Educational Codeforces Round 82 (Rated for Div. 2)

A. Erasing Zeroes

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

You are given a string s. Each character is either `0` or `1`.

You want all `1`'s in the string to form a contiguous subsegment. For example, if the string is `0`, `1`, `00111` or `01111100`, then all `1`'s form a contiguous subsegment, and if the string is `0101`, `100001` or `11111111111101`, then this condition is not met.

You may erase some (possibly none) `0`'s from the string. What is the minimum number of `0`'s that you have to erase?

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of test cases.

Then t lines follow, each representing a test case. Each line contains one string s ($1 \le |s| \le 100$); each character of s is either `0` or `1`.

Output

Print t integers, where the i-th integer is the answer to the i-th testcase (the minimum number of `0`'s that you have to erase from s).

Example

input
3 010011 0 1111000
output
2 0 0

Note

In the first test case you have to delete the third and forth symbols from string `010011` (it turns into `0111`).

B. National Project

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

Your company was appointed to lay new asphalt on the highway of length n. You know that every day you can either repair one unit of the highway (lay new asphalt over one unit of the highway) or skip repairing.

Skipping the repair is necessary because of the climate. The climate in your region is periodical: there are g days when the weather is good and if you lay new asphalt these days it becomes high-quality pavement; after that, the weather during the next b days is bad, and if you lay new asphalt these days it becomes low-quality pavement; again g good days, b bad days and so on.

You can be sure that you start repairing at the start of a good season, in other words, days $1, 2, \ldots, g$ are good.

You don't really care about the quality of the highway, you just want to make sure that **at least half of the highway** will have high-quality pavement. For example, if the $n = 5$ then at least 3 units of the highway should have high quality; if $n = 4$ then at least 2 units should have high quality.

What is the minimum number of days is needed to finish the repair of **the whole highway**?

Input

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

Next T lines contain test cases — one per line. Each line contains three integers n, g and b ($1 \le n, g, b \le 10^9$) — the length of the highway and the number of good and bad days respectively.

Output

Print T integers — one per test case. For each test case, print the minimum number of days required to repair **the whole highway if at least half of it should have high quality**.

Example

input
3 5 1 1 8 10 10 1000000 1 1000000
output
5 8 499999500000

Note

In the first test case, you can just lay new asphalt each day, since days $1, 3, 5$ are good.

In the second test case, you can also lay new asphalt each day, since days 1-8 are good.

C. Perfect Keyboard

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

Polycarp wants to assemble his own keyboard. Layouts with multiple rows are too complicated for him — his keyboard will consist of only one row, where all 26 lowercase Latin letters will be arranged in some order.

Polycarp uses the same password s on all websites where he is registered (it is bad, but he doesn't care). He wants to assemble a keyboard that will allow to type this password very easily. He doesn't like to move his fingers while typing the password, so, for each pair of adjacent characters in s, they should be adjacent on the keyboard. For example, if the password is `abacaba`, then the layout `cabdefghi...` is perfect, since characters `a` and `c` are adjacent on the keyboard, and `a` and `b` are adjacent on the keyboard. It is guaranteed that there are no two adjacent equal characters in s, so, for example, the password cannot be `password` (two characters `s` are adjacent).

Can you help Polycarp with choosing the perfect layout of the keyboard, if it is possible?

Input

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

Then T lines follow, each containing one string s ($1 \le |s| \le 200$) representing the test case. s consists of lowercase Latin letters only. There are no two adjacent equal characters in s.

Output

For each test case, do the following:

- if it is impossible to assemble a perfect keyboard, print `NO` (in upper case, **it matters in this problem**);
- otherwise, print `YES` (in upper case), and then a string consisting of 26 lowercase Latin letters — the perfect layout. Each Latin letter should appear in this string exactly once. If there are multiple answers, print any of them.

Example

input
5 ababa codedoca abcda zxzytyz abcdefghijklmnopqrstuvwxyza
output
YES bacdefghijklmnopqrstuvwxyz YES edocabfghijklmnpqrstuvwxyz NO YES xzytabcdefghijklmnopqrsuvw NO

D. Fill The Bag

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

You have a bag of size n. Also you have m boxes. The size of i-th box is a_i, where each a_i is an integer non-negative power of two.

You can divide boxes into two parts of equal size. Your goal is to fill the bag completely.

For example, if $n = 10$ and $a = [1, 1, 32]$ then you have to divide the box of size 32 into two parts of size 16, and then divide the box of size 16. So you can fill the bag with boxes of size 1, 1 and 8.

Calculate the minimum number of divisions required to fill the bag of size n.

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

The first line of each test case contains two integers n and m ($1 \le n \le 10^{18}, 1 \le m \le 10^5$) — the size of bag and the number of boxes, respectively.

The second line of each test case contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le 10^9$) — the sizes of boxes. It is guaranteed that each a_i is a power of two.

It is also guaranteed that sum of all m over all test cases does not exceed 10^5.

Output
For each test case print one integer — the minimum number of divisions required to fill the bag of size n (or -1, if it is impossible).

Example

input
3 10 3 1 32 1 23 4 16 1 4 1 20 5 2 1 16 1 8
output
2 -1 0

E. Erase Subsequences

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

You are given a string s. You can build new string p from s using the following operation **no more than two times**:

1. choose any subsequence $s_{i_1}, s_{i_2}, \ldots, s_{i_k}$ where $1 \le i_1 < i_2 < \cdots < i_k \le |s|$;
2. erase the chosen subsequence from s (s can become empty);
3. concatenate chosen subsequence to the right of the string p (in other words, $p = p + s_{i_1} s_{i_2} \ldots s_{i_k}$).

Of course, initially the string p is empty.

For example, let $s = \mathrm{ababcd}$. At first, let's choose subsequence $s_1 s_4 s_5 = \mathrm{abc}$ — we will get $s = \mathrm{bad}$ and $p = \mathrm{abc}$. At second, let's choose $s_1 s_2 = \mathrm{ba}$ — we will get $s = \mathrm{d}$ and $p = \mathrm{abcba}$. So we can build abcba from ababcd.

Can you build a given string t using the algorithm above?

Input
The first line contains the single integer T ($1 \le T \le 100$) — the number of test cases.

Next $2T$ lines contain test cases — two per test case. The first line contains string s consisting of lowercase Latin letters ($1 \le |s| \le 400$) — the initial string.

The second line contains string t consisting of lowercase Latin letters ($1 \le |t| \le |s|$) — the string you'd like to build.

It's guaranteed that the total length of strings s doesn't exceed 400.

Output
Print T answers — one per test case. Print YES (case insensitive) if it's possible to build t and NO (case insensitive) otherwise.

Example

input
4 ababcd abcba a b defi fed xyz x
output

```
YES
NO
NO
YES
```

F. Number of Components

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a matrix $n \times m$, initially filled with zeroes. We define $a_{i,j}$ as the element in the i-th row and the j-th column of the matrix.

Two cells of the matrix are *connected* if they share a side, and the elements in these cells are equal. Two cells of the matrix belong to the same *connected component* if there exists a sequence s_1, s_2, ..., s_k such that s_1 is the first cell, s_k is the second cell, and for every $i \in [1, k-1]$, s_i and s_{i+1} are connected.

You are given q queries of the form x_i y_i c_i ($i \in [1, q]$). For every such query, you have to do the following:

1. replace the element $a_{x,y}$ with c;
2. count the number of connected components in the matrix.

There is one additional constraint: for every $i \in [1, q-1]$, $c_i \le c_{i+1}$.

Input

The first line contains three integers n, m and q ($1 \le n, m \le 300$, $1 \le q \le 2 \cdot 10^6$) — the number of rows, the number of columns and the number of queries, respectively.

Then q lines follow, each representing a query. The i-th line contains three integers x_i, y_i and c_i ($1 \le x_i \le n$, $1 \le y_i \le m$, $1 \le c_i \le \max(1000, \lceil \frac{2 \cdot 10^6}{nm} \rceil)$). For every $i \in [1, q-1]$, $c_i \le c_{i+1}$.

Output

Print q integers, the i-th of them should be equal to the number of components in the matrix after the first i queries are performed.

Example

input
3 2 10 2 1 1 1 2 1 2 2 1 1 1 2 3 1 2 1 2 2 2 2 2 2 1 2 3 2 4 2 1 5
output
2 4 3 3 4 4 4 2 2 4

G. Sum of Prefix Sums

time limit per test: 6 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

We define the *sum of prefix sums* of an array $[s_1, s_2, \ldots, s_k]$ as $s_1 + (s_1 + s_2) + (s_1 + s_2 + s_3) + \cdots + (s_1 + s_2 + \cdots + s_k)$.

You are given a tree consisting of n vertices. Each vertex i has an integer a_i written on it. We define the value of the *simple* path from vertex u to vertex v as follows: consider all vertices appearing on the path from u to v, write down all the numbers written on these vertices in the order they appear on the path, and compute the *sum of prefix sums* of the resulting sequence.

Your task is to calculate the maximum value over all paths in the tree.

Input

The first line contains one integer n ($2 \le n \le 150000$) — the number of vertices in the tree.

Then $n - 1$ lines follow, representing the edges of the tree. Each line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), denoting an edge between vertices u_i and v_i. It is guaranteed that these edges form a tree.

The last line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 10^6$).

Output

Print one integer — the maximum value over all paths in the tree.

Example

input
4 4 2 3 2 4 1 1 3 3 7
output
36

Note

The best path in the first example is from vertex 3 to vertex 1. It gives the sequence $[3, 3, 7, 1]$, and the sum of prefix sums is 36.