2025 NTU ICPC Team Preliminary

National Taiwan University

2025/08/09

Language	Version	Compile Flags	Extensions
C	gcc 12.2.0	`-O2 -std=c11 -static -lm`	`.c`
C++	g++ 12.2.0	`-O2 -std=c++20 -static`	`.cc, .cpp`
Python3	3.11.2	`None`	`.py`

Problem	Problem Name	Time Limit	Memory Limit
A	ABABABABA	2 s	2048 MB
B	BaCoder Testing Procedure	1 s	2048 MB
C	Conveyors	1 s	2048 MB
D	Disregard the Light	2 s	2048 MB
E	Educational Problem	1 s	2048 MB
F	Fair Gambling	1 s	2048 MB
G	Grid Game	1 s	2048 MB
H	Harmony Graph	1.5 s	2048 MB
I	Imprisoned XII	1 s	2048 MB
J	Journey	1 s	2048 MB
K	Kindergarten Problem	2 s	2048 MB
L	Limited Rooks	2 s	2048 MB
M	Minimax Spanning Tree	2 s	2048 MB

This page is intentionally left blank.

A. ABABABABA

Problem ID: ababa

For any positive integer k, we call a string s a *k-ababa string* if there exist two **non-empty** strings a and b such that:

$$s = \underbrace{(a+b)+(a+b)+\ldots+(a+b)}_{k}+a,$$

where $+$ denotes string concatenation. For example, the string aabaabaa is a 2-*ababa string*, since we can choose $a =$ aa and $b =$ b. Strings a and b can be the same.

You are given a string s of length N. For every integer k from 1 to N, output the length of the longest substring of s that is a *k-ababa string*.

Input

The first line contains an integer T — the number of test cases.

The first line of each test case contains an integer N — the length of the string s.

The second line contains a string s of length N, consisting of lowercase letters.

- $1 \leq T \leq 10^5$
- $1 \leq N \leq 3 \cdot 10^5$
- It is guaranteed that the sum of N over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, output a single line with N integers. The k-th integer is the length of the longest *k-ababa string* that is a substring of s, or 0 if none exists.

Sample Input 1	Sample Output 1
4 9 abcabcabc 11 mississippi 12 qwqwqwqwqwqq 18 ntuicpcpreliminary	9 8 0 0 0 0 0 0 0 10 7 0 0 0 0 0 0 0 0 0 12 11 7 9 11 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Note

In the first test case of the sample input, `abcabcabc` is a 1-*ababa string* and `bcabcabc` is a 2-*ababa string*.

In the second test case of the sample input, `ississippi` is a 1-*ababa string* and `ississi` is a 2-*ababa string*.

B. BaCoder Testing Procedure

Problem ID: bacoder

After founding his competitive programming company BaCoder, baluteshih saw rapid business growth and soon expanded into various services. One of the most important services offered is problem setting for programming contests — especially for clients who want to host a contest but lack the ability to write good problems themselves.

This job isn't easy. Besides ensuring that the problems are interesting and meet the client's desired difficulty, every detail must be carefully checked. To guarantee quality, BaCoder has a strict internal problem verification workflow. For a contest with N problems, each problem has:

- One setter responsible for preparing everything (e.g., test data, checker, editorial),
- Two testers who review the setter's work and provide suggestions. One is designated the primary tester, and the other the secondary tester.

When assigning roles, suppose there are M people participating in this contest preparation, labeled from 1 to M. First, the setter for each problem is decided. Then, if someone is the setter for x problems, they must also serve as the primary tester for x problems and secondary tester for another x problems.

Assigning testers is tricky — doing it manually often leads to mistakes like assigning the same person as both testers, or assigning the setter to test their own problem. baluteshih finds this frustrating. To improve efficiency, he asks you to write a program that assigns testers such that:

- For each problem, the setter and the two testers are three distinct individuals,
- The tester assignment satisfies the workload requirements described above.

Input

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

For each test case, the first line contains two integers N and M: the number of problems and the number of participants.

The second line contains N integers $s_1, s_2, \ldots, s_N$, where s_i is the ID of the setter for the i-th problem.

- $1 \leq T \leq 10^5$
- $1 \leq M \leq N \leq 10^6$
- $1 \leq s_i \leq M$
- It is guaranteed that each of the M people is the setter for at least one problem.
- It is guaranteed that the sum of N over all test cases does not exceed 10^6.

Output

For each test case, if no valid tester assignment exists, output a single line containing `No`.

Otherwise, output three lines:

- The first line should be `Yes`.
- The second line should contain N integers $t_{1,1}, t_{1,2}, \ldots, t_{1,N}$, where $t_{1,i}$ is the primary tester for the i-th problem.
- The third line should contain N integers $t_{2,1}, t_{2,2}, \ldots, t_{2,N}$, where $t_{2,i}$ is the secondary tester for the i-th problem.

Your output will be considered correct if it satisfies all of the following conditions:

- $1 \leq t_{1,i}, t_{2,i} \leq M$ for all i.
- For each person y ($1 \leq y \leq M$), if y is the setter for exactly x problems, then y must also appear exactly x times as a primary tester and exactly x times as a secondary tester.
- For every problem i, the three people s_i, $t_{1,i}$, and $t_{2,i}$ must be distinct.

Sample Input 1	Sample Output 1
4 5 5 1 2 3 4 5 6 3 1 1 2 2 3 3 10 5 3 1 4 1 5 2 4 5 3 1 3 1 1 1 1	Yes 2 3 4 5 1 3 4 5 1 2 Yes 2 2 3 3 1 1 3 3 1 1 2 2 Yes 4 3 5 3 1 1 2 1 5 4 2 4 1 5 4 3 1 3 1 5 No

C. Conveyors

Problem ID: conveyor

You are given a container represented as an $N \times M$ grid. Each cell in the grid contains a type of cargo, where the cargo in row i and column j is denoted by $a_{i,j}$. Rows are numbered from top to bottom $1, 2, \ldots, N$; columns from left to right $1, 2, \ldots, M$. Each row is equipped with a conveyor belt. When you activate the conveyor belt in row i, all the cargo in that row shifts one cell to the right, with the rightmost cargo wrapping around to the leftmost cell. For example, if the row initially contains $[1, 2, 3, 4]$, after one shift it becomes $[4, 1, 2, 3]$.

In addition, there is an empty cell located above the first cell of the first column (i.e., just above $a_{1,1}$). You can move this empty cell by swapping it with the cargo directly above or below it, as long as such a cell exists. Specifically:

- If there is a cell below the empty cell, you may swap the empty cell with the cargo below it.
- If there is a cell above the empty cell, you may swap the empty cell with the cargo above it.

Your goal is to perform a sequence of operations to arrange the container into a "sorted" state. A container is considered sorted if it satisfies both of the following conditions:

- The empty cell is back at the topmost position of the first column.
- The cargo in row i and column j is exactly $(i - 1) \cdot M + j$.

Determine whether it's possible to reach the sorted state. If it is, output a sequence of operations to achieve it. Otherwise, state that it's impossible.

Input

The first line of the input contains two integers N and M, the number of rows and the number of columns.

Each of the next N lines describes one row of the container. The i-th of these lines contains M integers $a_{i,1}, a_{i,2}, \ldots, a_{i,M}$, the initial contents of row i.

- $1 \leq N, M \leq 10$
- $1 \leq a_{i,j} \leq N \cdot M$
- All $a_{i,j}$ are distinct.

Output

If the goal is not achievable, output a single line containing -1.

Otherwise, on the first line outputs an integer K, the number of operations you need to perform.

On the second line outputs K integers $b_1, b_2, \ldots, b_K$, where the i-th integer denotes the i-th operation:

- If $b_i = -2$, swap the empty cell with the one **below** it.
- If $b_i = -1$, swap the empty cell with the one **above** it.
- If $1 \leq b_i \leq N$, activate the conveyor in row b_i.

Your answer will be considered correct if:

- $0 \leq K \leq 5 \cdot 10^5$
- $\forall 1 \leq i \leq K, b_i = -1 \vee b_i = -2 \vee 1 \leq b_i \leq N$
- After performing all the operations, the container is sorted.

Note that you do **not** have to minimize the number of operations.

Sample Input 1	Sample Output 1
2 2 1 3 4 2	6 -2 1 -2 2 -1 -1

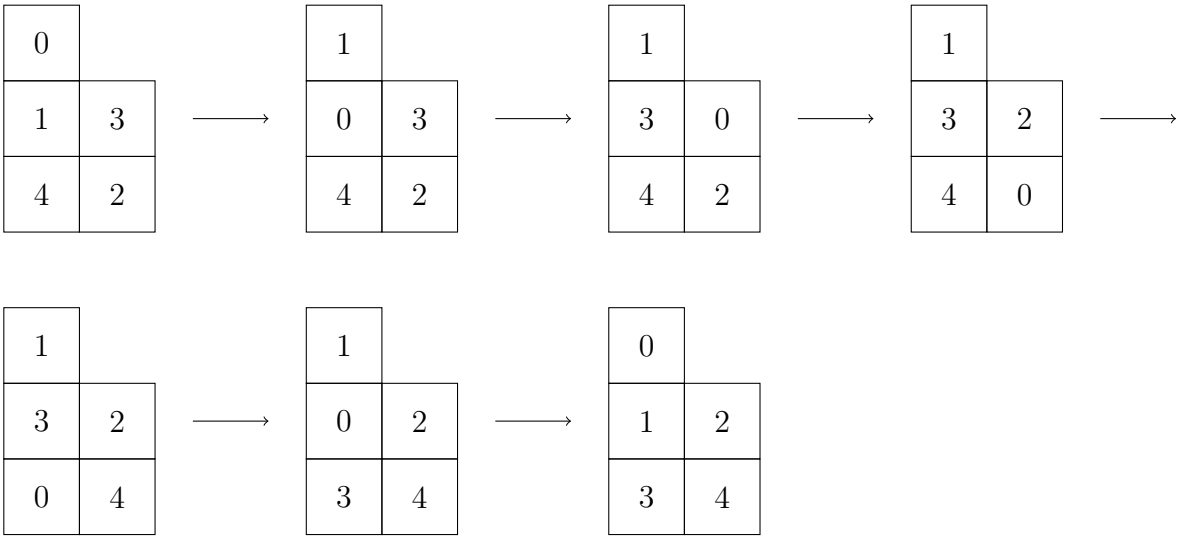
Sample Input 2	Sample Output 2
2 3 1 2 3 4 5 6	5 1 1 1 -2 -1

Sample Input 3	Sample Output 3
2 3 1 2 3 4 6 5	-1

Sample Input 4	Sample Output 4
3 3 1 2 3 9 4 6 5 7 8	13 -2 1 1 -2 2 -2 3 -1 2 -1 1 -1 3

Note

Below picture shows how to perform a sequence of operations to make the first sample sorted. Here, we use 0 to represent the empty cell.



This page is intentionally left blank.

D. Disregard the Light

Problem ID: light

Wiwi owns an enormous rectangular watermelon field whose sides are parallel to the axes. In the plane, its lower-left corner is at $(-10^{100}, -10^{100})$ and its upper-right corner is at $(10^{100}, 10^{100})$. To protect her crop from thieves, she has installed N lamps at specified points inside the field. Each lamp shines light in all directions, and any point illuminated by at least one lamp is recorded by a central computer. There are no obstacles on the field, so light rays aren't blocked.

Two days later, Wiwi discovered that some watermelons had been stolen, but the recording showed no trespassers.

"Impossible... how did they manage to disregard all the light?"

Further investigation revealed that the thieves hack into the power grid, cut the electricity for a few seconds, and in that blackout deploy a large mirror. The mirror then casts a shadow behind it, so the thieves can steal watermelons without being seen. Fearing they can't carry too many at once, the thieves choose a target segment $\overline{CD}$ in advance and only steal melons located on that segment that lie in shadow.

More precisely, the mirror could be represented by a segment $\overline{AB}$. A watermelon at point P is stolen if and only if:

- P lies on the target segment $\overline{CD}$.
- For **every** lamp L, the segment $\overline{PL}$ intersects $\overline{AB}$. Note that endpoints count as intersections.

Because Wiwi was out of funds after installing the lamps, she can't stop the thief's trick even now that she knows it. Still, she wants to estimate her losses. Wiwi runs Q simulations of possible thefts; for each one, she needs to compute the total **length** of the portion of segment $\overline{CD}$ that the thief could actually steal. Can you help her?

Input

The first line contains two integers N and Q, the number of lamps and the number of theft simulations.

Each of the next N lines contains two integers x_i, y_i, the coordinates of the i-th lamp.

Each of the following Q lines contains eight integers $x_A, y_A, x_B, y_B, x_C, y_C, x_D, y_D$, describing one theft simulation, where points (x_A, y_A) and (x_B, y_B) are endpoints of the mirror, and points (x_C, y_C) and (x_D, y_D) are endpoints of the target segment.

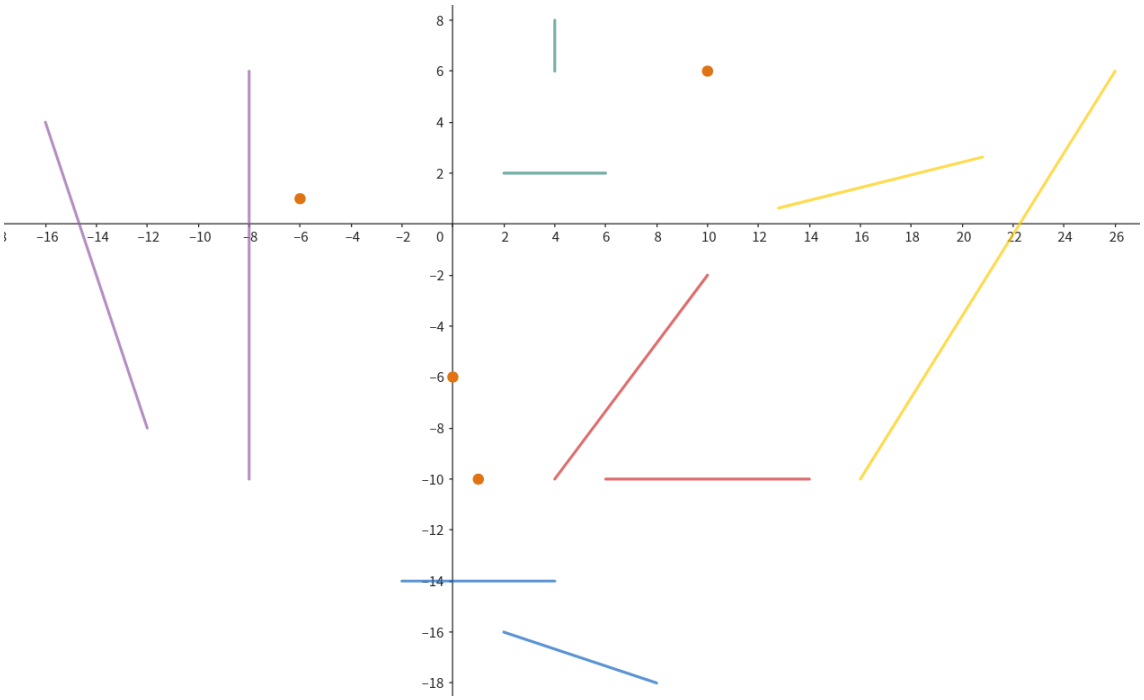
- $1 \leq N, Q \leq 2 \cdot 10^5$
- The absoulate value of all the coordinates are at most 10^9.
- No two lamps share the same position.
- No lamp lies exactly on the mirror in any simulation.
- No lamp lies exactly on the target segment in any simulation.
- The mirror segment and the target segment do not intersect.
- For any theft simulation, $(x_A, y_A) \neq (x_B, y_B)$ and $(x_C, y_C) \neq (x_D, y_D)$.

Output

Output Q lines. The i-th line should contain the answer to the i-th theft simulation. Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}. Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \leq 10^{-6}$.

Sample Input 1	Sample Output 1
4 5 -6 1 0 -6 1 -10 10 6 -2 -14 4 -14 2 -16 8 -18 4 -10 10 -2 6 -10 14 -10 2 2 6 2 4 8 4 6 -8 6 -8 -10 -12 -8 -16 4 12 2 20 4 16 -10 26 6	1.341572340677494 4.000000000000000 0.000000000000000 12.64911064067352 0.000000000000000

Note


8
6
4
2
−2
−4
−6
−8
−10
−12
−14
−16
−18
−16
−14
−12
−10
−8
−6
−4
−2
0
2
4
6
8
10
12
14
16
18
20
22
24
26

This page is intentionally left blank.

E. Educational Problem

Problem ID: education

Given a directed graph with N vertices and M edges, compute the maximum flow from the source vertex 1 to the sink vertex N.

You are not a baby, you know what maximum flow is.

Input

The first line contains two integers N and M — the number of vertices and edges, respectively.

Each of the next M lines contains three integers u_i, v_i, and c_i, describing a directed edge from vertex u_i to vertex v_i with capacity c_i.

- $2 \leq N \leq 200$
- $0 \leq M \leq N \cdot (N-1)$
- $1 \leq u_i, v_i \leq N$
- $1 \leq c_i \leq 10^9$
- The graph contains no self-loops and no multiple edges.

Output

Print an integer on the first line — the maximum flow value from vertex 1 to vertex N.

On the second line, print an integer p — the number of flow paths used in your decomposition.

Then, print p lines, each describing one flow path in the following format:

- Suppose the i-th path sends x_i units of flow from vertex 1 to vertex N, following the path $1 = v_1, v_2, \ldots, v_{k_i} = N$ (the path can be non-simple).
- Output $k_i + 2$ integers: first, the number x_i (the amount of flow sent along the path), followed by k_i (the length of the path), and then the k_i vertices $v_1, v_2, \ldots, v_{k_i}$.

Multiple valid outputs may exist. Your output will be considered correct if it satisfies all of the following conditions:

- $0 \leq p \leq 5 \cdot 10^4$.
- $\sum_{i=1}^{p} x_i$ equals the maximum flow value.
- For each edge (u_i, v_i) with capacity c_i, the total flow across all paths using this edge does not exceed c_i.
- Every path must use only edges present in the input, and must start at vertex 1 and end at vertex N.

Sample Input 1

```
4 5
1 2 4
2 3 2
3 4 3
1 3 1
2 4 1
```

Sample Output 1

```
4
3
1 3 1 2 4
1 3 1 3 4
2 4 1 2 3 4
```

Sample Input 2

```
4 2
1 2 10
3 4 10
```

Sample Output 2

```
0
0
```

F. Fair Gambling

Problem ID: gamble

As a student at National Taiwan University (NTU), you are faced with countless gambles — both big and small — even before you officially enroll. The very process of getting into NTU is a gamble in itself. No matter which admission route you take, there's no guarantee you won't have bad luck and encounter questions that don't play to your strengths, or underperform in a crucial exam or competition.

After entering the university, you'll continue to face gambles. For example, course selection is like a strategic game. Whether you can successfully enroll in the courses you want, whether a course is *cold* or *sweet*, and whether the grading is fair — these are all uncertainties you must carefully navigate.

Although these gambles are games with incomplete information and a lot of randomness, there are still ways to minimize your risks and losses. The best strategy is to gather as much information as possible and use it wisely in your decision-making. For instance, before course selection, you can research course reviews to avoid those with heavy workloads and stingy grading or ones where no one seems to understand what's going on. You can also investigate how much effort each course typically demands, allowing you to manage your semester in a way that's not too stressful, yields good grades, and still helps you learn something meaningful.

To that end, students have developed a unique set of terms for evaluating courses, such as:

- Sweet: The course gives out generous grades; most people can earn high marks.
- Cold: The course doesn't require much effort.
- Hard: The course requires a lot of effort or is very difficult — for example, heavy assignments, tough exams, and other demanding aspects make a course considered hard.
- Solid-sweet: Although getting a high grade isn't exactly easy, putting in sufficient effort almost guarantees a good result.

There can be disagreements when evaluating whether a course is truly sweet or cold, since each person has different expectations — some are satisfied just by passing, while others won't settle for less than an A+. Everyone has different abilities too, so the effort required and the difficulty of earning good grades can vary widely.

When collecting course information, it's also possible to receive misinformation — that's all part of the gamble too! You might come across outdated reviews from years ago, and the course

may have changed significantly since then. Or perhaps the reviewer was exceptionally talented and breezed through every course with ease, leading you to mistakenly believe a brutally difficult course is both cold and sweet, only to suffer through it yourself.

For example, if someone tells you that the course *Programming Techniques* is cold and sweet, you might want to be skeptical.

To avoid being misled, you carefully study the rules and workload of Programming Techniques and discover something called a summer assignment. "Oh no, I haven't done a summer assignment since junior high school," some might think. Wanting to protect your well-earned summer vacation, you manage to acquire insider information that the summer assignment contains N problems, and for each problem, it's sufficient if any one of your three team members solves it. This means you can strategically distribute the workload to ensure everyone gets to enjoy their summer.

However, things aren't that simple. Just like how the sweetness or coldness of a course can feel different depending on the person, for your team of three, the effort required to solve each problem also varies by person. The effort required for the i-th person to solve the j-th problem is denoted as $c_{i,j}$, and the total effort that the i-th person can put into the summer assignment cannot exceed p_i — otherwise, they'll find the course too hard and quit the team.

Fortunately, you don't have to worry too much. The teaching assistants are kind, so you don't need to solve all the problems in the assignment to get full marks.[1] However, the one piece of information you can't obtain is how many problems need to be solved to earn full credit. Until the assignment is officially released, this number remains unknown.

Therefore, you want to compute in advance: What is the maximum number of problems your team can solve, given each person's effort limit?

Note: Each problem only needs to be solved by one team member to count as completed. Even if two people solve the same problem, it only counts once. Since the majority of the effort for a problem falls on the person solving it, you can assume that each problem your team wants to solve is assigned to exactly one person. If the i-th person is assigned the j-th problem, they must spend $c_{i,j}$ effort to complete it.

Input

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

[1] This is not guaranteed in the real world.

For each test case, the first line contains four integers: N, p_1, p_2, and p_3, representing the number of problems in the summer assignment and the effort limits of the three team members.

The next three lines each contain N integers. The i-th of these lines contains $c_{i,1}, c_{i,2}, \ldots, c_{i,N}$, representing the amount of effort the i-th person needs to spend to solve each of the N problems.

- $1 \leq T \leq 200$
- $1 \leq N \leq 200$
- $1 \leq p_1, p_2, p_3 \leq 500$
- $1 \leq c_{i,j} \leq 500$
- It is guaranteed that the sum of N over all test cases does not exceed 200.

Output

For each test case, output a single line containing one integer representing the maximum number of problems your team can solve.

Sample Input 1	Sample Output 1
2 5 4 9 3 1 3 2 5 4 3 2 4 5 1 1 4 3 2 3 5 4 4 1 1 3 2 5 4 3 2 4 5 1 1 4 3 2 3	5 4

This page is intentionally left blank.

G. Grid Game

Problem ID: game

Alice and Bob are playing a game on a grid. The grid has 10^9 rows and 10^9 columns. Rows are numbered from top to bottom $1, 2, \ldots, 10^9$; columns from left to right $1, 2, \ldots, 10^9$. There are N cookies placed on the grid. The i-th cookie is at row r_i and column c_i. Multiple cookies may occupy the same cell.

Alice and Bob take turns moving cookies, with Alice going first. On Alice's turn, she chooses one cookie and moves it one cell upward. If this move would take the cookie off the top edge of the grid, she eats that cookie. On Bob's turn, he chooses one cookie and moves it one cell to the left. If this move would take the cookie off the left edge of the grid, he eats that cookie as well.

The player who eats the last remaining cookie wins. Assuming both players play optimally, determine who will win.

Input

The first line contains an integer T — the number of test cases.

The first line of each test case contains an integer N — the number of cookies.

The next N ines each contain two positive integers r_i and c_i — the row and column of the i-th cookie.

- $1 \leq T \leq 10^5$
- $1 \leq N \leq 3 \cdot 10^5$
- $1 \leq r_i, c_i \leq 10^9$
- It is guaranteed that the sum of N over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, if Alice wins, output `Alice`, otherwise output `Bob`.

Sample Input 1	Sample Output 1
2 1 3 5 2 1 1 1 1	Alice Bob

H. Harmony Graph

Problem ID: harmony

Do you know what an Eulerian circuit is? An Eulerian circuit is a (possibly non-simple) cycle that uses each edge exactly once. It can be defined for both undirected and directed graphs. In this problem, we focus on the directed case.

We define a graph as harmony if:

- It contains **exactly one** Eulerian circuit.

Here, two Eulerian circuits are considered the same if they are identical up to rotation. In particular, a graph with no edges is also considered harmony. See the following figure for example.

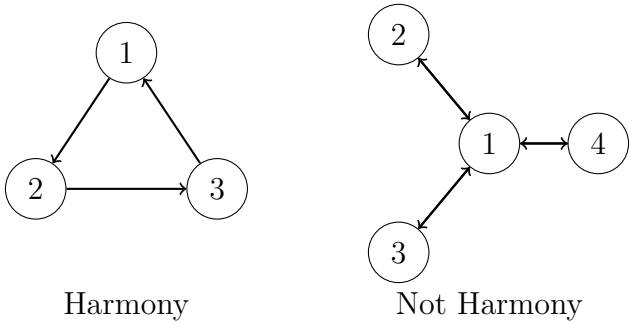


Figure H.1: Left: A harmony graph with exactly one Eulerian circuit. Right: A non-harmony graph with multiple Eulerian circuits.

You are given a directed graph with weighted edges. Your task is to find the harmony subgraph (a subgraph that is harmony) with the maximum total edge weight. A harmony subgraph with no edges is considered to have weight zero.

Input

The first line contains two integers N and M — the number of vertices and the number of directed edges.

Each of the next M lines contains three integers u_i, v_i, and w_i — representing a directed edge from node u_i to node v_i with weight w_i.

- $1 \leq N \leq 14$
- $0 \leq M \leq N \cdot (N-1)$
- $1 \leq u_i, v_i \leq N$
- $-10^7 \leq w_i \leq 10^7$
- The graph contains no self-loops and no multiple edges.

Output

Output a single integer — the maximum total edge weight among all harmony subgraphs of the given graph.

Sample Input 1	Sample Output 1
4 6 1 2 1 2 1 2 1 3 3 3 1 4 1 4 5 4 1 6	18

Sample Input 2	Sample Output 2
3 3 1 2 -1 2 3 -1 3 1 -1	0

I. Imprisoned XII

Problem ID: counting

Given two integers N and K, count the number of pairs of **non-decreasing** sequences (a, b), each of length N, that satisfy:

- $a_1 = b_1 = 0$
- $\forall 2 \le i \le N, a_i - a_{i-1} \le 1$
- $\forall 2 \le i \le N, b_i - b_{i-1} \le 1$
- $\forall 1 \le i \le N, a_i - b_i \le K$

Since the number may be huge, output the result modulo 998244353.

Input

The input only contains one line with two integers N, K.

- $1 \le N \le 10^6$
- $0 \le K \le N$

Output

Output a single line with one integer — the number of pairs of non-decreasing sequences (a, b) that satisfy the conditions, modulo 998244353.

Sample Input 1	Sample Output 1
4 1	56

Sample Input 2	Sample Output 2
8 3	15808

Sample Input 3	Sample Output 3
56562 48763	444024476

This page is intentionally left blank.

J. Journey

Problem ID: journey

There are N sightseeing spots numbered from 1 to N. The height of the i-th spot is h_i, and heights are strictly increasing. That is, $h_1 < h_2 < \ldots < h_N$.

Every pair of sightseeing spots is connected by a cable car, but to reduce costs, each cable car only runs from a lower spot to a higher one. That is, for every pair $i < j$, there is a cable car from spot i to spot j, and it takes exactly $h_j - h_i$ units of time to travel.

However, when there are multiple cable cars departing from the same spot, you **cannot** freely choose which one to take. Specifically, from spot i, there are $N - i$ possible outgoing cable cars, going to spots $i + 1, i + 2, \ldots, N$. At time t, the only cable car available is the one going to spot $i + 1 + (t \bmod (N - i))$.

For example, if $N = 6$ and you are at spot 3, the available destination changes as follows:

- At time 0, the cable goes to spot 4.
- At time 1, it goes to spot 5.
- At time 2, it goes to spot 6.
- At time 3, it goes to spot 4 again, and so on.

You may choose to wait at a spot for any amount of time if the current destination is not the one you want.

You are currently at spot 1 and it is now time T. You want to travel to spot N using the cable cars. What is the minimum amount of time needed to reach your destination? You may assume that switching cable cars takes no time.

Input

The first line of the input contains two integers N, T.

The second line of the input contains N integers $h_1, h_2, \ldots, h_N$.

- $2 \leq N \leq 1000$
- $0 \leq T \leq 10^9$
- $0 \leq h_1 < h_2 < \ldots < h_N \leq 10^9$

Output

Output a single line with one integer: the minimum amount of time needed to reach spot N from spot 1.

Sample Input 1	Sample Output 1
3 1 2 3 5	3

K. Kindergarten Problem

Problem ID: kindergarten

As a student in Taiwan, you might have experienced situations like these:

- When you were in kindergarten, you asked a question, but your teacher said, "You'll learn that in elementary school."
- When you got to elementary school and asked the same question again, your teacher replied, "You'll learn that in junior high."
- When you finally reached junior high, you still hadn't forgotten the question that had been bothering you all this time. But your teacher said, "That's not part of the Comprehensive Assessment Program. You'll learn it if you get into a top senior high school."
- Once you entered senior high school, you already knew what was coming. Your teacher said, "That's something you'll study in university."
- Then you got into NTU, thinking you'd finally get your answer. But your professor said, "Come on! You learned that back in kindergarten."

That's when you realized the problem: you went to the wrong kindergarten.

To prevent others from making the same mistake, you decided to start your own kindergarten, where teachers are committed to giving students the best education possible, and always do their best to answer every question.

In the playground of your kindergarten, there are N pillars lined up from left to right, numbered 1 through N. The i-th pillar has height h_i, and all pillars have distinct heights. Thanks to your education system, your students are not only full of knowledge, but also physically strong. Their favorite game is jumping between pillars.

For any two pillars i and j, if every pillar between them is shorter than both i and j, then a student can jump between i and j in one move, in either direction.

The students have invented many game rules, but the most popular one goes like this: the jury announces two pillars, s and t, and participants must reach t starting from s using valid jumps. There's one special rule: players can only move in the direction from s to t. That is, if $s < t$, a student can only jump to pillars with larger indices than their current position; if $s > t$, only to pillars with smaller indices.

Let $f(s, t)$ denote the minimum number of moves required to reach t from s under these rules.

This morning, the kids want to select a range of pillars with consecutive indices as their new competition arena. They define the *quality* of an interval $[\ell, r]$ as:

$$\sum_{\ell \le s < t \le r} f(s, t)$$

They have Q candidate intervals, but they don't know how to compute the quality efficiently, so they're asking for your help.

Input

The first line contains two integers N and Q, representing the number of pillars and the number of candidate intervals, respectively.

The second line contains N integers $h_1, h_2, \ldots, h_N$, where h_i denotes the height of the i-th pillar.

Each of the next Q lines contains two integers ℓ_i and r_i, indicating that the i-th candidate interval is $[\ell_i, r_i]$.

- $2 \le N \le 5 \cdot 10^5$
- $1 \le Q \le 5 \cdot 10^5$
- $1 \le h_i \le N$
- $h_i \ne h_j$ for all $i \ne j$
- $1 \le \ell_i < r_i \le N$

Output

Output Q lines. On the i-th line, output a single integer representing the quality of the i-th candidate interval.

Sample Input 1	Sample Output 1
10 5 4 5 1 3 2 10 8 9 6 7 1 10 2 5 3 8 9 10 5 6	95 8 25 1 1

This page is intentionally left blank.

L. Limited Rooks

Problem ID: rook

You are given an infinite chessboard with N obstacles placed on it. We call a rook is **limited** if it can move to only a finite number of squares. Your task is to count how many empty squares on the board will result in a limited rook if you place one there. Note that you cannot place a rook on a square that occupied by an obstacle.

A rook moves any number of squares in the same row or column, but cannot move to or jump over the obstacles. More precisely, a rook placed at square (r, c) can move to a square (r', c') only if one of the conditions is satisfied:

- $r' = r$, $c' > c$ and all squares (r, y) with $c < y \leq c'$ contain no obstacles.
- $r' = r$, $c' < c$ and all squares (r, y) with $c' \leq y < c$ contain no obstacles.
- $c' = c$, $r' > r$ and all squares (x, c) with $r < x \leq r'$ contain no obstacles.
- $c' = c$, $r' < r$ and all squares (x, c) with $r' \leq x < r$ contain no obstacles.

Input

The first line of the input contains an integer N, the number of obstacles.

Each of the next N line contains two integers r_i, c_i, indicating that the i-th obstacle is placed at (r_i, c_i).

- $1 \leq N \leq 5 \cdot 10^5$
- $0 \leq r_i, c_i \leq 10^9$
- No two obstacles share the same square.

Output

Output a single line with one integer represents the number of squares on the board will result in a limited rook if you place one there.

Sample Input 1	Sample Output 1
6 1 0 1 5 0 1 0 2 48763 2 56562 1	2

Sample Input 2	Sample Output 2
15 3 5 4 1 5 4 0 5 1 2 0 4 2 0 4 5 3 3 2 3 0 2 0 3 0 1 1 1 3 2	4

M. Minimax Spanning Tree

Problem ID: mst

Doludu and baluteshih are playing a game. The rules are simple: there is a connected undirected graph G with N vertices and M edges. Each edge is painted with one of K different colors. Doludu will assign a **unique** integer weight from 1 to K to each color. This assignment then determines the weight of each edge: the weight of an edge is equal to the weight of its color.

Once Doludu finishes assigning weights to the colors, baluteshih will choose a spanning tree $\mathcal{T}$ of G. His score is the total weight of the edges in $\mathcal{T}$. As with most games, baluteshih wants to maximize his score, while Doludu wants to minimize it.

However, baluteshih finds this game incredibly boring. Not only does Doludu's color-weight assignment determine the maximum possible score he can achieve, but finding the maximum spanning tree offers no challenge for him. After playing once, he got tired of it.

Still, he recorded the game they played: which edges existed in G, what color each edge was, and which edges were selected in his chosen spanning tree $\mathcal{T}$. A month later, he showed this record to Doludu and said, "Hey, this is the game we played a month ago!"

But baluteshih didn't write down the exact weights Doludu assigned to each color, so Doludu became suspicious and thought baluteshih might be lying.

Your task is to help baluteshih construct a set of color weights that would make Doludu believe the record is legitimate.

Doludu will consider the record legitimate if and only if the given spanning tree $\mathcal{T}$ is a maximum spanning tree under the edge weights derived from the color weights. Doludu knows baluteshih is smart and certainly picked the best possible tree after seeing the weights, but Doludu also remembers that he chose the weights arbitrarily, so **Doludu might not play optimally**.

Input

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

Each test case starts with a line of three integers: N, M, and K — the number of vertices, edges, and colors in the graph G.

Then follow M lines, each containing four integers u_i, v_i, c_i, and t_i, representing that the i-th

edge connects vertices u_i, v_i, is painted with color c_i, and is part of the spanning tree $\mathcal{T}$ if $t_i = 1$. If $t_i = 0$, that means the i-th edge is not in $\mathcal{T}$.

- $1 \leq T \leq 2.5 \cdot 10^5$
- $2 \leq N \leq 5 \cdot 10^5$
- $N - 1 \leq M \leq 5 \cdot 10^5$
- $1 \leq K \leq M$
- $1 \leq u_i, v_i \leq N$
- $1 \leq c_i \leq K$
- $t_i \in \{0, 1\}$
- G is guaranteed to be connected.
- The graph G may contain multiple edges, but no self-loops.
- The edges with $t_i = 1$ form a spanning tree of G.
- It is guaranteed that a valid solution exists.
- It is guaranteed that the sum of N over all test cases does not exceed $5 \cdot 10^5$.
- It is guaranteed that the sum of M over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, output a single line with K integers: $w_1, w_2, \ldots, w_K$, where w_i is the weight assigned to color i.

Your solution will be considered correct if all the following conditions are satisfied:

- $1 \leq w_i \leq K$
- $w_i \neq w_j$ for all $i \neq j$
- Given that the i-th edge's weight is w_{c_i}, the tree $\mathcal{T}$ is a maximum spanning tree of G. Note that the maximum spanning tree may not be unique. Your answer is considered correct as long as $\mathcal{T}$ has the maximum total weight.

Sample Input 1	Sample Output 1
2 5 7 3 1 2 1 1 2 3 2 1 3 4 1 1 4 5 2 1 1 5 3 0 2 3 1 0 2 4 3 0 6 8 5 2 6 1 0 5 1 3 1 1 5 3 0 5 6 2 1 5 3 1 1 5 1 3 0 4 2 2 1 1 4 4 1	2 3 1 2 5 4 3 1

This page is intentionally left blank.