Codeforces Round 1042 (Div. 3)

A. Lever

2 seconds, 256 megabytes

In Divergent Universe, The Lever iterates itself given two arrays a and b of length n. In each iteration, The Lever will do the following:

1. Choose a random index i such that $a_i > b_i$. Then, decrease a_i by 1. If there does not exist such i, ignore this step.
2. Choose a random index i such that $a_i < b_i$. Then, increase a_i by 1. If there does not exist such i, ignore this step.

After each iteration, the Lever will check if step 1 is ignored, and if so, it will end its iteration.

You're given the two arrays. Find the number of iterations that the Lever does. It can be shown this number is fixed over all possibilities of random indices that The Lever can choose for each step.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains one integer n ($1 \le n \le 10$).

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

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10$).

Output

For each test case, output one integer — the number of iterations that the Lever does.

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

output
3 1 10 7

In the first sample case:

In the first iteration, the Lever decreases a_1 by 1 and increases a_2 by 1, and a becomes $[6, 4]$.

In the second iteration, the Lever decreases a_1 by 1 and increases a_2 by 1, and a becomes $[5, 5]$.

In the third iteration, the Lever increases a_2 by 1, and a becomes $[5, 6]$. Since it fails to decrease an element, its iteration ends. Therefore, the answer is 3.

In the second sample case, the Lever does nothing in its first iteration, and thus it does only one iteration.

B. Alternating Series

2 seconds, 256 megabytes

You are given an integer n. Call an array a of length n **good** if:

- For all $1 \le i < n$, $a_i \cdot a_{i+1} < 0$ (i.e., the product of adjacent elements is negative).
- For all subarrays* with length at least 2, the sum of all elements in the subarray is positive†.

Additionally, we say a good array a of length n is **better** than another good array b of length n if $[|a_1|, |a_2|, \ldots, |a_n|]$ is lexicographically smaller‡ than $[|b_1|, |b_2|, \ldots, |b_n|]$. Note that $|z|$ denotes the absolute value of integer z.

Output a good array of length n such that it is better than every other good array of length n.

* An array c is a subarray of an array d if c can be obtained from d by the deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

† An integer x is positive if $x > 0$.

‡ A sequence a is lexicographically smaller than a sequence b if and only if one of the following holds:

- a is a prefix of b, but $a \ne b$; or
- in the first position where a and b differ, the sequence a has a smaller element than the corresponding element in b.

Input

The first line contains an integer t ($1 \le t \le 500$) — the number of test cases.

The single line of each test case contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the length of your array.

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 n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$), the elements of your array on a new line.

input
2 2 3

output
-1 2 -1 3 -1

In the first test case, because $a_1 \cdot a_2 = -2 < 0$ and $a_1 + a_2 = 1 > 0$, it satisfies the two constraints. In addition, it can be shown that the corresponding $b = [1, 2]$ is **better** than any other **good** array of length 2.

C. Make it Equal

2 seconds, 256 megabytes

Given two multisets S and T of size n and a positive integer k, you may perform the following operations any number (including zero) of times on S:

- Select an element x in S, and remove one occurrence of x in S. Then, either insert $x + k$ into S, or insert $|x - k|$ into S.

Determine if it is possible to make S equal to T. Two multisets S and T are equal if every element appears the same number of times in S and T.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains two integers n and k ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le 10^9$) — the size of S and the constant, respectively.

The second line contains n integers $S_1, S_2, \ldots, S_n$ ($0 \le S_i \le 10^9$) — the elements in S.

The third line contains n integers $T_1, T_2, \ldots, T_n$ ($0 \le T_i \le 10^9$) — the elements in T.

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 "YES" if it is possible to make S equal to T, and "NO" otherwise.

You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

input
5 1 3 1 2 1 8 4 12 3 5 6 2 9 8 4 11 2 7 2 8 2 9 3 2 0 1 0 1 0 1

output
YES YES YES NO NO

In the first test case, we can remove one occurrence of 1 from S and insert $|1 - k| = |1 - 3| = 2$ into S, making S equal to T.

In the second test case, we can remove one occurrence of 4 from S and insert $4 + k = 4 + 8 = 12$ into S, making S equal to T.

In the last test case, we can show that it is impossible to make S equal to T.

D. Arboris Contractio

2 seconds, 256 megabytes

Kagari is preparing to archive a tree, and she knows the cost of doing so will depend on its diameter*. To keep the expense down, her goal is to shrink the diameter as much as possible first. She can perform the following operation on the tree:

1. Choose two vertices s and t. Let the sequence of vertices on the simple path† from s to t be $v_0, v_1, \ldots, v_k$, where $v_0 = s$ and $v_k = t$.
2. Remove all edges along the path. In other words, remove edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$.
3. Connect vertices $v_1, v_2, \ldots, v_k$ directly to v_0. In other words, add edges $(v_0, v_1), (v_0, v_2), \ldots, (v_0, v_k)$.

It can be shown that the graph is still a tree after the operation.

Help her determine the minimum number of operations required to achieve the minimal diameter.

* The diameter of a tree is the longest possible distance between any pair of vertices. The distance itself is measured by the number of edges on the unique simple path connecting them.

† A simple path is a path between two vertices in a tree that does not visit any vertex more than once. It can be shown that the simple path between any two vertices is always unique.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the number of the vertices in the tree.

The following $n - 1$ lines of each test case describe the tree. Each of the lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$) that indicate an edge between vertex u and v. It is guaranteed that these edges form a tree.

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

Output

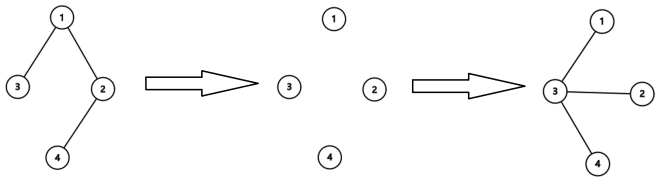
For each test case, output one integer — the minimum number of operations to minimize the diameter.

input
4 4 1 2 1 3 2 4 2 2 1 4 1 2 2 3 2 4 11 1 2 1 3 2 4 3 5 3 8 5 6 5 7 7 9 7 10 5 11

output
1 0 0 4

In the first test case, the diameter of the original tree is 3. Kagari can perform an operation on $s = 3$ and $t = 4$. As the figure depicts, the operations includes the following steps:

1. Remove edges $(3, 1)$, $(1, 2)$ and $(2, 4)$.
2. Add edges $(3, 1)$, $(3, 2)$ and $(3, 4)$.



After the operation, the diameter reduces to 2. It can be shown that 2 is the minimum diameter.

In the second test case, the diameter of the tree is 1. It can be shown that 1 is already the minimum, so Kagari can perform no operation.

E. Adjacent XOR

2 seconds, 256 megabytes

You're given an array a of length n. For each index i such that $1 \le i < n$, you can perform the following operation **at most once**:

- Assign $a_i := a_i \oplus a_{i+1}$, where $\oplus$ denotes the bitwise XOR operation. .

You can choose indices and perform the operations in any sequential order.

Given another array b of length n, determine if it is possible to transform a to b.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains one integer n ($2 \le n \le 2 \cdot 10^5$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{30}$).

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i < 2^{30}$).

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 "YES" (quotes excluded) if a can be transformed to b; otherwise, output "NO". You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

input
7 5 1 2 3 4 5 3 2 7 1 5 3 0 0 1 1 0 1 3 0 0 1 0 0 0 4 0 0 1 2 1 3 3 2 6 1 1 4 5 1 4 0 5 4 5 5 4 3 0 1 2 2 3 2 2 10 10 11 10

output
YES NO NO NO YES NO NO

In the first test case, you can perform the operations in the following order:

1. Choose index $i = 3$ and assign $a_3 := a_3 \oplus a_4 = 7$, and a becomes $[1, 2, 7, 4, 5]$.
2. Choose index $i = 4$ and assign $a_4 := a_4 \oplus a_5 = 1$, and a becomes $[1, 2, 7, 1, 5]$.
3. Choose index $i = 1$ and assign $a_1 := a_1 \oplus a_2 = 3$, and a becomes $[3, 2, 7, 1, 5]$.

F. Unjust Binary Life

2 seconds, 256 megabytes

Yuri is given two binary strings a and b, both of which are of length n. The two strings dynamically define an $n \times n$ grid. Let (i, j) denote the cell in the i-th row and j-th column. The initial value of cell (i, j) has the value of $a_i \oplus b_j$, where $\oplus$ denotes the bitwise XOR operation. .

Yuri's journey always starts at cell $(1, 1)$. From a cell (i, j), she can only move down to $(i + 1, j)$ or right to $(i, j + 1)$. Her journey is possible if there exists a valid path such that all cells on the path, including $(1, 1)$, have a value of 0.

Before her departure, she can do the following operation for any number of times:

- Choose one index $1 \le i \le n$, and flip the value of either a_i or b_i (0 becomes 1, and 1 becomes 0). The grid will also change accordingly.

Let $f(x, y)$ denote the minimum required operations so that Yuri can make her journey to the cell (x, y). You must determine the **sum** of $f(x, y)$ over all $1 \le x, y \le n$.

Note that each of these n^2 cases is independent, meaning you need to assume the grid is in its original state in each case (i.e., no actual operations are performed).

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains one integer n ($1 \le n \le 2 \cdot 10^5$).

The second line of each test case contains a binary string a ($|a| = n$, $a_i \in \{0, 1\}$).

The third line of each test case contains a binary string b ($|b| = n$, $b_i \in \{0, 1\}$).

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 sum of minimum operations over all possible cells.

input
3 2 11 00 2 01 01 4 1010 1101

output
5 4 24

In the first test case, the 2×2 grid is shown below.

$$\begin{matrix} 11 \\ 11 \end{matrix}$$

In the initial state, Yuri cannot reach any cell.

Yuri can flip a_1 so that the grid becomes:

$$\begin{matrix} 00 \\ 11 \end{matrix}$$

and Yuri can travel to cells $(1, 1)$ and $(1, 2)$.

On the other hand, Yuri can flip b_1 so that the grid becomes:

$$\begin{matrix} 01 \\ 01 \end{matrix}$$

and Yuri can travel to cells $(1, 1)$ and $(2, 1)$.

To move to the cell $(2, 2)$, it can be shown that she must perform at least two operations. For example, she can flip both a_1 and a_2 so that the grid becomes:

$$\begin{matrix} 00 \\ 00 \end{matrix}$$

Therefore, the answer is $1 + 1 + 1 + 2 = 5$.

G. Wafu!

3 seconds, 256 megabytes

To help improve her math, Kudryavka is given a set S that consists of n distinct positive integers.

Initially, her *score* is 1. She can perform an arbitrary number of the following operations on the set if it is not empty:

1. Let the minimum value of S be m.
2. Multiply her *score* by m.
3. Remove m from S.
4. For every integer i such that $1 \le i < m$, add i to the set S. It can be shown that no duplicates are added during this step.

She is addicted to performing operations, but after k operations, she realizes she forgot her *score*. Please help her determine her *score*, modulo $10^9 + 7$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

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 second line of each test case contains n integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le 10^9$, $s_i \ne s_j$) — the elements of the initial set S. It is guaranteed that the set S is not empty before each of the k operations is performed.

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 an integer indicating the answer modulo $10^9 + 7$.

input
4 2 3 1 3 3 6 5 1 4 2 100 2 100 5 15 1 2 3 4 5

output
3 24 118143737 576

Let us simulate the process in the first test case:

$$\{1, 3\} \xrightarrow{\text{remove } 1} \{3\} \xrightarrow[\text{add } 1,2]{\text{remove } 3} \{1, 2\} \xrightarrow{\text{remove } 1} \{2\}$$

The removed values are 1, 3 and 1 respectively, so her *score* is $1 \times 3 \times 1 = 3$.

In the second test case, the answer is $1 \times 4 \times 1 \times 2 \times 1 \times 3 = 24$.

H. Sea, You & copriMe

3 seconds, 256 megabytes

Umi is given an array a of length n, whose elements are integers between 1 and m. She loves coprime integers and wants to find four **distinct** indices p, q, r, s ($1 \le p, q, r, s \le n$), such that $\gcd(a_p, a_q) = 1$ and $\gcd(a_r, a_s) = 1$*.

If there are multiple solutions, you may output any one of them.

*$\gcd(x, y)$ denotes the greatest common divisor (GCD) of integers x and y.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($4 \le n \le 2 \cdot 10^5, 1 \le m \le 10^6$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le m$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$, and that the sum of m over all test cases does not exceed 10^6.

Output

For each test case:

- If no such set of four distinct indices exists, output one integer 0.
- Otherwise, output four **distinct** integers p, q, r, s ($1 \le p, q, r, s \le n$) that satisfy the condition. If there are multiple solutions, output any one of them.

input
5 4 15 4 7 9 15 4 10 1 2 4 8 5 15 6 10 11 12 15 5 15 6 10 11 14 15 6 10000 30 238 627 1001 1495 7429

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

In the first test case, $\gcd(a_1, a_3) = \gcd(4, 9) = 1$, $\gcd(a_2, a_4) = \gcd(7, 15) = 1$.

In the second test case, it can be shown that no such quadruple exists.