Mafia and Candies

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 mebibytes

In a certain kingdom, there are n cities. They are numbered by integers from 1 to n, with the capital being city 1. There are exactly $n-1$ bidirectional roads between the cities in this state, and they connect all cities to the capital (possibly indirectly).

Also, the mafia operates in the state. It has its own gang in each city. In city i, the number of mafiosos is exactly a_i.

Undoubtedly, there are also good people in the country, such as merchants, and you are one of them. You trade candies and periodically send them from one city to another. Unfortunately, your activities are complicated by the following circumstances.

1. Every time a cart with candies arrives in city i, the mafia activates. The mafiosos attack it and divide the captured loot equally among themselves. At the same time, the thieves do not want to repeat the fate from a certain famous song. Because of this, the leftover candies are left to the cart drivers as a consolation.
2. Gangs from neighboring cities can quarrel with each other. As a result of the clash, the road between the cities is destroyed. In the component separated from the capital, anarchy begins, and the number of mafiosos in all its cities increases. The king cannot allow such disorder; he sends masters to build a new road from a chosen city connected to the capital. Such a road can only lead to the end city of the destroyed road; after the road is built, all cities become connected again.
3. As a result of mafia propaganda or the king's raids, the composition of a gang can change. In this case, the number of mafiosos in city i becomes y.

As a hereditary businessman with connections, you have found out about all these events for years ahead. However, the only thing you care about is your business. For each transportation of x candies from u to v at time t, you want to know how many candies will reach their customer in the end.

Input

The first line contains two integers n and q ($2 \le n \le 10^5$, $1 \le q \le 10^5$): the number of cities in the kingdom and the number of events, respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{12}$): the initial number of mafiosos in each of the cities.

The next $n-1$ lines describe the roads of the kingdom; the j-th of these lines contains two integers x and y ($1 \le x, y \le n$): the numbers of cities connected by the j-th road. The given roads form a tree on n vertices.

Next is the description of the events. Each of the following q lines starts with an integer $t \in \{1, 2, 3\}$.

- If $t = 1$, the format is "$t\ \ u\ \ v\ \ x$": you want to send a cart with x ($1 \le x \le 10^{12}$) candies from city u to city v ($1 \le u, v \le n$). The mafia activates in each city on the simple path from u to v, inclusive.
- If $t = 2$, the format is "$t\ \ x\ \ y\ \ z\ \ d$" ($1 \le x, y, z \le n$): the gangs from cities x and y quarreled and destroyed the road. In each city of the component separated from the capital, d ($0 \le d \le 10^7$) new mafiosos joined the gang. Afterwards, a new road was built from city z to one of the cities x and y that became separated from the capital.
- If $t = 3$, the format is "$t\ \ i\ \ y$": the gang from city i ($1 \le i \le n$) now has size y ($1 \le y \le 10^{12}$).

In events of the second type, it is guaranteed that cities x and y are connected by a road at this moment, and after its destruction, the capital can still be reached from city z.

Output

In response to each event of the first type, output a single integer r: the number of candies you will manage to deliver from city u to city v.

In response to each event of the second type, output a single integer s: the total number of mafiosos in the component disconnected from the capital, just before the road from city z is built.

Example

standard input	*standard output*
4 11 5 6 2 5 1 2 4 2 2 3 1 4 4 7 1 3 4 5 1 4 3 5 3 4 4 1 2 4 11 1 2 4 9 2 4 2 3 3 1 2 4 9 2 3 4 1 0 1 2 4 9 1 2 4 11	2 1 0 1 3 7 1 7 3 0

Valid Assignment

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given an undirected graph without self-loops or multiple edges.

Assign a nonnegative integer value to each vertex of the graph. An assignment is *valid* if it satisfies two conditions:

- any two adjacent vertices have values that differ by exactly 1;
- if the value of a vertex is strictly less than the values of all its neighbors, then it is equal to 0.

Find the number of valid assignments modulo 998 244 353. Two assignments are different if there exists a vertex that has different values in these assignments.

Input

The first line contains an integer t, the number of test cases.

The first line of each test case contains two integers n and m ($2 \le n \le 10^6$, $1 \le m \le 10^6$): the number of vertices and edges in the graph.

Each of the next m lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$): the endpoints of the corresponding edge. The graph does not contain self-loops or multiple edges. Every vertex has at least one neighbor.

The sum of m over all test cases does not exceed 10^6.

Output

For each test case, output one integer: the number of valid assignments modulo 998 244 353.

Example

standard input	*standard output*
2 4 3 1 2 2 3 3 4 3 3 1 2 2 3 1 3	6 0

Reconstruction

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given an array a of n elements. Each element of the array belongs to the set $\{0, 1, 2\}$.

Let us define the reconstruction operation. From an array a of length k, an array b of length $k-1$ is built. Each b_i is computed from the pair $(a[i], a[i+1])$ as follows:

- if $a[i] = a[i+1]$, then $b[i] = a[i]$;
- otherwise, $b[i]$ is the number from the set $\{0, 1, 2\}$ that does not appear among $a[i]$ and $a[i+1]$.

After that, array a is replaced by array b.

The reconstruction is applied $n-1$ times until exactly one element remains. Determine it.

Input

The first line contains an integer t ($1 \le t \le 2 \cdot 10^5$): the number of test cases.

The first line of each test case contains an integer n ($2 \le n \le 10^6$).

The second line contains n integers $a[0], a[1], \ldots, a[n-1]$ ($0 \le a[i] \le 2$).

The sum of n over all test cases does not exceed 10^6.

Output

For each test case, output one integer: the element remaining after $n-1$ reconstructions.

Example

standard input	*standard output*
1 5 0 1 1 2 0	0

Penalties on a Segment of Cities

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

There are n cities. For each city i, we assign a value and pay the price: either the value is 0 and we pay p_i coins, or the value is 1 and we pay q_i coins.

In addition, m constraints are given. A constraint is defined by a segment $[\ell, r]$ and two penalties, a and b. If there is at least one city with value 0 among the cities from ℓ to r, inclusive, we pay a penalty of a coins. If there is at least one city with value 1 among them, we pay a penalty of b coins. The two penalties are independent: if both 0 and 1 are present, we pay $a + b$ coins.

Find the minimum total cost (prices plus penalties) among all ways to choose values for the cities.

Input

The first line contains an integer t: the number of test cases.

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

Each of the next n lines contains two integers p_i and q_i ($1 \le p_i, q_i \le 10^9$): the costs of choosing 0 and 1 for city i, respectively.

Each of the next m lines contains four integers ℓ, r, a, b ($1 \le \ell \le r \le n$, $1 \le a, b \le 10^9$): the parameters of a constraint.

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

The sum of m over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output one integer: the minimum total cost.

Example

standard input	*standard output*
2 3 2 1 5 5 1 3 3 1 3 10 10 2 2 100 1 1 1 7 8 1 1 1 1	20 8

Orientation of a Graph

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Consider a complete undirected graph on n vertices. Your task is to choose one of the two directions for each edge so that, for every vertex, the difference between its outdegree and indegree is divisible by d.

Find any such orientation, or determine that it does not exist.

Input

The first line contains an integer t ($1 \le t \le 10\,000$): the number of test cases.

The only line of each test case contains two integers n and d ($2 \le n \le 1000$, $2 \le d \le n$).

The sum of n over all test cases does not exceed 20 000.

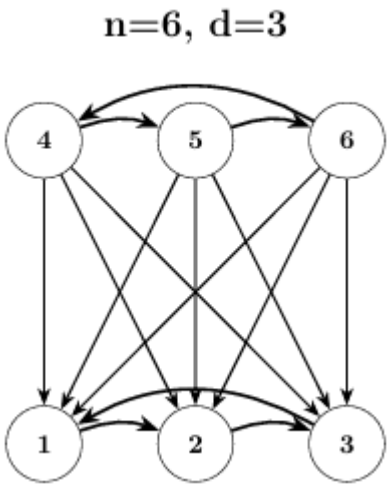
Output

For each test case, if an orientation does not exist, output "NO". Otherwise, output a string consisting of $\frac{n(n-1)}{2}$ characters. The first $n-1$ characters describe the orientation of the edges from vertex 1 (to vertices 2, 3, and so on up to n). The next $n-2$ characters describe the orientation of the edges from vertex 2 (to vertices 3, 4, and so on up to n). The process continues up to vertex $n-1$, for which you need to output one character, describing the orientation of the edge from $n-1$ to n. If the edge $i-j$ is oriented from i to j, output 1; otherwise, output 0.

Example

standard input	*standard output*
3 3 3 4 3 6 3	101 NO 100001000000101

Note



Trading

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

A trader trades cryptocurrency for n days. On day i, the price is a_i dollars per coin. Coins can be bought and sold in integer amounts. It is assumed that the trader initially has no coins but has enough dollars for any purchases.

The trader strictly follows a schedule. A *schedule* is a set of days marked for selling. On an unmarked day, the trader may either buy exactly one coin for a_i dollars or do nothing. On a marked day, the trader must sell at least one coin (but not more than he has) at the price of a_i dollars per coin and cannot buy.

A *strategy* under the given schedule is a choice, on each day, of how many coins to buy (on an unmarked day, at most one) or sell (on a marked day, at least one).

Consider a fixed array a. A trader's strategy is called *optimal* if, for this array a, there is no other strategy under any of the 2^n possible schedules with a larger total profit. The total profit is defined as the difference between the amount of dollars at the end of day n and the amount of dollars at the beginning of day 1 (coins are not counted).

You are given a schedule. Find the number of integer arrays a of length n with $1 \le a_i \le m$ for which there exists an optimal strategy under this schedule. Output the answer modulo 998 244 353.

Input

The first line contains an integer t, the number of test cases ($1 \le t \le 1695$).

The first line of each test case contains three integers n, m, and k ($1 \le n \le 5000$, $1 \le m \le 10^9$, $0 \le k \le n$): the number of days, the maximum price, and the number of marked days. If k is positive, the second line contains k integers $d_1 < d_2 < \ldots < d_k$ ($1 \le d_i \le n$): the marked days in increasing order. If $k = 0$, the second line is absent.

The sum of n over all test cases does not exceed 5000.

Output

For each test case, output one integer: the answer modulo 998 244 353.

Example

standard input	*standard output*
4 2 2 1 2 3 2 1 2 3 3 0 1 5 1 1	3 5 10 0

Note

First case. A strategy is optimal if its profit $a_2 - a_1$ is no less than the maximum possible, that is, $a_2 \ge a_1$. The arrays $(1, 1)$, $(1, 2)$, and $(2, 2)$ are suitable.

Second case. The profit of the strategy is $a_2 - a_1$. For optimality, it must be no less than the maximum possible profit over the whole array. The arrays $(1, 1, 1)$, $(1, 2, 1)$, $(1, 2, 2)$, $(2, 2, 1)$, and $(2, 2, 2)$ are suitable.

Third case. For $k = 0$, only the strategy with no actions is optimal (profit 0). This holds if the array is non-increasing ($a_1 \geq a_2 \geq a_3$). The number of such arrays is 10.

Fourth case. The trader is required to sell a coin on day 1, but initially he has no coins, and buying on a marked day is forbidden. No valid arrays exist.

XOR of All Children

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	1024 mebibytes

You are given a rooted tree with n vertices. Each vertex contains an integer. Every second, the values in all vertices are updated simultaneously: the new value of a vertex is the XOR of its current value and the current values of all its immediate children.

There are q independent queries. Each query is a pair (v, t): find the value in vertex v after exactly t seconds.

Input

The first line contains two integers n and q ($1 \le n \le 5 \cdot 10^5$, $1 \le q \le 5 \cdot 10^5$).

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($0 \le c_i < 2^{32}$): the initial values of the vertices.

The third line contains $n-1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$): the parent of each vertex (vertex 1 is the root).

Each of the following q lines contains two integers v and t ($1 \le v \le n$, $0 \le t \le 2 \cdot 10^5$): the vertex and the time of the query.

Output

For each query, output one integer: the value in vertex v after t seconds.

Example

standard input	*standard output*
7 9 5 3 6 1 4 7 2 1 1 2 2 3 3 1 0 1 1 1 2 2 1 2 2 3 3 4 5 5 2 6 1	5 0 5 6 3 3 1 4 7

A Near-Inversion Graph

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given a permutation $p_1, p_2, \ldots, p_n$ of the integers $1, 2, \ldots, n$. You need to answer q queries.

In each query, you are given two integers, ℓ and r, such that $1 \le \ell \le r \le n$. Consider all integers $i \in [\ell, r]$ and build an undirected graph on them. For each pair (i, j) such that $\ell \le i < j \le r$, add an edge between i and j if the following two conditions hold:

1. $p_i > p_j$;
2. There is no index k such that $i < k < j$ and $p_i > p_k > p_j$.

Your task is to determine the number of connected components in the resulting graph.

Input

The first line contains a single integer t ($1 \le t \le 10^4$), the number of test cases.

The first line of each test case contains two integers n and q ($1 \le n, q \le 5 \cdot 10^5$): the length of the permutation and the number of queries.

The second line contains n distinct integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$).

The next q lines describe the queries. Each of these lines contains two integers ℓ and r ($1 \le \ell \le r \le n$).

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

The sum of q over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, output q integers: the answers to all queries.

Example

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

Swap and Shift

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given a string of n lowercase English letters and an integer k. You can apply the following operation any number of times:

- choose two distinct indices i and j;
- swap s_i and s_j;
- then shift both letters forward by k positions in the alphabet cyclically (that is, if we number the letters of the alphabet from 0 for "a" to 25 for "z", the letter with number c becomes the letter with number $(c + k) \bmod 26$).

Determine whether it is possible to turn the string into a palindrome using such operations.

Input

The first line contains an integer t, the number of test cases ($1 \le t \le 5 \cdot 10^5$).

The first line of each test case contains two integers n and k ($2 \le n \le 10^6$, $0 \le k \le 25$).

The second line contains a string of length n consisting of lowercase English letters.

The sum of n over all test cases does not exceed 10^6.

Output

For each test case, print "Yes" if it is possible to obtain a palindrome, and "No" otherwise.

Example

standard input	*standard output*
4 3 0 aba 2 0 ab 4 1 abza 4 13 abcd	Yes No Yes No

Note

A palindrome is a string that reads the same forward and backward.

In the third test case of the example, applying the operation once to indices 1 and 3 produces the string "abba", which is a palindrome.

Minimum Spanning Forest

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

You are given an undirected graph with n vertices and, initially, no edges. The vertices are numbered by integers from 1 to n. Then m sets of edges are added to this graph. The i-th set is described by four integers, u_i, v_i, ℓ_i, and c_i, which mean the following:

- For each $k \in [0, \ell_i - 1]$, add an edge between vertices $u_i + k$ and $v_i + k$ with cost c_i.

Your task is to choose the maximum possible number of edges in the graph so that there is no cycle among the chosen edges, while minimizing the total cost of the chosen edges. In other words, find the weight of the minimum spanning forest of the resulting graph.

Input

The first line contains a single integer t ($1 \le t \le 10^4$), the number of test cases.

The first line of each test case contains two integers n and m ($1 \le n, m \le 5 \cdot 10^5$): the number of vertices in the graph and the number of edge sets.

The next m lines describe the edge sets. Each line contains four integers: u_i, v_i, ℓ_i, and c_i ($1 \le u_i, v_i \le n$, $1 \le \ell_i \le n$, $1 \le c_i \le 10^9$; additionally, $u_i + \ell_i - 1 \le n$ and $v_i + \ell_i - 1 \le n$).

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

The sum of m over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, output one integer: the weight of the minimum spanning forest of the resulting graph.

Example

standard input	*standard output*
4 1 1 1 1 1 1 4 1 1 2 3 1000000000 12 3 1 5 3 4 2 8 4 3 1 7 6 2 20 4 1 9 4 7 5 16 5 10 4 13 3 5 1 2 19 9	0 3000000000 24 151

Three Permutations

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	64 mebibytes

You are given three permutations a, b, and c of length n. Applying permutation c to array a produces an array d such that $d[c[i]] = a[i]$ for all i.

Before applying c to a, you may perform an operation on permutation c any number of times. In one operation, you can choose two indices i and j and swap $c[i]$ and $c[j]$.

For each k from 0 to n, find the minimum number of operations after which applying the modified permutation c to a produces an array that differs from b in exactly k positions, or determine that it is impossible.

Input

The first line contains an integer t ($1 \le t \le 5 \cdot 10^4$), the number of test cases.

The first line of each test case contains an integer n ($1 \le n \le 10^6$).

The next three lines contain permutations a, b, and c of length n. Each permutation contains every integer from 1 to n exactly once.

The sum of n over all test cases does not exceed 10^6.

Output

For each test case, output $n + 1$ integers, where the k-th of them is the minimum number of operations needed to obtain exactly k differences, or -1 if it is impossible.

Example

standard input	*standard output*
2 3 1 2 3 3 1 2 2 3 1 2 2 1 1 2 1 2	0 -1 1 2 1 -1 0

Note

In the first test case, the array is $a = [1, 2, 3]$. After applying the permutation $c = [2, 3, 1]$, the element from the 1-st position moves to the 2-nd, from the 2-nd to the 3-rd, and from the 3-rd to the 1-st, and the resulting array is $d = [3, 1, 2]$. In this case, it coincides with the array b. So, the answer for $k = 0$ differences is 0 operations.

Graph Built by XOR

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given an array a of n distinct non-negative integers. An undirected weighted graph with n vertices is built from the array: there is an edge between vertices i and j if and only if $a_i \oplus a_j \le m$, and its weight is $a_i + a_j$.

Find the length of the shortest path from vertex 1 to vertex n, or print -1 if no such path exists.

Input

The first line contains an integer t, the number of test cases ($1 \le t \le 10^5$).

The first line of each test case contains two integers n and m ($2 \le n \le 2 \cdot 10^5$, $0 \le m < 2^{30}$).

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

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

Output

For each test case, print one integer: the length of the shortest path from vertex 1 to vertex n, or -1 if no such path exists.

Example

standard input	*standard output*
3 4 2 0 2 3 1 3 0 0 1 2 5 7 0 4 7 3 6	1 -1 6

Board Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1.5 seconds
Memory limit:	1024 mebibytes

You are given a prime number p and a list S of n integers $a_1, a_2, \ldots, a_n$ which are all less than p. Two players take turns making moves.

On each move, the player chooses any number from the list S, removes it from S, and writes it on the board. After the move, if the product of all numbers written on the board is congruent to x^2 modulo p for some integer x, then the player who made the move gets 1 point.

The game ends after n moves, when the list S becomes empty.

Each player aims to maximize the difference between their own score and their opponent's score. Find this difference for the first player, assuming that both play optimally.

Input

The first line contains an integer t ($1 \le t \le 10^5$), the number of test cases.

The first line of each test case contains two integers n and p ($1 \le n \le 2 \cdot 10^6$, $2 \le p \le 1\,999\,999\,973$, p is prime).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i < p$).

The sum of n over all test cases does not exceed $2 \cdot 10^6$.

Output

For each test case, output one integer: the difference between the number of points of the first player and the second player under optimal play.

Example

standard input	*standard output*
1 3 5 1 2 3	2

Transfers in a Cactus Graph

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

In a certain transportation network, every vertex of the graph is a transfer hub. Being at vertex u, one can buy a ticket that allows traveling to any other vertex v. The cost of such a ticket is

$$b_u + p_u \cdot \text{dist}(u, v),$$

where $\text{dist}(u, v)$ is the length of the shortest path between vertices u and v in the graph.

After arriving at vertex v, one may immediately finish the trip or buy a new ticket from vertex v. The number of transfers is unlimited. The total trip cost is the sum of the costs of all purchased tickets.

The transportation network is a connected undirected weighted graph that is a cactus. In other words, each edge of the graph belongs to at most one simple cycle.

The graph contains n vertices and m edges. The length of each edge w_j is given. For each vertex i, the numbers b_i and p_i are given.

For every vertex v, find the minimum cost of a trip from vertex 1 to vertex v.

Input

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

The first line of each test case contains two integers n and m ($2 \le n \le 10^5$, $n - 1 \le m \le \lfloor 3(n-1)/2 \rfloor$).

The next m lines contain the description of the edges. Each of these lines contains three integers u, v, and w ($1 \le u, v \le n$, $1 \le w \le 10^4$), meaning a bidirectional edge between vertices u and v of length w. The graph is a connected cactus, it contains no self-loops and no multiple edges.

Each of the next n lines contains two integers: b_i and p_i ($0 \le b_i \le 10^9$, $1 \le p_i \le 10^4$).

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

The sum of m over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output n integers. The v-th of them should be equal to the minimum cost of a trip from vertex 1 to vertex v. For vertex 1, the answer is 0.

Example

standard input	*standard output*
2 3 2 1 2 5 2 3 4 10 1 20 2 30 5 3 2 1 2 5 2 3 5 0 10 1 1 100 10	0 15 19 0 50 56