Codeforces Round #787 (Div. 3)

A. Food for Animals

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

In the pet store on sale there are:

- a packs of dog food;
- b packs of cat food;
- c packs of universal food (such food is suitable for both dogs and cats).

Polycarp has x dogs and y cats. Is it possible that he will be able to buy food for all his animals in the store? Each of his dogs and each of his cats should receive one pack of suitable food for it.

Input

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

Then t lines are given, each containing a description of one test case. Each description consists of five integers a, b, c, x and y ($0 \le a, b, c, x, y \le 10^8$).

Output

For each test case in a separate line, output:

- YES, if suitable food can be bought for each of x dogs and for each of y cats;
- NO else.

You can output YES and NO in any case (for example, strings yEs, yes, Yes and YES will be recognized as a positive response).

Example

input
7 1 1 4 2 3 0 0 0 0 0 5 5 0 4 6 1 1 1 1 1 50000000 50000000 100000000 100000000 100000000 0 0 0 100000000 100000000 1 3 2 2 5
output
YES YES NO YES YES NO NO

B. Make It Increasing

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Given n integers $a_1, a_2, \ldots, a_n$. You can perform the following operation on them:

- select any element a_i ($1 \le i \le n$) and divide it by 2 (round down). In other words, you can replace any selected element a_i with the value $\left\lfloor \frac{a_i}{2} \right\rfloor$ (where $\lfloor x \rfloor$ is – round down the real number x).

Output the minimum number of operations that must be done for a sequence of integers to become strictly increasing (that is, for the condition $a_1 < a_2 < \cdots < a_n$ to be satisfied). Or determine that it is impossible to obtain such a sequence. Note that elements **cannot** be swapped. The only possible operation is described above.

For example, let $n = 3$ and a sequence of numbers $[3, 6, 5]$ be given. Then it is enough to perform two operations on it:

- Write the number $\left\lfloor \frac{6}{2} \right\rfloor = 3$ instead of the number $a_2 = 6$ and get the sequence $[3, 3, 5]$;
- Then replace $a_1 = 3$ with $\left\lfloor \frac{3}{2} \right\rfloor = 1$ and get the sequence $[1, 3, 5]$.

The resulting sequence is strictly increasing because $1 < 3 < 5$.

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.

The descriptions of the test cases follow.

The first line of each test case contains a single integer n ($1 \le n \le 30$).

The second line of each test case contains exactly n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 2 \cdot 10^9$).

Output
For each test case, print a single number on a separate line — the minimum number of operations to perform on the sequence to make it strictly increasing. If a strictly increasing sequence cannot be obtained, print "-1".

Example

input
7 3 3 6 5 4 5 3 2 1 5 1 2 3 4 5 1 1000000000 4 2 8 7 5 5 8 26 5 21 10 2 5 14
output
2 -1 0 0 4 11 0

Note
The first test case is analyzed in the statement.

In the second test case, it is impossible to obtain a strictly increasing sequence.

In the third test case, the sequence is already strictly increasing.

C. Detective Task

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp bought a new expensive painting and decided to show it to his n friends. He hung it in his room. n of his friends entered and exited there one by one. At one moment there was no more than one person in the room. In other words, the first friend entered and left first, then the second, and so on.

It is known that at the beginning (before visiting friends) a picture hung in the room. At the end (after the n-th friend) it turned out that it disappeared. At what exact moment it disappeared — there is no information.

Polycarp asked his friends one by one. He asked each one if there was a picture when he entered the room. Each friend answered one of three:

- *no* (response encoded with 0);
- *yes* (response encoded as 1);
- *can't remember* (response is encoded with ?).

Everyone except the thief either doesn't remember or told the **truth**. The thief can say anything (any of the three options).

Polycarp cannot understand who the thief is. He asks you to find out the number of those who can be considered a thief according to the answers.

Input

The first number t ($1 \le t \le 10^4$) — the number of test cases in the test.

The following is a description of test cases.

The first line of each test case contains one string s (length does not exceed $2 \cdot 10^5$) — a description of the friends' answers, where s_i indicates the answer of the i-th friend. Each character in the string is either `0` or `1` or `?`.

The given regularity is described in the actual situation. In particular, on the basis of answers, at least one friend can be suspected of stealing a painting.

It is guaranteed that the sum of string lengths over the entire input data set does not exceed $2 \cdot 10^5$.

Output
Output one positive (strictly more zero) number – the number of people who could steal the picture based on the data shown.

Example

input
8 0 1 1110000 ????? 1?1??0?0 0?0??? ??11 ??0??
output
1 1 2 5 4 1 1 3

Note
In the first case, the answer is 1 since we had exactly 1 friend.

The second case is similar to the first.

In the third case, the suspects are the third and fourth friends (we count from one). It can be shown that no one else could be the thief.

In the fourth case, we know absolutely nothing, so we suspect everyone.

D. Vertical Paths

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a rooted tree consisting of n vertices. Vertices are numbered from 1 to n. Any vertex can be the root of a tree.

A *tree* is a connected undirected graph without cycles. A *rooted tree* is a tree with a selected vertex, which is called the *root*.

The tree is specified by an array of parents p containing n numbers: p_i is a parent of the vertex with the index i. The *parent* of a vertex u is a vertex that is the next vertex on the shortest path from u to the root. For example, on the simple path from 5 to 3 (the root), the next vertex would be 1, so the parent of 5 is 1.

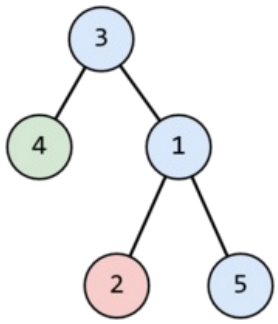
The root has no parent, so for it, the value of p_i is i (the root is the only vertex for which $p_i = i$).

Find such a set of paths that:

- each vertex belongs to exactly one path, each path can contain one or more vertices;
- in each path each next vertex — is a son of the current vertex (that is, paths always lead down — from parent to son);
- number of paths is **minimal**.

For example, if $n = 5$ and $p = [3, 1, 3, 3, 1]$, then the tree can be divided into three paths:

1. $3 \to 1 \to 5$ (path of 3 vertices),
2. 4 (path of 1 vertices).
3. 2 (path of 1 vertices).



Example of splitting a root tree into three paths for $n = 5$, the root of the tree — node 3.

Input

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

Each test case consists of two lines.

The first of them contains an integer n ($1 \le n \le 2 \cdot 10^5$). It is the number of vertices in the tree.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$). It is guaranteed that the p array encodes some rooted tree.

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

Output

For each test case on the first line, output an integer m — the minimum number of non-intersecting leading down paths that can cover all vertices of the tree.

Then print m pairs of lines containing path descriptions. In the first of them print the length of the path, in the second — the sequence of vertices specifying that path in the order from top to bottom. You can output the paths in any order.

If there are several answers, output any of them.

Example

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

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

```
4
```

E. Replace With the Previous, Minimize

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a string s of lowercase Latin letters.

The following operation can be used:

- select one character (from 'a' to 'z') that occurs at least once in the string. And replace all such characters in the string with the previous one in alphabetical order on the loop. For example, replace all 'c' with 'b' or replace all 'a' with 'z'.

And you are given the integer k — the maximum number of operations that can be performed. Find the minimum lexicographically possible string that can be obtained by performing no more than k operations.

The string $a = a_1a_2 \dots a_n$ is lexicographically smaller than the string $b = b_1b_2 \dots b_n$ if there exists an index k ($1 \le k \le n$) such that $a_1 = b_1$, $a_2 = b_2$, ..., $a_{k-1} = b_{k-1}$, but $a_k < b_k$.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) —the number of test cases in the test.

This is followed by descriptions of the test cases.

The first line of each test case contains two integers n and k ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le 10^9$) — the size of the string s and the maximum number of operations that can be performed on the string s.

The second line of each test case contains a string s of length n consisting of lowercase Latin letters.

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

Output

For each test case, output the lexicographically minimal string that can be obtained from the string s by performing **no more** than k operations.

Example

input
4 3 2 cba 4 5 fgde 7 5 gndcafb 4 19 ekyv
output
aaa agaa bnbbabb aapp

F. Vlad and Unfinished Business

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vlad and Nastya live in a city consisting of n houses and $n - 1$ road. From each house, you can get to the other by moving only along the roads. That is, the city is a tree.

Vlad lives in a house with index x, and Nastya lives in a house with index y. Vlad decided to visit Nastya. However, he remembered that he had postponed for later k things that he has to do before coming to Nastya. To do the i-th thing, he needs to come to the a_i-th house, things can be done in any order. In 1 minute, he can walk from one house to another if they are connected by a road.

Vlad does not really like walking, so he is interested what is the minimum number of minutes he has to spend on the road to do all things and then come to Nastya. Houses $a_1, a_2, \dots, a_k$ he can visit in any order. He can visit any house multiple times (if he wants).

Input

The first line of input contains an integer t ($1 \le t \le 10^4$) — the number of input test cases. There is an empty line before each test case.

The first line of each test case contains two integers n and k ($1 \le k \le n \le 2 \cdot 10^5$) — the number of houses and things,

respectively.

The second line of each test case contains two integers x and y ($1 \le x, y \le n$) — indices of the houses where Vlad and Nastya live, respectively.

The third line of each test case contains k integers $a_1, a_2, \ldots, a_k$ ($1 \le a_i \le n$) — indices of houses Vlad need to come to do things.

The following $n - 1$ lines contain description of city, each line contains two integers v_j and u_j ($1 \le u_j, v_j \le n$) — indices of houses connected by road j.

It is guaranteed that the sum of n for all cases does not exceed $2 \cdot 10^5$.

Output

Output t lines, each of which contains the answer to the corresponding test case of input. As an answer output single integer — the minimum number of minutes Vlad needs on the road to do all the things and come to Nastya.

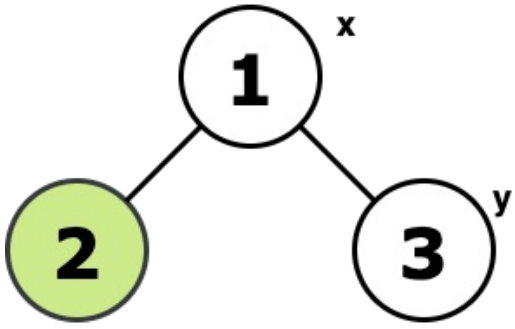
Example

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

output
3 7 2

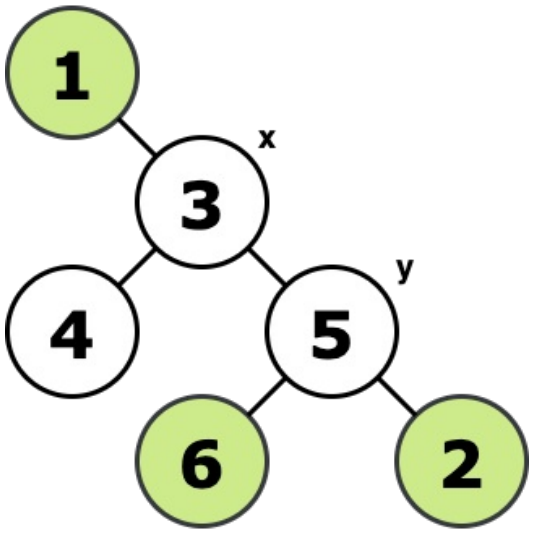
Note

Tree and best path for the first test case:



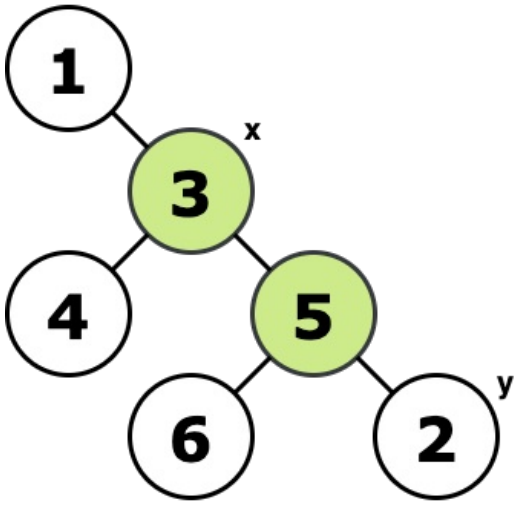
$1 \to 2 \to 1 \to 3$

Tree and best path for the second test case:



$3 \to 1 \to 3 \to 5 \to 2 \to 5 \to 6 \to 5$

Tree and best path for the third test case:



$3 \to 5 \to 2$

G. Sorting Pancakes

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Nastya baked m pancakes and spread them on n dishes. The dishes are in a row and numbered from left to right. She put a_i pancakes on the dish with the index i.

Seeing the dishes, Vlad decided to bring order to the stacks and move some pancakes. In one move, he can shift one pancake from any dish to the closest one, that is, select the dish i ($a_i > 0$) and do one of the following:

- if $i > 1$, put the pancake on a dish with the previous index, after this move $a_i = a_i - 1$ and $a_{i-1} = a_{i-1} + 1$;
- if $i < n$, put the pancake on a dish with the following index, after this move $a_i = a_i - 1$ and $a_{i+1} = a_{i+1} + 1$.

Vlad wants to make the array a**non-increasing**, after moving as few pancakes as possible. Help him find the minimum number of moves needed for this.

The array $a = [a_1, a_2, \ldots, a_n]$ is called non-increasing if $a_i \geq a_{i+1}$ for all i from 1 to $n - 1$.

Input

The first line of the input contains two numbers n and m ($1 \leq n, m \leq 250$) — the number of dishes and the number of pancakes, respectively.

The second line contains n numbers a_i ($0 \leq a_i \leq m$), the sum of all a_i is equal to m.

Output

Print a single number: the minimum number of moves required to make the array a non-increasing.

Examples

input
6 19 5 3 2 3 3 3

output
2

input
3 6 3 2 1
output
0

input
3 6 2 1 3
output
1

input
6 19 3 4 3 3 5 1
output
3

input
10 1 0 0 0 0 0 0 0 0 0 1
output
9

Note

In the first example, you first need to move the pancake from the dish 1, after which $a = [4, 4, 2, 3, 3, 3]$. After that, you need to move the pancake from the dish 2 to the dish 3 and the array will become non-growing $a = [4, 3, 3, 3, 3, 3]$.