BAPC 2024

The 2024 Benelux Algorithm Programming Contest

Problems

A “Aaawww...” or “Aaayyy!!!”

Time limit: 2s

You are a hardcore supporter for one of the teams at the Benelux Algorithm Programming Contest (BAPC). Unfortunately, the staff found you entering the contest area and popping other teams' balloons. As a result, you were kicked out of the building and you will not be allowed back in until the contest is over. As a hardcore supporter, you would not want to miss the award ceremony and not know the final rank of your favourite team while everyone inside already knows! Luckily, you know a way to follow the award ceremony from outside the building.

The audience going “Aaawww. . . ”. © VIA FotoCollectief on Flickr, used with permission

One hour before the end of the contest, the scoreboard freezes. Since you can access the frozen scoreboard online, you know the rank of each team and for which problems they have accepted, rejected or pending submissions. During the award ceremony, the results of all pending submissions are revealed, starting with the leftmost pending submission of the lowest ranking team with a pending submission. After every reveal, the scoreboard is updated and the next pending submission is chosen in the same way.

The BAPC is known for its great audience participation during the award ceremony. When the result of a pending submission is about to be revealed, the audience chants “Ooohhh. . . ” in anticipation. When the submission is rejected, the audience utters a sad “Aaawww. . . ”. When the submission is accepted, the crowd goes wild with an “Aaayyy!!!”. If an accepted submission causes a team to rise in the ranking, this chant goes on for longer. Specifically, an extra ‘y’ is added for each team they pass. For example, if they pass five teams, the audience will yell “Aaayyyyyyyy!!!” (with eight ‘y’s). While you cannot make out any words of the announcer, you can clearly hear the enthusiastic audience from outside. You want to use this to determine the final rank of your favourite team.

Input

The input consists of:

- One line with three integers n, m, and r ($2 \leq n \leq 100$, $1 \leq m \leq 100$, $1 \leq r \leq n$), the number of teams, the number of problems, and the rank of your favourite team in the frozen scoreboard.
- n lines with m characters, each character being either ‘`A`’, ‘`R`’, ‘`P`’, or ‘`N`’, indicating that a team's submission is either accepted, rejected, pending, or nothing. The teams are sorted by their rank in the frozen scoreboard, i.e. descending by number of accepted submissions in the frozen scoreboard, and further tie-breaking rules guarantee that all teams have a distinct rank.
- For each pending submission, one line with two strings, containing the audience chant at the reveal of the pending submission, in chronological order. The first string is “`Ooohhh...`” and the second string is either “`Aaawww...`” or “`Aaayyy!!!`”, with an additional ‘`y`’ for each team that is passed when rising in the ranking.

Output

Output the final rank of your favourite team.

Sample Input 1

```
2 3 2
AAP
APR
Ooohhh... Aaayyyy!!!
Ooohhh... Aaawww...
```

Sample Output 1

```
1
```

Sample Input 2

```
2 3 2
AAP
APR
Ooohhh... Aaayyyy!!!
Ooohhh... Aaayyyy!!!
```

Sample Output 2

```
2
```

Sample Input 3

```
4 4 3
AAPP
PNAA
PPAA
NAPN
Ooohhh... Aaayyyy!!!
Ooohhh... Aaayyyyyy!!!
Ooohhh... Aaawww...
Ooohhh... Aaayyy!!!
Ooohhh... Aaayyyy!!!
Ooohhh... Aaayyyy!!!
```

Sample Output 3

```
1
```

B Buggy Blinkers

Time limit: 4s

Recently, your car underwent a software update. Now, if you use the blinkers too much, the car shuts down, reporting a "buffer overflow", whatever that means! On the bright side, you are now welcome at the Broken-down Automobile Preservation Convention (BAPC).

A car indicating to turn left. CC BY-SA 3.0 by Scheinwerfermann on Wikimedia Commons

You found out late, so you have to drive there as quickly as possible! Still, of course, you have to obey all traffic rules. At each intersection, you should follow these rules, regardless of whether an intersection has roads in all directions or not:

- When turning left (or right) at an intersection, the left (or right) blinker must be on.
- When driving straight ahead, the blinkers must be off.
- U-turns are not allowed, i.e. you are not allowed to turn back the way you came.

To play it safe with your blinkers, you decide you are going to activate them at most k times. Luckily, you can still deactivate them at any time. This seems rather limiting, but you make one shrewd observation: as long as you keep your blinkers on (they do not turn off automatically), you can keep turning in the same direction.

The road network consists of intersections with roads between them. Roads always start and end in one of the four cardinal directions: north, east, south, or west. Furthermore, they never start and end at the same intersection. As an example, consider sample cases 1 through 3, visualized in Figure B.1 (next page). These samples only differ in their value of k.

To simplify navigation, you assume that each road can be traversed in the same amount of time, i.e. each road is considered to be of length 1. Find the shortest route from your current location to the BAPC, ensuring that you do not activate the blinkers more than k times. From your current location, you can drive in any direction without using your blinkers.

Input

The input consists of:

- One line with two integers n and k ($1 \leq n \leq 5000$, $0 \leq k \leq 20$), the number of intersections and the number of times the blinkers can be activated.
- n lines, the ith of which contains four integers v_i^{N}, v_i^{E}, v_i^{S}, and v_i^{W} ($0 \leq v_i^{\mathrm{N}}, v_i^{\mathrm{E}}, v_i^{\mathrm{S}}, v_i^{\mathrm{W}} \leq n$), the intersections that can be reached by taking the north, east, south, and west road from intersection i, or 0 to indicate that the road does not exist.

You start at intersection 1, and the BAPC is located at intersection n. Each intersection i has at most one road to each other intersection j. If this road exists, then intersection j has exactly one road to intersection i as well.

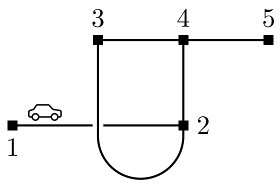


Figure B.1: Visualization of the first, second, and third sample input.

Output

If it is possible to drive from intersection 1 to n using the blinkers at most k times, output the length of the shortest such route. Otherwise, output "`impossible`".

Sample Input 1

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

Sample Output 1

```
3
```

Sample Input 2

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

Sample Output 2

```
4
```

Sample Input 3

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

Sample Output 3

```
impossible
```

Sample Input 4

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

Sample Output 4

```
impossible
```

C Concurrent Contests

Time limit: 4s

Because of some scheduling issues, there are m different programming contests scheduled on the same day, at the same time. There are n people who would like to participate in these contests, but because all contests happen simultaneously, each participant can only compete in one contest. Everyone wants to choose the contest in which to participate such that their expected winnings are maximized.

Prizes for all the different contests you can win. CC BY-SA 4.0 by JFS-Chatt on Wikimedia Commons

Every contest has a single cash prize for the winner (no-one else gets a prize). Furthermore, every participant has a skill level, which determines their winning probability. If the sum of skill levels over all participants in a contest is t, then the winning probability in this contest of a participant with skill level s is $\frac{s}{t}$.

Find a distribution of the participants over the contests, such that it is impossible for any person to switch to a different contest and increase their expected winnings. It is guaranteed that such a distribution exists.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 2 \cdot 10^5$, $1 \leq m \leq 100$), the number of contestants and the number of contests.
- One line with n integers s ($1 \leq s \leq 10^9$), the skill levels of the contestants.
- One line with m integers p ($1 \leq p \leq 10^9$), the prizes for the winners of the contests.

The sum of all skill levels is at most 10^9.

Output

For each contest, output the number of contestants that should participate in this contest, followed by the 1-based indices of the contestants that should participate in this contest.

If there are multiple valid solutions, you may output any one of them.

Sample Input 1

```
6 3
2 5 10 3 7 1
100 50 75
```

Sample Output 1

```
4 4 2 6 1
1 5
1 3
```

Sample Input 2

```
3 2
9 10 8
10 100
```

Sample Output 2

```
0
3 2 3 1
```

D Disgruntled Diner

Time limit: 2s

Tickets allow Diana to oversee kitchen operations. Unsplash Licence by Daniel Bradley on Unsplash

Diana is head chef at the Batavian Authentic Prestigious Cuisine. All orders are logged to a central computer, but to organize the kitchen, each ordered item is also printed on a separate *ticket*. On the back of each ticket, Diana writes the corresponding table number, so servers can easily check where they need to deliver the food. When a ticket is completed, Diana pins it to a board. Of course, there could be duplicate tickets, since a table may order the same item multiple times.

Halfway through service, a disgruntled diner complains: "We have been waiting for hours, but so far we were *only* served tomato soup! Could you please hurry up?" Diana must address this immediately, since the reputation of the restaurant is at stake! However, in the past, some customers have been dishonest in order to demand expedited service. So, before Diana instructs the kitchen staff, the complaint must be verified.

Since the board is always up-to-date, verification is a matter of checking the relevant tickets. Let t and m be the table number and menu item that describe the complaint, respectively. Diana seeks to verify the following claim: "All pinned-up tickets for table t correspond to menu item m." In particular, if there are no completed tickets for table t, Diana considers the claim to be true – the customer may have misspoken, but they surely deserve extra attention!

Unfortunately, on the board, only one side of each ticket is visible (either the menu item or table number) and Diana does not have time to flip hundreds of tickets. However, by cross-referencing with the central computer, it may be possible to safely ignore certain tickets. Help Diana determine the minimum set of pinned-up tickets that need to be flipped. Or can you (dis)prove the claim without flipping any tickets? You must decide which tickets should be flipped before Diana starts doing so – she is too strained to make deductions on the fly.

Input

The input consists of:

- One line with two integers n and k ($1 \leq k \leq n \leq 500$), the number of orders in the computer and the number of pinned-up tickets.
- One line with n strings representing the orders in the computer, each string consisting of one English uppercase letter (the menu item, `A-Z`) and one digit (the table number, `0-9`).
- One line with k characters representing the pinned-up tickets, each character being either an English uppercase letter or a digit.
- One line with a digit t (`0-9`) and an English uppercase letter m (`A-Z`), specifying the claim: "All pinned-up tickets for table t correspond to menu item m."

The pinned-up tickets are guaranteed to correspond to (a subset of) the orders in the computer.

Output

If, without flipping any tickets, the claim is provably true or false, output “`true`” or “`false`”, respectively. Otherwise, output the minimum number of pinned-up tickets that need to be flipped, followed by their 1-based indices, in any order.

Sample Input 1

```
6 4
A1 A2 B1 B2 C1 C2
A B 1 2
1 A
```

Sample Output 1

```
2
3 2
```

Sample Input 2

```
5 4
A1 B1 C2 C1 A2
2 A B 1
1 A
```

Sample Output 2

```
false
```

Sample Input 3

```
4 4
Z0 Z0 F9 F9
Z 0 9 9
4 F
```

Sample Output 3

```
true
```

E Extraterrestrial Exploration

Time limit: 1s

After years of planning and construction, you finally succeeded in making your own spacecraft! Immediately hopping aboard, you take the spaceship on a maiden voyage to the Big Anthropomorphic Pig Constellation. After a couple of years, you reach the constellation and land on the nearest planet to take some pictures and hopefully score some souvenirs. While haggling with a local souvenir seller using the built-in translator of your spacecraft, you are suddenly notified that the craft is low on fuel! Your extensive use of the translator has drained more fuel than expected, and you cannot get back to Earth. Frantically, you look for a refuelling station. Luckily, a local points you to a shady store, strikingly similar to the petrol stations you know from home.

The inspiration for your spaceship. CC BY-SA 3.0 by Edwtie on Wikimedia Commons

While the outside of the refuelling station may look similar to those on Earth, the inside is completely different. On a long shelf are a number of fuel canisters, with strange symbols on the side. From your research on rocket fuel, you deduce that these symbols probably denote the oxydilation level of the fuel in the canister. None of the rocket fuels burn on their own. Instead, combining two rocket fuels with oxydilation levels o_x and o_y yields a fuel with burn time $\sqrt{|o_x - o_y|}$. You can afford to buy three full canisters, and the burn time of rocket fuel is additive, so that combining rocket fuels with oxydilation levels o_x, o_y and o_z results in a total burn time of

$$\sqrt{|o_x - o_y|} + \sqrt{|o_y - o_z|} + \sqrt{|o_z - o_x|}.$$

You can only decode the symbols denoting the oxydilation levels with the translator of your spacecraft. Unfortunately, due to the low fuel levels of the spacecraft, you can only use your translator for 50 items, before the fuel fully runs out. Luckily, you know that rocket fuel is always stored in non-decreasing order of oxydilation levels. Can you figure out which three canisters of rocket fuel to buy to maximize total burn time?

Interaction

This is an interactive problem. Your submission will be run against an *interactor*, which reads from the standard output of your submission and writes to the standard input of your submission. This interaction needs to follow a specific protocol:

The interactor first sends one line with an integer n ($3 \leq n \leq 2 \cdot 10^5$), the number of types of rocket fuel.

Then, your program should make at most 50 queries to find the optimal three fuel canisters. Each query is made by printing one line of the form "`? i`" ($1 \leq i \leq n$). The interactor will respond with one line with an integer o_i ($|o_i| \leq 10^6$), the oxydilation value of the fuel in the ith canister.

When you have determined the optimal three *distinct* canisters x, y, and z ($1 \leq x, y, z \leq n$, $x \neq y$, $x \neq z$, $y \neq z$), print one line of the form "! x y z", after which the interaction will stop. Printing the answer does not count as a query.

If there are multiple valid solutions, you may output any one of them.

The interactor is not adaptive: the oxydilation levels of the fuel canisters are fixed up front, and do not depend on your queries.

Make sure you flush the buffer after each write.

A testing tool is provided to help you develop your solution.

Using more than 50 queries will result in a wrong answer.

Read **Sample Interaction 1** **Write**

Read	Write
`4`	
	`? 1`
`1`	
	`? 3`
`3`	
	`? 4`
`3`	
	`! 1 2 3`

Read **Sample Interaction 2** **Write**

Read	Write
`6`	
	`? 1`
`-5`	
	`? 6`
`5`	
	`? 2`
`-3`	
	`? 5`
`3`	
	`? 3`
`-1`	
	`? 4`
`1`	
	`! 4 6 1`

F Failing Factory

Time limit: 4s

All steps in your gigafactory working together in harmony (until something fails, of course). CC BY 2.0 by Steve Jurvetson on Flickr

The gigafactory for your new range of Battery-Assisted Postal Cars is finally up and running. This manufacturing plant is a highly complex facility, consisting of many individual steps, where the parts of each car are milled, stamped, welded, soldered, screwed, glued, assembled, tested, detailed, layered, painted, and cleaned. Every step is optimized to the tiniest detail, making them very complicated.

As you are preparing for a visit from your main investor, alarm bells start going off. One of the steps failed, causing a cascade of failures across the factory! After hurriedly resolving the failures, panic creeps up to you: what if a failure happens during the visit of the investor?

Currently, all processes in the factory are working, but your engineers determined that each of them has some independent probability of failing before the visit. As the visit is soon, there will be no time for any repairs, and as soon as a step fails, this will quickly halt all dependent steps as well.

Thus, you decide to show only a single processing step of your factory, and specifically, the one with the smallest probability of failing. As an example, consider the second sample case. The probability that step 1 fails is 0.72, but step 2 is slightly more stable with a failure probability of 0.6. Thus, you show step 2 to your investor, with a probability of 0.4 that it will *not* fail.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 10^5$, $0 \leq m \leq 10^5$), the number of steps and the number of dependencies between steps.
- One line with n floating point numbers p ($0 \leq p \leq 1$), the individual failure probability of each step. Each probability is given in decimal form[1] with exactly three digits after the decimal point.
- m lines, each with two integers a and b ($1 \leq a, b \leq n$, $a \neq b$), indicating that step a depends on step b: failure of step b will cause failure of step a.
  A direct dependency of one step on another occurs at most once.
  Cyclic dependencies are allowed.

Output

For the step with the smallest probability of failing, output the probability that it will *not* fail.

Your answer should have an *absolute* error of at most 10^{-200} or a *relative* error of at most 10^{-6}.

[1]When a floating-point number is written in decimal form, it is not in scientific notation.

Sample Input 1

```
2 2
0.600 0.300
1 2
2 1
```

Sample Output 1

```
0.28
```

Sample Input 2

```
2 1
0.300 0.600
1 2
```

Sample Output 2

```
0.4
```

Sample Input 3

```
4 3
0.999 0.994 0.998 0.996
1 2
2 3
3 4
```

Sample Output 3

```
0.004
```

Sample Input 4

```
4 4
0.999 0.994 0.998 0.996
1 2
2 3
3 4
4 1
```

Sample Output 4

```
4.8e-11
```

G Grocery Greed

Time limit: 2s

Recently, you have acquired the newest book in the self-help category: "Becoming A Professional Consumer", containing a wide variety of tips on how to buy as much as possible, while paying as little as possible. One of the things that you already discovered while reading the book is that you have been paying too much for your groceries all your life!

Paying by card for *only* the milk.
CC PDM 1.0 by U.S. Department of Agriculture on Flickr

This works as follows: in a supermarket, you can decide to pay with card or with cash. If you pay with cash, the amount you have to pay gets rounded to the nearest multiple of €0.05, and if you pay with card, it does not. So, depending on your groceries, it can be cheaper if you pay with the right method! You can minimize your spendings even further by splitting your groceries into multiple groups, and paying separately for every group.

You have already decided on a list of the things that you are going to buy, and you know their prices. What is the cheapest way to buy all these groceries?

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 2 \cdot 10^5$), the number of items you want to buy.
- One line with n floating point numbers p ($0.05 \leq p \leq 100.00$), the prices of the items in euros. Each price is given in decimal form[1] with exactly two decimal places.

Output

Output the minimal total amount of money you need to buy all the groceries, in euros. Your answer should have exactly two decimal places.

Sample Input 1	Sample Output 1
3 0.59 5.21 3.10	8.89

Sample Input 2	Sample Output 2
5 20.43 1.11 6.47 19.99 3.75	51.70

Sample Input 3	Sample Output 3
2 0.05 0.14	0.19

[1]When a floating-point number is written in decimal form, it is not in scientific notation.

Sample Input 4

```
4
1.00 3.00 5.00 2.00
```

Sample Output 4

```
11.00
```

Sample Input 5

```
3
68.79 61.18 0.58
```

Sample Output 5

```
130.53
```

H Horse Habitat

Time limit: 25s

Harold has inherited a huge habitat with hundreds of horses! He wants to train a handful of horses for the Bareback Arizona Phoenix Cowboys, which is a half-yearly happening honouring Arizonan horse riding history. Hence, Harold signed his horses up for the Hurdle Hopping event and he has requested your help handling the training program.

Harold hopes he herds honoured horses, just like *ome Loeks*. CC BY-SA 3.0 by Bouwe Brouwer on Wikimedia Commons

Hurdle Hopping courses have many possible layouts, each requiring a different rectangular area. However, not all of the land in the habitat is suitable for courses. Horses, moreover, need to train courses on multiple different grounds in order to learn to adapt to possible circumstances. Handling the training program, it is thus relevant that courses can be rebuilt in many different locations.

Handed to you is a map showing the habitat as a grid of unit squares with each square indicating whether the land is suitable for courses or not. Help Harold by answering a list of questions, each question asking the total number of possible locations in the habitat for a Hurdle Hopping course with a particular size.

Input

The input consists of:

- One line with three integers r, c, and q ($1 \leq r, c \leq 9 \cdot 10^6$, $r \cdot c \leq 9 \cdot 10^6$, $1 \leq q \leq 10^5$), the number of rows and columns of the grid, and the number of questions.
- r lines with c characters, each character being either '`.`' if the corresponding square indicates land suitable for courses or '`#`' otherwise.
- q lines, each with two integers h and w ($1 \leq h \leq r$, $1 \leq w \leq c$), indicating a question from Harold about the number of Hurdle Hopping courses with height h (number of rows in the grid) and width w (number of columns in the grid).

Output

For each of the q questions, output the number of possible locations for a grid-aligned Hurdle Hopping course of the requested height h (number of rows in the grid) and width w (number of columns in the grid).

Sample Input 1	Sample Output 1
1 7 1 #....#. 1 2	3

Sample Input 2

```
3 3 6
..#
#..
...
1 1
1 2
2 1
3 1
2 2
3 3
```

Sample Output 2

```
7
4
3
1
1
0
```

Sample Input 3

```
2 3 6
...
...
1 1
1 2
2 1
2 2
1 3
2 3
```

Sample Output 3

```
6
4
3
2
2
1
```

Sample Input 4

```
3 5 5
.....
..#..
.....
2 2
1 1
1 5
3 1
1 3
```

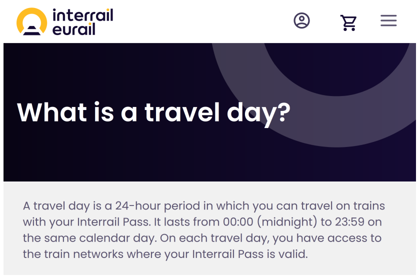
Sample Output 4

```
4
14
2
4
6
```

I Interrail Pass

Time limit: 2s

Interrail passes are the fun and cheap way to see more of Europe, especially if you combine your train trip with Businesslike And Penny-saving Computation! In particular, you would like to find the cheapest way to pay for your planned travels. You plan to take the train on n travel days, that are not necessarily consecutive. The individual fare is different for every day, and perhaps you can save money by buying some interrail passes.



What is a travel day?

A travel day is a 24-hour period in which you can travel on trains with your Interrail Pass. It lasts from 00:00 (midnight) to 23:59 on the same calendar day. On each travel day, you have access to the train networks where your Interrail Pass is valid.

Definition of a travel day from interrail.eu. © Eurail B.V., used non-commercially

There are k different types of interrail passes with varying costs. Each type of interrail pass can be obtained multiple times. An interrail pass is active for a period of p consecutive days, that starts on a day of your choice. The interrail pass covers the first d travel days during this period, which do not have to be consecutive. Note that an active interrail pass cannot be "paused": a day of travel counts towards the day count of each active pass, even when you pay the individual fare that day.

As an example, consider the fourth sample input, visualized in Figure I.1. It is definitely cheaper to buy interrail passes than to pay 4 individual fares. The cheapest solution is to buy two interrail passes of the first type, rather than one interrail pass of the second type.

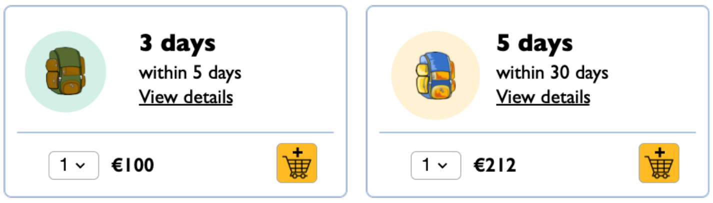


Figure I.1: Visualization of the types of interrail passes for the fourth sample input in a webshop. The first one can be activated for a period of 5 days, and can be used for 3 days within that period. The second one has a period of 30 days, and can be used for 5 days during that period.

Input

The input consists of:

- One line with two integers n and k ($1 \leq n \leq 10\,000$, $0 \leq k \leq 100$), the number of planned travel days, and the number of types of interrail passes available.
- n lines, each with two integers t and f ($0 \leq t \leq 10^6$, $1 \leq f \leq 10^5$), the travel day and the individual fare for that day. The n travel days are distinct and given in increasing order.
- k lines, each with three integers p, d, and c ($1 \leq p \leq 10^6$, $1 \leq d \leq p$, $1 \leq c \leq 10^5$), indicating a type of interrail pass that is valid for a period of p days, covers the first d travel days in that period, and costs c.

Output

Output the minimum amount you need to spend to cover all your planned travels.

Sample Input 1

```
2 1
0 10
1 10
2 2 15
```

Sample Output 1

```
15
```

Sample Input 2

```
2 1
0 10
2 10
2 2 15
```

Sample Output 2

```
20
```

Sample Input 3

```
3 1
0 10
1 10
2 10
5 2 15
```

Sample Output 3

```
25
```

Sample Input 4

```
4 2
3 80
5 90
24 70
26 60
5 3 100
30 5 212
```

Sample Output 4

```
200
```

Sample Input 5

```
4 1
42 9
43 2
44 9
45 9
4 3 20
```

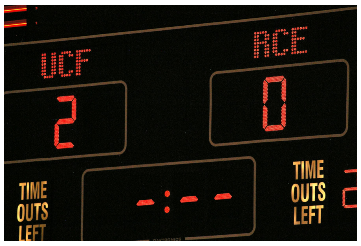
Sample Output 5

```
29
```

J Jumbled Scoreboards

Time limit: 1s

You were so hyped to attend the final game of the Ball And Paddle Competition, where the two best teams in the world compete to paddle as many balls into the opponent's goal as possible. But alas, you fell ill, and cannot join your friends. Luckily, your friends took lots of pictures during the match, and after the match concluded, they sent you all the pictures that they have. Because the messaging app uploads and downloads the pictures in parallel, you are wondering whether you received them in chronological order. It looks like the scoreboards in each picture are unique, and knowing that the score of a team can only increase over time, you should be able to figure this out. Feeling too sick to check the order of the pictures manually, you decide to write a program that checks temporal consistency based on the scoreboards that are in the picture.



One of the pictures that you received for the first sample input. CC BY 2.0 by Adam Baker on Flickr, modified

Given a list of intermediate scores from the match, determine whether the scores are in chronological order.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 100$), the number of pictures you received.
- n lines, each with two integers a and b ($0 \leq a, b \leq 100$), the scores of the two teams in one of the pictures.

Every pair of scores (a, b) in the input is unique.

The order of the scores in the input is the order in which you received the pictures.

Output

Output "`yes`" if the scores are in chronological order, or "`no`" if they are not.

Sample Input 1	Sample Output 1

4
1 0
2 0
4 0
4 1
	yes

Sample Input 2

```
3
0 0
1 0
0 2
```

Sample Output 2

```
no
```

Sample Input 3

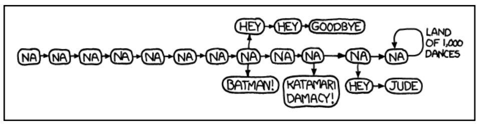
```
5
1 2
0 0
4 3
2 3
5 5
```

Sample Output 3

```
no
```

K Karaoke Compression

Time limit: 10s



Flowcharts are another compression method that you have considered. CC BY-NC 2.5 by Randall Munroe on xkcd.com

Next week, you will be hosting the Biannual Acoustic Popsong Convention. Of course, this convention also needs to include a karaoke night, featuring all your favourite acoustic pop songs! To impress all attendees, you have decided to prepare by learning the lyrics of all the songs by heart. But there is a problem: these lyrics are very long, so you will not have enough time left to learn all of this! However, you have noticed that a lot of the songs contain quite some repetitions. This gives you the idea of first compressing the lyrics, and then only learning the compressed version.

The compression scheme you will use works as follows. Let s be the string to compress. You select exactly one nonempty substring t of the lyrics, and replace as many occurrences of t in s as possible by a new character that did not occur in s before. Call the result of this s'. Now you only need to remember the substring t and the compressed string s'. You would like to know the minimal total length of these two strings, if you compress the lyrics in this manner.

As an example, consider the first sample case. In this case, you want to compress the lyrics "`nananananananananabatman`". If you replace the substring "`na`" by the character "`X`", the compressed string becomes "`XXXXXXXXbatman`". The total length of the substring and compressed string in this case is $2 + 14 = 16$. However, if you instead choose to replace the substring "`nana`", then the compressed string is "`XXXXbatman`" and the total length is $4 + 10 = 14$, which is optimal.

Input

The input consists of:

- One line with a string s ($1 \leq |s| \leq 5000$), the lyrics to compress.
  The string only consists of English lowercase letters (`a-z`).

Output

Output the minimal total length of the replaced substring and the compressed string.

Sample Input 1	Sample Output 1
`nananananananananabatman`	`14`

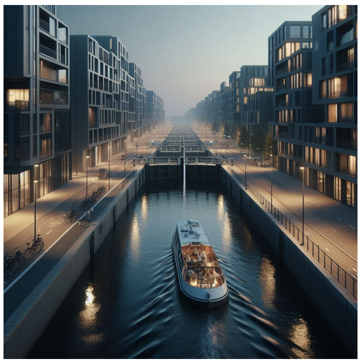
Sample Input 2	Sample Output 2
`abcabd`	`6`

Sample Input 3	Sample Output 3
`nocompression`	`14`

L Levelling Locks

Time limit: 3s

Oh, snap! A recent power outage not only left Groningen in darkness, but even worse, it caused a complete system failure of the city's historic and obsolete lock system, closing off the waterways. The lock consists of a series of identical chambers, with a gate between each pair of adjacent chambers that can be opened or closed. The system failure, however, caused all the gates to remain closed, blocking the water flow between the chambers. Normally, this would not bother you, but you are eagerly awaiting a shipment of your favourite snack: eierballen.


The ship carrying your favourite snack, eierballen, just moments before the power outage. Generated using Microsoft Copilot Designer

Fortunately, Lotte, a professional scuba diver, is up to the task of repairing the old lock. She devised a plan that is as simple as it is daring:

Lotte is currently aboard a helicopter en route to the lock. Upon arrival, she will dive into the cold water of a chamber of her choosing and start opening gates manually. By swimming in the already connected chambers, she can open the next closed gate to either her left or her right, forcing the water levels to equalize between the connected chambers.

However, swimming in deep waters is dangerous and should be avoided where possible. The danger of her mission can be quantified by the maximum depth she must swim to open all gates. Clearly, Lotte cannot avoid swimming in the water depth that eventually arises when all chambers are connected. But can she avoid swimming in any deeper water?

As an example, consider the first sample case, visualized in Figure L.1. When connecting chambers in the order given by the sample answer, Lotte will never swim in any water that is deeper than the final water level.

Help Lotte determine the order in which to connect the chambers to open all gates without exceeding the final water depth, or determine that it is impossible.

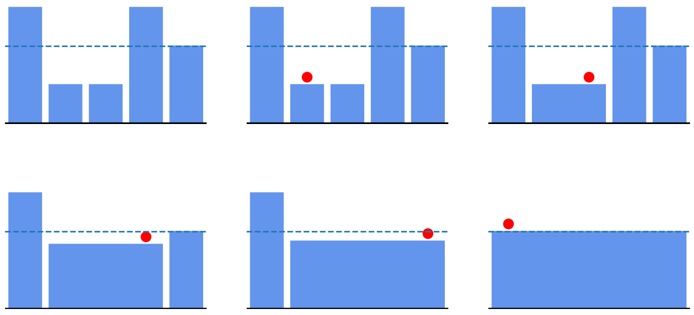


Figure L.1: Visualization of the first sample case. The horizontal dashed line indicates the final water level. Lotte can connect all chambers without swimming in any water that is deeper than this level.

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 2 \cdot 10^5$), the number of chambers.
- One line with n integers a ($1 \leq a \leq 10^8$), the water level in each chamber.

The space occupied by the gates is negligibly small.

Output

If it is possible to open all gates without exceeding the final water depth, output the order in which Lotte first enters, thus connects, each of the n chambers. Otherwise, output "`impossible`".

If there are multiple valid solutions, you may output any one of them.

Sample Input 1

```
5
3 1 1 3 2
```

Sample Output 1

```
2 3 4 5 1
```

Sample Input 2

```
3
1 2 1
```

Sample Output 2

```
impossible
```

M Museum Visit

Time limit: 4s

Every day is different in the *Groninger Museum*. Some days are nice, peaceful and quiet, and you can spend all day looking at the beautiful paintings, sculptures, and other artworks. Other days are busier, when weekends or public holidays fill the museum with hasty visitors, increased prices and screaming children. This discomfort varies a lot: some busy days are better because of extra *studentenkorting* (student discount) and some of the quiet days get worse because of earthquake risks.

The Groninger Museum, as seen from the canals. CC BY-SA 4.0 by Rob Koster on Wikimedia Commons

The museum also regularly hosts special limited-time exhibitions, such as those on the local football club FC Groningen, the Martinitoren, or the eierbal (a local delicacy). These exhibitions can be very irregular: some last for weeks, some last only a day, and there may be multiple exhibitions on the same day.

As a proud *Grunneger*, you want to visit each exhibition at least once. Luckily, you are subscribed to the newsletter so you know the start and end days of all the exhibitions in advance. Additionally, since you are a regular visitor at the museum, you have observed all the crowd and earthquake patterns, so you know exactly how much discomfort you will receive when you visit the museum on any specific day.

On which days should you visit the museum in order to minimize your total discomfort while still seeing all exhibitions that are planned in the foreseeable future? As an example, consider the first sample case. To minimize your total discomfort, you should visit the first two exhibitions on the second day and the last exhibition on the fourth or fifth day.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n, m \leq 2 \cdot 10^5$), the number of days in the foreseeable future and the number of exhibitions planned in those days.
- One line with n integers c ($1 \leq c \leq 10^9$), describing for each day the discomfort you will receive when you visit the museum.
- m lines, each with two integers s and e ($1 \leq s \leq e \leq n$), describing the start and end day of an exhibition. The start and end days are inclusive: the exhibition can be visited on day s, day e, and any day in between.

Output

Output the minimum total discomfort you will receive when visiting all exhibitions of the Groninger Museum.

Sample Input 1

```
5 3
1 1 3 1 1
1 3
2 3
3 5
```

Sample Output 1

```
2
```

Sample Input 2

```
6 3
1 2 4 4 2 1
1 4
2 5
3 6
```

Sample Output 2

```
3
```

Sample Input 3

```
11 2
3 1 4 1 5 9 2 6 5 3 5
5 10
1 1
```

Sample Output 3

```
5
```