VG101 — Introduction to Computer and Programming

Final Lab

Director: Yihao Liu
TA (J.G. Wu): Chenhao Ye, Luotian Xu, Jinze Liu
TA (Manuel): Jiayi Chen, Shuhan Wang, Zhi Lin, Zihao Shen — UM-JI (Fall 2018)

Goals of the Lab

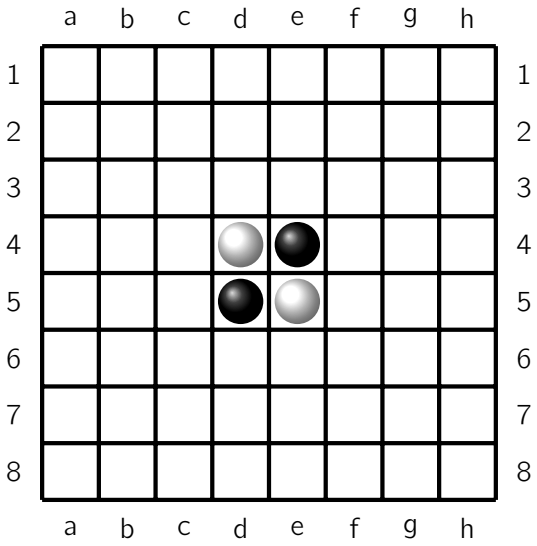
- Implement a strategy board game **Reversi**
- Implement an AI (Artificial Intelligence) playing **Reversi**
- Learning about API
- Compete with others and fight for HONOR!

1 Introduction

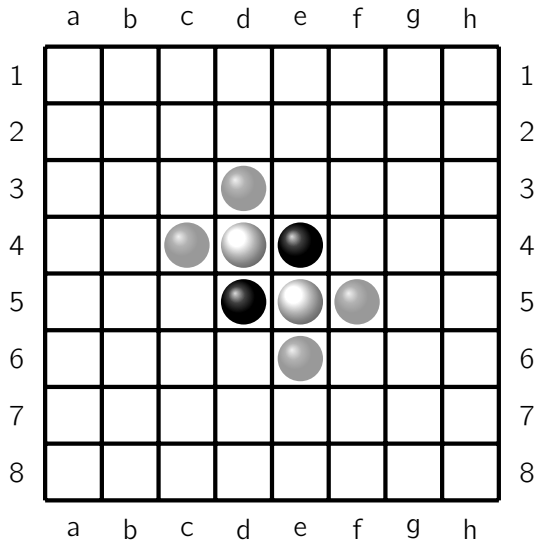
Welcome to your last lab! In this lab, you are going to design and implement a **Reversi** AI and compete with your peers. Have fun in this lab, and fight for honor!

1.1 Reversi Rules

The game begins with four disks placed in a square in the middle of the grid, two facing white side up, two pieces with the dark side up, with same-colored disks on a diagonal with each other. Convention has initial board position such that the disks with dark side up are to the north-east and south-west (from both players' perspectives), though this is only marginally meaningful to play (where opening memorization is an issue, some players may benefit from consistency on this). The dark player moves first.

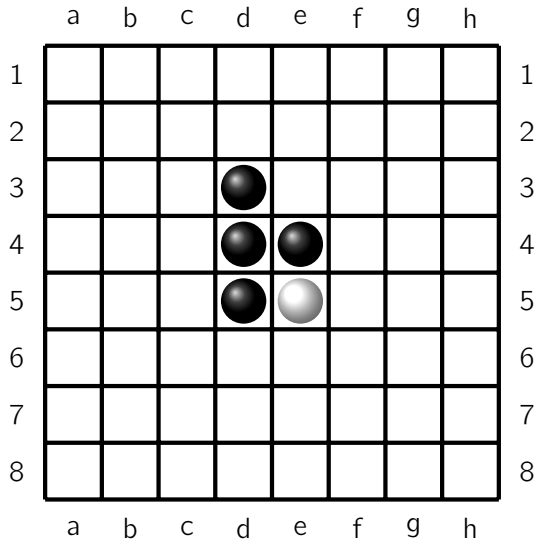


Dark must place a piece with the dark side up on the board, in such a position that there exists at least one straight (horizontal, vertical, or diagonal) occupied line between the new piece and another dark piece, with one or more contiguous light pieces between them. In the below situation, dark has the following options indicated by translucent pieces:

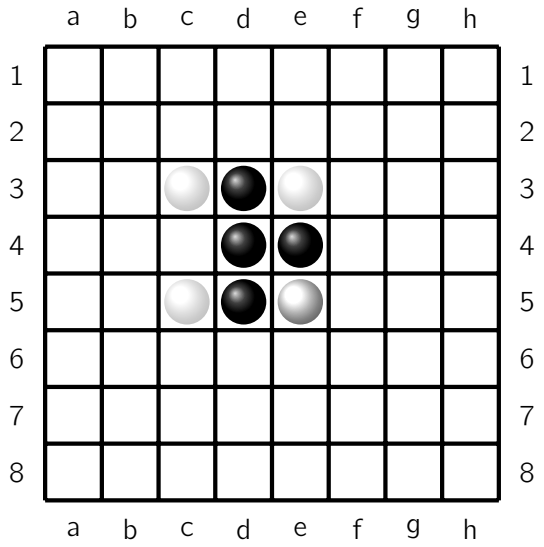


After placing the piece, dark turns over (flips, captures) all light pieces lying on a straight line between the new piece and any anchoring dark pieces. All reversed pieces now show the dark side, and dark can use them in later moves—unless light has reversed them back in the meantime. In other words, a valid move is one where at least one piece is reversed.

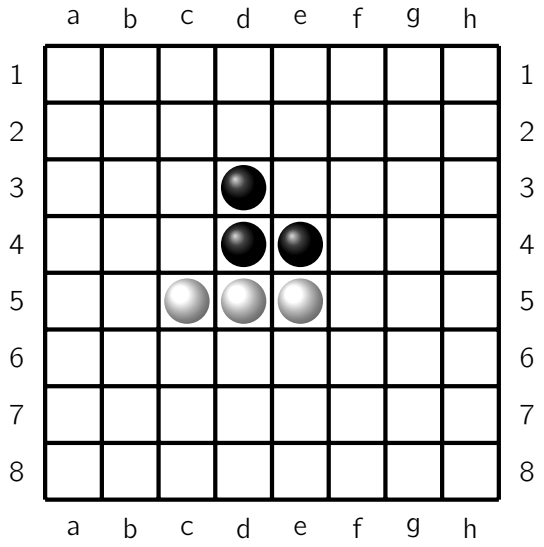
If dark decided to put a piece in the topmost location (all choices are strategically equivalent at this time), one piece gets turned over, so that the board appears thus:



Now light plays. This player operates under the same rules, with the roles reversed: light lays down a light piece, causing a dark piece to flip. Possibilities at this time appear thus (indicated by transparent pieces):

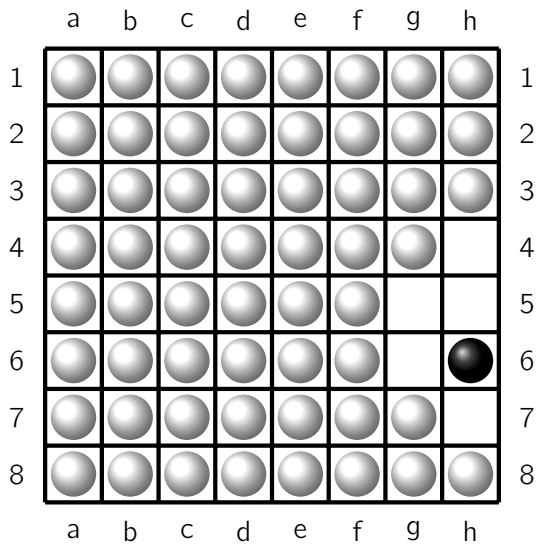


Light takes the bottom left option and reverses one piece:



Players take alternate turns. If one player cannot make a valid move, play passes back to the other player. When neither player can move, the game ends. This occurs when the grid has filled up or when neither player can legally place a piece in any of the remaining squares. This means the game may end before the grid is completely filled. This possibility may occur because one player has no pieces remaining on the board in that player's color. In over-the-board play this is generally scored as if the board were full (64–0).

Examples where the game ends before the grid is completely filled:



The player with the most pieces on the board at the end of the game wins.

2 Tasks

You are going to design a **Reversi** AI according the rules stated above, and submit it to the online judge (`reversi.sstia.tech`) to compete with your peers.

2.1 Compile Guide

Your **Reversi** program should be written in *C/C++*, the supported language levels are *C99, C11, C++98, C++11, C++14, C++17*, the supported compilers are *gcc* (*g++*) and *clang* (*clang++*).

Your program should be able to compile and run on our *Linux* hosted server, which means you are only allowed to use the standard libraries without the VC++ extensions and other libraries (such as multi-threading library). Here is a list of example forbidden features:

- `#include <windows.h>` `// Windows extensions`
- `#include <unistd.h>` `// Unix extensions`
- `fflush(stdin);` `// Undefined behavior`
- `itoa(int value, char *str, int base);` `// Non standard function`
- `strrstr(const char *str1, const char *str2);` `// Non standard function`

2.2 Application Programming Interface (API)

2.2.1 Standard Input

The standard input will always be tuples of commands and arguments, the format is

COMMAND <ARG1> <ARG2> <ARG3>

The first several commands are ensured to be:

```
START
PLACE d 5 (1|2)
PLACE e 5 (2|1)
```

```
PLACE e 4 (1|2)
PLACE d 4 (2|1)
DONE
```

In *START*, you can initialize you board and AI; in *PLACE*, your AI will get the position and owner of the four pieces, the third argument is 1 or 2, where 1 means this is your chess and 2 means this is your opponent's chess; After *DONE*, you should output *OK*, and the initialization is completed and the game begins.

If you are the first player (black) to play, you will receive a command *BEGIN* to notice you, then you start to play a chess (see the Standard Output section).

After you have played a chess, your opponent will play another one and you will receive a command *TURN X Y*, where *X* is an letter in a-h and *Y* is an integer between 1-8, which means the position of your opponent's chess. Then you should play a chess again (see the Standard Output section).

Note that you player may not be able to play a valid move, then you will receive a command *PASS*, and then you should also play a chess. If you are not able to play a valid move as well, the game will end in the next command.

When *END* is received, your program should exit with code 0.

2.2.2 Standard Output

Except receiving *DONE* and printing *OK*, you should not output anything rather than the position of your next chess to standard output. When you can not make a valid move, print *PASS* to the standard output.

All commands which need an output (*BEGIN* and *TURN*) must be responded in 1 second. Your output position must be valid. Or you will directly lose the game.

The output format is *X Y*, where *X* is an letter in a-h and *Y* is an integer between 1-8, which represents the position you play the next chess. You should add a newline after it to flush the output.

2.2.3 Standard Error

Your *Reversi* program should also support printing on standard error for debugging. Once a *BEGIN* or *TURN* is received, or you played a chess, you should print a board of the played chess (arbitrary) and the available positions (optional) to play on *stderr*.

You can use *cerr* in *C++* and *fprintf(stderr)* in *C* to support this feature.

2.2.4 Example

Black Input:

```
START
PLACE d 5 1
PLACE e 5 2
```

```
PLACE e 4 1
PLACE d 4 2
DONE
BEGIN
TURN e 5
...
END
```

White Input:

```
START
PLACE d 5 2
PLACE e 5 1
PLACE e 4 2
PLACE d 4 1
DONE
TURN d 3
...
END
```

Sample Board (X means black, O means white, ‘ means available positions):

```
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
1 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
2 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
3 |   |   |   | ` |   |   |   |   |
  +---+---+---+---+---+---+---+---+
4 |   |   | ` | O | X |   |   |   |
  +---+---+---+---+---+---+---+---+
5 |   |   |   | X | O | ` |   |   |
  +---+---+---+---+---+---+---+---+
6 |   |   |   |   | ` |   |   |   |
  +---+---+---+---+---+---+---+---+
7 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
8 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
```

2.3 Submission

You should write your code in one file and submit on our Online Judge System on https://reversi.sstia.tech. It will be opened several days after the lab released.

3 Matching and Grading Policy

3.1 Game

Each game is consisted of two rounds. Both players take the black side who serves first for one rounds for the sake of equality. Besides, we will apply four pieces in the center in a standard diagonal pattern, rather than being placed by players. (See the introduction)

3.2 ELO Ranking

We take advantage of *ELO Ranking* as the basic ranking method which is commonly applied in many online competitive games. Every participant has an initial rank as 1500 which will be increased, decreased or kept unchanged when there's a win, lose or draw game, respectively. The amount of changed rank is dependent on the rank difference between the two participants of one game. The higher the rank difference, the more amount of rank will be changed.

3.3 Matching

The matching mechanism can viewed as such a model: the server keeps looking for two feasible participants as opponents to hold a game. There are several rules defining what kind of participants are feasible to hold a game:

- These two participants' rank difference is no larger than 100 except one of them has just updated the AI and is of lower rank.
- Both participants have submitted one effective AI program (effectiveness means this AI is able to play at least one feasible step).
- These two participants' win game difference is no larger than 1 unless the one with such advantage is of lower rank. For example, if A has a 3-1 winning relation with B, A and B will not be matched together to hold a game any more unless A's rank is lower than B's rank.

Since there may exist multiple feasible games to be hold, in order to make sure those who needs to know the their AI's combat results can get their results quickly, we design the following priority for finding two players:

- The player who has just updated a new AI will be set with top priority.
- The player whose AI is under a win streak will be set with relatively high priority, ranked by length of streak.

3.4 Grading

3.4.1 Jigang's Section

The basic marks of this lab is 100 marks as a normal lab (2.5% in final grade), while you can at most earn 200 marks from this lab (twice the marks of a normal lab). The marks exceeding the 100 marks will be counted as lab bonus, and can be added to lab credits of final grade until all the 20% of lab credits in final grade are gained. You will be granted 25 marks as lab attendance after submitting an working AI. We will curve a function so that the mean of the lab is about 80 – 100 marks, in which active and competitive ones will get more marks according to their rank.

3.4.2 Manuel's Section

You will be granted the attendance of one lab (about 0.6% of the final grade) after submitting an working AI. You can at most get 2.5 marks of bonus in final grade according to your rank.

3.4.3 Schedule

The deadline of this lab is Dec. 9th, 11:59 p.m. We will grade your lab according to your final ranking.

3.5 Honor Code

This competition is following UM-SJTU JI Honor Code. **Don't share your codes or copy and plagiarize codes from any source.**

4 Reference

- https://en.wikipedia.org/wiki/Reversi