COMP3411 Artificial Intelligence

Term 1, 2024

Assignment 3 – Nine-Board Tic-Tac-Toe

Due: Friday 19 April, 10 pm
Marks: 16% of final assessment

Introduction

In this assignment you will be writing an agent to play the game of Nine-Board Tic-Tac-Toe. This game is played on a 3 x 3 array of 3 x 3 Tic-Tac-Toe boards. The first move is made by placing an X in a randomly chosen cell of a randomly chosen board. After that, the two players take turns placing an O or X alternately into an empty cell of the board corresponding to the cell of the previous move. (For example, if the previous move was into the upper right corner of a board, the next move must be made into the upper right board.)

The game is won by getting three-in-a row either horizontally, vertically or diagonally in one of the nine boards. If a player is unable to make their move (because the relevant board is already full) the game ends in a draw.

Getting Started

Copy the archive `src.zip` into your own filespace and unzip it. Then type

```
cd src
make all

./servt -x -o
```

You should then see something like this:

```
 . . . | . . . | . . .
 . . . | . . . | . . .
 . . . | . . . | . . .
 ------+-------+------
 . . . | . . . | . . .
 . . . | . . . | . . .
```

```
  . . . | . . x | . . .
  ------+-------+------
  . . . | . . . | . . .
  . . . | . . . | . . .
  . . . | . . . | . . .

next move for O ?
```

You can now play Nine-Board Tic-Tac-Toe against yourself, by typing a number for each move. The cells in each board are numbered 1, 2, 3, 4, 5, 6, 7, 8, 9 as follows:

```
+-----+
|1 2 3|
|4 5 6|
|7 8 9|
+-----+
```

To play against a computer player, you need to open another terminal window (and `cd` to the `src` directory).

Type this into the first window:

```
./servt -p 12345 -x
```

This tells the server to use port `12345` for communication, and that the moves for `X` will be chosen by you, the human, typing at the keyboard. (If port `12345` is busy, choose another 5-digit number.)

You should then type this into the second window (using the same port number):

```
./randt -p 12345
```

The program `randt` simply chooses each move randomly among the available legal moves. The Python program `agent.py` behaves in exactly the same way. You can play against it by typing this into the second window:

```
python3 agent.py -p 12345
```

You can play against a somewhat more sophisticated player by typing this into the second window:

```
./lookt -p 12345
```

(If you are using a Mac, type `./lookt.mac` instead of `./lookt`)

Writing a Player

Your task is to write a program to play the game of nine-board tic-tac-toe as well as you can. Your program will receive commands from the server `(init, start(), second_move(), third_move(), last_move(), win(), loss(), draw(), end())` and must send back a single digit specifying the chosen move.
(the parameters for these commands are explained in the comments of `agent.py`)

Communication between the server and the player(s) is illustrated in this brief example:

Player X		**Server**		**Player O**
	←	init		
		init	→	
	←	start(x)		
		start(o)	→	
		second_move(6,1)	→	
			←	7
	←	third_move(6,1,7)		
9	→			
		next_move(9)	→	
			←	6
	←	next_move(6)		
5	→			
		last_move(5)	→	
	←	win(triple)		
		loss(triple)	→	
	←	end		
		end	→	

Language Options

You are free to write your player in any language you wish.

1. If you write in Python, you should submit your .py files (including `agent.py`); your program will be invoked by:

   ```
   python3 agent.py -p (port)
   ```

2. If you write in Java, you should submit your .java files (no .class files). The main file must be called `Agent.java`; your program will be invoked by:

   ```
   java Agent -p (port)
   ```

3. If you write in C or C++, You should submit your source files (no object files) as well as a Makefile which, when invoked with the command "make", will produce an executable called `agent`; your program will be invoked by:

   ```
   ./agent -p (port)
   ```

If you wish to write in some other language, let us know.

Starter Code

Two types of starter code are provided. The src directory contains a minimally functioning agent in each language which connects to the socket and plays random moves `(agent.py, Agent.java, agent.c).` The directory code/ttt contains a standalone program in each language which plays normal (single board) tic-tac-toe and chooses its moves via alpha-beta search `(ttt.py, ttt.java, ttt.c).`

Note: You are free to use some method other than alpha-beta search if you wish. The starter code is simply meant to provide you with one viable option.

Testing Your Code

To play two computer programs against each other, you may need to open three windows. For example, to play `agent` against `lookt` using port `54321,` type as follows:

```
window 1:  ./servt -p 54321
window 2:  ./agent -p 54321
window 3:  ./lookt -p 54321
```

(Whichever program connects first will play X; the other program will play O.)
You can alternatively use the shell script `playt.sh,` and provide the executables

and port number as command-line arguments. Here are some examples:

```
./playt.sh ./agent ./lookt 12345
./playt.sh "java Agent" ./lookt 12346
./playt.sh "python3 agent.py" ./lookt 12347
```

The strength of `lookt` can be adjusted by specifying a maximum search depth (default value is 9; reasonable range is 1 to 18), e.g.

```
./playt.sh "python3 agent.py" "./lookt -d 6" 31415
```

Question

At the top of your code, in a block of comments, you must provide a brief answer (one or two paragraphs) to this Question:

> Briefly describe how your program works, including any algorithms and data structures employed, and explain any design decisions you made along the way.

Groups

This assignment may be done individually, or in groups of two students. Groups are determined by an SMS field called `pair3`. Every student has initially been assigned a unique `pair3` which is `"h"` followed by their student ID number, e.g. `h1234567`.

1. If you plan to complete the assignment individually, you don't need to do anything (but, if you do create a group with only you as a member, that's ok too).
2. If you wish to form a pair, you should go to the WebCMS page and click on "Groups" in the left hand column, then click "Create". Click on the menu for "Group Type" and select "pair". After creating a group, click "Edit", search for the other member, and click "Add". WebCMS assigns a unique group ID to each group, in the form of `"g"` followed by six digits (e.g. `g012345`). We will periodically run a script to load these values into SMS.

Submission

You should submit by typing:

```
give cs3411 hw3 ...
```

Remember to include all necessary files in your submission (including the one with the answer to the Question).

You can submit as many times as you like – later submissions will overwrite earlier ones. You can check that your submission has been received by using the following command:

```
3411 classrun -check
```

The submission deadline is Friday 19 April, 10 pm.
5% penalty will be applied to the mark for every 24 hours late after the deadline, up to a maximum of 5 days (in accordance with UNSW policy).

Additional information may be found in the FAQ and will be considered as part of the specification for the project.

Questions relating to the project can also be posted to the Forum on WebCMS.

If you have a question that has not already been answered on the FAQ or the Forum, you can email it to `cs3411@cse.unsw.edu.au`

Marking scheme

- 10 marks for performance against a number of pre-defined opponents.
- 6 marks for Algorithms, Style, Comments and answer to the Question

You should always adhere to good coding practices and style. In general, a program that attempts a substantial part of the job but does that part correctly will receive more marks than one attempting to do the entire job but with many errors.

Plagiarism Policy

Your program must be entirely your own work. In addition, soliciting another person (or an AI bot) to write code for you – either in person or through the Internet – is never permitted. Generally, the copying of code already available on the Internet is also forbidden. If you find some piece of "standard" code in a textbook, or on the Internet, which you would like to adapt and incorporate into your own assignment, you must email the lecturer in charge to ask if it is permissible to do so in the particular circumstances – in which case the source would have to be acknowledged in your submission, and you would need to demonstrate that you had done a substantial amount of work for the assignment yourself. Plagiarism detection software will be used to compare all submissions pairwise and serious penalties will be applied, particularly in the case of repeat offences.

DO NOT COPY FROM OTHERS; DO NOT ALLOW ANYONE TO SEE YOUR CODE

Please refer to the UNSW Policy on Academic Integrity and Plagiarism if you require further clarification on this matter.

Good luck!
