Using Minimax Algorithm to Make Connect 4 AI Agent

Abdullah Elsayed Ahmed 7459
Mohammad Ashraf Hamdy 7508
Samah Abdelaziz Draz 7889

March 20, 2024

Contents

1 Introduction

This is a basic connect4 game the player with more connected stack (Columns, Rows, Diagonals) will win the game. We used Optimized AI agent to play against human so that the AI agent will win most of the time.

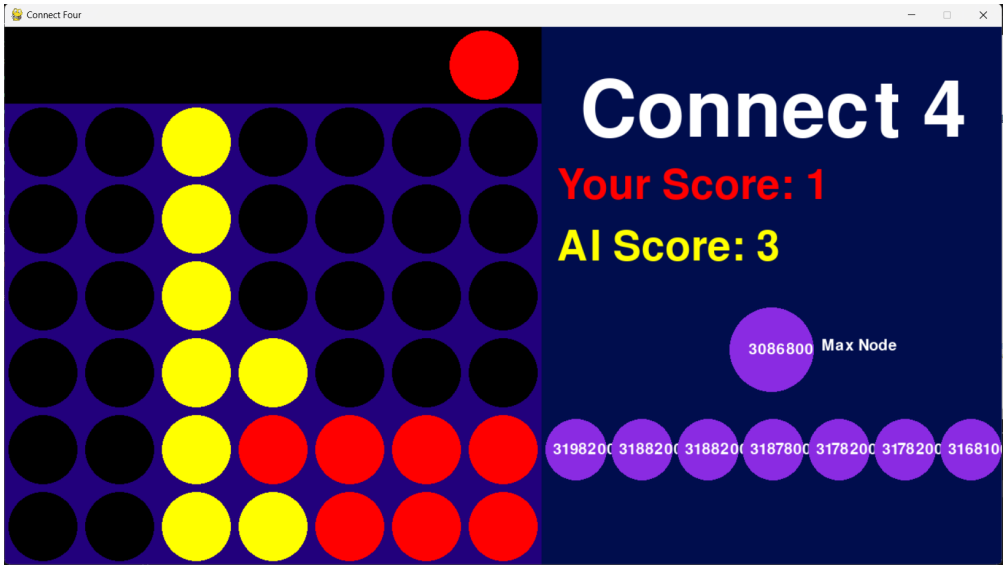


Figure 1: Connect4 game GUI

2 Game rules

Board size

Board size can be an where above or equal 7 in width and 6 in height.

Evaluation

Each player has it own score based on number of 4 connected piece in the same row, column and diagonals the more connected 4 gain more score.
Example: 1,1,1,1,1,1,1. will be considered as **3 points** for player 1.
The player with higher score win the game.

Valid move

The player can choose any column in the board but, It only legal if and only if the column isn't full.

Termination

Game terminate if and only if there is no more available spaces.

3 GUI

We made a sample user interface for the game using **pygame**, which enable the human to select the column by hovering above it and click with the mouse to drop the piece and after short time the AI response and the game goes on until the board if full.
The GUI also show the score for Ai agent and human and also corresponding minimax trees for each move.

4 AI agent

From mentioned rules We build our AI agent to fit those rules and try to get the max score to win.

Algorithm

We used Minimax as our main algorithm, It is an decision-making algorithm, It have been used in our code to play against human in Connect4 game. but We faced many challenges due to the full game tree is very huge $O(10^{35})$ so, We cuts the minimax tree at deeps K chosen before the game start.
We used different minimax algorithm

- Normal minimax.
- Pruning minimax.
- Expectation minimax.

and We will talk about the different between them latter in the report.

Scoring

Because We cut the minimax tree before termination state, We can't know the final score so, We had to choose a suitable heuristic algorithm to return a reasonable score.
A reasonable score tend to be high score if our AI agent will gain more points from a move and lower score from another move.
So our scoring will be based on the following:

Number of connected piece

[H] We give the AI agent more score for the move which will have more consecutive pieces.

- 4 connected: 1,000,000
- 3 connected: 40,000
- 2 connected: 10,000

Weighted center

The player with more pieces at the center will a high change of winning so, We make a 2D matrix which return higher score for the player with more pieces at the middle and less farther away.

300	400	500	700	500	400	300
400	600	800	1000	800	600	400
500	800	1100	1300	1100	800	500
500	800	1100	1300	1100	800	500
400	600	800	1000	800	600	400
300	400	500	700	500	400	300

Data structure

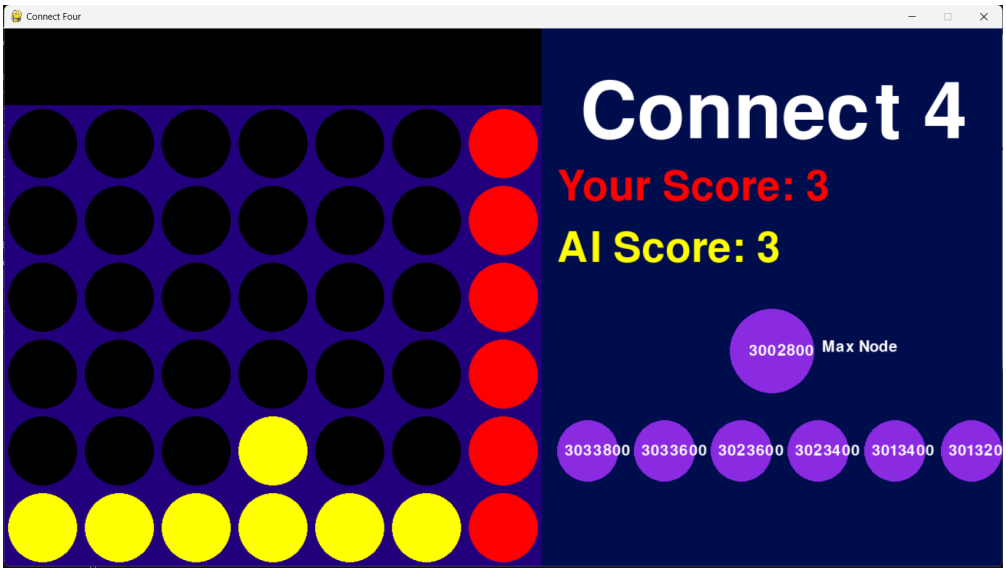
We used **2D array** to represent our game board and used **hash table** to store previous board scores.

5 Results

those are the result of our code each run shows the game board and corresponding minimax trees.

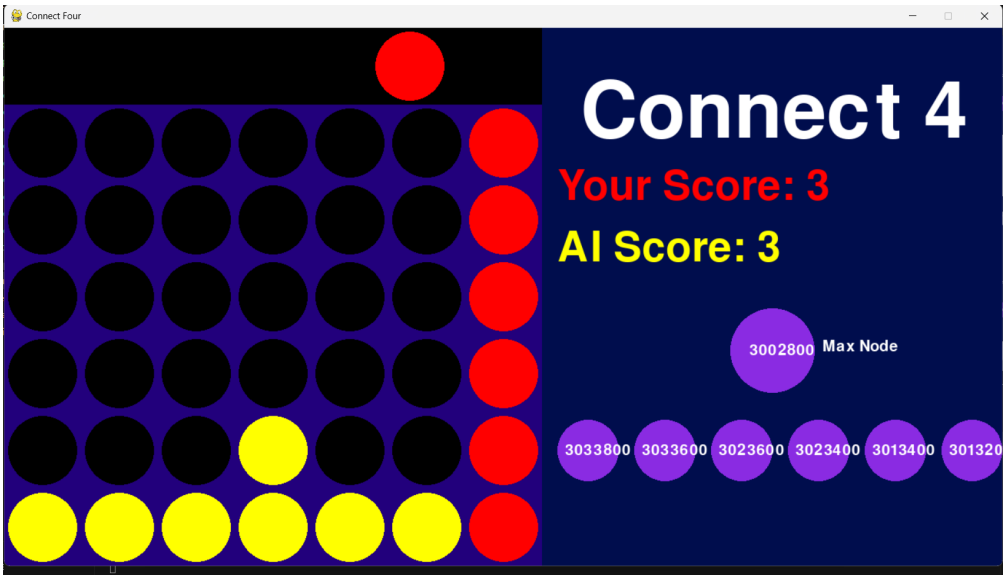
at k = 1

Normal minimax



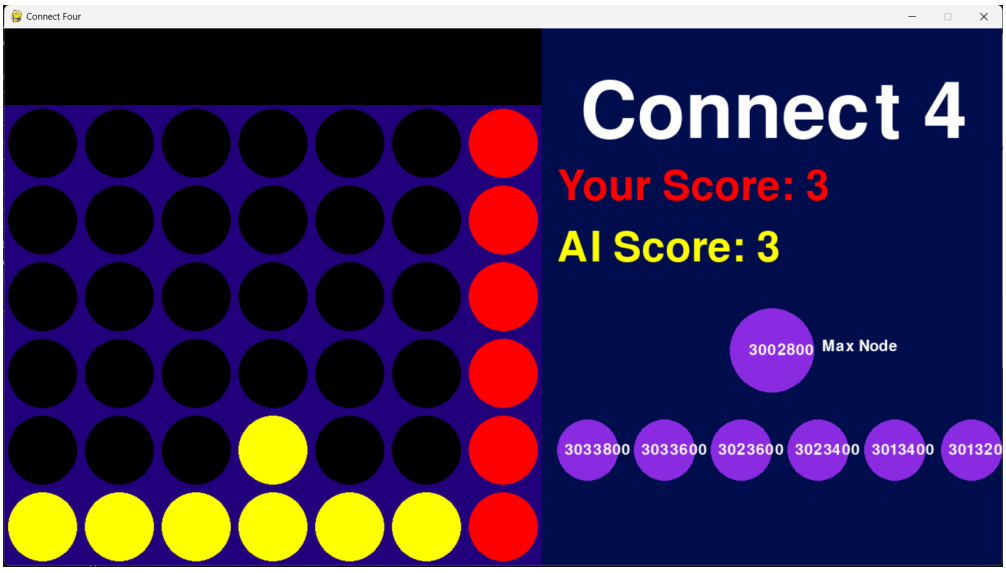
- Nodes: 48
- Time: 0.004051923751831055

Pruning minimax



- Nodes: 48
- Time: 0.0041010379791259766

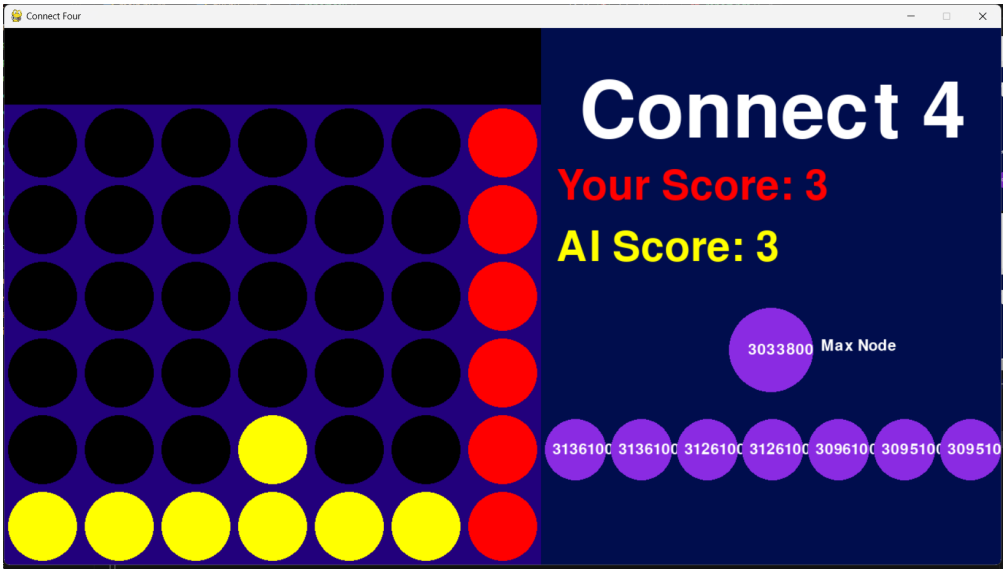
Expectation minimax



- Nodes: 48
- Time: 0.004051923751831055

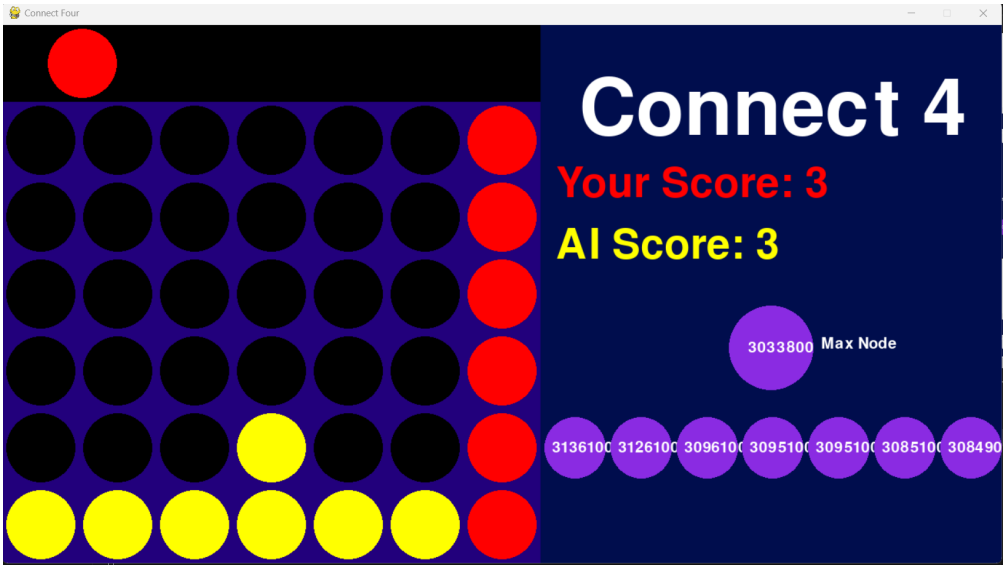
at k = 3

Normal minimax



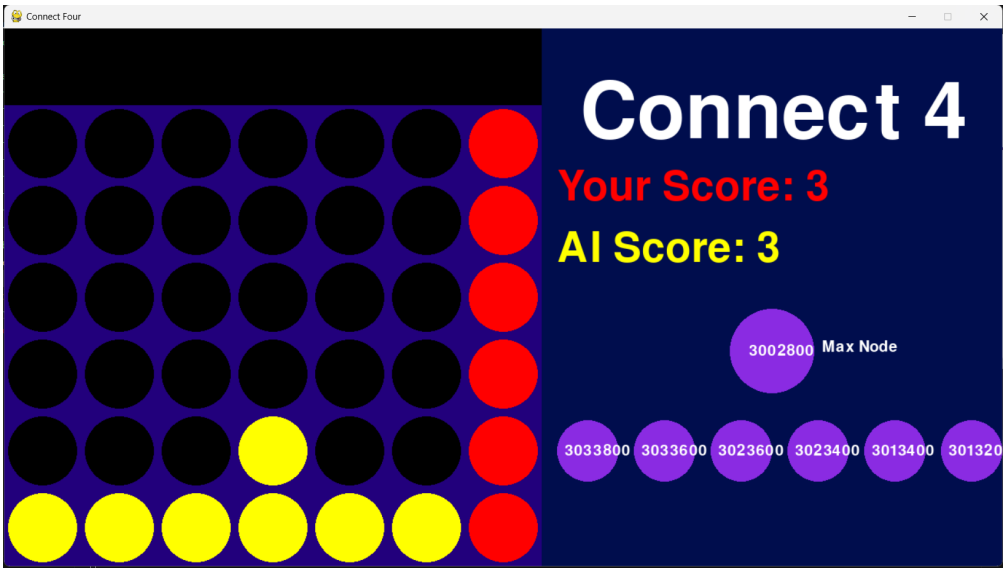
- Nodes: 2631
- Time: 0.12168431282043457

Pruning minimax



- Nodes: 1056
- Time: 0.04659581184387207

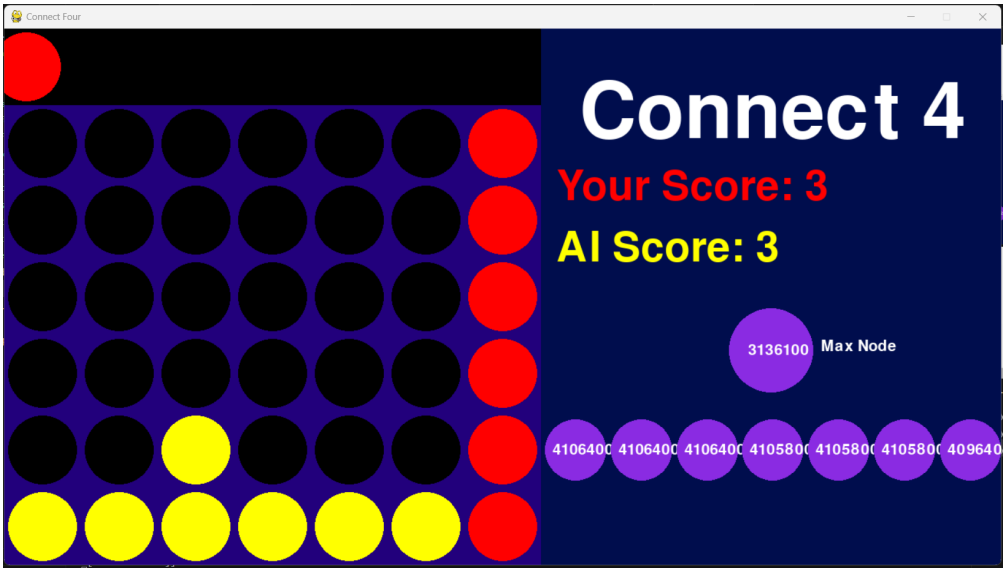
Expectation minimax



- Nodes: 1056
- Time: 0.04659581184387207

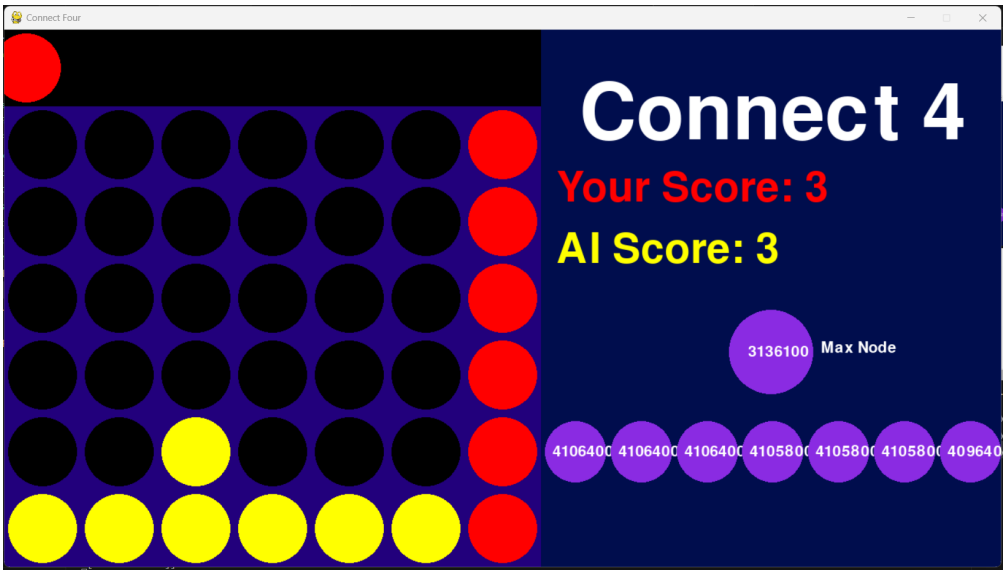
at k = 5

Normal minimax



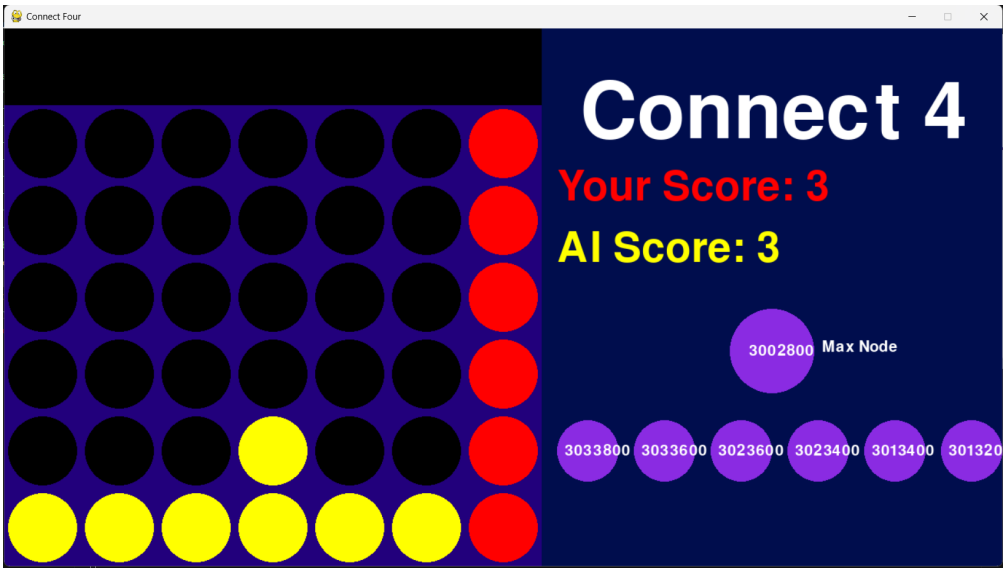
- Nodes: 62621
- Time: 1.756775140762329

Pruning minimax



- Nodes: 11710
- Time: 0.6393032073974609

Expectation minimax



- Nodes: 11710
- Time: 0.6393032073974609

6 Conclusion

From our result we can compare different minimax approach at different K values.

In General

Normal minimax

It is the worst of them because It has to check losing move also it is not important no more (due to a better move is available).

Pruning minimax

It is better version of normal minimax (minimax without alpha beta Pruning) due to the pruning process.

Expectation minimax

It is slightly better then Pruning minimax.