The 2nd Universal Cup

Uni
Cup

Stage 13: Shenyang

December 9-10, 2023

This problem set should contain 13 problems on 25 numbered pages.

Based on

icpc.foundation

International Collegiate Programming Contest (ICPC)

Hosted & Prepared by

东北大学
NORTHEASTERN UNIVERSITY
1923

Problem A. Intro: Dawn of a New Era

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Dear friends from across China, welcome to Shenyang! As the problem setters, along with the entire staff, we are deeply honored to have all of you here to enjoy a well-prepared event and witness the ninth consecutive year that Northeastern University (NEU) has hosted the ICPC Shenyang Regional Contest.

The past years have been challenging for most individuals. Human society underwent the COVID-19 pandemic, yet we fought against it and triumphed. Humanity continues to be tested when it seems like the Night Howlers have eroded some people's minds. And for the lovely university where we are currently situated, it has just successfully concluded its centennial celebration, turning over a new page for the next century. So, those are what inspired us to name this intro problem. We sincerely hope one can strive constantly for dreams rather than being willing to become the tears of the times. We also hope one can heal the world with new-age technologies, not to steer our civilization to a sunset!

As an integral part of Northeastern University's centennial celebration, the drone display utilized various colors to illuminate the night sky, creating pieces of stunning aerial artwork. In the intro problem, you are going to meet with a seemingly NP-hard task related to it.

Drone Display by Northeastern University

Suppose that there are n scenes in the drone display. The palette of each scene can be described by a set of integers identifying different colors, and we say the *main color* of a scene is the color with the largest integer in its palette.

The drone operator intends to arrange the orders of the scenes. For each pair of **adjacent** scenes after the reordering, if the main color of the previous scene is one of the colors present in the palette of the following scene, a *transition* will be contributed. Can you help construct an arrangement of scenes such that the number of transitions is maximized?

More formally, let S_i be the set of integers which describes the palette of the i-th scene. You need to construct a permutation $p_1, p_2, \ldots, p_n$ such that $\sum_{i=1}^{n-1} \left[\max\{S_{p_i}\} \in S_{p_{i+1}}\right]$ is maximized among all the permutations of length n, where $\left[\max\{S_{p_i}\} \in S_{p_{i+1}}\right]$ is 1 when $\max\{S_{p_i}\} \in S_{p_{i+1}}$ and 0 when $\max\{S_{p_i}\} \notin S_{p_{i+1}}$.

Recall that a permutation of length n is a sequence of n integers in which every integer from 1 to n appears exactly once.

Input

The first line contains an integer n ($2 \le n \le 10^5$), denoting the number of scenes in the drone display.

In the i-th of the next n lines, an integer m_i ($m_i \geq 1$) comes first, denoting the number of colors in the palette of the i-th scene. Then m_i distinct integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m_i}$ ($0 \leq a_{i,j} \leq 10^9$) follow, each identifying a color in the palette.

It is guaranteed that the sum of m_i over $i = 1, 2, \ldots, n$ does not exceed 2×10^5.

Output

In the first line, output an integer indicating the maximum attainable number of transitions among all arrangements of scenes.

In the second line, output a permutation $p_1, p_2, \ldots, p_n$ denoting that the p_i-th scene is played during the i-th period of time in chronological order. Your construction should maximize the number of transitions. If there are multiple arrangements of scenes that maximize the number of transitions, you may choose any to output its corresponding permutation.

Examples

standard input	standard output

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

3
1 1
1 2
1 3
	0
	1 2 3

Note

In the first sample case, scene 4 to scene 2, scene 2 to scene 3, scene 1 to scene 5 can all contribute a transition.

In the second sample case, no scenes share the common color. Thus, any permutation of length 3 will be acceptable.

Problem B. Turning Permutation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

A permutation of length n is a sequence of n integers in which every integer from 1 to n appears exactly once. For a permutation $p_1, p_2, \ldots, p_n$ of length n, let q_i denote the position where i appears, i.e., $p_{q_i} = i$. If for every $i = 2, 3, \ldots, n-1$ we have $(q_i - q_{i-1})(q_i - q_{i+1}) > 0$, then the permutation $p_1, p_2, \ldots, p_n$ is called a *turning permutation*.

Now given n and k, you need to find the k-th lexicographically smallest turning permutation of length n, or report that the number of turning permutations of length n is less than k.

To determine which of the two permutations of length n is lexicographically smaller, we compare their first elements. If they are equal, we compare the second, and so on. If we have two different permutations x and y of length n, then x is lexicographically smaller if $x_i < y_i$, where i is the first index at which the permutations x and y differ.

Input

The only line contains two integers n ($3 \le n \le 50$) and k ($1 \le k \le 10^{18}$), denoting the length of the permutation and the ranking position of the desired turning permutation in the lexicographically sorted list of all the turning permutations of length n, respectively.

Output

If the number of turning permutations of length n is less than k, output -1 in one line. Otherwise, output the k-th lexicographically smallest turning permutation of length n in one line.

Examples

standard input	standard output
`3 2`	`2 1 3`
`3 5`	`-1`
`4 6`	`3 1 2 4`
`4 11`	`-1`

Note

There are a total of 4 turning permutations of length 3, arranged in lexicographically ascending order: $[1,3,2]$, $[2,1,3]$, $[2,3,1]$, $[3,1,2]$. Therefore, for the first sample case, the 2nd lexicographically smallest turning permutation is $[2,1,3]$, and for the second sample case, the answer is -1.

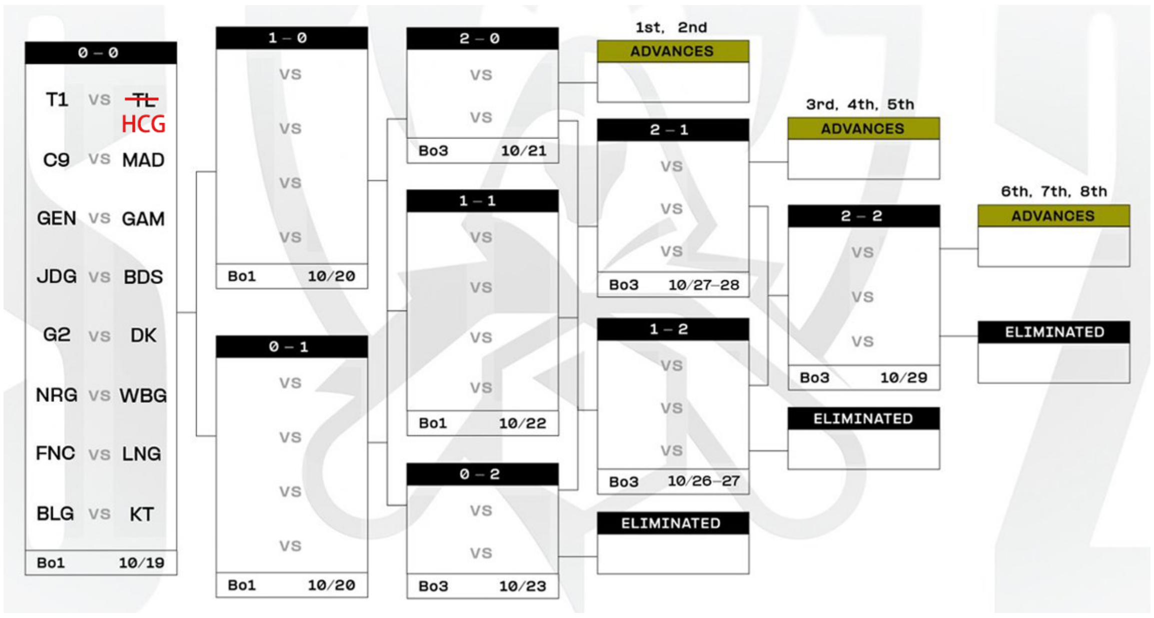
There are a total of 10 turning permutations of length 4, arranged in lexicographically ascending order: $[1,3,2,4]$, $[1,3,4,2]$, $[2,1,4,3]$, $[2,4,1,3]$, $[2,4,3,1]$, $[3,1,2,4]$, $[3,1,4,2]$, $[3,4,1,2]$, $[4,2,1,3]$, $[4,2,3,1]$. Therefore, for the third sample case, the 6th lexicographically smallest turning permutation is $[3,1,2,4]$, and for the fourth sample case, the answer is -1.

Problem C. Swiss Stage

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

After five years, the League of Legends World Championship (Worlds) returns to South Korea. This pinnacle event of the LoL esports season is scheduled to take place from October to November in the Land of the Morning Calm.

Worlds 2023 brings forth an exciting new format, featuring the Swiss Stage, along with new qualification paths for teams.



The Swiss Stage of Worlds 2023
(Credit: Riot Games)

In the Swiss Stage of Worlds 2023, there are 5 rounds in total for the 16 participating teams. In each round, the teams will be paired based on each team's win-loss record, where the teams with the same number of wins and losses will play each other. Rounds related to advancement and elimination, i.e., rounds for the teams accumulating two wins or two losses, will adopt the best-of-three rule, where the team who first defeats the other team in two out of three games wins the round, and the other loses (If one team defeats the other team consecutively in the first game and the second game, there will be no need for the third game). Other rounds will adopt the best-of-one rule, where the team who defeats the other team in the only game wins the round, and the other loses. The teams accumulating three wins advance to the next stage, while those accumulating three losses bid farewell to the tournament.

Now team HCG has x wins and y losses, and you, the coach of team HCG, would like to know the **minimum** number of **additional** games that team HCG needs to play to advance to the next stage.

Input

The only line contains two integers x and y ($0 \le x, y \le 2$), denoting the current number of wins and losses of team HCG.

Output

Output a single integer, denoting the minimum number of additional games that team HCG needs to play to advance to the next stage.

Examples

standard input	standard output
0 1	4
1 2	4

Note

In the first sample case, team HCG can win two rounds with the best-of-one rule and one round with the best-of-three rule to advance, with at least 4 additional games.

In the second sample case, team HCG must win two rounds with the best-of-three rule consecutively to advance, with at least 4 additional games.

Problem D. Dark LaTeX vs. Light LaTeX

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

In the not-so-distant future, LaTeX, the venerable typesetting system that had been a staple of academia and publishing for decades, found itself evolving into a couple of derivatives — "Dark LaTeX" and "Light LaTeX". These derivatives primarily optimized the appearance from different angles to adapt to screenless display technology.

For compatibility trial, the LaTeX lab has obtained two non-empty strings containing only lowercase English letters through analysis — the Dark LaTeX string $S = s_1 s_2 \cdots s_{|S|}$ and the Light LaTeX string $T = t_1 t_2 \cdots t_{|T|}$, where $|S|$ denotes the length of S, and $|T|$ denotes the length of T. An integer quadruple (p, q, u, v) is considered *transferrable* if and only if $1 \le p \le q \le |S|$, $1 \le u \le v \le |T|$, and $s_p s_{p+1} \cdots s_q t_u t_{u+1} \cdots t_v$ is a square string.

Your task is to help the lab find out the number of transferrable quadruples.

Recall that a square string is a string of even length in which the first half is identical to the second half. For example, "`aaaa`" and "`abcabc`" are square strings, while "`aaa`" and "`abcabd`" are not.

Input

The input consists of two lines, where the first line contains the Dark LaTeX string S, and the second line contains the Light LaTeX string T.

It is guaranteed that both S and T consist only of lowercase English letters and their lengths do not exceed 5 000.

Output

Output an integer in one line, indicating the number of transferrable quadruples.

Examples

standard input	standard output
`abab` `ab`	`8`
`abab` `abaaab`	`29`

Note

In the first sample case, the transferrable quadruples are $(1,1,1,1)$, $(1,2,1,2)$, $(1,3,2,2)$, $(2,2,2,2)$, $(2,4,1,1)$, $(3,3,1,1)$, $(3,4,1,2)$, $(4,4,2,2)$.

Problem E. Sheep Eat Wolves

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Farmer John has x sheep and one day, y wolves came to the river where he was herding the sheep.

For safety reasons, Farmer John wants to immediately transport all x sheep back to his house on the other side of the river. There is a boat on the riverbank that Farmer John can use to transport up to p animals with him at a time. Farmer John can only use this boat to transport animals back and forth between the two riverbanks.

However, if there is a group of animals that are not under Farmer John's supervision, and this group includes both wolves and sheep, and the number of wolves is strictly greater than the number of sheep plus q, the wolves will eat the sheep. A group of animals can be under Farmer John's supervision if and only if they are on the boat or on the same side of the river as Farmer John.

Now Farmer John wants to know the minimum number of trips required to safely transport all the sheep back to his house on the other side of the river, or determine if it is impossible.

Input

The only line contains four integers, x ($1 \le x \le 100$) denoting the number of sheep, y ($1 \le y \le 100$) denoting the number of wolves, p ($1 \le p \le 100$) denoting the maximum number of animals that can be transported in one trip with Farmer John, and q ($0 \le q \le 100$) denoting the threshold at which the wolves will eat the sheep when the number of wolves in a group not under Farmer John's supervision is strictly greater than the number of sheep plus q.

Output

Output a line containing a single integer, indicating the minimum number of trips required to safely transport all x sheep back to Farmer John's house on the other side of the river. If it is impossible to safely transport all x sheep back home, output -1 in one line.

Examples

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

Note

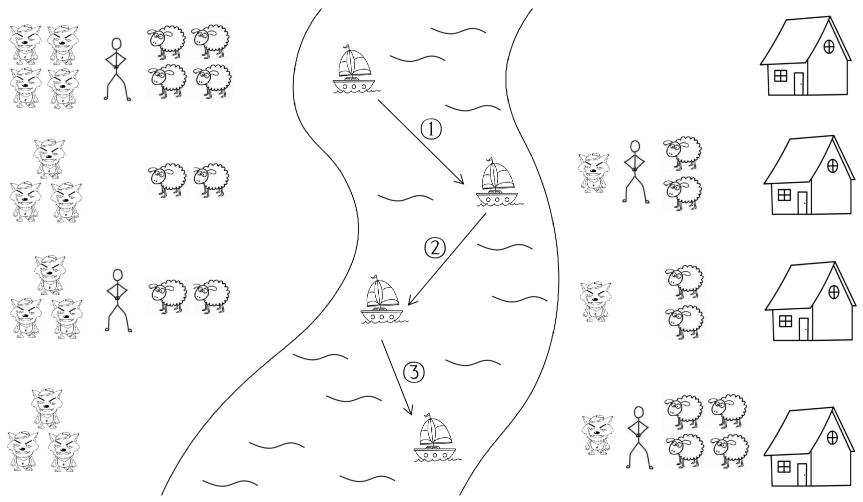


Figure: One possible solution for the first sample case

Problem F. Ursa Minor

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Pixel Art: Ambition, Sulfox, and Charles

Sulfox the fennec fox celebrated his 20th birthday with great joy as he received a special gift — a pixel game named World Rebuilder. In this game, he plays the role of an interstellar explorer who journeys toward Polaris, the brightest star in the constellation Ursa Minor, and terraforms planets along the way.

A game level can be determined by a target sequence $a_1, a_2, \ldots, a_n$ and a tool sequence $b_1, b_2, \ldots, b_m$. At the beginning of this level, there are n (the length of the target sequence) continents with initial altitudes of 0, arranged circularly around the equator of the planet where Sulfox landed.

The "Batch Edit" tool allows Sulfox to modify the altitudes of the continents simultaneously. Whenever he uses it, he should firstly select a number k from the tool sequence and choose an arbitrary **real number** x. Then, he can select exactly k **consecutive** continents and add x to each of their altitudes. The level is cleared when the altitudes of the n continents, starting from some continent in some direction, are $a_1, a_2, \ldots, a_n$ in order.

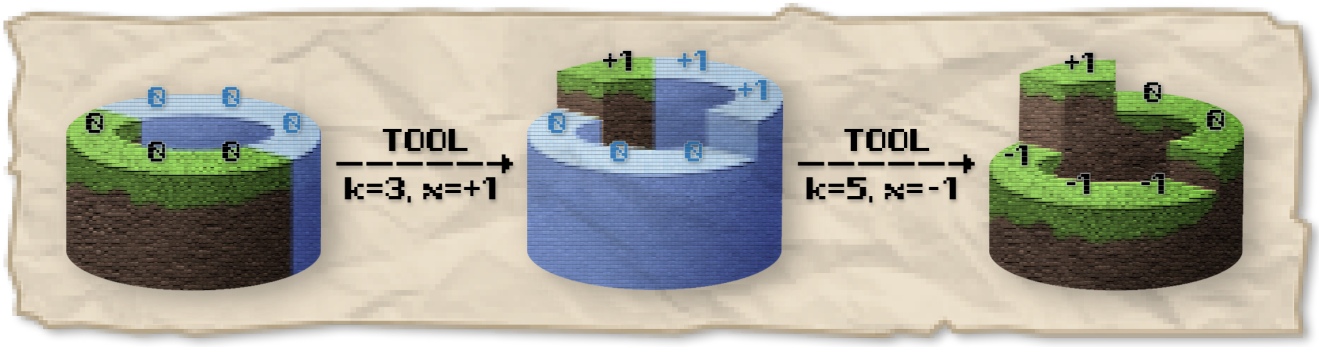


Figure: How to adjust the altitudes with the "Batch Edit" tool
(The selected continents are highlighted)

In fact, the level designer for this game is quite a lazy guy, for he generated each level by selecting a contiguous subsequence from a global target sequence $A_1, A_2, \ldots, A_N$ and a global tool sequence $B_1, B_2, \ldots, B_M$ respectively. What's worse, he didn't even bother to check whether each level is possible to clear, leaving Sulfox stuck at some levels marked as "easy"!

Luckily, as a video game, it certainly has multiple version updates to fix bugs (or maybe introduce new bugs, who knows?). Now given the global target sequence $A_1, A_2, \ldots, A_N$ and the global tool sequence $B_1, B_2, \ldots, B_M$ in the initial version, you need to handle Q events of two kinds:

- Game Update: Change the value of A_p to v in a new version.

- Level Query: Query if Sulfox can clear the level determined by target sequence $A_l, A_{l+1}, \ldots, A_r$ and tool sequence $B_s, B_{s+1}, \ldots, B_t$ in the latest version if he can use the tool any number of times.

Input

The first line contains three integers N, M, and Q $(1 \le N, M, Q \le 2 \times 10^5)$, denoting the length of the global target sequence, the length of the global tool sequence, and the number of events, respectively.

The second line contains N integers $A_1, A_2, \ldots, A_N$ $(0 \le A_i \le 10^9)$, denoting the global target sequence in the initial version.

The third line contains M integers $B_1, B_2, \ldots, B_M$ $(1 \le B_i \le N)$, denoting the global tool sequence.

Then, each of the next Q lines contains an event in one of the following two formats:

- `U p v` $(1 \le p \le N,\ 0 \le v \le 10^9)$, denoting an update that changes the value of A_p to v in a new version.
- `Q l r s t` $(1 \le l \le r \le N,\ 1 \le s \le t \le M)$, denoting a query of if Sulfox can clear the level determined by target sequence $A_l, A_{l+1}, \ldots, A_r$ and tool sequence $B_s, B_{s+1}, \ldots, B_t$ in the latest version. It is guaranteed that each value of $B_s, B_{s+1}, \ldots, B_t$ does not exceed $r - l + 1$.

Output

For each query, output "`Yes`" in one line if Sulfox can clear the level determined by the given target sequence and the given tool sequence in the latest version, or otherwise, output "`No`" in one line.

Example

standard input	standard output
6 4 5 1 1 4 5 1 4 3 3 2 4 Q 1 5 1 2 Q 2 5 3 4 U 5 2 Q 1 6 1 2 Q 2 5 3 4	Yes No No Yes

Problem G. Military Maneuver

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	512 megabytes

A military maneuver is going on a two-dimensional Cartesian plane, and n enemy targets are hiding somewhere on the battlefield, whose locations are known to our headquarters.

Our headquarters will airdrop a beacon in a rectangular region with sides parallel to the coordinate axes uniformly at random to expose all the enemy targets to our troops on the battlefield so that our troops can surround all the enemy targets. The bottom-left corner of the region is at coordinate (x_l, y_l) while the top-right corner is at coordinate (x_r, y_r).

After being dropped, the beacon will firstly receive two parameters r and R that satisfy $0 \le r \le R$ from our headquarters, then scan an annulus region, that is, the region lying between two concentric circles, where the radius of the inner circle is r and that of the outer circle is R, and finally mark those enemy targets hiding in the scanned region (including the boundary).

However, the beacon can only scan a unit area in a unit of time, and the commander would like to know the expected minimum time for the beacon to scan the designated annulus region so that it can mark all the enemy targets.

Input

The first line contains four integers x_l, y_l, x_r, and y_r ($-10\,000 \le x_l, y_l, x_r, y_r \le 10\,000$, $x_l < x_r$, $y_l < y_r$), denoting the coordinates of the bottom-left and the top-right corners of the rectangular region where the beacon will be dropped.

The second line contains a single integer n ($2 \le n \le 2\,000$), denoting the number of enemy targets on the battlefield.

Each of the following n lines contains two integers x and y ($-10\,000 \le x, y \le 10\,000$), denoting an enemy target located at coordinate (x, y).

It is guaranteed that no two enemy targets share the same locations.

Output

Output a single real number, indicating the expected minimum time for the beacon to scan the designated annulus region.

Your answer is acceptable if its absolute or relative error does not exceed 10^{-6}. Formally speaking, suppose that your output is a and the jury's answer is b, your output is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Examples

standard input	standard output

`0 0 2 2`
`2`
`3 1`
`1 3`
	`8.377580409572781970`

`0 0 2 2`
`2`
`5 1`
`1 3`
	`37.699111843077518863`

Note

In the first sample case, if the beacon is dropped to $(0.5, 1.5)$, the minimum time as well as the minimum area of the feasible annulus region is 4π. The expected minimum time when the beacon dropped in the rectangular region uniformly at random is $\frac{3}{8}\pi$.

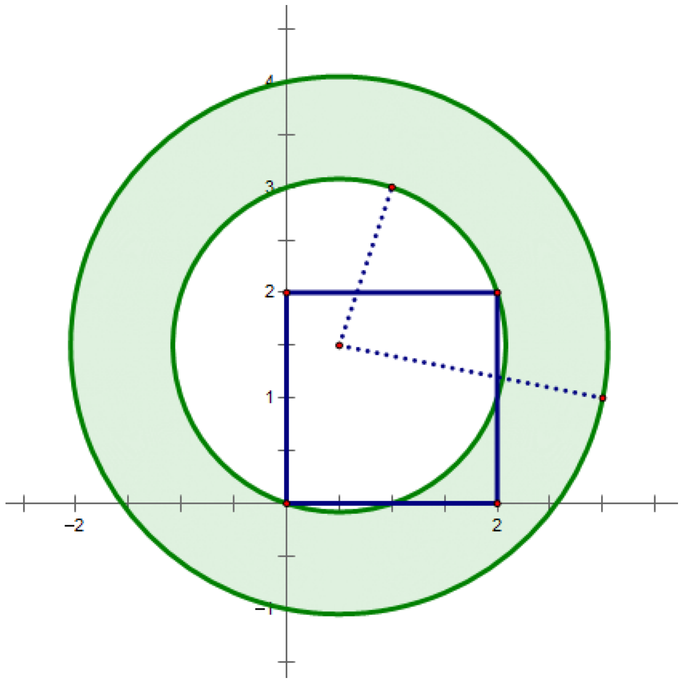


Figure: The feasible annulus region for the beacon at $(0.5, 1.5)$

Problem H. Line Graph Sequence

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

In the mathematical discipline of graph theory, the line graph of a simple undirected graph G is another simple undirected graph $L(G)$ that represents the adjacency between every two edges in G.

Precisely speaking, for an undirected graph G without self-loops or multiple edges, its line graph $L(G)$ is a graph such that

- each vertex of $L(G)$ represents an edge of G; and
- two vertices of $L(G)$ are adjacent if and only if their corresponding edges share a common endpoint in G.

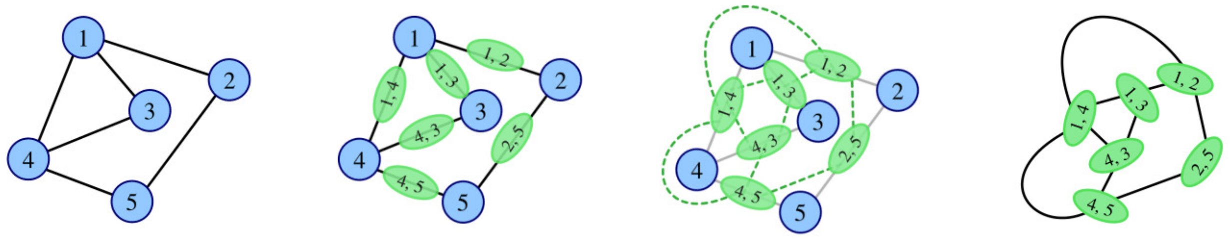


Figure: Generation of the Line Graph

Given a simple undirected graph G, you need to find the minimum number of vertices among all the graphs in sequence $L^0(G), L^1(G), \ldots, L^{k-1}(G)$, where $L^0(G) = G$ and $L^t(G) = L(L^{t-1}(G))$ for each positive integer t.

Input

The input contains several test cases, and the first line contains a single integer T ($1 \le T \le 10^5$), denoting the number of test cases.

For each test case:

The first line contains three integers n ($1 \le n \le 10^5$), m ($0 \le m \le \min\left(\frac{n(n-1)}{2}, 10^5\right)$), and k ($1 \le k \le 10^5$), denoting the number of vertices and edges in graph G and the length of the line graph sequence.

Then m lines follow, each of which contains two integers u and v ($1 \le u, v \le n$), denoting an undirected edge connecting the u-th and the v-th vertices in graph G. It is guaranteed that graph G contains no self-loops or multiple edges.

It is guaranteed that the total number of vertices and edges in all test cases do not exceed 10^5 respectively.

Output

For each test case, output a line containing a single integer, indicating the minimum number of vertices among all the graphs in the sequence $L^0(G), L^1(G), \ldots, L^{k-1}(G)$.

Example

standard input	standard output

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

Problem I. Three Rectangles

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

A rectangular board with sides parallel to the coordinate axes lies on a two-dimensional plane. The bottom-left corner of the region is at coordinate $(0, 0)$ while the top-right corner is at coordinate (H, W).

Given three rectangular cards, you need to put these three cards fully inside the board to cover the whole board and find the number of different placements. Here the cards should not be rotated or flipped, the sides of each card should be parallel to the coordinate axes, and the corners of each card should have integral coordinates.

Two placements are considered different if and only if there exists a card with different locations in the two placements. Since the answer may be large, output it modulo $10^9 + 7$.

Input

The input contains several test cases, and the first line contains a single integer T ($1 \le T \le 10^5$), denoting the number of test cases.

For each test case:

The first line contains two integers H and W ($1 \le H, W \le 10^9$), denoting the height and the width of the rectangular board.

Then three lines follow, the i-th of which contains two integers h_i ($1 \le h_i \le H$) and w_i ($1 \le w_i \le W$), denoting the height and the width of the i-th rectangular card.

Output

For each test case, output a line containing a single integer, indicating the number of different placements that satisfy the conditions, modulo $10^9 + 7$.

Examples

standard input	standard output

5
2 2
1 1
1 1
1 1
2 2
1 1
1 2
1 2
2 2
1 1
1 2
2 1
2 2
1 2
1 2
1 2
2 2
1 2
1 2
2 1
	0
	8
	4
	6
	4

4
1 3
1 1
1 2
1 3
1 4
1 1
1 2
1 3
1 5
1 1
1 2
1 3
1 6
1 1
1 2
1 3
	6
	12
	14
	6

1
1000000000 1000000000
1 1
1 1
1000000000 1000000000
	2401

Problem J. Graft and Transplant

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Alice and Bob have a tree with n vertices numbered from 1 to n, and they want to play a game on this tree to determine the only winner. They decide to perform the following operation alternately, with Alice going first. In each operation, they can choose two adjacent vertices u and v, and make all the undirected edges connecting u, except the one connecting u and v, reconnect to v instead of u. In short, each edge (u, w) where $w \neq v$ changes to (v, w) after the operation.

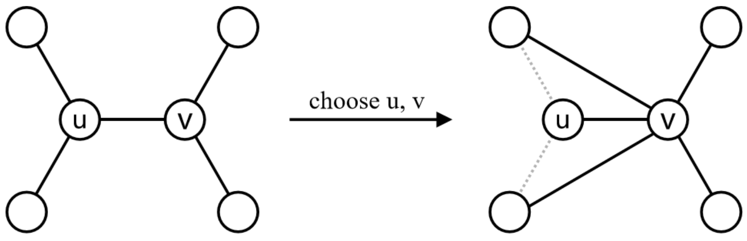


Figure: The “Graft and Transplant” Operation

However, they soon realize that such an operation will probably result in a game that never ends, so they add an extra rule that the two trees before and after an operation must not be isomorphic. More formally, let $V(S)$ be the set of vertices in a tree S and $V(T)$ be the set of vertices in a tree T. Tree S and tree T are *isomorphic* if there exists a bijection $f : V(S) \to V(T)$ such that for all pairs of vertices (u, v) in $V(S)$, u and v are connected by an edge in S if and only if vertices $f(u)$ and $f(v)$ are connected by an edge in T. Namely, $f(s) = t$ implies vertex s in S corresponds to vertex t in T.

In this scenario, when one player cannot perform any valid operations, the other player will win the game. Assuming both Alice and Bob are clever enough and will adopt the best strategy to win or prevent the opponent from winning if one is unable to win, you need to predict the winner.

Input

The first line contains an integer n ($2 \le n \le 50$), denoting the number of vertices in the tree.

Then $n-1$ lines follow, each of which contains two integers u and v ($1 \le u, v \le n$), denoting an undirected edge connecting vertices u and v. It is guaranteed that the given edges form a tree.

Output

Output “`Alice`” if Alice will win the game, “`Bob`” if Bob will win the game, or “`Draw`” if the game under the extra rule will still never end if both of them perform their operations optimally.

Examples

standard input	standard output

4
1 2
2 3
3 4
	Alice

4
1 2
1 3
1 4
	Bob

Problem K. Maximum Rating

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

A rating system, usually used in sports, games, and competitive programming platforms, is a method to rank the skill level of their players or users in relatively impartial ways. The *rating* of an individual is the numerical evaluation of competitive performance, which is directly comparable even at different times.

Sulfox is an ICPC contestant who has participated in n AtForces rounds, where the rating change for the i-th round is a_i. The initial rating and maximum rating are both 0. After each round, the rating is increased by the rating change for that round. If the rating at that point is **strictly greater** than the current maximum rating, the maximum rating will be updated to the current rating.

Now Sulfox has hacked into AtForces' back-end database, which enables him to arrange these n rounds in any order. He wonders how many values of k exist satisfying that there is at least an arrangement of the n rounds that updates the maximum rating exactly k times. Additionally, he wants to know the result each time after some updates that modify the rating change for one of the n rounds.

Input

The first line contains two integers n and q ($1 \le n, q \le 2 \times 10^5$), denoting the number of AtForces rounds and the number of updates respectively.
The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$), denoting the rating changes for each round.
Then q lines follow, each containing two integers x ($1 \le x \le n$) and v ($-10^9 \le v \le 10^9$), denoting an update that modifies the rating change for the x-th round to v.

Output

After each update, output a line containing an integer, representing the number of k satisfying that there exists at least an arrangement of the n rounds where the maximum rating is updated exactly k times.

Example

standard input	standard output

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

Note

In the sample case:

- After the first update, the rating changes for each round are $[1, 2, 4]$, and the maximum rating can only be updated 3 times.
- After the second update, the rating changes for each round are $[1, -2, 4]$, and the maximum rating can be updated 1 or 2 times.
- After the third update, the rating changes for each round are $[-3, -2, 4]$, and the maximum rating can be updated 0 or 1 times.
- After the fourth update, the rating changes for each round are $[-3, -2, 1]$, and the maximum rating can be updated 0 or 1 times.

- After the fifth update, the rating changes for each round are $[-3, 1, 1]$, and the maximum rating can be updated 0, 1, or 2 times.

Problem L. Rook Detection

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

This is an interactive problem.

Consider an $n \times n$ chessboard, where its rows are numbered 1 to n from top to bottom, and its columns are numbered 1 to n from left to right. The square on the x-th row and the y-th column is denoted as (x, y) and is either empty or occupied by a neutral rook. According to the rules of chess, a rook can move any number of squares along a row or column in one step but cannot leap over other pieces. In this scenario, the *control area* of a rook consists of the square it occupies and the squares it can reach in one step.

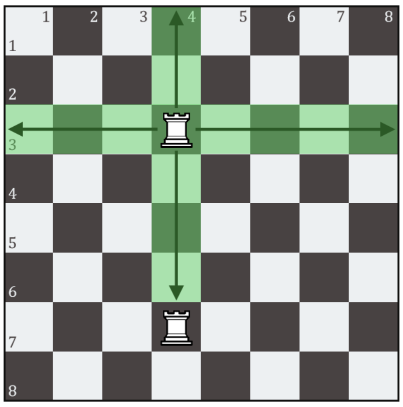


Figure 1: The control area of the rook at (3,4)

A square is considered "*controlled*" if and only if it belongs to one or more control areas. The known information is that all squares are controlled initially, so clearly, there are at least n rooks on the chessboard. Whether each square is empty or occupied by a rook, however, is unknown to you. Your task is to locate at least n of the hidden rooks by making no more than $\lceil \log_2 n \rceil + 2$ queries.

In each query, you need to split the chessboard into two tiers by determining which squares should comprise the upper tier and which the lower tier. The rooks are only allowed to move within the tier to which they belong, and they can leap over gaps but cannot leap over other pieces in the same tier. The definitions of control areas and controlled squares do not change. As a response, the interactor will inform you whether each square is controlled after the splitting. Then, the chessboard will return to the initial state.

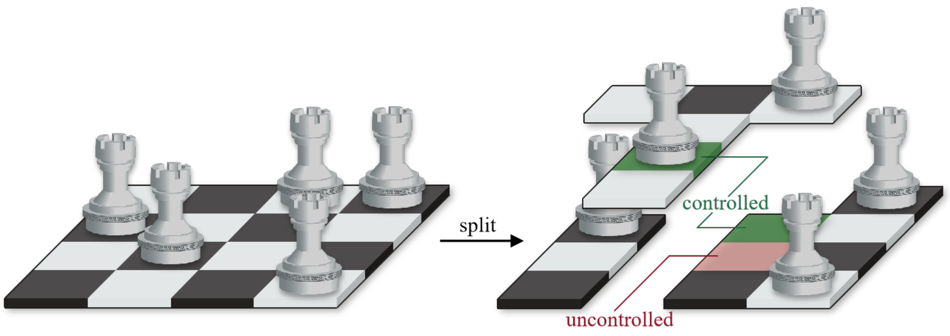


Figure 2: Split the T-shaped upper tier out of the chessboard

You may assume that the locations of all the rooks are fixed in advance. Note again that it is not necessary to find all the rooks when the actual number of rooks is greater than n.

Interaction Protocol

The first line contains an integer T $(1 \le T \le 10^4)$, denoting the number of test cases.

The first line of each test case contains an integer n ($3 \le n \le 500$), denoting the size of the chessboard. It is guaranteed that the number of test cases that $n \ge 10$ does not exceed 5.

To initiate an interaction behavior, you should firstly print a character *opt* ($opt \in \{$'?' , '!'$\}$) in one line and then print an $n \times n$ binary matrix (in n lines, each containing a binary string). After printing the binary matrix, do not forget to flush the output. To do this, use `fflush(stdout)` or `cout.flush()` in C++, `System.out.flush()` in Java, or `stdout.flush()` in Python.

opt = '?' represents making a query. If the character in the x-th row and y-th column of your binary matrix is '1', the square (x, y) will comprise the upper tier, or otherwise, it will comprise the lower tier. The interactor will return an integer *code* ($code \in \{0, 1\}$) in one line. If $code = 1$, you will get `Wrong Answer` because you made more than $\lceil \log_2 n \rceil + 2$ queries. If $code = 0$, you will then receive another $n \times n$ binary matrix from the interactor. If the character in the x-th row and y-th column of the received binary matrix is '1', the square (x, y) is controlled after the splitting, or otherwise, it is uncontrolled.

opt = '!' represents answering the locations. If the character in the x-th row and y-th column of your binary matrix is '1', the square (x, y) is occupied by a rook in your judgment, or otherwise, whether it is empty or occupied by a rook will not affect the verification of your answer. The interactor will return an integer *code* ($code \in \{0, 1\}$) in one line. If $code = 1$, you will get `Wrong Answer` because your answer failed to match the actual locations of the rooks or you did not find enough rooks. If $code = 0$, you will enter the next test case if the current test case is not the last one.

If you attempt to print *opt* that is neither '?' nor '!', or if your binary matrix is badly formatted, such as some of the n rows has a length other than n or contains a character that is neither '0' nor '1', the interactor will return $code = -1$ and you will get `Wrong Answer` as well.

Please terminate your program immediately when you receive some *code* other than 0 to avoid any unexpected verdicts.

Example

standard input	standard output
1	
3	
	?
	101
	000
	101
0	
110	
111	
111	
	?
	111
	100
	100
0	
111	
111	
101	
	!
	010
	001
	100
0	

Note

The blank lines in the sample case are added for readability. In your output, extra spaces or blank lines will be ignored.

Problem M. Outro: True Love Waits

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

People often harbor various interpretations of life. Some navigate it smoothly, discovering its meaning at the loftiest peaks, while some encounter misfortune time and time again, lamenting the fleeting nature of life akin to the falling of snow, yet burning it to fight for better situations.

Regardless, life is intricate and filled with worthwhile things to explore. For programmers like you, it is reasonable to conceptualize life as touring an infinite graph — a graph consisting of an infinite number of vertices and edges, to be precise. Its vertices, indicating all possibilities of life, are numbered starting from 0, and for every unordered pair of vertex u and vertex v, if the binary representations of u, v differ by exactly one bit which is the w-th bit from low to high, then there exists an undirected edge between them with weight w. For example, the undirected edge $(2, 6)$ with weight 3 exists in the graph because $2 = (10)_2$, $6 = (110)_2$, where the third bit is the only bit different, but the edge $(5, 6)$ does not exist because $5 = (101)_2$, $6 = (110)_2$, where there are two bits different between the binary representations.

Initially, you stand on vertex s, and you believe that true love, not necessarily being a certain person, should inhabit on vertex t. Then you begin the tour to seek true love in life from dawn to dusk. Every moment, you feel unsatisfied with the vertex you are currently on and decide to let go of it. Therefore, among all the edges connecting to this vertex, you choose the one with minimum weight, move through it to the other vertex, and permanently cut this edge from the graph (Uh, it seems like swearing never to come back). Since the tour is endless and the faith in true love is boundless, you become interested in when you will reach vertex t for the k-th time (Particularly, the initial state is regarded as the first time you reach vertex s). Find out the number of edges you will cut along the tour, modulo $10^9 + 7$, or report it's impossible to reach vertex t for the k-th time.

Input

The input contains several test cases, and the first line contains a single integer T $(1 \le T \le 10^5)$, denoting the number of test cases.

For each test case, the only line contains three integers s, t, k $(0 \le s, t < 2^{10^6}, 1 \le k \le 10^9)$, where s, t are given in binary representation without any extra leading zeros, and k is given in decimal representation.

It is guaranteed that the sum of the lengths of binary representations over all test cases does not exceed 10^7.

Output

For each test case, if it's impossible to reach vertex t for the k-th time, output -1 in one line. Otherwise, output the number of edges you will cut in one line, modulo $10^9 + 7$.

Example

standard input	standard output

4
1 10 1
1 10 2
100 0 2
11 11 3
	2
	-1
	9
	20

Note

In the first test case, you will reach vertex 2 for the first time by trail $1 \xrightarrow{w=1} 0 \xrightarrow{w=2} 2$, so there will be 2 edges cut. Also starting from vertex 1, it can be proved that once you let go of vertex 2 when you reach it for the first time, you will really never come back forever. Thus, the answer for the second test case is -1.