Problem A. King of String Comparison

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given two strings s, t of length n. Find the number of pairs (l, r) of integers such that $1 \le l \le r \le n$ and the substring $s_l s_{l+1} s_{l+2} \dots s_r$ is **lexicographically smaller** than $t_l t_{l+1} t_{l+2} \dots t_r$.

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

- a is a prefix of b, but $a \ne b$;
- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

Input

The first line of the input contains a single integer n $(1 \le n \le 200\,000)$.

The second and the third lines of the input contain strings s, t of length n correspondingly, each consisting only of lowercase Latin letters.

Output

Output a single integer — the number of pairs (l, r) of integers such that $1 \le l \le r \le n$ and the substring $s_l s_{l+1} s_{l+2} \dots s_r$ is **lexicographically smaller** than $t_l t_{l+1} t_{l+2} \dots t_r$.

Examples

standard input	standard output
3 aab aba	4
4 icpc cool	4
4 zyzz life	0
7 trivial problem	16
18 goodluckandhavefun letthestrongestwin	112

Note

In the first sample, there are 4 such pairs: $(1, 2), (1, 3), (2, 2), (2, 3)$.

Problem B. New Queries On Segment Deluxe

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

You know those problems where you are given an array of length roughly 10^5 and you have to process roughly 10^5 queries about something on a segment? Yes, this is one of those problems. And it should be persistent, because why not.

Consider $k \times n$ matrix A (with k rows and n columns). For a given matrix we can construct the array B as follows: $B_j = \sum_{i=1}^{k} A_{ij}$.

There will be up to $q + 1$ versions of the matrix. The j-th element in i-th row of t-th version of A is denoted as $A_{ij}^{(t)}$. The j-th element of the array B corresponding to t-th version of A is denoted as $B_j^{(t)}$.

You are given the 0-th version of the matrix A. You have to process q queries of 3 types:

- `1 t p l r x` : add x to $A_{pi}^{(t)}$ for $l \le i \le r$, thus creating a new version of the matrix
- `2 t p l r y` : set $A_{pi}^{(t)}$ to be equal to y for $l \le i \le r$, thus creating a new version of the matrix
- `3 t l r` : print $\min_{i=l}^{r} B_i^{(t)}$

The version of the matrix A created after the i-th query will be called the i-th version. Thus version numbers can be from 0 to q inclusive, but some of the integers from 0 to q may not have the correspondent version.

Input

The first line of input contains 3 integers k, n, q ($1 \le k \le 4$, $1 \le n \le 250\,000$, $1 \le q \le 20\,000$) — the dimensions of the matrix and the number of queries respectively.

The i-th of the next k lines contains n integers $A_{i1}^{(0)}, A_{i2}^{(0)}, \ldots, A_{in}^{(0)}$ ($|A_{ij}^{(0)}| \le 10^8$).

The next q lines describe the queries in the format explained earlier. It is guaranteed that t refers to a valid already existing version of the matrix, $1 \le p \le k$, $1 \le l \le r \le n$, $|x| \le 10^4$, $|y| \le 10^8$.

It is guaranteed that there exists at least one query of type 3.

Output

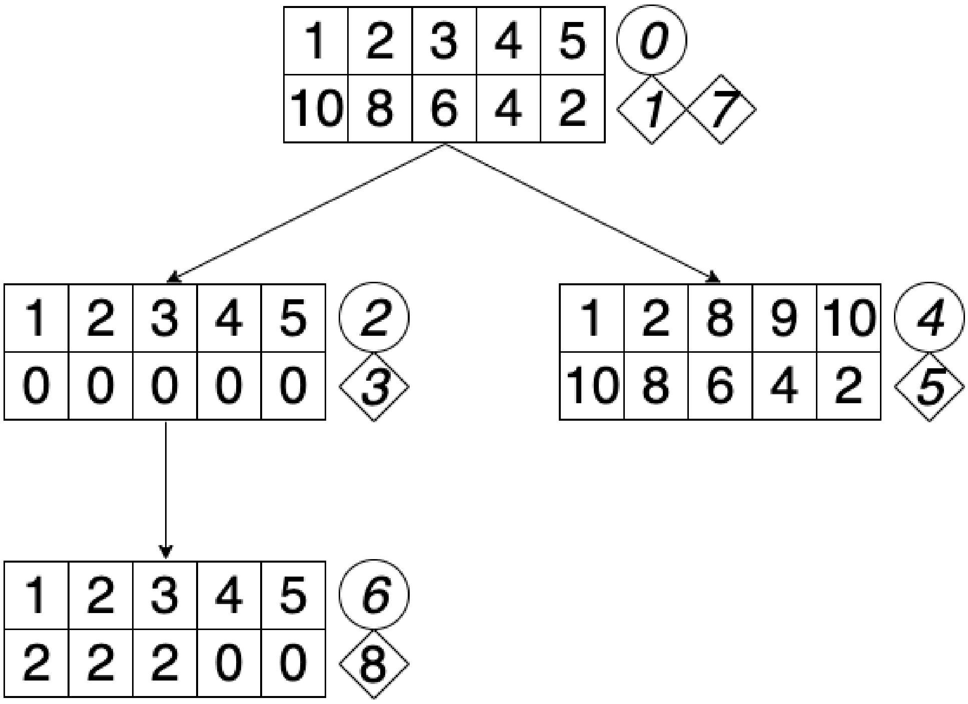
Print the answers to the queries of type 3 in the order in which the queries were given, on separate lines.

Example

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

Note

Here is how the versions of the matrix will look like:



The number in a circle is the version, the numbers in rhombuses are queries of type 3.

Problem C. Werewolves

Input file: `standard input`
Output file: `standard output`
Time limit: 2 seconds
Memory limit: 512 megabytes

You are given a colored tree on n nodes, node i has color c_i. As a reminder, a tree on n nodes is a connected graph with $n - 1$ edges.

Compute the number of connected subgraphs of this tree, for which there exists some color (majority color), such that **strictly more than half** of the nodes of this subgraph have this color.

Since the answer can be quite large, compute it modulo 998 244 353.

Input

The first line of input contains one integer n ($1 \le n \le 3000$) — the number of nodes in the tree.

The second line contains n integers $c_1\ c_2\ \dots\ c_n$ ($1 \le c_i \le n$) — the colors of the nodes.

The i-th of next $n-1$ lines contains 2 integers u_i, v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), representing the edge (u_i, v_i) of the tree. It is guaranteed that the given graph is a tree.

Output

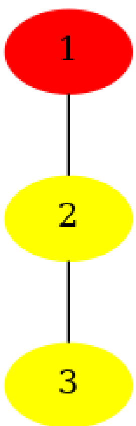
Print a single integer — answer to the problem modulo 998 244 353.

Examples

standard input	standard output
3 2 3 3 1 2 2 3	5
4 1 1 3 3 1 2 1 3 1 4	8

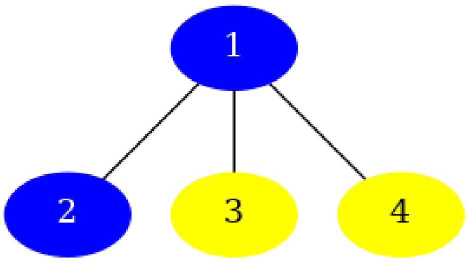
Note

In the following pictures, we use blue for color 1, red for color 2, and yellow for color 3. The first example looks as follows:



The tree has a total of 6 non-empty connected subgraphs: 3 of size 1, 2 of size 2 and 1 of size 3, the latter in fact being the whole tree. All such subgraphs of sizes 1 and 3 have a majority color. For those of size 2 only the subgraph induced by vertices 1 and 2 does not have a majority color (red and yellow both appear equally often in it). Therefore, there are $6 - 1 = 5$ connected subgraphs with a majority color.

The second example looks as follows, and it has 8 connected subgraphs with a majority color:



Problem D. Many LCS

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

Given K, construct two non-empty **binary** strings S and T of length **at most 8848** such that they have exactly K different longest common subsequences. More formally, if L is the length of the longest common subsequence of S and T, there should exist exactly K distinct binary strings of length L which are subsequences of both S and T.

It is guaranteed that under the constraints of this problem such strings always exist.

Input

The only line of input contains a single integer K ($1 \leq K \leq 10^9$).

Output

Print non-empty binary strings S and T on separate lines. The length of each of them should not exceed 8848. They can have different lengths.

If there is more than one solution, you can print any one of them.

Examples

standard input	standard output
`1`	`1111` `00`
`2`	`10` `01`
`3`	`010` `1001`
`100`	`1000100000101110000101010100011` `1100001000101010100101010011100`

Note

In the first example, the longest common subsequence of `1111` and `00` has length 0, and there exists only one string of length 0 which is a subsequence of both of them — the empty string.

In the second example, the length of the longest common subsequence of strings `10` and `01` is 1, and there are 2 strings of length 1 which are subsequences of both S and T: `0` and `1`.

In the second example, the length of the longest common subsequence of strings `010` and `1001` is 2, and there are 3 strings of length 2 which are subsequences of both S and T: `00`, `01`, and `10`.

It would be disrespectful to make strings longer than Everest...

Problem E. Replace Sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Consider an array A and a set B of integers such that **all numbers in A and B are distinct**. Your task is to turn A into a sorted array. To do this you can take any number from B and replace any element of A with it. You can perform this operation any number of times, but each element of B can be used at most once.

Determine the minimum number of operations needed to turn A into a sorted array, or determine that it is impossible.

Input

The first line of input contains two integers N and M $(1 \leq N, M \leq 5 \cdot 10^5)$ — the sizes of A and B respectively.

The second line contains N integers $A_1, A_2, \ldots, A_N$.

The third line contains M integers $B_1, B_2, \ldots, B_M$.

All the $(N + M)$ elements are distinct, positive and do not exceed 10^9.

Output

If it is impossible to turn A into a sorted array, print -1. Otherwise, print the minimum number of operations needed.

Examples

standard input	standard output

`4 1`
`2 6 13 10`
`5`
	`-1`

`4 2`
`2 6 13 10`
`5 4`
	`2`

`4 3`
`2 6 13 10`
`5 4 19`
	`1`

Note

In all three examples, the issue is that $13 > 10$, so we have to change at least one of them.

In the first one, we can decrease 13 by replacing it with 5, but it breaks the other side, so there is no solution.

In the second one, we also have 4, which we can use to fix the broken side. It is impossible to do with less than 2 operations.

In the third example we can finally increase the last element, thus fixing A in 1 operation.

Problem F. to Pay Respects

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are playing a game, and you are going to fight the secret boss. In this game, the boss doesn't attack you, but they can cast regeneration spells.

The fight consists of exactly N rounds, in each round the following actions can happen, in this order:

1. The boss can choose to cast the "Regeneration" spell.
2. You can choose to cast the "Poison" spell if you have any mana left.
3. You attack with a sword, dealing X damage.
4. All the passive effects are applied.

There are two types of passive effects: regeneration and poison. The effects stack, which means that the current state of the boss can be described with three integers: current health points (hp), current poison stacks (p) and current regeneration stacks (r). At the beginning of the fight, there are no poison stacks and no regeneration stacks ($p = r = 0$). Each poison stack deals P damage, each regeneration stack heals R health points.

Spells have the following effects:

"Regeneration": increase the number of regenerations stacks r by 1.

"Poison": increase the number of poison stacks p by 1. **If the number of regeneration stacks is strictly positive** ($r > 0$), then decrease it by 1.

After the round the hp will decrease by $X + P \cdot p - R \cdot r$ (this value can be negative if the boss heals faster than you deal damage).

For each round you know if the boss will cast "Regeneration". You have enough mana to cast "Poison" K times (you don't have to use all of your mana). What's the largest total damage you can deal to the boss, in other words, what is the maximum value of $hp_{start} - hp_{end}$? Assume that $hp_{start} = 10^{1000}$, so you can't actually kill the boss in N rounds. Boss hp can go higher than the initial value (see the third sample case).

Input

The first line of the input contains 5 integers N, X, R, P, K ($1 \le N, X, R, P \le 10^6$, $0 \le K \le N$).

The second line of the input contains a binary string of length N. The i-th character of this string is `1`, if the boss casts "Regeneration" at the beginning of the i-th round, and `0` otherwise.

Output

Output a single integer — the largest total damage you can deal during the fight.

Examples

standard input	standard output
`2 1010 1 1 1` `01`	`2021`
`3 2 1 1 1` `001`	`8`
`10 1 10 40 1` `1111111111`	`-40`

Note

Let's look at the first sample. We can cast the "Poison" spell at most once. Let's look at what will happen if we cast this spell during the first round.

- During the first round, we apply a "Poison" spell, so at the end of this round there will be 0 regeneration stacks, and 1 poison stack. Therefore, the *hp* will decrease by $X + P \cdot 1 - R \cdot 0 = 1011$ this round.
- At the beginning of the second round, the boss will cast the "Regeneration" spell, so there will be 1 regeneration stack and 1 poison stack at the end of the second round. So, the *hp* will decrease by $X + P \cdot 1 - R \cdot 1 = 1010$ this round. Overall, the health of the boss decreased by $1011 + 1010 = 2021$.

Now let's look at what will happen if we cast this spell during the second round.

- During the first round, no spells are applied, so at the end of this round there will be 0 regeneration stacks, and 0 poison stacks. Therefore, the *hp* will decrease by $X + P \cdot 0 - R \cdot 0 = 1010$ this round.
- At the beginning of the second round, the boss will cast the "Regeneration" spell, so that there will be one regeneration stack after that. Then, we will we apply a "Poison" spell, decreasing the number of regeneration stacks by one. So, there will be 0 regeneration stacks and 1 poison stack at the end of the second round. Therefore, the *hp* will decrease by $X + P \cdot 1 - R \cdot 0 = 1011$ this round. Overall, the health of the boss decreased by $1010 + 1011 = 2021$ again.

So, no matter when we cast the "Poison" spell in this sample, we will still decrease the *hp* by 2021.

Problem G. Max Pair Matching

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given $2n$ pairs (a_i, b_i) of integers. Consider a complete graph on $2n$ vertices and define the weight of the edge (ij) to be $w_{ij} = max(|a_i - a_j|, |a_i - b_j|, |b_i - a_j|, |b_i - b_j|)$.

Determine the maximum weight of the matching in this graph.

In other words, consider all ways to select n edges of this graph such that no two chosen edges have a common endpoint. What is the maximum possible total weight of these edges?

Input

The first line of the input contains a single integer n $(1 \le n \le 10^5)$.

The i-th of the next $2n$ lines contain two integers a_i and b_i $(0 \le a_i, b_i \le 10^9)$.

Output

Print a single integer — the maximum weight of the matching in this graph.

Example

standard input	standard output
2 0 10 7 7 9 4 2 15	18

Note

Adjacency matrix:
$$\begin{matrix} 0 & 7 & 9 & 15 \\ 7 & 0 & 3 & 8 \\ 9 & 3 & 0 & 11 \\ 15 & 8 & 11 & 0 \end{matrix}$$

Problem H. Colourful Permutation Sorting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

You are given a permutation $p_1, p_2, \ldots, p_n$ of integers from 1 to n. Each position from 1 to n is colored in one of k colors. We want to sort the permutation, and for that, we can apply any number of operations of the following types:

- Swap any two elements. This operation costs S coins;
- Choose any color i, and permute the elements on positions of color i as you wish. This operation costs C_i coins.

Note that the positions are colored, not the elements, so when you swap two elements, the positions won't change their colors.

Find the minimum number of coins you need to spend to sort the permutation.

Input

The first line of the input contains a single integer T $(1 \le T \le 10^3)$ — the number of independent test cases you need to process. The description of the test cases follows.

The first line of each test case contains two integers n and k $(1 \le n \le 10^5,\ 1 \le k \le 5)$ — the size of the permutation and the number of colors.

The second line of each test case contains $(k+1)$ integers $S, C_1, C_2, \ldots, C_k$ $(0 \le S, C_i \le 10^9)$ — the costs of the operations.

The third line of each test case contains n integers $p_1, p_2, \ldots, p_n$ $(1 \le p_i \le n$, all p_i are distinct$)$ — the permutation.

The fourth line of each test case contains n integers col_i $(1 \le col_i \le k)$ — the colors of the positions.

The sum of n over all test cases in one file does not exceed 10^5.

Output

For each test case print a single integer — the minimum number of coins you need to spend to sort the permutation.

Example

standard input	standard output
4 4 1 1 10 2 3 4 1 1 1 1 1 4 1 10 1 2 3 4 1 1 1 1 1 6 2 10 1 1 5 2 4 6 1 3 1 2 1 2 1 2 4 3 6 7 8 9 1 2 3 4 2 2 3 2	3 1 12 0

Note

In the first test case, we can sort the permutation by applying the "Swap" operation 3 times: $(2, 3, 4, 1) \to (4, 3, 2, 1) \to (4, 2, 3, 1) \to (1, 2, 3, 4)$. This way you will spend 3 coins.

Another way to sort it would be to permute all elements on positions of color 1, but this would cost 10 coins, and we can do cheaper.

In the second test case (which differs from the first one only in the costs of operations), however, it's cheaper to just permute all elements on positions of color 1, spending 1 coin on this.

In the third test case, one of the optimal sequences of operations would be the following:

- Permute the elements on positions of color 2 to obtain the permutation $(5, 2, 4, 3, 1, 6)$. This operation costs 1 coin.
- Swap elements p_3, p_4. The permutation is now $(5, 2, 3, 4, 1, 6)$. This operation costs 10 coins.
- Permute the elements on positions of color 1 to obtain the permutation $(1, 2, 3, 4, 5, 6)$. This operation costs 1 coin.

In total, we spent 12 coins.

In the fourth test case, the permutation is already sorted, so we don't have to spend anything.

Problem I. Flood Fill

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Given are two black and white $N \times M$ images A and B.

The "flood fill" tool works as follows: you choose any cell (x, y), locate its connected component and flip the colors of all the cells in the component (if the cell was black, it becomes white, and if it was white, it becomes black). The connected component of the cell is the set of cells you can reach by going up/down/left/right without changing color.

You can apply the "flood fill" tool to image A any number of times. What is the minimum number of cells in which A can be different from B after some sequence of operations?

Input

The first line of input contains two integers N and M $(1 \leq N, M \leq 100)$ — the dimensions of the images.

Each of the next N lines contains a binary string of length M, describing the corresponding row of the image A.

Each of the next N lines contains a binary string of length M, describing the corresponding row of the image B.

Here `0` corresponds to the cell colored white, `1` corresponds to the cell colored black.

Output

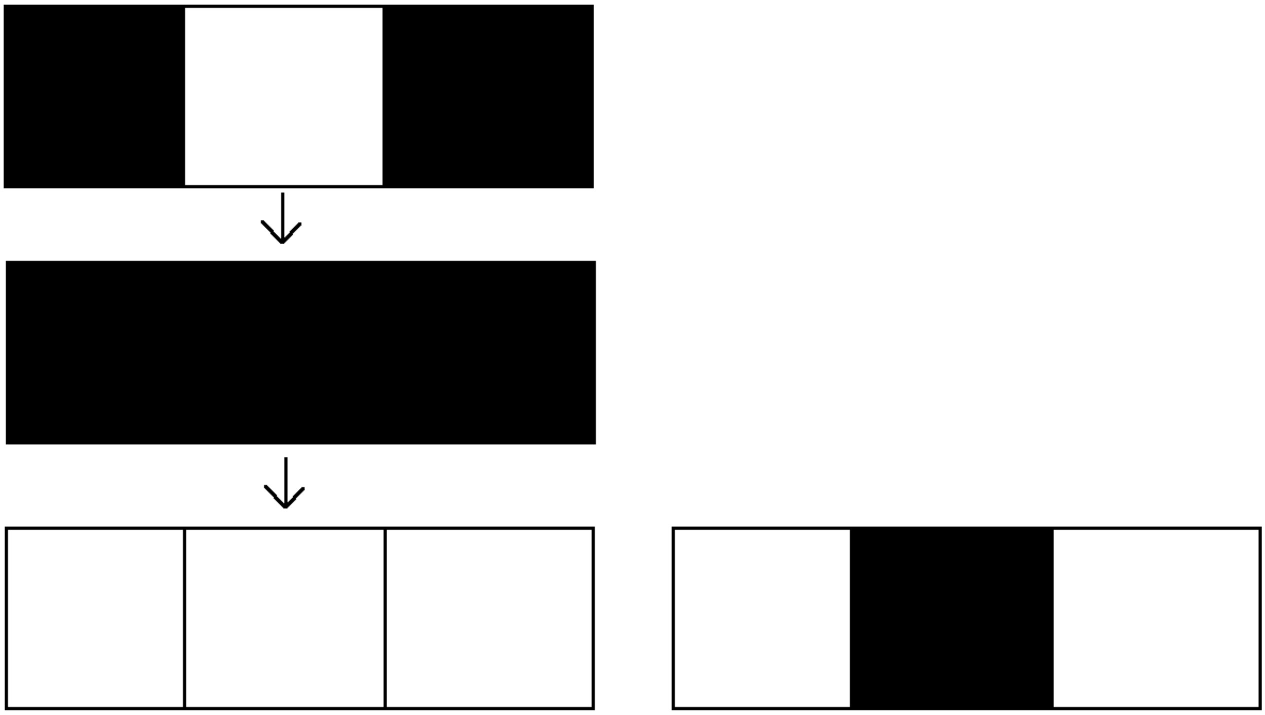
Output a single integer — the minimum possible number of cells in which A can be different from B after some sequence of operations.

Examples

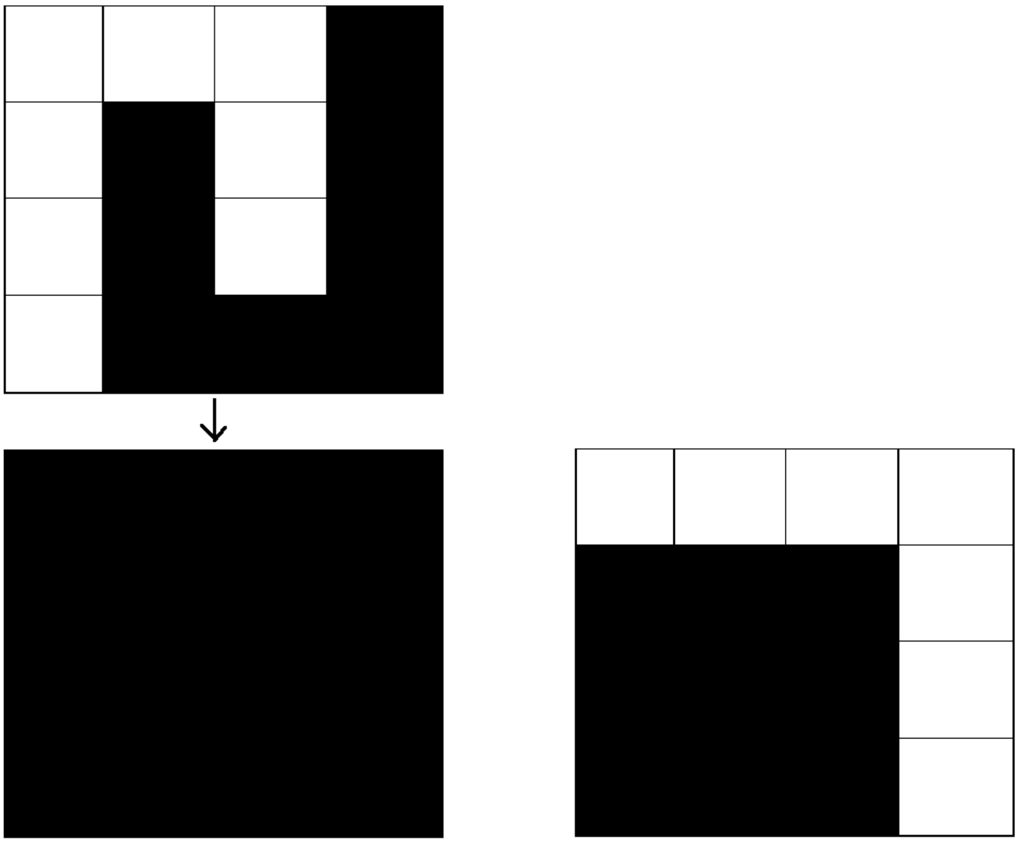
standard input	standard output
1 3 101 010	1
4 4 0001 0101 0101 0111 0000 1110 1110 1110	7

Note

In the first example, you can apply the tool to the middle cell twice. This way, two images will differ only in 1 cell.



In the second example, you can just make the entire image black. This way, two images will differ in 7 cells.



Problem J. ABC Legacy

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given a string S of length $2n$, consisting of the characters `A`, `B` and `C`. Determine if S can be split into n non-intersecting subsequences, each of which forms one of the strings "`AB`", "`AC`", "`BC`". If it is possible, find such a splitting.

Input

The first line of input contains one integer n ($1 \le n \le 10^5$).

The second line of input contains a string S of length $2n$, consisting of the characters `A`, `B` and `C`.

Output

If the splitting is not possible, print "`NO`" (without quotes).

If the splitting is possible, print "`YES`" (without quotes), followed by n lines, each describing two indices for the i-th subsequence ($1 \le l_i < r_i \le 2n$).

Examples

standard input	standard output
3 BABBCC	YES 3 5 1 6 2 4
2 CBAC	NO
1 AA	NO
3 ABCACB	YES 2 3 4 6 1 5

Problem K. Amazing Tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Consider an undirected tree. The following algorithm constructs a post-order traversal of the tree:

```
fun dfs(v):
    mark v as used
    for u in neighbours(v):
        if u is not used:
            dfs(u)
    append v to order
```

The post-order traversal will be in the list *order*.

You are allowed to choose the order of neighbors for each vertex as well as the starting vertex. What is the lexicographically minimal *order* you can get?

Input

The first line of input contains one integer T ($1 \le T \le 10^5$) — the number of test cases you need to process. Description of the test cases follows.

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

The i-th of the next $n-1$ lines contains two integers u_i, v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), meaning that there is an undirected edge (u_i, v_i) in the tree. It is guaranteed that the given graph is a tree.

The sum of n over all test cases in one test file does not exceed $2 \cdot 10^5$.

Output

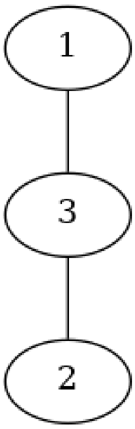
For each test case print the lexicographically minimal order on a separate line.

Example

standard input	standard output
3 3 1 3 3 2 3 2 1 1 3 7 1 2 1 3 2 4 2 5 3 6 3 7	1 2 3 2 1 3 4 5 2 1 6 3 7

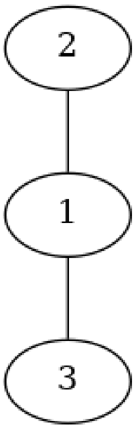
Note

The first test looks as follows:



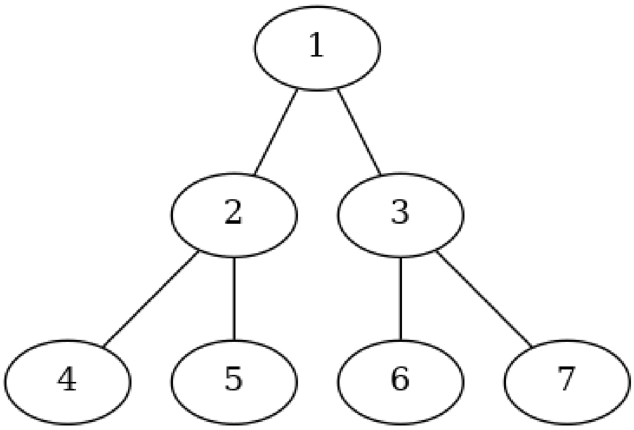
By starting in vertex 1 we can only get order 2 3 1. By starting in vertex 2 we can only get order 1 3 2. By starting in vertex 3 we can get two orders: 1 2 3 and 2 1 3. The lexicographically minimal of the four orders is 1 2 3.

The second test looks as follows:



By starting in vertex 1 we can get two orders: 2 3 1 and 3 2 1. By starting in vertex 2 we can only get order 3 1 2. By starting in vertex 3 we can only get order 2 1 3. The lexicographically minimal of the four orders is 2 1 3.

The third test looks as follows:



The lexicographically minimal order is 4 5 2 1 6 3 7 it can be obtained by starting in node 7.

Problem L. Jason ABC

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given a string S of length $3n$, consisting of the characters `A`, `B` and `C`. You are allowed to perform the following operation:

- Select some subsegment of this string and a character c (one of `A`, `B` and `C`). Then, replace all the characters on the subsegment with c.

Find the smallest number of times that you would have to apply the operation above to get a string which contains each of characters `A`, `B` and `C` exactly n times. It can be shown that it is always possible to get such a string.

In addition, find a sequence of operations of the smallest possible length. If there are many such sequences, you can output any of them.

Input

The first line of input contains a single integer n ($1 \le n \le 3 \cdot 10^5$).

The second line of the input contains a string S of length $3n$, consisting of the characters `A`, `B` and `C`.

Output

In the first line print the minimum number of operations k.

In the i-th of the next k lines print 2 integers l_i, r_i and a character c_i ($1 \le l_i \le r_i \le 3n$, $c_i \in \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$), denoting that in the i-th operation you will replace each of the characters $S_{l_i}, S_{l_i+1}, \ldots, S_{r_i}$ with c_i.

If there is more than one solution with a minimum number of operations, you can print any one of them.

Examples

standard input	standard output
1 AAA	2 2 3 B 3 3 C
1 CAB	0
3 ABABCABAB	1 1 2 C

Note

In the first sample, the string will undergo the following transformations:

`AAA` $\to$ `ABB` $\to$ `ABC`.

In the second sample, the string already contains exactly one `A`, one `B` and one `C`.

In the third sample, the string will undergo the following transformation:

`ABABCABAB` $\to$ `CCABCABAB`. Now, it contains each letter 3 times.

Problem M. Counting Phenomenal Arrays

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Let's call an array $[a_1, a_2, \ldots, a_k]$ of positive integers **phenomenal**, if the product of its elements is equal to the sum of its elements (i.e. if $a_1 a_2 \ldots a_k = a_1 + a_2 + \ldots + a_k$) .

For example, the array $[2, 2]$ is phenomenal, because $2 \cdot 2 = 2 + 2 = 4$, and $[3, 1, 2]$ is phenomenal, because $3 \cdot 1 \cdot 2 = 3 + 1 + 2 = 6$, but the array $[2, 3]$ is not phenomenal, as $2 \cdot 3 \neq 2 + 3$.

Let $f(i)$ denote the number of phenomenal arrays of size i. It can be shown that for any fixed $i \geq 2$ there is only a finite number of phenomenal arrays of size i.

You are given an integer n. Find $f(2), f(3), \ldots, f(n)$. As these numbers can be very big, output them modulo P, where P is a given prime number.

Input

The only line of the input contains two integers n, P ($2 \leq n \leq 2 \cdot 10^5$, $10^8 \leq P \leq 10^9$, P is prime).

Output

Output $n - 1$ integers — the values $f(2), f(3), \ldots, f(n)$ modulo P.

Example

standard input	standard output
`7 804437957`	`1 6 12 40 30 84`

Problem N. A-series

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

There are $N + 1$ different sizes of paper: $A0$, $A1$, $A2$, $\ldots$, AN, where each size is twice larger than the next one.

You have a_0 pieces of paper of size $A0$, a_1 of size $A1$, $\ldots$, a_N pieces of size AN. You want to obtain **at least** b_0 pieces of size $A0$, b_1 of size $A1$, $\ldots$, b_N pieces of size AN. At any point you can fold and cut a paper in half, obtaining two pieces of smaller size (e.g. $A4 \to A5 \times 2$). What is the minimum number of cuts needed to obtain the required pieces?

Input

The first line contains a single integer N $(1 \le N \le 2 \cdot 10^5)$.

The second line contains $N + 1$ integers $a_0, a_1, \ldots, a_N$ $(0 \le a_i \le 10^9)$.

The third line contains $N + 1$ integers $b_0, b_1, \ldots, b_N$ $(0 \le b_i \le 10^9)$.

Output

Output a single integer — the minimum number of cuts needed to obtain the required pieces, or -1, if it's not possible to obtain them.

Examples

standard input	standard output
1 10 0 0 19	10
1 10 0 0 21	-1
3 2021 11 21 10 10 21 11 2021	1758