Problem A. Digital Nim

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Algosia and Bajtek invented a new game, which they called *Digital Nim*. This game is played by two players, who take turns to make a move. The game requires one pile containing a certain number of stones. In their turn, a player must take from the pile any positive number of stones they choose; however, their options are limited. If there are currently x stones in the pile, then the number chosen by the player cannot exceed the sum of the digits of x in its decimal representation. For example, if there are currently 4257 stones in the pile, one can take between 1 and $4 + 2 + 5 + 7 = 18$ stones. The player who takes the last stone wins.

Your task is to determine who will win if there are initially n stones in the pile, Algosia makes the first move, and both play optimally. Additionally, you need to solve multiple test cases.

Input

The first line of the standard input contains a single integer t ($1 \le t \le 10\,000$), indicating the number of test cases.

The following t lines contain descriptions of the test cases. Each of them consists of a single integer n ($1 \le n \le 10^{18}$), indicating the initial number of stones in the pile.

For your convenience, all values of n are pairwise different and sorted in increasing order.

Output

Output t lines. The i-th of them should contain one word `Algosia` or `Bajtek`, indicating the winner in the i-th test case.

Example

standard input	standard output
4 1 10 42 190	Algosia Bajtek Algosia Algosia

Problem B. The Doubling Game 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

The Doubling Game is more of a puzzle than a game. The game board consists of n fields numbered with integers from 1 to n. The fields are connected by $n-1$ segments such that from every field one can move to any other field by only traveling along the connecting segments. In other words, the game board forms a tree. Initially, every vertex of the tree contains exactly one token.

The only available move is to select two fields connected by a segment with the same (positive) number of tokens and transfer all tokens from one of these fields to the other.

Your task is to count how many different token arrangements the board can have after any (possibly empty) sequence of moves. Two arrangements are considered different if and only if there is at least one field that contains a different number of tokens in both arrangements. Since the number of arrangements can be very large, it is enough if you provide the remainder of the division by $10^9 + 7$.

Input

The first line of input contains one integer n ($1 \le n \le 3 \cdot 10^5$) indicating the number of fields on the board.

The next $n-1$ lines contain two integers each a_i and b_i ($1 \le a_i, b_i \le n; a_i \ne b_i$) indicating that fields a_i and b_i are connected by a segment.

Output

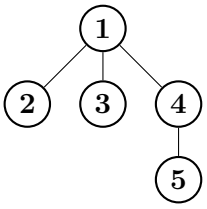
In the only line of the output, there should be one integer indicating the remainder of the division by $10^9 + 7$ of the number of possible token arrangements.

Example

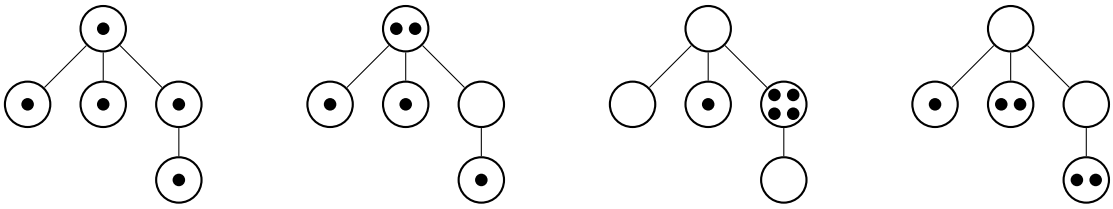
standard input	standard output
5 1 2 1 3 1 4 4 5	21

Note

The game board in the sample test (along with vertex numbers) looks as follows:



Some of the possible token arrangements, for instance:



Problem C. Cliques

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

Given is a tree $\mathcal{T}$ with n vertices numbered with consecutive natural numbers from 1 to n. Using it, we will create an undirected, initially empty graph $\mathcal{G}$. There are two types of operations to be performed:

- `+` v w $(1 \le v \le w \le n)$ – a vertex is added to the graph $\mathcal{G}$, which is labeled with a pair of numbers (v, w).
- `-` v w $(1 \le v \le w \le n)$ – one vertex labeled with the pair of numbers (v, w) is removed from the graph $\mathcal{G}$.

The pairs of numbers written in the vertices of graph $\mathcal{G}$ correspond to paths from the given tree $\mathcal{T}$ – these two numbers indicate the indices of the two ends of such a path, and they can be equal if the path consists of a single vertex.

At any given time, two vertices of the graph $\mathcal{G}$ are connected by an edge if the paths from $\mathcal{T}$ corresponding to them have at least one vertex in common. After adding a new vertex to the graph $\mathcal{G}$, edges are attached to it according to this rule, and when a vertex is removed, all the edges incident to it are also removed.

Your task is, after each operation, to output the number of non-empty subsets of vertices of the graph $\mathcal{G}$ that form cliques. A clique is a subgraph in which every pair of vertices is connected by an edge. Since this number can be very large, it's sufficient to output its remainder when divided by $10^9 + 7$.

Input

The first line of standard input contains one integer n $(2 \le n \le 200\,000)$, indicating the number of vertices of the tree $\mathcal{T}$.

Each of the next $n - 1$ lines contains two integers. The numbers in the i-th of these lines are a_i and b_i $(1 \le a_i, b_i \le n)$, indicating the existence in the tree $\mathcal{T}$ of an edge connecting vertices numbered a_i and b_i. It is guaranteed that the given edges describe a valid tree.

The next line contains one integer q $(1 \le q \le 50\,000)$, indicating the number of modifications to the graph $\mathcal{G}$.

Each of the next q lines is of one of the two possible types:

- `+` v w $(1 \le v \le w \le n)$ – a vertex is added to the graph $\mathcal{G}$, corresponding to the path between vertices v and w of the tree $\mathcal{T}$.
- `-` v w $(1 \le v \le w \le n)$ – one vertex corresponding to the path between vertices v and w of the tree $\mathcal{T}$ is removed from the graph $\mathcal{G}$.

Multiple vertices in the graph $\mathcal{G}$ can have the same pair of numbers written in them. It's guaranteed that when instructed to remove a vertex with a certain pair of numbers, at least one such vertex exists in the graph $\mathcal{G}$. When instructed to remove, only one vertex with the corresponding path should be removed, even if more of them currently exist.

Output

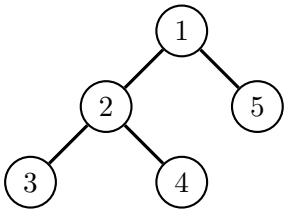
The output should contain q lines – the i-th of them should contain one integer, the number of non-empty subsets of vertices of the graph $\mathcal{G}$ that form cliques after the i-th modification. This number should be given as a remainder when divided by $10^9 + 7$.

Example

standard input	standard output
5 1 2 5 1 2 3 4 2 6 + 4 5 + 2 2 + 1 3 - 2 2 + 2 3 + 4 4	1 3 7 3 7 9

Note

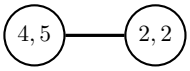
The tree $\mathcal{T}$ from the sample test looks as follows:



The following figures show the graph $\mathcal{G}$ after consecutive modifications.

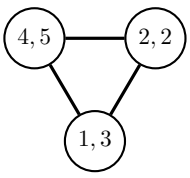
The graph $\mathcal{G}$ after the first modification:

The graph $\mathcal{G}$ after the second modification:

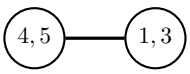


Both vertices in $\mathcal{G}$ are connected by an edge because the common vertex in $\mathcal{T}$ for both paths is vertex number 2.

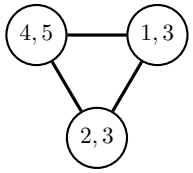
The graph $\mathcal{G}$ after the third modification:



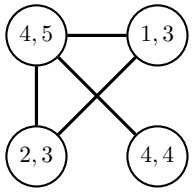
The graph $\mathcal{G}$ after the fourth modification:



The graph $\mathcal{G}$ after the fifth modification:



The graph $\mathcal{G}$ after the last modification:



Problem D. Colonization

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

Recently, Bytecja has begun the colonization of a newly discovered continent and its countless surrounding islands. Unfortunately, the indigenous people are not pleased with the arrival of the colonizers.

The colonization process on an island is as follows. There are n villages on the island, connected by $n-1$ bidirectional roads in such a way that you can get from any village to any other. In other words, the layout of villages and roads on each island forms a tree. Initially, in a village chosen by the colonizers, k colonizers appear, making this village colonized. Then, the colonizers can freely move along the roads or wait for the movements of other colonizers. A village is colonized the moment a colonizer arrives.

Taking over the whole island would be simple even with one colonizer, but there's a catch – if a colonized village is left without a waiting colonizer and it neighbors an uncolonized village, the locals might raid it, leading to a tragic outcome. So, this situation must be avoided. A colonizer has to guard such a village or be on the road between these villages, as the locals will definitely not bypass him.

The question for a given setup is the minimum number of k for which there's a choice of the initial village and a strategy for the colonizers' movements that allows for the whole island to be colonized.

Your task is to determine, given the number n, for each k, the possible road layouts on the island which require exactly k colonizers to be conquered. Two road layouts are considered different if the trees they represent are not isomorphic (meaning you need to count **unlabeled trees**). In other words, two road layouts are different if there is no bijection between villages in one layout and the villages in the other layout such that two villages in the first layout are connected by a road if and only if their corresponding villages in the second layout are connected by a road.

Since these numbers can be very large, you can provide the modulus of these numbers by a given prime.

Input

The first and only line of the standard input contains two integers n and p ($2 \le n \le 500$; $10^8+7 \le p \le 10^9+7$; p is a prime number), representing the considered number of villages on the island and the mentioned prime number, respectively.

Output

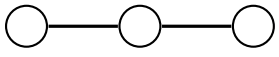
The first and only line of the standard output should contain n integers. The k-th number should represent the number of different road layouts among the n villages that require exactly k colonizers to be conquered, provided modulo p.

Examples

standard input	standard output
3 100000007	1 0 0
6 300000007	1 5 0 0 0 0
10 1000000007	1 104 1 0 0 0 0 0 0 0

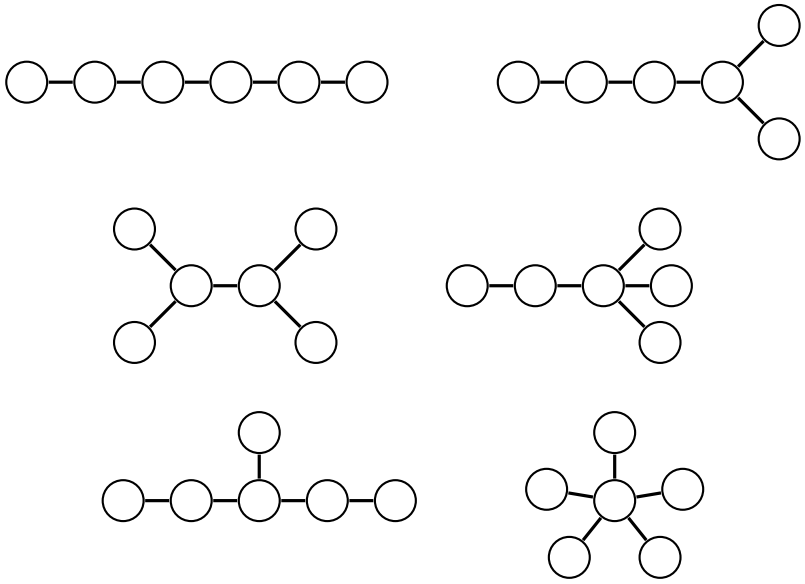
Note

In the first sample test, there is only one possible road layout and it looks as follows:



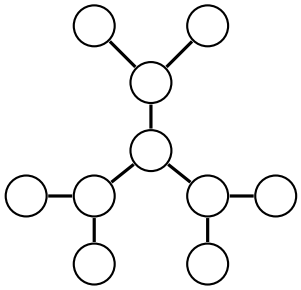
To colonize such an island, only one colonizer is needed, provided he doesn't start in the central village.

In the second sample test, there are 6 possible road layouts and they look as follows:



Only the top-left of these layouts can be colonized with the help of just one colonizer. For all the others, exactly two of them are required. For example, in the middle-right layout, they can start in the far-left village, move together two villages to the right, and then one of them will wait for the other, who will then consecutively colonize the remaining three villages.

In the third sample test, there is only one layout that requires three colonizers and it looks as follows:



Problem E. Bus Lines

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

In Byteopolis, there are n bus stops, numbered with integers from 1 to n, connected by $n-1$ bidirectional roads in such a way that one can travel from any stop to any other only via the roads. In other words, the bus stops and their connecting roads form a tree.

There are m bus lines operating in Byteopolis. The bus of the i-th of these lines continuously[1] shuttles back and forth along the shortest path between stops c_i and d_i. One can get on or off any bus at any stop along its route, including the terminal stops. The lines are planned such that one can get from any stop to any other using only the buses and not travelling any distance on the roads in any other way.

Let $f(i, j)$ where $(1 \le i, j \le n)$, denote the minimum number of bus lines needed to travel from stop i to stop j. Specifically, for every i, $f(i, i) = 0$, and for all pairs (i, j), $f(i, j) = f(j, i)$. Your task, for every stop i, is to compute the value of $\sum_{j=1}^{n} f(i, j)$.

Input

The first line of the standard input contains a single integer n $(2 \le n \le 200\,000)$, the number of bus stops in Byteopolis.

The next $n-1$ lines contain two integers a_i and b_i $(1 \le a_i, b_i \le n)$ indicating that bus stops a_i and b_i are connected by a road. It's guaranteed that the roads of Byteopolis describe a valid tree.

The subsequent line of the input contains a single integer m $(1 \le m \le 200\,000)$, the number of bus lines.

The next m lines contain two integers c_i and d_i $(1 \le c_i, d_i \le n; c_i \ne d_i)$ indicating that there's a bus shuttling between stops c_i and d_i. It's guaranteed that one can get from any stop to any other using only the buses.

Output

In a single output line, output n integers, where the i-th of them should be equal to $\sum_{j=1}^{n} f(i, j)$.

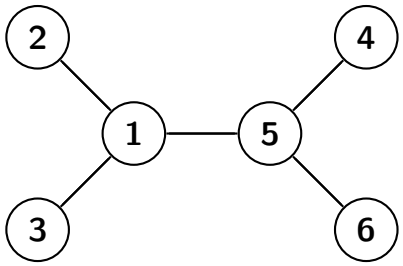
Example

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

Note

The layout of stops and roads in the sample test is represented as follows:

[1] Buses in Byteopolis are advanced and intelligent vehicles that don't need drivers and are powered by solar energy, so they never have to stop moving along their designated route.



If we wanted to travel from stop 2 to stop 4, we would first have to use the second bus line (operating between stops 2 and 3) and get off at stop 1. Next, we should use the first bus line (operating between stops 6 and 1) and get off at stop 5. Finally, we should reach our destination using the last bus line. It's not possible to move between these stops using only two lines, so $f(2, 4) = 3$.

All values of $f(i, j)$ in the sample test are presented in the matrix below, where the cell located in the i-th row and j-th column contains the value of $f(i, j)$. The numbers in the program's output should be equal to the sums of the values in consecutive rows:

	1	2	3	4	5	6
1	**0**	**1**	**1**	**2**	**1**	**1**
2	**1**	**0**	**1**	**3**	**2**	**2**
3	**1**	**1**	**0**	**3**	**2**	**2**
4	**2**	**3**	**3**	**0**	**1**	**1**
5	**1**	**2**	**2**	**1**	**0**	**1**
6	**1**	**2**	**2**	**1**	**1**	**0**

Problem F. Max-Min

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

You are given a sequence $a_1, \ldots, a_n$ consisting of n integers. There will be a total of q operations performed on this sequence. Each of them involves increasing or decreasing a single element of the sequence by 1. After each operation, print the value of the following expression:

$$\sum_{i=1}^{n} \sum_{j=i}^{n} \left(\max_{i \le k \le j} (a_k) - \min_{i \le k \le j} (a_k) \right).$$

Input

The first line of input contains two integers n and q ($1 \le n, q \le 500\,000$), representing the length of the sequence and the number of operations, respectively. The second line contains n integers $a_1, \ldots, a_n$ ($|a_i| \le 100\,000$), indicating the initial values of the sequence elements. The next q lines describe the individual operations. Each of them is of one of the two types:

- the symbol `+` and an integer p ($1 \le p \le n$) – operation to increase the value of a_p by one,
- the symbol `-` and an integer p ($1 \le p \le n$) – operation to decrease the value of a_p by one.

Output

The output should contain q lines, and each of them should contain a single integer. The number in the i-th line should contain the sought value of the expression after performing the first i operations.

Example

standard input	standard output
3 6 0 0 -1 + 3 + 3 - 2 - 2 + 2 + 1	0 2 5 8 5 6

Note

The sequence after consecutive operations looks as follows:

- $0, 0, 0,$
- $0, 0, 1,$
- $0, -1, 1,$
- $0, -2, 1,$
- $0, -1, 1,$
- $1, -1, 1.$

Problem G. Matrix Inverse

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	512 megabytes

Given a prime number p, we call a square array of size $n \times n$ with values ranging from 0 to $p-1$ inclusive, a *matrix*. The value in the i-th row and j-th column (for $1 \le i, j \le n$) of matrix M is denoted as $M_{i,j}$. Since we don't know of any other large prime numbers than $10^9 + 7$, we'll now assume that $p = 1\,000\,000\,007$.

We call matrix B the *inverse* of matrix A if the remainder when dividing the sum

$$\sum_{\ell=1}^{n} A_{i,\ell} \cdot B_{\ell,j}$$

by p is 1 when $i = j$, or 0 otherwise.

Matrix A is given. We guarantee that there is exactly one matrix B that is the inverse of matrix A. Your task is to find matrix B.

To make your life easier, we also give you matrix C, which differs from the sought matrix by at most **twelve** elements. Your task is to print a list of elements in which matrix C differs from the desired matrix B.

Input

The first line of input contains one integer n ($1 \le n \le 2000$) denoting the dimension of matrix A.

The next n lines describe matrix A; the i-th of these lines contains n integers, where the j-th number is element $A_{i,j}$ ($0 \le A_{i,j} \le p - 1$). We guarantee that there exists exactly one matrix B that is the inverse of matrix A.

The subsequent n lines contain a description of matrix C in a similar format. You can assume that matrix C differs from matrix B in at most twelve elements. Specifically, there are at most twelve distinct pairs of natural numbers (i, j) such that $1 \le i, j \le n$ and $B_{i,j} \ne C_{i,j}$.

Output

The first line of the output should contain a single integer k ($0 \le k \le 12$) denoting the number of elements in matrix C that need to be changed to obtain matrix B.

Each of the next k lines should contain three integers. The numbers in the i-th row, sequentially x_i, y_i, and w_i ($1 \le x_i, y_i \le n$, $0 \le w_i \le p - 1$), indicate that in the sought matrix B, it holds that $B_{x_i,y_i} = w_i$, while $C_{x_i,y_i} \ne w_i$. The changed elements should be printed in ascending order of row numbers, and in the event of ties, in ascending order of column numbers.

Note that according to the problem conditions, the correct output is uniquely defined.

Examples

standard input	standard output
2 1 1 0 2 1 2 3 500000004	2 1 2 500000003 2 1 0
3 0 1 0 0 0 1 1 0 0 0 0 1 1 0 0 0 1 0	0

Note

The inverse of the matrix $A = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}$ is the matrix $B = \begin{bmatrix} 1 & 500\,000\,003 \\ 0 & 500\,000\,004 \end{bmatrix}$. Hence, in the matrix C provided in the input, two elements need to be changed.

Problem H. Inverse Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	45 seconds
Memory limit:	1024 megabytes

Consider the following task. You are given a tree (a connected undirected graph without cycles) with n ($n \geq 1$) vertices. You need to count the number of correct colorings of its vertices with n colors. A coloring is considered correct if every two vertices separated by at most two edges have different colors. We consider two colorings different if there exists a vertex that has different colors in both of these colorings. As this number can be quite large, you should give its remainder when divided by $10^9 + 7$.

Your task is to solve the inverse problem – given a number r from the interval $[1, 10^9 + 6]$, find any tree with the smallest number of vertices for which the answer to the above problem is the number r. It can be proven that for each possible number r from the given range, there exists at least one such tree.

Input

In the first line of standard input, there is one integer t ($1 \leq t \leq 10$), representing the number of test cases.

In the next t lines, there is one integer each. The number in the i-th of these lines is r_i ($1 \leq r_i \leq 10^9 + 6$).

All values of r_i are pairwise distinct.

Output

The output should contain t blocks: the i-th of these blocks should contain an answer for the i-th test case.

The block describing a tree should begin with a line containing a single positive integer n_i – the number of vertices in your tree.

In the next $n_i - 1$ lines of the block, there should be two integers each. The numbers in the j-th of these lines, $a_{i,j}$ and $b_{i,j}$ ($1 \leq a_{i,j}, b_{i,j} \leq n_i$), should indicate the existence of an edge connecting the vertices with numbers $a_{i,j}$ and $b_{i,j}$. The tree's vertices are numbered with integers from 1 to n_i. If there are multiple trees with the minimum number of vertices for which the remainder of the division of the number of colorings by $10^9 + 7$ is r_i, you can print any of them.

Example

standard input	standard output
4 2 360 1 509949433	2 1 2 5 1 2 2 3 3 4 3 5 1 10 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10

Note

In the last test case, the number of tree colorings is 1509949440, which gives 509949433 modulo $10^9 + 7$.

Problem I. Boxes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

You are given n boxes of sizes $a_1, a_2, \ldots, a_n$. All box sizes are powers of two. In a box of size r, you can fit other boxes with a total size not exceeding $\frac{r}{2}$ (and similarly, in these boxes, you can fit other boxes, and so on). A box retains its fixed size regardless of the packing structure inside it.

Your task is to plan how to nest the boxes in such a way that the number of boxes not nested inside anything is minimized.

Input

In the first line of standard input, there is a single integer t ($1 \le t \le 500\,000$), indicating the number of test cases. The descriptions of the test cases are given in the next $2t$ lines, and each of these descriptions consists of two lines.

The first line of the test case description contains one integer n ($1 \le n \le 100\,000$), indicating the number of boxes.

The second line of the test case description contains a sequence of n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{12}$; a_i are powers of the number 2 with non-negative integer exponents), indicating the sizes of the subsequent boxes.

The sum of the values of n for all test cases will not exceed 500 000.

Output

The output should contain t lines, and each of them should contain one integer. The number in the i-th line should indicate the minimum possible number of outer (i.e., not packed into any other) boxes in the i-th test case.

Example

standard input	standard output
4 5 1 2 1 1 8 3 1 1 1 6 1 1 1 4 1 2 3 8 4 2	1 3 3 1

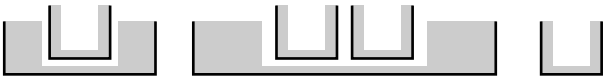
Note

Sample optimal box packings are shown below.

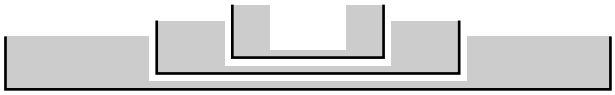
In the first test case:

In the second test case:

In the third test case:



In the fourth test case:



Problem J. Sequence Folding

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

Given is a binary sequence $a_1, a_2, \ldots, a_n$, where n is a power of two. In one move, you can choose any of its elements and change it to its opposite (i.e., choose index i from the interval $[1, n]$ and change a_i to $1 - a_i$). Additionally, if at some point the sequence is a palindrome (i.e., for every valid index i, $a_i = a_{n+1-i}$ holds), you can cut the right half of the sequence, leaving only $a_1, a_2, \ldots, a_{\frac{n}{2}}$, and then change n to the length of the new sequence (i.e., divide it by 2). Such an action is not counted as a move – by the number of moves, we mean only the number of element changes. You can freely interleave changes of the sequence elements and halving its length. Of course, you can't shorten the sequence if it consists of only one element.

Determine the minimum number of moves needed to make the sequence consist of only one element.

Additionally, n can be large, and the sequence is given by a list of positions of ones in it (the rest of the elements are zeros).

Input

In the first line of standard input, there are two integers n and m ($1 \le n \le 10^{18}$; $1 \le m \le 100\,000$; $m \le n$; n is a power of 2 with a non-negative integer exponent), representing respectively the length of the sequence and the number of ones in it.

In the second line, there is a sequence of m integers $p_1, p_2, \ldots, p_m$ ($1 \le p_i \le n$; $p_i < p_{i+1}$), indicating the positions of the consecutive ones in the sequence $a_1, a_2, \ldots, a_n$.

Output

The output should contain one integer, indicating the minimum number of moves required to shorten the sequence $a_1, a_2, \ldots, a_n$ to a single-element sequence.

Example

standard input	standard output
8 3 1 5 8	2

Note

In the sample test, the sequence `10001001` is given. There are three ways to shorten it to one element using two moves:

- $\texttt{10001001} \xrightarrow[+1]{} \texttt{10011001} \longrightarrow \texttt{1001} \longrightarrow \texttt{10} \xrightarrow[+1]{} \texttt{11} \longrightarrow \texttt{1}$
- $\texttt{10001001} \xrightarrow[+1]{} \texttt{10011001} \longrightarrow \texttt{1001} \longrightarrow \texttt{10} \xrightarrow[+1]{} \texttt{00} \longrightarrow \texttt{0}$
- $\texttt{10001001} \xrightarrow[+1]{} \texttt{10000001} \longrightarrow \texttt{1000} \xrightarrow[+1]{} \texttt{0000} \longrightarrow \texttt{00} \longrightarrow \texttt{0}$

There is no way to shorten the sequence to one element in fewer than two moves, so the result is the number 2.

Problem K. Jump Graph

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

Given a permutation of numbers from 1 to n, we can create a directed graph with n vertices based on it. In such a graph, one can jump from any element of the permutation to any other, as long as the elements skipped along the way (excluding the starting one) are smaller than the target. Such a graph is called the *jump graph* of the permutation.

Formally, from vertex i there's an edge to vertex j ($i \neq j$), if p_j is larger than every p_k for k between i and j (not counting positions i or j). In other words, for $i \neq j$ the edge $i \to j$ exists, if $p_j > p_k$ for every k such that $\min(i, j) < k < \max(i, j)$.

Your task is, given a permutation, for each vertex in its jump graph, determine the sum of distances to all other vertices. The distance from one vertex to another is the number of edges on the shortest path from the first vertex to the second. It can be proven that the jump graph is always strongly connected, so there exists a path between any two vertices.

Input

The first line of the standard input contains a single integer n ($2 \leq n \leq 300\,000$), indicating the length of the permutation.

The second line contains a sequence $p_1, p_2, \ldots, p_{n-1}, p_n$ ($1 \leq p_i \leq n$; $p_i \neq p_j$ for $i \neq j$) which is a permutation of integers from 1 to n.

Output

The only line of the output should contain n integers: the i-th of them should indicate the sum of distances from the i-th vertex of the jump graph to all other vertices.

Example

standard input	standard output
6 1 6 3 2 5 4	11 7 7 7 6 8

Note

The adjacency matrix of the jump graph for the sample permutation is given below. If there's an edge from the i-th vertex to the j-th, then in the i-th row and j-th column there's a value of 1, otherwise, there's a value of 0.

	1	2	3	4	5	6
1	**0**	**1**	**2**	**3**	**2**	**3**
2	**1**	**0**	**1**	**2**	**1**	**2**
3	**2**	**1**	**0**	**1**	**1**	**2**
4	**2**	**1**	**1**	**0**	**1**	**2**
5	**2**	**1**	**1**	**1**	**0**	**1**
6	**2**	**1**	**2**	**2**	**1**	**0**

Problem L. Spectacle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

The chess club is organizing a chess spectacle. The club has n chess players numbered from 1 to n, where the i-th one has a *rating*[2] r_i. In the spectacle, $2k$ chess players will participate, who will be paired in k pairs, and in these pairs, they will simultaneously play k games. For the spectacle to be thrilling, the club wants the largest rating difference between the chess players in a pair to be as small as possible.

Your task is for every k from 1 to $\lfloor \frac{n}{2} \rfloor$ to calculate the smallest possible maximum rating difference of the chess players in a pair, if the club optimally chooses $2k$ chess players and pairs them.

Input

In the first line of the standard input, there is one integer n ($2 \le n \le 200\,000$), indicating the number of chess players.

In the second line, there are n integers, where the i-th one is r_i ($1 \le r_i \le 10^{18}$), indicating the rating of the i-th player.

Output

In the only output line, there should be $\lfloor \frac{n}{2} \rfloor$ integers. The k-th one should indicate the sought result if the club wants to create k pairs of chess players.

Example

standard input	standard output
6 100 13 20 14 10 105	1 5 6

Note

For $k = 1$, we need to pair chess players with numbers 2 and 4.

For $k = 2$, we can, for example, create the following pairs: $(4, 5)$ and $(1, 6)$.

For $k = 3$, we need to create the following pairs: $(1, 6)$, $(2, 5)$, and $(3, 4)$.

[2] A *rating* in chess is a number describing the skills of a player. The higher this number, the better the player is at chess.

Problem M. Pattern Search

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Bajtek is attending the course Introduction to Algorithms. Today, he learned efficient algorithms to solve the problem of pattern searching within a text. He already implemented an efficient program that reads from input the text s and pattern t, both made up of lowercase English letters, and determines the number of occurrences of pattern t as a substring of text s.

To test the correctness of his program, Bajtek generated a sample text s and pattern t. The program gave the correct answer, but Bajtek did not like that correct answer... He very much wanted to modify his test so that the number of pattern occurrences was as high as possible. Unfortunately, Bajtek's text editor struggles with editing large files. It only allows for arbitrary changes in the order of letters in the text s and – separately – for arbitrary changes in the order of letters in the pattern t.

Can you help Bajtek and determine the maximum number of occurrences of the pattern t in the text s if Bajtek makes an optimal edit to his test?

Input

The first line of input consists of one integer z ($1 \le z \le 100\,000$) – the number of independent test sets. The description of each test set is in a separate line and consists of two words s and t, made up of lowercase English letters.

You can assume that $1 \le |t| \le |s| \le 2\,000\,000$. Furthermore, the total length of words in the input file will not exceed $4\,000\,000$ characters.

Output

The output should have z lines; the i-th of them should contain one integer – the maximum number of occurrences of the pattern in the text after editing Bajtek's i-th test set.

Example

standard input	standard output
2 bajkaaall aal abca cba	2 1

Note

Explanation of the first example: Bajtek can change the order of letters in the text s to `balalajka`, and in the pattern t – to `ala`. After editing, the pattern appears twice in the text.