(Adv.) Competitive Programming

Submit until end of contest, via the judge interface

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Problem: contest-design (1 second timelimit)

It is the year 2100. After numerous iterations of (Advanced) Competitive Programming, the problem library of the CompProg team has grown to a staggering n problems! Nowadays, designing a contest simply consists of picking the right problems from this library.

The problems in the library are ordered by difficulty and tagged with some of the 16 tags describing different categories. These include things like `dp`, `geometry`, or, even worse, `bit-magic`. To identify the tags, hexadecimal digits are used, i.e. 0 – 9 and a – f.

To create a contest, some number of problems are picked in ascending difficulty. Going from one problem to the next in this order, each problem should share a tag with the previous one. Since students usually work on problems from easiest to hardest, this allows them to find something familiar in each new problem they attempt.

directed Graph path

The simplest part is always picking the easiest and hardest problem for a contest, based on the proficiency of the participants. However, you have found that even with the large library of problems, it is not always possible to find a set of problems between those two that satisfies the criteria from above. This happens even when you allow an arbitrary number of problems to be picked for the contest. Since this happens so often, you now want to automate this check.

Input The input begins with a line containing n ($2 \leq n \leq 2 \cdot 10^5$), the number of problems, and q ($1 \leq q \leq 2 \cdot 10^5$), the number of contest ideas to evaluate. The next n lines contain the problems in ascending difficulty, where no two problems are considered of equal difficulty. Each problem consists of a tag string, containing the hexadecimal digits of the tags the problem is tagged with, in ascending order. Every problem is tagged with at least one tag.

der Graph explodiert?

DP, vielleicht

oder RMQ?

After that, q lines each contain a contest idea in the form of e and h ($1 \leq e < h \leq n$), the position in the problem list of the easiest and hardest problem respectively.

Output For each contest idea, output `yes` if a set of increasing difficulty problems starting at e and ending at h exist, where each two adjacent problems share at least one tag. Otherwise, print `no`.

eine aufsteigende Folge von a nach b

von a nach b laufen, schauen, welche Tags ich erreichen kann, und ob die mich zum Ziel führen

vielleicht doch RMQ

Sample input

```
4 3
01
12
23
34
1 2
2 4
1 4
```

Sample output

```
yes
yes
yes
```

```
5 3
07
79d
9
79
d
1 4
1 5
3 5
```

```
yes
yes
no
```

Sample notes In the first sample, every problem shares a tag with the next easier one, so we can simply use every problem between e and h for each contest.

In the second sample, we can build contests with problems $(1, 2, 4)$ for contest idea #1, and $(1, 2, 5)$ for idea #2, however, idea #3 makes it impossible to build a valid contest.