Codeforces Round #617 (Div. 3)

A. Array with Odd Sum

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array a consisting of n integers.

In one move, you can choose two indices $1 \le i, j \le n$ such that $i \ne j$ and set $a_i := a_j$. You can perform such moves any number of times (possibly, zero). You can choose different indices in different operations. The operation := is the operation of assignment (i.e. you choose i and j and replace a_i with a_j).

Your task is to say if it is possible to obtain an array with an odd (not divisible by 2) sum of elements.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 2000$) — the number of test cases.

The next $2t$ lines describe test cases. The first line of the test case contains one integer n ($1 \le n \le 2000$) — the number of elements in a. The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 2000$), where a_i is the i-th element of a.

It is guaranteed that the sum of n over all test cases does not exceed 2000 ($\sum n \le 2000$).

Output

For each test case, print the answer on it — "YES" (without quotes) if it is possible to obtain the array with an odd sum of elements, and "NO" otherwise.

Example

input
5 2 2 3 4 2 2 8 8 3 3 3 3 4 5 5 5 5 4 1 1 1 1
output
YES NO YES NO NO

B. Food Buying

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Mishka wants to buy some food in the nearby shop. Initially, he has s burles on his card.

Mishka can perform the following operation any number of times (possibly, zero): choose some **positive integer number** $1 \le x \le s$, buy food that costs exactly x burles and obtain $\lfloor \frac{x}{10} \rfloor$ burles as a cashback (in other words, Mishka spends x burles and obtains $\lfloor \frac{x}{10} \rfloor$ back). The operation $\lfloor \frac{a}{b} \rfloor$ means a divided by b rounded down.

It is guaranteed that you can always buy some food that costs x for any possible value of x.

Your task is to say the maximum number of burles Mishka can spend if he buys food optimally.

For example, if Mishka has $s = 19$ burles then the maximum number of burles he can spend is 21. Firstly, he can spend $x = 10$ burles, obtain 1 burle as a cashback. Now he has $s = 10$ burles, so can spend $x = 10$ burles, obtain 1 burle as a cashback and spend it too.

You have to answer t independent test cases.

Input
The first line of the input contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

The next t lines describe test cases. Each test case is given on a separate line and consists of one integer s ($1 \le s \le 10^9$) — the number of burles Mishka initially has.

Output
For each test case print the answer on it — the maximum number of burles Mishka can spend if he buys food optimally.

Example

input
6 1 10 19 9876 12345 1000000000
output
1 11 21 10973 13716 1111111111

C. Yet Another Walking Robot

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is a robot on a coordinate plane. Initially, the robot is located at the point $(0, 0)$. Its path is described as a string s of length n consisting of characters 'L', 'R', 'U', 'D'.

Each of these characters corresponds to some move:

- 'L' (left): means that the robot moves from the point (x, y) to the point $(x - 1, y)$;
- 'R' (right): means that the robot moves from the point (x, y) to the point $(x + 1, y)$;
- 'U' (up): means that the robot moves from the point (x, y) to the point $(x, y + 1)$;
- 'D' (down): means that the robot moves from the point (x, y) to the point $(x, y - 1)$.

The company that created this robot asked you to optimize the path of the robot somehow. To do this, you can remove **any non-empty substring** of the path. But this company doesn't want their customers to notice the change in the robot behavior. It means that if before the optimization the robot ended its path at the point (x_e, y_e), then after optimization (i.e. removing some single substring from s) the robot also ends its path at the point (x_e, y_e).

This optimization is a low-budget project so you need to remove **the shortest** possible **non-empty** substring to optimize the robot's path such that the endpoint of his path doesn't change. It is possible that you can't optimize the path. Also, it is possible that after the optimization the target path is an empty string (i.e. deleted substring is the whole string s).

Recall that the substring of s is such string that can be obtained from s by removing some amount of characters (possibly, zero) from the prefix and some amount of characters (possibly, zero) from the suffix. For example, the substrings of "LURLLR" are "LU", "LR", "LURLLR", "URL", but not "RR" and "UL".

You have to answer t independent test cases.

Input
The first line of the input contains one integer t ($1 \le t \le 1000$) — the number of test cases.

The next $2t$ lines describe test cases. Each test case is given on two lines. The first line of the test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the robot's path. The second line of the test case contains one string s consisting of n characters 'L', 'R', 'U', 'D' — the robot's path.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$).

Output
For each test case, print the answer on it. If you cannot remove such **non-empty** substring that the endpoint of the robot's path doesn't change, print -1. Otherwise, print two integers l and r such that $1 \le l \le r \le n$ — endpoints of the substring you remove. The value $r - l + 1$ should be minimum possible. If there are several answers, print any of them.

Example

input

```
4
4
LRUD
4
LURD
5
RRUDU
5
LLDDR
```

output

```
1 2
1 4
3 4
-1
```

D. Fight with Monsters

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n monsters standing in a row numbered from 1 to n. The i-th monster has h_i health points (hp). You have your attack power equal to a hp and your opponent has his attack power equal to b hp.

You and your opponent are fighting these monsters. Firstly, you and your opponent go to the first monster and fight it till his death, then you and your opponent go the second monster and fight it till his death, and so on. A monster is considered dead if its hp is less than or equal to 0.

The fight with a monster happens in turns.

1. You hit the monster by a hp. If it is dead after your hit, **you gain one point** and you both proceed to the next monster.
2. Your opponent hits the monster by b hp. If it is dead after his hit, **nobody gains a point** and you both proceed to the next monster.

You have some secret technique to force your opponent to skip his turn. You can use this technique at most k times **in total** (for example, if there are two monsters and $k = 4$, then you can use the technique 2 times on the first monster and 1 time on the second monster, but not 2 times on the first monster and 3 times on the second monster).

Your task is to determine the maximum number of points you can gain if you use the secret technique optimally.

Input

The first line of the input contains four integers n, a, b and k ($1 \le n \le 2 \cdot 10^5, 1 \le a, b, k \le 10^9$) — the number of monsters, your attack power, the opponent's attack power and the number of times you can use the secret technique.

The second line of the input contains n integers $h_1, h_2, \dots, h_n$ ($1 \le h_i \le 10^9$), where h_i is the health points of the i-th monster.

Output

Print one integer — the maximum number of points you can gain if you use the secret technique optimally.

Examples

input

```
6 2 3 3
7 10 50 12 1 8
```

output

```
5
```

input

```
1 1 100 99
100
```

output

```
1
```

input

```
7 4 2 1
1 3 5 4 2 7 6
```

output

```
6
```

E1. String Coloring (easy version)

time limit per test: 1 second
memory limit per test: 256 megabytes

input: standard input
output: standard output

This is an easy version of the problem. The actual problems are different, but the easy version is almost a subtask of the hard version. Note that the constraints and the output format are different.

You are given a string s consisting of n lowercase Latin letters.

You have to color **all** its characters **one of the two colors** (each character to exactly one color, the same letters can be colored the same or different colors, i.e. you can choose exactly one color for each index in s).

After coloring, you can swap **any** two neighboring characters of the string that are colored **different** colors. You can perform such an operation arbitrary (possibly, zero) number of times.

The goal is to make the string sorted, i.e. all characters should be in alphabetical order.

Your task is to say if it is possible to color the given string so that after coloring it can become sorted by **some** sequence of swaps. Note that you have to restore only coloring, not the sequence of swaps.

Input
The first line of the input contains one integer n ($1 \le n \le 200$) — the length of s.

The second line of the input contains the string s consisting of exactly n lowercase Latin letters.

Output
If it is impossible to color the given string so that after coloring it can become sorted by **some** sequence of swaps, print "NO" (without quotes) in the first line.

Otherwise, print "YES" in the first line and **any** correct coloring in the second line (the coloring is the string consisting of n characters, the i-th character should be '0' if the i-th character is colored the first color and '1' otherwise).

Examples

input
9 abacbecfd
output
YES 001010101

input
8 aaabbcbb
output
YES 01011011

input
7 abcdedc
output
NO

input
5 abcde
output
YES 00000

E2. String Coloring (hard version)

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is a hard version of the problem. The actual problems are different, but the easy version is almost a subtask of the hard version. Note that the constraints and the output format are different.

You are given a string s consisting of n lowercase Latin letters.

You have to color **all** its characters **the minimum number of colors** (each character to exactly one color, the same letters can be colored the same or different colors, i.e. you can choose exactly one color for each index in s).

After coloring, you can swap **any** two neighboring characters of the string that are colored **different** colors. You can perform such an operation arbitrary (possibly, zero) number of times.

The goal is to make the string sorted, i.e. all characters should be in alphabetical order.

Your task is to find the minimum number of colors which you have to color the given string in so that after coloring it can become sorted by **some** sequence of swaps. Note that you have to restore only coloring, not the sequence of swaps.

Input
The first line of the input contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the length of s.

The second line of the input contains the string s consisting of exactly n lowercase Latin letters.

Output
In the first line print one integer res ($1 \le res \le n$) — the minimum number of colors in which you have to color the given string so that after coloring it can become sorted by **some** sequence of swaps.

In the second line print **any** possible coloring that can be used to sort the string using some sequence of swaps described in the problem statement. The coloring is the array c of length n, where $1 \le c_i \le res$ and c_i means the color of the i-th character.

Examples

input
9 abacbecfd
output
2 1 1 2 1 2 1 2 1 2

input
8 aaabbcbb
output
2 1 2 1 2 1 2 1 1

input
7 abcdedc
output
3 1 1 1 1 1 2 3

input
5 abcde
output
1 1 1 1 1 1

F. Berland Beauty

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n railway stations in Berland. They are connected to each other by $n - 1$ railway sections. The railway network is connected, i.e. can be represented as an undirected tree.

You have a map of that network, so for each railway section you know which stations it connects.

Each of the $n - 1$ sections has some integer value of the *scenery beauty*. However, these values are not marked on the map and you don't know them. All these values are from 1 to 10^6 inclusive.

You asked m passengers some questions: the j-th one told you three values:

- his departure station a_j;
- his arrival station b_j;
- minimum scenery beauty along the path from a_j to b_j (the train is moving along the shortest path from a_j to b_j).

You are planning to update the map and set some value f_i on each railway section — the *scenery beauty*. The passengers' answers should be consistent with these values.

Print any valid set of values $f_1, f_2, \ldots, f_{n-1}$, which the passengers' answer is consistent with or report that it doesn't exist.

Input

The first line contains a single integer n ($2 \le n \le 5000$) — the number of railway stations in Berland.

The next $n-1$ lines contain descriptions of the railway sections: the i-th section description is two integers x_i and y_i ($1 \le x_i, y_i \le n, x_i \ne y_i$), where x_i and y_i are the indices of the stations which are connected by the i-th railway section. All the railway sections are bidirected. Each station can be reached from any other station by the railway.

The next line contains a single integer m ($1 \le m \le 5000$) — the number of passengers which were asked questions. Then m lines follow, the j-th line contains three integers a_j, b_j and g_j ($1 \le a_j, b_j \le n$; $a_j \ne b_j$; $1 \le g_j \le 10^6$) — the departure station, the arrival station and the minimum scenery beauty along his path.

Output

If there is no answer then print a single integer -1.

Otherwise, print $n-1$ integers $f_1, f_2, \ldots, f_{n-1}$ ($1 \le f_i \le 10^6$), where f_i is some valid scenery beauty along the i-th railway section.

If there are multiple answers, you can print any of them.

Examples

input
4 1 2 3 2 3 4 2 1 2 5 1 3 3
output
5 3 5

input
6 1 2 1 6 3 1 1 5 4 1 4 6 1 3 3 4 1 6 5 2 1 2 5
output
5 3 1 2 1

input
6 1 2 1 6 3 1 1 5 4 1 4 6 1 1 3 4 3 6 5 3 1 2 4
output
-1