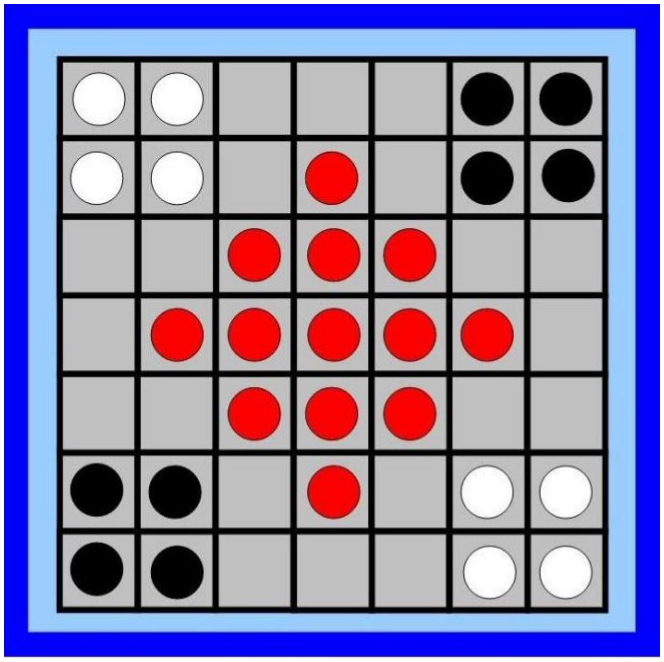
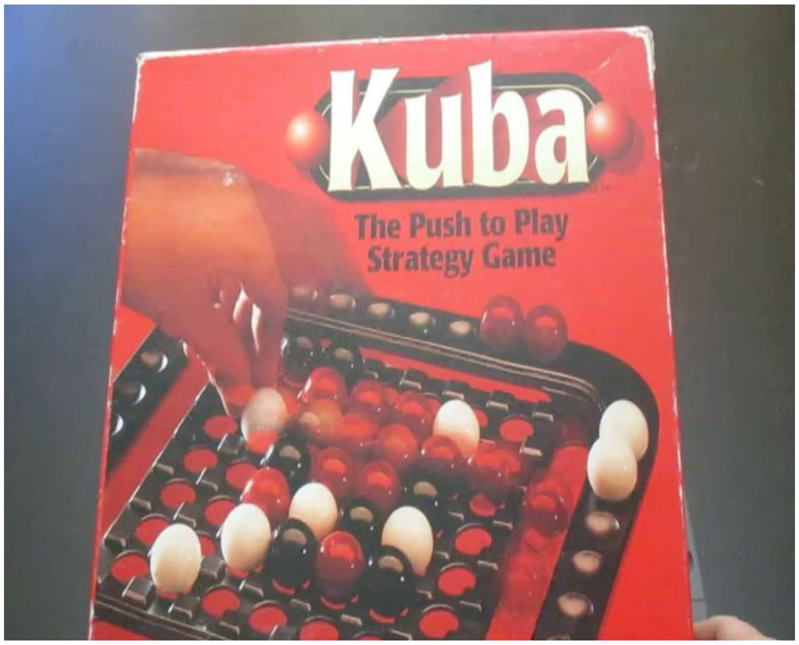
KUBA – Game Analysis and Implementation

Develop the game of AI and analyze the algorithm that can facilitate accept or reject a move: -





Dictionary Data Structure is used to store the game state of the board.

X - Denotes the empty spaces on the board, R - Neutral Red Marbles W - White Marbles - Player 1, B - Black Marbles - Player 2

```
1: ['W', 'W', 'X', 'X', 'X', 'B', 'B'],
    2: ['W', 'W', 'X', 'R', 'X', 'B', 'B'],
    3: ['X', 'X', 'R', 'R', 'R', 'X', 'X'],
    4: ['X', 'R', 'R', 'R', 'R', 'R', 'X'],
    5: ['X', 'X', 'R', 'R', 'R', 'X', 'X'],
    6: ['B', 'B', 'X', 'R', 'X', 'W', 'W'],
    7: ['B', 'B', 'X', 'X', 'X', 'W', 'W']
}
```

Object Oriented Structure Design.

Class Player having the attributes of Name, Color, Count – Normal get and set functions
Class Board having the attribute of Game State

Class Kuba Board having the attributes of Player 1, Player 2, board, winner, turn, player1 previous board and player2 previous board.

Rajat Kumar

Main Functions: -

make_move() – Most Important function.

Thought Process on developing the function: -

- Player provides the coordinates of the position of movement point.
- Player also provides the direction [Left(L), Right(R), Forward (F), Backward (B)]
- Code identifies whether the Player provided coordinates position contains the player marble color or not – if not return false.
- Code identifies that the directions is one of L, R, F, B
- To validate that the move can be made or not. We identify the four directions of movement from the provided coordinates (named as North, South, West, and East)
  For example: if player makes a move of (7, 7) – in direction L
  Then we check what is at positions (7,8), (7,6), (6,7) & (8,7) – If it is out of bound it returns 'X' as well, because a move can be made edge of the board.
- Depending upon the directions provided we identify the freedom in that direction and move the marbles accordingly.

  For example –

```
['W', 'W', 'X', 'X', 'X', 'B', 'B']
['W', 'W', 'X', 'R', 'X', 'B', 'B']
['X', 'X', 'R', 'R', 'R', 'X', 'X']
['X', 'R', 'R', 'R', 'R', 'R', 'X']
['X', 'X', 'R', 'R', 'R', 'X', 'X']
['B', 'B', 'X', 'R', 'X', 'W', 'W']
['B', 'B', 'X', 'X', 'X', 'W', 'W']
```

  Marble Count (White, Black, Red) is - (8, 8, 13)

  Player provides the coordinates as (7, 7) and direction L

  North (6,7) – W
  South (8,7) – out of bound – X
  West (7,6) – W
  East (7, 8) – out of bound – X
  To make this move, on needs to East Side should return X.

  Once, this has been validated, you need to entire row that is 7th row as there is shifting.

```
['W', 'W', 'X', 'X', 'X', 'B', 'B']
['W', 'W', 'X', 'R', 'X', 'B', 'B']
['X', 'X', 'R', 'R', 'R', 'X', 'X']
['X', 'R', 'R', 'R', 'R', 'R', 'X']
['X', 'X', 'R', 'R', 'R', 'X', 'X']
['B', 'B', 'X', 'R', 'X', 'W', 'W']
```

Rajat Kumar

```
['B', 'B', 'X', 'X', 'W', 'W', 'X']
```

Ko- Rule Scenario –

```
['W', 'W', 'X', 'X', 'X', 'B', 'B']
['W', 'W', 'X', 'R', 'X', 'B', 'B']
['X', 'X', 'R', 'R', 'R', 'X', 'X']
['X', 'R', 'R', 'R', 'R', 'R', 'X']
['X', 'X', 'R', 'R', 'R', 'X', 'X']
['B', 'B', 'X', 'R', 'X', 'W', 'W']
['X', 'B', 'B', 'X', 'W', 'W', 'X']
```

Player A provides the coordinates as (7, 6) and direction L

```
['W', 'W', 'X', 'X', 'X', 'B', 'B']
['W', 'W', 'X', 'R', 'X', 'B', 'B']
['X', 'X', 'R', 'R', 'R', 'X', 'X']
['X', 'R', 'R', 'R', 'R', 'R', 'X']
['X', 'X', 'R', 'R', 'R', 'X', 'X']
['B', 'B', 'X', 'R', 'X', 'W', 'W']
['X', 'B', 'B', 'W', 'W', 'X', 'X']
```

Player B provides the coordinates as (7, 2) and direction R

```
['W', 'W', 'X', 'X', 'X', 'B', 'B']
['W', 'W', 'X', 'R', 'X', 'B', 'B']
['X', 'X', 'R', 'R', 'R', 'X', 'X']
['X', 'R', 'R', 'R', 'R', 'R', 'X']
['X', 'X', 'R', 'R', 'R', 'X', 'X']
['B', 'B', 'X', 'R', 'X', 'W', 'W']
['X', 'X', 'B', 'B', 'W', 'W', 'X']
```

Player A provides the coordinates as (7, 6) and direction L

```
Ko rule is violated at this move – Same move is repeated by Player A
False
        ['W', 'W', 'X', 'X', 'X', 'B', 'B']
        ['W', 'W', 'X', 'R', 'X', 'B', 'B']
        ['X', 'X', 'R', 'R', 'R', 'X', 'X']
        ['X', 'R', 'R', 'R', 'R', 'R', 'X']
        ['X', 'X', 'R', 'R', 'R', 'X', 'X']
        ['B', 'B', 'X', 'R', 'X', 'W', 'W']
        ['X', 'X', 'B', 'B', 'W', 'W', 'X']
```

Rajat Kumar

Capture Red – Scenario –

```
['X', 'W', 'X', 'X', 'X', 'B', 'B']
['X', 'W', 'X', 'R', 'R', 'B', 'B']
['W', 'X', 'R', 'R', 'R', 'X', 'X']
['X', 'W', 'R', 'R', 'R', 'R', 'R']
['X', 'X', 'R', 'R', 'W', 'X', 'X']
['X', 'X', 'X', 'X', 'B', 'X', 'R']
['X', 'X', 'B', 'B', 'X', 'B', 'X']
```

Player A provides the coordinates as (4, 2) and direction R

Red marble is captured in 4th row by Player A
Captured Marbles count for Player A is 1
Captured Marbles count for Player B is 0
Marble Count (White, Black, Red) is - (5, 8, 12)

```
['X', 'W', 'X', 'X', 'X', 'B', 'B']
['X', 'W', 'X', 'R', 'R', 'B', 'B']
['W', 'X', 'R', 'R', 'R', 'X', 'X']
['X', 'X', 'W', 'R', 'R', 'R', 'R']
['X', 'X', 'R', 'R', 'W', 'X', 'X']
['X', 'X', 'X', 'X', 'B', 'X', 'R']
['X', 'X', 'B', 'B', 'X', 'B', 'X']
```

validate_board ()

It basically checks Ko – Rule Scenario (Explained – Above) by comparing the old board state stored in player1 previous board or player2 previous board and compare it with the current board

check_game_winner ()

It called under validate board function and winner is set based on 2 scenarios: -

- If more than 7 red marbles are captured.
- If any player has no marbles left.

Rajat Kumar

Performance & Space Analysis: -

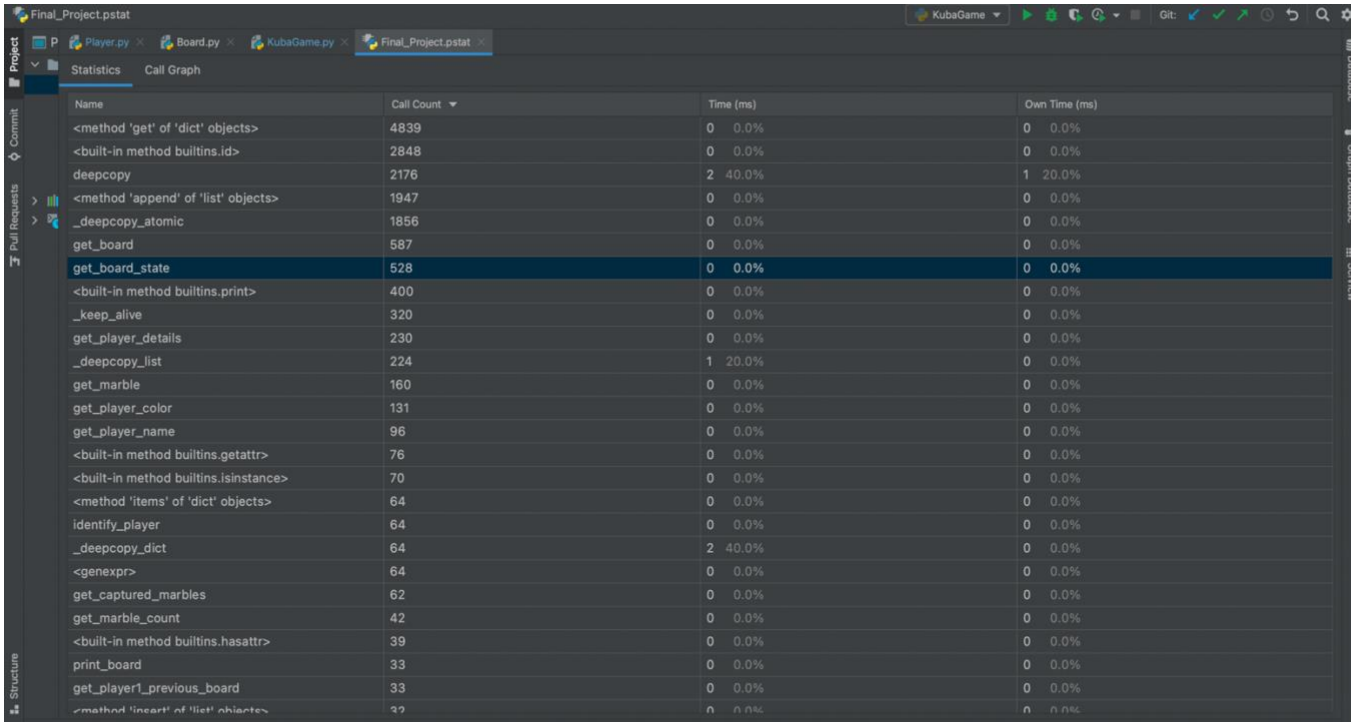
The main analysis/ discussion point in the developed program is an issue of Space Complexity, how to prevent frequent creating and saving deep copies of the board.
Since, it is critical part of the implementation, there is possibility of better workaround that space.

Using of dictionary is useful, as updating a particular row is O (1) in case the movements are left and right.

In case of board update when movement is Forward or Backward, all the values are first fetched in the list and then updated depending on the coordinate. It should take O(n).

For comparing the two dictionaries it takes O(n) equality checks with n the number of items.

Getting the marble count – Iterates through the entire board that is a dictionary, O(n).



Name	Call Count	Time (ms)	Own Time (ms)
<method 'get' of 'dict' objects>	4839	0 0.0%	0 0.0%
<built-in method builtins.id>	2848	0 0.0%	0 0.0%
deepcopy	2176	2 40.0%	1 20.0%
<method 'append' of 'list' objects>	1947	0 0.0%	0 0.0%
_deepcopy_atomic	1856	0 0.0%	0 0.0%
get_board	587	0 0.0%	0 0.0%
get_board_state	528	0 0.0%	0 0.0%
<built-in method builtins.print>	400	0 0.0%	0 0.0%
_keep_alive	320	0 0.0%	0 0.0%
get_player_details	230	0 0.0%	0 0.0%
_deepcopy_list	224	1 20.0%	0 0.0%
get_marble	160	0 0.0%	0 0.0%
get_player_color	131	0 0.0%	0 0.0%
get_player_name	96	0 0.0%	0 0.0%
<built-in method builtins.getattr>	76	0 0.0%	0 0.0%
<built-in method builtins.isinstance>	70	0 0.0%	0 0.0%
<method 'items' of 'dict' objects>	64	0 0.0%	0 0.0%
identify_player	64	0 0.0%	0 0.0%
_deepcopy_dict	64	2 40.0%	0 0.0%
<genexpr>	64	0 0.0%	0 0.0%
get_captured_marbles	62	0 0.0%	0 0.0%
get_marble_count	42	0 0.0%	0 0.0%
<built-in method builtins.hasattr>	39	0 0.0%	0 0.0%
print_board	33	0 0.0%	0 0.0%
get_player1_previous_board	33	0 0.0%	0 0.0%
<method 'insert' of 'list' objects>	32	0 0.0%	0 0.0%

Rajat Kumar