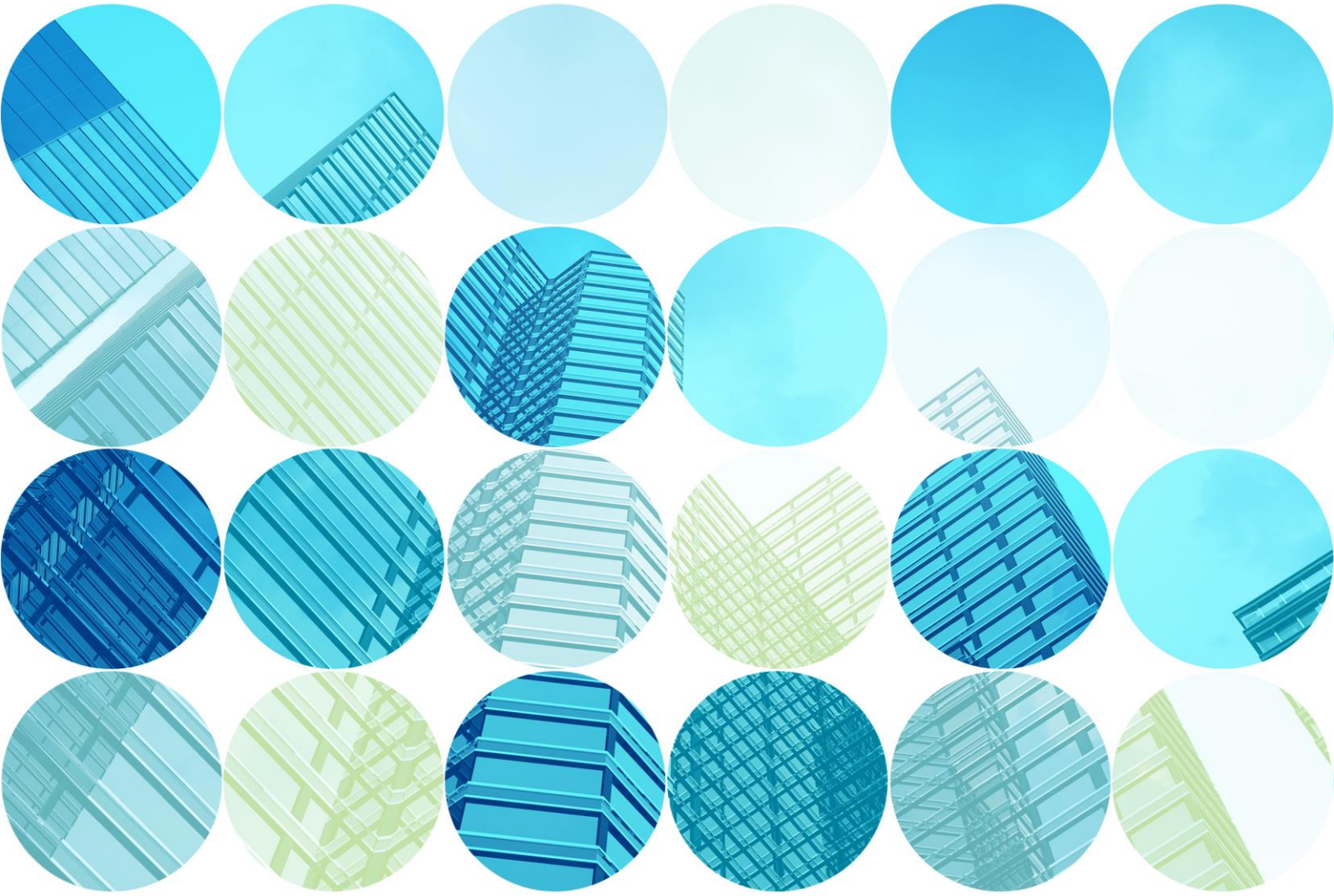
Introduction to
Artificial intelligence

جامعة الإسكندرية
ALEXANDRIA
UNIVERSITY
كلية الحاسبات وعلوم البيانات

ASSIGNMENT 2

Connect 4 Game

with Minimax Algorithm



Team Members:

Abdelrahman Raslan 20221460102

Rewan Salah 20221447143

1. Problem

Using Minimax Algorithm to Make Connect 4 AI Agent, the user chooses who play first and AI vs AI mode

---

2. Program Description

Connect 4 is a two-player game in which the players first choose a color and then take turns dropping their colored discs from the top into a grid. The pieces fall straight down, occupying the next variable space within the column. The objective of the game is to connect four of one's own discs of the same color next to each other vertically, horizontally, or diagonally. The two players keep playing until the board is full. The winner is the player with having a greater number of connected fours.

---

3. Algorithms

a) Minimax without alpha-beta pruning

- Mini-max algorithm is a recursive or backtracking algorithm that is used in decision-making and game theory. It provides an optimal move for the player assuming that the opponent is also playing optimally.

- Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.

```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, minimax(child, depth - 1, FALSE))
        return value
    else (* minimizing player *)
        value := +∞
        for each child of node do
            value := min(value, minimax(child, depth - 1, TRUE))
        return value
```

- In this algorithm two players play the game; one is called MAX and the other is called MIN
- Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit

b) Minimax with alpha-beta pruning

- Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree.Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning. This involves two threshold parameters Alpha and beta for future expansion, so it is called alpha-beta pruning. It is also called as Alpha-Beta Algorithm . Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.

```
function alphabeta(node, depth, α, β, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, alphabeta(child, depth − 1, α, β, FALSE))
            α := max(α, value)
            if α ≥ β then
                break (* β cut-off *)
        return value
    else
        value := +∞
        for each child of node do
            value := min(value, alphabeta(child, depth − 1, α, β, TRUE))
            β := min(β, value)
            if β ≤ α then
                break (* α cut-off *)
        return value
```

- The two-parameter can be defined as:
    - Alpha: The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -∞.
    - Beta: The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +∞.

---

4. Libraries used

- import java.util.Objects;
- import java.util.Scanner;
- import java.awt.*;
- import java.awt.geom.*;
- import javax.swing.*;
- import javax.swing.border.LineBorder;
- import java.awt.event.ActionEvent;
- import java.awt.event.ActionListener;

---

5. Use Data Structures

- Arrays
- Trees

---

6. Contained classes:

a. AlphaBeta

- Alpha Beta Algorithm
- Declare min - max value and max level(depth) number, pruned value .
- Minimax: method to detect which opponent then Calculating score of the board, check if the board is empty or not if it isn't empty, it updates the score by calculating (connecting four current-connects four of an opponent) *max level. return score when achieving a max level.
    - If max will play (AI) then iterate on all columns to show columns are full and show if it to the current player + value and if it to opponent it prints – value on the border, Call minimax recursively and return the maximum to best. Undo the move if it is minimum to the current player, then calculate best, alpha then if beta smaller than or equal to alpha then pruning it
    - If man will play (Human), iterate on all columns, check columns are full, Call minimax recursively and return the maximum to best. Undo the move if it is minimum to the current player, then calculate best, alpha then if beta smaller than or equal to alpha then pruning it
- PlayBestMove: this method will return the best possible next move for the current board and apply the minimax algorithm to get the best move, then apply the minimax algorithm to that possible move. If the value of the next state is more than the best value, then update best. Finally, it calculates the time of applied the algorithm.

b. Minimax

- Implementation of Minmax Algorithm.
- Declare max depth and number of nodes
- Minimax(): this method Set the player, Calculating the score of the board and Terminal States by checking if the board is empty or not then calculating score of the state
    1. if max will play (AI) then iterate on all columns, check if columns are full or not then Call minimax recursively and return the maximum to best, Undo the move if it will decrease the score, If this minimizer's move then Call minimax recursively and return the minimum to best
- playBestMove: method will return the best possible next move for the current board by Applying the minimax algorithm to get the best move and If the value of the next state is more than the best value, then update best.

c. Main

- Show all play modes and ask user to chose one

d. AI_VS_AI

- This class contain AI vs AI mode
- Frist ask for depth of both agent
- Then play and show board and depth
- After that AI agent 1 plays best move and AI agent 2 plays the best move after that
- At the end show who is the winner

e. AI_VS_Human

- This class contain AI vs Human mode that AI start the first
- Ask human after AI plays to the column he want to play in
- Show board and depth
- At the end show who is the winner

f. Human_VS_AI

- This class contain Human vs AI mode that Human start the first
- Ask human the column he want to play in at the first
- Show board and depth
- At the end show who is the winner

g. Board

- Run GUI in main Function
- Declare the rows and the columns
- Board (): constructor to make a board's rows and columns
- addPiece (): add the player's move and check if the columns are full or not to add this piece in these columns
- Printboard (): method to display the board with the move
- getScore(): method to calculate the score by checking vertically , horizontally, and diagonal
- isBoardFull(): method to check if the board is full or not
- countFours() : method to calculate how each player types counts four.
- checkWinner(): method to check the winner
- undoMove(): method to undo moves
- setPlayer(): method to set player type
- calculateScore():method to calculate the score

h. Piece

- It declares and sets a piece 's player type as if it is ai or human, gets the value of the piece's state and sets a new value of the state.

i. ClientGUI

- Run GUI in the main Function

j. CreateRoundButton

- It makes the button of the board a circle shape.
- CreateRoundButton(): constructor to set and calculate dimension size as length and width of the button
- paintComponent, paintBorder: Methods to set the color and size of the button
- contains: Method to set the shape.

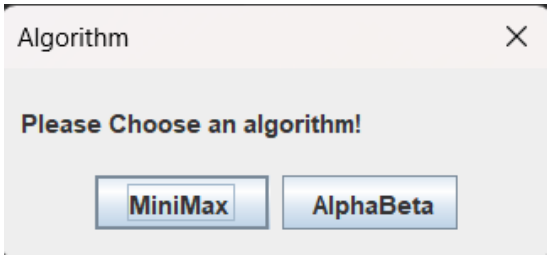
k. Game_GUI

- The interface that appears to the user as we make an Option Pane, the o user to choose the algorithm he wants the show input dialog to allow the user to write the depth as much as the user writes big number the game becomes more difficult. Then the board appears .it is so friendly used. If the player play in filled columns ,the information message will appear This column is full when the board is full , a message will appear which player is win and the score of connects 4.

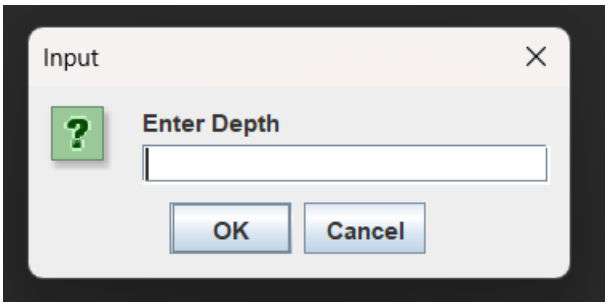
---

7. Simple Run with GUI

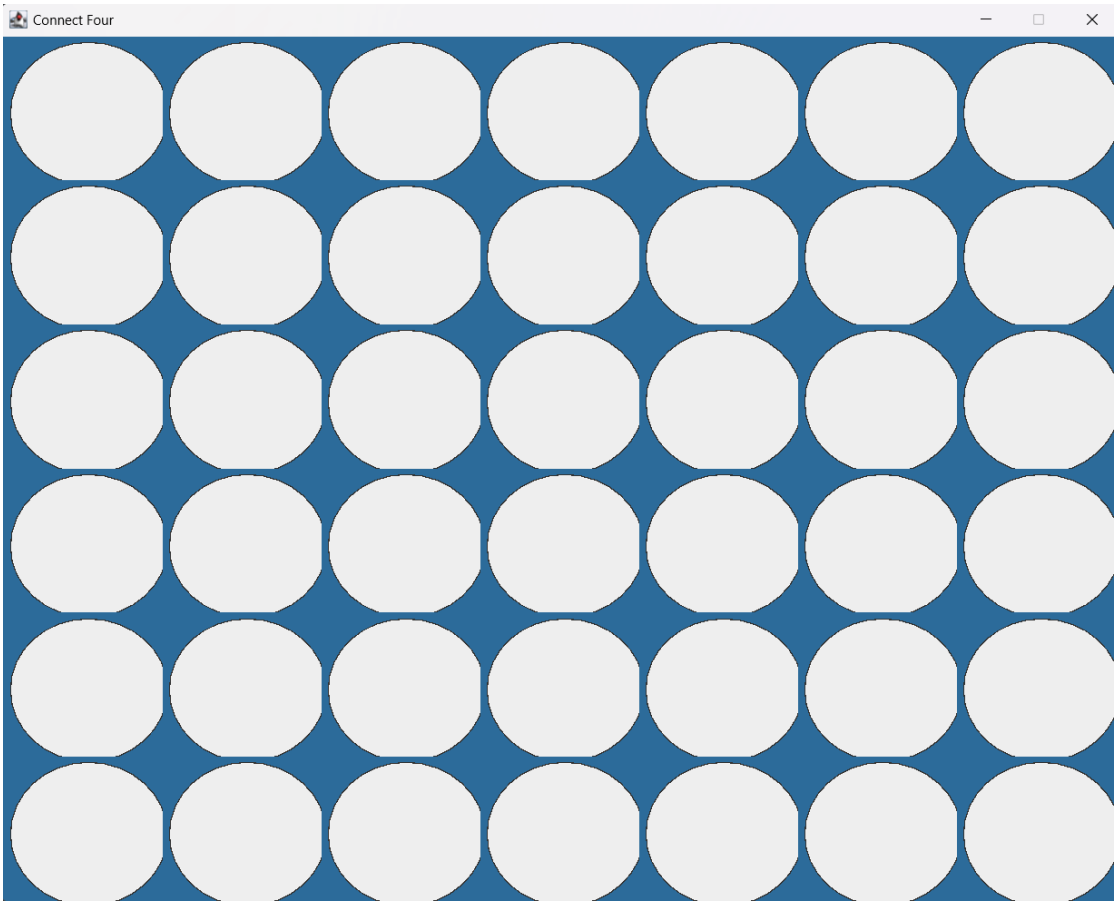
i. Ask to Choose Algorithm



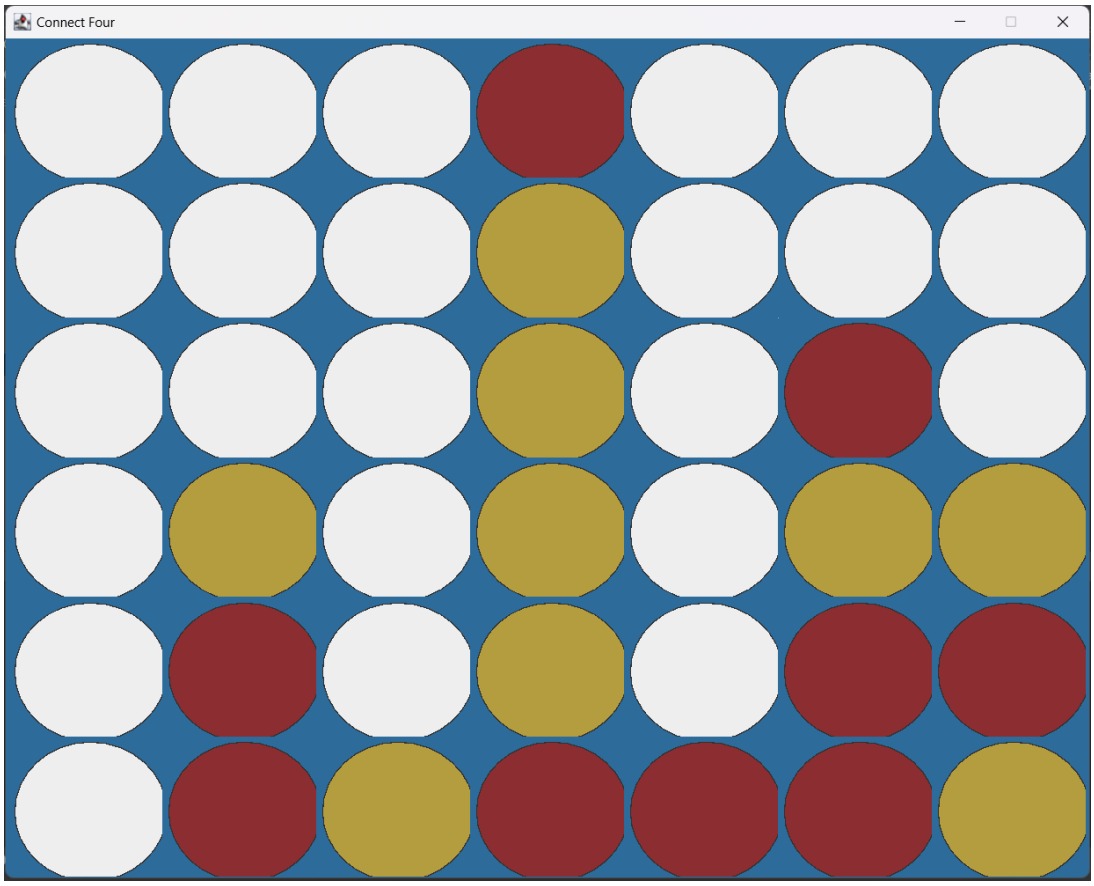
ii. Ask to Enter Depth



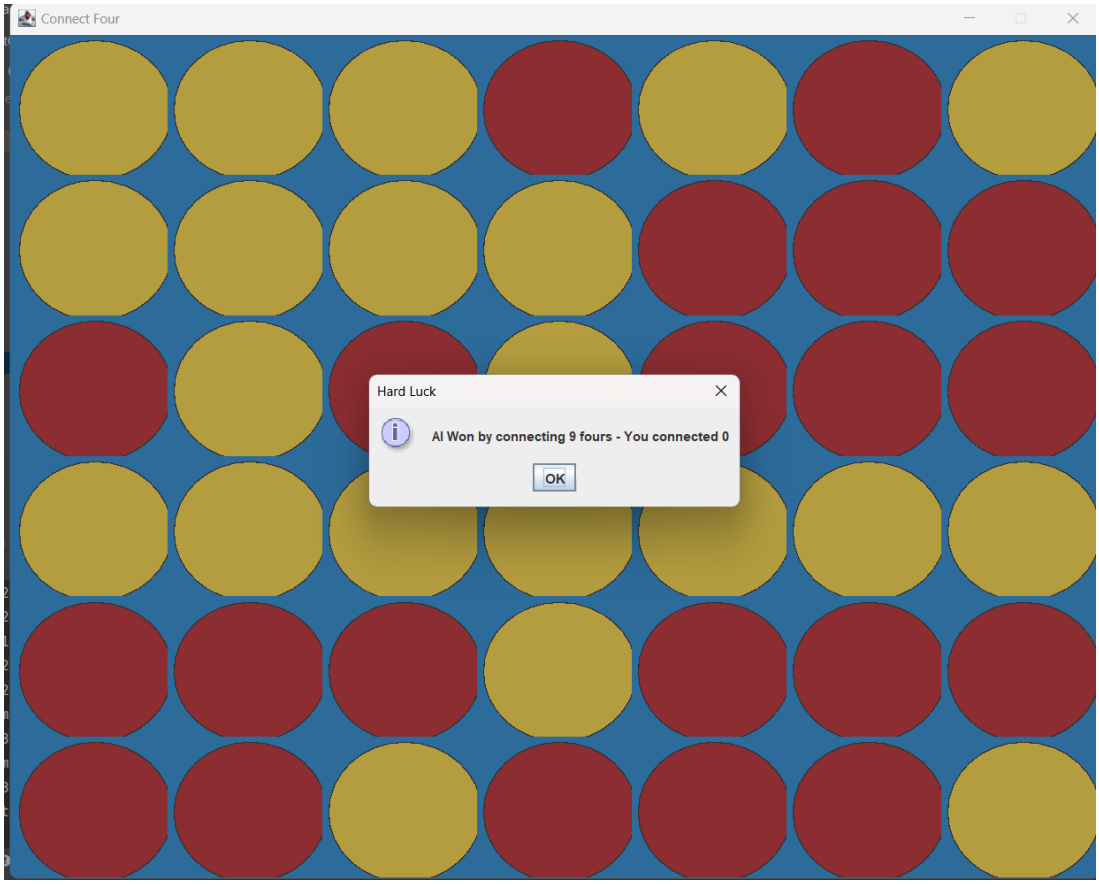
iii. Board of The Game



iv. Game Run



v. Game Finish



vi. Report the information

```
nodes expanded in Minimax= 4763
Minimax without Alpha Beta took (1.7313) milliseconds
nodes expanded in Minimax= 4764
Minimax without Alpha Beta took (0.0535) milliseconds
AI agent won by connecting 9 fours

Process finished with exit code 0
```

---

8. Simple Run with Console

i. Main menu

```
-------------Modes-------------
1. AI vs AI
2. Human Starts
3. AI Starts

Enter Mode :
```

ii. AI vs AI

```
Enter Mode : 1
Enter the depth of Ai 1 : 2
Enter the depth of Ai 2 : 1
```

```
Ai played in col: 2
Ai: 2
-->2) In depth = 0
    | 1 | 1 | 0 | 1 | 2 | 2 | 1 |
    | 2 | 2 | 1 | 2 | 1 | 1 | 2 |
    | 1 | 1 | 2 | 1 | 2 | 2 | 1 |
    | 2 | 2 | 1 | 2 | 1 | 2 | 2 |
    | 1 | 1 | 2 | 1 | 2 | 1 | 1 |
    | 2 | 1 | 1 | 2 | 1 | 2 | 2 |
nodes expanded in Minimax= 4315
Minimax without Alpha Beta took (0.2537) milliseconds
Ai2: 3
---->2) In depth = 0
      | 1 | 1 | 2 | 1 | 2 | 2 | 1 |
      | 2 | 2 | 1 | 2 | 1 | 1 | 2 |
      | 1 | 1 | 2 | 1 | 2 | 2 | 1 |
      | 2 | 2 | 1 | 2 | 1 | 2 | 2 |
      | 1 | 1 | 2 | 1 | 2 | 1 | 1 |
      | 2 | 1 | 1 | 2 | 1 | 2 | 2 |
nodes expanded in Minimax= 4317
Minimax without Alpha Beta took (0.2593) milliseconds
Ai played in col: 2
Ai: 4
nodes expanded in Minimax= 4318
Minimax without Alpha Beta took (0.037) milliseconds
Ai2: 4
The game is draw as both of AI agents connected: 4 fours

Process finished with exit code 0
```

iii. Human Start

```
Enter Mode : 2
Enter the column you want to play in, Human :
5
```

```
    | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
    | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
    | 0 | 0 | 0 | 2 | 0 | 2 | 1 |
-->4) In depth = 2
    | 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 | 0 | 0 | 2 |
    | 0 | 0 | 0 | 0 | 2 | 2 | 1 |
-->5) In depth = 2
    | 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 | 0 | 2 | 2 |
    | 0 | 0 | 0 | 0 | 0 | 2 | 1 |
-->6) In depth = 2
    | 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 | 0 | 0 | 2 |
    | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
    | 0 | 0 | 0 | 0 | 0 | 2 | 1 |
Pruned nodes: 0
Minimax with Alpha Beta took (324.8913) milliseconds
Ai played in col: 3
Ai: 0
Enter the column you want to play in, Human :
4
```

```
Minimax with Alpha Beta took (0.4533) milliseconds
Ai played in col: 0
Ai: 16
Enter the column you want to play in, Human :
1
This column is full, please choose another.
Human: 0
------>0) In depth = 0
        | 2 | 1 | 2 | 1 | 1 | 1 | 1 |
        | 1 | 2 | 1 | 2 | 1 | 1 | 1 |
        | 1 | 2 | 2 | 2 | 1 | 1 | 1 |
        | 1 | 2 | 2 | 2 | 1 | 1 | 1 |
        | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
        | 1 | 2 | 2 | 1 | 2 | 2 | 1 |
Pruned nodes: 0
Minimax with Alpha Beta took (0.262) milliseconds
Ai played in col: 0
Ai: 17
Enter the column you want to play in, Human :
2
This column is full, please choose another.
Human: 0
Pruned nodes: 0
Minimax with Alpha Beta took (0.0175) milliseconds
Ai played in col: 0
Ai: 18
AI agent won by connecting 18 fours

Process finished with exit code 0
```

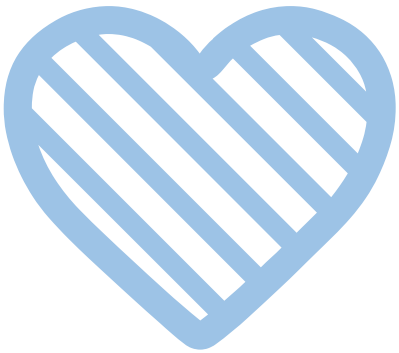
iv. AI Start

```
    | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
    | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
    | 0 | 0 | 0 | 2 | 0 | 0 | 1 |
-->4) In depth = 2
    | 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 | 0 | 0 | 2 |
    | 0 | 0 | 0 | 0 | 2 | 0 | 1 |
-->5) In depth = 2
    | 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 | 0 | 0 | 2 |
    | 0 | 0 | 0 | 0 | 0 | 2 | 1 |
-->6) In depth = 2
    | 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 | 0 | 0 | 2 |
    | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
    | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
Pruned nodes: 0
Minimax with Alpha Beta took (403.9342) milliseconds
Ai played in col: 2
Ai: 0
Enter the column you want to play in, Human :
```

```
Human: 0
------>0) In depth = 0
       | 2 | 2 | 1 | 2 | 1 | 1 | 1 |
       | 1 | 2 | 1 | 2 | 1 | 1 | 1 |
       | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
       | 1 | 2 | 1 | 2 | 1 | 1 | 1 |
       | 1 | 1 | 2 | 1 | 2 | 1 | 2 |
       | 1 | 2 | 1 | 2 | 1 | 2 | 1 |
Pruned nodes: 0
Minimax with Alpha Beta took (0.1576) milliseconds
Ai played in col: 0
Ai: 20
Enter the column you want to play in, Human :
6
This column is full, please choose another.
Human: 0
Pruned nodes: 0
Minimax with Alpha Beta took (0.0203) milliseconds
Ai played in col: 0
Ai: 21
AI agent won by connecting 21 fours

Process finished with exit code 0
```

THANK YOU



https://github.com/abdelrahman-23/Connect4-minimax