Problems

Problem A: Thief

Time limit: 1s; Memory limit: 512 MB

A thief breaks into a house containing **N** items. The i-th item has a weight w_i and a value c_i . The thief has a bag with a weight capacity of **W**. If the total weight of the items exceeds **W**, the bag will tear, and the thief won't be able to carry anything.

After filling the bag, the thief thinks, *"I can carry the bag with one hand, so I can grab one more item with my other hand."* Acting on this thought, the thief uses his free hand to take an additional item, provided its weight does not exceed **H**.

Determine the maximum total value of items the thief can carry.

Input

The first line contains three positive integers **N**, **W** and **H** ($1 \le N \le 100$, $1 \le W, H \le 10^9$). The following **N** lines each contain two positive integers w_i and c_i ($1 \le w_i \le 10^9$,$1 \le ci \le 10^3$).

Output

A single line containing the maximum total value the thief can take.

Sample

Input	Output
3 10 2 2 5 1 9 4 6	20
3 2 4 2 5 1 9 4 6	15

Explanation:

- For example 1: The bag can hold all items, so the total value carried is 20.
- For example 2: The thief can put item 2 in the bag, and carry item 3 by hand.

Problem B: Puzzle About Magical Quadruples

Time limit: 1s; Memory limit: 512 MB

Quynh Anh is an adventurous young programmer who loves solving puzzles with numbers. One day, while wandering through a digital forest named ESC21, she stumbled upon an enchanted scroll that described a peculiar puzzle involving arrays and bitwise operations. The puzzle seemed so intriguing that she decided to challenge herself and her friends to solve it.

Here's how the puzzle works:

Quynh Anh has a collection of **N** non-negative integers, represented as an array **A** where **A[1]**, **A[2]**, ..., **A[N]** are the elements. She defines a *super beautiful quadruple* as a tuple of four indices **(i, j, p, q)** such that:

- **$1 \le i, j, p, q \le N$** (note that they are not necessarily distinct).
- **A[i] & A[j] & A[p] & A[q] = 0**, where **&** denotes the bitwise **AND** operation.

In other words, the bitwise **AND** of the four chosen elements at these indices must equal zero.

But that's not all! Quynh Anh wants to explore these quadruples in a very specific order. The quadruples should be sorted lexicographically, meaning that a quadruple **(b[1], b[2], b[3], b[4])** is considered smaller than another quadruple **(c[1], c[2], c[3], c[4])** if there exists an index **i** ($1 \le i \le 4$) such that:

- **b[1]** = **c[1]**, **b[2]** = **c[2]**, ..., **b**[**i-1**] = **c[i-1]**
- and **b[i]** < **c[i]**.

Your challenge is to help Quynh Anh find the **k**-th lexicographically smallest quadruple from all possible super beautiful quadruples that satisfy the above conditions.

Input

- The first line contains two integers **N** and **k** ($1 \le N \le 5000$, $1 \le k \le N^4$).
- The second line contains **N** integers, representing the array **A**. ($0 \le A[x] \le 10^6$ for all $1 \le x \le N$)

Output

- Output four integers, **i, j, p, q**, representing the **k**-th lexicographically smallest super beautiful quadruple. In the case of **k** is greater than the total number of beautiful quadruples, print **-1**.

Sample

Input	Output
6 1 1 4 7 10 13 16	1 1 1 2
6 2 1 4 7 10 13 16	1 1 1 4
6 1120 1 4 7 10 13 16	6 6 6 5
6 1121 1 4 7 10 13 16	-1

Problem C: Sum of LCMs

Time limit: 2s; Memory limit: 512 MB

Given a sequence **A** consisting of **N** positive integers.

Consider all non-empty subsets of the sequence **A**, and calculate the **least common multiple (LCM)** of each subset. Find the sum of all these LCMs.

Since the result can be very large, return the remainder of the sum modulo 10^9+7.

Input

- The first line contains a positive integer **T**, the number of test cases ($1 \le \mathbf{T} \le 40$).
- For each test case:
  - The first line contains a single positive integer **N**. ($1 \le \mathbf{N} \le 100$).
  - The next N lines each contain a positive integer **A**[i]. ($1 \le \mathbf{A}[i] \le 500$).

.

Output

- **T** lines, each containing a single positive integer representing the answer for each test case.

Sample

Input	Output
4 5 1 1 1 1 1 3 1 2 3 6 4 4 4 2 2 2 10 9 5 12 58 1 85 24 90 100 99	31 23 238 651657343

Problem D: Divide and Candies

Time limit: 3s; Memory limit: 512 MB

Anh is a fourth-grade student who dreams of competing in the ICPC World Finals. Today, he is studying division operations, and to make the study more challenging, Anh wants to convert all numbers into different numerical bases before doing the division operations.

Anh has *n* candies arranged in a circle, each with a positive integer flavor value *a[i]*, and a positive integer base *m*. To have more fun while eating candies, Anh will play a game. He intends to convert each flavor number *a[i]* into base *m*, resulting in the base-m representation *b[i]*. The game process is as follows:

- Anh selects and eats one candy *i* from the circle.
- After eating the chosen candy, he identifies the two candies adjacent to it. Let these adjacent candies *j* and *k*, which have base-m representation *b[j]* and *b[k]*.
- Anh computes the sum of fractions: $\frac{b_j}{b_k} + \frac{b_k}{b_j}$

He then writes this sum as a decimal number in base 10 and counts the number of digits in the fractional part (both non-repeating and repeating digits). Example: $\frac{5}{14} + \frac{14}{5} = 3.1(571428)$. He writes down 1 non-repeating digit and 6 repeating digits, totaling 7 digits after the decimal point. Similarly, $\frac{3}{2} = 1.5$ has 1 non-repeating digit and 0 repeating digits, and $\frac{5}{1} = 5$ has 0 non-repeating digits and 0 repeating digits.

- After performing the computation and eating the candy *i*, Anh arranges the two candies *j* and *k* that are adjacent to *i* to be adjacent to each other again, maintaining the circular arrangement. This reduces the number of candies by one after each step.
- Anh repeats the candy-eating process until only two candies remain.
- For the final two candies, Anh performs the fractional operation as described above and counts the digits in the fractional part for those two candies. He then eats the last two candies and finishes the game.

Although the game is fun at first, Anh becomes tired of writing so many digits, so he wants to eat all the candies while writing the smallest number of digits. Help Anh determine the

minimum total number of digits he needs to write down throughout the entire process of eating all *n* candies.

Input

- The first line contains two integers *n* ($2 \leq n \leq 80$) and *m* ($2 \leq m \leq 10^6$)
- The next line contains *n* integers a_i ($1 \leq a_i \leq 10^6$)

Output

- One line contains an integer that represents the minimum total number of digits Anh needs to write down.

Sample

Input	Output
4 10 1 5 7 3	4
2 10 14 5	7
5 2 4 7 5 11 6	17
5 104 654074 236092 654237 254045 253310	4605392

Explanation

Explanation for example 1:

- Anh can eat the 3rd candy with value 7 first, then he needs to write down 2 digits, since $\frac{5}{3} + \frac{3}{5} = 3.2(6)$.
- Then with candies 1 5 3 left, he can eat 3rd candy with value 3, then he writes down 1 digit, since $\frac{5}{1} + \frac{1}{5} = 5.2$
- Then with candies 1 5 left, he eats 2 candies and writes down 1 digit, since $\frac{5}{1} + \frac{1}{5} = 5.2$
- Anh writes down a totally of 4 digits, and this is the minimum value he can achieve.

Explanation for example 3:

- Anh first converts each number to base 2: 4 is 100, 7 is 111, 5 is 101, 11 is 1011, and 6 is 110. The array b now is 100, 111, 101, 1011, 110
- Anh eats 2nd candy with value 111 and writes down 6 digits, since $\frac{100}{101} + \frac{101}{100} = 2.00(0099)$. The remaining candies are 100, 101, 1011, 110.
- Anh then eats 3rd candy with value 1011 and writes down 5 digits. The remaining candies are 100, 101, 110.
- Anh then eats 2nd candy with value 101 and writes down 3 digits. The remaining candies are 100, 110.
- Anh eats two remaining candies and writes down 3 digits.
- Anh writes down a total of 6 + 5 + 3 + 3 = 17 digits.

Problem E: Island Challenge

Time limit: 1s; Memory limit: 512 MB

In a video game, your hero has a challenge at an island. The challenge is visiting all shrines in this island.

The island can be described by $R \times C$ rectangle table. Each cell of the table is either a mountain, a shrine or an empty plot. Shrines are labeled as characters 'S', mountains are labeled as characters '#', empty plots are labeled as characters '.' . These mountains are so high and dangerous that the hero can not enter these cells, the hero can only enter shrines or empty plots.

Initially, the hero starts at the cell $(1, 1)$ - guaranteed to be an empty plot. If he is in the cell (i, j), he can only move to the cells $(i + 1, j), (i - 1, j), (i, j + 1), (j, i - 1)$ and not allowed to leave the island. Each such move takes 1 unit of time.

To win the challenge, your hero have to visit all N shrines in the minimum time. Given the map of the island, find the minimum time.

Input

- The first line contains three positive integers R, C, N. ($1 \leq R, C \leq 100$, $1 \leq N \leq 15$)

- In the next R lines, each line contains C characters consists of characters '.', '#', or 'S' - that is an empty plot, a moutain or a shrine respectively. It is guaranteed that the map consists of N characters 'S', and the cell (1, 1) is guaranteed to be an empty plot.

Output

- Print an integer which is the minimum time to visit all N shrines in the island. If no such path exists, print -1.

Sample

Input	Output
3 6 3 .S..#. ..#.#. S.#S..	9

3 3 2 .#. #S. ..S	-1

Explanation

Explanation for example 1: The path was: $(1,1) - (2,1) - (3,1) - (3,2) - (2,2) - (1,2) - (1,3) - (1,4) - (2,4) - (3,4)$, the hero made 9 moves.

Explanation for example 2: There is no path to visit shrines.

Problem F: Three Buildings

Time limit: 1s; Memory limit: 512 MB

You are the owner of a construction company in the city. There is n workers in your company, and there are three buildings to be built (let us call those buildings A, B, C).

You are trying to assign workers to these three buildings - specifically, up to n_A, n_B, n_C (respectively) workers to building A, building B, and building C (respectively). One worker can be assigned to at most one building, and workers do have preferences over which building(s) they want to be assigned to.

For instance, suppose $n = 4$ and $n_A = n_B = n_C = 1$ (let us label the four workers 1, 2, 3, and 4).

- Only worker 1 wants to be assigned to building A.

- Only worker 1 wants to be assigned to building B.

- Both workers 2 and 3 want to be assigned to building C.

- Worker 4 does not want to be assigned to any building.

In this case, because $n_C = 1$, we must choose either worker 2 or worker 3 for building C. Worker 1 can only be assigned to either building A or building B. Therefore, we can assign at most two workers to the buildings in this example.

Given n, n_A, n_B, and n_C and the preferences of n workers, write a program that computes how to assign maximum number of workers to the buildings.

Input

- The first line will contain the number of test cases, T. ($1 \leq T \leq 10$)

- For each test case, the first line will contain n and the second line will contain n_A, n_B, and n_C separated by a whitespace. ($1 \leq n \leq 10{,}000$, $1 \leq n_A, n_B, n_C \leq n$)

- The following three lines will describe which workers want to be assigned to each building.

- The first of the three lines will begin with m_A (the number of workers who want to be assigned to building A), followed by m_A numbers (representing workers who are labeled

from 1 to n), separated by a whitespace. Likewise, the second line will contain m_B followed by m_B numbers, and the third line m_C followed by m_C numbers. ($1 \leq m_A, m_B, m_C \leq n$)

Output

- For each test case print the maximum number of workers you can assign to the three buildings.

Sample

Input	Output
2 5 2 1 1 4 1 2 3 4 1 4 1 5 4 1 1 1 1 1 1 1 2 2 3	4 2

Explanation

Explanation for testcase 1: Two out of three workers (1, 2, 3) should be assigned to building A. Worker 4 should be assigned to building B. Worker 5 should be assigned to building C.

Explanation for testcase 2: Described in the problem statement.

Problem G: Corner-Shared Cells

Time limit: 1s; Memory limit: 512 MB

Consider a squared-cell grid with **M** horizontal rows and **N** vertical columns. We denote **(i, j)** as the cell on the intersection of **i**-th row and **j**-th column. Given integers **p** and **q**, Gia Han is wondering how many cells share at least one corner with the cell **(p, q)**. Two cells **(i, j)** and **(k, l)** are said to share a corner if $|\mathbf{i} - \mathbf{k}| \leq \mathbf{1}$ and $|\mathbf{j} - \mathbf{l}| \leq \mathbf{1}$. Let's program to help her calculate the answer!.

Input

The first line contains two integers **M**, **N** ($1 \leq \mathbf{M}, \mathbf{N} \leq 200$)
The second line contains two integers **p**, **q** ($1 \leq \mathbf{p} \leq \mathbf{M}$, $1 \leq \mathbf{q} \leq \mathbf{N}$).

Output

Print the number of cells in the grid which share at least one corner with the cell (**p**, **q**).

Sample

Input	Output
4 3 4 2	5
4 3 3 2	8

Explanation of the first case:

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)
(3,1)	(3,2)	(3,3)
(4,1)	**(4,2)**	(4,3)

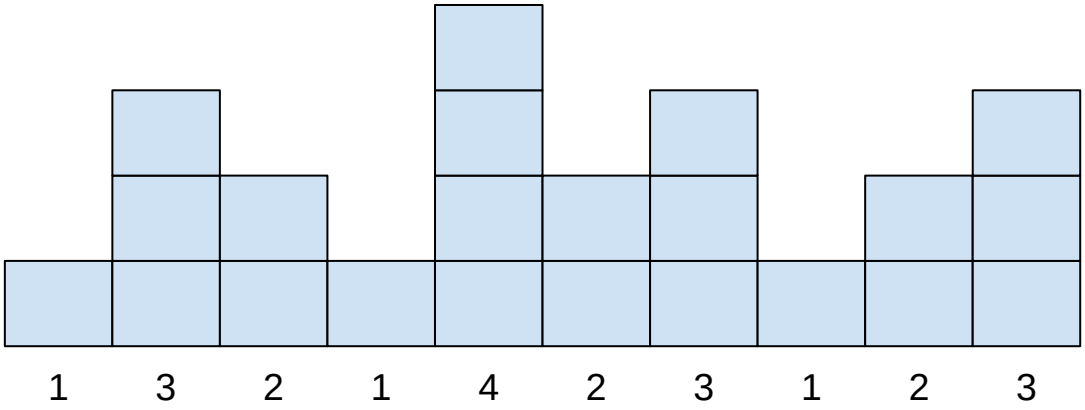
The cell **(4,2)** shares at least one corner with the cells **(3,1)**, **(3,2)**, **(3,3)**, **(4,1)**, and **(4,3)**.

Problem H: Painting competition

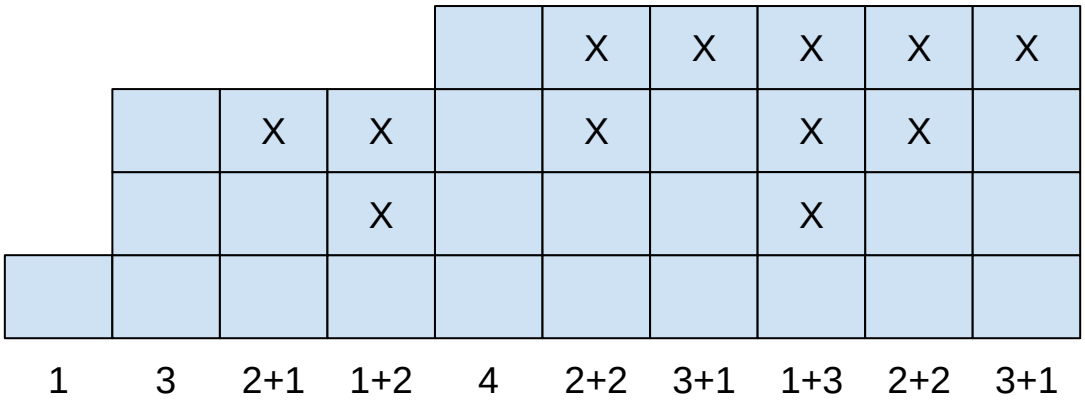
Time limit: 3s; Memory limit: 512 MB

You're joining a painting competition. The problem is drawing a line and controlling its stroke.

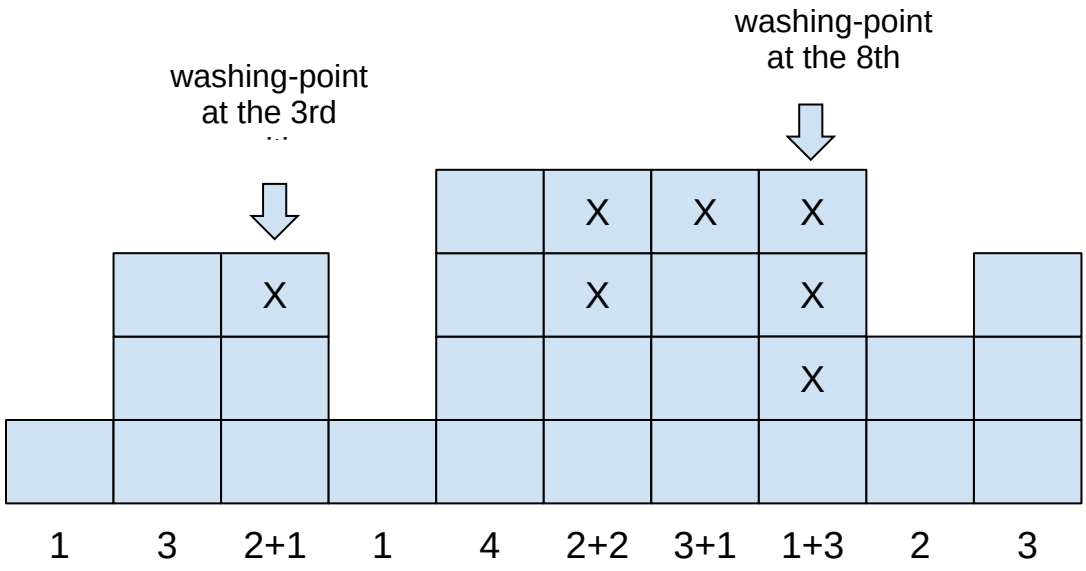
The problem can be described as an array of length n containing multiple integers a_i ($1 \leq i \leq n$) - expected stroke at position i-th. Contestants need to follow these integers to draw from left to right and control the line stroke. For example, if the exercise is an array $a = [1, 3, 2, 1, 4, 2, 3, 1, 2, 3]$ ($n = 10$), the expected line will be:



However, the paintbrush is too difficult to control. After increasing the line stroke, you could not decrease it until washing it. Then, the picture became (X pixels are caused by the paintbrush problem).



There are some moments in the competition, the contestants **have to** wash their paintbrush called **washing-point**. Afterward, you can restart the drawing with the required stroke from the next point. For example, if you wash the paintbrush at 3rd and 8th position (arrow symbols), the stroke will be reset immediately after those positions.



A sub-problem $a[l,r]$ is a contiguous subsequence $a_l\ a_{l+1}\ a_{l+2} \ldots . a_r$ of the original problem (supposed that the problem is only this subsequence and its **washing-point** at that moment, the remaining parts of the original problem will be ignored).

The sub-problem score $P(a[l,r])$ will be calculated by the number of incorrect pixels of the picture (number of X pixels).

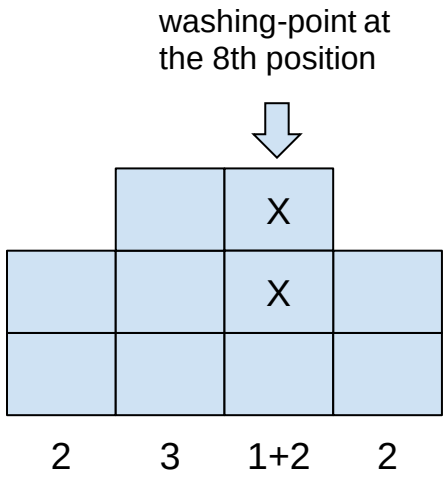
For example, if there is no **washing-point** in the original problem:

$$P(a[1,n]) = 1+2+2+1+3+2+1 = 12$$

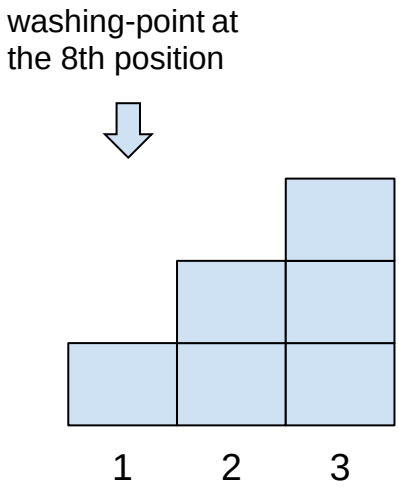
If there 2 **washing-point** at 3rd and 8th positions

$$P(a[1,n]) = 1 + 2 + 1 + 3 = 7$$

There are more examples for $a[6,9]$ and $a[8,10]$ (with **washing-points**).



P(a[6, 9]) = 2



P(a[8, 10]) = 0

However, the original problem is too long and boring. So the organizers decided to give contestants the following requests overtime (instead of completing the whole original problem):

- Request 1: Change the expected stroke at a specific position to a new value. In other words, given two numbers p and x, assign $a[p] = x$. This change will be retained.
- Request 2: Add a new **washing-point** at p. It guarantees that the position p is not an existing **washing-point**. This change will be retained.
- Request 3: Ask the contestant to complete the sub-problem $a[l, r]$.

Given the original problem and several requests. You need to answer the sub-problem score $P(a[l, r])$ that the contestant will get for each **request 3**.

Input

The first line contains two integers n and q ($1 \le n, q \le 2 \cdot 10^5$). The size of the given problem and the number of requests correspondingly.

The second line contains n integers a_i ($1 \le i \le n, 1 \le a_i \le 15$). The original problem description.

The next q lines describe requests. Each request will be one of following input formats:

- $1\ p\ x$: Change the expected stroke at p-th position to x ($1 \leq p \leq n, 1 \leq x \leq 15$).
- $2\ p$: Add a new **washing-point** at the p-th position ($1 \leq p \leq n$). It guarantees that the p-th position is not an existing **washing-point**. In other words, we will NOT have any two 2nd type requests with the same p value.
- $3\ l\ r$: Calculate the point $P(a[l,r])$ that the contestant will get supposed the sub-problem is $a[l,r]$ ($1 \leq l \leq r \leq n$). Then output the $P(a[l,r])$ value.

Output

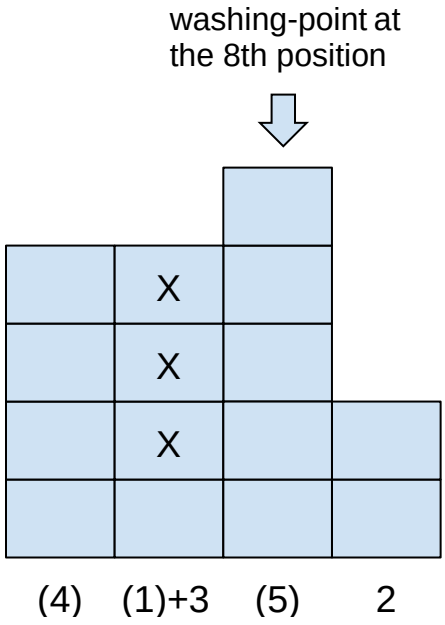
Each line of output will be an answer for the 3rd request.

Sample

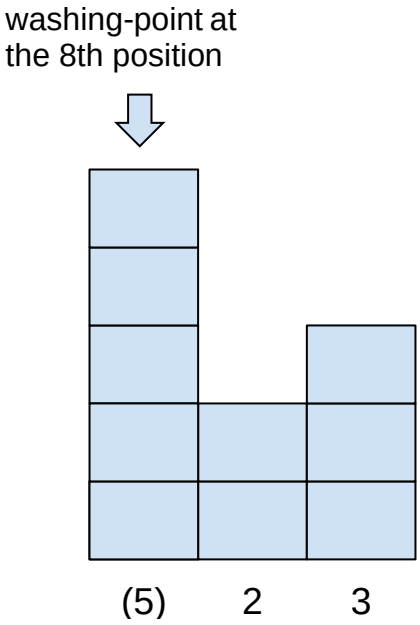
Input	Output
10 11 1 3 2 1 4 2 3 1 2 3 3 1 10 2 8 2 3 3 1 10 3 6 9 3 8 10 1 8 5 1 6 4 1 7 1 3 6 9 3 8 10	12 7 2 0 3 0
8 6 1 2 4 1 6 2 4 3 3 2 4 2 6 3 1 8 1 4 3 1 5 1 3 3 5	3 8 4

Explanation of the first case:

The 10th and 11th request will be - (x) value mean the expected stroke was changed by a previous request:



P(a[6, 9]) = 3



P(a[8, 10]) = 0

Problem I: Help Me Boss

Time limit: 1s; Memory limit: 512 MB

You lead a team of n employees, each assigned a unique ID from 1 to n, with your ID being 1. Every employee, except you, has exactly one **direct** supervisor whose ID is smaller than theirs.

- If employee i directly supervises employee j, then employee i is considered a supervisor of employee j.
- If employee i supervises employee j, and employee j supervises employee k, then employee i is also considered a supervisor of employee k.

Each employee is assigned exactly one task. Employees can either complete their task or escalate it to one of their supervisors. An employee can handle multiple tasks, but the more tasks they take on, the more stressed they become. Specifically, each employee i has a stress factor f_i. If employee i handles m tasks, their total stress level is $f_i \times m$.

In all possible scenarios, determine how many employees have the potential to become the most stressed, meaning they end up with the highest total stress level among all n employees.

Input

- The first line contains a single positive integer n ($2 \leq n \leq 10^6$) - the number of employees.
- The second line contains n - 1 integers a_i ($2 \leq i \leq n$)$, where a_i represents the direct supervisor of employee i and $1 \leq a_i < i$.

- The third line contains n integers f_i ($1 \leq f_i \leq 10^{12}$), where f_i is the stress factor of employee i.

Output

Output a single integer — the number of employees who have the potential to become the most stressed.

Sample

Input	Output
3 1 1 2 1 1	1
4 1 1 2 1 1 1 1	4
5 1 1 3 3 2 1 1 1 4	3

Explanation for example 2: all employees have the have the potential to become the most stressed, if each employee completes their own task, the stress level for each employee will be equal to 1.

Explanation for example 3: employees 1, 3, 5 have the have the potential to become the most stressed. Here's how employee 3 could become the most stressed:

- Employee 1 completes his own task → Employee 1's total stress level = $f_1 \times 1 = 2$.
- Employee 2 completes his own task → Employee 2's total stress level = $f_2 \times 1 = 1$.
- Employee 3 completes three tasks: his own task, and task from employees 4 and 5 → Employee 3's total stress level = $f_3 \times 3 = 3$.
- Employees 4 and 5 don't handle any task → their total stress level = 0.

Problem J: Path Queries

Time limit: 1s; Memory limit: 512 MB

In graph theory, a tree is a connected undirected graph which does not have any cycles. A tree containing n vertices has exactly n - 1 edges. For every pair of vertices (u; v) in the tree, there is exactly one simple path between u and v. A simple path is a path which passes through each vertex at most once.

You are given a tree containing n vertices. These vertices are numbered from 1 to n, inclusive. You are also given q queries. In each query, you are given 3 distinct integers x; y and z. You need to determine if there exists a simple path starting at vertex x, passing through vertex y and ending at vertex z

Input

- The first line contains two integers n and q safety ($3 \le n \le 10^5$, $1 \le q \le 10^5$).
- In the next n - 1 lines, each contains two integers u and v safety ($1 \le u, v \le n$) indicating that two vertices u and v is connected by an edge.
- In the last q lines, each contains three distinct integers x, y, z ($1 \le x, y, z \le n$) describing a query.
- It is guaranteed that the given edges form a tree.

Output

- For each query, print **YES** if there exists a simple path satisfying the above conditons, otherwise print **NO**.

Sample

Input	Output
5 3 1 2 2 3 2 4 1 5 1 2 3 2 3 4 4 2 1	YES NO YES

Problem K: Ants

Time limit: 3s; Memory limit: 512 MB

Given an n x n map ($n \leq 100$) where each cell is **randomly filled** with a number. Over time, several ant nests are born on this map. Each ant nest occupies a cell in the map and is represented by the symbol '*****'. The number of ant nests is no more than 15. Additionally, there is always at least one cell in the map that does not contain the '*' symbol.

Initially, there is no relationship between the ant nests, and each nest belongs to a different ant species. A researcher wants to merge all these species into a single species. He plans to guide all ants from two different species to a location on the map where there are no ant nests to perform the merging process. Ants can move in four directions: Up, Down, Left, and Right, and they are not allowed to move diagonally. The movement cost is the sum of the numbers on the map along the path taken by the ants to the desired location. Ants can move into their own nest or into another species' nest without incurring any cost. After two species merge, all ants transform into a single species, and they can return to their original nests without additional cost.

Your task is to calculate the **minimum** total cost for the researcher to achieve his goal.

Input

- The first line contains an integer n ($1 \leq n \leq 100$).
- The next n lines represent the map, where each cell contains either a positive integer (less than or equal to 10^9) or the symbol '*'. It is guaranteed that there is at least one cell without the '*' symbol.

Output

- The minimum total cost required as described in the problem.

Sample

Input	Output
3 5 6 10 3 * 4 3 3 *	6

4 10 7 * 10 5 * 1 8 * 6 * 1 3 7 9 9	14

Explanation:

Explanation for Example 1: The two ant nests move to cell (3, 2) to perform the merging.

Explanation for Example 2:

- Suppose the ant species are initially labeled as A, B, C, and D. The map is as follows:

10 7 A 10
5 B 1 8
C 6 D 1
3 7 9 9

- A and B merge at cell (2,3) with a cost of 1 + 1 = **2**, forming species E:

10 7 E 10
5 E 1 8
C 6 D 1
3 7 9 9

- Next, E and D merge at cell (2,3) with a cost of 1 + 1 + 1 = **3**, forming species F:

10 7 F 10
5 F 1 8
C 6 F 1
3 7 9 9

- Finally, C and F merge at cell (2,3) with a cost of (1 + 5) + 1 + 1 + 1 = **9**.

The total cost is 2 + 3 + 9 = **14**.

In this example, all the merging processes happen at the same location, but in other cases, the locations might differ. Feel free to try other cases to explore different scenarios.

Problem L: Largest Number

Time limit: 3s; Memory limit: 512 MB

You are given an integer N. Arrange the numbers 1, 2,…, N (all written in decimal representation) in a way so that the resulting number, written in decimal expansion, is the largest. For example, if N = 10, the resulting number is 98765432110.

Given two integers N and K, find the K-th leftmost digit from such number representation, or -1 if the resulting number has less than K digits.

Input

- The first line contains an integer T ($1 \le T \le 100$), the number of test cases.
- Each following T lines, each contains two numbers N and K ($0 < N, K \le 10^{18}$).

Output

- For each test case, output the K-th leftmost digit of the resulting number on a single line, or -1 if that digit does not exist.

Sample

Input	Output
3 10 1 10 10 5 7	9 1 -1

Problem M: Marathon Tournament

Time limit: 3s; Memory limit: 512 MB

This year, an ICPC Regional Contest is going to be hosted in Hanoi. Together with a very tough contest to determine world finalists, the organizers decide to host a marathon event, giving contestants opportunities to explore the loveliness of Hanoi.

The venue of this marathon event consists of n junctions. These junctions are numbered from 1 to n. There are m bidirectional roads, which are numbered from 1 to m. The i-th one connects two junctions u_i and v_i and has a length of l_i. These roads guarantee that travelling between every pair of junctions is always possible.

The marathon tournament consists of multiple rounds. In each round, participants start at some junction, pass through several roads, and finish at some other junction. All participants run on exactly the same route, which is chosen by the organizers. The following conditions hold for all rounds:

- There is going to be at least one rounds.
- In all rounds, the starting junction differs from the finishing one.
- The starting junction of every round is the same as the finishing junction of the previous round.
- In all rounds, the organizers always choose a shortest path from the starting junction to the finishing junction. A shortest path is a path with minimum possible total length of roads

The organizers define the *difficulty set* of the tournament as the set of distinct values of rounds' lengths. Formally, a tournament consisting of k rounds can be represented as a sequence of junctions $x_0, x_1, \ldots, x_k$ so that:

- The first round starts at junction $x0$ and finishes at junction x_1,
- The second round starts at junction $x1$ and finishes at junction x_2,
- The third round starts at junction $x2$ and finishes at junction x_3,
- … ,
- The last round starts at junction xk-1 and finishes at junction x_k.

Let $d(a, b)$ be the distance of a shortest path from junction a to junction b. The *difficulty set* of the above tournament is the set of values which appears in the following sequence at least once: $d(x_0, x_1), d(x_1, x_2),\ldots, d(x_{k-1}, x_k)$.
To make an interesting tournament, the organizers have q plans of choosing routes. In each plan, the organizers have a set of numbers $s_1, s_2,\ldots, s_c$. The organizers would like to know if it is possible to make a tournament so that its *difficulty set* equals to this set.

Please help the organizers for a wonderful ICPC event!

Input

The first line of the input contains a single integer θ – the number of test cases. Then θ test cases follow, each is presented as below:

- The first line is an empty line.
- The second line contains three integers n, m and q ($1 \le n \le 400$, $0 \le m \le n(n-1)/2$, $1 \le q \le 2\times10^5$) – the number of junctions, the number of roads and the number of plans, respectively.
- In the next m lines, the i-th one contains three integers u_i, v_i and l_i ($1 \le u_i, v_i \le n$; $1 \le l_i \le 10^4$) representing the i-th road.
- In the last q lines, each contains an integer c ($1 \le c \le 2\times10^5$) followed by c integers $s_1, s_2,\ldots, s_c$ ($1 \le s_1 < s_2 < \ldots < s_c \le 10^7$) representing a plan.

It is guaranteed that:

- The sum of n over all test cases does not exceed 1600.
- The sum of c over all plans over all test cases does not exceed 8×10^5.

Output

- For each test case, print a single line containing exactly q words. The i-th one among them represents the answer for the i-th plan, which is either **Yes** (meaning that it is possible to make such a tournament) or **No** (no such tournament exists).

Sample

Input	Output
2 5 4 3 3 4 1 2 1 2 3 2 3 4 5 4 2 1 2 2 3 4 3 3 4 8 5 4 3 3 4 1 2 1 2 3 2 3 4 5 4 2 4 6 2 3 5 3 4 5 6	No Yes Yes Yes Yes Yes

Explanation:

Explanation first test case:

- In the second plan, one possible tournament is: 3 → 2 → 4. We have $d(3, 2) = 3$ and $d(2, 4) = 4$.
- In the third plan, one possible tournament is: 3 → 2 → 4 → 5 → 2. We have $d(3, 2) = 3$, $d(2, 4) = d(4, 5) = 4$ and $d(5, 2) = 8$.