

## CS510 -Assignment 4

### 1) Random Agent

The Othello game is initially designed for 2 Human players, when we use the random function the player's move gets picked at random and the game continues. Here I have stored the possible moves for the particular state in a list, and used the random.choice function to pick a move.

Output for 2 random players:

```
Last login: Sun Nov 14 19:44:58 on ttys019
(base) brindakulkarni@BRINDAs-MacBook-Air Othello_Code % python3 main.py random random

Current state, O to move:
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .

0: Player O to 2,4
1: Player O to 3,5
2: Player O to 4,2
3: Player O to 5,3
Random move: 3
Player O to 5,3

Current state, X to move:
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .

0: Player X to 3,2
1: Player X to 5,2
2: Player X to 5,4
Random move: 1
Player X to 5,2
```

....

```
Current state, O to move:
0 0 0 0 0 0 0
X X X X X 0 . X
0 X 0 X X X X X
0 0 X 0 X X X X
. X 0 X X X 0 X
X X 0 0 X 0 0 X
0 X 0 0 0 X 0 0
0 0 0 0 0 0 0

0: Player O to 1,6
1: Player O to 4,0
Random move: 1
Player O to 4,0

Current state, X to move:
0 0 0 0 0 0 0
X X X X X 0 . X
0 X 0 X X X X X
0 0 X 0 X X X X
0 0 0 X X X 0 X
0 0 0 0 X 0 0 X
0 X 0 0 0 X 0 0
0 0 0 0 0 0 0

0: Player X to 1,6
Random move: 0
Player X to 1,6

*** Final winner: 0 ***
0 0 0 0 0 0 0
```

```

((base) brindakulkarni@BRINDAs-MacBook-Air Othello_Code % python3 main.py random random

Current state, 0 to move:
. . . . .
. . . . .
. . . 0 X . . .
. . . X 0 . . .
. . . . .
. . . . .
. . . . .

0: Player 0 to 2,4
1: Player 0 to 3,5
2: Player 0 to 4,2
3: Player 0 to 5,3
Random move: 0
Player 0 to 2,4

Current state, X to move:
. . . . .
. . . . .
. . . . 0 . . .
. . . 0 0 . . .
. . . X 0 . . .
. . . . .
. . . . .
. . . . .

0: Player X to 2,3
1: Player X to 2,5
2: Player X to 4,5
Random move: 2
Player X to 4,5

```

.....

```

Current state, 0 to move:
X X X X X X X X
. X X X X X X 0
X X X X X 0 0
X 0 0 0 X X X X
X 0 0 0 0 X 0 X
0 0 0 0 0 X 0 X
0 0 X 0 X 0 0 X
0 . 0 0 0 0 0 X

0: Player 0 to 1,0
1: Player 0 to 7,1
Random move: 0
Player 0 to 1,0

Current state, X to move:
X X X X X X X X
0 0 0 0 0 0 0 0
0 0 X X X X 0 0
0 0 0 0 X X X X
0 0 0 0 0 X 0 X
0 0 0 0 0 X 0 X
0 0 X 0 X 0 0 X
0 . 0 0 0 0 0 X

0: Player X to 7,1
Random move: 0
Player X to 7,1

*** Final winner: X ***
X X X X X X X X
0 X 0 0 0 0 0 0
0 X X X X X 0 0
0 X 0 0 X X X X
0 X 0 0 0 X 0 X
0 X 0 0 0 X 0 X
0 X X 0 X 0 0 X
0 X X X X X X X

```

## 2) Minimax agent

The Minimax algorithm is used to find the best possible value according to the heuristic provided. It is a recursive algorithm that searches all the possible moves to find the one with the best outcome. It uses the Depth First Search technique. I have considered the number of coloured disks as the heuristic for the minimax algorithm.

```
(base) brindakulkarni@BRINDAS-MacBook-Air Othello_Code % python3 main.py minimax random 3

Current state, O to move:
. . . . .
. . . . .
. . . . .
. . . 0 X . .
. . . X 0 . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .

0: Player O to 2,4
1: Player O to 3,5
2: Player O to 4,2
3: Player O to 5,3
3
Player O to 5,3

Current state, X to move:
. . . . .
. . . . .
. . . . .
. . . 0 X . .
. . . 0 0 . .
. . . 0 . . .
. . . . .
. . . . .
. . . . .

0: Player X to 3,2
1: Player X to 5,2
2: Player X to 5,4
Random move: 2
Player X to 5,4
```

□ □ □ □

```
Current state, 0 to move:
```

```

. X X X . X O X
O O X X X O O X
O O X X O X O O
O X X O X X O O
O O X O X X X O
O O O X O O X O
O O X X X X X O
O O O X O O O O

```

0: Player 0 to 0,4

9

Player 0 to 0,4

Current state, X to move:

```

. X X X 0 0 0 X
0 0 X 0 0 0 0 X
0 0 0 X 0 X 0 0
0 0 X 0 X X 0 0
0 0 X 0 X X X 0
0 0 0 X 0 0 X 0
0 0 X X X X X 0
0 0 0 X 0 0 0 0

```

0: Player X to 0,0

Random move: 0

Player X to 0,0

```
*** Final winner: 0 ***
```

X	X	X	X	0	0	0	X
0	X	X	0	0	0	0	X
0	0	X	X	0	X	0	0
0	0	X	X	X	X	0	0
0	0	X	0	X	X	X	0
0	0	0	X	0	0	X	0
0	0	X	X	X	X	X	0
0	0	0	X	0	0	0	0

```
(base) brindakulkarni@BRINDAs-MacBook-Air Othello_Code % python3 main.py random minimax 3

Current state, 0 to move:
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+

0: Player 0 to 2,4
1: Player 0 to 3,5
2: Player 0 to 4,2
3: Player 0 to 5,3
Random move: 0
Player 0 to 2,4

Current state, X to move:
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+
+-----+

0: Player X to 2,3
1: Player X to 2,5
2: Player X to 4,5
2
Player X to 4,5
```

....

```
Current state, 0 to move:
. . 0 X X X X X
0 0 X X X X X 0
X X 0 X 0 X X 0
X X 0 X 0 0 X 0
X X X 0 0 X X 0
X 0 X 0 0 X X 0
X X 0 0 X 0 X 0
X 0 0 0 0 0 0 0

0: Player 0 to 0,1
Random move: 0
Player 0 to 0,1

Current state, X to move:
. 0 0 X X X X X
0 0 0 X X X X 0
X X 0 0 0 X X 0
X X 0 X 0 0 X 0
X X X 0 0 X X 0
X 0 X 0 0 X X 0
X X 0 0 X 0 X 0
X 0 0 0 0 0 0 0

0: Player X to 0,0
0
Player X to 0,0

*** Final winner: X ***
X X X X X X X X
X X 0 X X X X 0
X X X 0 0 X X 0
X X 0 X 0 0 X 0
X X X 0 0 X X 0
X 0 X 0 0 X X 0
X X 0 0 X 0 X 0
X 0 0 0 0 0 0 0
```

### 3) Alpha Beta Pruning

The Alpha- Beta-Pruning algorithm is an optimization that improves the MiniMax algorithm. Alpha-Beta-Pruning is a strategy for pruning nodes that aren't needed to analyze the alternative moves instead of exploring the complete tree. The best position value that Max player can reach using a pessimistic evaluation is Alpha, while the best position value that Max player can reach using an optimistic evaluation is Beta. In the beginning of the search, alpha is -9999, while beta is +9999. At each depth, the program evaluates the nodes using optimistic and pessimistic criteria, then compares alpha and beta.

```
[(base) brindakulkarni@BRINDAs-MacBook-Air Othello_Code % python3 main.py random alphabeta 3

Current state, O to move:
. . . . .
. . . . .
. . . . .
. . . 0 X . . .
. . . X 0 . . .
. . . . .
. . . . .
. . . . .

0: Player O to 2,4
1: Player O to 3,5
2: Player O to 4,2
3: Player O to 5,3
Random move: 2
Player O to 4,2

Current state, X to move:
. . . . .
. . . . .
. . . . .
. . . 0 X . . .
. . 0 0 0 . . .
. . . . .
. . . . .
. . . . .

0: Player X to 3,2
1: Player X to 5,2
2: Player X to 5,4
2
Player X to 5,4
```

....

```
Current state, O to move:
. . X X X X X
0 X X X 0 X 0 X
X 0 X 0 X 0 X X
X X 0 X X 0 0 X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X

0: Player O to 0,1
Random move: 0
Player O to 0,1

Current state, X to move:
. 0 X X X X X
0 0 0 X 0 X 0 X
X 0 X 0 X 0 X X
X X 0 X X 0 0 X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X

0: Player X to 0,0
0
Player X to 0,0

*** Final winner: X ***
X X X X X X X X
X X 0 X 0 X 0 X
X 0 X 0 X 0 X X
X X 0 X X 0 0 X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

```
(base) brindakulkarni@BRINDAS-MacBook-Air Othello_Code % python3 main.py alphabeta random 3

Current state, 0 to move:
. . . . .
. . . . .
. . . . .
. . . 0 X . .
. . . X 0 . .
. . . . .
. . . . .
. . . . .

0: Player 0 to 2,4
1: Player 0 to 3,5
2: Player 0 to 4,2
3: Player 0 to 5,3
3
Player 0 to 5,3

Current state, X to move:
. . . . .
. . . . .
. . . . .
. . . 0 X . .
. . . 0 0 . .
. . . 0 . . .
. . . . .
. . . . .

0: Player X to 3,2
1: Player X to 5,2
2: Player X to 5,4
Random move: 0
Player X to 3,2
```

```
Current state, 0 to move:
X X . 0 X X X .
X X 0 0 0 X X X
X 0 X X X 0 X 0
X X 0 X X X 0 0
X X 0 0 X X X 0
X 0 0 0 0 0 X 0
X X X 0 X 0 X 0
X 0 0 0 0 0 0 0

0: Player 0 to 0,7
0
Player 0 to 0,7

Current state, X to move:
X X . 0 0 0 0 0
X X 0 0 0 X 0 0
X 0 X X X 0 X 0
X X 0 X X X 0 0
X X 0 0 X X X 0
X 0 0 0 0 0 X 0
X X X 0 X 0 X 0
X 0 0 0 0 0 0 0

0: Player X to 0,2
Random move: 0
Player X to 0,2

*** Final winner: 0 ***
X X X 0 0 0 0 0
X X X X 0 X 0 0
X 0 X X X 0 X 0
X X 0 X X X 0 0
X X 0 0 X X X 0
X 0 0 0 0 0 X 0
X X X 0 X 0 X 0
X 0 0 0 0 0 0 0
```

Alphabeta Pruning computes the result faster than minimax algorithm at the same depth.