ETH Zürich FS 2017
Institute of Theoretical Computer Science
Prof. Angelika Steger, Prof. Emo Welzl
Dr. Johannes Lengler
Nemanja Škorić, Miloš Trujić

Algorithmen und Wahrscheinlichkeit
Programming Exercises 2

Deadline: 10:15 @ 23.3.2017

Exercise 1 – *Reconstructing the password*

We all know that remembering passwords is not easy, but writing down a password on a piece of paper is not secure enough. That is why you decided to write down your password in an intricate way. The password is an $n + 2$ characters string $a_1 a_2 \ldots a_{n+2}$ and you wrote down on n separate papers all the possible three letter continuous substrings: $a_1a_2a_3$, $a_2a_3a_4$, …, $a_na_{n+1}a_{n+2}$. However, over time you forgot your original password as well as the order of the papers containing the three letter substrings. To make things worse, you are not even sure if you didn't lose some of the papers. Although not a perfect check, you want to check if you can construct at least one string out of the papers you have.

Input The first line of the input file contains an integer $1 \le t \le 20$ denoting the number of test cases that follow. Each of the t test cases is described as follows.

- It starts with a line containing an integer `n`, denoting the number of papers you found, such that $1 \le n \le 5 \cdot 10^3$.
- The next n lines contain three letter strings denoting the substring written on the papers.

Output For each test case output on a separate line `yes` if there exists a string of length $n + 2$ such that after applying the procedure explained above you get exactly the n different substrings from the input. Otherwise, you should output `no`.

Points This exercise gives 1 point.

Sample Input

```
4
3
abc
cde
bcd
3
abc
bca
bcd
2
aaa
bbb
1
aaa
```

Sample Output

```
yes
no
no
yes
```

Exercise 2 – *Dining table*

You decided to organise a dinner party at your home. Since you are an outgoing person your friend pool is very large, but unfortunately not all of your friends are acquainted with each other. In order for the evening to be successful and entertaining for everybody you make the following plan.

For each guest, you write down the list of people she is a friend with. You know that the friendships are a symmetric relation, i.e. that if a person a is a friend of a person b then the person b is a friend of the person a.

Your plan is simple: you want to place each guest on one of the two sides of your table and furthermore, you want to place all her friends on the other side of the table, exactly across, so that the conversations are flowing smoothly. You are sure that your table is large enough, but from a first glance at your friendship lists you cannot deduce whether such a placement is possible.

Luckily, you are a fairly good programmer and decide to rely on your programming skills to check if your plan for the evening can happen. Moreover, if your seating plan can work out, you want to know who will be seated on the same side of the table as your best friend.

Input The first line of the input file contains an integer $1 \leq t \leq 20$ denoting the number of test cases that follow. Each of the t test cases is described as follows.

- It starts with a line containing three integers `n m r`, separated by space, denoting the number of people attending your dinner party, the total number of friendships, and the name of your best friend, such that $1 \leq n \leq 10^4$, $1 \leq m \leq 10^5$, and $0 \leq r \leq n-1$.
- The next m lines contain two integers `a b`, separated by space, indicating that the person a is a friend with the person b (and the other way around, remember – friendships are symmetric!), where $0 \leq a, b \leq n-1$.

Output For each test case output one line with the people seated on the same side of the table as your best friend r ordered increasingly while respecting your seating plan, or `no` if your seating plan is not possible.

Points This exercise gives 1 point.

Sample Input

```
3
4 4 1
0 1
1 2
2 3
3 0
9 8 5
0 1
0 2
1 3
1 4
1 5
2 6
6 7
6 8
```

Sample Output

```
1 3
0 3 4 5 6
no
```

```
5 7 0
0 1
1 2
2 3
3 4
4 0
3 1
0 3
```

Challenge Exercise

This is a challenge exercise. No bonus points can be obtained for this exercise, but the first three students to solve it will win a book prize. You are only eligible for the book prize if the solution is entirely your own. *If you want to compete for the prize, then do not gather any hints (or solutions, obviously) from your colleagues or from any other external sources.*

The goal of the exercise is to solve the following problem from the online UVa judge: `https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=18&page=show_problem&problem=1571`

First you need to create an account with the username matching your nethz account or your real name on the UVa judge. Upon successful submission, send a proof of your submission and the judge approving it, along with the exact time of your submission, to either `Nemanja Škorić` or `Miloš Trujić`.

The first three students with the correct submission may choose one of the following books as a prize:

- Peter Winkler: Mathematical Puzzles - A Connoisseur's Collection
- Béla Bollobás: The Art of Mathematics - Coffee Time in Memphis
- Simon Singh: Fermats letzter Satz
- Martin Aigner und Günter Ziegler: Das Buch der Beweise
- Angelika Steger: Diskrete Strukturen 1: Kombinatorik, Graphentheorie, Algebra

Points This exercise gives no bonus points.