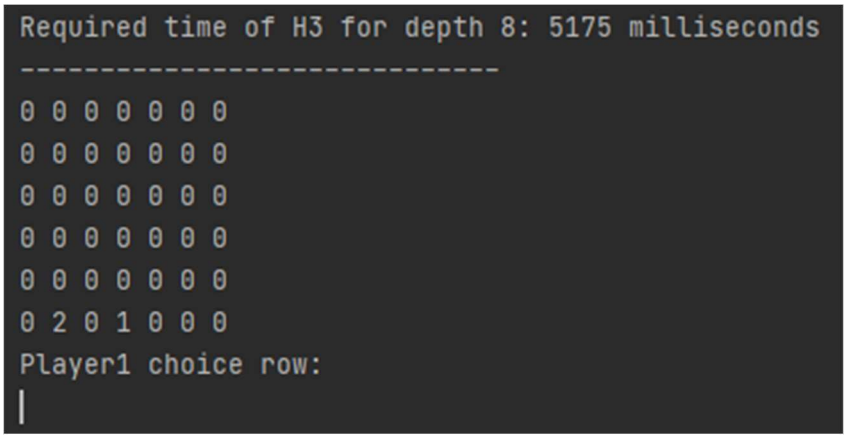


Sample Outputs

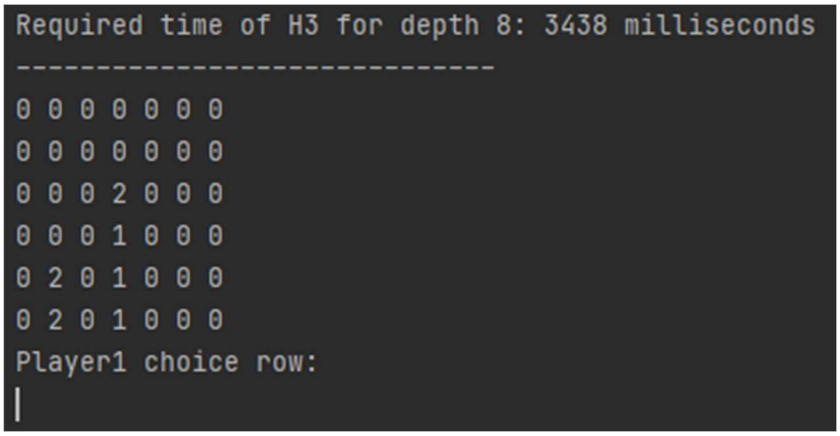
Starting menu

```
Welcome! please select the operation:
1. Human to Human
2. Human to Ai(H3)
3. Ai(h1) to Ai(h3)
4. Ai(h2) to Ai(h3)
5. Ai(h1) to Ai(h2)
Your Choice:
```

Human to Ai (H3) – Required time for maximum depth: 5175 milliseconds

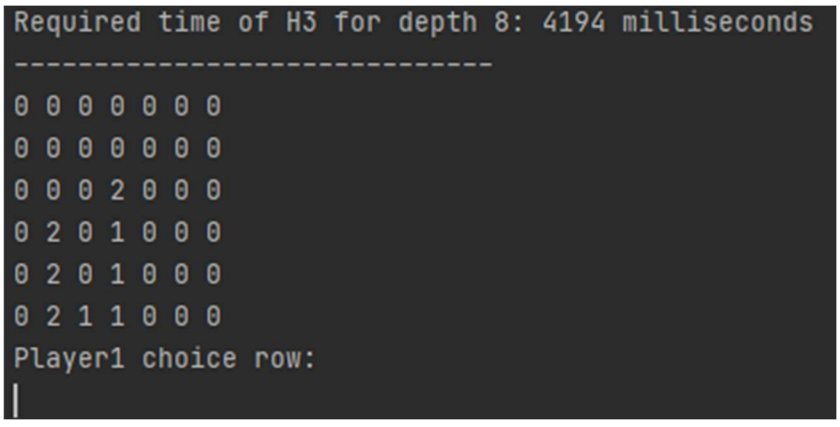


Human to Ai (H3) – AI can block human from winning: The middle column



Required time is now lesser. Since it catches my intent and blocks me immediately without expanding further.

Human to Ai (H3) – AI tries to win: The second column



Human to Ai (H3) – AI can catch diagonals

```
Required time of H3 for depth 8: 2181 milliseconds
-----------------------------
0 0 0 0 0 0 0
0 0 0 1 0 0 0
0 1 0 2 0 0 0
0 2 1 1 0 0 0
0 2 2 1 0 0 0
2 2 1 1 2 0 0
Player1 choice row:
```

It blocked me from winning. Again, it took much lesser time to decide.

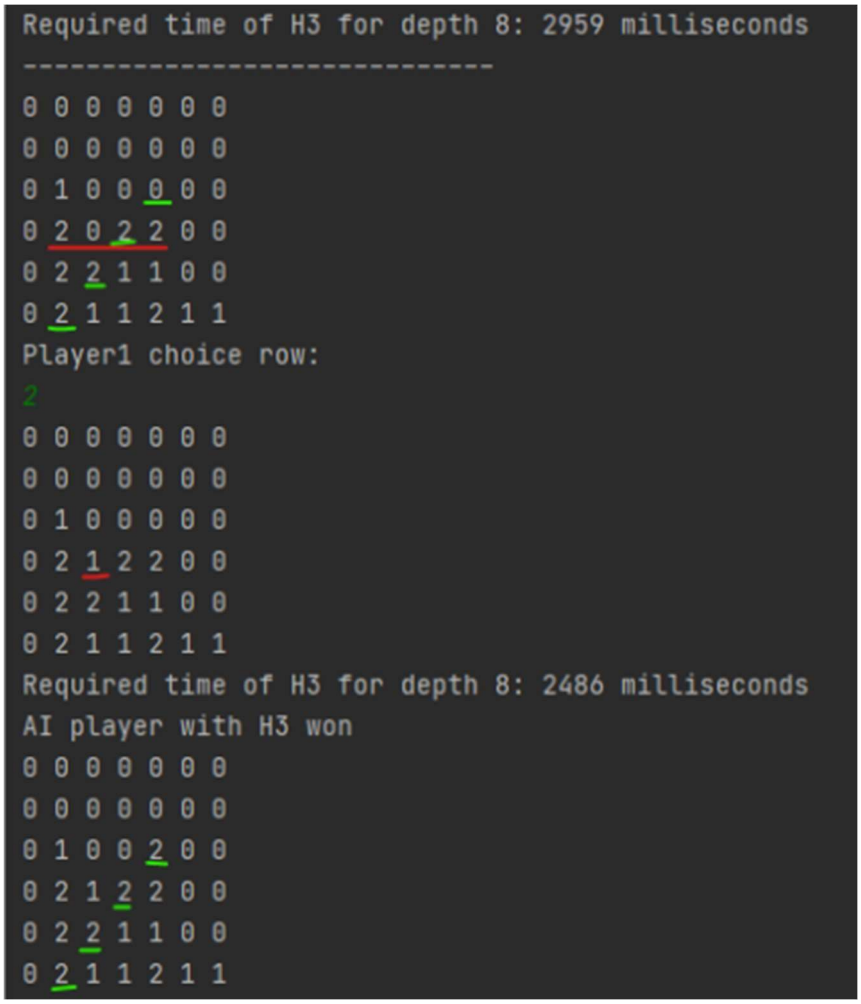
Human to Ai (H3) – AI can catch diagonals.

```
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 1 0 0 0 0 0
0 2 0 2 0 0 0
0 2 0 1 0 0 0
0 2 1 1 0 1 0
Required time of H3 for depth 8: 3218 milliseconds
-----------------------------
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 1 0 0 0 0 0
0 2 0 2 0 0 0
0 2 0 1 0 0 0
0 2 1 1 2 1 0
Player1 choice row:
```

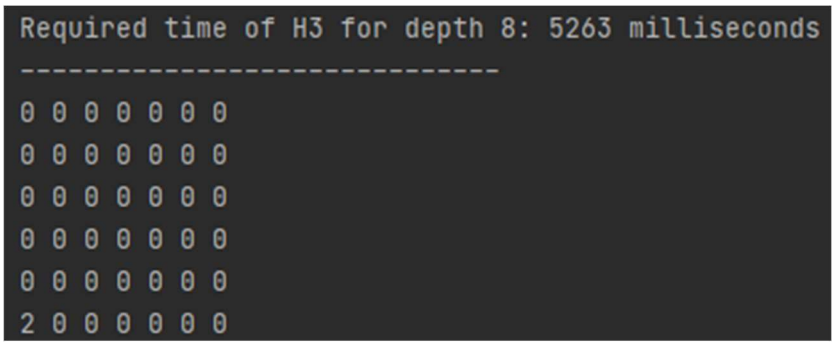
Human to Ai (H3) – AI is faster as the game approaches the end: 0 milliseconds.

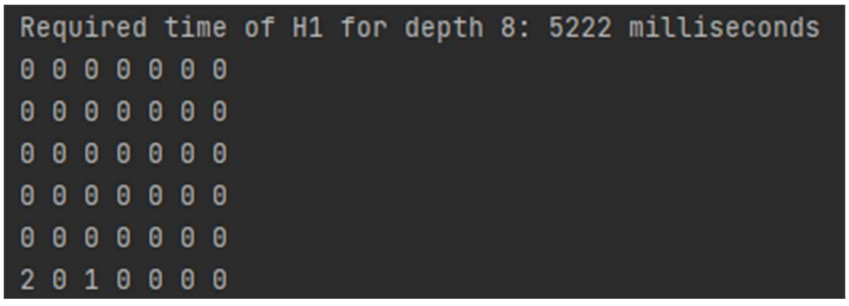
```
Required time of H3 for depth 8: 0 milliseconds
-----------------------------
1 2 2 2 1 0 0
2 2 1 1 2 0 0
2 1 1 2 2 0 0
2 2 1 1 1 2 1
1 2 2 1 2 1 1
2 2 1 1 2 1 1
Player1 choice row:
```

Human to Ai(H3) – AI won. It was inevitable for me to escape from loss.

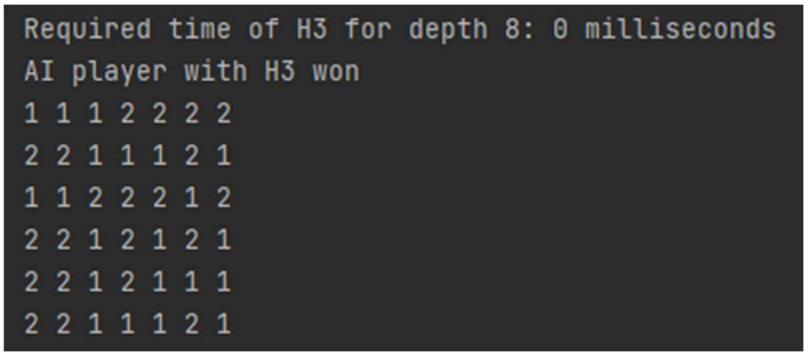


Ai(H1) to Ai(H3) – Required times for maximum ply: 5263 milliseconds for H3 and 5222 milliseconds for H1.

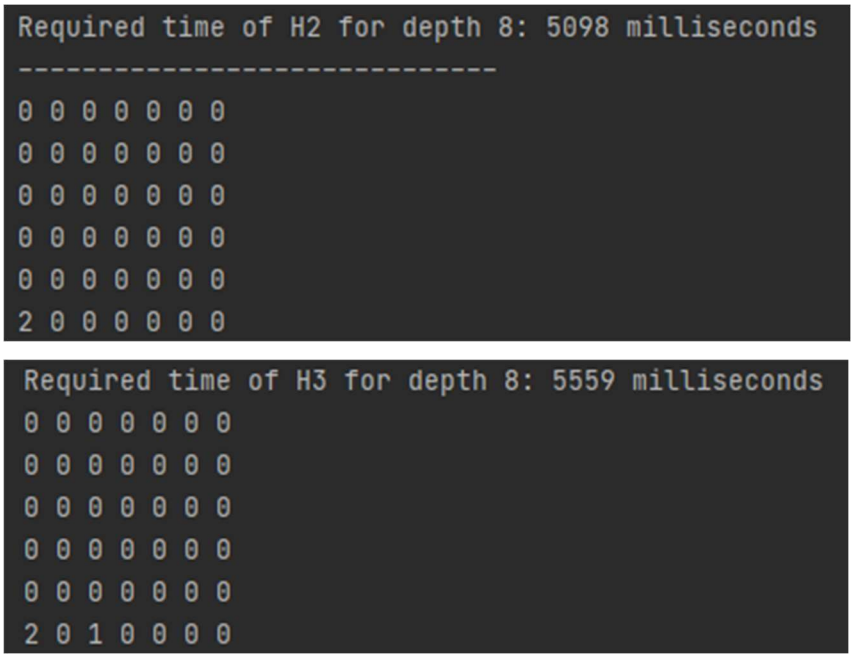




Ai(H1) to Ai(H3) – AI with H3 has won the game.



Ai(H2) to Ai(H3) – Required times for maximum ply: 5098 milliseconds for H2, 5559 milliseconds for H3.



Ai(H1) to Ai(H2) - Required times for maximum ply: 5175 milliseconds for H2, 5002 milliseconds for H1.

```
Required time of H2 for depth 8: 5175 milliseconds
------------------------------
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 2 0 0 0 0
Required time of H1 for depth 8: 5002 milliseconds
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 2 1 0 0 0
```

Conclusion

Once we let AI players compete each other, it is observed that as the depth level of the minimax algorithm increases, the AI player with H3 is almost unbeatable. For instance, at depth level 7, AI player with H3 becomes a very powerful player. On the contrary, at lower depth levels such as 3, other AI players with H1 and H2 can beat the player with H3.

As the game approaches to the end, the time required for expansion decreases. In the case of winning or preventing the opponent from winning, the required time is lesser.