Tic – Tac – Toe Lab

AI – 30 Nov ‘18

Part 1: Due M, Dec 3

1. Implement an $N \times N$ Tic-Tac-Toe board as a length N^2 string where $N = 3$. Use tokens ‘X’ and ‘O’ to indicate the two players, and we will adopt the convention that ‘X’ always goes first. Players alternate turns and may play to any unfilled square. The game is over when there are N tokens of the same type in any row, column, or diagonal, or when all the spaces are full (ie. after N^2 turns).

2. For any board, determine whether the game is over, and if so whether X wins, O wins, or there is a tie. If the game is not over, determine the set of all possible moves remaining.

3. What is the total number of distinct games that may be played? A game is defined by the sequence of positions to which tokens are played. Games are distinct if the sequence of tokens is different, even if the final board state is the same. An upper bound is 9!

4. Determine the final number of board states broken down by category: How many distinct drawn boards are there? How many final board states are there where X has won in 5, in 7, and in 9 steps, respectively. How many final board states are there where O has won in 6 and 8 steps, respectively.

Analysis: Drawn boards and boards where X wins in 9 steps both have 5 X and 4 O tokens. The sum of their numbers must be no more than the selection of 5 objects from among 9: $\binom{9}{4} = \frac{9\cdot8\cdot7\cdot6}{4\cdot3\cdot2} = 126$. This approach bounds the number of 8 step O wins, as it’s less than selecting one item (the empty square) from 9, and then selecting 4 from the remaining 8: $\binom{9}{1,4} = \frac{9!}{1!\cdot4!\cdot4!} = 630$. The number of 7 step X wins is less than $\binom{9}{2,4} = \frac{9!}{2!\cdot4!\cdot3!} = 1260$.

Part 2: Rough draft due W, Dec 5; Final version due F, Dec 7

Have your script play a tic-tac-toe game with a human. The input to your script will be either a board or a token or neither. If it gets no token, then the script decides which token it will use. If it gets a token, that is the token that the human opponent will be using. In these two cases, play starts with an empty board. If the script gets a board rather than a token, then it should assume that it is to make the next move, and hence its token is implied.

In all cases, the game is to be played to its conclusion.

When it is the human player’s turn, s/he will press a single key to indicate the position to which their token will be played (a number from 0 to 8). The script should respond to a key press and not require the Enter key to be pressed. In other words, don’t use `input`. If the escape key is pressed, play should terminate.

For each turn, the corresponding 2D board should be printed. In addition, at a minimum, prior to the script making a move, it should categorize (and print out such categorization) each of its possible moves as L (losing), T (tying), or W (winning), under the assumption of perfect play from both sides going forward.