CS 165A – Artificial Intelligence, Spring 2020

Machine Problem 2

(100 points)

Due *Thursday, June 4, 2020 11:59pm*

Notes:

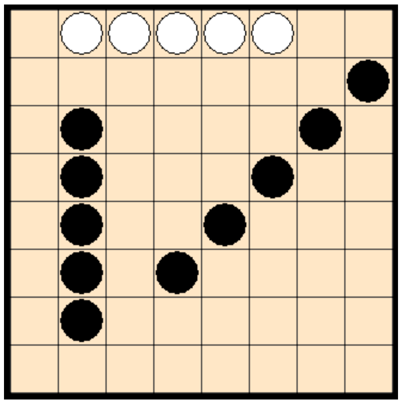
- Make sure to read the "Policy on Academic Integrity" on the course syllabus.
- Any updates or corrections will be posted on Piazza.
- You must work individually for this assignment.
- Each student must turn in their report and code electronically.
- Responsible TA for Machine Problem 2: Karl Wang (changhai_wang@ucsb.edu)
- Visit the course website for tournament status

1 Problem Definition

You are going to implement a program that plays Five-in-a-Row (also called Gobang or Gomoku) with a human opponent. Your program should be able to search for the best move so that it ends up winning over a human opponent (or at least make the match very challenging). Towards this end, you should explore the **Minimax** algorithm and other adversarial search algorithm improvements, like alpha-beta pruning, in order to quickly calculate the program's next move and eventually win.

2 Five-in-a-Row

Five-in-a-Row is a strategy board game for two players, usually played with Go pieces (black and white stones) on a go board. Black plays first and players alternate in placing a stone of their color in an empty cell. The winner is the first player to get an unbroken row of **at least** five stones horizontally, vertically, or diagonally.



You can read this Wikipedia article on Five-in-a-Row: https://en.wikipedia.org/wiki/Gomoku

Finally, there are many websites online where you can practice playing Five-in-a-Row for free.

3 Technical Details

3.1 Rules

The black player places the first piece. Then each player take turn placing pieces. The first player to form a chain of at least 5 wins. If board is full and there is no chain greater than 5, then it ends in a tie.

Your program should not take more than **30 seconds** to decide each of its move. The human player is allowed as much time as they need for their move. Running out of time means an automatic loss. Note that the grading platform is the Linux machines in CSIL. Make sure that your program does not exceed the 30 seconds per move rule on these machines (it might be running faster in your laptops).

3.2 Input

Your program should be able to process the following two optional command line arguments:

- `-n <size>`: We will allow sizes 5*5 to 26*26. If n is not specified, the default value should be $11*11$.
- `-l`: If this option is specified, the human opponent is going to play with the light colors. If the option is not specified, the human player will be playing with the dark colors. In both cases, the dark players move first.

Once the execution begins, your program should be able to interact with the human player through a command line interface. The command line interface should support only one command:

- `<move>`: This command is valid when it is the turn of the human player. `<move>` indicates the player's choice of next move, and should be represented as a `<letter><number>` pair (without spaces in between) where the `<letter>` indicates the column (a, b, c, …) and `<number>` indicates the row (1, 2, 3, …) of the board. If the player tries an invalid move the program should display an error message and prompt the player for a new move again.

3.3 Output

Whenever your program makes a move, and whenever it successfully reads in a move made by its opponent, it must print that move to the standard output in the following format:

```
Move played: <move>
```

The referee program will look for this line in your program's output to determine your program's next move. You may additionally print other outputs, such as ASCII art representation of the current board, or declare the winner at the end of the game. These outputs will be ignored by the referee. You can check the outputs of the provided programs to see how your output should look like in order to be compatible with the referee script.

3.4 Implementation Restrictions

You may use C/C++, Java, or Python3 for this programming assignment. You can use C/C++, Java, and Python's standard library. You are also allowed to use libraries that are installed on CSIL for every user, except those that implements machine learning algorithms, such as scikit-learn. You are not allowed to use any framework or library that is not already installed on CSIL. Also keep in mind that your submission can't exceed 10MBs.

4 Makefile and Executable

If you are writing in C or C++, then you should write a Makefile that compiles your binary. Name the binary "`gobang`". If your choice of language is Python or Java, then you need to provide a wrapper script that executes your code. Name this script "`gobang`". For python you might have the following simple script:

```
#!/bin/bash
```

```
python3 gobang.py $@
```
and for Java:
```
#!/bin/bash
java gobang $@
```
The “`$@`” in these scripts allow command line arguments to be directly passed into your code.

The grading script will run your program like so:
```
make
./gobang <arguments>
```

5 Judging

You will be provided a ‘*referee*’ script that will allow for two programs to compete. Technically, each program will have no knowledge that it plays with another program, they will both think that they compete with a human. The referee script will be redirecting the moves of each program to the command line interface of the other program to simulate these human players. Using the referee, you should be able to test your program with its own self as the opponent. The referee will also enable having a *tournament* to find which implementation performs the best, given 30 seconds per move.

In order to render your program compatible with the referee you need to have an executable and abide by the specific input and output rules. Your program must be compatible with the referee for grading as well. Test your code using the referee script to ensure compatibility.

6 Report Preparation

As for the report, it should be between 1 and 2 pages in length (no more than 2 pages) with at least 11pt font size and should consist of at least the following sections:
- **Architecture:** Brief explanation of your code architecture (describe the classes and basic functionality)
- **Search:** How you search for the best move (which algorithm you use, what heuristics, optimizations, etc)
- **Challenges:** The challenges you faced and how you solved them
- **Weaknesses:** Weaknesses in your method (you cannot say the method is without flaws) and suggest ways to overcome them.

Note that the most important part is the Search section. There should be only one pdf report which includes all the above (no additional readme file).

7 What and How to Submit

Once you are finished developing your code, copy all necessary source files (including header files, source code files, the pdf report, Makefile, etc.) into a directory named “mp2” and submit the directory with the `turnin` command. Do not submit files individually. Note that all directory and file names are case sensitive. For this assignment you need to issue the following command:
```
turnin mp2@changhai_wang mp2
```

7.1 Detailed Submission Guidelines

- C/C++: Include a "`Makefile`" which builds the program by default (e.g. typing "`make`" in the mp2 directory should build the `gobang` executable program), your source code files (.cpp, .c, .hpp), your header files ".h" if you have any, your PDF report, and any other files that are needed to compile and run your code successfully.
- Java: Include your source code file ".java", class files ".class", an executable wrapper script named "`gobang`" that executes your java code, your PDF report, and any other files that are needed to build and run your code successfully.
- Python: Include your source code files ".py", an executable wrapper script named "`gobang`" that executes your python code, your PDF report, and any other files that are needed to run your code successfully.

In all cases, regardless of language, the grader should be able to just run "`make`" (if code is in C/C++) and then execute the "`gobang`" executable.

8 Grading

Grade Breakdown:
- **20%** Complete Report
- **10%** includes Makefile and/or executable that matches specification.
- **10%** accepts command line arguments; produces required outputs; no segfaults, exceptions, or logic bugs during gameplay
- **60%** Gameplay performance (your program beats our program for specific values of n and a maximum of 30 seconds per move)

Plagiarism Warning: We are going to use software to check plagiarism. You are expected to finish the project by yourself without using any code from others.

8.1 Performance

You will be graded based on your program's ability to win other AI programs! The main weight of the grade (60%) will be based on your program's performance so you should make your search is as optimal as possible. You will get:
- 0/60% if your program fails to beat an adversary that randomly chooses their next move
- 30/60% if your program beats an adversary that randomly chooses their next move
- 40/60% if your program implements the basic minimax algorithm with depth >= 2
- 50/60% if your program also applies alpha-beta pruning.
- 60/60% if your program beats the baseline program (minimax with depth=2 and alpha-beta pruning)

The baseline implementation is provided to you so that you can test your own program before the final submission.

9 Tournaments!

Participation in the Tournaments is optional but can gain additional credit (and fame!). There will be two kinds of tournaments, a pre-submission tournament and a post-submission tournament. To enter both tournaments, you need to submit a text file named "`TOURNAMENT`". This file should contain a number that specifies your preferred board size. There should be nothing else in this file besides the number. Submissions without this file will not participate in either tournaments.

9.1 Pre-submission tournament

The pre-submission tournament will have a defending champion. Every time you submit your code, if your submission contains the text file TOURNAMENT, your program will be tested versus the current defending champion. If you win, you become the defending champion. If you do not win, the defending champion will remain the same. To become the defending champion, you also need to beat our simple AI programs which form the baseline, so you can't become the defending champion just by implementing random moves. For every full day you spend as the defending champion you get 2% extra credit (plus the FAME of being the defending champion). You will be emailed the results of your submission once the match is over. Matches will be performed at least once per day. Do not flood the turnin submission system or you will get banned from the tournament (only submit once you have the results of your previous submission).

9.2 Post-submission tournament

The post-submission tournament will be a tournament between all final submissions that contain the text file TOURNAMENT. The program that wins the tournament gets 50% extra credit. The two runner-ups get 30% and 20% (for second and third position).

9.3 Gobang match setup

A single match consists of 3 games. One game with board size of 11, one game where the board size if specified by one player, and one game where the board size is specified by the other player. If you don't specify a board size then it will be set to 11 by default. Keep in mind that in the game where your specified size is used, you will be playing as the Light player. In the first game where the size is 11 the order will be random.

9.4 Winning a match

The winner of a match is the player that has at least two wins (best out of three). In case of a tie in a game during the pre-submission tournament the defending champion gets the win. In case of a tie during the post-submission tournament the winner will be the player that took the least total time to submit their moves during the game. If you tie with our programs during grading it will count as your win.