CONTENTS

# Cmput 496 Assignment 1

Due Jan 29, 11:55pm. Submit via eClass submission link on main eClass course page.
Late submission (with 20% deduction) Jan 31, 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 near the beginning of the Lecture 2 slides.

In this assignment, you extend a copy of our `Go1` program from class by adding a scoring function. A scoring function is called after a game ends. It determines who won the game, and by how many points. The main steps are:

1. Find connected components of empty points on the given Go board.
2. For each component, determine whether it is black territory, white territory, or neutral space.
3. Count how many points each player has achieved.
4. Combine the counts of both players, plus the komi, to compute the score of the game.
5. Implement a new GTP command `score` for the Go1 player, which computes the score as above, and prints it in a human readable format as specified below.

## Setup

1. First, make sure you have completed your activities 1b, 1c, 1d and 1e to set up Python 3 and NumPy, GoGui, the initial cmput_496 code, and do the sample session with the Go1 program.
2. Go to directory `assignment1` within your `cmput_496` code directory. `assignment1` contains:
   - An exact copy of the Go1 program. Modify this copy to implement your solution.
   - A file `assignment1-public-tests.gtp` with sample test cases for your scoring function

## Public Test Cases

The public test cases are in file `assignment1-public-tests.gtp` in the `assignment1` directory. All the tests currently fail, since the Go1 program does not yet contain a scoring function. Your job is to implement it. 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 Go board of size 9x9 or smaller, and will be checked with `gogui-regress` as explained in Activity 1e. 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.

## GTP COMMANDS FOR SETUP

We will set up test positions for your program using the GTP commands below. All these commands are already implemented in Go1. Do not change them.

- `boardsize, clear_board`

  These commands set the board size (between 2 and 9 in our tests) and initialize the board to all empty.

- `play color coordinate`

  Play a move of given color on the given coordinate. In our tests, we will only use legal moves.

- `komi value`

  Set the komi to a given value.

## GTP COMMAND FOR SCORING

Your main task in this assignment is to implement a new GTP command for scoring the game. The format of the new command is:

```
score
```

The format of the reply is of one of the following three cases:

1. If Black won the game:

   ```
   = B+n
   ```

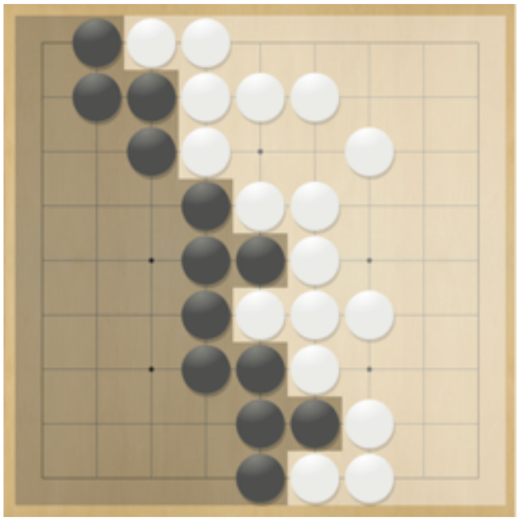
   Where n is the score from Black's point of view.
2. If White won the game:

   ```
   = W+n
   ```

   Where n is the score from White's point of view.
3. If the game is a draw (jigo):

   ```
   = 0
   ```

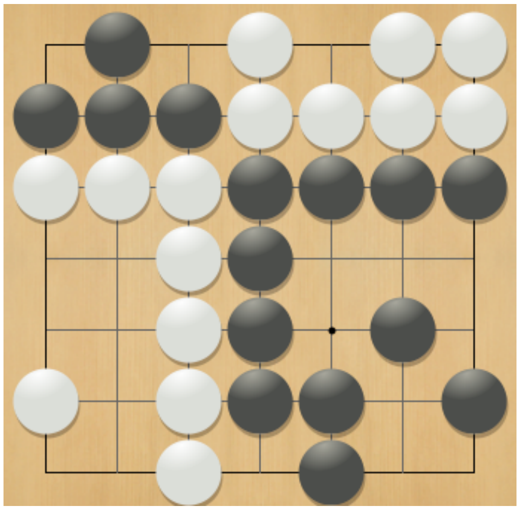
## EXAMPLES WITH PICTURES



This final game position is the scoring example from Lecture 1. Black has 37 points. White has 44 points on the board, plus the komi of 7.5 = 51.5 total. White won by 14.5 points, so your program should react to the score command as follows:

```
score
= W+14.5
```

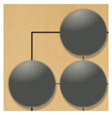
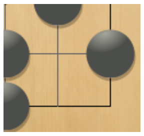
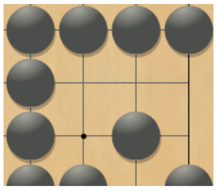
## More Detailed Example



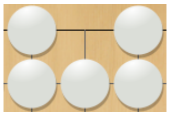
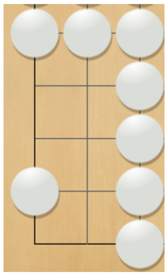
This is a more detailed example with neutral points. One such point, in the top row, is due to a seki (coexistence). The other neutral point, at the middle of the bottom row, was not filled by either player by mistake. Both passed, so the game ended at this point.

How to count:

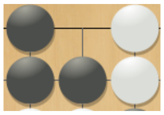
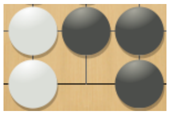
- First, count all stones on the board as points for the player who played them. Black has 15 stones and White also has 15.
- Next, look at all connected sets of empty points (there are seven in this example) and determine whether they are Black territory, White territory, or neutral. See the cases below.
  Three areas are surrounded by Black stones. The sizes are 1, 3 and 5. So Black has 9 points of territory.

 (top left) +  (bottom right) +  (right)

- Two areas are surrounded by White stones. White has 1 + 7 = 8 points of territory.

 (top right) +  (bottom left)

- Two areas have neighbors of both colors. These are neutral areas and are not counted.

 (top) +  (bottom)

- Assume the komi is 0.5.
  Black has a total of 15 + 9 points (stones + territory).
  White has a total of 15 + 8 + 0.5 points (stones + territory + komi).
  Score 24 - 23.5 = 0.5.
- score command and program response:

```
score
= B+0.5
```

## What to Submit

Submit a single tgz file called `a1.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. In your `cmput_496` directory, make sure the content of `assignment1` folder is correct, then create your submission with a command like `tar -cvzf a1.tgz assignment1`
2. If you see extra files included in your tgz file that do not belong, you can use tar options such as `--exclude=.DS_Store` or `--exclude=__pycache__` to get rid of them.
3. Use the Linux script command to log your testing session
4. Copy your `a1.tgz` into a new directory

5. Unpack it
6. Add the `util` directory from cmput_496
7. Run `gogui-regress` with your program, using `assignment1-public-tests.gtp` as input
8. Stop `script` logging here with the command `exit`
9. Add file `presubmission.log` to your assignment1, and compress it again
10. Use `tar -tf` for a final check of your `a1.tgz` contents. Use the checklist below
11. 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 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.

  The GTP command which implements your score function should go into the file `gtp_connection_go1.py` within your `assignment1` folder. You can follow the provided function `hello_cmd` as an example for how to write a GTP command. Like `hello_cmd`, your score command needs to register itself in the `__init__` function of `GtpConnectionGo1`, so the GTP interpreter knows about it.

  The simplest approach is to put your code which implements the scoring algorithm into the same file.

- **Never modify code in other directories** such as util. You are not even allowed to submit that. We will test your modified `assignment1` directory together with the original `util`.
- Setting and accessing komi: Komi is set at initialization time, and through the GTP command `komi`. You can assume that each test case will set the komi through the GTP command.

  The current value of komi can be read from the field self.komi in `class GtpConnectionGo1` (which inherits the field from `GtpConnection`).

- The score function can be called multiple times within one test file.
- Your code may not modify the player's board. Copy it first if you want to modify it.
- Our tests will only use legal GTP commands with meaningful arguments. For example, moves played will be legal, komi will be a reasonable number, and the board size will be a number in range from 2 to 9 inclusive. You do not need to implement any error checking.
- \+ instead of + in the `.gtp` file: Did you wonder why the `.gtp` file uses \+ to match the '+' character? The reason is that replies in `gogui-regress` use a limited version of regular expression syntax. Characters such as + and * have special meaning and need to be escaped with the \.

Created: Dec 9, 2017 Last modified: Jan 8, 2018

*Martin Müller*