Task Report

Jaskirat Singh

***Overview*—A simple command-line based application built in python that meets the requirements of Mr. Raccoon. Two different approaches used: brute force picking and random shuffling, along with some interesting results.**

I. Problem statement

A command line based application for Mr. Raccoon for his dinner party, along with the following rules:

- You cant draw your own name.
- You cant draw the name of your partner if you have one.

If any of the above happens, then:

- The ongoing process stops.
- Gifts are sent back to original people
- Reshuffling occurs for all people.

II. Use cases:

- As a member of the family, I want to be able to register to the gift exchange in order to receive a gift.
- If I have a partner, I dont want to receive the gift brought by my partner.
- As the organizer of the gift exchange (Mr. Raccoon), I want to be able to just launch the whole process automatically and get back the results.

III. Input format:

There are two different parts: **deployment and usage part** and **simulation checking part**, the latter is used to check and compare the efficiencies of the two algorithms used.

A. Deployment and usage part:

So, for the deployment and usage part. All of the input is taken from stdin and output is written to stdout.

Input:

- The first line contains the names of all the participants that took part in the game.
- The next line contains a single integer the number of couples, call it n.
- The next n lines consist of a pair of 2 integers that form a couple.

For example:

```
1,2,3,4,5,6,7,8,9,10
2
2,5
3,9
```

Output: It then tells the user on the command line who has to give the gifts to whom.

```
1 gives his gift to 7
2 gives his gift to 1
3 gives his gift to 5
4 gives his gift to 3
5 gives his gift to 8
6 gives his gift to 10
7 gives his gift to 2
8 gives his gift to 9
9 gives his gift to 6
10 gives his gift to 4
```

B. Experimentation part:

This was done primarily to compare the efficiency of the two algorithms I had implemented. The idea is to execute the code on some specified family multiple times and take the average of the same. So, the input is taken from stdin as well as by passing command line arguements. From stdin, the family input is taken in the previously specified format, while the number of simulations is passed as argv, i.e. if the input file is located in ./tests/10.txt, and you have to simulate the experiment K times, the run the following:

```
python simulation.py K < tests/10.txt
```

This will print the average execution time for the algorithm over these K simulations. Now, in the experimentation part, there is a parameter called $algoused$.

- If algoused is set to 1, then whole of the list is shuffled at once and exchanges are created for all. If there is some case where a person x has to give his gift to his partner or himself, the simulation restarts.
- If algoused is set to 2, then a one-by-one simulation occurs. For each person x, a random person y is popped from the hat and paired with x. If y is same as or the partner of x, the whole simulation restarts, as mentioned in the problem statement.

The results observed have been stated below, averaged over 50 simulations.

TABLE I
SUMMARY OF RUNNING TIMES OF BOTH ALGORITHMS

Member count	Random shuffle Time(s)	Brute Force select time(s)
10	$4.1 * 10^{-5}$	$6.5 * 10^{-5}$
100	$2.9 * 10^{-4}$	$4.9 * 10^{-4}$
1000	0.002	0.004
10000	0.012	0.037
100000	0.115	1.278
1000000[1]	0.98	**583.39**

[1] Note that all of the simulations mentioned in the table have been averaged over 50 times, however, the marked one is just a one run result