Problem A. Airport Assignments

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

An airport has n_1 aircraft waiting for service, numbered from 1 to n_1, and n_2 gates arranged from left to right, numbered from 1 to n_2. There are m compatible aircraft–gate pairs. A pair (u, v) means that aircraft u can be assigned to gate v.

For every contiguous gate zone $[\ell, r]$ (where $1 \le \ell \le r \le n_2$), the airport operations team considers assignments subject to the following rules:

- all aircraft $1, 2, \ldots, n_1$ are available;
- only gates with numbers in $[\ell, r]$ may be used;
- an aircraft may be assigned only to a compatible gate.

Each aircraft can be assigned to at most one gate, and each gate can serve at most one aircraft. Let $f(\ell, r)$ be the maximum number of aircraft that can be assigned simultaneously within gate zone $[\ell, r]$.

To generate the airport's audit checksum, compute

$$\sum_{1 \le \ell \le r \le n_2} f(\ell, r) \cdot \ell \cdot r \cdot ((\ell \oplus r) + 1),$$

where $\oplus$ denotes the bitwise XOR operation.

Output the value of this checksum modulo 998 244 353.

Input

The first line contains a single integer t ($1 \le t \le 2000$), the number of test cases. For each test case:

The first line contains three integers n_1, n_2, and m ($1 \le n_1, n_2 \le 5000$, $1 \le m \le 10^4$), representing the number of aircraft, the number of gates, and the number of compatible aircraft–gate pairs.

Each of the next m lines contains two integers u and v ($1 \le u \le n_1$, $1 \le v \le n_2$), meaning that aircraft u can be assigned to gate v.

No compatible pair is listed more than once. The sum of n_1 does not exceed 5000. The sum of n_2 also does not exceed 5000.

Output

For each test case, output a single line containing one integer: the required audit checksum modulo 998 244 353.

Example

standard input	*standard output*
4 3 3 5 1 1 2 1 2 2 2 3 3 3 3 4 6 1 2 1 4 2 3 2 1 3 4 3 3 2 2 1 1 2 4 3 5 1 3 2 1 3 3 4 2 4 1	81 529 12 81

Problem B. Best Coupon Tour

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

There are n cities, numbered from 1 to n. There are also $n-1$ bidirectional roads connecting these cities, so that for every two cities x and y, there exists a path from x to y. The i-th road connects cities u_i and v_i, and has a cost c_i. Every time you pass through road i, you spend c_i coins.

You are now at city 1 and want to travel to visit all cities. Every second, if you are at city x, you can choose a city y that is directly connected with city x by a road, move to city y, and pay the cost of the respective road. Each road can be passed **at most twice**. When you have visited all cities and are at city 1, you can finish your travel. The cost of the travel is the total cost of all the moves you made.

There is also an integer k. During your travel, you can use a free coupon once, making your following k moves have no cost.

For every k from 0 to $2n-2$, you should calculate the minimum cost of a travel that visits all cities and uses the coupon optimally.

Input

The first line contains a single integer t ($1 \le t \le 400$), the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 2000$), denoting the number of cities.

The next $n-1$ lines describe the roads. The i-th of these lines contains three integers, u_i, v_i, and c_i ($1 \le u_i, v_i \le n$, $0 \le c_i \le 10^9$), denoting the cities connected by the i-th road and the cost of that road.

There is a path between any pair of cities. The sum of n does not exceed 2000.

Output

For each test case, output one line containing $2n-1$ integers: the answers for k from 0 to $2n-2$.

Example

standard input	*standard output*
2 5 1 2 2 2 3 3 2 4 1 4 5 3 7 1 2 1 1 3 1 1 4 4 3 5 5 3 6 1 6 7 4	18 15 12 10 8 5 3 2 0 32 27 22 21 17 13 12 8 4 3 2 1 0

Note

In the first test case, we can always travel along the path $1, 2, 3, 2, 4, 5, 4, 2, 1$, with the cost sequence $2, 3, 3, 1, 3, 3, 1, 2$. The cost-free segments for each k are shown below.

k	cost-free segment	answer
0	[]	18
1	[3]	15
2	[3, 3]	12
3	[2, 3, 3]	10
4	[3, 3, 1, 3]	8
5	[3, 3, 1, 3, 3]	5
6	[2, 3, 3, 1, 3, 3]	3
7	[2, 3, 3, 1, 3, 3, 1]	2
8	[2, 3, 3, 1, 3, 3, 1, 2]	0

Problem C. Clara's Clock

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

Clara bought a new digital alarm clock. The keypad of the alarm clock is shown in the figure. The alarm time is entered on the keypad in the format `HH:MM:SS`, after which the `R` button is pressed. The colons separating hours, minutes, and seconds must also be entered from the keypad, so the full sequence of keypresses consists of 9 keys: two digits of hours, a colon, two digits of minutes, a colon, two digits of seconds, and the key `R`.

1	2	3
4	5	6
7	8	9
:	0	R

A feature of the alarm clock is that any two-digit numbers from `00` to `99` can be entered as hours, minutes, and seconds. In this case, the alarm clock automatically replaces the entered number of hours with its remainder when divided by 24, and the minutes and seconds with their remainder when divided by 60. For example, typing `99:75:83` sets the same alarm time as `03:15:23`. Thus, Clara can set the same alarm time in several different ways.

The time to move a finger from one key A to another key B is equal to the Manhattan distance between the positions of these two keys in the integer coordinate system $|x_A - x_B| + |y_A - y_B|$, where the key in the top-left corner (key `1`) has coordinates $x = 1, y = 1$, and the key in the bottom-right corner (key `R`) has coordinates $x = 4, y = 3$. The keypress itself occurs instantaneously and takes 0 units of time. Typing is done with one finger, and before typing begins, Clara can place her finger over any key, so the movement time to the first pressed key is not taken into account, while the movement time to the final `R` key is included. The total typing time is the sum of the times of all finger movements.

Since Clara is tired, she does not want to spend much time setting the alarm, and now she is interested in finding the fastest way to type a given time.

Input

The only line contains the desired alarm time recorded in the standard way: hours from `00` to `23`, minutes and seconds from `00` to `59` (in the format `HH:MM:SS`).

Output

Print the fastest typing sequence found in the format `HH:MM:SS` (where HH, MM, SS are two-digit numbers from `00` to `99` entered on the keypad).

Examples

standard input	standard output
19:26:58	67:86:58
17:28:00	17:88:00
00:47:59	00:47:59

Problem D. Dealer's Foresight

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

An alien broker with extrasensory abilities has managed to predict the daily stock prices per kilogram of various valuable resources for many days ahead. Now, his goal for each resource is to choose two such days (the first for buying the resource, and the second, later one, for selling it) to maximize the profit obtained.

Write a program that, given the resource prices per kilogram over N consecutive days, determines the maximum possible profit from buying and subsequently selling one kilogram of this resource.

Input

The first line of the standard input contains the integer T — the number of test cases ($1 \le T \le 8$).

For each test case, the first line contains the number of days N ($5 \le N \le 1\,000\,000$) for which the speculator predicted the price. The second line contains N integers p_i — the prices per kilogram of the resource on the respective days ($0 \le p_i \le 2^{16}$).

Output

For each test case, in a separate line of the standard output, output a single integer — the maximum possible profit per kilogram of the resource. If it is impossible to make a profit (for example, if prices are constantly falling), output `0`.

Example

standard input	standard output
3 5 7 2 20 19 4 13 3 34 5 12 21 7 33 14 25 46 17 11 2 7 46 33 25 21 17 11 1	18 43 0

Problem E. Ethical Bridges

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

In a state consisting of N islands numbered from 1 to N, the government wants to build bridges so that one can drive from any island to any other island by car, and the construction cost is not too high. To avoid corruption schemes, the government does not want to encourage companies that suspiciously offer the lowest price. Instead, it decided to support companies that offer the second cheapest way to connect all the islands. Your task is to find this cost.

Input

The first line of the standard input contains the integer T — the number of test cases ($1 \le T \le 10$).

For each test case, the first line contains two integers — the number of islands N ($1 \le N \le 1000$) and the number of possible pairs of islands M that can be connected by a bridge ($1 \le M \le 100\,000$). Each of the following M lines contains 3 integers — the numbers of the two connected islands A_i and B_i, and the cost C_i of building a bridge between them ($1 \le A_i, B_i \le N$, $1 \le C_i \le 2^{16}$).

Output

For each test case, on a separate line of the standard output, the program should output the second cheapest cost of connecting the islands (which is strictly greater than the optimal/minimal cost).

If it is impossible to connect all the islands, the program should output `Impossible`. If all possible spanning trees have the same minimal cost (i.e., there is no second cheapest option), output `All minimal`.

Example

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

Problem F. Facing Floodlights

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

A straight road is divided into n consecutive sections, numbered from 1 to n. Directional floodlights can be installed along the road. Each floodlight is characterized by its position x (an integer between 1 and n) and the direction it faces: left or right.

All floodlights share the same beam range L, which is a non-negative integer.

- A floodlight at position x facing right illuminates all road sections with indices in the segment $[x, x + L]$.
- A floodlight at position x facing left illuminates all road sections with indices in the segment $[x-L, x]$.

Only road sections with indices between 1 and n exist; any part of a beam outside this range is ignored.

Initially, there are no floodlights. You need to process q operations of two types:

- **Install a floodlight**. An operation of the form "`1` d x" installs a new floodlight at position x.

  If $d = 0$, the floodlight faces left.

  If $d = 1$, the floodlight faces right.

  Multiple floodlights may be installed at the same position. Operations are given in chronological order, and no floodlight is ever removed.

- **Illumination query**. An operation of the form "`2` ℓ r" asks the following question:

  Consider all floodlights installed so far. Suppose they all have the same beam range L. Find the smallest integer $L \geq 0$ such that every road section with index in $[\ell, r]$ is illuminated by at least one floodlight. Floodlights outside $[\ell, r]$ may help illuminate sections inside it. If no integer L can illuminate every section in $[\ell, r]$, no matter how large L is, output -1 for this query.

Input

The first line contains two integers n and q ($1 \leq n, q \leq 3 \cdot 10^5$), representing the number of road sections and the number of operations.

Each of the next q lines describes one operation in the order in which it must be processed:

- "`1` d x": install a floodlight at position x facing direction d ($d \in \{0, 1\}$, $1 \leq x \leq n$). Here, $d = 0$ means left and $d = 1$ means right.
- "`2` ℓ r": query the minimum beam range L for the segment $[\ell, r]$ ($1 \leq \ell \leq r \leq n$).

Output

For each query of the form "`2` ℓ r", output a single line containing one integer:

- the minimum integer $L \geq 0$ such that every road section in $[\ell, r]$ is illuminated, or
- -1 if this is impossible.

Example

standard input	*standard output*
10 11 1 0 6 2 5 8 1 1 3 2 9 10 2 3 9 1 1 8 2 1 9 2 1 8 1 0 4 2 3 7 2 2 9	-1 7 6 5 5 4 4

Problem G. Gapped Copies

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

You are given a string s of length n.

For each position i where $1 \le i \le n$, let t_i be the string obtained from s by deleting the i-th character of s (and keeping the relative order of all other characters).

For two strings x and y, let $\text{lcp}(x, y)$ denote the *length* of their longest common prefix, that is, the largest integer $\ell \ge 0$ such that the first ℓ characters of x and y are equal (if $x_1 = y_1$, $x_2 = y_2$, ..., $x_\ell = y_\ell$).

Your task is to choose two **different** positions i and j such that the value $\text{lcp}(t_i, t_j)$ is as large as possible, and compute this maximum possible value.

Formally, you need to find

$$\max_{1 \le i < j \le n} \text{lcp}(t_i, t_j).$$

You do not need to output i or j, only the maximum value of $\text{lcp}(t_i, t_j)$.

Input

The first line contains a single integer t ($1 \le t \le 2 \cdot 10^5$), the number of test cases. For each test case:

A single line contains a string s that consists of lowercase English letters ($2 \le |s| \le 5 \cdot 10^5$).

The sum of $|s|$ does not exceed $5 \cdot 10^5$.

Output

For each test case, output a single integer on a separate line, representing the maximum possible value of $\text{lcp}(t_i, t_j)$ over all pairs of indices $i \ne j$.

Example

standard input	*standard output*
5 cdce eaaaa dacda cacdd zz	2 4 3 4 1

Problem H. How Many Trees?

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

You are given m depth-first search (DFS) orders of an unknown rooted tree.

We consider rooted trees with n vertices labeled from 1 to n, where vertex 1 is the root. The tree is undirected and connected, and has exactly $n - 1$ edges.

DFS definition. For a fixed rooted tree, a DFS order is obtained as follows:

- All vertices are initially unvisited.
- Start at the root 1, mark it as visited and **output** it.
- When you are at some vertex v, consider all children of v in the rooted tree. Choose its children in **any** order. For each chosen child u that is still unvisited, go to u, mark it as visited, output u, and continue the DFS from u in the same way.

For a fixed tree, different choices of the order in which the children of each vertex are processed may produce different DFS orders. All DFS orders are permutations of $1, 2, \ldots, n$ and always start with 1.

You are given m permutations of $1, 2, \ldots, n$, each starting with 1. You are told that each of them is a DFS order of the **same** rooted tree (with root 1) in the sense defined above (possibly using different choices of child orders for different DFS runs).

Your task is to determine how many different rooted trees could produce **all** of these m DFS orders.

Two rooted trees are considered different if their sets of edges are different.

Because the answer can be very large, you should output it modulo $10^9 + 7$.

Input

The first line contains two integers n and m ($1 \le n, m \le 500$).

Each of the next m lines contains a permutation $p_{k,1}, p_{k,2}, \ldots, p_{k,n}$ of the integers $1, 2, \ldots, n$, describing the k-th DFS order.

For every k, all $p_{k,i}$ are distinct and $p_{k,1} = 1$.

Output

Output a single integer: the number of rooted trees with vertex set $1, 2, \ldots, n$ and root 1 for which **every** given sequence is a valid DFS order of that tree, taken modulo $10^9 + 7$.

Examples

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

Problem I. I Will Be the Champion

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

You and your two teammates are participating in a programming contest, but due to the final exams, you all have to leave early. Each of you has a specific departure time ℓ_i (the time in minutes after the contest starts when you must leave). After the contest, you review the solutions and realize that each problem is perfectly suited for one of you: problem i is best solved by person p_i, who can complete it in c_i minutes without any wrong submissions. Unfortunately, due to the pressure of exams, you didn't perform well in the original contest.

In a dream, you see the champion team from the live broadcast: they solved all n problems with a total penalty time t. When you wake up, you discover you've been reborn on the morning of the contest day. Now, with perfect memory of all problem details and the champion's performance, you want to determine if there's a way to beat them this time.

The contest rules state:

- Only one computer is available, so problems must be solved sequentially (one after another).
- Each problem i must be assigned to person p_i, who must start solving it before their departure time ℓ_{p_i}.
- If person p_i starts solving problem i at time s, they must finish by $s + c_i \le \ell_{p_i}$.
- The total penalty time is the sum of the completion times for all solved problems.

Given n, t, and the parameters p_i and c_i for each problem, determine if there exists an order of solving the problems such that:

- All n problems are solved.
- The total penalty time is **strictly less** than t.

If such an arrangement exists, output "`YES`"; otherwise, output "`NO`".

Input

The first line contains a single integer q ($1 \le q \le 10^5$), the number of test cases. For each test case:

The first line contains four integers: n, ℓ_1, ℓ_2, and ℓ_3 ($1 \le n \le 10^5$, $1 \le \ell_1, \ell_2, \ell_3 \le 3 \cdot 10^7$), representing the number of problems and the departure times (in minutes after contest start) of the three teammates.

Each of the following n lines contains two integers, p_i and c_i ($1 \le p_i \le 3$, $1 \le c_i \le 300$), where p_i denotes the designated teammate for problem i, and c_i is the required time (in minutes) for them to complete it.

The last line contains a single integer t ($1 \le t \le 10^{13}$), representing the total penalty time of the champion team.

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

Output

For each test case, if there exists an order of solving problems that satisfies the constraints (all problems solved, penalty strictly less than t), output "`YES`"; otherwise, output "`NO`".

Examples

standard input	*standard output*
2 3 100 150 175 1 100 2 25 3 50 401 5 100 200 300 1 30 1 30 1 40 2 110 3 50 1275	YES NO
1 1 100 300 300 1 300 300	NO

Problem J. Just the Right Shift

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

You are given a sequence of n integers: $a_1, a_2, \ldots, a_n$.

Consider all n possible cyclic shifts (rotations) of this sequence. For a starting position p ($1 \le p \le n$), the corresponding cyclic shift is the sequence

$$b_1, b_2, \ldots, b_n = a_p, a_{p+1}, \ldots, a_n, a_1, a_2, \ldots, a_{p-1}.$$

For this shifted sequence, define its **prefix maximum sequence** $c_1, c_2, \ldots, c_n$ as

$$c_i = \max(b_1, b_2, \ldots, b_i) \text{ for all } i = 1, 2, \ldots, n.$$

We compare two sequences of the same length by the usual lexicographical order: c is lexicographically smaller than d if and only if there exists an index i such that $c_1 = d_1$, $c_2 = d_2$, ..., $c_{i-1} = d_{i-1}$, and $c_i < d_i$.

Your task is to choose a cyclic shift of the original sequence such that its prefix maximum sequence is the lexicographically smallest among all n possible cyclic shifts.

Input

The first line contains a single integer t ($1 \le t \le 2 \cdot 10^5$), the number of test cases. For each test case:

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

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

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

Output

For each test case, output one line containing n integers: the lexicographically smallest prefix maximum sequence after a cyclic shift.

Example

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

Problem K. Kid's Games

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

During a physical education class, school students are playing the following game.

The game field is represented as a coordinate plane. Each of the n students stands at some point with integer coordinates: the i-th student chooses the point (x_i, y_i). Note that multiple students can end up at the exact same point.

Once everyone is in place, the coach takes out a long rope, chooses three students standing at three distinct, non-collinear points, and asks them to stretch the rope between them, thereby enclosing a triangular area of the field. Everyone who ends up inside the triangle or on its boundary (including the three students holding the rope) is declared a winner, while the others are considered losers.

The coach stretches the rope in such a way that the number of winners is maximized. When doing so, he counts students standing at the same point individually: the count of students matters to him, not the number of occupied points.

There may be multiple ways to stretch the rope to achieve the maximum number of winners, and it is not known in advance which one the coach will choose. For each student, determine whether they can end up among the winners — that is, whether there exists a triangle with the maximum possible number of winners that contains the point chosen by this student inside or on its boundary.

Input

The first line of standard input contains an integer T — the number of test cases ($1 \le T \le 10$).

For each test case, the first line contains the number of points N ($3 \le N \le 100$). The i-th of the following N lines contains two integer coordinates x and y — the coordinates of the i-th student's point ($0 \le x, y \le 10^5$). It is guaranteed that there are at least three non-collinear points in the set.

Output

For each test case, output on a separate line of standard output the number of participants who have a chance to win the game.

Example

standard input	standard output
3 3 0 0 0 100000 100000 0 4 0 0 0 1 1 0 0 0 5 2 2 5 0 0 0 5 5 0 5	3 4 5