Assignment 4: Mancala

Mancala is one of the oldest known board games that are still widely played today. It is a two-player, turn-based strategy game where the goal is to collect the most stones in your storage. An image of the game can be seen below.

The objective of the game is to capture more stones than your opponent. How to play:

1. The Mancala board is made up of two rows of six holes, or pits, each.
2. Four stones are placed in each of the 12 holes.
3. Each player has a store (called a Mancala) on their right side of the Mancala board.
4. The game begins with one player picking up all of the pieces in any one of the holes on their side.
5. Moving counter-clockwise, the player deposits one of the stones in each hole until the stones run out.
6. If the player runs into their own store, they deposit one piece in it. If they run into their opponent's store, they skip it.
7. If the last piece dropped is in the player's own store, they get a free turn.
8. If the last piece is dropped in an empty hole on the player's side, they capture that piece and any pieces in the hole directly opposite.
9. Captured pieces are always placed in the current player's store.
10. The game ends when all six spaces on one side of the Mancala board are empty.
11. The player who still has pieces on their side of the board when the game ends captures all those pieces.
12. The pieces in each store are counted. The winner is the player with the most pieces.

The rules are taken from: [https://www.thesprucecrafts.com/how-to-play-mancala-409424Links to an external site.](https://www.thesprucecrafts.com/how-to-play-mancala-409424). There is also a video with the rules.

What to do: Your assignment now is to apply the Minimax algorithm with your own utility function to play against three different bots **(5 points)**:

- p3_bot (if you win, you get a 3)

- p4_bot (if you win, you get a 4)
- p5_bot (if you win, you get a 5)

Your report has to cover the key parts as follows:

1. Explain the used utility function. Additionally, you have to add the code of the utility function to the report, explaining it with comments

How to Use the Given Code:

You will use the files in assignment 4 files.rar Download assignment 4 files.rar. Your solution will be written in P_client.py. **Make sure you have installed numpy, matplotlib and that you are running python version 3.13.**

Your bot will interact with the Mancala game server by receiving the state of the board and determining which move to make based on the current player's turn.

Variables Description:

- **board**: A list of integers representing the number of stones in each pit and store. This includes all pits for both players and both stores.
- **playerTurn**: Indicates which player's turn it is. 1 means it's Player 1's turn and you can choose a pit from positions 0 to 5 (Player 1's side). 2 means it's Player 2's turn, allowing you to choose from positions 7 to 12 (Player 2's side).
- **move**: The pit you choose to empty, represented by a number from 1 to 6. This is independent of the side you are playing on; 1 corresponds to the first pit on your side, 6 to the last pit.

Steps to Run Your Bot:

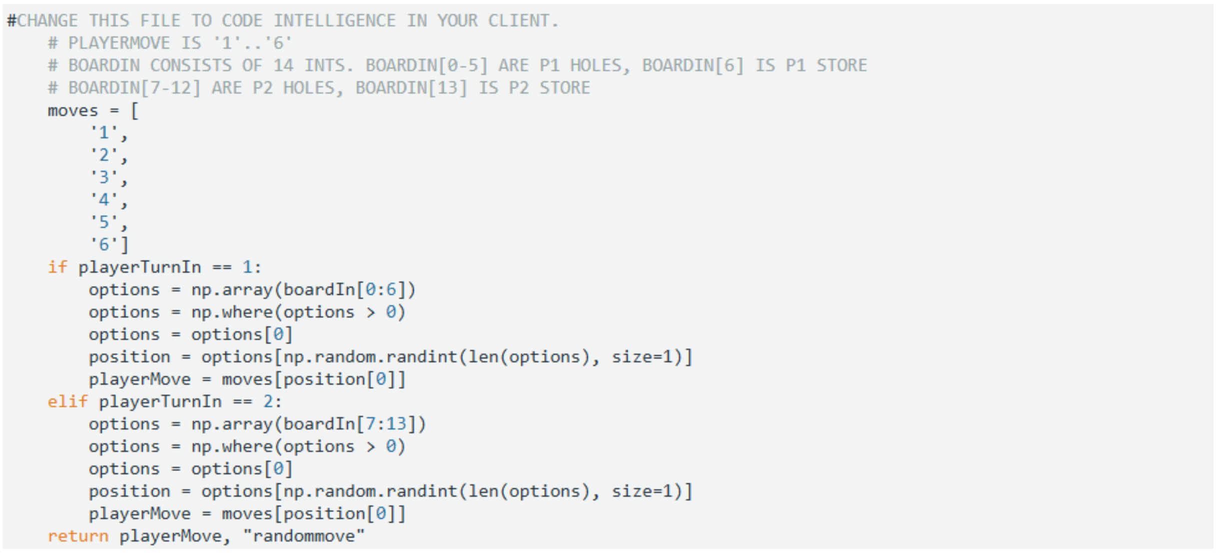
1. **Start the Mancala game server**: Make sure it is running before you start any clients.
2. **Run Your Client (P_client.py)**: This should be done within 20 seconds after starting the Mancala server to ensure proper synchronization.
3. **Check for Connection Issues**: If your bot does not connect or function properly, restart the process up to three times. If issues persist, talk to a teacher.

During Gameplay:

- A folder will be created to store the data for each game session, which includes moves and outcomes for further analysis.

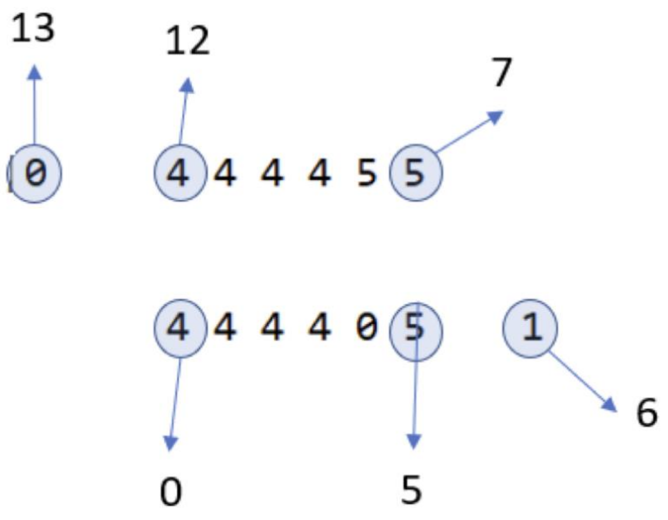
Testing Against Bots:

- To test your bot, you will be competing against predefined bots (P3_bot.pyc, P4_bot.pyc, P5_bot.pyc), each representing different levels of difficulty. These files should be run as separate processes after your client has connected to the server.
- Your grade for the assignment is dependent on what bot your implementation is able to beat! I.e. if you beat p3 you'll get a 3, etc.
- Here, in P_client.py you have to change the code:



```
#CHANGE THIS FILE TO CODE INTELLIGENCE IN YOUR CLIENT.
    # PLAYERMOVE IS '1'..'6'
    # BOARDIN CONSISTS OF 14 INTS. BOARDIN[0-5] ARE P1 HOLES, BOARDIN[6] IS P1 STORE
    # BOARDIN[7-12] ARE P2 HOLES, BOARDIN[13] IS P2 STORE
    moves = [
        '1',
        '2',
        '3',
        '4',
        '5',
        '6']
    if playerTurnIn == 1:
        options = np.array(boardIn[0:6])
        options = np.where(options > 0)
        options = options[0]
        position = options[np.random.randint(len(options), size=1)]
        playerMove = moves[position[0]]
    elif playerTurnIn == 2:
        options = np.array(boardIn[7:13])
        options = np.where(options > 0)
        options = options[0]
        position = options[np.random.randint(len(options), size=1)]
        playerMove = moves[position[0]]
    return playerMove, "randommove"
```

- When printing the game, you can see the indexes in the following positions:



Pre-conditions in order to present your code:

- Your Minimax algorithm is allowed to check up to d = 3 (meaning that you can check your first action, the first action of the opponent and your second action). After that, you have to use the utility function. Example moves:
    - MAX, MIN, MAX
    - MAX, MAX, MAX
    - MAX, MIN, MIN
    - ...
- **Change your playerName to “Name_LastName”**
- The code is prepared so that your bot has to give a desired movement in less than 0.2 seconds. The match consists of 10 games.
- If you win against P3_bot and P5_bot, but lose against P4_bot, you get a 3.
- In order to win against a player, you have to have more won games than lost games.

- You will not pass if you just code a random generator. So, you have to propose a utility function.

Conditions to approve assignment 4:

To pass this assignment, you must:

1. **Submit the report according to the criteria (and present and send in your code).**
2. **Have a total score of at least 3.**

SCORES 3-5

- Your score < 3: U
- 3 <= your score <= 3.75: 3
- 3.75 < your score < 4.75: 4
- 4.75 <= your score: 5 **SCORES A-F**
- Your score < 3: F
- 3 <= your score < 3.4: E
- 3.4 <= your score < 3.8: D
- 3.8 <= your score < 4.2: C
- 4.2 <= your score < 4.6: B
- 4.6 <= your score: A