Gomoku Program Design Major Project

v1.0.0 [2023-10-16]

Note: This content may be subject to updates at any time. Please refer to the latest version for accuracy.

Project Overview

This project is an individual project that students are required to complete **independently**. The task is to implement a "Gomoku" move program (referred to as the brain program) using the C programming language. The program should use `stdin` to receive the opponent's moves and use `stdout` to output its own moves (see input and output format). There are time and memory constraints (see competition rules), although the algorithm is not restricted.

Students are required to submit the source code of their brain program to a designated online evaluation platform. The platform will conduct machine-to-machine "Gomoku" matches between all participants' brain programs. Points will be awarded to the winners of each match. The final grade for students in this project will be determined by their ranking based on the cumulative points earned in the matches on the platform (see ranking and scoring rules).

Rules of Competition

1. Each brain program will compete against the brain programs of other students in multiple matches. Each match consists of multiple games.
2. To ensure fairness, each match will have an even number of games. For each side, half of the games will be played as black and the other half as white.

Competition Rules

These are the rules of the Gomoku game in its original form.

1. The game is played on a square grid with dimensions of 12x12, as shown in the "Appendix - Game Board."
2. The game is played by two players, referred to as Black and White. Black plays the first move, and then players take turns placing their own pieces on empty positions on the board.
3. The game ends when a player successfully forms a straight line of five or more of their own colored pieces, either horizontally, vertically, or diagonally.
4. If all the empty positions on the board are filled and no player has achieved the winning condition, the game is considered a draw.
5. When it is their turn, the brain program should provide a move within 2 seconds.
6. In each game, the total time used by the brain program should not exceed 90 seconds (excluding the opponent's move time).
7. The brain program should not consume more than 350MB of memory at any given time.
8. The brain program will continue running while the opponent is making their move.

Note:

1. If either side exceeds the time or space limit, or if the program exits abnormally, that side will be considered as losing the game.
2. The submitted code will be run on a cloud platform, and depending on the platform's workload, the teaching assistant may adjust the hardware configuration to manage the load. Therefore, the code should be able to stay within the time limit on different CPU configurations.
3. The brain program will run on a single CPU core, so using multiple threads will not improve the speed.
4. The brain program will run in a sandboxed environment with isolated execution and without file I/O permissions. Calling restricted functions may result in abnormal program termination.

Ranking and Scoring Rules

1. For each match, the side that wins the majority of games will be considered the winner of that match, and the other side will be considered the loser. If the number of wins is equal for both sides, the match will be considered a draw.
2. Winning a match earns 3 points, drawing a match earns 1 point, and losing a match earns 0 points.
3. A student's total score is the sum of the points earned by their brain program in matches against other students' latest brain programs.

4. Students can submit multiple versions of their brain program, and each submission will reset the total score. The total score will accumulate based on the latest submission.
5. To prevent abuse of resources, each student has a limit on the submission frequency, including a minimum time interval between two code submissions and a limit on the total consumed time for code submissions in a day. These submission frequency limits may be adjusted at any time, and the adjustments will be announced in the QQ group. Students can also see the current submission frequency limits on the submission interface of the online evaluation platform.
6. The size of the code submitted by students should not exceed 1MB. In general, the code should not be long enough to exceed this limit.
7. Students are strictly prohibited from plagiarizing existing code from the internet or from other students (including non-final versions of the code). If any instance of plagiarism is identified and verified, it will be treated as cheating in this course.

Example of Competition Scoring

For example, suppose there are four students, A, B, C, and D, who have submitted brain programs A1, B1, C1, and D1. The ranking will be based on the total scores from the following six matches (each match consists of 2 games, totaling 12 games):

A1 vs B1, A1 vs C1, A1 vs D1
B1 vs C1, B1 vs D1
C1 vs D1

Later, if A1 submits a new brain program, A2, the rankings will be based on the total scores from the following six matches:

A2 vs B1, A2 vs C1, A2 vs D1
B1 vs C1, B1 vs D1
C1 vs D1

Please note that the initial brain program A1 from student A will no longer be considered in the rankings. The rankings always reflect the results of matches between the latest brain programs of each student.

Input & Output Format

The brain program for the students' Gomoku game needs to receive commands from standard input (`stdin`) and respond accordingly by outputting the response to standard output (`stdout`). Each command will be on a separate line. The response from the brain program should also be on a separate line (followed by a newline character `\n`). Here are the commands that the brain program needs to support:

START [FIELD]

Before the start of the game, the brain program will definitely receive this instruction, which indicates the relevant information about our side. `FIELD` represents the color of the stones our side will play. When `FIELD` is 1, it means our side plays with black stones, and when `FIELD` is 2, it means our side plays with white stones.

After receiving this instruction, the brain program needs to respond with `OK` within 1 second. Otherwise, it will be considered a loss.

Example

The platform sends the instruction:

```
START 1
```

The brain program replies:

```
OK
```

PLACE X Y

This instruction represents a move by the opponent. `X` and `Y` are the row and column coordinates (starting from 0) where the opponent wants to place their stone.

The brain program does not need to reply to this instruction.

Example

The platform sends the instruction:

```
PLACE 1 1
```

The brain program does not need to respond.

TURN

This instruction represents the turn for our side to make a move. After receiving this instruction, the brain program calculates its move and responds with the row and column coordinates where it wants to place the stone. The brain program needs to make the move response within a specified time limit, otherwise, it will be considered a loss.

This instruction may appear directly after the `START` instruction, indicating the beginning of the game with our side playing as black stones. It may also appear after a `PLACE` instruction, indicating that it is our side's turn to make a move after the opponent has placed their stone. Note that if the game ends directly after the opponent's move, there will be no `TURN` instruction following the `PLACE` instruction.

Example

The platform sends the instruction:

```
TURN
```

The brain program replies:

```
1 1
```

END [FIELD]

This instruction represents the end of the game, where `FIELD` indicates the winner. When `FIELD` is 0, it signifies a draw. When `FIELD` is 1, it means our side wins, and when `FIELD` is 2, it means the opponent wins. After receiving this instruction, the brain program does not need to make any response and can decide on its own whether to exit the program (during evaluation, regardless of whether the brain program exits voluntarily or not, it will eventually be closed). This instruction can occur at any time, for example, if it appears before 'BEGIN', it may indicate that the game ended due to the opponent's program crashing.

Example

The platform sends the instruction:

```
END 1
```

The brain program does not need to respond.

Debug Instruction

For debugging purposes, the brain program can output `DEBUG [MESSAGE]` at any time (it needs to occupy a separate line and cannot exceed one line). This content will be logged but not processed by the platform. Students can download the complete log from the platform to facilitate bug debugging. The brain program can output debug messages multiple times, but each `MESSAGE` cannot exceed 16KB (any excess will be truncated), and the cumulative size of all `MESSAGE` cannot exceed 32KB (any excess will be ignored), and it will not be logged.

Note: This debug output cannot replace the response to any other instructions. For example, if a `TURN` instruction is received and a coordinate needs to be responded, even after outputting `DEBUG [MESSAGE]`, the brain program still needs to continue outputting the coordinate.

Example

The brain program outputs:

```
DEBUG Hello World!
```

Direct Termination Resulting in Loss

Here are some situations that may cause the game to be terminated directly and result in a loss.

Responding when not allowed to respond

For example, if our side has just responded to a `TURN` instruction and needs to wait for the next `TURN` before making a move. If our side continues to output content (excluding debug output) at this time, it will be considered a direct loss.

Incorrect response format or invalid move

For example, for a `TURN` instruction, our side needs to respond with a pair of coordinates separated by a space. If the response format is incorrect (excluding debug output), it will be considered a direct loss. If the responded coordinates are outside the chessboard or the specified position is not an empty intersection, it will also result in a loss.

Exceeding time limits or space limits

Time limits include the time limit for a single instruction and the total time limit. For the `START` instruction, a response is required within 1 second. For the `TURN` instruction, a response needs to be made within the specified time limit for a single move, as stated in the competition rules. The total time limit is also specified in the competition rules, starting from `START` and ending at `END`, excluding the time taken by the opponent's move.

Program crash

If the program crashes due to various reasons (such as accessing invalid memory addresses), it will result in an instant loss.

Programming Notes

1. The `scanf` function blocks when there is no data available to read, so avoid reading instructions when it is not expected. For example, if it's your AI's turn to output and you execute a `scanf` statement, your program will wait indefinitely until it times out.
2. In most cases, the output of `printf` is buffered and not immediately displayed, which can cause timeouts. You can use the `fflush` function to solve this problem: immediately after `printf`, include the statement `fflush(stdout)` to flush the buffer.
3. If you want to utilize the opponent's move time for calculations, you may need to use multithreading techniques. Since the platform environment is Windows, it is recommended to use relevant APIs specific to the Windows platform to avoid compatibility issues.
4. Special reminder: If you plan to use advanced algorithms, make sure you can answer questions about the algorithm during the defense, as failure to do so may be considered plagiarism.

Instructions for Online Evaluation Platform

1. Open the platform address http://10.60.43.52:8888/ on your computer using a newer version of Chrome or Firefox browser.
2. Click the 'Sign In' button located in the upper right corner of the navigation bar to log in. Both the username and password are your student ID (if you are unable to log in, please contact the teaching assistant in the group).
3. If this is your first login, you will be prompted to change your password to a new one to prevent unauthorized use of your account. You will also be asked to enter a nickname. This nickname will be displayed on the homepage scoreboard, visible to everyone, and can be freely modified in the future.
4. Click on 'Scoreboard' in the navigation bar to view the scoreboard, which shows the current total score ranking. The scoreboard is updated every ten minutes.

5. After clicking on 'Submission' in the navigation bar, you can switch to your own submission records (My Submissions), other people's submission records (All Submissions), or the page for submitting new code (Submit New Brain) in the menu on the right side.
6. When submitting new code, please merge your code into a single file for submission. Multiple file submissions are not supported. You can also choose different compilers.

Evaluation Environment

See here.

References

1. Rules of Gomoku Competition
2. Play Gomoku

Appendix

Chessboard

	0	1	2	3	4	5	6	7	8	9	10	11	
0													0
1													1
2													2
3													3
4													4
5													5
6													6
7													7
8													8
9													9
10													10
11													11
	0	1	2	3	4	5	6	7	8	9	10	11	

The chessboard is depicted in the following diagram. The coordinate of the bottom-left corner is (11, 0), the coordinate of the bottom-right corner is (11, 11), the coordinate of the top-left corner is (0, 0), and the coordinate of the top-right corner is (0, 11).