Educational Codeforces Round 121 (Rated for Div. 2)

A. Equidistant Letters

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

You are given a string s, consisting of lowercase Latin letters. Every letter appears in it no more than twice.

Your task is to rearrange the letters in the string in such a way that for each pair of letters that appear exactly twice, the distance between the letters in the pair is the same. You are not allowed to add or remove letters.

It can be shown that the answer always exists. If there are multiple answers, print any of them.

Input

The first line contains a single integer t ($1 \le t \le 10^3$) — the number of testcases.

Each testcase consists of a non-empty string s, consisting of lowercase Latin letters. Every letter appears in the string no more than twice. The length of the string doesn't exceed 52.

Output

For each testcase, print a single string. Every letter should appear in it the same number of times as it appears in string s. For each pair of letters that appear exactly twice, the distance between the letters in the pair should be the same.

If there are multiple answers, print any of them.

Example

input
3 oelhl abcdcba ac
output
hello ababcdc ac

Note

In the first testcase of the example, the only letter that appears exactly twice is letter 'l'. You can rearrange the letters arbitrarily, since there are no distances to compare.

In the second testcase of the example, the letters that appear exactly twice are 'a', 'b' and 'c'. Initially, letters 'a' are distance 6 apart, letters 'b' are distance 4 apart and letters 'c' are distance 2 apart. They are not the same, so we have to rearrange the letters. After rearrangement, letters 'a' are distance 2 apart, letters 'b' are distance 2 apart and letters 'c' are distance 2 apart. They are all the same, so the answer is valid.

In the third testcase of the example, there are no letters that appear exactly twice. Thus, any rearrangement is valid. Including not changing the string at all.

B. Minor Reduction

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

You are given a decimal representation of an integer x without leading zeros.

You have to perform the following reduction on it **exactly once**: take two neighboring digits in x and replace them with their sum without leading zeros (if the sum is 0, it's represented as a single 0).

For example, if $x = 10057$, the possible reductions are:

- choose the first and the second digits 1 and 0, replace them with $1 + 0 = 1$; the result is 1057;
- choose the second and the third digits 0 and 0, replace them with $0 + 0 = 0$; the result is also 1057;
- choose the third and the fourth digits 0 and 5, replace them with $0 + 5 = 5$; the result is still 1057;
- choose the fourth and the fifth digits 5 and 7, replace them with $5 + 7 = 12$; the result is 10012.

What's the largest number that can be obtained?

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

Each testcase consists of a single integer x ($10 \le x < 10^{200000}$). x doesn't contain leading zeros.

The total length of the decimal representations of x over all testcases doesn't exceed $2 \cdot 10^5$.

Output
For each testcase, print a single integer — the largest number that can be obtained after the reduction is applied exactly once. The number should not contain leading zeros.

Example

input
2 10057 90
output
10012 9

Note
The first testcase of the example is already explained in the statement.

In the second testcase, there is only one possible reduction: the first and the second digits.

C. Monsters And Spells

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

Monocarp is playing a computer game once again. He is a wizard apprentice, who only knows a single spell. Luckily, this spell can damage the monsters.

The level he's currently on contains n monsters. The i-th of them appears k_i seconds after the start of the level and has h_i health points. As an additional constraint, $h_i \le k_i$ for all $1 \le i \le n$. All k_i are different.

Monocarp can cast the spell at moments which are positive integer amounts of second after the start of the level: $1, 2, 3, \dots$. The damage of the spell is calculated as follows. If he didn't cast the spell at the previous second, the damage is 1. Otherwise, let the damage at the previous second be x. Then he can choose the damage to be either $x + 1$ or 1. A spell uses mana: casting a spell with damage x uses x mana. Mana doesn't regenerate.

To kill the i-th monster, Monocarp has to cast a spell with damage at least h_i at the exact moment the monster appears, which is k_i.

Note that Monocarp can cast the spell even when there is no monster at the current second.

The mana amount required to cast the spells is the sum of mana usages for all cast spells. Calculate the least amount of mana required for Monocarp to kill all monsters.

It can be shown that it's always possible to kill all monsters under the constraints of the problem.

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

The first line of the testcase contains a single integer n ($1 \le n \le 100$) — the number of monsters in the level.

The second line of the testcase contains n integers $k_1 < k_2 < \dots < k_n$ ($1 \le k_i \le 10^9$) — the number of second from the start the i-th monster appears at. All k_i are different, k_i are provided in the increasing order.

The third line of the testcase contains n integers $h_1, h_2, \dots, h_n$ ($1 \le h_i \le k_i \le 10^9$) — the health of the i-th monster.

The sum of n over all testcases doesn't exceed 10^4.

Output
For each testcase, print a single integer — the least amount of mana required for Monocarp to kill all monsters.

Example

input
3 1 6 4 2 4 5 2 2 3 5 7 9

2 1 2

output

10 6 7

Note

In the first testcase of the example, Monocarp can cast spells $3, 4, 5$ and 6 seconds from the start with damages $1, 2, 3$ and 4, respectively. The damage dealt at 6 seconds is 4, which is indeed greater than or equal to the health of the monster that appears.

In the second testcase of the example, Monocarp can cast spells $3, 4$ and 5 seconds from the start with damages $1, 2$ and 3, respectively.

In the third testcase of the example, Monocarp can cast spells $4, 5, 7, 8$ and 9 seconds from the start with damages $1, 2, 1, 1$ and 2, respectively.

D. Martial Arts Tournament

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

Monocarp is planning to host a martial arts tournament. There will be three divisions based on weight: lightweight, middleweight and heavyweight. The winner of each division will be determined by a single elimination system.

In particular, that implies that the number of participants in each division should be a power of two. Additionally, each division should have a non-zero amount of participants.

n participants have registered for the tournament so far, the i-th of them weighs a_i. To split participants into divisions, Monocarp is going to establish two integer weight boundaries x and y ($x < y$).

All participants who weigh strictly less than x will be considered lightweight. All participants who weigh greater or equal to y will be considered heavyweight. The remaining participants will be considered middleweight.

It's possible that the distribution doesn't make the number of participants in each division a power of two. It can also lead to empty divisions. To fix the issues, Monocarp can invite an arbitrary number of participants to each division.

Note that Monocarp can't kick out any of the n participants who have already registered for the tournament.

However, he wants to invite as little extra participants as possible. Help Monocarp to choose x and y in such a way that the total amount of extra participants required is as small as possible. Output that amount.

Input

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

The first line of each testcase contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of the registered participants.

The second line of each testcase contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$) — the weights of the registered participants.

The sum of n over all testcases doesn't exceed $2 \cdot 10^5$.

Output

For each testcase, print a single integer — the smallest number of extra participants Monocarp is required to invite after he chooses the weight boundaries x and y.

Example

input

4 4 3 1 2 1 1 1 6 2 2 2 1 1 1 8 6 3 6 3 6 3 6 6

output

0 2 3 2

Note

In the first testcase of the example, Monocarp can choose $x = 2$ and $y = 3$. Lightweight, middleweight and heavyweight divisions will have 2, 1 and 1 participants, respectively. They all are powers of two, so no extra participants are required.

In the second testcase of the example, regardless of the choice of x and y, one division will have 1 participant, the rest will have 0.

Thus, Monocarp will have to invite 1 participant into both of the remaining divisions.

In the third testcase of the example, Monocarp can choose $x = 1$ and $y = 2$. Lightweight, middleweight and heavyweight divisions will have 0, 3 and 3 participants, respectively. So an extra participant is needed in each division.

In the fourth testcase of the example, Monocarp can choose $x = 8$ and $y = 9$. Lightweight, middleweight and heavyweight divisions will have 8, 0 and 0 participants, respectively. Middleweight and heavyweight division need an extra participant each.

E. Black and White Tree

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a tree consisting of n vertices. Some of the vertices (at least two) are black, all the other vertices are white.

You place a chip on one of the vertices of the tree, and then perform the following operations:

- let the current vertex where the chip is located is x. You choose a black vertex y, and then move the chip along the first edge on the simple path from x to y.

You are not allowed to choose the same black vertex y in two operations in a row (i. e., for every two consecutive operations, the chosen black vertex should be different).

You end your operations when the chip moves to the black vertex (if it is initially placed in a black vertex, you don't perform the operations at all), or when the number of performed operations exceeds 100^{500}.

For every vertex i, you have to determine if there exists a (possibly empty) sequence of operations that moves the chip to some black vertex, if the chip is initially placed on the vertex i.

Input

The first line contains one integer n ($3 \le n \le 3 \cdot 10^5$) — the number of vertices in the tree.

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($0 \le c_i \le 1$), where $c_i = 0$ means that the i-th vertex is white, and $c_i = 1$ means that the i-th vertex is black. At least two values of c_i are equal to 1.

Then $n - 1$ lines follow, each of them contains two integers u_i and v_i ($1 \le u_i, v_i \le n$; $u_i \ne v_i$) — the endpoints of some edge. These edges form a tree.

Output

Print n integers. The i-th integer should be equal to 1 if there exists a (possibly empty) sequence of operations that moves the chip to some black vertex if it is placed on the vertex i, and 0 if no such sequence of operations exists.

Example

input
8 0 1 0 0 0 0 1 0 8 6 2 5 7 8 6 5 4 5 6 1 7 3
output
0 1 1 1 1 0 1 1

F. A Random Code Problem

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an integer array $a_0, a_1, \ldots, a_{n-1}$, and an integer k. You perform the following code with it:

```
long long ans = 0; // create a 64-bit signed variable which is initially equal to 0
for(int i = 1; i <= k; i++)
{
 int idx = rnd.next(0, n - 1); // generate a random integer between 0 and n - 1, both inclusive
                               // each integer from 0 to n - 1 has the same probability of being chosen
 ans += a[idx];
 a[idx] -= (a[idx] % i);
}
```

Your task is to calculate the expected value of the variable ans after performing this code.

Note that the input is generated according to special rules (see the input format section).

Input

The only line contains six integers n, a_0, x, y, k and M ($1 \le n \le 10^7$; $1 \le a_0, x, y < M \le 998244353$; $1 \le k \le 17$).

The array a in the input is constructed as follows:

- a_0 is given in the input;
- for every i from 1 to $n - 1$, the value of a_i can be calculated as $a_i = (a_{i-1} \cdot x + y) \bmod M$.

Output

Let the expected value of the variable ans after performing the code be E. It can be shown that $E \cdot n^k$ is an integer. You have to output this integer modulo 998244353.

Examples

input
3 10 3 5 13 88
output
382842030

input
2 15363 270880 34698 17 2357023
output
319392398

Note

The array in the first example test is $[10, 35, 22]$. In the second example, it is $[15363, 1418543]$.