CSC 215-01 Artificial Intelligence

Project 4: Solving (Wild) Tic-Tac-Toe using Minimax Search

Name	**Meghana Rao Kanneganti, Nikitha Mandhana**
ID	**302804645, 303315350**
Course Title	**CSC- 215, Artificial Intelligence**
Assignment ID	**Project 4**
Due Date	**April 24, 2023**

Problem Statement :
The project's purpose is to create a minimax algorithm for both tic-tac-toe and wild tic-tac-toe games. Using the minimax method, the player selects an index with the best chance of winning the game. In wild tic-tac-toe, the user gets to choose either "x" or "o" at every opportunity.

Methodology:
We began the study by making use of the notebooks given by the professor. Initially, a player selects its index by constructing a tree till the child node and allocating the score to each index using the minimax method to select the index with the highest likelihood of winning the game. To complete the assignment, we created the minimax algorithm.

After developing the minimax algorithm :

- In the scenario of optimal against optimal, the game always ends in a draw since both players play optimally.
- In the event of random vs optimum, the optimal player usually wins, but there are a few circumstances where the game finishes in a tie.
- In the situation of optimal vs. you, we attempted several times to play with the optimal player. When playing optimally, the game ends in a tie, however when playing randomly, the optimum player wins. We attempted to win the game, but the optimum player refused to give us that opportunity!

Wild tic-tac-toe is not the same as tic-tac-toe. We have two players in wild tic-tac-toe, and each player can select an ‘x' or a 'o' to place on the board. When we play wild tic-tac-toe with optimal Vs optimal, we obtain two outcomes: if P1 utilizes the center box, he wins, or we get a tie. When we compare random vs optimum execution, the optimal always wins. When you play optimal vs you, you win; and when we play randomly, the optimal wins.

Experimental Results and Analysis:

Tic-Tac-Toe

- **Optimal vs Optimal**

```
print ( " OPTIMAL VS OPTIMAL OUTPUT" )
print ( " Games x won = {}" .format(x_won))
print ( " Games o won = {}" .format(o_won))
print ( " Games Drawn = {}" .format(draw))
```

```
OPTIMAL VS OPTIMAL OUTPUT
Games x won= 0
Games o won= 0
Games Drawn= 50
```

- **Random vs Optimal**

```
print ( " RANDOM VS OPTIMAL OUTPUT. x = random, o = optimal " )
print ( " Games x won = {}" .format(x_won))
print ( " Games o won = {}" .format(o_won))
print ( " Games Drawn = {}" .format(draw))
```

```
RANDOM VS OPTIMAL OUTPUT. x =random, o=optimal
Games x won= 0
Games o won= 37
Games Drawn= 13
```

- **You vs Optimal**

```
o :

  o  |  1  |  o  |
-----------------
  3  |  x  |  5  |
-----------------
  6  |  7  |  x  |
-----------------

Please enter a valid cell (1, 3, 5, 6, 7): 6

x :

  o  |  1  |  o  |
-----------------
  3  |  x  |  5  |
-----------------
  x  |  7  |  x  |
-----------------

o :

  o  |  o  |  o  |
-----------------
  3  |  x  |  5  |
-----------------
  x  |  7  |  x  |
-----------------

o Wins!
]: 'o'
```

Wild Tic-Tac-Toe

- **Optimal vs Optimal**

```
print ( " P1 = OPTIMAL VS P2 = OPTIMAL OUTPUT" )
print ( " Games P1 won = {}" .format(p1_won))
print ( " Games P2 won = {}" .format(p2_won))
print ( " Games Drawn = {}" .format(draw))
```

```
P1 = OPTIMAL VS P2 = OPTIMAL OUTPUT
Games P1 won= 20
Games P2 won= 0
Games Drawn= 0
```

```
print ( " P1 = OPTIMAL VS P2 = OPTIMAL when middle element is not chosen" )
print ( " Games P1 won = {}" .format(p1_won))
print ( " Games P2 won = {}" .format(p2_won))
print ( " Games Drawn = {}" .format(draw))
```

```
P1 = OPTIMAL VS P2 = OPTIMAL when middle element is not chosen
Games P1 won= 0
Games P2 won= 0
Games Drawn= 20
```

- **Random vs Optimal**

```
print ( " P1 = RANDOM VS P2 = OPTIMAL OUTPUT" )
print ( " Games P1 won = {}" .format(p1_won))
print ( " Games P2 won = {}" .format(p2_won))
print ( " Games Drawn = {}" .format(draw))
```

```
P1 = RANDOM VS P2 = OPTIMAL OUTPUT
Games P1 won= 0
Games P2 won= 18
Games Drawn= 2
```

- **You vs Optimal**

```
Please enter a valid cell (1, 2, 3, 5, 6, 7, 8): 5
Please enter a valid character (x, o): 0
Please enter a valid character (x, o): x

P1 :

  x  |  1  |  2  |
-----------------
  3  |  x  |  x  |
-----------------
  6  |  7  |  8  |
-----------------

P2 :

  x  |  1  |  2  |
-----------------
  x  |  x  |  x  |
-----------------
  6  |  7  |  8  |
-----------------

P2 Wins!

'P2'
```

Results :

Name	Optimal vs Optimal (Time in seconds)	Random vs Optimal (Time in seconds)
Tic-Tac-Toe (Without pruning)	0.6581	0.1276
Wild-Tic-Tac-Toe (Without pruning)	48.0203	5.9372
Wild-Tic-Tac-Toe (With pruning)	0.2972	0.1192

Task Division and Project Reflection:

We both have worked together on the implementation of Minmax Scalar function by putting in all our thoughts and probabilities together. We came across a few challenges such as while dealing with the implementation of tic-tac-toe we had appended current with values first later realized we have to append with lists. In wild tic-tac-toe we had difficulty how we have to give choice of x and o on each turn. We gained some knowledge of problem-solving ability and also how to develop a game like tic-tac-toe and a modified wild tic-tac toe.

Additional Features:

We have implemented a more efficient AI using alpha-beta pruning for the Wild tic tac toe and we have seen a great improvement in terms of performance by reducing the number of nodes needed for evaluation in the minmax algorithm. The algorithm maintains two values, alpha and beta, which represent the best score that the maximizing player can achieve and the best score that the minimizing player can achieve, respectively.