Codeforces Round #658 (Div. 1)

A1. Prefix Flip (Easy Version)

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

This is the easy version of the problem. The difference between the versions is the constraint on n and the required number of operations. You can make hacks only if all versions of the problem are solved.

There are two binary strings a and b of length n (a binary string is a string consisting of symbols 0 and 1). In an operation, you select a prefix of a, and simultaneously invert the bits in the prefix (0 changes to 1 and 1 changes to 0) and reverse the order of the bits in the prefix.

For example, if $a = 001011$ and you select the prefix of length 3, it becomes 011011. Then if you select the entire string, it becomes 001001.

Your task is to transform the string a into b in at most $3n$ operations. It can be proved that it is always possible.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases. Next $3t$ lines contain descriptions of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 1000$) — the length of the binary strings.

The next two lines contain two binary strings a and b of length n.

It is guaranteed that the sum of n across all test cases does not exceed 1000.

Output

For each test case, output an integer k ($0 \le k \le 3n$), followed by k integers $p_1, \ldots, p_k$ ($1 \le p_i \le n$). Here k is the number of operations you use and p_i is the length of the prefix you flip in the i-th operation.

Example

input
5 2 01 10 5 01011 11100 2 01 01 10 0110011011 1000110100 1 0 1
output
3 1 2 1 6 5 2 5 3 1 2 0 9 4 1 2 10 4 1 2 1 5 1 1

Note

In the first test case, we have $01 \to 11 \to 00 \to 10$.

In the second test case, we have $01011 \to 00101 \to 11101 \to 01000 \to 10100 \to 00100 \to 11100$.

In the third test case, the strings are already the same. Another solution is to flip the prefix of length 2, which will leave a unchanged.

A2. Prefix Flip (Hard Version)

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

This is the hard version of the problem. The difference between the versions is the constraint on n and the required number of operations. You can make hacks only if all versions of the problem are solved.

There are two binary strings a and b of length n (a binary string is a string consisting of symbols 0 and 1). In an operation, you select a prefix of a, and simultaneously invert the bits in the prefix (0 changes to 1 and 1 changes to 0) and reverse the order of the bits in the prefix.

For example, if $a = 001011$ and you select the prefix of length 3, it becomes 011011. Then if you select the entire string, it becomes 001001.

Your task is to transform the string a into b in at most $2n$ operations. It can be proved that it is always possible.

Input
The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases. Next $3t$ lines contain descriptions of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the length of the binary strings.

The next two lines contain two binary strings a and b of length n.

It is guaranteed that the sum of n across all test cases does not exceed 10^5.

Output
For each test case, output an integer k ($0 \le k \le 2n$), followed by k integers $p_1, \ldots, p_k$ ($1 \le p_i \le n$). Here k is the number of operations you use and p_i is the length of the prefix you flip in the i-th operation.

Example

input
5 2 01 10 5 01011 11100 2 01 01 10 0110011011 1000110100 1 0 1
output
3 1 2 1 6 5 2 5 3 1 2 0 9 4 1 2 10 4 1 2 1 5 1 1

Note
In the first test case, we have $01 \to 11 \to 00 \to 10$.

In the second test case, we have $01011 \to 00101 \to 11101 \to 01000 \to 10100 \to 00100 \to 11100$.

In the third test case, the strings are already the same. Another solution is to flip the prefix of length 2, which will leave a unchanged.

B. Unmerge

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

Let a and b be two arrays of lengths n and m, respectively, with no elements in common. We can define a new array $\text{merge}(a, b)$ of length $n + m$ recursively as follows:

- If one of the arrays is empty, the result is the other array. That is, $\text{merge}(\emptyset, b) = b$ and $\text{merge}(a, \emptyset) = a$. In particular, $\text{merge}(\emptyset, \emptyset) = \emptyset$.
- If both arrays are non-empty, and $a_1 < b_1$, then $\text{merge}(a, b) = [a_1] + \text{merge}([a_2, \ldots, a_n], b)$. That is, we delete the first element a_1 of a, merge the remaining arrays, then add a_1 to the beginning of the result.
- If both arrays are non-empty, and $a_1 > b_1$, then $\text{merge}(a, b) = [b_1] + \text{merge}(a, [b_2, \ldots, b_m])$. That is, we delete the first element b_1 of b, merge the remaining arrays, then add b_1 to the beginning of the result.

This algorithm has the nice property that if a and b are sorted, then $\text{merge}(a, b)$ will also be sorted. For example, it is used as a subroutine in merge-sort. For this problem, however, we will consider the same procedure acting on non-sorted arrays as well. For example, if $a = [3, 1]$ and $b = [2, 4]$, then $\text{merge}(a, b) = [2, 3, 1, 4]$.

A permutation is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2,3,1,5,4]$ is a permutation, but $[1,2,2]$ is not a permutation (2 appears twice in the array) and $[1,3,4]$ is also not a permutation ($n=3$ but there is 4 in the array).

There is a permutation p of length $2n$. Determine if there exist two arrays a and b, each of length n and with no elements in common, so that $p=\text{merge}(a,b)$.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases. Next $2t$ lines contain descriptions of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 2000$).

The second line of each test case contains $2n$ integers $p_1, \ldots, p_{2n}$ ($1 \le p_i \le 2n$). It is guaranteed that p is a permutation.

It is guaranteed that the sum of n across all test cases does not exceed 2000.

Output

For each test case, output "YES" if there exist arrays a, b, each of length n and with no common elements, so that $p=\text{merge}(a,b)$. Otherwise, output "NO".

Example

input
6 2 2 3 1 4 2 3 1 2 4 4 3 2 6 1 5 7 8 4 3 1 2 3 4 5 6 4 6 1 3 7 4 5 8 2 6 4 3 2 5 1 11 9 12 8 6 10 7
output
YES NO YES YES NO NO

Note

In the first test case, $[2,3,1,4]=\text{merge}([3,1],[2,4])$.

In the second test case, we can show that $[3,1,2,4]$ is not the merge of two arrays of length 2.

In the third test case, $[3,2,6,1,5,7,8,4]=\text{merge}([3,2,8,4],[6,1,5,7])$.

In the fourth test case, $[1,2,3,4,5,6]=\text{merge}([1,3,6],[2,4,5])$, for example.

C. Mastermind

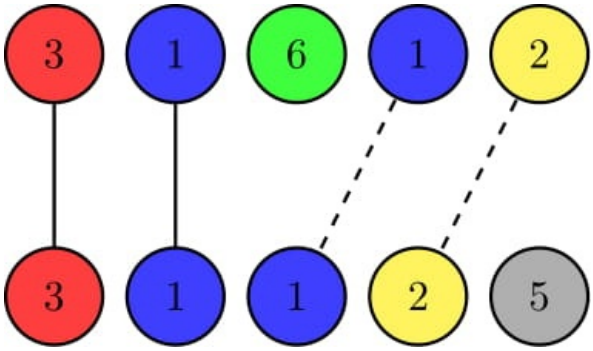
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

In the game of Mastermind, there are two players — Alice and Bob. Alice has a secret code, which Bob tries to guess. Here, a code is defined as a sequence of n colors. There are exactly $n+1$ colors in the entire universe, numbered from 1 to $n+1$ inclusive.

When Bob guesses a code, Alice tells him some information about how good of a guess it is, in the form of two integers x and y.

The first integer x is the number of indices where Bob's guess correctly matches Alice's code. The second integer y is the size of the intersection of the two codes as multisets. That is, if Bob were to change the order of the colors in his guess, y is the maximum number of indices he could get correct.

For example, suppose $n=5$, Alice's code is $[3,1,6,1,2]$, and Bob's guess is $[3,1,1,2,5]$. At indices 1 and 2 colors are equal, while in the other indices they are not equal. So $x=2$. And the two codes have the four colors $1,1,2,3$ in common, so $y=4$.



Solid lines denote a matched color for the same index. Dashed lines denote a matched color at a different index. x is the number of solid lines, and y is the total number of lines.

You are given Bob's guess and two values x and y. Can you find one possibility of Alice's code so that the values of x and y are correct?

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases. Next $2t$ lines contain descriptions of test cases.

The first line of each test case contains three integers n, x, y ($1 \le n \le 10^5, 0 \le x \le y \le n$) — the length of the codes, and two values Alice responds with.

The second line of each test case contains n integers $b_1, \ldots, b_n$ ($1 \le b_i \le n + 1$) — Bob's guess, where b_i is the i-th color of the guess.

It is guaranteed that the sum of n across all test cases does not exceed 10^5.

Output

For each test case, on the first line, output "YES" if there is a solution, or "NO" if there is no possible secret code consistent with the described situation. You can print each character in any case (upper or lower).

If the answer is "YES", on the next line output n integers $a_1, \ldots, a_n$ ($1 \le a_i \le n + 1$) — Alice's secret code, where a_i is the i-th color of the code.

If there are multiple solutions, output any.

Example

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

Note

The first test case is described in the statement.

In the second test case, $x = 3$ because the colors are equal at indices $2, 4, 5$. And $y = 4$ because they share the colors $1, 1, 1, 2$.

In the third test case, $x = 0$ because there is no index where the colors are the same. But $y = 4$ because they share the colors $3, 3, 5, 5$.

In the fourth test case, it can be proved that no solution exists.

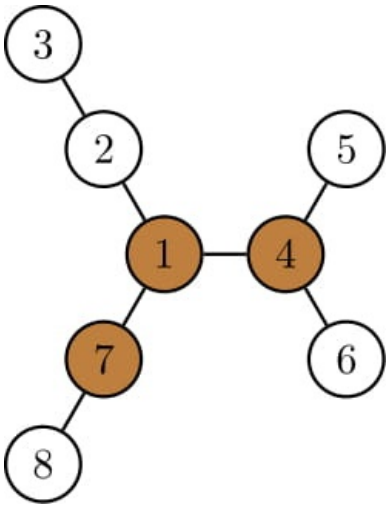
D. The Majestic Brown Tree Snake

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

There is an undirected tree of n vertices, connected by $n-1$ bidirectional edges. There is also a snake stuck inside of this tree. Its head is at vertex a and its tail is at vertex b. The snake's body occupies all vertices on the unique simple path between a and b.

The snake wants to know if it can reverse itself — that is, to move its head to where its tail started, and its tail to where its head started. Unfortunately, the snake's movements are restricted to the tree's structure.

In an operation, the snake can move its head to an adjacent vertex not currently occupied by the snake. When it does this, the tail moves one vertex closer to the head, so that the length of the snake remains unchanged. Similarly, the snake can also move its tail to an adjacent vertex not currently occupied by the snake. When it does this, the head moves one unit closer to the tail.



Let's denote a snake position by (h,t), where h is the index of the vertex with the snake's head, t is the index of the vertex with the snake's tail. This snake can reverse itself with the movements $(4,7) \to (5,1) \to (4,2) \to (1,3) \to (7,2) \to (8,1) \to (7,4)$. Determine if it is possible to reverse the snake with some sequence of operations.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases. The next lines contain descriptions of test cases.

The first line of each test case contains three integers n, a, b ($2 \le n \le 10^5, 1 \le a, b \le n, a \ne b$).

Each of the next $n-1$ lines contains two integers u_i, v_i ($1 \le u_i, v_i \le n, u_i \ne v_i$), indicating an edge between vertices u_i and v_i. It is guaranteed that the given edges form a tree.

It is guaranteed that the sum of n across all test cases does not exceed 10^5.

Output

For each test case, output "YES" if it is possible for the snake to reverse itself, or "NO" otherwise.

Example

input
4 8 4 7 1 2 2 3 1 4 4 5 4 6 1 7 7 8 4 3 2 4 3 1 2 2 3 9 3 5 1 2 2 3 3 4 1 5 5 6 6 7 1 8 8 9 16 15 12 1 2 2 3 1 4 4 5 5 6 6 7 4 8

```
8 9
8 10
10 11
11 12
11 13
13 14
10 15
15 16
```

output

```
YES
NO
NO
YES
```

Note

The first test case is pictured above.

In the second test case, the tree is a path. We can show that the snake cannot reverse itself.

In the third test case, we can show that the snake cannot reverse itself.

In the fourth test case, an example solution is:

$(15, 12) \to (16, 11) \to (15, 13) \to (10, 14) \to (8, 13) \to (4, 11) \to (1, 10)$

$\to (2, 8) \to (3, 4) \to (2, 5) \to (1, 6) \to (4, 7) \to (8, 6) \to (10, 5)$

$\to (11, 4) \to (13, 8) \to (14, 10) \to (13, 15) \to (11, 16) \to (12, 15).$

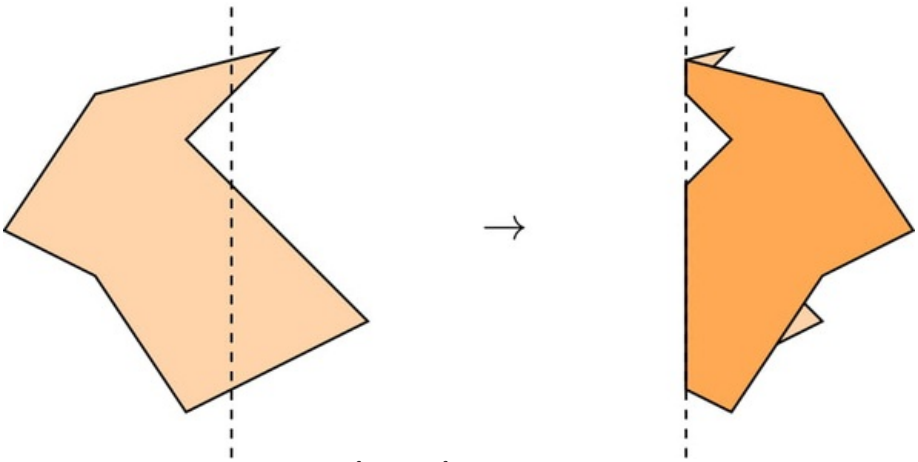
E. Origami

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

After being discouraged by 13 time-limit-exceeded verdicts on an ugly geometry problem, you decided to take a relaxing break for arts and crafts.

There is a piece of paper in the shape of a simple polygon with n vertices. The polygon may be non-convex, but we all know that proper origami paper has the property that **any horizontal line intersects the boundary of the polygon in at most two points.**

If you fold the paper along the vertical line $x = f$, what will be the area of the resulting shape? When you fold, the part of the paper to the left of the line is symmetrically reflected on the right side.



Your task is to answer q independent queries for values $f_1, \ldots, f_q$.

Input

The first line contains two integers n, q $(3 \le n \le 10^5, 1 \le q \le 10^5)$ — the number of polygon vertices and queries, respectively.

Each of the next n lines contains two integers x_i, y_i $(|x_i|, |y_i| \le 10^5)$ — the coordinates of the i-th point of the polygon. The polygon has an edge connecting each pair of adjacent points in the input, and also an edge between (x_1, y_1) and (x_n, y_n). It is guaranteed that the polygon is non-degenerate and that any horizontal line intersects the boundary of the polygon in at most two points. In particular, no boundary edge is strictly horizontal. Two adjacent sides may be collinear.

Each of the next q lines contains a single integer f_i $(\min\limits_{j=1}^{n}(x_j) < f_i < \max\limits_{j=1}^{n}(x_j))$ — the x-coordinate of the i-th fold query. It is guaranteed that all f_i are distinct.

Output

For each query, output the area A_i of the paper if you fold it along the line $x = f_i$.

Your answer is considered correct if its absolute or relative error does not exceed 10^{-4}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \leq 10^{-4}$.

Examples

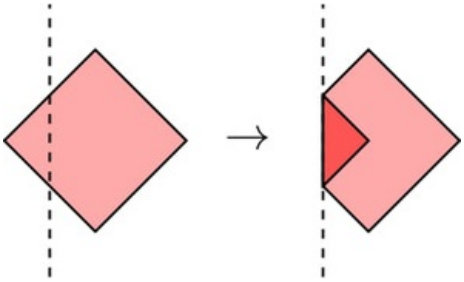
input
4 7 0 10 10 0 0 -10 -10 0 -9 -5 -1 0 1 5 9
output
199.0000000000 175.0000000000 119.0000000000 100.0000000000 119.0000000000 175.0000000000 199.0000000000

input
4 1 0 0 0 2 2 4 2 2 1
output
3.0000000000

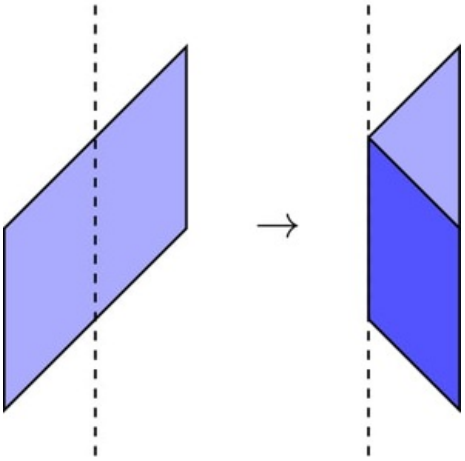
input
9 4 0 -3 2 -2 -1 -1 0 4 2 5 1 6 -2 3 -1 1 -3 0 0 -2 -1 1
output
11.1250000000 11.7500000000 10.3446969697 11.3333333333

Note

The first test, with the fold $f = -5$:



The second test, with the fold $f = 1$:



The third test, with the fold $f = -1$: