



CSE481: Artificial Intelligence

Intelligent Mancala Game

Brief description of the Game:

Mancala is a two-player game, each player has 6 pockets facing them and one mancala at their right. Each pocket is filled with 4 stones at the beginning of the game, and the player that starts the game chooses one pocket on their side of the board to take all the stones that it acquires and distribute them in the neighboring pockets in an anti-clockwise direction, one stone in each pocket.

The player can put the balls in their pockets, mancala, or the opponent's pockets but they're not allowed to place them in the opponent's mancala. If the last piece a player drops is in their mancala, they get to play again for another turn. The game has two modes: With stealing and without stealing. Stealing means that if the last piece a player drops is in an empty pocket on their side, they'll capture that piece and any piece in the hole directly opposite to it.

The main goal is to collect the most stones, so any piece you collect you need to place in your mancala. The game ends when one side has all their pockets empty. The player that still has stones in their pockets takes all these pieces and puts them into their mancala. Count all the pieces in the mancala and the player with the most stones wins.

Implementation:

We've created two main classes: AI and GAME. The AI class is responsible for the AI player that plays against the human in the console, including a function to calculate all possible moves that could result from the current board and the minimax function that creates a tree and uses DFS to implement the alpha-beta pruning to return the board that would lead to the best score. Regarding the Game class it contains the initialization of the board and the print in console function.

Possible_moves function: it takes two parameters: a value to indicate which player's turn it is and the current board. The board is made of a list of two players, from index 0 to 6 is the first player with index 6 being the mancala, and from index 7 to 13 it's the second player, mancala in index 13. We give each player a turn flag to identify them, player 1 -mainly the human player- has flag of 0 for the first part of the list and player two -mainly the AI player- takes the flag of 1. The function takes the current board and returns a dictionary of all possible moves that could come out of this board, depending on the player. The key of the dictionary is the board and the value is which turns follows this board. Meaning, if this board would result in the current player to move another

move, we save that as the value of the board in the dictionary. This function goal is to get all the leafs of each node in the tree by predicting all possible moves.

Minimax Function : It takes multiple parameters including the mode of game whether stealing or not and the current board to build a tree with all possible states(moves on the board) keeping track if any possible move could be excluded because of cutoff till reaching a certain depth which indicates the difficulty of the game or reaching that this move is the last move on the board, and we have a winner.

- First, the turn of the player determines whether the starting of estimation the sequence of the game play is maximizing or minimizing , if it is the AI's turn it will act as maximizer and if it is the human turn it will act as minimizer . for maximizer turn , we start with initial alpha which value is initially equal negative infinity till reaching the wanted depth then calculating the heuristic score for this move in this depth and return to the previous move to check the score with alpha if it larger then it will update it , keep checking every move in this level and getting its score , if it is larger than previous move score it update the alpha and so on , for each move I check if alpha is greater than beta so we cutoff and not complete investigating other moves in this level and return to the above depth. We keep track of every move with its heuristic score till reaching the best path for each maximizer and minimizer.
- For minimizer turn , we start with initial beta which value is initially equal infinity till reaching the wanted depth then calculating the heuristic score for this move in this depth and return to the previous move to check the score with beta if it smaller then it will update it , keep checking every move in this level and getting its heuristic score , if it is smaller than previous move score it update the beta and so on , for each move I check if alpha is greater than beta so we cutoff and not complete investigating other moves in this level and return to the above depth.
- We keep tracking for every move and the turn for whom player to play now.
- For minimax client-server : the only change from the previous algorithm is both AI play vs each other so both are maximizer , so for the server will use Minimax_alphabeta function and for the client Minimax_alphabeta_client , the only change is that we change the player turn flag that indicate the maximizer turn to be = 0 not =1 as it in the Minimax_alphabeta function.

Play Function : this function takes the turn of the player, the board, the mode either stealing or not and the pocket number he wants to move from. Then we start checking if the player is 0 or 1 so that we can know the index of it's mancala. Then we start distributing on the other blocks, after that we check if I'm at the stealing mode and the

last stone I dropped is in an empty pocket on my side so I steal the stones from the player's pocket in front of me and collect all in my mancala plus that 1 stone.

Also this function checks if the last stone I dropped is in my mancala so I win a free turn ! and this function returns the board after change and the player's turn.

Is_finalboard : This function takes the board and checks if there's a complete row of zeros in one of the player's side if yes so this is the final board that ends the game

Bonus:

1. Support various difficulty levels corresponding to different game tree depths.(depth = 3,5,8 : for easy , medium, and hard levels)
2. Allowing 2 AIs to play against each other:

Using socket programming in python, we have a client code and a server code which can run on the same machine or on different machines, On the terminal window we print the board of the game and then after each player's turn to play the new board is sent to the other player via sockets, both players play with the AI algorithm already used in the normal game mood, either stealing or normal mode

The image shows two terminal windows side-by-side, both running a Python script for a Mancala game. The left window shows the game in progress, with the board state displayed as a list of lists. The board is represented by two rows of seven pockets each, with the first row for player 1 and the second row for player 2. The board state is shown as a list of lists, where each inner list represents a player's side of the board. The board is updated after each move, and the player whose turn it is to move is indicated. The right window shows the final state of the game, where the second player (client) has won. The board state is shown as a list of lists, and the message "Second player (client) won!" is displayed.

```

C:\Windows\System32\cmd.exe
[ ] ( 0 ) ( 0 ) ( 1 ) ( 1 ) ( 1 ) ( 0 ) [ ]
[6] [ ] ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 1 ) ( 0 ) [ ]
[ ] 1 2 3 4 5 6
Client played:
[ ] ( 0 ) ( 1 ) ( 0 ) ( 1 ) ( 1 ) ( 0 ) [ ]
[6] [ ] ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 1 ) ( 0 ) [ ]
[ ] 1 2 3 4 5 6
Client played:
[ ] ( 0 ) ( 1 ) ( 0 ) ( 1 ) ( 1 ) ( 0 ) [ ]
[6] [ ] ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 1 ) [ ]
[ ] 1 2 3 4 5 6
Client played:
[ ] ( 1 ) ( 0 ) ( 0 ) ( 1 ) ( 1 ) ( 0 ) [ ]
[6] [ ] ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 1 ) [ ]
[ ] 1 2 3 4 5 6
Second player (client) won!
D:\4th-cse\2ndterm\AI\final2\AImancala>

C:\Windows\System32\cmd.exe
[ ] ( 0 ) ( 0 ) ( 1 ) ( 1 ) ( 1 ) ( 0 ) [ ]
[6] [ ] ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 1 ) ( 0 ) [ ]
[ ] 1 2 3 4 5 6
Client played:
[ ] ( 0 ) ( 1 ) ( 0 ) ( 1 ) ( 1 ) ( 0 ) [ ]
[6] [ ] ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 1 ) ( 0 ) [ ]
[ ] 1 2 3 4 5 6
Client played:
[ ] ( 0 ) ( 1 ) ( 0 ) ( 1 ) ( 1 ) ( 0 ) [ ]
[6] [ ] ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 1 ) [ ]
[ ] 1 2 3 4 5 6
Client played:
[ ] ( 1 ) ( 0 ) ( 0 ) ( 1 ) ( 1 ) ( 0 ) [ ]
[6] [ ] ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 1 ) [ ]
[ ] 1 2 3 4 5 6
Second player (client) won!
D:\4th-cse\2ndterm\AI\final2\AImancala>

```

This screenshot shows that the client (2nd AI) has won.

3. Support game saving and loading: the user has the option to save his game and when he runs the game again he has the choice to load the last game

Utility function:

_____ Four heuristics are derived from game practicing,

1. Prefer “play again” as long as the total number of balls in the current player’s mancala is greater than or equals to the opponent.
Explanation: sometimes the opponent tries to make you run out of your balls first while collecting a large number of balls in his pockets, so when you finish your balls’ “endgame” condition, all of the remaining balls will go to his mancala.
2. Prefer the furthest valid pocket from my mancala, to avoid getting stoled and running out of balls firstly.

If stealing mode is activated

3. Prevent the stealing move of the opponent, an extra score is given for the move that prevents the stealing. The score is weighted with the number of balls, the opponent can steal.
4. Steals checks if there is a move that can steal the opponent’s balls, also the score is weighted with the number of balls.

Snapshots:

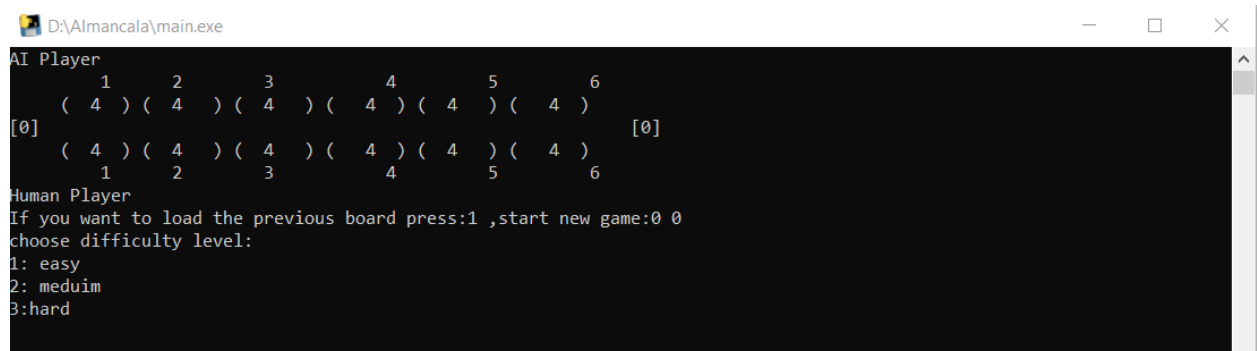
Start of the game:



A screenshot of a Windows application window titled "D:\Almancala\main.exe". The window displays a text-based Mancala game interface. At the top, it says "AI Player" followed by a row of six pairs of parentheses, each containing a "4", representing the initial state of the board. Below this, there is a line with "[0]" at both ends. Another row of six pairs of parentheses with "4"s follows, with "[0]" at the right end. Below that is a row of six numbers: 1, 2, 3, 4, 5, 6. The text "Human Player" appears next. At the bottom, a prompt reads: "If you want to load the previous board press:1 ,start new game:0".

```
D:\Almancala\main.exe
AI Player
  1      2      3      4      5      6
( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
[0]                                     [0]
  1      2      3      4      5      6
( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
  1      2      3      4      5      6
Human Player
If you want to load the previous board press:1 ,start new game:0
```

Choose the difficulty level:



A screenshot of the same Windows application window, now showing a prompt for difficulty level selection. The game board state is identical to the previous snapshot. Below the "Human Player" text, the prompt "choose difficulty level:" is displayed, followed by three options: "1: easy", "2: meduim", and "3:hard".

```
D:\Almancala\main.exe
AI Player
  1      2      3      4      5      6
( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
[0]                                     [0]
  1      2      3      4      5      6
( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
  1      2      3      4      5      6
Human Player
If you want to load the previous board press:1 ,start new game:0 0
choose difficulty level:
1: easy
2: meduim
3:hard
```

Stealing and choosing who to play first (user or AI):

```
D:\Almancala\main.exe
AI Player
  1      2      3      4      5      6
( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
[0]                                     [0]
  1      2      3      4      5      6
Human Player
If you want to load the previous board press:1 ,start new game:0 0
choose difficulty level:
1: easy
2: meduim
3:hard
3
For stealing mode press: 1 else press: 0 1
choose 0 if you want to play first else 1 : 1
```

AI played in hard mode and you're asked after each move you make or AI makes if you want to save the game to load later:

```
D:\Almancala\main.exe
AI Player
  1      2      3      4      5      6
( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
[0]                                     [0]
  1      2      3      4      5      6
Human Player
If you want to load the previous board press:1 ,start new game:0 0
choose difficulty level:
1: easy
2: meduim
3:hard
3
For stealing mode press: 1 else press: 0 1
choose 0 if you want to play first else 1 : 1
If you want to save the game press: 1 else press: 0 0
AI Player
  1      2      3      4      5      6
( 0 ) ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
[1]                                     [0]
  1      2      3      4      5      6
( 5 ) ( 5 ) ( 5 ) ( 4 ) ( 4 ) ( 4 )
  1      2      3      4      5      6
Human Player
If you want to save the game press: 1 else press: 0
```

Player chose the index of the pocket to move it's stones, and board changed, then AI started to play (it played twice because it chose a board that leads to repetition):

```
Human Player
If you want to save the game press: 1 else press: 0 0
Enter a move: 1
AI Player
      1      2      3      4      5      6
    ( 0 ) ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
[1]                                     [0]
    ( 0 ) ( 6 ) ( 6 ) ( 5 ) ( 5 ) ( 5 )
      1      2      3      4      5      6
Human Player
AI Player
      1      2      3      4      5      6
    ( 1 ) ( 5 ) ( 5 ) ( 0 ) ( 4 ) ( 4 )
[2]                                     [0]
    ( 0 ) ( 6 ) ( 6 ) ( 5 ) ( 5 ) ( 5 )
      1      2      3      4      5      6
Human Player
AI Player
      1      2      3      4      5      6
    ( 1 ) ( 6 ) ( 6 ) ( 1 ) ( 5 ) ( 0 )
[2]                                     [0]
    ( 0 ) ( 6 ) ( 6 ) ( 5 ) ( 5 ) ( 5 )
      1      2      3      4      5      6
Human Player
If you want to save the game press: 1 else press: 0
```

A move that leads player to steal and the result:

```
AI Player
      1      2      3      4      5      6
    ( 0 ) ( 0 ) ( 9 ) ( 1 ) ( 7 ) ( 3 )
[4]                                     [3]
    ( 2 ) ( 1 ) ( 8 ) ( 1 ) ( 0 ) ( 9 )
      1      2      3      4      5      6
Human Player
If you want to save the game press: 1 else press: 0 0
Enter a move: 4
```



```

      1      2      3      4      5      6
    ( 0 ) ( 0 ) ( 9 ) ( 1 ) ( 7 ) ( 3 )
[4]                                     [3]
      1      2      3      4      5      6
    ( 2 ) ( 1 ) ( 8 ) ( 1 ) ( 0 ) ( 9 )
      1      2      3      4      5      6
Human Player
If you want to save the game press: 1 else press: 0 0
Enter a move: 4
AI Player
      1      2      3      4      5      6
    ( 0 ) ( 0 ) ( 9 ) ( 1 ) ( 0 ) ( 3 )
[4]                                     [11]
      1      2      3      4      5      6
    ( 2 ) ( 1 ) ( 8 ) ( 0 ) ( 0 ) ( 9 )
      1      2      3      4      5      6
Human Player

```

Final result:

```

AI Player
      1      2      3      4      5      6
    ( 1 ) ( 1 ) ( 1 ) ( 0 ) ( 1 ) ( 3 )
[18]                                     [22]
      1      2      3      4      5      6
    ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 0 ) ( 0 )
      1      2      3      4      5      6
Human Player
AI won!
press close to exit

```

Saving game:

```
( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
[0]                                     [0]
( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
  1   2   3   4   5   6
Human Player
If you want to load the previous board press:1 ,start new game:0 0
choose difficulty level:
1: easy
2: meduim
3:hard
3
For stealing mode press: 1 else press: 0 0
choose 0 if you want to play first else 1 : 0
If you want to save the game press: 1 else press: 0 0
Enter a move: 1
AI Player
  1   2   3   4   5   6
( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
[0]                                     [0]
( 0 ) ( 5 ) ( 5 ) ( 5 ) ( 5 ) ( 4 )
  1   2   3   4   5   6
Human Player
AI Player
  1   2   3   4   5   6
( 5 ) ( 0 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
[1]                                     [0]
( 1 ) ( 6 ) ( 5 ) ( 5 ) ( 5 ) ( 4 )
  1   2   3   4   5   6
Human Player
If you want to save the game press: 1 else press: 0 1
```

Loading game:

```
AI Player
      1      2      3      4      5      6
    ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
[0]
      1      2      3      4      5      6
Human Player
If you want to load the previous board press:1 ,start new game:0 1
choose difficulty level:
1: easy
2: meduim
3:hard
3
AI Player
      1      2      3      4      5      6
    ( 5 ) ( 0 ) ( 4 ) ( 4 ) ( 4 ) ( 4 )
[1]
      1      2      3      4      5      6
    ( 1 ) ( 6 ) ( 5 ) ( 5 ) ( 5 ) ( 4 )
Human Player
If you want to save the game press: 1 else press: 0
```

Github Link: <https://github.com/AlaaMohamed99/Almancala>

Main project video: <https://www.youtube.com/watch?v=366bWnFaPM4>

Networking video: <https://youtu.be/r1ZySx5sYjo>

Team members and the work load:

Names	Codes	Workload
Alaa Mohamed Abdelrahman	1600281	AI class, possible_moves function, linking, Save, Difficulty level bonus
Tasnim Ahmed Medhat	1600431	play function, get_empty_pockets, calc_score, Is_finalboard. Networking bonus
Baraa Magdy El Sayed	1600392	1.Project design 2.Heuristic function 3- save and load bonus
Panse Yasser Mohamed	1600386	In GAME class: initialization of the board and print function, Heuristic function, Networking bonus
Reem Nasser Abd El Fatah	1600594	AI class, Minimax functions, linking, Difficulty level bonus Networking bonus linking