MIAKI Presents

Inter University Programming Contest (IUPC)

BITFEST 2025

Hosted By
Department of Computer Science and Engineering
Khulna University of Engineering and Technology

Supported by

You get 11 problems, 14 pages and 300 minutes.

Problem A. Perpendicular Parking

An inter-university programming contest is about to take place inside the Tower of Hanoi. n universities have come to participate in the contest, each having $(n + 1)$ teams. Each team brought their own car, but all the teams of a university arrived together and will leave together. Two teams have cars of the same color if and only if they are from the same university.

The parking lot in front of the tower consists of a horizontal driving lane and $(n + 1)$ vertical parking lanes. Each parking lane can fit up to $(n + 1)$ cars in a single straight line. At the time of arrival, the teams of each university parked their cars side-by-side (in different parking lanes) and gave their car keys to the parking attendant.

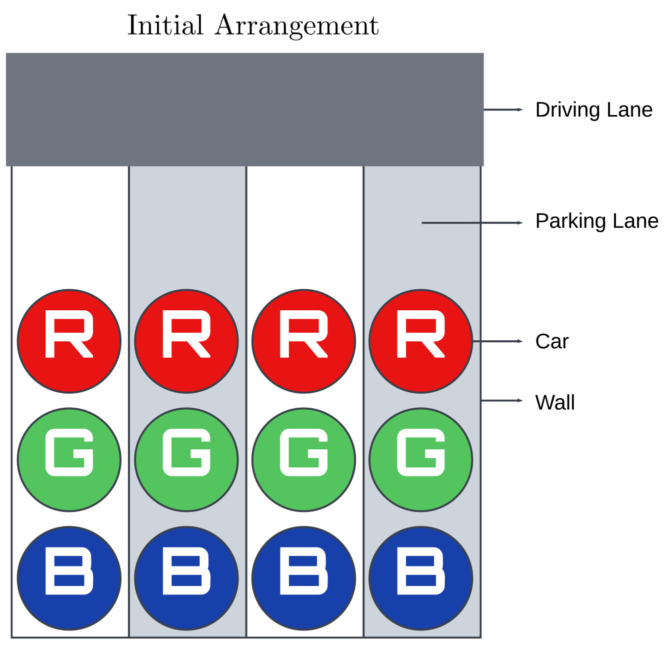
After all the teams have arrived, the attendant notices a critical issue. In the current arrangement, cars from different universities are mixed within the same parking lanes. Since the only way to enter or exit a parking lane is through the driving lane, all cars except those of the last-arriving university are blocked by cars of teams that arrived later.

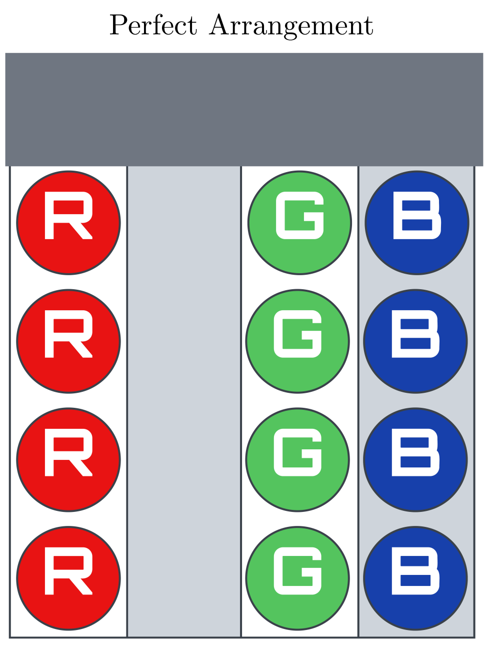
After the contest ends, a contestant may discuss the problems with their classmates, spend time sightseeing around the beautiful tower, catch the concert on the top floor, or leave immediately. So the departure order of universities is not predictable.

To resolve the issue, the parking attendant needs to rearrange the cars in a 'perfect arrangement' so that all the cars belonging to the same university are parked entirely within a single parking lane. This new arrangement would allow any university to leave independently without obstruction, regardless of the departure order.

To rearrange the cars, the attendant can perform some moves. In each move, the attendant will take the frontmost car from any parking lane, drive it through the driving lane, and park it in another parking lane that is not full. However, stopping a car in the middle of the driving lane is strictly forbidden. Formally, a move can be described by a pair (x, y), performing which means driving the frontmost car from the x-th lane to the y-th lane.

Now, the attendant wants to rearrange the cars before the contest ends. So he asked for your help. You need to tell him a sequence of at most $50n^2$ moves to rearrange the cars in a perfect arrangement or let him know that it is not possible.





Input

The first line of the input contains a single positive integer t $(1 \le t \le 10)$ — the number of test cases. Then t test cases follow.

Each test case contains only one integer in a line n $(2 \le n \le 200)$ — the number of universities.

The sum of $50 \cdot n^2$ is at most 2×10^6 over all test cases.

Output

For each test case, output the result as follows:

- If it is **not possible** to achieve a perfect arrangement with a sequence of at most $50n^2$ moves:
  - Output a single line containing -1.
- If it is **possible**:
  - In the first line, output a single integer m $(1 \le m \le 50n^2)$ — the total number of moves in the sequence.
  - The next m lines should specify the sequence of moves *in the exact order they are to be performed* to achieve a perfect arrangement. Each of these m lines should correspond to one move in the sequence.
  - To describe the i-th move $(1 \le i \le m)$, output two integers in a line x_i and y_i $(1 \le x_i, y_i \le n+1$, $x_i \ne y_i)$ — the source and destination lane in that move, respectively.

Sample Input	**Sample Output**

1
3
	21
	1 3
	2 4
	1 2
	1 2
	3 1
	3 1
	4 1
	4 1
	4 3
	4 3
	2 4
	3 4
	3 4
	3 4
	3 2
	4 3
	4 3
	2 4
	2 3
	2 3
	2 4

Problem B. Baskets and Stones

You have a pile of n stones, a basket and a biased coin. The coin has p $(0 < p < 1)$ probabilty of landing as heads and $1 - p$ probabilty of landing as tails. You also have an infinite supply of baskets elsewhere.

With nothing else to do, you decided to pass your time by continuously flipping the coin. If the coin lands as heads, you take a basket from the infinite supply and add it to the baskets in front of you. And if the coin lands as tails, you uniformly randomly choose a stone from the pile and uniformly randomly choose a basket from in front of you. Then you put the chosen stone inside the chosen basket. You continue this process until the stones pile is empty.

The above seems like a very boring process, I know. So you don't have to follow through. Instead, assume that after the process has ended, $X_1, X_2, X_3, \cdots$ are the count of stones in each of the basket in front of you. You have to compute the expected value of $X_1^2 + X_2^2 + \cdots$.

Oh, and you will be computing the value modulo $10^9 + 7$ (1000000007). See input and output for details.

Input

First line of the input will contain an integer T $(1 \leq T \leq 10^5)$ denoting the number of independent test cases. Each of the next T lines will contain two integers n $(1 \leq n \leq 10^5)$, the number of stones in pile and p $(1 < p < 10^9 + 7)$, the probabilty of the coin landing heads. In particular the probabilty can be expressed as $\frac{a}{b}, b \neq 0$ as a fraction and you are given $a \cdot b^{-1} \bmod 10^9 + 7$

Output

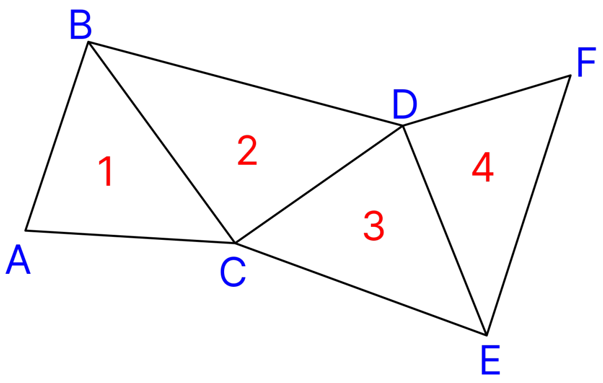
For each of the test cases, print the expected value modulo $10^9 + 7$. In particular, the expected value can be proven to always be representable as a fraction $\frac{a}{b}, b \neq 0$, you need to print $a \cdot b^{-1} \bmod 10^9 + 7$

Sample Input	Sample Output

1
1 500000004
2 500000004
	1
	3

Problem C. Effective Rendering

In OpenGL, the *GL_TRIANGLE_STRIP* rendering sequence is used to draw a series of connected triangles. A set of vertices is provided in a sequence, where each consecutive three vertices form a triangle. The figure below shows a triangulation of six points A, B, C, D, E, and F. The sequence [A, B, C, D, E, F] follows the *GL_TRIANGLE_STRIP* rendering mode, creating four triangles ([A, B, C], [B, C, D], [C, D, E], [D, E, F]) formed by the six points, covering the whole area of the polygon [A, B, D, F, E, C] without any overlaps.



Source: Wikipedia

You are given N boundary points of a simple 2D polygon in clockwise order. You need to output a sequence of the given points that follows the *GL_TRIANGLE_STRIP* rendering sequence, where the triangulation needs to cover the whole area of the polygon, and the max area of a triangle must be minimized. Note that no triangle can degenerate or form an area outside the polygon. Also, there should not be any overlap between any two triangles. If there is no such sequence, your output will be -1.

Input

The first line of the input will contain an integer $T(1 \leq T \leq 200)$ denoting the number of test cases. The first line of each test case will contain an integer $N(3 \leq N \leq 100)$. The next N lines of each test case contain two integers X and Y $(-10^9 \leq X, Y \leq 10^9)$, denoting the boundary points of the given polygon in clockwise order. The summation of N over all test cases doesn't exceed 1000.

Output

For each of the test cases, if there is not a *GL_TRIANGLE_STRIP* sequence possible to meet the criteria, print -1, otherwise print a space-separated sequence of point indices (1-based). If there are multiple possible sequences, output the lexicographically least sequence.

Sample Input	Sample Output

1
4
0 0
0 2
2 2
2 0
	1 2 4 3

Problem D. Dysbinaria

Mr. X is really having a hard time with his Boolean Algebra course. He thinks he has Dysbinaria, which he claims causes him difficulty understanding binary numbers. However, his professor, Ms. Y, thinks it's a made-up condition and refuses to give him a break. She believes Mr. X has been slacking off recently and has decided to teach him a lesson. So, she gave him a challenging assignment, telling him that if he solves it, he will pass the course.

Hearing this, Mr. X couldn't help but cheer with joy. But his joy was short-lived. Once he read the description of the assignment, tears began to drip from his eyes. And believe me, it's not the tear of joy. In the assignment, Ms. Y gave Mr. X a random array of integers and asked him to perform various queries involving bitwise operations and array manipulations. After performing some of these operations, Mr. X needs to compute the XOR sum of a specific range in the array.

Though Mr. X may or may not have Dysbinaria, one thing is certain: he's not dumb. He understands that tackling this problem alone is impossible. And so, he turns to you, his good friend, for help. Will you step in and help him out of this misery, or will you let him suffer through the pain of his binary woes? The clock is ticking, and Mr. X can't take much more of this!

You are given an array of integers that Mr. X needs to work with, and your task is to help him perform a series of operations on specific subranges of the array.

You need to implement the following methods to make sure Mr. X doesn't get lost in the maze of binary confusion:

- `setBit(l, r, k)`: Set the k^{th} bit to 1 for each number in the range $[l, r]$.
- `clearBit(l, r, k)`: Clear the k^{th} bit (set it to 0) for each number in the range $[l, r]$.
- `flip(l, r)`: Flip the i^{th} bit of each number in the range $[l, r]$ if the number of elements in this range that have the i^{th} bit set is odd.
- `rotateSubarray(l, r, m, d)`: Rotate the subarray $[l, r]$, m times. If $d = 1$, rotate clockwise; if $d = 2$, rotate anticlockwise.
- `subarrayXORSum(l, r)`: Returns the XOR sum of all elements in the range $[l, r]$. More formally, it returns the value of $a_l \oplus a_{l+1} \oplus a_{l+2} \cdots \oplus a_r$.

Input

The first line contains an integer T $(1 \leq T \leq 10^5)$, the number of test cases. Each test case starts with two integers n and q $(1 \leq n, q \leq 3 \times 10^5)$, where n is the number of elements in the array, and q is the number of operations to perform. The next line contains n space-separated integers $a_1, a_2, \ldots, a_n$ $(0 \leq a_i \leq 2^{31} - 1)$. Each of the following q lines represents a query of one of the following types.

`1 l r k` – Perform `setBit(l, r, k)`
`2 l r k` – Perform `clearBit(l, r, k)`
`3 l r` – Perform `flip(l, r)`
`4 l r m d` – Perform `rotateSubarray(l, r, m, d)`
`5 l r` – Calculate the XOR sum by performing `subarrayXORSum(l, r)`

The first integer in each query, t_i $(1 \leq t_i \leq 5)$, represents the type of the query. The values l and r in each query represent the subarray $[l, r]$ $(1 \leq l \leq r \leq n)$. The value k $(1 \leq k \leq 31)$ represents the position of the bit on which the operations are to be performed. The value m $(1 \leq m \leq n)$ represents the number of rotations, and d $(d \in \{1, 2\})$ represents the direction of the rotations. It is guaranteed that the sum of n is at most 3×10^5 and the sum of q is at most 3×10^5 over all test cases. For each test case, there will be at least one query of *type* 5.

Output

For each test case, output the result of each *type* 5 query on a new line.

Sample Input	Sample Output

1
3 7
1 2 3
1 1 3 2
5 1 3
3 1 3
4 1 3 1 1
4 1 3 2 2
2 3 3 1
5 1 2
	2
	1

Explanation

For the given sample input:

Initial array: $[1, 2, 3]$

Operation 1: `setBit(1, 3, 2)` – Setting the 2^{nd} bit of each number in the subarray $[1, 3]$ results in the array $[3, 2, 3]$.

Operation 2: `subarrayXORSum(1, 3)` – Perform XOR sum on the subarray $[1, 3]$: $3 \oplus 2 \oplus 3 = 2$.

Operation 3: `flip(1, 3)` – The binary representations of the numbers in the range $[1, 3]$ are: $\{3 \to 11,\ 2 \to 10,\ 3 \to 11\}$. Looking at the bit positions, it's clear that only the 2^{nd} bit requires a flip. After flipping the 2^{nd} bit, the array becomes: $[1, 0, 1]$

Operation 4: `rotateSubarray(1, 3, 1, 1)` – Rotate the subarray $[1, 3]$ clockwise by 1 position, resulting in the array $[1, 1, 0]$.

Operation 5: `rotateSubarray(1, 3, 2, 2)` – Rotate the subarray $[1, 3]$ anticlockwise by 2 positions, resulting in the array $[0, 1, 1]$.

Operation 6: `clearBit(3, 3, 1)` – The subarray contains only one value, 1. Clearing the 1^{st} bit of 1 changes it to 0, resulting in the array $[0, 1, 0]$.

Operation 7: `subarrayXORSum(1, 2)` – Perform XOR sum on the subarray $[1, 2]$: $0 \oplus 1 = 1$.

Problem E. Escape Plan

After waking up from sleep, you find yourself inside the storage room of a fuel station in a city attacked by zombies. The first thing you check is that your Cybertruck with 1000 horsepower and infinite fuel capacity is still parked outside, but currently it has zero fuel. So you immediately start refueling and discover a map of the city.

The city has N important junctions numbered from 1 to N. i^{th} junction has a fuel station with a limited amount of fuel F_i. N junctions are connected with $N-1$ roads. i^{th} road connects junction u_i and v_i bidirectionally and has a fuel consumption value of w_i, means your Cybertruck needs w_i ammount of fuel to travel the road.

Currently, you are at the fuel station of junction S. You can escape through any outer junction that is connected with only one junction.

Although your aim is to escape the city as quickly as possible, stopping at junctions to refuel is extremely dangerous. So, you need to find a route that requires the fewest stops.

You can refuel at any fuel station only once. It is guaranteed that you can reach any junction from the starting junction by traveling along the roads, provided you have enough fuel.

If your Cybertruck runs out of fuel in the middle of a road, you'll surely be killed. So, solve the problem seriously.

Input

The first line of the input will contain an integer $T(1 \le T \le 100)$ denoting the number of test cases. Then T test cases follow. The first line of each test case will contain two integers N and $S(1 \le N \le 16, 1 \le S \le N)$. The second line of each test case will contain N space separated integers $F_1, F_2, \ldots, F_N(1 \le F_i \le 10^9)$. Next $N-1$ line of each test case will contain three integers u_i, v_i and $w_i(1 \le u_i, v_i \le N, 1 \le w_i \le 10^9)$ describing each road.

Output

For each of the test cases, print the fewest number of additional stops (excluding the starting junction) needed to escape the city. If it is impossible to escape the city print `"impossible"`(without quotes).

Sample Input	Sample Output

```
2
7 1
2 10 4 7 3 5 1
1 2 2
5 7 3
1 5 2
2 3 20
1 4 15
5 6 1
3 1
5 10 10
1 2 10
1 3 10
```
	```
	1
	impossible
	```

Problem F. Funny Fairy Tale

You are given an array of N integers, $A_1, A_2, \ldots, A_N$. In one operation, you can pick a subsequence(could be empty) of the array. Then you can change the sign of any elements in that subsequence. As a result of this operation, you get a new subsequence. Note that these changes **do not** affect the original array A.

For example, if the chosen subsequence is $[4, 5, 4, 4]$; it has 2 distinct integers: $\{4, 5\}$. You can change it into $[4, -5, 4, -4]$, which has three distinct integers: $\{4, -5, -4\}$.

You are tasked with answering Q queries. In each query, you are given an integer K, and your goal is to determine, for all possible subsequences you can make from the array after applying the operation, how many of them have the number of distinct integers that is a multiple of K.

Please note that: A subsequence of an array is found by removing zero or more elements from the array. Two resulting subsequences are different if they have different sizes or the index sets of the numbers included in them are different or there is an index in the subsequences where they have different values.

Input

The first line has two numbers, N and Q $(1 \leq N, Q \leq 10^5)$.

The next line contains N integers in the range $[1 \ldots N]$.

Then, there are Q lines. Each line contains an integer K $(1 \leq K \leq 10^5)$.

Output

For each query, print the answer in a single line. Since the result can be very large, print it modulo 998244353.

Sample Input	**Sample Output**

3 3
2 1 2
1
2
3
	27
	15
	5

Explanation

All possible subsequences those can be made:

$\{\} \rightarrow \{\}$
$\{2\} \rightarrow \{2\}, \{-2\}$
$\{1\} \rightarrow \{1\}, \{-1\}$
$\{2\} \rightarrow \{2\}, \{-2\}$
$\{2,1\} \rightarrow \{2,1\}, \{-2,1\}, \{2,-1\}, \{-2,-1\}$
$\{2,2\} \rightarrow \{2,2\}, \{-2,2\}, \{2,-2\}, \{-2,-2\}$
$\{1,2\} \rightarrow \{1,2\}, \{-1,2\}, \{1,-2\}, \{-1,-2\}$
$\{2,1,2\} \rightarrow \{2,1,2\}, \{-2,1,2\}, \{2,-1,2\}, \{2,1,-2\}, \{-2,-1,2\}, \{-2,1,-2\}, \{2,-1,-2\}, \{-2,-1,-2\}$

Now,
There are 1 subsequence with 0 distinct integers.
There are 8 subsequences with 1 distinct integers.
There are 14 subsequences with 2 distinct integers.
There are 4 subsequences with 3 distinct integers.

Problem G. The Scientist

Shil is a mad genius who wants to create the "Elixir of Life". He has discovered a new element that can be used to create the elixir. The special thing about this element is that it has multiple states and can be transformed from one state to another through a reaction.

He knows about N different states of the element. He also has a list of M **unique** reactions that can be performed on that element. Each reaction is of the form (A_i, B_i, C_i). Here, A_i and B_i represent the two states of the element involved in the reaction, and C_i is the time (in nanoseconds) required to transform the element from one state into the other. Transformations are bi-directional, meaning the element can be transformed from state A_i to state B_i and vice versa, both taking C_i nanoseconds. One day, he starts a chain reaction where the element is transformed from state X to state Y and then from state Y to state Z, and so on. Unfortunately, he lost track of the transformations. All he has is a log of S snapshots recorded at regular intervals. The snapshots are taken exactly T nanoseconds apart, starting from $t = 0$. An element can stay in the same state for an arbitrary amount of time, and it is guaranteed that at the time of each snapshot, the element is in one of the N states.

Creating the elixir is a complex process. It might require the knowledge of the transformations that took place between the snapshots. Shil has Q queries for you. For the i^{th} query, he needs to know if it was possible for the element to be in state P_i between time L_i and R_i nanoseconds (inclusive). As his assistant, it is your job to help him answer these queries. Each query is independent of the others.

Input

The first line of the input will contain two integers $N (1 \le N \le 100)$ and $M (0 \le M \le \frac{N\times(N-1)}{2})$ denoting the number of states and the number of reactions respectively.

The next M lines will contain three integers $A_i, B_i, C_i (1 \le A_i, B_i \le N, A_i \ne B_i, 1 \le C_i \le 10^9)$ denoting the two states and the time (in nanoseconds) required to transform one state into the other. It is guaranteed that all reactions are unique, that is, no pair of (A_i, B_i) appears more than once.

The next line will contain two integers $S (1 \le S \le 100)$ and $T (1 \le T \le 10^9)$ denoting the number of snapshots and the time interval between each snapshot.

After that, there will be a line containing S space separated integers $X_i (1 \le X_i \le N)$ denoting the state of the element at the time of the i^{th} snapshot.

The next line will contain an integer $Q (1 \le Q \le 10^5)$ denoting the number of queries.

Each of the next Q lines will contain three integers $P_i, L_i, R_i (1 \le P_i \le N, 0 \le L_i \le R_i \le 10^9)$ denoting the state of the element for the i^{th} query, the start time, and the end time.

Output

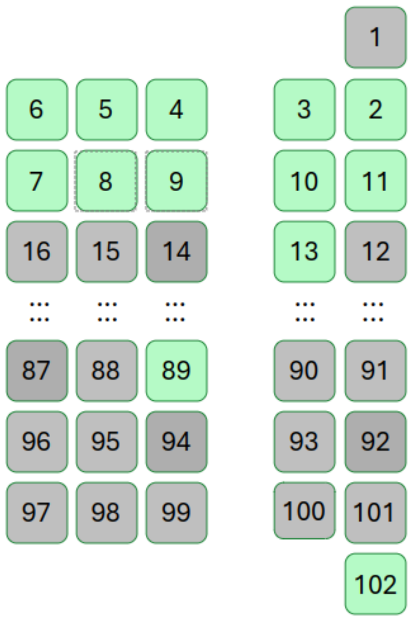
For the i^{th} query, print "Yes" if it was possible for the element to be in state P_i between time L_i and R_i nanoseconds (inclusive), and "No" otherwise (without quotes).

Sample Input	Sample Output
`5 5`	`Yes`
`3 5 1`	`No`
`2 1 2`	`Yes`
`1 3 3`	`Yes`
`4 3 4`	`Yes`
`4 5 1`	
`5 5`	
`4 3 2 3 5`	
`5`	
`5 0 5`	
`4 6 12`	
`1 1 12`	
`3 16 17`	
`4 16 17`	

Problem H. Ticket Dilemma

You and your friend want to participate in Impossibly Unsolvable Problems Challenge(IUPC). As you have to travel by train, you have to buy tickets for both of you.

You find that the seats are arranged in the following manner.



Seats are numbered from 1 to 102 and the seat arrangement is fixed as shown in the image. Some seats are already taken and some are available.
You are given an array of available seat numbers, find how many ways two adjacent seats can be chosen.

Input

The first line of the input will contain an integer $T (1 \leq T \leq 100)$ denoting the number of test cases. The first line of each test case will contain an integer $N (1 \leq N \leq 102)$ denoting the number of seats available. The second line of each test case will contain N unique integers $s_1, s_2, \ldots, s_N (1 \leq s_i \leq 102)$ denoting the seat numbers that are available.

Output

For each test case, print an integer S, representing the number of ways to choose two adjacent seats. In the next S lines, print a pair of space-separated integers x_i and y_i such that x_i and y_i are adjacent seat numbers and $x_i < y_i$, in increasing order (i.e., if $i < j$ then $x_i < x_j$).

Sample Input	**Sample Output**

1
13
2 3 4 5 6 7 8 9 10 11 13 89 102
	6
	2 3
	4 5
	5 6
	7 8
	8 9
	10 11

Problem I. Travel on the grid II

You are given a grid of a size $N \times M$. You need to travel from cell $(1, 1)$ to (N, M). From any position (X, Y), you can travel to only one of the two cells $(X + 1, Y)$ or $(X, Y + 1)$. There are mines in different cells. When any mine explodes, it can kill anyone standing on it's cell or anyone standing on the 4 other adjacent cells of the mine (top, bottom, left, right). Stepping into any of these five cells will trigger the mine and it explodes.

Before starting the journey, you can choose any number of mines on the grid and defuse them beforehand. Once a mine is defused, it will not blast in the future.

Now, you need to answer in how many ways you can reach from $(1, 1)$ to (N, M) by diffusing the minimum number of mines.

Input

The first line will contain a single integer $T (1 \leq T \leq 10)$ denoting the number of test cases. First line of each test case will contain two integers $N (1 \leq N \leq 1000)$ and $M (1 \leq M \leq 1000)$, denoting the size of the grid. The following N lines will contain a string, each of them will be of length M. Each character of the string will be either '`.`' (ASCII code 46) or '`#`' (ASCII code 35). The character '`.`' denotes a safe place and the character '`#`' denotes the cell has a mine on it.

Output

For each test case, print the number of ways to reach (N, M) from $(1, 1)$ by defusing the minimum number of mines. As this number can be very big, print the number modulo $10^9 + 7 (1000000007)$.

Sample Input	Sample Output
1 3 3 .## .#. ###	4

Problem J. Earthquake

The villages of Stronghold are interconnected by a network of bridges. Each bridge has a "strength" representing its resilience. Recently, earthquakes have threatened these bridges: when an earthquake strikes with power X, it destroys all bridges with strength less than or equal to X. As bridges collapse, villages may become isolated, forming disconnected groups. A group of villages is considered connected if one can travel between any two villages in the group without crossing a destroyed bridge. If no such path exists between two groups of villages, they are considered disconnected.

The Council of Elders, concerned about the impact of these earthquakes, needs to determine how Stronghold would be affected for every possible level of damage. Specifically, they want to know the minimum earthquake power that would split the villages into at least K disconnected groups, where K ranges from 1 to N.

Your task is to calculate these critical values for each test case.

Input

The first line of input contains an integer T ($1 \le T \le 1000$), the number of test cases. Each test case begins with two space-separated integers N ($2 \le N \le 10^5$) and M ($1 \le M \le 2 \times 10^5$), the number of villages and the number of bridges, respectively. The next M lines of each test case describe the bridges. Each line contains three space-separated integers u, v, and s ($1 \le u, v \le N$ and $1 \le s \le 10^9$), representing a bridge connecting villages u and v with strength s. It is guaranteed that the total sum of N across all test cases does not exceed 2×10^5, and the total sum of M does not exceed 2×10^5. There is at most one bridge between any two villages, and it is possible for the villages to be disconnected.

Output

For each test case, print N space-separated integers in one line, where the k-th integer represents the minimum earthquake power that will split the villages into at least k disconnected groups.

Sample Input	Sample Output

2
5 5
1 2 5
1 3 2
2 3 4
3 4 7
4 5 1
3 3
1 2 3
2 3 4
1 3 5
	0 1 4 5 7
	0 4 5

Explanation for the first case:

- $K = 1$: You don't need any earthquake because the villages are in a single connected group: {1, 2, 3, 4, 5}. So the answer is 0.
- $K = 2$: Earthquake power 1 (bridge 4-5 destroyed). Villages split into groups: {1, 2, 3, 4}, {5}.
- $K = 3$: Earthquake power 4 (bridges 1-3, 2-3, 4-5 destroyed). Villages split into groups: {1, 2}, {3, 4}, {5}
- $K = 4$: Earthquake power 5 (all bridges except 3-4 destroyed). Villages split into groups: {1}, {2}, {3, 4}, {5}
- $K = 5$: Earthquake power 7 (all bridges destroyed). Villages split into groups: {1}, {2}, {3}, {4}, {5}.

Problem K. The Beast

A terrifying beast is attacking the land. To defeat it, the king has deployed N shooters, each firing bullets at their own steady pace. The i^{th} shooter fires a bullet every i^{th} minute.

The beast requires exactly K bullets to be destroyed. If multiple shooters fire at the same minute, all bullets count toward the total. Your task is to determine the **exact minute** when the beast is destroyed.

Input

The first line contains a single integer $T(1 \leq T \leq 1000)$ denoting the number of test cases. Each of the next T lines contains two integers N and $K(1 \leq N, K \leq 10^6)$

Output

For each test case, print the answer - the minute at which the beast gets destroyed.

Sample Input	Sample Output

4
2 10
10 10
5 10
5 11
	7
	5
	5
	6

Explanation for $N = 5, K = 10$:
At each minute, the following shooters fire bullets:

- **1st minute:** Shooter 1 (1 bullet)
- **2nd minute:** Shooter 1, Shooter 2 (2 bullets)
- **3rd minute:** Shooter 1, Shooter 3 (2 bullets)
- **4th minute:** Shooter 1, Shooter 2, Shooter 4 (3 bullets)
- **5th minute:** Shooter 1, Shooter 5 (2 bullets)

The total number of bullets fired is $1 + 2 + 2 + 3 + 2 = 10$, and the beast is destroyed at the 5^{th} minute.