Codeforces Round #702 (Div. 3)

A. Dense Array

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

Polycarp calls an array dense if the greater of any two adjacent elements is not more than twice bigger than the smaller. More formally, for any i ($1 \le i \le n-1$), this condition must be satisfied:

$$\frac{\max(a[i], a[i+1])}{\min(a[i], a[i+1])} \le 2$$

For example, the arrays $[1, 2, 3, 4, 3]$, $[1, 1, 1]$ and $[5, 10]$ are dense. And the arrays $[5, 11]$, $[1, 4, 2]$, $[6, 6, 1]$ are **not** dense.

You are given an array a of n integers. What is the minimum number of numbers you need to add to an array to make it dense? You can insert numbers anywhere in the array. If the array is already dense, no numbers need to be added.

For example, if $a = [4, 2, 10, 1]$, then the answer is 5, and the array itself after inserting elements into it may look like this: $a = [4, 2, \underline{\mathbf{3}}, \underline{\mathbf{5}}, 10, \underline{\mathbf{6}}, \underline{\mathbf{4}}, \underline{\mathbf{2}}, 1]$ (there are other ways to build such a).

Input

The first line contains one integer t ($1 \le t \le 1000$). Then t test cases follow.

The first line of each test case contains one integer n ($2 \le n \le 50$) — the length of the array a.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 50$).

Output

For each test case, output one integer — the minimum number of numbers that must be added to the array to make it dense.

Example

input
6 4 4 2 10 1 2 1 3 2 6 1 3 1 4 2 5 1 2 3 4 3 12 4 31 25 50 30 20 34 46 42 16 15 16
output
5 1 2 1 0 3

Note

The first test case is explained in the statements.

In the second test case, you can insert one element, $a = [1, \underline{\mathbf{2}}, 3]$.

In the third test case, you can insert two elements, $a = [6, \underline{\mathbf{4}}, \underline{\mathbf{2}}, 1]$.

In the fourth test case, you can insert one element, $a = [1, \underline{\mathbf{2}}, 4, 2]$.

In the fifth test case, the array a is already dense.

B. Balanced Remainders

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

output: standard output

You are given a number n (**divisible by** 3) and an array $a[1 \dots n]$. In one move, you can increase any of the array elements by one. Formally, you choose the index i ($1 \le i \le n$) and **replace** a_i with $a_i + 1$. You can choose the same index i multiple times for different moves.

Let's denote by c_0, c_1 and c_2 the number of numbers from the array a that have remainders 0, 1 and 2 when divided by the number 3, respectively. Let's say that the array a has balanced remainders if c_0, c_1 and c_2 are equal.

For example, if $n = 6$ and $a = [0, 2, 5, 5, 4, 8]$, then the following sequence of moves is possible:

- initially $c_0 = 1$, $c_1 = 1$ and $c_2 = 4$, these values are not equal to each other. Let's increase a_3, now the array $a = [0, 2, 6, 5, 4, 8]$;
- $c_0 = 2$, $c_1 = 1$ and $c_2 = 3$, these values are not equal. Let's increase a_6, now the array $a = [0, 2, 6, 5, 4, 9]$;
- $c_0 = 3$, $c_1 = 1$ and $c_2 = 2$, these values are not equal. Let's increase a_1, now the array $a = [1, 2, 6, 5, 4, 9]$;
- $c_0 = 2$, $c_1 = 2$ and $c_2 = 2$, these values are equal to each other, which means that the array a has balanced remainders.

Find the minimum number of moves needed to make the array a have balanced remainders.

Input

The first line contains one integer t ($1 \le t \le 10^4$). Then t test cases follow.

The first line of each test case contains one integer n ($3 \le n \le 3 \cdot 10^4$) — the length of the array a. It is guaranteed that the number n is divisible by 3.

The next line contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 100$).

It is guaranteed that the sum of n over all test cases does not exceed $150\,000$.

Output

For each test case, output one integer — the minimum number of moves that must be made for the a array to make it have balanced remainders.

Example

input
4 6 0 2 5 5 4 8 6 2 0 2 1 0 0 9 7 1 3 4 2 10 3 9 6 6 0 1 2 3 4 5
output
3 1 3 0

Note

The first test case is explained in the statements.

In the second test case, you need to make one move for $i = 2$.

The third test case you need to make three moves:

- the first move: $i = 9$;
- the second move: $i = 9$;
- the third move: $i = 2$.

In the fourth test case, the values c_0, c_1 and c_2 initially equal to each other, so the array a already has balanced remainders.

C. Sum of Cubes

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

You are given a positive integer x. Check whether the number x is representable as the sum of the cubes of two positive integers.

Formally, you need to check if there are two integers a and b ($1 \le a, b$) such that $a^3 + b^3 = x$.

For example, if $x = 35$, then the numbers $a = 2$ and $b = 3$ are suitable ($2^3 + 3^3 = 8 + 27 = 35$). If $x = 4$, then no pair of numbers a and b is suitable.

Input

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

Each test case contains one integer x ($1 \le x \le 10^{12}$).

Please note, that the input for some test cases won't fit into 32-bit integer type, so you should use at least 64-bit integer type in your programming language.

Output

For each test case, output on a separate line:

- "YES" if x is representable as the sum of the cubes of two positive integers.
- "NO" otherwise.

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

Example

input
7 1 2 4 34 35 16 703657519796
output
NO YES NO NO YES YES YES

Note

The number 1 is not representable as the sum of two cubes.

The number 2 is represented as $1^3 + 1^3$.

The number 4 is not representable as the sum of two cubes.

The number 34 is not representable as the sum of two cubes.

The number 35 is represented as $2^3 + 3^3$.

The number 16 is represented as $2^3 + 2^3$.

The number 703657519796 is represented as $5779^3 + 7993^3$.

D. Permutation Transformation

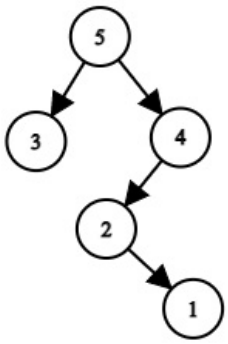
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A permutation — is a sequence of length n integers from 1 to n, in which all the numbers occur exactly once. For example, $[1]$, $[3, 5, 2, 1, 4]$, $[1, 3, 2]$ — permutations, and $[2, 3, 2]$, $[4, 3, 1]$, $[0]$ — no.

Polycarp was recently gifted a permutation $a[1 \dots n]$ of length n. Polycarp likes trees more than permutations, so he wants to transform permutation a into a rooted binary tree. He transforms an array of different integers into a tree as follows:

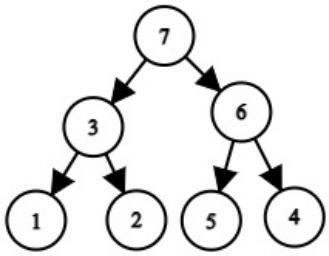
- the maximum element of the array becomes the root of the tree;
- all elements to the left of the maximum — form a left subtree (which is built according to the same rules but applied to the left part of the array), but if there are no elements to the left of the maximum, then the root has no left child;
- all elements to the right of the maximum — form a right subtree (which is built according to the same rules but applied to the right side of the array), but if there are no elements to the right of the maximum, then the root has no right child.

For example, if he builds a tree by permutation $a = [3, 5, 2, 1, 4]$, then the root will be the element $a_2 = 5$, and the left subtree will be the tree that will be built for the subarray $a[1 \dots 1] = [3]$, and the right one — for the subarray $a[3 \dots 5] = [2, 1, 4]$. As a result, the following tree will be built:



The tree corresponding to the permutation $a = [3, 5, 2, 1, 4]$.

Another example: let the permutation be $a = [1, 3, 2, 7, 5, 6, 4]$. In this case, the tree looks like this:



The tree corresponding to the permutation $a = [1, 3, 2, 7, 5, 6, 4]$.

Let us denote by d_v the depth of the vertex a_v, that is, the number of edges on the path from the root to the vertex numbered a_v. Note that the root depth is zero. Given the permutation a, for each vertex, find the value of d_v.

Input

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

The first line of each test case contains an integer n ($1 \le n \le 100$) — the length of the permutation.

This is followed by n numbers $a_1, a_2, \ldots, a_n$ — permutation a.

Output

For each test case, output n values — $d_1, d_2, \ldots, d_n$.

Example

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

E. Accidental Victory

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

A championship is held in Berland, in which n players participate. The player with the number i has a_i ($a_i \ge 1$) tokens.

The championship consists of $n - 1$ games, which are played according to the following rules:

- in each game, two random players with non-zero tokens are selected;
- the player with more tokens is considered the winner of the game (in case of a tie, the winner is chosen randomly);
- the winning player takes all of the loser's tokens;

The last player with non-zero tokens is the winner of the championship.

All random decisions that are made during the championship are made equally probable and independently.

For example, if $n = 4$, $a = [1, 2, 4, 3]$, then one of the options for the game (there could be other options) is:

- during the first game, the first and fourth players were selected. The fourth player has more tokens, so he takes the first player's tokens. Now $a = [0, 2, 4, 4]$;

- during the second game, the fourth and third players were selected. They have the same number of tokens, but in a random way, the third player is the winner. Now $a = [0, 2, 8, 0]$;
- during the third game, the second and third players were selected. The third player has more tokens, so he takes the second player's tokens. Now $a = [0, 0, 10, 0]$;
- the third player is declared the winner of the championship.

Championship winners will receive personalized prizes. Therefore, the judges want to know in advance which players have a chance of winning, i.e have a non-zero probability of winning the championship. You have been asked to find all such players.

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

The first line of each test case consists of one positive integer n ($1 \le n \le 2 \cdot 10^5$) — the number of players in the championship.

The second line of each test case contains n positive integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the number of tokens the players have.

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

Output
For each test case, print the number of players who have a nonzero probability of winning the championship. On the next line print the numbers of these players in **increasing** order. Players are numbered starting from one in the order in which they appear in the input.

Example

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

F. Equalize the Array

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

Polycarp was gifted an array a of length n. Polycarp considers an array beautiful if there exists a number C, such that each number in the array occurs either zero or C times. Polycarp wants to remove some elements from the array a to make it beautiful.

For example, if $n = 6$ and $a = [1, 3, 2, 1, 4, 2]$, then the following options are possible to make the array a array beautiful:

- Polycarp removes elements at positions 2 and 5, array a becomes equal to $[1, 2, 1, 2]$;
- Polycarp removes elements at positions 1 and 6, array a becomes equal to $[3, 2, 1, 4]$;
- Polycarp removes elements at positions $1, 2$ and 6, array a becomes equal to $[2, 1, 4]$;

Help Polycarp determine the minimum number of elements to remove from the array a to make it beautiful.

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

The first line of each test case consists of one integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — array a.

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

Output
For each test case, output one integer — the minimum number of elements that Polycarp has to remove from the array a to make it beautiful.

Example

input
3 6 1 3 2 1 4 2 4 100 100 4 100 8

1 2 3 3 3 2 6 6

output
2 1 2

G. Old Floppy Drive

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

Polycarp was dismantling his attic and found an old floppy drive on it. A round disc was inserted into the drive with n integers written on it.

Polycarp wrote the numbers from the disk into the a array. It turned out that the drive works according to the following algorithm:

- the drive takes one positive number x as input and puts a pointer to the first element of the a array;
- after that, the drive starts rotating the disk, every second moving the pointer to the next element, counting the sum of all the elements that have been under the pointer. Since the disk is round, in the a array, the last element is again followed by the first one;
- as soon as the sum is at least x, the drive will shut down.

Polycarp wants to learn more about the operation of the drive, but he has absolutely no free time. So he asked you m questions. To answer the i-th of them, you need to find how many seconds the drive will work if you give it x_i as input. Please note that in some cases the drive can work infinitely.

For example, if $n = 3, m = 3$, $a = [1, -3, 4]$ and $x = [1, 5, 2]$, then the answers to the questions are as follows:

- the answer to the first query is 0 because the drive initially points to the first item and the initial sum is 1.
- the answer to the second query is 6, the drive will spin the disk completely twice and the amount becomes $1 + (-3) + 4 + 1 + (-3) + 4 + 1 = 5$.
- the answer to the third query is 2, the amount is $1 + (-3) + 4 = 2$.

Input

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

The first line of each test case consists of two positive integers n, m ($1 \le n, m \le 2 \cdot 10^5$) — the number of numbers on the disk and the number of asked questions.

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

The third line of each test case contains m positive integers $x_1, x_2, \ldots, x_m$ ($1 \le x \le 10^9$).

It is guaranteed that the sums of n and m over all test cases do not exceed $2 \cdot 10^5$.

Output

Print m numbers on a separate line for each test case. The i-th number is:

- -1 if the drive will run infinitely;
- the number of seconds the drive will run, otherwise.

Example

input
3 3 3 1 -3 4 1 5 2 2 2 -2 0 1 2 2 2 0 1 1 2

output
0 6 2 -1 -1 1 3