CS 165A – Artificial Intelligence, Winter 2017

Machine Problem 2

(100 points)

Due *Tuesday, March 14 2017, 11:59pm*

Notes:

- Make sure to re-read the "Policy on Academic Integrity" on the course syllabus.
- Any updates or corrections will be posted on the Announcements page in web page of the course, http://www.cs.ucsb.edu/~xyan/classes/CS165A-2017winter/announcements.html, so check there occasionally.
- You have to work individually for this assignment.
- Each student must turn in their report and code electronically.
- Responsible TA for Machine Problem 2: Yan Tang yantang@cs.ucsb.edu
- Visit **http://cs.ucsb.edu/~cs165a** for tournament status

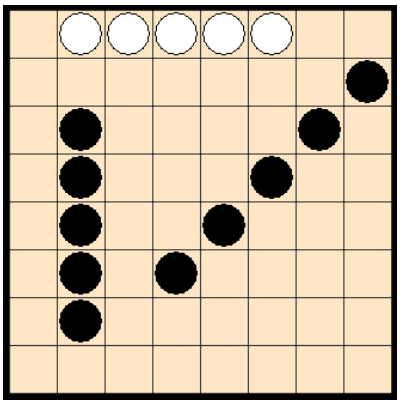
---

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 (Gobang)

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 also watch this video for more information on playing Five-in-a-Row: https://www.youtube.com/watch?v=402iV4RfTM4

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

3. Technical Details

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 just display an error message and prompt the player for a new move again.

After the human player has specified their move, your program should start calculating its next move and play it. Then it is the human player's turn again. During the game, your program should display the board and update it after every move. Your program should not take more than **30 seconds** to decide on its new move. The human player is allowed as much time as they need for their move.

You will also be provided a '*referee*' script that will allow for two programs to compete with each other! 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 (more information on the tournament soon). Note: In order to render your program compatible with the referee you need to have an executable binary and abide by specific output rules. Your program must be compatible with the referee for grading as well.

Executable: If you are writing in C or C++ then you should get a binary by default after compiling (using a makefile). However, if you choice of language is python or Java then you also need to provide an executable wrapper script that just executes your code. Name this script "Gobang". For python you might have the following simple script:

```
#!/bin/bash
python Gobang.py $@
```

and for Java:

```
#!/bin/bash
java Gobang $@
```

The above scripts will just execute your python or Java code by simply running the wrapper scripts like this:

```
./Gobang
```

These scripts also allow command line arguments which will be directly passed to your code.

Output: In every turn, your program must print the game board using basic ASCII art, followed by the last played move, followed by whose turn it is (COM vs. human).
At the end, when either player wins or there are no valid moves left, your program should state who won the match by indicating the color and the nature of the player (human vs. COM). You can check the outputs of the provided programs to see how your output should look like in order to be compatible with what the referee script expects. The most important part is that you print a line with the move that was just played, like this:

```
Move played: a1
```

Implementation Restrictions: You are not allowed to use any third party frameworks or non-native libraries for your implementation. You can use standard and native libraries that are installed on CSIL for every user (the definition of standard or native libraries includes any piece of code that does not require installation or downloading). Also keep in mind that your submission can't exceed 10MBs.

4. Instructions on What and How to Submit

Use the CSIL **`turnin`** command to submit the necessary files and directories. Note that all directory and file names are case sensitive. For this assignment you need to issue the following command:

```
% turnin mp2@cs165a mp2
```

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". The grader will compile and execute your programs. So, your code must compile without referencing any external files or libraries that are not included in your submission. Standard header files, such as iostream.h, or other native libraries and modules are fine to use as long as they are available in CSIL.

You may use C/C++, Java, or Python (v.2.7 only) for this programming assignment. You may use any OS/compiler version for development, but your final code must compile and run on the CSIL machines. **So make sure you compile and run the final version of your code on CSIL machines before turning it in.**

5. 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, 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, 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, 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.

Note: Put all the necessary files in a directory named mp2 and submit the directory as a whole. When the grader extracts your submission, all your submitted files should be in *email*/mp2/ (where email will be your email address that the turnin command provides automatically).

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. Grading

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. If your program can win an adversary that randomly chooses their next move you will get 30 out of the 60%. If your program implements the basic minimax algorithm with some depth >= 2, you will get **40** out of the 60%. If your program also applies alpha-beta pruning, you will get **50** out of the 60%.
To test which features you implemented during grading, we will have your program compete with our own implementations. If your program wins all of them, then you should expect to get the full 60% credit. These implementations are provided to you so that you can test your own program before the final submission. Note that the grading platform is the Linux machines in CSIL. Make sure that your program does not not exceed the 30 seconds per move rule on these machines (it might be running faster in your laptops).

Grade Breakdown:

- **20%** Complete Report
- **10%** Makefile and execution specifications
- **10%** Correct program specifications: command line arguments, no segfaults or exceptions, no 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. 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.

- 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 in batches automatically every hour or two depending on the load, so 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).
- 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).

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. You need to specify your choice of size in the TOURNAMENT file within your submission: just type the number and nothing else.

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.