CSCI 6511 Project 1 Zhihan Jiang G26063532

Part 1 – Tic Tac Toe

File: ttt.py

Execution:

$ python ttt.py

Description:

The program itself is a computer vs computer tic tac toe solver.

Two players are noted as black with a stone-value of 1 and white with a stone-value of 4.

The program would generate the history of a game (every state the stones on the board were placed) as well as its result which is always expected to be a 'DRAW' since both players are applying strongest strategy.

The core of this problem is to implement the strongest strategy. My strategy applied here can be summarized as follows:

Always pick the highest available from the following queue.

1. **Block:** Check 9 rows (3 vertical, 3 horizontal, 2 diagonal), if any row has and only has two of my stone, complete the row
2. **Counter Block:** Check 9 rows, if any row has and only has two of opponents' stone, complete the row
3. **Flock:** for all empty placed on the board, find one of them so that we could create two rows with, an only with two of my stones.
4. **Counter Flock:** for all empty places on the board, find one of them so that the opponent could create a Flock, place my stone there to prevent that. But before I place my stone, I must check if it would create a situation where my opponent could flock
5. **Place-at-Center:** as the name suggested
6. **Place-at-Opposite-Corner:** as the name suggested
7. **Place-at-Available-Corner:** as the name suggested
8. **Place-at-Mid-of-Edge:** as the name suggested

Part 2 – Maze

File: maze.py, maze.txt

Execution:

$ python maze.py

Description:

The logic is as following:

If any of the two given inputs is 1, return false. Else, call DFS algorithm to perform check:

Outputs:

Start Point	End Point	Result
(1, 34)	(15, 47)	Y
(1, 2)	(3, 39)	Y
(0, 0)	(3, 77)	N
(1, 75)	(8, 79)	N (End Point is 1)
(1, 75)	(39, 40)	N (End Point is 1)