# TCG HW2 Description

Due date: 14:20 (UTC+8), December 21, 2017

## HW2 Description

- Implement the $9 \times 9$ GO using Monte-Carlo Tree Search.
- For rule of go, see slide9.pdf from page 3 to page 9.

## HW2 Requirements

- Implement *UCB* score.
- Implement *UCT*.
- Print a *principal variation* of your *UCT* tree to a file after each genmove command.
- Implement *progressive pruning*.
- Bonus: Other improved techniques.
- Provide your Makefile or specfied how to compile your codes (Step-by-Step) in the report.
- The excutable produced by makefile must name by your student id.
- TA will hold a round robin tournament on a linux machine, so your code must be compiled and run under linux. And there will be about $30^2$ games, so each program is limited with one thread.

## Solution Package

- Submit page: `http://w.csie.org/~tcg/2017/`
- Package structure:
  - Your ID [R05xxxxxx/B02xxxxxx/...]
    - code // A folder contains all your codes and makefile
    - report.pdf // Your report
- Compress your folder into a "zip" file
- Due to server limitation, the file size is restricted to 2M bytes

## Grading policy

- Basic grading policy:
  - Implement the basic requirement.
  - Defeat the random version program.
  - Your report.
- Advance grading policy:
  - Other enhencement
  - Program ranking in the whole class

## Last Year Othello 8x8 Tournament Result

| rank | sid | score |
|---|---|---|
| 1 | R05921058 | 58 |
| 2 | R04944002 | 48 |
| 3 | R04922030 | 47 |
| 4 | B02902105 | 43 |
| 5 | B02902011 | 40 |
| 6 | R04922024 | 39 |
| 7 | B01902112 | 33 |
| 8 | R05922089 | 32 |
| 9 | B02705021 | 31 |
| 10 | R03922164 | 31 |

| rank | Bonus() |
|---|---|
| $[1, 1]$ | $5$ |
| $[2, 3]$ | $4$ |
| $[4, 6]$ | $3$ |
| $[7, 10]$ | $2$ |
| $[11, 33]$ | $0$ |

## Principal variation

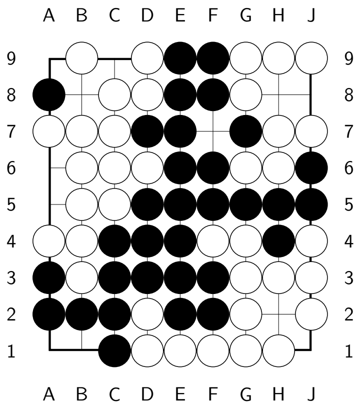
The Principal variation (PV) is a sequence of moves that programs consider best and therefore expect to be played.

1. (m0 = best move at root, winrate at root, simiulation times at root)
2. (m1 = best move after m0 played, winrate after m0 played, simiulation times after m0 played)
3. (m2 = best move after m0,m1 played, winrate after m0,m1 played, simiulation times after m0,m1 played)
4. ...
5. until reach leaf of search tree

`https://en.wikipedia.org/wiki/Variation_(game_tree)`

# Principal variation example

**white to play, ko at F7**



Principal variation

1. w A1, win rate = $0.995$, sim = $123467$
2. b H8, win rate = $0.005$, sim = $107122$
3. w F7, win rate = $0.996$, sim = $102345$
4. b B1, win rate = $0.004$, sim = $71221$
5. w A1, win rate = $0.997$, sim = $62345$
6. $\dots$

## Protocol and Notes

- **G**o **T**ext **P**rotocol (GTP): often run with graphic user interface
  - run with option -display to auto display board
  - run with option -nodisplay or no option to disable auto display board.
- Note:
  - Your code will be tested with GTP version
  - All debug message should only output to **file** or **standard error**
  - Do not change the output format of GTP function
  - The time limit is **10 second** per move.

## Basic Commands for GTP version

- These functions is based on the Go Text Protocol
    - Reference: `http://www.lysator.liu.se/~gunnar/gtp/`
- Implemented command
    - protocol_version // Display the version of current protocol
    - name // Show the program name
    - version // Show the version of program
    - known_command // Ask program knows command or not.
    - list_commands // Show the list of all known commands
    - boardsize // Set the board size, currently only 9 is legal.
    - clear_board // Reset the board state.
    - komi // Set the number of komi (e.g. 6.5, 7, 7.5)
    - play // Play White/Black stone on game board
    - genmove // Call the engine to generate next move.
    - undo // Back to previous move
    - showboard // Display the current game board
    - quit // End the program

## Description of GTP Command

- boardsize *size*
  - Set the boardsize as *size*
  - The template code only support $size = 9$
- clear_board
  - clear the gameboard
- komi *num*
  - set the komi as *num*, default is 7.
- play $b/w$ [ABCDEFGHJ][1-9]
  - like put, put $b/w$'s sonte at column [A-J], row [1-9]
  - row id is down to top.
  - column id is left to right.
- genmove $b/w$
  - generate $b/w$'s move
- undo
  - undo one move
- showboard
  - show current gameboard

## About Default State of GTP Engine

The document of GTP says “There is no default state for any state variable. When first started, the engine may set these as it likes. A controller which has some specific opinion about these values must set them explicitly with the appropriate commands, including clearing the board”.
This means user needs to command clear_board to confirm the state of a GTP engine, so do not tell TA that the board is not updated correctly without clear_board.

## About the Template Code

- The variable in the template code is naming as follows:
    - Define constant: all upper letters.
        - BOARDSIZE, BOUNDARYSIZE.
    - Array: Initial character is upper letter.
        - Board, MoveList
    - Non-array variable: all letter is lower case
        - There are two exceptions, X and Y.
        - game_length, num_legal_move

## Board Structure: Board[BOUNDARYSIZE][BOUNDARYSIZE]

```
      0  1  2  3  4  5  6  7  8  9  10
  0   *  *  *  *  *  *  *  *  *  *  *
  1   *  .  .  .  .  .  .  .  .  .  *   9
  2   *  .  .  .  .  .  .  .  .  .  *   8
  3   *  .  .  +  .  .  .  +  .  .  *   7
  4   *  .  .  .  .  .  .  .  .  .  *   6
  5   *  .  .  .  .  +  .  .  .  .  *   5
  6   *  .  .  .  .  .  .  .  .  .  *   4
  7   *  .  .  +  .  .  .  +  .  .  *   3
  8   *  .  .  .  .  .  .  .  .  .  *   2
  9   *  .  .  .  .  .  .  .  .  .  *   1
 10   *  *  *  *  *  *  *  *  *  *  *
         A  B  C  D  E  F  G  H  J
```

- There is no I in the column index.
- BOUNDARYSIZE: 11
- BOARDSIZE: 9
- Board[i][j] is (x,y) = (j, 10-i) in the game board

## Genmove Function

- *gen_legal_move*(Board, turn, game_length, GameRecord, MoveList)
    - generate all the legal move
    - return the number of legal moves.
- *rand_pick_move*(*num_legal_moves*, *MoveList*)
    - randomly pick one legal move
    - return the selected move.
    - **You should replace this function.**
- *do_move*(*Board*, *turn*, *move*)
    - update the current board with “move“

## *gen_legal_move* Function

- For each empty intersection
    - Check if the empty intersection is a legal move
    - Check if the legal move will result in a repeat board
    - Add the move to move list.
        - each move is a 3 digit integers *eij*
        - *e* denote this is a capture move (1) or not (0).
        - *ij* denote the location of Board[i][j]
        - e.g. $123$: put stone in Board[2][3] is a capture move.
        - e.g. $056$: put stone in Board[5][6] is not a capture move.

## Function for Checking Legal Move

- $count\_neighboorhood\_state(Board, X, Y, turn, *empt, *self, *oppo, *boun, NeighboorhoodState)$
  - return the number of
    - Empty intersection
    - Self intersection
    - Opponent intersection
    - Boundary intersection
  - Record the state of each neighborhood in NeighboorhoodState.
- $count\_liberty(X, Y, Board, Liberties)$
  - count the number of liberties in each direction's string.
  - The result is saved in Liberties.
  - Using DFS method.

## Legal Move

- A move is legal if
  - At least one neighborhood intersection is empty.
  - One of the self string has more than one liberty.
    - And it's not a self-eye.
  - One of the opponent string has only one liberty.

## Do the move

- Update the Board with
  - play Black/White [ABCDEFGHJ][123456789]
- update_board(Board, X, Y, turn)
  - put turn's piece in (X, Y)
  - will not check if (X, Y) is a legal move.
- update_board_check(Board, X, Y, turn)
  - put turn's piece in (X, Y)
  - will check if (X, Y) is a legal move.
  - return 1 if (X, Y) is a legal move
  - return 0 if (X, Y) is a inlegal move

## Avoid the repeat board

- GameRecord[][][]
- game_length
- Check all boards in the GameRecord.

## Result Counting

- final_score(*Board*)
  - black area: black stones + black eyes
  - white area: white stones + white eyes
  - result: black area - white area
- final result = final_score - komi
  - $> 0$: B+[result]
  - $= 0$: 0 (draw)
  - $< 0$: W+[-result]

## Introduction of GoGui

- Homepage: `http://gogui.sourceforge.net/`
  - You can find the download link here.
- Run a computer selfplay
  1. Game ⇒ game size ⇒ 9
  2. Game ⇒ Game info ⇒ Komi 7
  3. Program ⇒ New Program
     - Command: the path to your execution file.
  4. Program ⇒ Attached ⇒ Your program
  5. Game ⇒ Computer Color ⇒ Both

## Selfplay Via GoGui

- gogui-twogtp
  - -white [white program name]
  - -black [black program name]
  - -games [number of games]
  - -sgffile [filename]
    - filename.dat: statistic result
    - filename-0.sgf - filename-[N-1].sgf
- Example: gogui-twogtp -white white.exe -black black.exe -games 10 -alternate -size 9 -komi 7 -verbose -sgffile record_name -auto
- Using Gogui to display: gogui -program "gogui-twogtp -white white.exe -black black.exe -games 10 -alternate -size 9 -komi 7 -sgffile record_name" -size 9 -computer-both -auto
- For more detail see `https://www.mankier.com/1/gogui-twogtp`

## About -games and -sgffile in gogui-twogtp

- -games N means gogui-twogtp will play N games.
- -sgffile [filename] means the result will be saved with prefix "filename"
- If filename.data exists and contains k games.
  - If N $\leq$ k, then gogui-twogtp will do nothing.
  - If N $>$ k, then gogui-twogtp will play exact N-k games.
  - If you want to play exact N games:
    - remove filename.data
    - or add option -force to overwrite filename.dat
- Files with extension sgf are the game record of each game.
  - Index from 0 to N-1
  - Can be opened by gogui
    - File $\Rightarrow$ Open.

## Other notification

- When each game start, the protocol will call function *reset*(*Board*).
  - Beware to initial all your self data structure here.
- Gogui can show the graphic user interface via Xming and pietty.
  - Start Xming
  - pietty ⇒ putty mode
  - session: host name or ip
  - Connection ⇒ SSH ⇒ X11:
    - Select "Enable X11 forwarding"
    - X display location: 127.0.0.1:0
  - Open
- c-library std::clock is cpu time. Tournament uses real time wall clock to judge. Take a look at c++11 <chrono> library.
- c rand() vs c++11 <random>
- Other gui supports GTP: sabaki
  `http://sabaki.yichuanshen.de/`