2026 ICPC Nanchang Invitational and Jiangxi Provincial Collegiate Programming Contest

Contest Session

May 17, 2026

Problem List

A	The Difficulty of Marathon
B	UNO
C	Tourist
D	Eating Cherries
E	Critical Perfect
F	One Touch Drawing
G	The Meaning of the World
H	BABA BOOEY
I	Gotta Go Fast
J	Journey of GCD
K	Legacy Code
L	Little Z's Function
M	Kingdam of Construction

This problem set should contain 13 (thirteen) problems on 20 (twenty) numbered pages. Please inform a runner immediately if something is missing from your problem set.

Problem A. The Difficulty of Marathon

If you don't want to read the background of the problem, please skip directly to the last paragraph.

In 490 BC, after the Greek victory in the Battle of Marathon, the messenger Pheidippides ran approximately 40 kilometers from the Marathon Plain to Athens to announce the triumph, and died shortly after declaring victory. To commemorate this event, the marathon was introduced as an event at the first modern Olympic Games in Athens in 1896, with an initial distance of about 40 kilometers. The route was adjusted to 42.195 kilometers at the 1908 London Olympic Games, and this standard was officially adopted by World Athletics in 1921. The women's marathon became an official Olympic event at the 1984 Los Angeles Olympic Games.

Marathon can easily lead to extreme physical exhaustion. Runners often experience "hitting the wall" (glycogen depletion) between 30 and 35 kilometers, causing a breakdown of both physical strength and willpower. Long-distance running also poses injury risks to the knees, ankles, calves, and other body parts. Additionally, it is difficult to control the pace at the start; the second half of the race is a lonely mental struggle; weather changes affect physical comfort; improper timing of refueling can cause gastrointestinal discomfort and electrolyte imbalance. Participating in a marathon also requires high standards of daily systematic endurance and strength training. Participating without proper preparation carries a high risk of injury.

However, the biggest difficulty of marathon lies in the extremely low registration success rate. The success rate for popular marathons is often below 10%, and for beginners, it can be as low as 2%.

If you are a marathon beginner, what do you think is the biggest difficulty of running a marathon?

For each draw factor N and registration number M, you are considered successful if N divides M. Given the registration number and draw factor, determine whether you are selected.

Input

The first line contains a positive integer $T\,(1 \le T \le 100)$, representing the number of test cases.

The next T lines each contain two positive integers $M, N\,(1 \le M, N \le 100000)$, representing your registration number M and draw factor N.

Output

For each test case, output one line with an integer: output `1` if selected, and `0` if not selected.

Example

standard input	standard output
2 2026 2026 2 2025	1 0

Problem B. UNO

Troubadour_Ggmz and Fengmi123 are playing a special UNO game. UNO cards have four colors: red, yellow, green, and blue, denoted by `R`, `Y`, `G`, and `B`, respectively. Each color contains n cards, with ranks being any integers from 1 to n; there may be two cards with the same color and rank. The whole deck has $4n$ cards.

At the beginning of the game, Troubadour_Ggmz arranges all $4n$ cards in a row, then Fengmi123 can perform the following operation any number of times: choose two adjacent cards, and if they are **the same color** or **the same rank**, swap their positions.

The goal of the game is to rearrange all cards into a state that is both color-ordered and rank-ordered, i.e.:

- The color order is `R` $<$ `Y` $<$ `G` $<$ `B`, meaning all red cards appear before all yellow cards, all yellow cards appear before all green cards, and all green cards appear before all blue cards;
- For cards of the same color, the sequence of ranks must be non-decreasing.

Fengmi123 wants to know whether she can achieve the target state through a sequence of allowed operations (possibly zero).

Input

There are multiple test cases. The first line contains an integer t ($1 \le t \le 1000$), the number of test cases. For each test case:

The first line contains a positive integer n ($1 \le n \le 1000$).

Then $4n$ lines follow, each containing a positive integer x_i and a character c_i, where $1 \le x_i \le n$, $c_i \in \{\texttt{R}, \texttt{Y}, \texttt{G}, \texttt{B}\}$, indicating the rank and color of the i-th card in order. It is guaranteed that each color appears exactly n times, but **cards with the same color and rank may appear multiple times**.

It is guaranteed that the sum of n over all test cases does not exceed 1000.

Output

For each test case, if the target state can be reached, output a line containing "`YES`", otherwise output a line containing "`NO`".

You may output the answer in any case (upper or lower). For example, the strings "`yEs`", "`yes`", "`Yes`", and "`YES`" will all be recognized as positive responses.

Example

standard input	standard output
2 2 1 R 2 Y 2 R 1 G 1 Y 2 B 2 G 1 B 2 1 R 2 Y 1 Y 2 B 2 R 1 G 2 G 1 B	YES NO

Problem C. Tourist

The Construction Kingdom is a famous tourist attraction, attracting thousands of tourists every year. The kingdom consists of n towns and m directed roads. Each road has a length of 1. It is guaranteed that among the initial m roads, there are no duplicate roads and no road from a town to itself.

Later, due to the continuous prosperity of tourism, the Construction Kingdom decided to expand the roads. Starting from day 1, over the next k days, a new directed road from x to y will be built each day. Due to improper planning, it may happen that an already existing road is built, or a road from a town to itself is built.

Fengmi is a free-spirited tourist. She has q travel wishes and a desired distance w. Each wish has a start s and an end t (s_i and t_i may be the same). She wants to know, for each wish, the earliest day on which she can start from town s, and under the condition that she is allowed to pass through towns and roads multiple times, exactly travel a distance of w to reach town t, or report that it is impossible. Please help her solve this problem.

Input

The first line contains two integers n and m ($2 \le n \le 200$, $1 \le m \le n(n-1)$) — the number of towns and the number of initial roads.

Each of the next m lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), indicating a directed road from u_i to v_i.

The next line contains an integer k ($1 \le k \le n^2$) — the number of roads to be built in the following days.

Each of the next k lines contains two integers x_i and y_i ($1 \le x_i, y_i \le n$), meaning that on day i, a new directed road from x_i to y_i is built.

The next line contains two integers q and w ($1 \le q \le n^2$, $1 \le w \le 10^9$) — the number of wishes and the desired distance.

Each of the next q lines contains two integers s_i and t_i ($1 \le s_i, t_i \le n$) — the start and end of the i-th wish.

Output

For each wish, output a single integer — the earliest day that the wish can be fulfilled (i.e., after the road built on that day, there exists a walk of length exactly w from s to t). If it is never possible, output -1.

Example

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

Note

In the sample, the desired distance is $w = 8$.

- For the first wish, it can be proved that it is impossible.
- For the second wish, after the roads $4 \to 5$ and $5 \to 5$ are built, one can walk along the town sequence $[5, 5, 5, 5, 5, 5, 5, 5, 5]$ to reach the destination.
- For the third wish, initially one can walk along the town sequence $[1, 3, 1, 3, 1, 3, 1, 3, 1]$ to the destination.
- For the fourth wish, after the roads $4 \to 5$, $5 \to 5$ are built, one can walk along the town sequence $[4, 5, 5, 5, 5, 5, 5, 5, 5]$ to the destination.
- For the fifth wish, after the roads $4 \to 5$, $5 \to 5$, $2 \to 5$, $5 \to 6$ are built, one can walk along the town sequence $[4, 5, 5, 5, 5, 5, 5, 5, 6]$ to the destination.

Problem D. Eating Cherries

Jiangqiao loves eating cherries! Before the cherries ripened, he had already arrived at the cherry tree...

A cherry tree consists of n nodes (numbered 1 to n) and $n-1$ branches, which connect all the nodes together.

Jiangqiao classifies the cherries on the whole cherry tree into m types (numbered 1 to m) based on size, color, sweetness, etc., where node i has k_i types of cherries.

A **main branch** is the path from the root node to a certain leaf node without repeating any node.

That is, for root node u, the main branch from u to a leaf node v is a node sequence $[u = x_0, x_1, x_2, \ldots, x_s = v]$, where node x_i is connected to node x_{i+1} $(i < s)$ by a branch, and $x_i \neq x_j$ whenever $i \neq j$.

A **leaf node** refers to a node that is connected to only one branch with other nodes. Specifically, **the root node is not a leaf node**.

Obviously, for each leaf node, the main branch from the root to it is uniquely determined.

Jiangqiao can cast magic. **Each spell** can choose a leaf node and **arbitrarily change** the cherry types and the number of types on that node (including making it cherry-free).

He wants to know, for each node $1, 2, ..., n$ serving as the root, the **minimum number of spells** needed so that the **set of cherry types** on every main branch is the same.

Input

The first line contains two integers n and m ($2 \le n \le 2 \times 10^5$, $1 \le m \le 6$) — the number of nodes and the number of cherry types.

Each of the next $n-1$ lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \neq v_i$) — a branch on the tree.

Each of the next n lines describes a node. The i-th of these lines starts with a non-negative integer k_i ($0 \le k_i \le m$) — the number of cherry types at node i. If $k_i > 0$, it is followed by k_i distinct integers $t_{i,1}, t_{i,2}, \ldots, t_{i,k_i}$ ($1 \le t_{i,j} \le m$) — the cherry types.

It is guaranteed that all nodes are connected by the branches.

Output

A single line containing n non-negative integers, where the i-th integer is the answer when node i is the root.

Example

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

Problem E. Critical Perfect

Fengmi is challenging a high-difficulty level in a music game. There are n notes falling in order, and the i-th note has a base score a_i. When the player hits the first i notes, the **cumulative score** $s_i = \sum_{k=1}^{i} a_k$ determines the trigger condition for the combo effect.

Specifically, the i-th note has a target interval $[l_i, r_i]$ for the cumulative score. Only if s_i falls into this interval can the Critical Perfect combo effect performance be triggered; otherwise, the effect chain will be interrupted.

As the owner of the "Area Amplification" item, Fengmi can use this item any number of times. Each use allows her to **select a contiguous segment of notes** $[l, r]$ and increase the base score of each note in this segment by 1. The number of uses affects the final level score, so Fengmi wants to achieve a Critical Perfect combo throughout the entire level with the minimum number of uses.

Please help her calculate the minimum number of times the Area Amplification item must be used so that for every note i, the cumulative score s_i satisfies $l_i \le s_i \le r_i$, or report that no solution exists.

Input

There are multiple test cases. The first line contains an integer t ($1 \le t \le 5000$), the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 5000$) — the number of notes.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the base score of each note.

Each of the next n lines contains two integers l_i and r_i ($0 \le l_i \le r_i \le 10^{18}$) — the target interval for the cumulative score after note i.

It is guaranteed that the sum of n over all test cases does not exceed 5000.

Output

For each test case, output a single integer — the minimum number of item uses. If it is impossible, output -1.

Example

standard input	standard output
2 3 6 2 8 8 16 16 32 28 36 2 0 0 1 1 0 0	4 -1

Note

For the first test case in the sample, one feasible operation scheme is as follows: use the item 4 times on the interval $[1, 3]$. The original base score sequence becomes $[10, 6, 12]$, and the cumulative scores become $[10, 16, 28]$, which satisfies the conditions. The total number of operations is 4. It can be proved that no scheme with fewer operations exists.

For the second test case in the sample, it can be proved that no feasible operation scheme exists.

Problem F. One Touch Drawing

Fengmi recently became obsessed with a small game called "One Touch Drawing". In this game, the map is an undirected graph containing n vertices and m edges. The graph may contain multiple edges and self-loops, and is not necessarily connected. The player needs to start from some vertex, walk along the edges of the graph, passing each edge exactly once, and finally stop at any vertex. After all edges have been passed exactly once, the game ends immediately.

This game adds a special "teleportation" mechanism: there are k distinct pairs of "jump points". Each pair (a, b) has the following property: whenever the player reaches vertex a or vertex b by traversing an edge, they are immediately teleported to the other vertex in the pair, and continue moving after the teleportation.

Special attention is required:

- The starting point does not trigger teleportation (i.e., if the starting point is a jump point, no teleportation occurs when starting from that point);
- The ending point also does not trigger teleportation (i.e., if the last step reaches a jump point, the game ends immediately without teleportation).

Fengmi wants to know whether there exists a path such that the player starts from some vertex, passes each edge exactly once, and strictly follows the above jumping rules. If it exists, output any such path (represented as a sequence of visited vertices), or report that no solution exists.

Input

The first line contains a positive integer t ($1 \le t \le 5 \times 10^4$) — the number of test cases.

For each test case:

The first line contains three positive integers n, m, and k ($2 \le n \le 10^5$, $1 \le m \le 10^6$, $1 \le k \le \lfloor \frac{n}{2} \rfloor$) — the number of vertices, the number of edges, and the number of jump point pairs, respectively.

Each of the next m lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$), describing an undirected edge (multiple edges and self-loops may exist).

Each of the next k lines contains two integers a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$), indicating that (a_i, b_i) is a pair of jump points. It is guaranteed that all a, b are distinct (i.e., each vertex appears in at most one jump pair).

It is guaranteed that the sum of n over all test cases does not exceed 10^5, and the sum of m does not exceed 10^6.

Output

For each test case, if such a path exists: Output an integer L on the first line — the number of visited vertices (including the start and end); On the second line, output L integers separated by spaces, representing the sequence of vertex indices.

If it does not exist, output a single line containing -1.

Note: The path sequence must record the actual vertices reached (including vertices reached after teleportation). For example, going from vertex 1 to 2 (assuming 2 is a jump point and teleports to 3), then from 3 to 4, the output sequence would be $[1, 2, 3, 4]$.

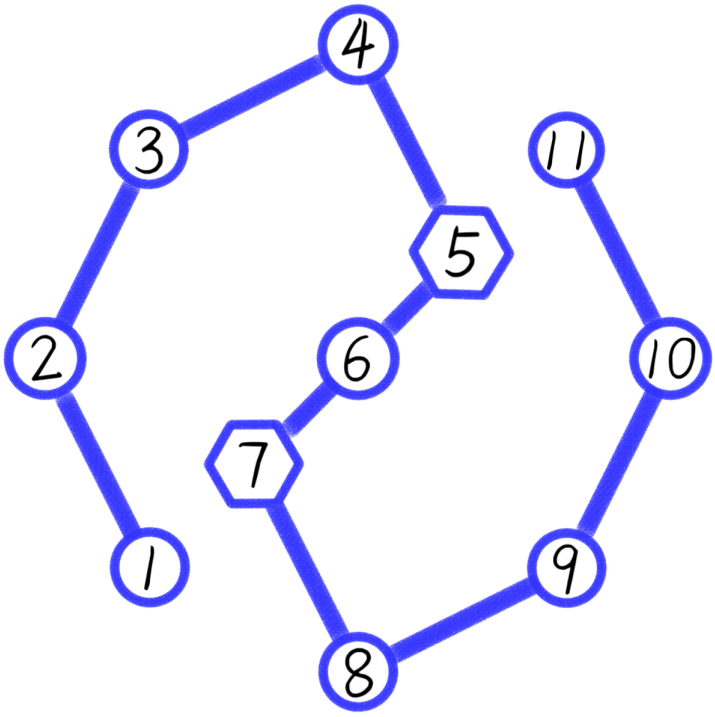
Example

standard input	standard output
2 11 10 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 5 7 9 24 1 1 2 1 3 1 3 1 4 1 5 1 5 1 6 1 7 1 7 1 8 1 9 1 9 2 3 2 3 3 4 4 5 4 5 5 6 6 7 6 7 7 8 8 9 8 9 9 2 9 3	13 1 2 3 4 5 7 6 5 7 8 9 10 11 30 5 6 7 8 9 3 4 5 4 1 9 3 2 9 3 2 1 9 3 1 8 9 3 1 7 6 1 5 1 7

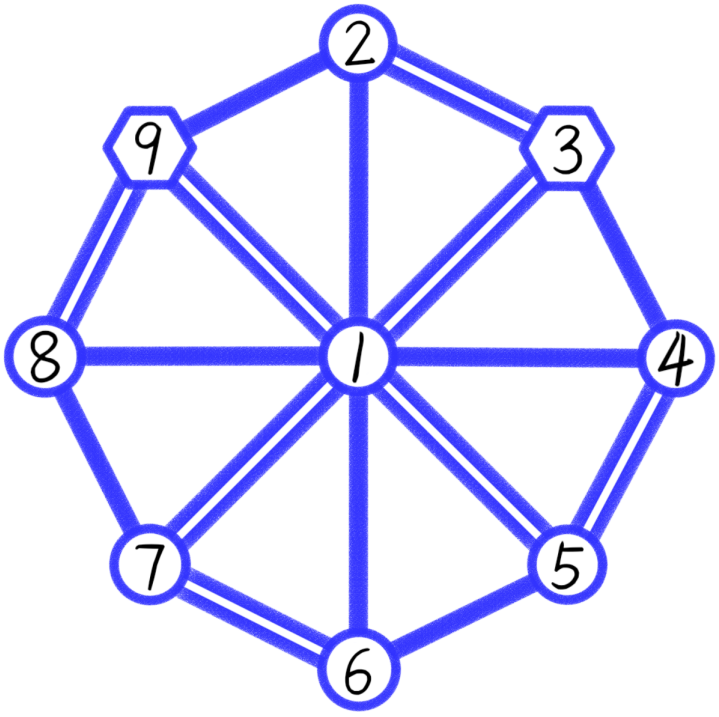
Note

For the first test case in the sample, $(5, 7)$ is a jump point pair, as shown in the figure.

A valid path is: $1 \to 2 \to 3 \to 4 \to 5$ (teleports) $\to 7 \to 6 \to 5$ (teleports) $\to 7 \to 8 \to 9 \to 10 \to 11$.



For the second test case in the sample, as shown in the figure:



Problem G. The Meaning of the World

The meaning of the world must lie outside the world
All things in the world are as they are and happen as they happen
There is no value in the world
—— Tractatus Logico-Philosophicus

Fengmi gives you a string s of length n, consisting only of '`(`' and '`)`'.

You need to support q operations:

1. Flip the k-th bracket, i.e., change '`(`' to '`)`' and '`)`' to '`(`'.
2. Query the length of the longest valid parentheses substring in the interval $[l, r]$.

Formally, you need to compute:

$$f(l, r) = \max(\{j - i + 1 \mid l \le i \le j \le r \wedge \text{valid}(i, j)\} \cup \{0\})$$

where $\text{valid}(i, j)$ indicates that the substring $s[i..j]$ is a valid parentheses sequence.

In this problem, a valid parentheses sequence is defined as follows:

1. The empty sequence is a valid sequence.
2. If A is a valid sequence, then `(`A`)` is also a valid sequence.
3. If A and B are both valid sequences, then AB is also a valid sequence.

Input

The first line contains two integers n and q ($1 \le n, q \le 2 \times 10^5$) — the length of the bracket string and the number of operations.

The second line contains a string s of length n, consisting only of the characters '`(`' and '`)`'. The given string is not guaranteed to be a valid parentheses sequence.

The next q lines each describe one of the following two operations:

- `1 k` ($1 \le k \le n$): flip the k-th bracket.
- `2 l r` ($1 \le l \le r \le n$): query the length of the longest valid parentheses substring in the interval $[l, r]$.

Output

For each operation of type 2, output a single integer on its own line — the answer.

Examples

standard input	standard output
8 5 ()(()()) 2 1 8 1 4 2 1 8 1 7 2 1 8	8 4 4
12 6 ((())(()())) 2 1 12 1 3 2 1 12 1 8 2 1 12 2 4 9	12 10 12 2

Note

Sample 1:

- Initial string: `()(()())`.
- First query on interval $[1, 8]$: the longest valid parentheses substring is `()(()())` (positions $1 - 8$), length 8.
- Flip the 4th bracket: `()())())`.
- Second query on interval $[1, 8]$: the longest valid parentheses substring is `()()` (positions $1 - 4$), length 4.
- Flip the 7th bracket: `()())(()`.
- Third query on interval $[1, 8]$: the longest valid parentheses substring is `()()` (positions $1-4$), length 4.

Sample 2:

- Initial string: `((())(()())`.
- First query on interval $[1, 12]$: the longest valid parentheses substring is the entire string, length 12.
- Flip the 3rd bracket: `((()))(()())`.
- Second query on interval $[1, 12]$: the longest valid parentheses substring is `())(()())` (positions $3 - 12$), length 10.
- Flip the 8th bracket: `((()))())())`.
- Third query on interval $[1, 12]$: the longest valid parentheses substring is `((()))())())` (positions $1 - 12$), length 12.
- Fourth query on interval $[4, 9]$: the longest valid parentheses substring is `()` (positions $4 - 5$), length 2.

Problem H. BABA BOOEY

Alice and Bob take turns playing a game, with Alice going first. Before the first move, there is an array of non-negative integers $a_1, a_2, \cdots, a_n$ of length n, along with a positive integer k. In each move, the current player chooses two integers x, y such that $0 \le x \le k$, $1 \le y \le n$, and $x \le a_y$. Then, k is updated to x, and a_y is updated to $a_y - x$. A player who chooses $x = 0$ loses immediately. Who has the winning strategy?

Input

There are multiple test cases. The first line contains an integer t ($1 \le t \le 10^5$) — the number of test cases.

For each test case:

The first line contains two integers n and k ($1 \le n \le 10^6$, $1 \le k \le 10^9$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$).

It is guaranteed that the sum of n over all test cases in a single input file does not exceed 2×10^6.

Output

For each test case, output a single string on a new line.

If the first player (Alice) has a winning strategy for the game described, output "`Alice`"; otherwise, output "`Bob`".

Example

standard input	standard output
3 2 2 1 2 4 5 15 15 8 9 11 2 4 2 9 12 0 15 9 14 5 6 12	Alice Alice Bob

Note

In the first test case, the only winning first moves for Alice are $(x, y) = (1, 1)$ and $(x, y) = (1, 2)$.

Problem I. Gotta Go Fast

Have you ever played Eggy Party? inf loves playing Eggy Party.

Four eggies are stacked into a four-story eggy tower, with the bottom initially at coordinate 0. From bottom to top, they are sequentially labeled A, B, C, and D, where D is the topmost eggy.

The total length of the track is l, which consists of l segments of length 1. Segment i connects coordinates $i-1$ and i ($1 \le i \le l$).

The terrain of the i-th segment is given by the character S_i. If the eggy tower walks from coordinate $i-1$ to coordinate i, the cost is:

- Flat ground `0`: t_0 seconds;
- Mud pit `1`: t_1 seconds;
- Speed pad `2`: t_2 seconds.

The eggy tower can perform two types of operations:

- **Normal Advance:** If the current eggy tower is at coordinate $x < l$, it can advance forward to coordinate $x+1$. The time cost of this advance **depends only on the terrain of segment** $x+1$.
- **Super Start:** If there is an eggy tower containing j eggies at coordinate x ($j \ge 2$), the bottommost eggy can instantly throw the $j-1$ eggies above it to coordinate $\min(l, x+d)$, where d is a given positive integer. This operation does not trigger any terrain time cost, i.e., it takes 0 seconds. After the operation, the bottommost eggy at the original position is eliminated, leaving that position empty; a new eggy tower containing $j-1$ eggies is formed at coordinate $\min(l, x+d)$, and the vertical order of the remaining eggies stays the same.

Please compute the minimum time required for the initially topmost eggy D to reach the destination coordinate l.

Input

The first line contains two positive integers l and d ($1 \le d < l \le 2 \times 10^5$).

The second line contains three positive integers t_0, t_1, t_2 ($1 \le t_2 < t_0 < t_1 \le 10^9$).

The third line contains a string S of length l, consisting only of the characters `0`, `1`, and `2`.

Output

Output a single integer representing the minimum total time.

Examples

standard input	standard output
10 2 10 20 5 0000000000	40
12 3 10 100 1 111000222000	3
10 2 10 100 1 2220110000	13
8 3 10 100 1 00000000	0

Note

For Example 1: One optimal strategy is to consecutively trigger Super Start 3 times to reach coordinate 6, and then walk through the final 4 flat ground segments. The total time cost is $4 \times 10 = 40$.

For Example 2: One optimal strategy is to first consecutively trigger Super Start twice to reach coordinate 6, walk through 3 speed pads to reach coordinate 9, and finally trigger Super Start again to reach the destination. The total time cost is $1 + 1 + 1 = 3$.

For Example 3: One optimal strategy is to first walk through the first 3 speed pads and 1 flat ground segment to reach coordinate 4, and then consecutively trigger Super Start 3 times to reach the destination. The total time cost is $1 + 1 + 1 + 10 = 13$.

For Example 4: Consecutively trigger Super Start 3 times. The coordinate changes as $0 \to 3 \to 6 \to 8$ (the last time triggers the $\min(8, 6 + 3)$ limit). Walking is not needed throughout the process, and the total time cost is 0.

Problem J. Journey of GCD

In the magical kingdom of Numerian, there are n cities, each with its own magical power level a_i. The king has built bidirectional roads between every pair of cities. For the road connecting city u and city v, its Guardian Charm Degree (GCD) equals $\gcd(a_u, a_v)$, which is the greatest common divisor of their magical power levels.

For a traveler moving from city x to y, the Magical Immunity Number (MIN) of the journey is the minimum GCD among all roads taken. The royal advisors consider a journey "well-immune" if and only if its MIN is maximized.

You are tasked with answering q queries from the king. For each query (x, y), output the maximum possible MIN among all paths from city x to city y.

In particular, if $x = y$, the traveler never leaves the city, so the MIN is simply the city's own magical power a_x.

Input

The first line contains two integers n and q ($1 \le n, q \le 10^6$).

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^6$), representing the magical power of each city.

Each of the next q lines contains two integers x and y ($1 \le x, y \le n$), representing a query.

Output

For each query, output a single integer representing the maximum possible MIN.

Example

standard input	standard output
6 3 6 15 22 33 35 63 5 6 3 5 2 2	7 3 15

Note

For the sample, the magical power levels are $a = [6, 15, 22, 33, 35, 63]$.

Query 1: The optimal path is $5 \to 6$, and the answer is $\gcd(35, 63) = 7$.

Query 2: The optimal path is $3 \to 4 \to 6 \to 5$. The edge values are $\gcd(22, 33) = 11$, $\gcd(33, 63) = 3$, and $\gcd(63, 35) = 7$, so the answer is $\min(11, 3, 7) = 3$.

Query 3: Since $x = y$, the traveler never leaves the city, so the answer is $a_2 = 15$.

Problem K. Legacy Code

You have a piece of legacy code:

for $t = 1$ **to** k **do**
$a_{x_1} \leftarrow (a_{u_1} \times a_{v_1}) \bmod MOD$
$a_{x_2} \leftarrow (a_{u_2} \times a_{v_2}) \bmod MOD$
$a_{x_3} \leftarrow (a_{u_3} \times a_{v_3}) \bmod MOD$
$\vdots$
$a_{x_m} \leftarrow (a_{u_m} \times a_{v_m}) \bmod MOD$
end for

where $MOD = 998\,244\,353$ and $x_i, u_i, v_i \in \{1, \ldots, n\}$ for $i = 1, \ldots, m$.

Given the initial values of the variables $a_1, a_2, \ldots, a_n$, compute their final values after the loop runs k times. You need to answer q such queries.

Input

The first line contains three integers n, m and q ($1 \le n \le 200$, $1 \le m \le 2 \times 10^5$, $1 \le q \le 200$).

The next m lines each contain three integers x_i, u_i, v_i ($1 \le x_i, u_i, v_i \le n$).

Each of the next q lines represents a query. Each query contains $n + 1$ integers:

- the number of iterations k ($1 \le k \le 10^6$),
- the initial values of the n variables $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 998\,244\,353$).

Output

For each query, output a single line containing n integers — the final values of $a_1, a_2, \ldots, a_n$ after executing the code, each modulo $998\,244\,353$.

Example

standard input
7 8 4 3 1 2 4 3 3 5 6 7 7 1 3 2 2 2 6 6 2 1 5 6 7 7 7 1 2 1 1 1 1 1 1 2 1 2 3 4 5 6 7 3 1 1 0 1 0 1 0 100 11 45 14 19 19 8 10
standard output
1 1 2 4 1 1 16 36864 16 4032 16257024 96 384 227524445 0 1 1 1 0 1 1 244858606 337827833 584144609 343808937 412777304 279017446 926699892

Problem L. Little Z's Function

Little Z was obsessed with number-theoretic functions such as Euler's totient function φ* back in high school.

One day, Little Z decides to play a game with his robotic friend. First, Little Z gives a permutation† p of length n. Then he asks his friend multiple queries. For a given interval $[l, r]$ $(1 \le l < r \le n)$, find the maximum value of $\varphi(p_i \times p_j)$ where $l \le i < j \le r$.

Formally, for each query with two integers l, r, you need to compute:

$$\max_{l \le i < j \le r} \varphi(p_i \times p_j)$$

Little Z has solved exactly 0 queries. Please help him compute the remaining q queries.

Input

The first line contains a positive integer n $(2 \le n \le 5 \times 10^5)$ — the length of the permutation.

The second line contains n distinct integers $p_1, p_2, \ldots, p_n$ $(1 \le p_i \le n)$ — the given permutation p.

The third line contains a positive integer q $(1 \le q \le 5 \times 10^5)$ — the number of queries.

Each of the next q lines contains two integers l_i and r_i $(1 \le l_i < r_i \le n)$ — the endpoints of the i-th query.

Output

For each query, output a single integer — the maximum value.

Example

standard input	standard output
6 3 2 5 4 1 6 8 4 5 2 6 3 5 2 3 3 4 1 5 4 5 2 6	2 8 8 4 8 8 2 8

*$\varphi(n)$ denotes the number of positive integers not exceeding n that are coprime to n.

†A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array), and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

Problem M. Kingdam of Construction

You are managing the valve matrix of the "Kingdam of Construction".

Given an $n \times m$ matrix a (both n and m are even), where all elements of a are distinct positive integers between 1 and $n \times m$.

You can perform the following two types of operations in any order:

- Choose row i ($1 \le i \le n$) and reverse it left-to-right, i.e. for all $1 \le j \le \frac{m}{2}$, swap $a_{i,j}$ and $a_{i,m-j+1}$;
- Choose column j ($1 \le j \le m$) and reverse it top-to-bottom, i.e. for all $1 \le i \le \frac{n}{2}$, swap $a_{i,j}$ and $a_{n-i+1,j}$.

Determine whether it is possible to transform a into the $n \times m$ target matrix b after a finite number of operations, where b is filled in row-major order with $1, 2, \ldots, n \times m$, i.e. $b_{i,j} = (i-1) \times m + j$. If it is possible, construct an operation sequence with **no more than** 9961 **operations** (it is also allowed to perform no operation, i.e. an empty operation sequence). It can be proved that if an operation sequence exists, then there also exists one with no more than 9961 operations.

Input

The first line contains a positive integer t ($1 \le t \le 25$), representing the number of test cases. For each test case:

The first line contains two positive integers n and m ($2 \le n \le 134$, $2 \le m \le 134$), representing the number of rows and columns of matrix a.

The next n lines each contain m positive integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$, representing the elements of matrix a.

It is guaranteed that both n and m are even; it is also guaranteed that every integer from 1 to $n \times m$ appears exactly once in matrix a.

Output

For each test case:

If it is possible to transform matrix a into the target matrix b in finitely many operations, print "`Yes`" on the first line.

Then print one non-negative integer *cnt* on the next line, representing the number of operations. You must ensure that $0 \le cnt \le 9961$ (for each test case; that is, there is no restriction on the sum of *cnt* over all test cases).

Then print *cnt* lines, each containing two integers *op* and *idx* ($op \in \{0, 1\}$, $1 \le idx \le n$ if $op = 0$, otherwise $1 \le idx \le m$), where $op = 0$ means reversing row *idx*, and $op = 1$ means reversing column *idx*.

If no valid sequence of operations exists, output only one line containing "`No`".

You can output the answer in any case (upper or lower). For example, the strings "`yEs`", "`yes`", "`Yes`", and "`YES`" will be recognized as positive responses.

Example

standard input	standard output
4 2 4 4 3 2 1 8 7 6 5 2 4 1 3 2 4 5 6 7 8 4 2 1 2 3 4 7 8 5 6 4 4 4 15 14 16 8 6 11 12 5 7 10 9 13 3 2 1	Yes 2 0 1 0 2 Yes 5 1 2 0 1 1 2 0 1 1 2 No No

Note

For the second sample test case, the operation process is as follows:

$$\begin{bmatrix}1 & 3 & 2 & 4\\ 5 & 6 & 7 & 8\end{bmatrix} \xrightarrow{\text{column, 2}} \begin{bmatrix}1 & \mathbf{6} & 2 & 4\\ 5 & \mathbf{3} & 7 & 8\end{bmatrix} \xrightarrow{\text{row, 1}} \begin{bmatrix}\mathbf{4} & \mathbf{2} & \mathbf{6} & \mathbf{1}\\ 5 & 3 & 7 & 8\end{bmatrix}$$
$$\xrightarrow{\text{column, 2}} \begin{bmatrix}4 & \mathbf{3} & 6 & 1\\ 5 & \mathbf{2} & 7 & 8\end{bmatrix} \xrightarrow{\text{row, 1}} \begin{bmatrix}\mathbf{1} & \mathbf{6} & \mathbf{3} & \mathbf{4}\\ 5 & 2 & 7 & 8\end{bmatrix} \xrightarrow{\text{column, 2}} \begin{bmatrix}1 & \mathbf{2} & 3 & 4\\ 5 & \mathbf{6} & 7 & 8\end{bmatrix}.$$

For the third and fourth sample test cases, the target matrix b is $\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\\ 7 & 8\end{bmatrix}$ and $\begin{bmatrix}1 & 2 & 3 & 4\\ 5 & 6 & 7 & 8\\ 9 & 10 & 11 & 12\\ 13 & 14 & 15 & 16\end{bmatrix}$, respectively. It can be proved that both cases have no solution.