SI100B *On-site* Programming 2

teafrogsf/Cao Yuhan

October 23, 2021

Part I

Mahjong Logs

0 README

Please read the whole problem first, then you can start at task 1. If you are having trouble solving a task, you can turn to the end of the document for hints. It is recommended to try to solve the problem independently first.

NOTE: You are NOT allowed to import any third-party library, such as `csv`. You can use `matplotlib` for task 3, but when you submit your code in Autolab, **please comment all contents related to this library**.

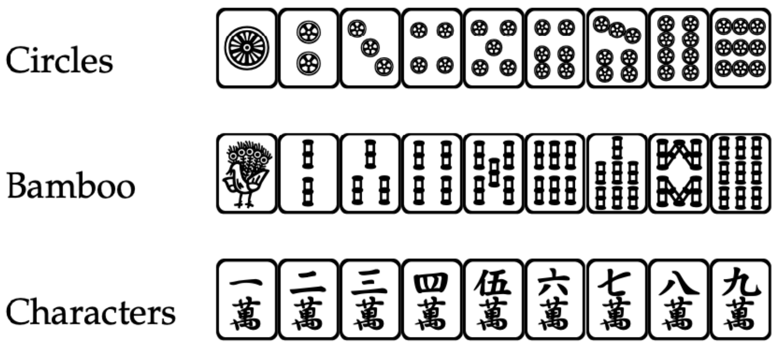
1 Background

As a student who aspires to be a theoretical computer scientist, Little Frog also has a keen interest in Mahjong.

Mahjong is a long-established board game. There are 34 basic tiles in classical mahjong, which are presented below.

1.1 The Three Suits

In particular, the one of bamboo is often decorated with birds, the design of which often varies be- tween mahjong sets). For convenience, we use “1p, 2p, 3p, 4p, 5p, 6p, 7p, 8p, 9p” to represent circles (饼/筒), “1s, 2s, 3s, 4s, 5s, 6s, 7s, 8s, 9s” to represent bamboos (条/索) and “1m, 2m, 3m, 4m, 5m, 6m, 7m, 8m, 9m” to represent characters (万). For an instance, “7m” represents the tile “七萬”.



1.2 The Honors

In addition to the suit tiles, there are seven different honors: four winds and three dragons. The winds are shown in the order east-south-west-north (東-南-西-北), and the dragons are shown in the order white-green-red (白-發-中). The shorthand for honors is: “1z”, “2z”, “3z”, “4z” for “東”, “南”, “西” and ” 北 “, and “5z”, “6z”, “7z” for “白”, “發” and “中”.

Winds
Dragons

With **four of each of the above tiles**, a mahjong set consists of 136 tiles.

There are no fixed rules for mahjong. In this problem, we just need to consider a simplified rule of classical mahjong. Under this rule, mahjong is played by 4 players. When a game starts, each player draws **13 initial tiles** in his/her hand, and the player who draws a tile at the beginning is the dealer. A player’s turn contains two phases: first, the player will **draw a tile** from the remaining tiles (the wall, 牌山), and **judge if he/she wins** (和牌). If he/she wins, the game ends. If not, the player will take the second phase of his/her turn, which is to drop a tile from his/her hand (the player can decide which card to drop). The four players will take their turn one by one until one player **wins or the wall is empty**. We do NOT consider the situation where tiles are obtained from other players instead of the wall. After a game ends, the player who drops after the dealer will become the new dealer next time.

The 14 tiles in a player’s hand (the status after drawing a tile from the wall) is called the mahjong hand. We say a player wins **if and only if** his/her mahjong hand is composed of exactly four sets and a pair. A set is either a Chow (吃; 顺子), or a Pung (碰; 刻子). A Chow is three consecutive tiles in the same suit. Chows CANNOT be made by dragons or winds. A Pung is composed of three identical tiles. A pair contains two identical tiles. An example of Chow and Pung is shown below.

We don’t need to consider some special rules like thirteen orphans or seven pairs in this problem. For example, if some player’s hand is: [1s, 1s, 1s, 2m, 3m, 4m, 5m, 6m, 7m, 1z, 1z, 1z, 8s, 8s], then he/she wins the game because these 14 tiles can be decomposed as four sets: [1s, 1s, 1s] (a Pung); [2m, 3m, 4m] (a Chow); [5m, 6m, 7m] (a Chow); [1z, 1z, 1z] (a Pung), and a pair: [8s, 8s].

One day, Little Frog, Little Dog, Little Shadow, and Little Diversion (?) are playing mahjong. Little Frog wants to record their games, so he writes a program to synchronize their actions in every turn (including drawing 13 initial tiles). However, he forgets to separate each game and now he only has a lot of lines consisting of various tiles. Now you need to write a program to help him, or he will be exhausted by the separation.

2 Input Format

The input data is read in from a .csv file.

The first line of input contains one string s, indicating the initial dealer.

The next line contains four strings, indicating nicknames of four players.

The next several lines contains four strings, indicating the tiles they draw or drop. **Each column** corresponds to one player’s drawing or dropping. If this player do nothing in this line, there will be an empty string.

Note that they will draw 13 initial tiles at the beginning of **each game**.

We provided two sample input test.csv, test2.csv. You can use

```
solve("test.csv")
```

to test your program (Just as we provide in mahjong.py). But when you submit your code, comment it.

3 Task 1

3.1 Description

First, you need to solve the most important problem - Find every winner. You should print the winner of each game and calculate the win rate of each player.

You are NOT required to solve whether a set of tiles can win. we provided a function in check.py, and you can import it by

```
from check import CheckWin
```

If you call it, it will return a boolean value if you pass in a valid parameter, i.e. a list consisting of 14 valid tiles.

Note that the parameter of function is passed by object reference, so if you do not want change your list after passing it into `CheckWin`, you may need to use `deepcopy`.

3.2 Output Format

You should output your answer in winner.csv.

The first several lines of output, each containing a string, indicate the winner of each game. If a game draws (which means nobody wins), output a string `Draw`.

Then, output a blank line to seperate two parts.

The next four lines, each containing a string and a number, indicate the win rate (the number of win games/the number of games) of each player, in the order of the input file. Each win rate is output as a percentage, with two decimal places.

3.3 Sample Output of test.csv (In the Form of Text)

```
Dog
Shadow
Diversion
Frog

Frog,25.00%
Dog,25.00%
Shadow,25.00%
```

```
Diversion,25.00%
```

3.4 Sample Output of test2.csv (In the Form of Text)

```
Draw

Dog,0.00%
Frog,0.00%
Diversion,0.00%
Shadow,0.00%
```

4 Task 2

4.1 Description

Next, you need to solve a easy problem - Find win rates of every tile.

In this task, we only consider the last tile of each game, i.e. after drawing this tile, a player wins. The win rate of each tile is

$$\frac{\text{the number of occurrences as the last tile}}{\text{the number of games}}$$

4.2 Output Format

You should output your answer in tile.csv.

All lines, each containing a string and a number, indicate the win rate of each last tile, in the descending order of win rate. If the win rates of two tiles are same, output in the lexicographical order of tiles.

4.3 Sample Output of test.csv (In the Form of Text)

```
6m,75.00%
9m,25.00%
```

4.4 Sample Output of test2.csv (In the Form of Text)

(There is nothing in tile.csv, so it is empty.)

5 Task 3

5.1 Description

Finally, you need to solve a harder problem - Record and visualize the changes in points for each game.

To solve this task, you may need to some basic knowledge of recursion and backtracking, and you are allowed to **modify** check.py or write a new one for yourself.

If a player win a game, he/she will collect points from other three players. The rules of calculating points are as follows:

Let the basic points be $b = 3600$, the number of Pungs in the mahjong hand be m, the number of Pungs consisting of the Honors in the mahjong hand be h, then the player will get

$$(b + 1200h) \times 1.5^m$$

points. However, if $m = 0$, the player will get 7200 points instead.

If a game draws, points of every players will not change.

If a mahjong hand has different ways of decomposing, choose the way whose point is the maximum one.

The basic point of each player is 50000. negative point is accepted.

5.2 Visualization

You can use `matplotlib` to visualize the answer. First, you need to

```
import matplotlib.pyplot as plt
```

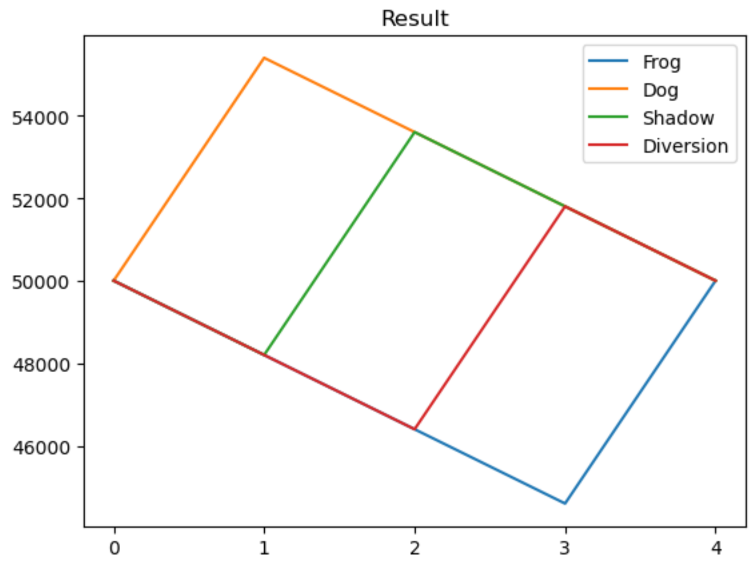
You should let the number of every games be the x-axis, and the points of four players of each games be the y-axis.

You can use `plt.xticks` to set the x-axis.

You can use `plt.title` and `plt.legend` to set the title and the legend.

You can use `plt.show` to test you result. However, what we need is your `pyplot` object, so please make sure that your graph is TOTALLY same as the one required by the standard.

The figure corresponding to the sample output is as follows:



If you haven't installed `matplotlib` before, you can use

```
python -m pip install -U pip
python -m pip install -U matplotlib
```

to install it.

This part will be checked **on-site**.

5.3 Output Format

You should output your answer in battle.csv.

There are five lines for each game in output data.

The first line contains a string, indicating the number of game.

The next four lines, each containing a string and a number, indicating the player and his/her points after this game, in the order of the input file.

After each game, output a blank line to seperate games.

5.4 Sample Output of test.csv (In the Form of Text)

```
Game 0
Frog,50000
Dog,50000
Shadow,50000
Diversion,50000

Game 1
Frog,48200
Dog,55400
Shadow,48200
Diversion,48200

Game 2
Frog,46400
Dog,53600
Shadow,53600
Diversion,46400

Game 3
Frog,44600
Dog,51800
Shadow,51800
Diversion,51800

Game 4
Frog,50000
Dog,50000
Shadow,50000
Diversion,50000
```

5.5 Sample Output of test2.csv (In the Form of Text)

```
Game 0
```

```
Dog,50000
Frog,50000
Diversion,50000
Shadow,50000

Game 1
Dog,50000
Frog,50000
Diversion,50000
Shadow,50000
```

6 Data Range & Grading Criteria

Test Case	Size of Input File	Special Cases
1	$\leq$ 2kb	1,2
2		
3	$\leq$ 5kb	2
4		
5		
6		
7	$\leq$ 30kb	
8		
9		
10		

Case 1: There is only one game in the input file.

Case 2: There are no invalid situations in the input file.

For each test case, if you solved task 1, you will get 3.5 points; If you solved task 2, you will get 1.4 points; If you solved task 3, you will get 2.1 points.

Online submissions close at 11:59AM on Monday, October 25. NO late submission is allowed. ONLY the last submission will be recorded as your final score.

You can submit up to 30 times. After this threshold, each additional submission is penalized by 10% of points.

7 Submission Method

You need to package check.py, mahjong.py into a .tar file and submit it into Autolab.
You can use

```
tar -cvf mahjong.tar check.py mahjong.py
```

to package, or you can just use any archiver software, e.g. Bandizip.

8 Hints

All your content should be implemented in `solve(filename)` (of course, you can call your custom function in `solve`).

This question will NOT require any ability of designing algorithms, the size given is just a reference for you to generate your own data and test your program. However, poorly implemented code may not be able to get through all the data in the allotted time, i.e. 5 seconds~~, but this is difficult to achieve~~.

For task 3, you could always get the best way (i.e. get the maximum point) to decompose the mahjong hand by using `CheckWin` we provided. Why?

Good luck and have fun.