Technocup 2020 - Elimination Round 4

A. Happy Birthday, Polycarp!

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

Hooray! Polycarp turned n years old! The Technocup Team sincerely congratulates Polycarp!

Polycarp celebrated all of his n birthdays: from the 1-th to the n-th. At the moment, he is wondering: how many times he turned *beautiful* number of years?

According to Polycarp, a positive integer is *beautiful* if it consists of only one digit repeated one or more times. For example, the following numbers are beautiful: 1, 77, 777, 44 and 999999. The following numbers are not beautiful: 12, 11110, 6969 and 987654321.

Of course, Polycarpus uses the decimal numeral system (i.e. radix is 10).

Help Polycarpus to find the number of numbers from 1 to n (inclusive) that are beautiful.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input. Then t test cases follow.

Each test case consists of one line, which contains a positive integer n ($1 \le n \le 10^9$) — how many years Polycarp has turned.

Output

Print t integers — the answers to the given test cases in the order they are written in the test. Each answer is an integer: the number of beautiful years between 1 and n, inclusive.

Example

input
6 18 1 9 100500 33 1000000000
output
10 1 9 45 12 81

Note

In the first test case of the example beautiful years are 1, 2, 3, 4, 5, 6, 7, 8, 9 and 11.

B. Make Them Odd

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

There are n positive integers $a_1, a_2, \ldots, a_n$. For the one move you can choose any even value c and divide by two **all** elements that equal c.

For example, if $a = [6, 8, 12, 6, 3, 12]$ and you choose $c = 6$, and a is transformed into $a = [3, 8, 12, 3, 3, 12]$ after the move.

You need to find the minimal number of moves for transforming a to an array of only odd integers (each element shouldn't be divisible by 2).

Input

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

The first line of a test case contains n ($1 \le n \le 2 \cdot 10^5$) — the number of integers in the sequence a. The second line contains positive integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

The sum of n for all test cases in the input doesn't exceed $2 \cdot 10^5$.

Output
For t test cases print the answers in the order of test cases in the input. The answer for the test case is the minimal number of moves needed to make **all** numbers in the test case odd (i.e. not divisible by 2).

Example

input
4 6 40 6 40 3 20 1 1 1024 4 2 4 8 16 3 3 1 7
output
4 10 4 0

Note
In the first test case of the example, the optimal sequence of moves can be as follows:

- before making moves $a = [40, 6, 40, 3, 20, 1]$;
- choose $c = 6$;
- now $a = [40, 3, 40, 3, 20, 1]$;
- choose $c = 40$;
- now $a = [20, 3, 20, 3, 20, 1]$;
- choose $c = 20$;
- now $a = [10, 3, 10, 3, 10, 1]$;
- choose $c = 10$;
- now $a = [5, 3, 5, 3, 5, 1]$ — all numbers are odd.

Thus, all numbers became odd after 4 moves. In 3 or fewer moves, you cannot make them all odd.

C. As Simple as One and Two

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

You are given a non-empty string $s = s_1 s_2 \ldots s_n$, which consists only of lowercase Latin letters. Polycarp does not like a string if it contains at least one string "one" or at least one string "two" (or both at the same time) as a **substring**. In other words, Polycarp does not like the string s if there is an integer j ($1 \le j \le n-2$), that $s_j s_{j+1} s_{j+2}$ ="one" or $s_j s_{j+1} s_{j+2}$ ="two".

For example:

- Polycarp does not like strings "oneee", "ontwow", "twone" and "oneonetwo" (they all have at least one substring "one" or "two"),
- Polycarp likes strings "oonnee", "twwwo" and "twnoe" (they have no substrings "one" and "two").

Polycarp wants to select a certain set of indices (positions) and remove all letters on these positions. All removals are made at the same time.

For example, if the string looks like s ="onetwone", then if Polycarp selects two indices 3 and 6, then "on**e**tw**o**ne" will be selected and the result is "ontwne".

What is the minimum number of indices (positions) that Polycarp needs to select to make the string liked? What should these positions be?

Input
The first line of the input contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input. Next, the test cases are given.

Each test case consists of one non-empty string s. Its length does not exceed $1.5 \cdot 10^5$. The string s consists only of lowercase Latin letters.

It is guaranteed that the sum of lengths of all lines for all input data in the test does not exceed $1.5 \cdot 10^6$.

Output
Print an answer for each test case in the input in order of their appearance.

The first line of each answer should contain r ($0 \le r \le |s|$) — the required minimum number of positions to be removed, where $|s|$

is the length of the given line. The second line of each answer should contain r different integers — the indices themselves for removal in any order. Indices are numbered from left to right from 1 to the length of the string. If $r = 0$, then the second line can be skipped (or you can print empty). If there are several answers, print any of them.

Examples

input
4 onetwone testme oneoneone twotwo
output
2 6 3 0 3 4 1 7 2 1 4

input
10 onetwonetwooneoonetwooo two one twooooo ttttwo ttwwoo ooone onnne oneeeee oneeeeeeetwooooo
output
6 18 11 12 1 6 21 1 1 1 3 1 2 1 6 0 1 4 0 1 1 2 1 11

Note

In the first example, answers are:

- "on**e**tw**o**ne",
- "testme" — Polycarp likes it, there is nothing to remove,
- "**o**ne**o**ne**o**ne",
- "**t**wo**t**wo".

In the second example, answers are:

- "**o**netw**o**netw**oo**neooo**n**e**t**wooo",
- "**t**wo",
- "**o**ne",
- "**t**wooooo",
- "tttt**w**o",
- "ttwwoo" — Polycarp likes it, there is nothing to remove,
- "ooo**n**e",
- "onnne" — Polycarp likes it, there is nothing to remove,
- "**o**neeeee",
- "**o**neeeeeee**t**wooooo".

D. Let's Play the Words?

time limit per test: 3 seconds

memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has n **different** binary words. A word called binary if it contains only characters '0' and '1'. For example, these words are binary: "0001", "11", "0" and "0011100".

Polycarp wants to offer his set of n binary words to play a game "words". In this game, players name words and each next word (starting from the second) must start with the last character of the previous word. The first word can be any. For example, these sequence of words can be named during the game: "0101", "1", "10", "00", "00001".

Word reversal is the operation of reversing the order of the characters. For example, the word "0111" after the reversal becomes "1110", the word "11010" after the reversal becomes "01011".

Probably, Polycarp has such a set of words that there is no way to put them in the order correspondent to the game rules. In this situation, he wants to reverse some words from his set so that:

- the final set of n words still contains **different** words (i.e. all words are unique);
- there is a way to put all words of the final set of words in the order so that the final sequence of n words is consistent with the game rules.

Polycarp wants to reverse minimal number of words. Please, help him.

Input

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

The first line of a test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of words in the Polycarp's set. Next n lines contain these words. All of n words aren't empty and contains only characters '0' and '1'. The sum of word lengths doesn't exceed $4 \cdot 10^6$. All words are **different**.

Guaranteed, that the sum of n for all test cases in the input doesn't exceed $2 \cdot 10^5$. Also, guaranteed that the sum of word lengths for all test cases in the input doesn't exceed $4 \cdot 10^6$.

Output

Print answer for all of t test cases in the order they appear.

If there is no answer for the test case, print -1. Otherwise, the first line of the output should contain k ($0 \le k \le n$) — the minimal number of words in the set which should be reversed. The second line of the output should contain k distinct integers — the indexes of the words in the set which should be reversed. Words are numerated from 1 to n in the order they appear. If $k = 0$ you can skip this line (or you can print an empty line). If there are many answers you can print any of them.

Example

input
4 4 0001 1000 0011 0111 3 010 101 0 2 00000 00001 4 01 001 0001 00001
output
1 3 -1 0 2 1 2

E. Two Fairs

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

There are n cities in Berland and some pairs of them are connected by two-way roads. It is guaranteed that you can pass from any city to any other, moving along the roads. Cities are numerated from 1 to n.

Two fairs are currently taking place in Berland — they are held in two different cities a and b ($1 \le a, b \le n$; $a \ne b$).

Find the number of pairs of cities x and y ($x \ne a, x \ne b, y \ne a, y \ne b$) such that if you go from x to y you will have to go through both fairs (the order of visits doesn't matter). Formally, you need to find the number of pairs of cities x, y such that any path from x to y goes through a and b (in any order).

Print the required number of pairs. The order of two cities in a pair does not matter, that is, the pairs (x, y) and (y, x) must be taken into account only once.

Input

The first line of the input contains an integer t ($1 \le t \le 4 \cdot 10^4$) — the number of test cases in the input. Next, t test cases are specified.

The first line of each test case contains four integers n, m, a and b ($4 \le n \le 2 \cdot 10^5$, $n - 1 \le m \le 5 \cdot 10^5$, $1 \le a, b \le n$, $a \ne b$) — numbers of cities and roads in Berland and numbers of two cities where fairs are held, respectively.

The following m lines contain descriptions of roads between cities. Each of road description contains a pair of integers u_i, v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$) — numbers of cities connected by the road.

Each road is bi-directional and connects two different cities. It is guaranteed that from any city you can pass to any other by roads. There can be more than one road between a pair of cities.

The sum of the values of n for all sets of input data in the test does not exceed $2 \cdot 10^5$. The sum of the values of m for all sets of input data in the test does not exceed $5 \cdot 10^5$.

Output

Print t integers — the answers to the given test cases in the order they are written in the input.

Example

input
3 7 7 3 5 1 2 2 3 3 4 4 5 5 6 6 7 7 5 4 5 2 3 1 2 2 3 3 4 4 1 4 2 4 3 2 1 1 2 2 3 4 1
output
4 0 1

F. Beautiful Rectangle

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

You are given n integers. You need to choose a subset and put the chosen numbers in a beautiful rectangle (rectangular matrix). Each chosen number should occupy one of its rectangle cells, each cell must be filled with exactly one chosen number. Some of the n numbers may not be chosen.

A rectangle (rectangular matrix) is called beautiful if in each row and in each column all values are different.

What is the largest (by the total number of cells) beautiful rectangle you can construct? Print the rectangle itself.

Input

The first line contains n ($1 \le n \le 4 \cdot 10^5$). The second line contains n integers ($1 \le a_i \le 10^9$).

Output

In the first line print x ($1 \le x \le n$) — the total number of cells of the required maximum beautiful rectangle. In the second line print p and q ($p \cdot q = x$): its sizes. In the next p lines print the required rectangle itself. If there are several answers, print any.

Examples

input

```
12
3 1 4 1 5 9 2 6 5 3 5 8
```

output

```
12
3 4
1 2 3 5
3 1 5 4
5 6 8 9
```

input

```
5
1 1 1 1 1
```

output

```
1
1 1
1
```

G. Tree Elimination

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Vasya has a tree with n vertices numbered from 1 to n, and $n-1$ edges numbered from 1 to $n-1$. Initially each vertex contains a token with the number of the vertex written on it.

Vasya plays a game. He considers all edges of the tree by increasing of their indices. For every edge he acts as follows:

- If both endpoints of the edge contain a token, remove a token from one of the endpoints and write down its number.
- Otherwise, do nothing.

The result of the game is the sequence of numbers Vasya has written down. Note that there may be many possible resulting sequences depending on the choice of endpoints when tokens are removed.

Vasya has played for such a long time that he thinks he exhausted all possible resulting sequences he can obtain. He wants you to verify him by computing the number of distinct sequences modulo $998\,244\,353$.

Input

The first line contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of vertices of the tree.

The next $n-1$ lines describe edges of the tree. The i-th of these lines contains two integers u_i, v_i ($1 \le u_i, v_i \le n$) — endpoints of the edge with index i. It is guaranteed that the given graph is indeed a tree.

Output

Print a single integer — the number of distinct sequences modulo $998\,244\,353$.

Examples

input

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

output

```
5
```

input

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

output

```
10
```

Note

In the first sample case the distinct sequences are $(1), (2,1), (2,3,1), (2,3,4,1), (2,3,4,5)$.

Int the second sample case the distinct sequences are
$(2,6,5,3), (2,6,5,7), (2,6,7,2), (2,6,7,5), (2,7,3), (2,7,5), (7,1,3), (7,1,5), (7,2,3), (7,2,5)$.