Algorithms and Data structures 2019: Second assignment

November 22, 2019

1 Instructions

IMPORTANT: This assignment consists of two parts. The first part is similar to the previous assignment, where you have to solve a problem, implement the solution and write a report. Most points can be obtained in the first part. The second part will be a competition, where additional points can be obtained by winning against your fellow students.

You may work in groups of two. For this assignment you have to hand in two things:

- Source code. We will run the code for grading.
- A report. In the report you have to explain the algorithm and analyse it.
- Additionally, if you participate in part two: source code and an explanation of the strategy.

Source code You are allowed to submit a solution in C, C++, Java or Python 3. You are only allowed to use the Standard Library corresponding to your selected language.

You will find a set of examples to test your solution on Brightspace. We will set up a website, on which you can upload your code. It will be automatically be tested against test cases.

Report Besides handing in code, we would like to receive a report – **of at least 2 and at most 10 pages** – in which you explain your algorithm and analyse its correctness and runtime complexity.

Submission The deadline for sending in your solution is on January 9th. You must submit your solutions via Brightspace. Only one team member has to submit a solution; the names and student numbers of both team members must be mentioned in the report.

Grading Grades will be determined as follows (you can earn 110 points!)

- 40 points for the test results. We will be running several tests and you will get points for every correct answer within the time limit. If your code does not compile or does not read and write via `stdin` and `stdout`, you will get zero points on the test cases.
- 20 points for the explanation of your algorithm.
- 10 points for the correctness analysis.
- 10 points for the complexity analysis.
- 10 points for the quality of the code.
- (Part 2) 5 points for the explanation of your strategy.
- (Part 2) 15 points based on the ranking in the competition.

If you have any questions, do not hesitate to send a email to Timo Maarse, `t.maarse@student.ru.nl`.

2 Actor Actress Matching

Veronique and Mark play the following game: Veronique starts by naming an actress x_1, then Mark responds with an actor y_1 which has co-starred with x_1, then Veronique names an actress x_2 which co-stars with y_1, and so on. Of course they are not allowed to name the same actress or actor twice. The first player who gets stuck (i.e. is not able to respond any more) loses.

Given a fixed set of actresses X, a fixed set of actors Y and a fixed list of movies with casts, this is a finite game where both players have all information. So either Veronique or Mark should have a winning strategy (i.e. able to always win, no matter what moves the other player makes).

Veronique always starts. Can you decide which player has the winning strategy?

2.1 Input

Your program will be given the following input via `stdin`:

- On the first line: an integer n, with $1 \leq n$, denoting the number of actors and actresses (i.e. $n = |X| = |Y|$) and an integer m, with $0 \leq m$, denoting the number of movies (we still need to fix upper bounds on n and m, these will be in the order of 1000),
- then n lines with names of actresses,
- then n lines with names of actors,
- then m times the following:
  - one line with the name of a movie,
  - one line with an integer s, such that $1 \leq s \leq 1000$, denoting the size of the cast of that movie
  - then s lines with a name of an actress or actor.

You may assume all names are written using alphabetic characters only ([a-z,A-Z] and no whitespace). Furthermore, all names are unique.

2.2 Output

The output (via `stdout`) should contain the name of the player with the winning strategy. So either "`Veronique`" or "`Mark`".

We will check exactly with this specification. Any other output (besides whitespace) will be regarded as a wrong answer. Please do not output any debug information on stdout (this happened way too often in the previous assignment)!

2.3 Examples

2.3.1. Input:

```
2 2
DianaKruger
MelanieLaurent
BradPitt
NormanReedus
IngloriousBasterds
3
DianaKruger
MelanieLaurent
BradPitt
Sky
2
DianaKruger
NormanReedus
```

Output:

```
Mark
```

2.3.2. Input:

```
2 1
AmandaPlummer
ScarlettJohansson
JohnTravolta
SamuelLJackson
PulpFiction
3
AmandaPlummer
SamuelLJackson
JohnTravolta
```

Output:

```
Veronique
```

Hint: Who wins if there is a perfect matching?

3 Part two: competition

For this part you will play the above game against other teams. Of course, since you already know the winning strategy, doing exactly this would be boring. So we add the following scoring mechanism:

A move $x_i \to y_i$ will be awarded k points, where k is the number of movies where x_i and y_i both play in. (Similarly for a move $y_i \to x_{i+1}$.) If you lose in the end, you will get 0 points. If you win, you get average score of your moves. (If you win without making any moves, you'll get the highest possible k as score.)

Your program will run against every other program twice; once where you make the initial move (i.e. you are Veronique), and once where you make the second move (i.e. you are Mark). Your score will be summed up among all matches.

Example From example 2.3.1, there are three sensible plays:

- `DianaKruger`, `BradPitt`, `MelanieLaurent` $\mapsto$ 1 point for Veronique.
- `DianaKruger`, `NormanReedus` $\mapsto$ 1 point for Mark.
- `MelanieLaurent`, `BradPitt`, `DianaKruger`, `NormanReedus` $\mapsto$ 1 points for Mark.

3.1 Input / Output

Exactly as before (see above), followed by:

- `Veronique` if you start as Veronique or `Mark` if you start as Mark.

Then an interactive part starts:

- If it is your turn: output a name from the input. [1] If the name is used before, or does not co-star with the previous name, or does not appear in the input at all, your move is invalid and you lose.
- If it is not your turn, wait for input (waiting happens automatically when using the default methods for `stdin`). You will get the move from the other player as input.

Your program will get a total running time of 30 seconds. If your program runs longer, it will lose. If one player loses, both processes will be terminated.

We will use a real dataset derived from IMDb. The size of the input is similar to the bounds given in the first part.

[1] It might be a good idea to flush stdout, depending on your language of choice.

Example I/O Here is an example trace. Input is in red, ouput in blue. This is what one player saw:

```
2 2
DianaKruger
MelanieLaurent
BradPitt
NormanReedus
IngloriousBasterds
3
DianaKruger
MelanieLaurent
BradPitt
Sky
2
DianaKruger
NormanReedus
Mark
MelanieLaurent
BradPitt
DianaKruger
NormanReedus
```

This is what the other player saw:

```
2 2
DianaKruger
MelanieLaurent
BradPitt
NormanReedus
IngloriousBasterds
3
DianaKruger
MelanieLaurent
BradPitt
Sky
2
DianaKruger
NormanReedus
Veronique
MelanieLaurent
BradPitt
DianaKruger
NormanReedus
IGiveUp
```