Problem A. Arboreal Colors

Input file:	`standard input`
Output file:	`standard output`
Balloon Color:	`Orange`

Given a binary tree with N nodes, your task is to manage the coloring of its nodes based on a series of queries. The tree nodes are numbered from 1 to N. Initially, all nodes are colored green.

You are then given Q queries, each query can be one of 2 formats:

- Color X: means you should color node X, its parent, and its direct children, red.
- Count: means you should print the number of green nodes.

Input

The first line contains a single integer N ($2 \leq N \leq 10^6$) — the number of nodes in the binary tree.

Each of the following $N - 1$ lines contain the edges. Each edge is described by two integers a and b ($1 \leq a, b \leq N$), which means that there's an edge between nodes a and b. It's guaranteed that the given edges form a binary tree.

The following line contains a single integer Q ($1 \leq Q \leq 2 \times N$) — the number of queries.

The next Q lines describe the queries and can be of one of two formats:

- `Color X` — where X ($1 \leq X \leq N$) is the node that should be colored red along with its parent and direct children.
- `Count` — a query asking for the number of green nodes.

Output

For each `Count` query, output a single integer, the number of green nodes.

Example

standard input	standard output
13 7 10 11 7 13 12 1 6 2 3 6 5 5 4 4 2 5 8 8 9 7 1 1 13 8 Count Color 4 Count Color 7 Count Color 1 Color 6 Count	13 10 6 4

Note

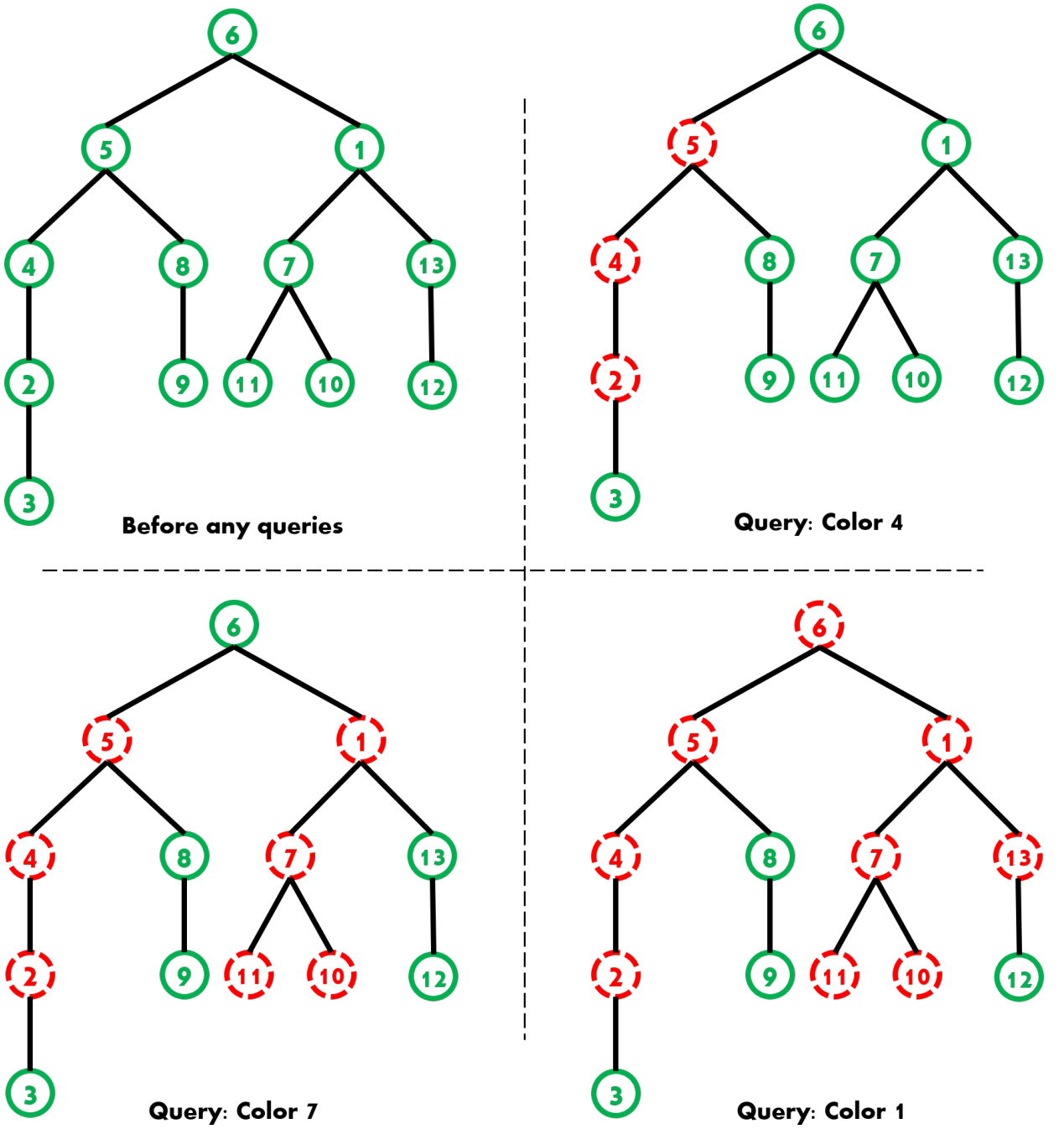


Illustration of the test case. Note that Query "Color 6" has no effect, since 6 and it's direct children have already been colored red.

Problem B. Baffling Dismissals

Input file:	`standard input`
Output file:	`standard output`
Balloon Color:	`Red`

An ACPC judge is a well known engineer who has worked for multiple companies before, and so a lot of ACPC contestants ask her for job referrals, especially now with the recent layoffs.

She has a list of companies with available job openings. Each job opening is associated with a minimum required skill value, and a number of people needed. Similarly, each contestant has a skill value. Our judge decided that whenever she gets a new request for a referral, she would assign engineers in the following way:

1. Invalidate any assignments she made in the past and start assigning all over again.
2. Go through the companies from the highest minimum-required skill value to the lowest.
3. For each company, assign the engineers (who are not assigned yet) with the highest skills, as long as their skill meets the company's requirements.

An engineer can be assigned to exactly one company, or can remain unassigned (because their skill doesn't meet the minimum requirements of any company, or because more skilled people have already been assigned to that company and filled the vacancies).

Note that referrals are dynamic, everyday more and more contestants reach out to our judge, so she our judge has asked for your help! You have to handle these new engineer additions and, on request, tell a particular engineer which company would our judge refer this engineer to, according to her strategy, at this point in time.

Input

The first line contains an integer N ($1 \le N \le 10^5$) — the number of engineers.

Each of the following N lines contains the id ($1 \le id \le 4 \times 10^5$) and $skill$ ($1 \le skill \le 10^5$) of the i^{th} employee. **Skill values are unique among engineers.**

The following line contains an integer M ($1 \le M \le 10^5$) — the number of companies.

Each of the following M lines contains a string $company$ — the name of the company ($1 \le |company| \le 5$), an integer r ($1 \le r \le 10^5$) — the minimum required skill for the job, and e ($1 \le e \le 2 \times 10^5$) — the number of engineers this company needs. **Skill set values are unique across all companies.**

The following line contains an integer Q ($1 \le Q \le 4 \times 10^5$) — the number of queries (dynamic referrals).

Each of the next Q lines contains one of the following formats:

- `C id s`: where id is the new engineer's ID and s ($0 \le s \le 10^5$) is their skill value.
- `A id`: where id is the engineer's ID. For this type of queries, you need to print the $company$ this engineer is referred to, according to our judge's strategy. **Company names are case sensitive.**

Output

For each query of type `A`, print the company this engineer is referred to, according to our judge's strategy. **If the engineer is not referred to a company at this point in time, print "NOMATCH"**.

Example

standard input	standard output
3 1 11 2 22 3 33 3 Micr 10 1 Meta 20 2 Goog 30 3 11 A 1 A 2 A 3 C 4 100 C 5 101 C 6 102 C 7 103 A 7 A 2 A 4 A 1	Micr Meta Goog Goog Micr Meta NOMATCH

Problem C. Cyclic Paths

Input file:	`standard input`
Output file:	`standard output`
Balloon Color:	`Dark Blue`

You are an adventurous traveler preparing to explore a land filled with n cities connected by m one-way roads. Your journey is set to begin in City s and will span k days.

The land you're about to venture into is quite unique. Some cities, known as **Metro Cities**, have the curious property that you can leave the city, travel through a series of one-way roads, and find yourself back where you started. These cities are steeped in lore and said to offer endless adventures.

In contrast, there are cities known as the **Countryside**, where the road network doesn't offer a way to return to the same city once you leave.

Each day, you'll set out to explore and will choose an adjacent city to visit, making your selection entirely at random. For instance, if you find yourself in a city that has d adjacent cities connected by one-way roads, the chance of picking any specific city as your next destination is $\frac{1}{d}$.

Your goal is to determine the probability that, after k days of exploration, you'll end up in one of the Metro Cities.

Input

The first line contains a single integer T ($1 \leq T \leq 100$), specifying the number of adventures you're planning.

For each adventure, the first line contains four integers:

- n ($2 \leq n \leq 100$) — the number of cities.
- m ($1 \leq m \leq n \times (n-1)$) — the number of one-way roads.
- s ($1 \leq s \leq n$) — the city where your adventure begins.
- k ($0 \leq k \leq 10^7$) — the number of days your journey will last.

Each of the next m lines contains two integers u and v ($1 \leq u, v \leq n, u \neq v$), indicating that there is a one-way road from city u to city v. Each pair (u, v) is unique per adventure. All cities are interconnected; the road network forms a single, unified component.

It is guaranteed that the cumulative sum of n across all test cases will not exceed 300.

Output

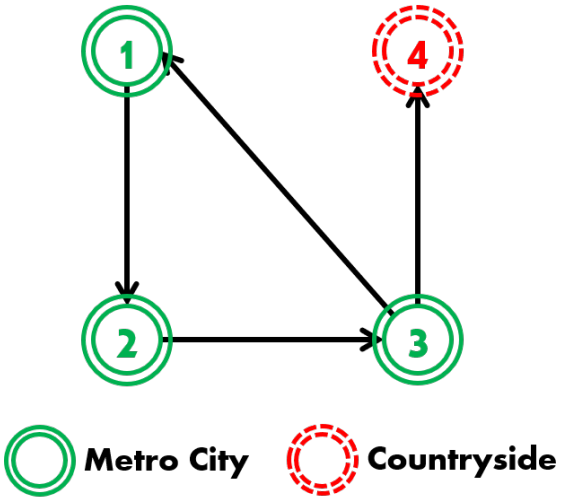
For each adventure, output a single line containing an integer z, which is the probability of ending up in a **Metro City** after k days of exploration, computed modulo $10^9 + 7$.

The answer z is defined precisely as follows: it is the probability that after k days of exploration, you will find yourself in a Metro City, represented as an irreducible fraction p/q. The number z must satisfy the modular equation $z \times q \equiv p \mod 10^9 + 7$, and be between 0 and $10^9 + 6$, inclusive. It can be shown that, under the constraints of this problem, such a number z always exists and is uniquely determined.

Example

standard input	standard output
2 3 3 2 4 1 2 2 3 3 1 4 4 1 3 1 2 2 3 3 1 3 4	1 500000004

Note



An illustration of the cities in the second test case.

Problem D. Draftsman Interface

Input file:	`standard input`
Output file:	`standard output`
Balloon Color:	`Yellow`

You are building a text editor that supports 4 types of operations:

1. Add a given string S, N times to the list of strings that are currently shown in the text editor.
2. Commit the current list of strings (the ID of the commit is the number of commits made before + 1).
3. Check out a specific commit ID.
4. Given a query string Q, find the total number of occurrences of the current strings in the text editor in Q. In other words, for each string in the editor, count the number of times it appears as a substring in Q, and then add this count to the answer of the query.

Input

The first line contains an integer O — the number of operations you will do.

The next O lines will each have an operation to perform.

Operations have the following form:

- `1` S N — Add string S N times ($1 \leq N \leq 10^9$).
- `2` — Create a new commit.
- `3` C — Check out commit with $ID = c$.
- `4` Q — Query string Q.

It's guaranteed that:

- The total sum of lengths of input strings (S) used in operations of the first kind is $\leq 10^6$.
- The total sum of lengths of query strings (Q) $\leq 10^6$
- The total number of commits and checkouts (operations of the 3^{rd} and 4^{th} type) $\leq 10^6$.
- All strings consist of only lowercase English letters.

Output

For each query of type 4, output a new line with the corresponding answer

Example

standard input	standard output
18 2 1 he 5 1 she 3 2 1 her 4 1 hers 1 4 shers 2 3 1 1 sh 1 4 shers 2 3 2 4 shers 3 3 4 shers 3 4 4 shers	13 1 8 13 1

Note

Explanation of committing:

- To commit means to save the current state of the editor and give it a unique ID.
- To checkout a specific commit given an ID means to restore the state of the editor at that commit ID.

Here is an illustration explaining the test case:

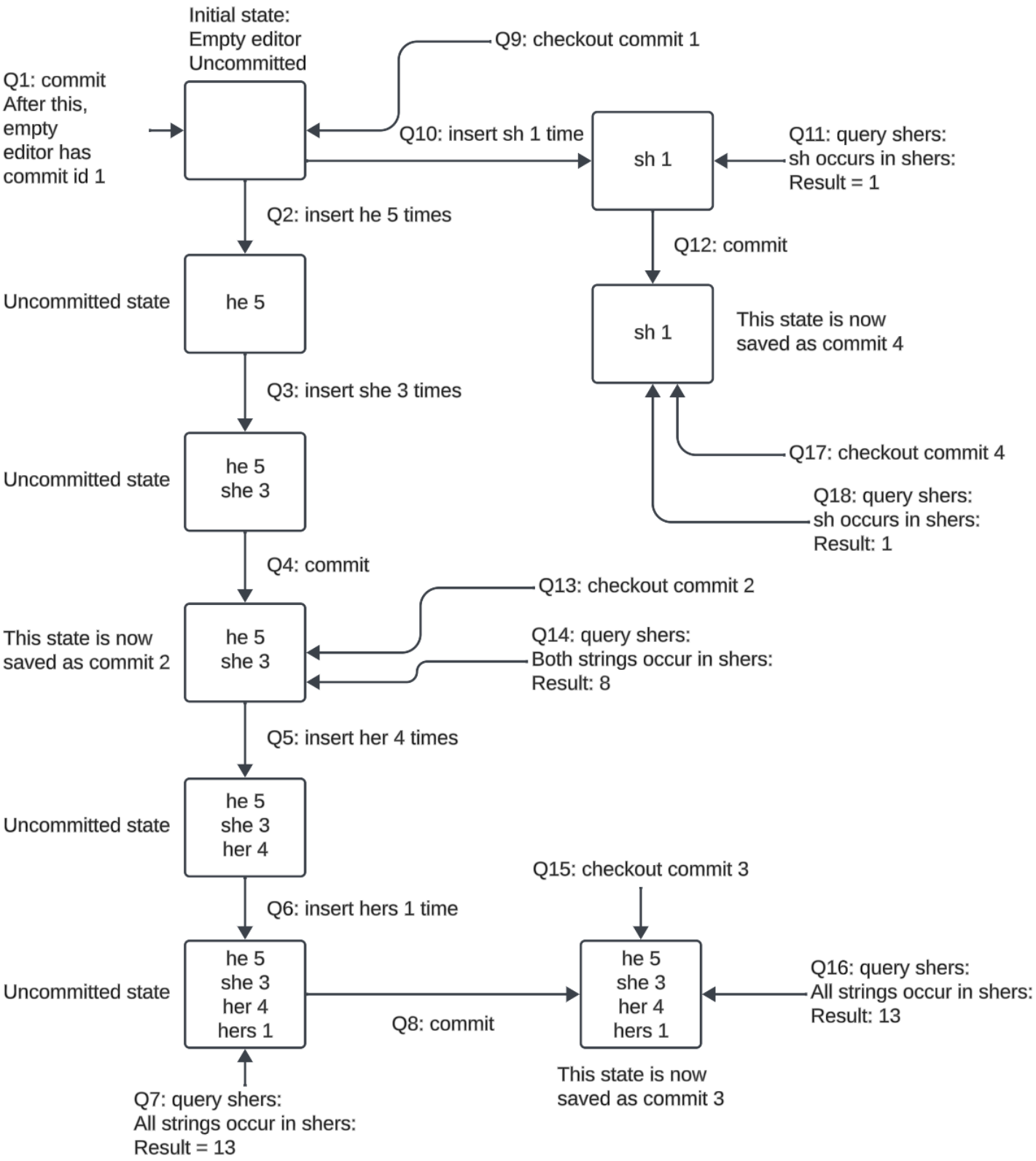


Illustration of the first test case step by step.

Problem E. Equity Fluctuations

Input file:	`standard input`
Output file:	`standard output`
Balloon Color:	`Green`

The stock market is a fascinating world, with prices going up and down based on various factors. In this problem, you are given a history of stock prices in the form of an integer array. Your task is to determine how often a particular trend in stock prices has occurred in the past.

Definition of a Trend:

A trend is a sequence of relative movements (ups and downs) of the stock price. The actual values don't matter, but rather the direction of the change. For instance, if the stock prices for 5 days were [4, 5, 3, 4, 4], the trend would be [up, down, up, constant].

Input

- The first line contains a single integer N ($2 \leq N \leq 10^6$) — the length of the stock price array.
- The second line contains N space-separated integers $p_1, p_2, \ldots, p_N$ ($1 \leq p_i \leq 10^9$) — the stock prices at different points in time.
- The third line contains a single integer Q ($1 \leq Q \leq N$) — the number of queries.
- The next Q lines each contain a single integer X ($2 \leq X \leq N$) — the number of recent stock prices whose trend you need to match against in the past data.

Output

For each query, output a single integer — the number of times the trend of the last X prices occurred in the past.

Examples

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

Problem F. Filling Grids

Input file:	`standard input`
Output file:	`standard output`
Balloon Color:	`Purple`

Imagine we have a grid of $N \times M$ cells, initially white, and we want to color all its cells black. There are rules for coloring it.

You will be given a value for every row and every column, and you can either select a row, or a column.

Let's say you want to color a row, and the value for that row is X, then you can color exactly X cells of that row, starting from the leftmost non-colored cell, and going to the right, skipping already black cells, but not white ones. Once you decide to color a row or column, you have to color the row/column to the end, leaving zero white cells, and you have to color exactly X cells. For columns, you start with the top most non-colored cell, going down, following the same rules. All the column and row values have to be fully used.

Output any order of operations you can use to achieve a fully black grid, while using all column/row values provided.

Input

The first line contains two integers N and M ($1 \leq N, M \leq 1000$) — the dimensions of the grid.

The second line contains N integers R_i ($1 \leq R_i \leq 1000$) — the value of each row.

The third line contains M integers C_i ($1 \leq C_i \leq 1000$) — the value for each column.

Output

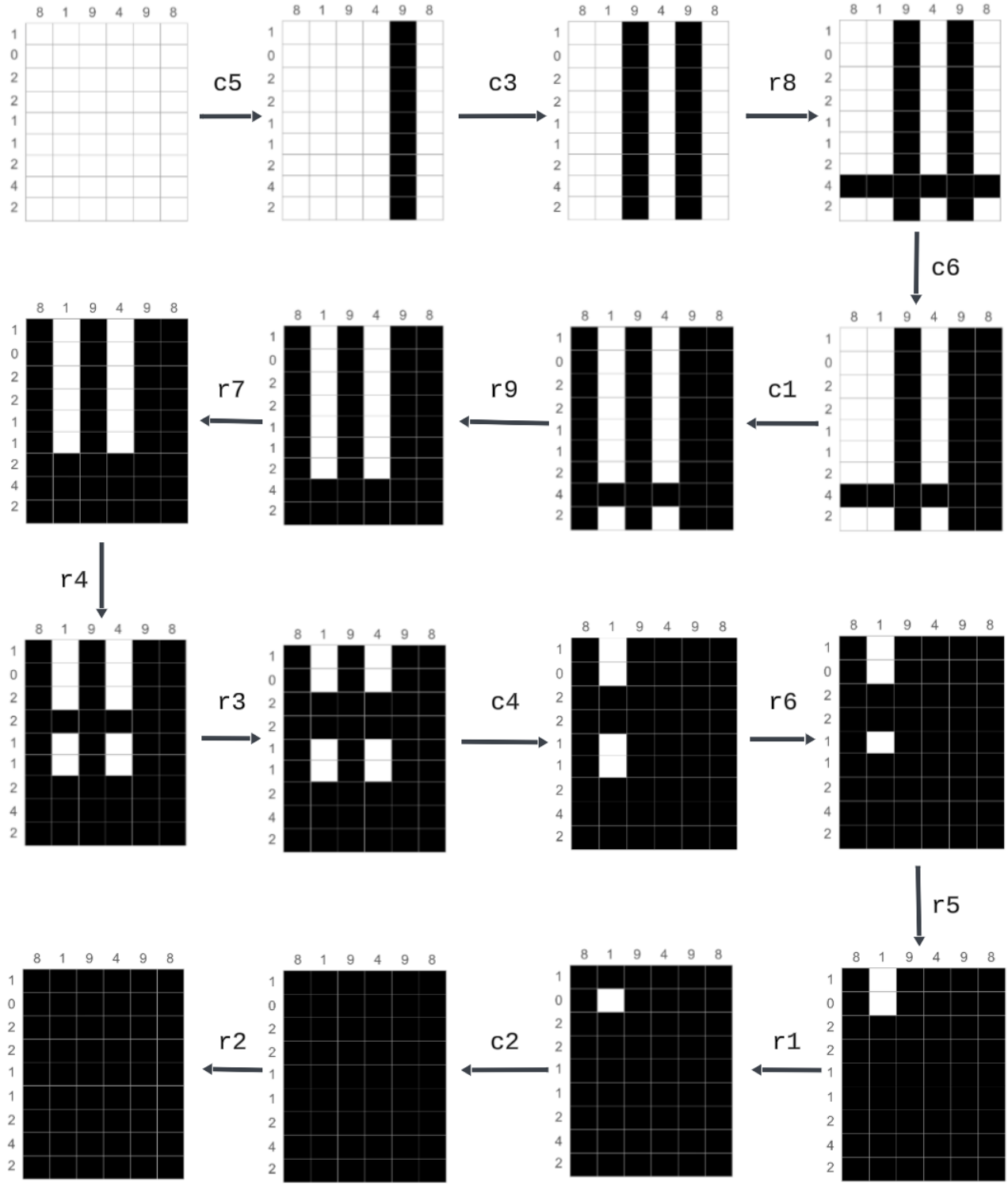
If it is impossible to color all cells of the grid using all the row/column values given, print -1.

If it is possible, print N + M strings separated by a space, "c" or "r" for row or column, followed directly by the index (1-based) of that row or column. This sequence has to be a valid order of operations, that leaves the grid fully black.

Examples

standard input	standard output
9 6 1 0 2 2 1 1 2 4 2 8 1 9 4 9 8	c5 c3 r8 c6 c1 r9 r7 r4 r3 c4 r6 r5 r1 c2 r2
3 1 0 1 0 2	r2 c1 r3 r1
3 3 3 2 1 1 1 1	-1
3 3 3 3 2 1 0 0	r2 r1 c1 r3 c3 c2
3 3 3 3 3 1 0 0	-1
3 3 3 3 0 2 1 0	-1
2 2 2 1 0 1	r1 c2 r2 c1

Note



Step by step coloring of the first test case.

Problem G. Golfing Minimally

Input file:	`standard input`
Output file:	`standard output`
Balloon Color:	`Bronze`

You are given N line segments on a $2D$ plane, and there is a hole located at point (X, Y) on the same plane. Your task is to count how many segments the ball can bounce off to reach the hole, without bouncing on other segments on the way to that segment or on the way to the hole after bouncing. The ball always starts at the origin $(0, 0)$. It's important to ensure that the ball travels in a straight line and only bounces off one segment on its path to the hole.

Input

The first line contains three integers N, X, and Y ($1 \le N \le 10^3, -10^6 \le X, Y \le 10^6$) — the number of line segments and the coordinates of the hole, respectively.

Each of the next N lines contains four integers x_1, y_1, x_2, y_2 ($-10^6 \le x_1, y_1, x_2, y_2 \le 10^6$) — the coordinates of the endpoints of the i-th line segment. It is guaranteed that each line segment is not a point and does not start or end at the origin.

It is guaranteed that:

- All segments are of positive length.
- No segment passes by the origin.
- Segments do not intersect.

Output

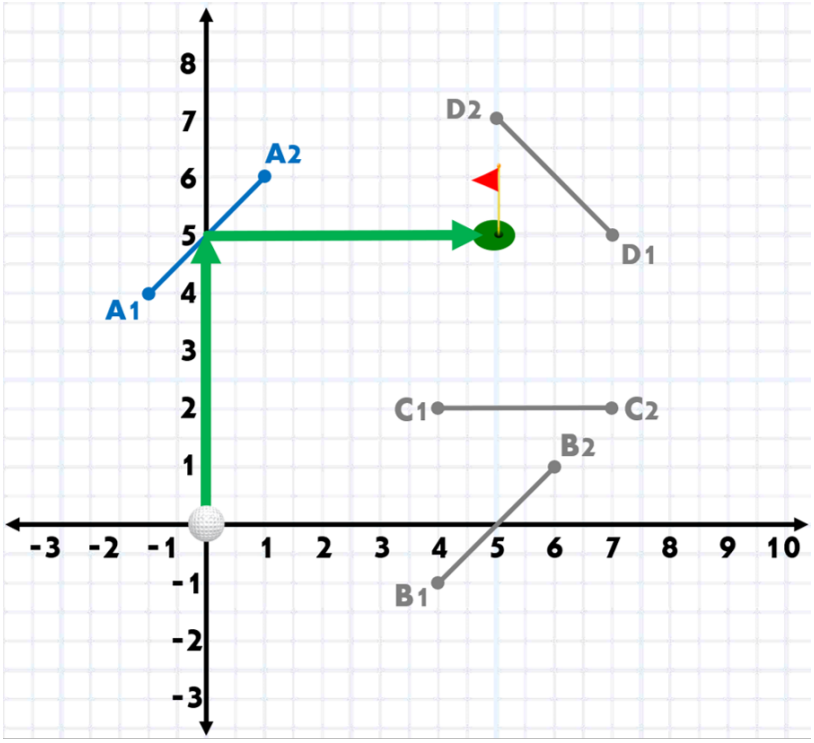
Print a single integer — the number of line segments that a ball starting from the origin can bounce off to end up in the hole.

Example

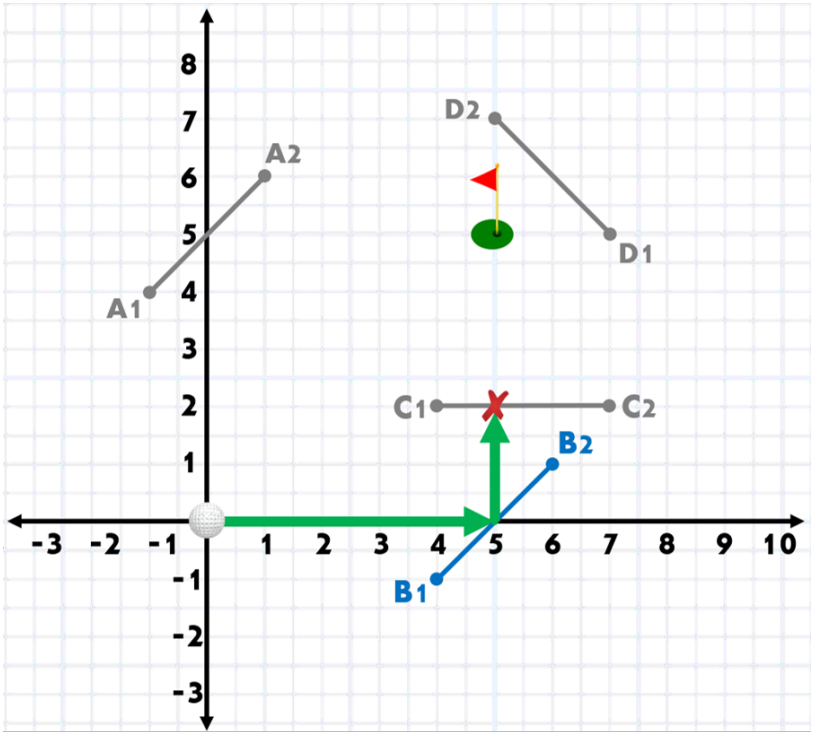
standard input	standard output
4 5 5 -1 4 1 6 4 -1 6 1 4 2 7 2 7 5 5 7	2

Note

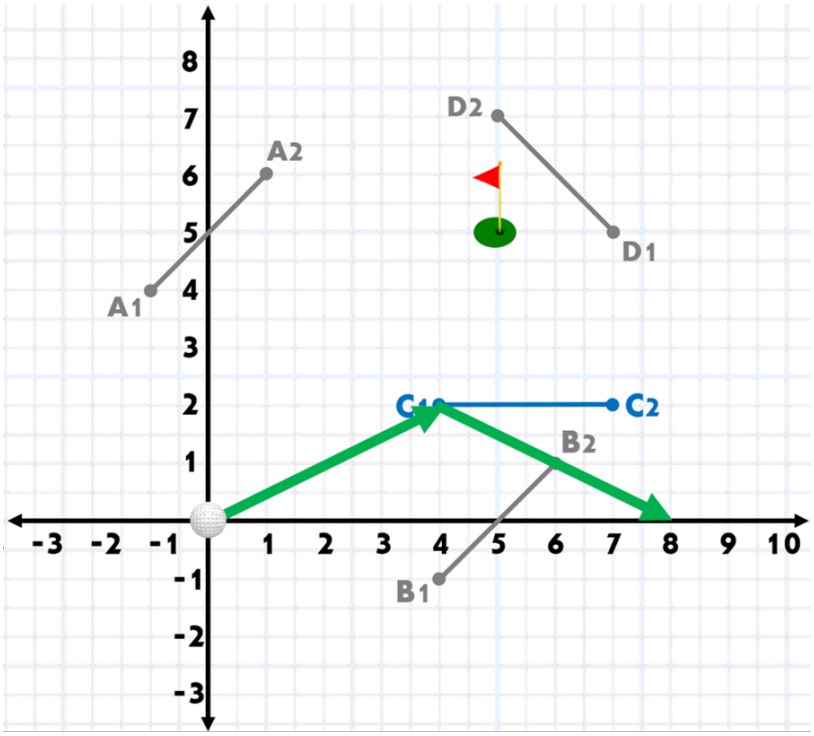
Illustration of the test case:



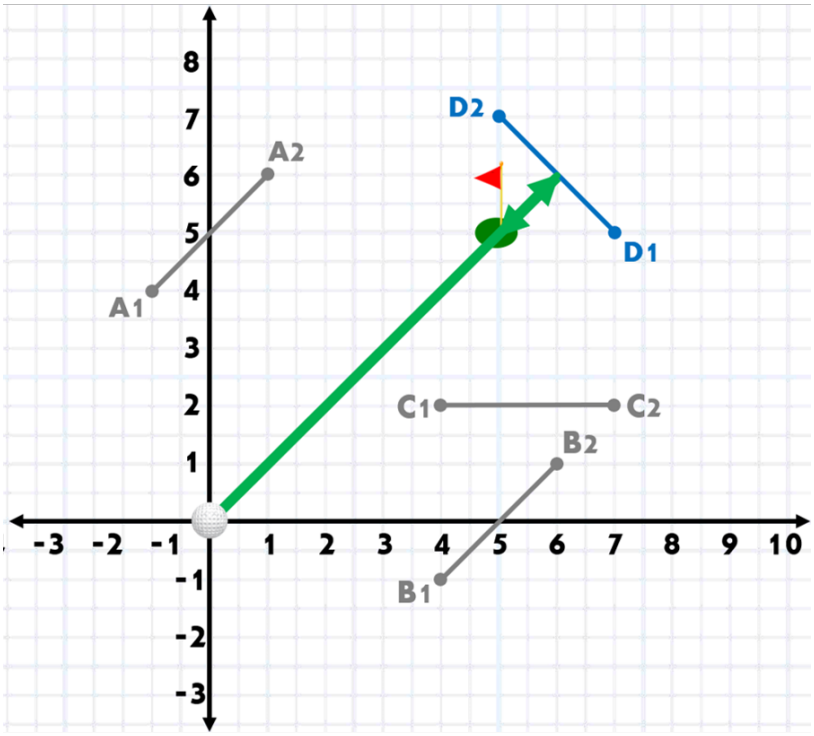
The ball bounces off (A_1, A_2)



The ball cannot bounce off (B_1, B_2)



The ball cannot bounce off (C_1, C_2)



The ball bounces off (D_1, D_2)

Problem H. Hidden Absence

Input file:	`standard input`
Output file:	`standard output`
Balloon Color:	`Black`

You have written this piece of code, and run dfs(1, 1) on a tree that has N nodes.

Algorithm 1 Depth-First Search

```
1: procedure DFS(node, parent)
2:     print node
3:     for each next in graph[node] do
4:         if next ≠ parent then
5:             DFS(next, node)
6:         end if
7:     end for
8:     print node
9: end procedure
```

Some values of the printed sequence are lost (represented as 0s), your task is to restore any valid sequence, or state that it's impossible.

Input

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

For each test case:

- The first line contains an integer N ($1 \leq N \leq 10^5$) — the number of nodes in the tree.
- The second line contains $2 * N$ numbers, the sequence from the code, and 0 for every lost number.

Output

For every test case, print a line saying "YES" if you could restore any valid sequence, or "NO" if you could not. If the answer is "YES", print a second line containing $2 * N$ numbers that represent any valid sequence.

Example

standard input	standard output
4 4 0 0 0 0 0 0 0 0 3 0 0 0 3 0 0 2 0 2 2 1 4 1 2 4 3 3 0 4 1	YES 1 2 2 3 3 4 4 1 YES 1 2 3 3 2 1 YES 1 2 2 1 NO

Problem I. Identical Reflections

Input file:	`standard input`
Output file:	`standard output`
Balloon Color:	`Pink`

Given N positive integers, find the minimum number of digits that should be changed so that each pair of integers can form a palindrome when the union of their digits is arranged in any order.

Input

The first line contains an integer N ($2 \leq N \leq 10^4$) — the number of provided integers.

Each of the following N lines contains an integer n_i ($1 < n_i < 10^{100}$).

Output

Print a single integer — the minimum number of digits needed to be changed.

Example

standard input	standard output
4 13 14 23 113	2

Note

- For the pair $(13, 14)$, it can't be arranged to form a palindrome.
- For the pair $(13, 23)$, it can't be arranged to form a palindrome.
- For the pair $(13, 113)$, it can be arranged to form a palindrome (e.g. 13131 or 31113).
- Hence, the result is 2 (we can change $1\underline{4}$ to $1\underline{3}$ or $1\underline{1}$, and $\underline{2}3$ to $\underline{3}3$).

Problem J. Juxtaposing Parity

Input file:	standard input
Output file:	standard output
Balloon Color:	White

We have a tree of N nodes and weighted edges. The longest path in this tree (or any tree) is a simple path of maximum length. There can be more than 1 longest path in a tree. In our problem, there is a person standing at every leaf node that is on one of the longest paths.

We want to group these lonely people, such that:

- Each group consists of exactly K people.
- The distance (in terms of the number of edges, not the sum of edge weights) between any pair in the group is the same and maximized.
- The sum of edge weights for every pair of members in a group is even (for even groups) or odd (for odd groups).

Your task is to determine the maximum number of even and odd groups (independently) that can be created while satisfying the above conditions.

Input

The first line contains two integers N and K ($3 \leq N \leq 1 \times 10^5$) ($2 \leq K \leq N$) — the number of nodes and the number of people in a group, respectively.

The following $N-1$ lines describe the tree. Each line contains three integers u, v, w representing an edge between nodes u and v with weight w ($1 \leq u, v \leq N$), ($1 \leq w \leq 1 \times 10^9$).

Output

Output two integers X and Y, the maximum possible number of even groups and the maximum possible number of odd groups that can be created, respectively.

Examples

standard input	standard output
5 3 1 2 3 3 1 10 1 5 8 1 4 4	1 0
9 2 4 2 979 3 1 669 4 8 972 4 5 67 4 6 985 4 9 991 3 7 355 3 4 214	2 1

Note

Below is an illustration of a valid grouping of the second test case. Note that the maximum number of even groups is 2, but these groups could be (6, 1) and (7, 5) as the illustration below, or (6, 7) and (1,

5), or (1, 2) and (7, 9) ...etc. Similarly, the maximum number of odd groups is 1, but this group could be (8, 1) as the illustration below, or (8, 7).

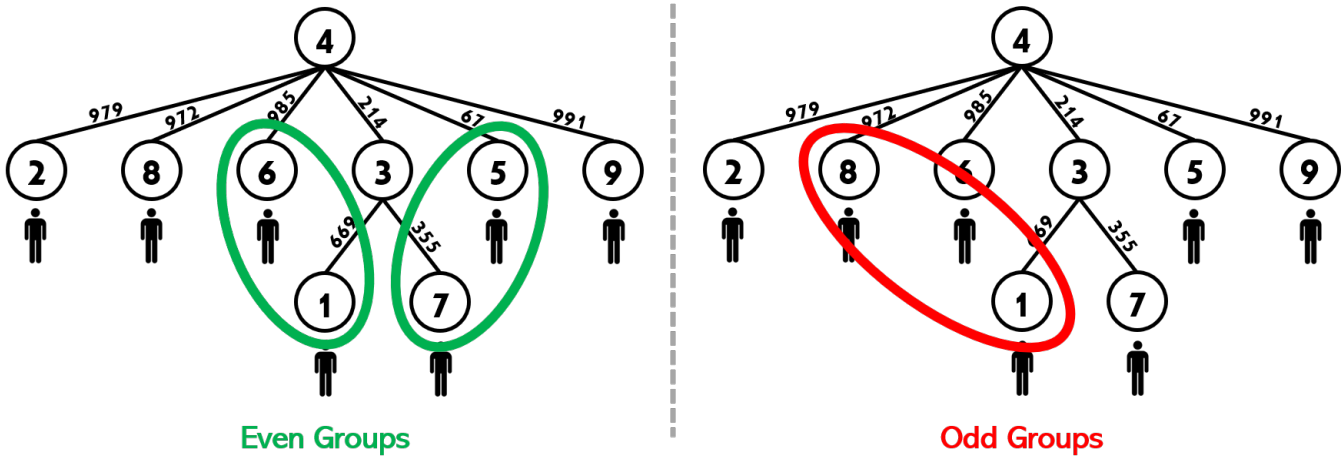


Illustration of a valid even/odd grouping for the second test case.

Problem K. Keyline Subdivisions

Input file:	standard input
Output file:	standard output
Balloon Color:	Silver

You have a tree with N nodes, rooted at node 1. There are K special nodes.

Let's define d to be an arbitrary subtree size, such that if we take all nodes with subtree size equal to d and $d+1$ and look at their combined set of subtrees, it will contain all the special nodes.

Your task is to print all possible values of d.

Input

The first line contains 2 integers: N and K ($1 \leq N, K \leq 10^5$).

$N-1$ lines follow, each line has 2 integers U, and V, which means there is an edge between node U and node V.

The last line contains K integers, the special nodes.

All nodes are 1 based.

Output

Print one line, containing a list of integers separated by a space, the values of d that solve the problem.

Examples

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

Problem L. Languid Guardian

Input file:	`standard input`
Output file:	`standard output`
Balloon Color:	`Rose`

You are a guard at an art gallery, but you prefer to sit rather than stand during your shift. The gallery is a long hallway with chairs placed at specific positions. Unfortunately, your boss insists that you only sit if you can still see a specific valuable painting from your chair.

The hallway is represented as a number line, with the painting positioned at point 0. There are three chairs, each at a distinct position in the hallway. Given the positions of these chairs and a range within which the painting is visible from a chair, determine whether you can sit down during your shift and still keep an eye on the painting.

Input

The first line contains two integers L and R ($-10^3 \leq L < 0 < R \leq 10^3$), representing the range within which the painting is visible from a chair. If you're sitting at a position p, you can see the painting if ($L \leq p \leq R$).

The second line contains three distinct integers C_1, C_2, and C_3 ($-10^3 \leq C_i \leq 10^3$), the positions of the three chairs.

Output

Print "Yes" if you can sit on any chair while still being able to see the painting. Otherwise, print "No".

Examples

standard input	standard output
-2 3 -3 -1 4	Yes
-2 3 -3 5 6	No

Note

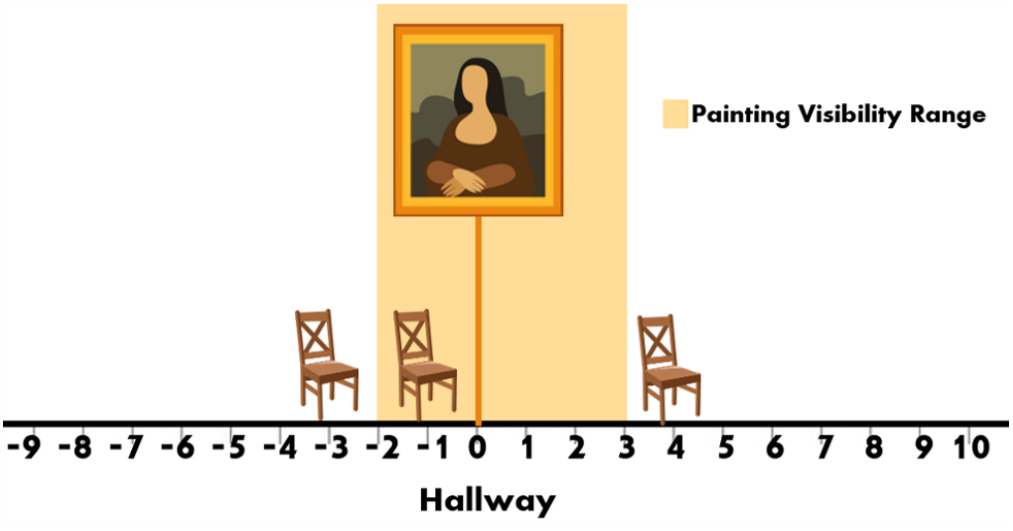


Illustration of the first test case.