Contents

Cmput 455 Assignment 1

Due Feb 1, 11:55pm. Submit via eClass submission link on the main eClass course page.
Late submission (with 20% deduction) Feb 3, 11:55pm. Submit via separate eClass link for late submissions.

- First, read the Assignments page for general questions about assignments.
- Also, check the Assignment 1 preview in the lecture slides.

In this assignment, you implement a random player for the game Gomoku, also called Five in a Row. You do this by modifying the **assignment 1 starter code** which is based on the `Go0` program from class, but modified to make your job easier. The main steps are:

1. Implement the rules of Gomoku, including creating a list of legal moves, detecting a win, and playing a move.
2. Write a random Gomoku player based on the Go0 player in the starter code.
3. Test your program on the public test cases, and on more tests of your own design.

Setup

1. First, make sure you have completed your Activities up to and including 3c. In these activities you set up Python 3, NumPy, and GoGui 1.5.1, install and run the Go0 and Go1 programs, do the sample session with the Go1 program.
2. Download and expand assignment1.tgz . Go to the directory `assignment1`, which contains:
   - A modified copy of the Go0 program as your starter code. Start with this code to implement your solution.
   - A file `assignment1-public-tests.gtp` with sample test cases for your program.

Rules of Gomoku

Gomoku or Five in a Row is played on a Go board. The game starts with an empty board, and players Black and White alternate to place a stone of their color on a grid point. Black goes first. The goal of the game is to create a

line of five or more stones of your own color, either horizontally, vertically or diagonally. The first player to achieve that goal wins. If neither player creates five in a row, then the game is a draw.

For more information on the game, you can see the wikipedia page. Note that many variants of the rules are discussed on that page. We use what they call **Free-style gomoku**, which defines the win as five or more stones in a row, and puts no restrictions on which moves are allowed.

Some Implementation Differences between Go and Gomoku

When implementing your solution based on the starter code, be aware of the commonalities and differences between the two games. First, the rules of Gomoku are much simpler. Checking legality of moves is simpler, there is no capturing, no suicide, and no ko rule to deal with. However, finding out whether the game is over is more complex. In Go, this is defined by consecutive passes. In Gomoku, you need to check if a player has five in a row.

GTP Commands

We will test your program using the GTP commands below. These commands are already implemented in the starter code, but the implementation either is for Go or is just a stub that returns "unknown" or similar. You only need to change or implement the commands as indicated below.

- `boardsize`

  This command sets the board size (between 5 and 19 in our tests). It also initializes the board to all empty. You will not need to change this command.

- `clear_board`

  This command re-initializes the board to all empty, and throws away all the moves played so far. You will not need to change this command.

- `play color coordinate`

  Play a move of given color on the given coordinate. You will need to change this command to follow the Gomoku rules instead of the Go rules. Also, you need to implement different error handling. See Details of the play command below.

- `gogui-analyze_commands, gogui-rules_game_id, gogui-rules_board, gogui-rules_board_size, gogui-rules_side_to_move`

  These commands provide support for the game Gomoku in the GoGui 1.5.1 interface. These commands are already implemented and you do not have to change them.

- `gogui-rules_legal_moves`

  Produce a list of all legal moves for the current player, in alphabetic order.

  Usage example (on empty 3x3 board):

  ```
  gogui-rules_legal_moves
  = a1 a2 a3 b1 b2 b3 c1 c2 c3
  ```

  You will need to change this command to use Gomoku rules instead of Go rules. See Details of the gogui-rules_legal_moves command below.

- `genmove color`

  This command generates and plays a random move. This command implements random choice from the list of moves created by `legal_moves`. You will need to change this command for end of game handling. See Details for end of game below.

- `gogui-rules_final_result`

  This command checks if the game is over and outputs one of the following game results:

  ```
  black
  white
  draw
  unknown
  ```

  See Details for the gogui-rules_final_result command below. Also see Details for end of game.

Public Test Cases

The public test cases are in file `assignment1-public-tests.gtp` in the `assignment1` directory. Most tests currently fail, since the program does not implement those commands correctly yet. See Testing Procedure below for examples.

Note that our given test cases only cover a small number of cases. For evaluation, we will use a more comprehensive set of tests.

Testing Procedure

All our tests will be on a Gomoku board of size 19x19 or smaller. As in `assignment1-public-tests.gtp`, test cases are written as text files containing a sequence of GTP commands. Test cases contain both unnumbered GTP commands to set up a position, and numbered GTP commands to do the tests. Each test is followed by the expected correct answer.

What to Submit

Submit a single tgz file called `assignment1.tgz` which contains exactly the following (and nothing else):

A single directory `assignment1` which contains all the files in your solution, namely:

1. All the files from the original `assignment1` directory, but with the python code modified to solve the assignment.
2. Your `presubmission.log` file.
3. A file `readme.txt` which lists the names and student IDs of your team. List the designated submitter first.

Do Your Own Pre-submission Test

Follow the steps below on a standard undergraduate machine, and create a text file `presubmission.log` that shows a copy of your command line presubmission testing.

1. Make sure the content of your `assignment1` folder is correct, then create your submission with a command like `tar -cvzf assignment1.tgz assignment1`
2. Use the Linux script command to log your testing session
3. Copy your `assignment1.tgz` into a new directory

4. Unpack it
5. Run `gogui-regress` with your program, using `assignment1-public-tests.gtp` as input
6. Stop `script` logging here with the command `exit`
7. Add file `presubmission.log` to your assignment1, and compress it again
8. Use `tar -tf` for a final check of your `assignment1.tgz` contents. Use the checklist below
9. If you see extra files included in your tgz file that do not belong:
    - Use tar options such as `--exclude=__pycache__` to get rid of them.
10. Fix any problems, then submit on eClass when you're ready

Marking

There will be total of 5 marks for this assignment.

- 1 mark for your file `presubmission.log` which shows the log of your correct test execution, and your readme.txt file (see below).
- 2 marks for passing our public test cases
- 2 marks for passing our private test cases

Code that does not run, or just hardcodes the public tests case by case, will receive a mark of 0. If your code needs fixes, you need to submit a revised version before the late submission deadline. TA will not attempt to fix your code under any circumstances. Use the discussion forum or consult the teaching team in case of questions.

Details - Read Them All!

- Where to implement things: Implement all your code in the `assignment1` folder. You may create new files within this folder, but it might not be necessary for this assignment.
- Our tests will only use legal GTP commands with meaningful arguments, except for testing the error handling in the play command. You do not need to implement any other error checking.

Details of the play command

Attempt to play a move of given color on the given coordinate. First check if the move is legal according to the rules of Gomoku. If yes, then play it and update the board according to the rules. If no, then output an error message and do not change the board. The format of an error message is:

```
illegal move: "copy of the input argument(s)" reason
```

The possible reasons for error messages are listed below. Check them in the order given and output the first error only:

- wrong color
- wrong coordinate
- occupied

In the starter code, this command is implemented for Go, not for Gomoku.

Details of the gogui-rules_legal_moves command

If the game is over, return an empty list. Otherwise, return a list of all empty points on the board in sorted order.

In the starter code, this command always returns an empty list.

The Go0/Go1 engine also implements a legal_moves command for the game of Go. You can just ignore that.

Details for the gogui-rules_final_result command

Here, one of the first three answers `black, white, draw` should only be given if the game is over. The fourth answer `unknown` should always be returned if this command is called for a board where the game is not over yet.

In the starter code, this command just returns `unknown`.

Details for end of game

There are two different cases which end the game. First, if a player creates a row of five or more stones, that player wins. Second, if the board is completely filled with stones, but no player has five in a row, then the game is a draw. The `gogui-rules_final_result` command should output either the color of the winner, or draw.

The `genmove` command at the end of a game: Before the end of a game, `genmove` will play a random legal move. If the game is over because the opponent just made five in a row, the response of `genmove` should be:

```
resign
```

If the game is over because the board is full and it is a draw, the response should be

```
pass
```

This is the only circumstance in which pass should be generated. The reason for this specification is that we can more easily use the tools for Go in this way.

In the starter code, `genmove` is implemented for Go, not for Gomoku.

Created: Dec 27, 2018 Last modified: Jan 8, 2021

Martin Müller and Ting-Han Wei
