The 3rd Universal Cup

Stage 10: West Lake

September 28-29, 2024

This problem set should contain 13 problems (A to M) on 21 numbered pages.

Problem A. Italian Cuisine

Time limit:	1 second
Memory limit:	1024 megabytes

BaoBao has prepared a pizza for you! This pizza is a convex polygon with stuffed crust along all of its edges. However, the crust is delicate, allowing you to cut only through its vertices and not into the middle of the edges. Unfortunately, there's a sizable circular piece of pineapple on the pizza that you absolutely want to avoid.

Calculate the largest single piece of pineapple-free pizza that you can obtain with a single straight cut and output its size. A piece is considered pineapple-free when no part of the pineapple falls strictly within it. That is, the area of intersection of the pineapple and the pizza is 0.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($3 \le n \le 10^5$) indicating the number of vertices of the pizza.

The second line contains three integers x_c, y_c, and r ($-10^9 \le x_c, y_c \le 10^9$, $1 \le r \le 10^9$), indicating the coordinates of the center of the pineapple and its radius.

For the following n lines, the i-th line contains two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$) indicating the coordinate of the i-th vertex. The vertices are listed in counter-clockwise order. No two points coincide. However there might be three points lying on the same line.

It is guaranteed that no part of the pineapple lies outside of the boundaries of the pizza. It's also guaranteed that the sum of n of all the test cases does not exceed 10^5.

Output

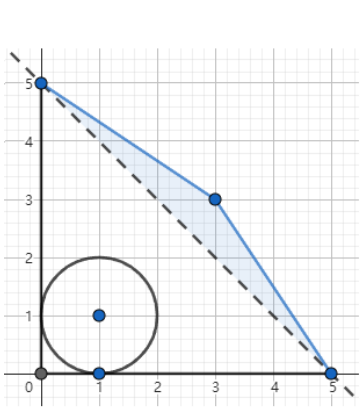
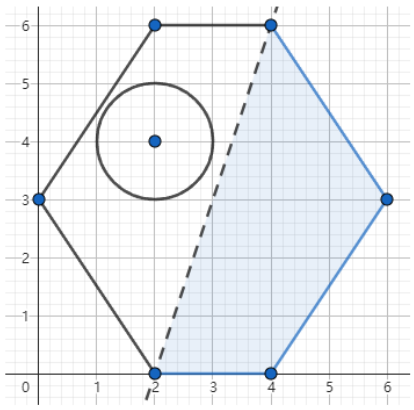
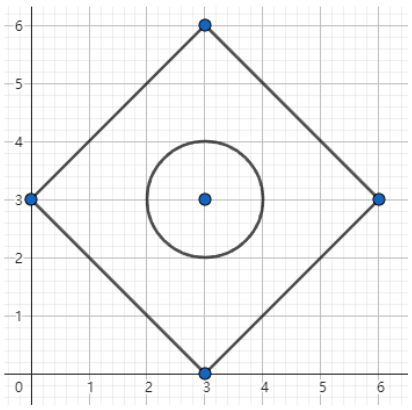
For each test case, output one line containing one integer, indicating the size of the largest piece of pineapple-free pizza multiplied by 2. It can be proven that this value will always be an integer. If there's no way to get a pineapple-free piece, output 0.

Example

standard input	standard output
3 5 1 1 1 0 0 1 0 5 0 3 3 0 5 6 2 4 1 2 0 4 0 6 3 4 6 2 6 0 3 4 3 3 1 3 0 6 3 3 6 0 3	5 24 0

Note

The sample test cases are shown below.

Problem B. Generated String

Time limit: 3 seconds
Memory limit: 1024 megabytes

You are given a template string $S = s_1 s_2 \cdots s_n$ of length n. A generated string is a string formed by concatenating several substrings of S. Formally, each generated string $T = f(k, \{l_i\}_{i=1}^k, \{r_i\}_{i=1}^k)$ is described by a positive integer k and k pairs of integers (l_i, r_i), where $T = s[l_1 : r_1] + s[l_2 : r_2] + \cdots + s[l_k : r_k]$. Here $s[l : r]$ denotes the substring $s_l s_{l+1} \cdots s_r$, and $+$ denotes string concatenation.

Your task is to maintain a multiset $\mathbb{A}$ of strings, supporting the following three types of operations:

- `+` $k\ l_1\ r_1\ l_2\ r_2\ \cdots\ l_k\ r_k$: Insert $f(k, \{l_i\}_{i=1}^k, \{r_i\}_{i=1}^k)$ into the multiset $\mathbb{A}$.
- `-` t: Erase the string inserted in the t-th operation from the multiset $\mathbb{A}$. It is guaranteed that the t-th operation is an inserting operation and the inserted string is not erased at this time.
- `?` $k\ l_1\ r_1\ l_2\ r_2\ \cdots\ l_k\ r_k\ m\ u_1\ v_1\ u_2\ v_2\ \cdots u_m\ v_m$: Answer the number of strings in the multiset $\mathbb{A}$ that begin with string $f(k, \{l_i\}_{i=1}^k, \{r_i\}_{i=1}^k)$ and end with $f(m, \{u_i\}_{i=1}^m, \{v_i\}_{i=1}^m)$.

Input

There is only one test case in each test file.

The first line contains two integers n and q ($1 \le n, q \le 10^5$), indicating the length of S and the number of operations.

The second line contains a string $s_1 s_2 \cdots s_n$ consisting of lower-cased English letters, indicating the template string.

For the following q lines, the i-th line contains an operation in the format described above. It is guaranteed that $1 \le l_i \le r_i \le n$, $1 \le u_i \le v_i \le n$. It's also guaranteed that the sum of k in all operations of type `+`, plus the sum of k in all operations of type `?`, plus the sum of m in all operations of type `?`, will not exceed 3×10^5.

Output

For each operation of type `?`, output one line containing one integer indicating the answer.

Example

standard input	standard output
8 7 abcaabbc + 3 1 3 2 4 3 8 + 2 1 4 1 8 + 1 2 4 ? 1 5 6 1 7 8 - 3 + 1 2 5 ? 1 2 3 1 5 5	2 1

Problem C. Permutation

Time limit: 1 second
Memory limit: 1024 megabytes

This is an interactive problem.

There is a hidden permutation of n. Recall that a permutation of n is a sequence where each integer from 1 to n (both inclusive) appears exactly once. Piggy wants to unravel the permutation with some queries.

Each query must consist of a sequence (not necessarily a permutation) with n integers ranging from 1 to n (both inclusive). Each query is answered with an integer x, indicating the number of the positions where the corresponding element in Piggy's query sequence matches that of the hidden permutation. For example, if the hidden permutation is $\{1, 3, 4, 2, 5\}$ and the sequence Piggy asks is $\{2, 3, 5, 2, 5\}$, he'll receive 3 as the answer.

As Piggy is busy recently, he gives this problem to you. Find the permutation with no more than 6666 queries.

Input

There is only one test case in each test file.

The first line of the input contains an integer n ($1 \le n \le 10^3$) indicating the length of the hidden permutation.

Interaction Protocol

To ask a query, output one line. First output `0` followed by a space, then print a sequence of n integers ranging from 1 to n separated by a space. After flushing your output, your program should read a single integer x indicating the answer to your query.

If you want to guess the permutation, output one line. First output `1` followed by a space, then print a permutation of n separated by a space. After flushing your output, your program should exit immediately.

Note that the answer for each test case is pre-determined. That is, the interactor is not adaptive. Also note that your guess does not count as a query.

To flush your output, you can use:

- `fflush(stdout)` (if you use `printf`) or `cout.flush()` (if you use `cout`) in C and C++.
- `System.out.flush()` in Java.
- `stdout.flush()` in Python.

Example

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

Note

Please note that if you receive a Time Limit Exceeded verdict, it is possible that your query is invalid or the number of queries exceeds the limit.

Problem D. Collect the Coins

Time limit: 1 second
Memory limit: 1024 megabytes

There are 10^9 cells arranged in a row, numbered from 1 to 10^9 from left to right. Two robots are patrolling in the cells. The maximum speed of each robot is v cells per second (v is an integer), indicating that if a robot is currently in cell p, it can move to any cell p' in the next second, as long as $1 \le p' \le n$ and $|p' - p| \le v$.

There will be n coins appearing in the cells. The i-th coin will appear at the t_i-th second in cell c_i. If a robot is in the same cell at that time, it can pick up the coin. Otherwise the coin will instantly disappear.

More formally, during each second, the following steps will happen in order:

- Each robot can move to a position no more than v cells away (it is OK to just stay in its current cell).
- Coins appear in the cells.
- If at least one robot is in the same cell with a coin, that coin is collected.
- All uncollected coins disappear.

Your task is to determine the initial positions of two robots before the 1-st second, and move them wisely to collect all the coins with the smallest possible v. Output this optimal value of v.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 10^6$) indicating the number of coins appearing in the cells.

For the following n lines, the i-th line contains two integers t_i and c_i ($1 \le t_i, c_i \le 10^9$) indicating the time when the i-th coin appears and the position where the i-th coin appears. It's guaranteed that $t_i \le t_{i+1}$ for all $1 \le i < n$. It's also guaranteed that $t_i \ne t_j$ or $c_i \ne c_j$ for all $i \ne j$.

It's guaranteed that the sum of n of all test cases will not exceed 10^6.

Output

For each test case, output one line containing one integer, indicating the smallest possible maximum speed of the robots. If it's impossible to collect all the coins, output `-1` instead.

Example

standard input	standard output
3 5 1 1 3 7 3 4 4 3 5 10 1 10 100 3 10 100 10 1000 10 10000	2 0 -1

Problem E. Normal Friends

Time limit: 8 seconds
Memory limit: 1024 megabytes

Segment trees are a data structure that Little Q is very fond of. They have simple structures, excellent time complexities, and powerful functionalities, so Little Q once spent a long time studying some properties of segment trees.

Recently, Little Q has started studying segment trees again, but with a difference: she has focused on more generalized segment trees. In a normal segment tree, for an interval $[l, r]$, we would take $\text{mid} = \left\lfloor \frac{l+r}{2} \right\rfloor$, and then split this interval into $[l, \text{mid}]$ and $[\text{mid}+1, r]$. In generalized segment trees, mid is not required to be exactly the midpoint of the interval, but mid still must satisfy $l \leq \text{mid} < r$. It is not hard to find that in generalized segment trees, the depth of the tree can reach $O(n)$ levels.

Little Q doesn't know how to implementer balanced BSTs, so she wants to use segment trees to implement all operations of balanced BSTs. One of the most famous operations that cannot be implemented using segment trees is reversing an interval, but Little Q is not convinced and wants to implement it using segment trees.

Specifically, when Little Q reverses an interval, she will first partition this interval into several maximal intervals represented by nodes in the segment tree; suppose from smallest to largest they are represented by the k nodes $a_1, a_2, \ldots, a_k$ in the segment tree. Then, Little Q will strip these k subtrees and reattach them to the segment tree in reverse order, that is, the subtree originally at the position of a_1 is replaced with the subtree of a_k, the position originally of a_2 is replaced with that of a_{k-1}, and so on; the position originally of a_k is replaced with the subtree of a_1. Finally, Little Q will swap the left and right children of all nodes in these k subtrees. It is easy to find that after such an operation, the tree is still a generalized segment tree, and the order of the elements from left to right is exactly the result after reversing the interval.

Given such a generalized segment tree, Little Q needs to perform m operations; each time, given a prefix $[1, x]$, she reverses it. At the same time, Little Q wants to know the value of k, that is, the number of intervals partitioned in the current generalized segment tree during the reverse operation.

Of course, Little Q easily solved this problem, which made her more confident not to learn how to write balanced BSTs. So, can you solve it?

Input

The first line contains two positive integers n and m ($2 \leq n, m \leq 3 \times 10^5$), representing the size of the tree and the number of operations.

The next line contains $n-1$ positive integers, given in depth-first search (DFS) order, providing the splitting point mid for each interval.

The next m lines each contain a positive integer x, representing an operation of reversing the prefix $[1, x]$.

Output

Output m lines, each line containing a positive integer, representing the value of k during each operation.

Example

standard input	standard output
3 3 2 1 2 3 2	1 1 2

Problem F. Triangle

Time limit: 1 second
Memory limit: 1024 megabytes

Given n strings $S_1, S_2, \cdots, S_n$ consisting of lower-cased English letters, we say three strings S_a, S_b and S_c form a triangle, if all the following constraints are satisfied:

- $S_a + S_b > S_c$ or $S_b + S_a > S_c$.
- $S_a + S_c > S_b$ or $S_c + S_a > S_b$.
- $S_b + S_c > S_a$ or $S_c + S_b > S_a$.

Here $+$ is the string concatenation operation and strings are compared by lexicographic order. For example, `ba`, `cb` and `cbaa` forms a triangle, because:

- `cb` + `ba` = `cbba` > `cbaa`.
- `cbaa` + `ba` = `cbaaba` > `cb`.
- `cb` + `cbaa` = `cbcbaa` > `ba`.

Count the number of integer tuples (a, b, c) such that $1 \le a < b < c \le n$ and S_a, S_b, S_c forms a triangle.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 3 \times 10^5$) indicating the number of strings.

For the following n lines, the i-th line contains a string S_i ($1 \le |S_i| \le 3 \times 10^5$) consisting of lower-cased English letters.

It's guaranteed that the total length of the strings in a single test case does not exceed 3×10^5, and the total length of strings of all test cases does not exceed 10^6.

Output

For each test case, output one line containing one integer indicating the number of valid tuples.

Example

standard input	standard output
3 6 cbaa cb cb cbaa ba ba 3 sdcpc sd cpc 1 ccpc	16 0 0

Problem G. Stop the Castle 2

Time limit:	1 second
Memory limit:	1024 megabytes

There are n castles and m obstacles on a chessboard with 10^9 rows and 10^9 columns. Each castle or obstacle occupies exactly one cell and all occupied cells are distinct. Two castles can attack each other, if they're on the same row or the same column, and there are no obstacles or other castles between them. More formally, let (i, j) be the cell on the i-th row and the j-th column. Two castles positioned at (i_1, j_1) and (i_2, j_2) can attack each other, if one of the following conditions is true:

- $i_1 = i_2$ and for all $\min(j_1, j_2) < j < \max(j_1, j_2)$, there is no obstacle or castle positioned at (i_1, j).
- $j_1 = j_2$ and for all $\min(i_1, i_2) < i < \max(i_1, i_2)$, there is no obstacle or castle positioned at (i, j_1).

You have to remove k obstacles from the chessboard, but you don't want too many castles to attack each other. Minimize the number of pairs of castles which can attack each other after removing exactly k obstacles from the chessboard.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains three integers n, m, and k ($1 \le n, m \le 10^5$, $1 \le k \le m$) indicating the number of castles, the number of obstacles, and the number of obstacles you have to remove.

For the following n lines, the i-th line contains two integers r_i and c_i ($1 \le r_i, c_i \le 10^9$) indicating that the i-th castle is located on the r_i-th row and the c_i-th column.

For the following m lines, the i-th line contains two integers r'_i and c'_i ($1 \le r'_i, c'_i \le 10^9$) indicating that the i-th obstacle is located on the r'_i-th row and the c'_i-th column.

It's guaranteed that the occupied cells are distinct. It's also guaranteed that neither the sum of n nor the sum of m of all test cases will exceed 10^5.

Output

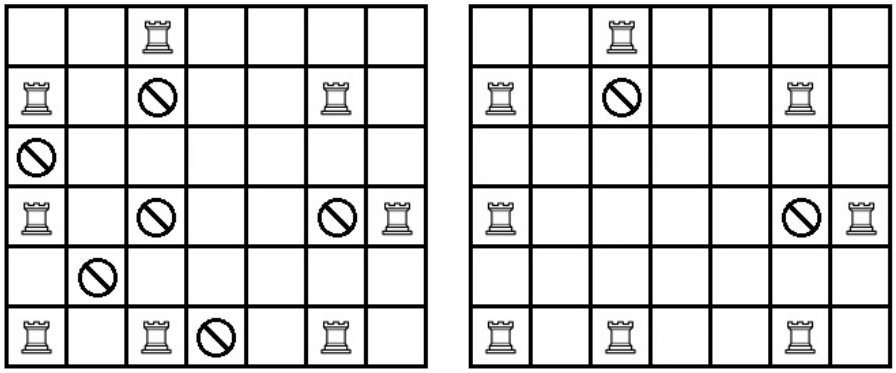
For each test case, first output one line containing one integer indicating the smallest number of pairs of castles which can attack each other after removing exactly k obstacles. Then output another line containing k distinct integers $b_1, b_2, \cdots, b_k$ ($1 \le b_i \le m$) separated by a space, indicating the indices of obstacles you want to remove. If there are multiple valid answers, you can output any of them.

Example

standard input	standard output
3 8 6 4 1 3 2 1 2 6 4 1 4 7 6 1 6 3 6 6 2 3 3 1 4 3 4 6 5 2 6 4 3 2 1 10 12 10 10 10 11 1 4 1 5 1 3 2 1 1 2 1 2 2 2 3	4 6 3 2 5 2 1 0 1 2

Note

For the first sample test case, the image on the left shows the original chessboard, and the image on the right shows the chessboard after removing 4 obstacles. After removing the obstacles, the pairs of castles which can attack each other are: the 2-nd and the 4-th castles, the 4-th and the 6-th castles, the 6-th and the 7-th castles, the 7-th and the 8-th castles.



For the third sample test case, as there is only 1 castle, there is no pair of castles which can attack each other.

Problem H. Intersection of Paths

Time limit: 5 seconds
Memory limit: 1024 megabytes

There is a tree with n vertices and $(n-1)$ edges, where the i-th edge connects vertices u_i and v_i, and has a weight of w_i.

Your task is to process q queries. The i-th query can be described as three integers a_i, b_i and k_i. This query will temporarily change the weight of the a_i-th edge to b_i. After that you should choose $2k_i$ distinct vertices $s_1, s_2, \cdots, s_{k_i}, e_1, e_2, \cdots, e_{k_i}$ and consider the k_i simple paths on the tree, where the p-th path starts from vertex s_p and ends at vertex e_p. We say an edge is good, if it is contained in all k_i paths. Maximize the total weights of good edges.

Note again that the change in the weight of each query is temporary. After each query you should change back the weight.

Input

There is only one test case in each test file.

The first line contains two integers n and q ($2 \le n \le 5 \times 10^5$, $1 \le q \le 5 \times 10^5$) indicating the number of vertices and the number of queries.

For the following $(n-1)$ lines, the i-th line contains three integers u_i, v_i and w_i ($1 \le u_i, v_i \le n$, $1 \le w_i \le 10^9$) indicating that the i-th edge connects vertices u_i and v_i, and has a weight of w_i.

For the following q lines, the i-th line contains three integers a_i, b_i and k_i ($1 \le a_i \le n-1$, $1 \le b_i \le 10^9$, $1 \le k_i \le \lfloor \frac{n}{2} \rfloor$) indicating the i-th query.

Output

For each query output one line containing one integer indicating the answer.

Example

standard input	standard output
7 3 1 2 20 2 3 10 2 4 40 4 6 10 1 5 30 5 7 10 2 100 1 5 50 2 2 100 3	160 110 20

Note

For the first query, choose $s_1 = 3$ and $e_1 = 7$.

For the second query, choose $s_1 = 4$, $s_2 = 6$, $e_1 = 7$ and $e_2 = 5$.

For the third query, choose $s_1 = 3$, $s_2 = 4$, $s_3 = 6$, $e_1 = 5$, $e_2 = 1$ and $e_3 = 7$.

Problem I. Find Yourself

Time limit: 1.5 seconds
Memory limit: 1024 megabytes

The Quantum Monster is a peculiar creature that exists simultaneously in different world lines in the form of a wave function. As long as there is a possible world line where the Quantum Monster has not been found, you can never capture it.

Now, there is a Quantum Monster hiding in a connected undirected graph with n nodes and m edges. You wish to capture it. The entire process proceeds in a loop as follows:

1. The Quantum Monster moves along an edge in the graph to an adjacent node.
2. You select some nodes in the graph to observe once. You can know whether the Quantum Monster is in the set of nodes you observed.
3. If the historical information you have is sufficient to uniquely determine the Quantum Monster's location, you succeed in capturing it, and the loop ends. Otherwise, return to step 1.

You want to know whether it is possible to capture this Quantum Monster. In other words, determine whether there exists a capturing strategy such that, regardless of the Quantum Monster's initial position and movement plan, you can uniquely determine its location in a finite number of steps.

Input

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

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

The next m lines each contain two integers u and v ($1 \le u, v \le n$), indicating that there is an undirected edge connecting u and v in the graph.

It is guaranteed that all graphs are connected undirected graphs with no self-loops or multiple edges, and the sum of m over all test cases does not exceed 10^6.

Output

For each test case, output a single line containing the string `YES` if a capturing strategy exists, or `NO` if it does not.

Example

standard input	standard output
3 3 3 1 2 2 3 3 1 4 4 1 2 2 3 3 4 4 1 6 6 1 2 2 3 3 4 4 5 5 6 6 1	NO YES NO

Problem J. Sheriruth

Time limit:	5 seconds
Memory limit:	1024 megabytes

You're given a simple directed graph $G = (V, E)$, in which $V = \{0, 1, 2, \ldots, n-1\}$ and $|E| = m$.

If $a, b, c \in V$ satisfy all of the following restrictions:

- $b \neq c$
- $(a \to b) \in E$
- $(a \to c) \in E$
- $(b \to c) \notin E$

Then we would join the edge $b \to c$ into the edge set E.

Keep doing such operation until you can't do it. We can prove that the final graph $G' = (V, E')$ is unique.

Then we would ask you q questions. For each question, we would give you nodes $u, v \in V$. You need to answer how many paths are from u to v on G' and along which we'd not meet any node for more than once.

For the reason that the answer may be too large, you need only to tell us the answer mod S.

Input

The first line of the input contains four integers n, m, q and S ($1 \leq n \leq 5 \times 10^5$, $0 \leq m \leq 10^6$, $1 \leq q \leq 10^6$, $1 \leq S \leq 2^{30}$).

The following m lines gives out each edge in E. Each line contains two integers u and v, ($0 \leq u, v \leq n-1, u \neq v$), which means an edge $u \to v$ in E. It is guaranteed that there are no multiple edges or self loops in the graph.

The following q lines describes all the questions. Each line contains two integers u and v ($0 \leq u, v \leq n-1$).

Output

Output q lines. The i-th line of these lines contains a single integer, indicating the answer modulo S.

Example

standard input	standard output
11 9 30 998244353 0 1 0 2 3 4 5 4 6 5 7 8 8 9 9 8 10 9 0 1 0 2 1 0 1 2 2 0 2 1 3 4 3 5 3 6 4 3 4 5 4 6 5 3 5 4 5 6 6 3 6 4 6 5 7 8 7 9 7 10 8 7 8 9 8 10 9 7 9 8 9 10 10 7 10 8 10 9	2 2 0 1 0 1 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 0 0 1 0 0 1 1

Problem K. Palindromic Polygon

Time limit:	1 second
Memory limit:	1024 megabytes

There is a convex polygon with n vertices. Vertices are numbered from 1 to n (both inclusive) in counter-clockwise order, and vertex i has a value of $f(i)$.

We say a subset of the vertices is palindromic, if their values constitute a palindrome in counter-clockwise order. More formally, let's say the subset contains k vertices $v_0, v_1, \cdots, v_{k-1}$ in counter-clockwise order. There should exist an integer d such that $0 \le d < k$, and for all $0 \le i < k$ we have $f(v_{(d+i) \bmod k}) = f(v_{(d-1-i) \bmod k})$.

Among all palindromic subsets, find the one whose convex hull has the largest size.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($3 \le n \le 500$) indicating the number of vertices of the convex polygon.

The second line contains n integers $f(1), f(2), \cdots, f(n)$ ($1 \le f(i) \le 10^9$) where $f(i)$ is the value of the i-th vertex.

For the following n lines, the i-th line contains two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$) indicating the coordinates of the i-th vertex. The vertices are listed in counter-clockwise order. The convex polygon is guaranteed to have positive size and no two vertices coincide. However there might be three vertices lying on the same line.

It's guaranteed that the sum of n of all test cases does not exceed 10^3.

Output

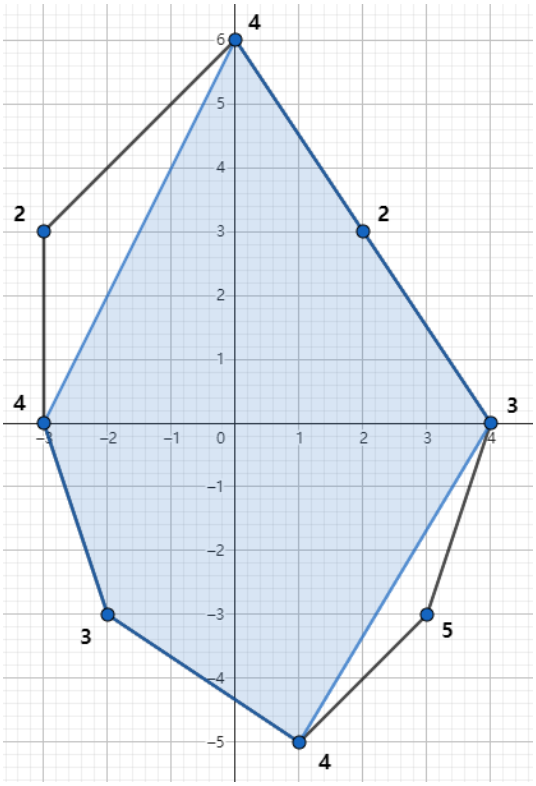
For each test case output one line containing one integer, indicating the size of the largest convex hull of a palindromic subset, multiplied by 2. It can be proven that this value is always an integer.

Example

standard input	standard output
3 8 2 4 2 4 3 4 5 3 2 3 0 6 -3 3 -3 0 -2 -3 1 -5 3 -3 4 0 3 1 2 3 0 0 1 0 0 1 3 1 1 1 0 0 1 0 0 1	84 0 1

Note

The first sample test case is illustrated below. Choose vertices 2, 4, 5, 6, 8 and consider $d = 1$, then the value sequence $\{4, 3, 4, 3, 4\}$ is a palindrome.



Problem L. Cosmic Travel

Time limit: 2 seconds
Memory limit: 1024 megabytes

BaoBao is a cosmic traveler, shuttling between an infinite number of parallel universes. Each universe is numbered with an integer starting from 0.

There are n magic apples in each universe. Although these universes have many similarities, there are still slight differences between them. The magic power of the i-th apple in the j-th universe is $a_i \oplus j$. Here $\oplus$ denotes bitwise exclusive or operation.

BaoBao is very indecisive, so he prepared q traveling plans. Each traveling plan can be described by three integers l, r and k, which means BaoBao will travel to each universe numbered from l to r (both inclusive), and collect the apple with the k-th smallest magic power among the n apples in each universe.

For each traveling plan, please help BaoBao calculate the sum of the magic power of the apples he collects. Note that the traveling plan does not really remove the apple from each universe. That is, each query is independent.

Input

There is only one test case in each test file.

The first line contains two integers n and q ($1 \le n, q \le 10^5$) indicating the number of apples in each universe and the number of plans.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($0 \le a_i < 2^{60}$).

For the following q lines, the i-th line contains three integers l_i, r_i and k_i ($0 \le l_i \le r_i < 2^{60}$, $1 \le k_i \le n$) indicating the i-th traveling plan.

Output

For each traveling plan output one line containing one integer indicating the answer for that plan. As the answer might be large, output it modulo 998244353.

Example

standard input	standard output
8 3 2 0 2 4 0 5 2 6 1 1 6 2 7 5 0 1048575 4	4 23 720895450

Problem M. The Quest for El Dorado

Time limit:	1 second
Memory limit:	1024 megabytes

There is a kingdom with n cities and m bi-directional railways connecting the cities. The i-th railway is operated by the c_i-th railway company, and the length of the railway is l_i.

You'd like to travel around the country starting from city 1. For your travel, you have bought k railway tickets. The i-th ticket can be represented by two integers a_i and b_i, meaning that if you use this ticket, you can travel through some railways in one go, as long as they're all operated by company a_i and their total length does not exceed b_i. It is also allowed to just stay in the current city when using a ticket. You can only use the tickets one at a time, and you can only use each ticket once.

As you find it a burden to determine the order to use the tickets, you decided to use the tickets just in their current order. More formally, you're going to perform k operations. In the i-th operation, you can either choose to stay in the current city u; Or choose a different city v, such that there exists a path from city u to city v where all railways in the path are operated by company a_i and the total length of railways does not exceed b_i, and finally move to city v.

For each city, determine if it is possible to arrive in that city after using all k tickets.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains three integers n, m, and k ($2 \le n \le 5 \times 10^5$, $1 \le m \le 5 \times 10^5$, $1 \le k \le 5 \times 10^5$) indicating the number of cities, the number of railways and the number of tickets.

For the following m lines, the i-th line contains four integers u_i, v_i, c_i, and l_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$, $1 \le c_i \le m$, $1 \le l_i \le 10^9$) indicating that the i-th railway connects city u_i and v_i. It is operated by company c_i and its length is l_i. Note that there might be multiple railways connecting the same pair of cities.

For the following k lines, the i-th line contains two integers a_i and b_i ($1 \le a_i \le m$, $1 \le b_i \le 10^9$) indicating that if you use the i-th ticket, you can travel through some railways in one go, if they're all operated by company a_i and their total length does not exceed b_i.

It's guaranteed that the sum of n, the sum of m, and the sum of k of all test cases will not exceed 5×10^5.

Output

For each test case, output one line containing one string $s_1 s_2 \cdots s_n$ of length n where each character is either 0 or 1. If you can travel from city 1 to city i with these k tickets, then $s_i = 1$; Otherwise $s_i = 0$.

Example

standard input	standard output
2 5 6 4 1 2 1 30 2 3 1 50 2 5 5 50 3 4 6 10 2 4 5 30 2 5 1 40 1 70 6 100 5 40 1 30 3 1 1 2 3 1 10 1 100	11011 100

Note

For the first sample test case:

- To arrive in city 4, you can use the 1-st ticket to move from city 1 to city 2, then stay in city 2 when using the 2-nd ticket, then use the 3-rd ticket to move from city 2 to city 4, then stay in city 4 when using the 4-th ticket.
- To arrive in city 5, you can use the 1-st ticket to move from city 1 to city 5 by going through the 1-st and the 6-th railway, then stay in city 5 when using the following tickets.
- As you cannot change the order to use the tickets, you cannot arrive in city 3.