2019-2020 ICPC, Asia Jakarta Regional Contest (Online Mirror, ICPC Rules, Teams Preferred)

A. Copying Homework

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Danang and Darto are classmates. They are given homework to create a permutation of N integers from 1 to N. Danang has completed the homework and created a permutation A of N integers. Darto wants to copy Danang's homework, but Danang asks Darto to change it up a bit so it does not look obvious that Darto copied.

The difference of two permutations of N integers A and B, denoted by $diff(A, B)$, is the sum of the absolute difference of A_i and B_i for all i. In other words, $diff(A, B) = \Sigma_{i=1}^{N} |A_i - B_i|$. Darto would like to create a permutation of N integers that maximizes its difference with A. Formally, he wants to find a permutation of N integers B_{max} such that $diff(A, B_{max}) \geq diff(A, B')$ for all permutation of N integers B'.

Darto needs your help! Since the teacher giving the homework is lenient, any permutation of N integers B is considered different with A if the difference of A and B is at least N. Therefore, you are allowed to return any permutation of N integers B such that $diff(A, B) \geq N$.

Of course, you can still return B_{max} if you want, since it can be proven that $diff(A, B_{max}) \geq N$ for any permutation A and $N > 1$. This also proves that there exists a solution for any permutation of N integers A. If there is more than one valid solution, you can output any of them.

Input

Input begins with a line containing an integer: N ($2 \leq N \leq 100\,000$) representing the size of Danang's permutation. The next line contains N integers: A_i ($1 \leq A_i \leq N$) representing Danang's permutation. It is guaranteed that all elements in A are distinct.

Output

Output in a line N integers (each separated by a single space) representing the permutation of N integers B such that $diff(A, B) \geq N$. As a reminder, all elements in the permutation must be between 1 to N and distinct.

Examples

input
4 1 3 2 4
output
4 2 3 1

input
2 2 1
output
1 2

Note

Explanation for the sample input/output #1

With $A = [1, 3, 2, 4]$ and $B = [4, 2, 3, 1]$, $diff(A, B) = |1 - 4| + |3 - 2| + |2 - 3| + |4 - 1| = 3 + 1 + 1 + 3 = 8$. Since $8 \geq 4$, $[4, 2, 3, 1]$ is one of the valid output for this sample.

B. Cleaning Robots

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

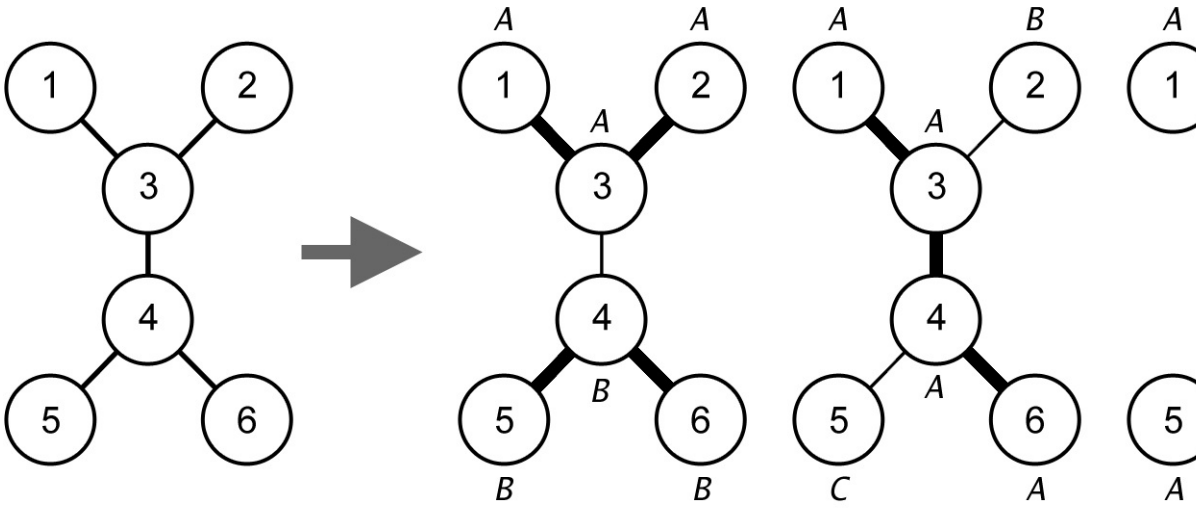
The new ICPC town has N junctions (numbered from 1 to N) which are connected by $N - 1$ roads. It is possible from one junction to go to any other junctions by going through one or more roads. To make sure all the junctions are well-maintained, the government environment agency is planning to deploy their newest advanced cleaning robots. In addition to its cleaning ability, each robot is also equipped with a movement ability such that it can move from one junction to any other junctions connected by roads. However, as you might have guessed, such robots are not cheap. Therefore, the agency is considering the following deployment plan.

Let T_k be the set of junctions which should be cleaned by the k^{th} robot (also known as, the robot's task), and $|T_k| \geq 1$ be the number of junctions in T_k. The junctions in T_k form a **path**, i.e. there exists a sequence of $v_1, v_2, \ldots, v_{|T_k|}$ where $v_i \in T_k$ and $v_i \neq v_j$ for all $i \neq j$ such that each adjacent junction in this sequence is connected by a road. The union of T for all robots is equal to the set of all junctions in ICPC town. On the other hand, no two robots share a common junction, i.e. $T_i \cap T_j = \emptyset$ if $i \neq j$.

To avoid complaints from citizens for an inefficient operation, the deployment plan should be *irreducible*; in other words, there should be no two robots, i and j, such that $T_i \cup T_j$ forms a (longer) path. Note that the agency does not care whether the number of robots being used is minimized as long as all the tasks are irreducible.

Your task in this problem is to count the number of feasible deployment plan given the town's layout. A plan is feasible if and only if it satisfies all the above-mentioned requirements.

For example, let $N = 6$ and the roads are $\{(1, 3), (2, 3), (3, 4), (4, 5), (4, 6)\}$. There are 5 feasible deployment plans as shown in the following figure.



- The first plan uses 2 robots (labeled as A and B in the figure) to clean $\{1, 2, 3\}$ and $\{4, 5, 6\}$.
- The second plan uses 3 robots (labeled as A, B, and C in the figure) to clean $\{1, 3, 4, 6\}$, $\{2\}$, and $\{5\}$.
- The third plan uses 3 robots to clean $\{1, 3, 4, 5\}$, $\{2\}$, and $\{6\}$.
- The fourth plan uses 3 robots to clean $\{1\}$, $\{2, 3, 4, 6\}$, and $\{5\}$.
- The fifth plan uses 3 robots to clean $\{1\}$, $\{2, 3, 4, 5\}$, and $\{6\}$.

No other plans are feasible in this case. For example, the plan $\{\{1, 3\}, \{2\}, \{4, 5, 6\}\}$ is not feasible as the task $\{1, 3\}$ and $\{2\}$ can be combined into a longer path $\{1, 3, 2\}$. The plan $\{\{1, 2, 3, 4\}, \{5\}, \{6\}\}$ is also not feasible as $\{1, 2, 3, 4\}$ is not a path.

Input

Input begins with a line containing an integer: N ($1 \le N \le 100\,000$) representing the number of junctions. The next $N - 1$ lines each contains two integers: u_i v_i ($1 \le u_i < v_i \le N$) representing a road connecting junction u_i and junction v_i. It is guaranteed that it is possible from one junction to go to any other junctions by going through one or more roads.

Output

Output in a line an integer representing the number of feasible deployment plans. As this output can be large, you need to modulo the output by $1\,000\,000\,007$.

Examples

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

input
5 1 2 2 3 2 4 4 5
output
3

Note

Explanation for the sample input/output #1

This is the example from the problem description.

C. Even Path

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Pathfinding is a task of finding a route between two points. It often appears in many problems. For example, in a GPS navigation software where a driver can query for a suggested route, or in a robot motion planning where it should find a valid sequence of movements to do some tasks, or in a simple maze solver where it should find a valid path from one point to another point. This problem is related to solving a maze.

The maze considered in this problem is in the form of a matrix of integers A of $N \times N$. The value of each cell is generated from a given array R and C of N integers each. Specifically, the value on the i^{th} row and j^{th} column, cell (i, j), is equal to $R_i + C_j$. Note that all indexes in this problem are from 1 to N.

A path in this maze is defined as a sequence of cells $(r_1, c_1), (r_2, c_2), \ldots, (r_k, c_k)$ such that $|r_i - r_{i+1}| + |c_i - c_{i+1}| = 1$ for all $1 \le i < k$. In other words, each adjacent cell differs only by 1 row or only by 1 column. An **even path** in this maze is defined as a path in which all the cells in the path contain only even numbers.

Given a tuple $\langle r_a, c_a, r_b, c_b \rangle$ as a query, your task is to determine whether there exists an even path from cell (r_a, c_a) to cell (r_b, c_b). To simplify the problem, it is guaranteed that both cell (r_a, c_a) and cell (r_b, c_b) contain even numbers.

For example, let $N = 5$, $R = \{6, 2, 7, 8, 3\}$, and $C = \{3, 4, 8, 5, 1\}$. The following figure depicts the matrix A of 5×5 which is generated from the given array R and C.

+	3	4	8	5	1
6	9	10	14	11	7
2	5	6	10	7	3
7	10	11	15	12	8
8	11	12	16	13	9
3	6	7	11	8	4

Let us consider several queries:

- $\langle 2, 2, 1, 3 \rangle$: There is an even path from cell $(2, 2)$ to cell $(1, 3)$, e.g., $(2, 2), (2, 3), (1, 3)$. Of course, $(2, 2), (1, 2), (1, 3)$ is also a valid even path.
- $\langle 4, 2, 4, 3 \rangle$: There is an even path from cell $(4, 2)$ to cell $(4, 3)$, namely $(4, 2), (4, 3)$.
- $\langle 5, 1, 3, 4 \rangle$: There is no even path from cell $(5, 1)$ to cell $(3, 4)$. Observe that the only two neighboring cells of $(5, 1)$ are cell $(5, 2)$ and cell $(4, 1)$, and both of them contain odd numbers (7 and 11, respectively), thus, there cannot be any even path originating from cell $(5, 1)$.

Input

Input begins with a line containing two integers: N Q ($2 \le N \le 100\,000$; $1 \le Q \le 100\,000$) representing the size of the maze and the number of queries, respectively. The next line contains N integers: R_i ($0 \le R_i \le 10^6$) representing the array R. The next line contains N integers: C_i ($0 \le C_i \le 10^6$) representing the array C. The next Q lines each contains four integers: r_a c_a r_b c_b ($1 \le r_a, c_a, r_b, c_b \le N$) representing a query of $\langle r_a, c_a, r_b, c_b \rangle$. It is guaranteed that (r_a, c_a) and (r_b, c_b) are two different cells in the maze and both of them contain even numbers.

Output

For each query in the same order as input, output in a line a string "YES" (without quotes) or "NO" (without quotes) whether there exists an even path from cell (r_a, c_a) to cell (r_b, c_b).

Examples

input
5 3

```
6 2 7 8 3
3 4 8 5 1
2 2 1 3
4 2 4 3
5 1 3 4
```

output

```
YES
YES
NO
```

input

```
3 2
30 40 49
15 20 25
2 2 3 3
1 2 2 2
```

output

```
NO
YES
```

Note

Explanation for the sample input/output #1

This is the example from the problem description.

D. Find String in a Grid

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a grid G containing R rows (numbered from 1 to R, top to bottom) and C columns (numbered from 1 to C, left to right) of uppercase characters. The character in the r^{th} row and the c^{th} column is denoted by $G_{r,c}$. You also have Q strings containing uppercase characters. For each of the string, you want to find the number of occurrences of the string in the grid.

An occurrence of string S in the grid is counted if S can be constructed by starting at one of the cells in the grid, going right 0 or more times, and then going down 0 or more times. Two occurrences are different if the set of cells used to construct the string is different. Formally, for each string S, you would like to count the number of tuples $\langle r, c, \Delta r, \Delta c \rangle$ such that:

- $1 \leq r \leq R$ and $r \leq r + \Delta r \leq R$
- $1 \leq c \leq C$ and $c \leq c + \Delta c \leq C$
- $S = G_{r,c}G_{r,c+1} \ldots G_{r,c+\Delta c}G_{r+1,c+\Delta c} \ldots G_{r+\Delta r,c+\Delta c}$

Input

Input begins with a line containing three integers: $R\ C\ Q$ ($1 \leq R, C \leq 500$; $1 \leq Q \leq 200\,000$) representing the size of the grid and the number of strings, respectively. The next R lines each contains C uppercase characters representing the grid. The c^{th} character on the r^{th} line is $G_{r,c}$. The next Q lines each contains a string S containing uppercase characters. The length of this string is a positive integer not more than $200\,000$. The sum of the length of all Q strings combined is not more than $200\,000$.

Output

For each query in the same order as input, output in a line an integer representing the number of occurrences of the string in the grid.

Examples

input

```
3 3 5
ABC
BCD
DAB
ABC
BC
BD
AC
A
```

output

```
2
3
1
0
2
```

input

```
2 3 3
AAA
AAA
A
AAA
AAAAA
```

output

```
6
4
0
```

Note

Explanation for the sample input/output #1

- There are 2 occurrences of "ABC", represented by the tuples $\langle 1, 1, 1, 1 \rangle$ and $\langle 1, 1, 0, 2 \rangle$.
- There are 3 occurrences of "BC", represented by the tuples $\langle 1, 2, 0, 1 \rangle$, $\langle 1, 2, 1, 0 \rangle$, and $\langle 2, 1, 0, 1 \rangle$.
- There is 1 occurrence of "BD", represented by the tuple $\langle 2, 1, 1, 0 \rangle$.
- There is no occurrence of "AC".
- There are 2 occurrences of "A", represented by the tuples $\langle 1, 1, 0, 0 \rangle$ and $\langle 3, 2, 0, 0 \rangle$.

E. Songwriter

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

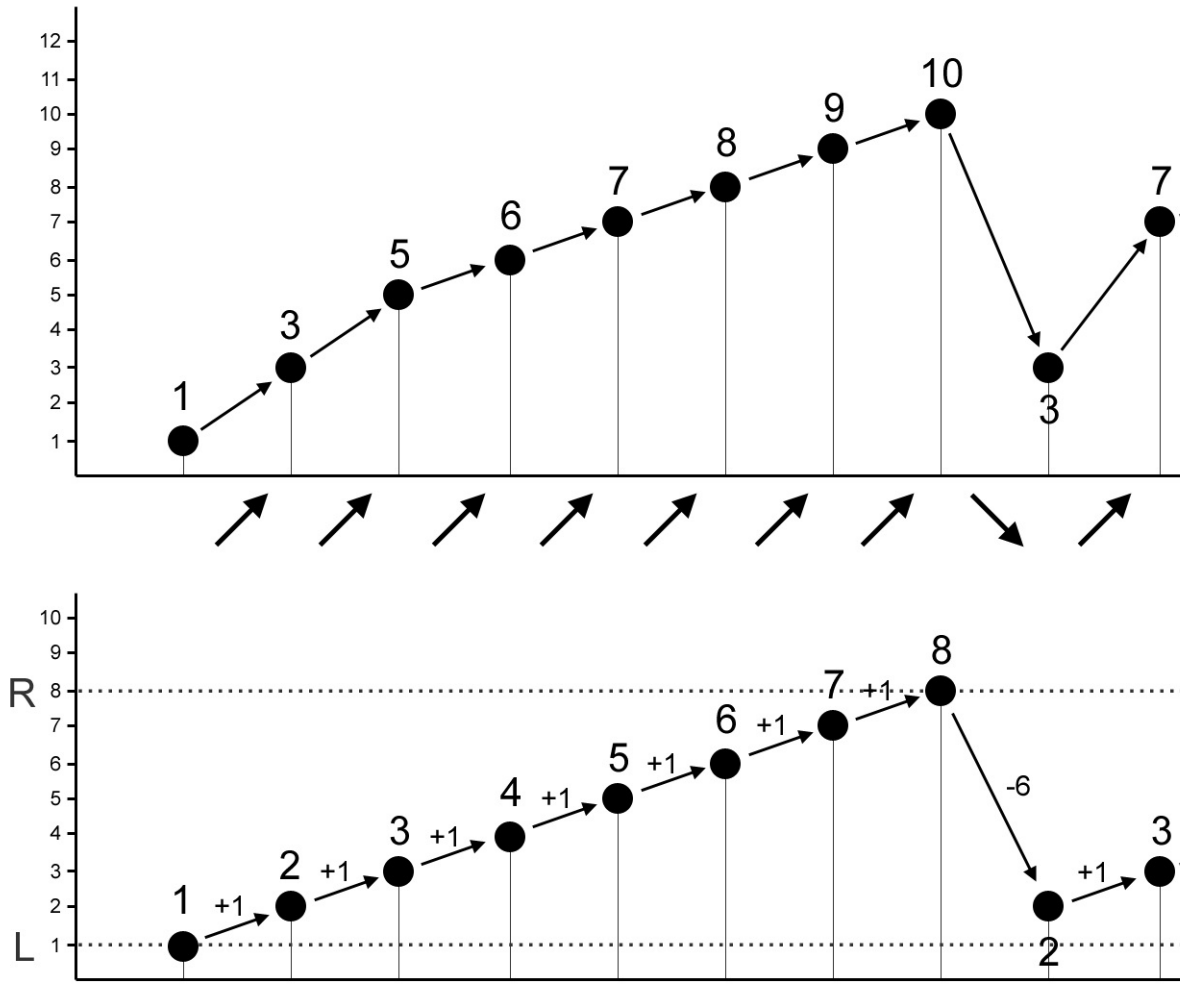
Andi is a mathematician, a computer scientist, and a songwriter. After spending so much time writing songs, he finally writes a catchy melody that he thought as his best creation. However, the singer who will sing the song/melody has a unique vocal range, thus, an adjustment may be needed.

A melody is defined as a sequence of N notes which are represented by integers. Let A be the original melody written by Andi. Andi needs to adjust A into a new melody B such that for every i where $1 \leq i < N$:

- If $A_i < A_{i+1}$, then $B_i < B_{i+1}$.
- If $A_i = A_{i+1}$, then $B_i = B_{i+1}$.
- If $A_i > A_{i+1}$, then $B_i > B_{i+1}$.
- $|B_i - B_{i+1}| \leq K$, i.e. the difference between two successive notes is no larger than K.

Moreover, the singer also requires that all notes are within her vocal range, i.e. $L \leq B_i \leq R$ for all $1 \leq i \leq N$.
Help Andi to determine whether such B exists, and find the lexicographically smallest B if it exists. A melody X is lexicographically smaller than melody Y if and only if there exists j ($1 \leq j \leq N$) such that $X_i = Y_i$ for all $i < j$ and $X_j < Y_j$.

For example, consider a melody $A = \{1, 3, 5, 6, 7, 8, 9, 10, 3, 7, 8, 9, 10, 11, 12, 12\}$ as shown in the following figure. The diagonal arrow up in the figure implies that $A_i < A_{i+1}$, the straight right arrow implies that $A_i = A_{i+1}$, and the diagonal arrow down implies that $A_i > A_{i+1}$.



Supposed we want to make a new melody with $L = 1$, $R = 8$, and $K = 6$. The new melody $B = \{1, 2, 3, 4, 5, 6, 7, 8, 2, 3, 4, 5, 6, 7, 8, 8\}$ as shown in the figure satisfies all the requirements, and it is the lexicographically smallest possible.

Input

Input begins with a line containing four integers: N L R K ($1 \leq N \leq 100\,000$; $1 \leq L \leq R \leq 10^9$; $1 \leq K \leq 10^9$) representing the number of notes in the melody, the vocal range (L and R), and the maximum difference between two successive notes in the new melody, respectively. The next line contains N integers: A_i ($1 \leq A_i \leq 10^9$) representing the original melody.

Output

Output in a line N integers (each separated by a single space) representing the lexicographically smallest melody satisfying all the requirements, or output -1 if there is no melody satisfying all the requirements. Note that it might be possible that the lexicographically smallest melody which satisfies all the requirements to be the same as the original melody.

Examples

input
16 1 8 6 1 3 5 6 7 8 9 10 3 7 8 9 10 11 12 12
output
1 2 3 4 5 6 7 8 2 3 4 5 6 7 8 8

input
16 1 8 6 1 3 5 6 7 8 9 10 3 7 8 9 10 11 12 13
output
-1

input
16 1 10 10 1 3 5 6 7 8 9 10 3 7 8 9 1 11 12 13
output
1 2 3 4 5 6 7 8 1 2 3 4 1 2 3 4

Note

Explanation for the sample input/output #1

This is the example from the problem description.

F. Regular Forestation

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A forestation is an act of planting a bunch of trees to grow a forest, usually to replace a forest that had been cut down. Strangely enough, graph theorists have another idea on how to make a forest, i.e. by cutting down a tree!

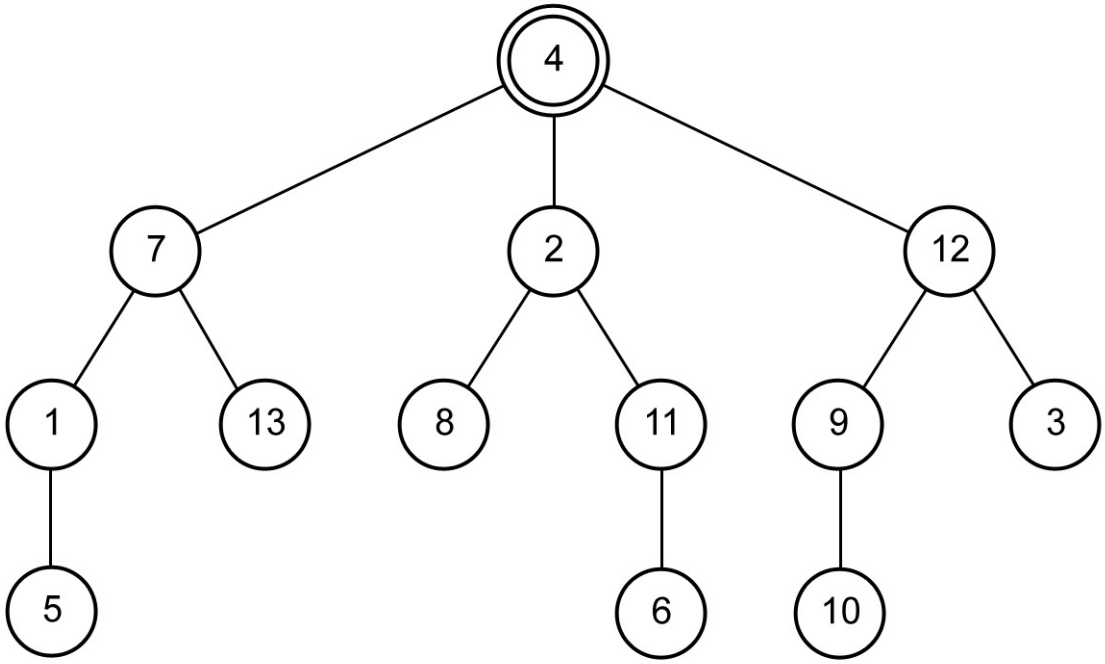
A tree is a graph of N nodes connected by $N - 1$ edges. Let u be a node in a tree U which degree is at least 2 (i.e. directly connected to at least 2 other nodes in U). If we remove u from U, then we will get two or more disconnected (smaller) trees, or also known as forest by graph theorists. In this problem, we are going to investigate a special forestation of a tree done by graph theorists.

Let $V(S)$ be the set of nodes in a tree S and $V(T)$ be the set of nodes in a tree T. Tree S and tree T are **identical** if there exists a bijection $f : V(S) \to V(T)$ such that for all pairs of nodes (s_i, s_j) in $V(S)$, s_i and s_j is connected by an edge in S if and only if node $f(s_i)$ and $f(s_j)$ is connected by an edge in T. Note that $f(s) = t$ implies node s in S corresponds to node t in T.

We call a node u in a tree U as a good cutting point if and only if the removal of u from U causes two or more disconnected trees, and all those disconnected trees are pairwise identical.

Given a tree U, your task is to determine whether there exists a good cutting point in U. If there is such a node, then you should output the maximum number of disconnected trees that can be obtained by removing exactly one good cutting point.

For example, consider the following tree of 13 nodes.



There is exactly one good cutting point in this tree, i.e. node 4. Observe that by removing node 4, we will get three identical trees (in this case, line graphs), i.e. $\{5, 1, 7, 13\}$, $\{8, 2, 11, 6\}$, and $\{3, 12, 9, 10\}$, which are denoted by A, B, and C respectively in the figure.

- The bijection function between A and B: $f(5) = 8$, $f(1) = 2$, $f(7) = 11$, and $f(13) = 6$.
- The bijection function between A and C: $f(5) = 3$, $f(1) = 12$, $f(7) = 9$, and $f(13) = 10$.
- The bijection function between B and C: $f(8) = 3$, $f(2) = 12$, $f(11) = 9$, and $f(6) = 10$.

Of course, there exists other bijection functions for those trees.

Input

Input begins with a line containting an integer: N ($3 \leq N \leq 4000$) representing the number of nodes in the given tree. The next $N - 1$ lines each contains two integers: a_i b_i ($1 \leq a_i < b_i \leq N$) representing an edge (a_i, b_i) in the given tree. It is guaranteed that any two nodes in the given tree are connected to each other by a sequence of edges.

Output

Output in a line an integer representing the maximum number of disconnected trees that can be obtained by removing exactly one good cutting point, or output -1 if there is no such good cutting point.

Examples

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

input
6 1 2 1 3 2 4 3 5 3 6
output
-1

Note

Explanation for the sample input/output #1

This is the example from the problem description.

G. Performance Review

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Randall is a software engineer at a company with N employees. Every year, the company re-evaluates its employees. At the end of every year, the company replaces its several worst-performing employees and replaces with the same number of new employees, so that the company keeps having N employees. Each person has a constant performance and can be represented by an integer (higher integer means better performance), and no two people have the same performance.

The performance of the initial employees are represented by an array of integers $A = [A_1, A_2, \ldots, A_N]$ where A_i is the performance of the i^{th} employee. Randall is employee 1, so his performance is A_1. We will consider the first M years. At the end of the i^{th} year, the company replaces its R_i worst-performing employees and replaces with R_i new employees. The performance of these new employees are represented by an array of integers $B_i = [(B_i)_1, (B_i)_2, \ldots, (B_i)_{R_i}]$ where $(B_i)_j$ is the performance of the j^{th} new employee.

He will consider Q scenarios. On the i^{th} scenario, he will change the value of $(B_{X_i})_{Y_i}$ to Z_i. For each scenario, Randall is wondering whether he will still be in the company after M years. Note that the changes in each scenario are kept for the subsequent scenarios.

Input

Input begins with a line containing three integers: N M Q ($2 \leq N \leq 100\,000$; $1 \leq M, Q \leq 100\,000$) representing the number of

employees, the number of years to be considered, and the number of scenarios, respectively. The next line contains N integers: A_i ($0 \le A_i \le 10^9$) representing the performance of the initial employees. The next M lines each contains several integers: R_i $(B_i)_1$, $(B_i)_2$, $\cdots$, $(B_i)_{R_i}$ ($1 \le R_i < N$; $0 \le (B_i)_j \le 10^9$) representing the number of employees replaced and the performance of the new employees, respectively. It is guaranteed that the sum of R_i does not exceed 10^6. The next Q lines each contains three integers: X_i Y_i Z_i ($1 \le X_i \le M$; $1 \le Y_i \le R_{(X_i)}$; $0 \le Z_i \le 10^9$) representing a scenario. It is guaranteed that all integers in all A_i, $(B_i)_j$, and Z_i (combined together) are distinct.

Output
For each scenario in the same order as input, output in a line an integer 0 if Randall will not be in the company after M years, or 1 if Randall will still be in the company after M years.

Example

input
5 3 3 50 40 30 20 10 4 1 2 3 100 1 4 2 6 7 1 3 300 2 1 400 2 1 5

output
1 0 1

Note
Explanation for the sample input/output #1

Randall performance is represented by 50. For the first scenario, the value of $(B_1)_3$ is updated to 300, causes the following:

- Initially, the performance of the employees is $[50, 40, 30, 20, 10]$.
- At the end of the first year, 4 worst-performing employees are replaced by employees with performance $[300, 100, 2, 1]$. Therefore, the performance of the employees is $[300, 100, 50, 2, 1]$.
- At the end of the second year, the performance of the employees is $[300, 100, 50, 4, 2]$.
- At the end of the third year, the performance of the employees is $[300, 100, 50, 7, 6]$.

Therefore, Randall will still be in the company after 3 years.
For the second scenario, the value of $(B_2)_1$ is updated to 400, causes the following:

- Initially, the performance of the employees is $[50, 40, 30, 20, 10]$.
- At the end of the first year, the performance of the employees is $[300, 100, 50, 2, 1]$. Recall that the change in the first scenario is kept for this scenario as well.
- At the end of the second year, the performance of the employees is $[400, 300, 100, 50, 2]$.
- At the end of the third year, the performance of the employees is $[400, 300, 100, 7, 6]$.

Therefore, Randall will not be in the company after 3 years.

H. Twin Buildings

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

As you might already know, space has always been a problem in ICPC Jakarta. To cope with this, ICPC Jakarta is planning to build **two** new buildings. These buildings should have a shape of a rectangle of the same size. Now, their problem is to find land to build the buildings.

There are N lands available for sale. The i^{th} land has a rectangular shape of size $L_i \times W_i$. For a good feng shui, the building's side should be parallel to the land's sides.

One way is to build the two buildings on two different lands, one on each land (not necessarily with the same orientation). A building of size $A \times B$ can be build on the i^{th} land if and only if at least one of the following is satisfied:

- $A \le L_i$ and $B \le W_i$, or
- $A \le W_i$ and $B \le L_i$.

Alternatively, it is also possible to build two buildings of $A \times B$ on the i^{th} land with the same orientation. Formally, it is possible to build two buildings of $A \times B$ on the i^{th} land if and only if at least one of the following is satisfied:

- $A \times 2 \le L_i$ and $B \le W_i$, or
- $A \times 2 \le W_i$ and $B \le L_i$, or
- $A \le L_i$ and $B \times 2 \le W_i$, or
- $A \le W_i$ and $B \times 2 \le L_i$.

Your task in this problem is to help ICPC Jakarta to figure out the largest possible buildings they can build given N available lands. Note that ICPC Jakarta has to build two buildings of $A \times B$; output the largest possible for $A \times B$.

Input
Input begins with a line containing an integer: N ($1 \le N \le 100\,000$) representing the number of available lands. The next N lines each contains two integers: L_i W_i ($1 \le L_i, W_i \le 10^9$) representing the size of the land.

Output
Output in a line a number representing the largest building that ICPC Jakarta can build with exactly one decimal point (see sample input/output for clarity).

Examples

input
2 5 5 3 4

output
12.5

input
2 2 5 4 3

output
8.0

input
3 10 1 9 8 7 6

output
42.0

Note
Explanation for the sample input/output #1

Two buildings of 2.5×5 can be built both on the first land.

Explanation for the sample input/output #2

Two buildings of 2×4 can be built each on the first and second lands.

Explanation for the sample input/output #3

Two buildings of 7×6 can be built each on the second and third lands.

I. Mission Possible

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Allen, a government secret service, has been assigned to infiltrate a mafia secret base to uncover crucial information regarding the mafia's operations.

The secret base is a rectangular bounded by (x_L, y_L), (x_L, y_R), (x_R, y_L), and (x_R, y_R) in a Cartesian coordinate system where $x_L < x_R$ and $y_L < y_R$. There are N sensors placed inside the secret base. The i^{th} sensor is located at (x_i, y_i) and has an effective sensing radius of r_i which can detect any person who is **strictly** within the radius of r_i from (x_i, y_i). In other words, the i^{th} sensor can detect a person at location (x_a, y_a) if and only if the Euclidean distance of (x_i, y_i) and (x_a, y_a) is strictly less than r_i. It is also known that the Euclidean distance of any two sensors i and j is strictly larger than $r_i + r_j$. Note that the Euclidean distance of two points, (x_a, y_a) and (x_b, y_b), is $\sqrt{|x_a - x_b|^2 + |y_a - y_b|^2}$.

Allen begins his infiltration mission at location (x_s, y_s), and his target is located at (x_t, y_t). Allen has the power to run extremely fast in a straight line while he needs to spend extra time to change his running trajectory (to adjust his footing). Although he is a fast runner, he still needs to make sure that none of the sensors detect him while he is running, i.e. there is no point in his running trajectory which is strictly within a sensor effective sensing radius.

Let $P = \{(x_{p_1}, y_{p_1}), \ldots, (x_{p_{|P|}}, y_{p_{|P|}})\}$ be the set of locations where Allen changes his running trajectory, thus, Allen's running trajectory with P is $(x_s, y_s) \to (x_{p_1}, y_{p_1}) \to \cdots \to (x_{p_{|P|}}, y_{p_{|P|}}) \to (x_t, y_t)$ where $(x_a, y_a) \to (x_b, y_b)$ implies that Allen is running from (x_a, y_a) to (x_b, y_b) in a straight line. The set P is feasible if and only if with P, Allen is not detected by any sensor and is not running out of the secret base (although, Allen is allowed to run along the secret base perimeter). Note that x_p and y_p, $(x_p, y_p) \in P$, are not necessarily integers; they can be **real numbers**.

Your task in this problem is to find any one feasible P which contains no more than 1000 points.

Input

Input begins with a line containing five integers: N x_L y_L x_R y_R ($0 \le N \le 50$; $0 \le x_L < x_R \le 1000$; $0 \le y_L < y_R \le 1000$) representing the number of sensors and the secret base (x_L, y_L, x_R, y_R), respectively. The next line contains two integers: x_s y_s ($x_L < x_s < x_R$; $y_L < y_s < y_R$) representing Allen's initial location. The next line contains two integers: x_t y_t ($x_L < x_t < x_R$; $y_L < y_t < y_R$) representing Allen's target location. It is guaranteed that $x_s \neq x_t$ or $y_s \neq y_t$. The next N lines each contains three integers: x_i y_i r_i ($x_L < x_i - r_i < x_i + r_i < x_R$; $y_L < y_i - r_i < y_i + r_i < y_R$; $1 \le r_i \le 1000$) representing a sensor at location (x_i, y_i) with an effective sensing radius of r_i. It is guaranteed that the Euclidean distance of any two sensors i and j is larger than $r_i + r_j$. It is also guaranteed that the Euclidean distance of (x_s, y_s) and (x_t, y_t) to any sensor i is larger than r_i.

Output

Output in a line an integer representing the size of a feasible P. The next $|P|$ lines each contains two real numbers (separated by a single space); the j^{th} line contains x_j y_j representing the j^{th} point in P. You may output any feasible P with no more than 1000 points.

Due to the nature of the output (floating point), let us define an epsilon ϵ to be 10^{-6} to verify the output. Consider $Q_1 = (x_s, y_s)$, $Q_{j+1} = P_j$ for all $1 \le j \le |P|$, and $Q_{|P|+2} = (x_t, y_t)$. Then, P is considered correct if and only if P contains no more than 1000 points and all of the following are satisfied:

- $x_L - \epsilon \le x_{p_k} \le x_R + \epsilon$ and $y_L - \epsilon \le y_{p_k} \le y_R + \epsilon$ for all $1 \le k \le |P|$ (Allen is not running out of the secret base).
- For all $1 \le k < |Q|$, let S_k be the line segment connecting Q_k and Q_{k+1} (Allen is running in straight line). For all $1 \le i \le N$, let $(x_{k,i}, y_{k,i})$ be the point along S_k that is the closest to the i^{th} sensor's location, (x_i, y_i). Let $d_{k,i}$ be the Euclidean distance between $(x_{k,i}, y_{k,i})$ and (x_i, y_i). Then, the constraint $r_i \le d_{k,i} + \epsilon$ should be satisfied (Allen is not detected by any sensor).
- All points in Q are distinct. Two points, (x_a, y_a) and (x_b, y_b), are considered distinct if and only if $|x_a - x_b| > \epsilon$ or $|y_a - y_b| > \epsilon$.

Examples

input
3 2 2 50 26 4 14 48 14 15 13 7 36 16 6 46 18 3

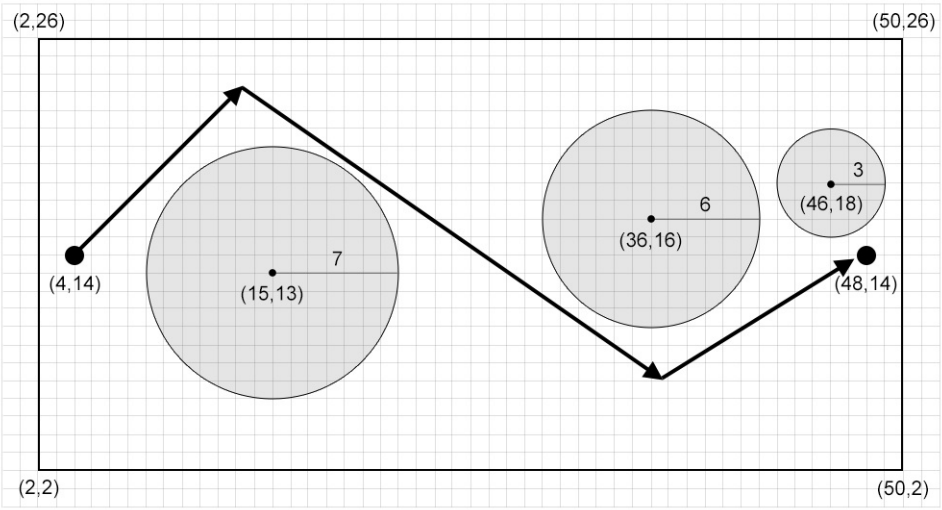
output
2 13.25 23.1234567 36.591003 7.1

input
1 0 0 1000 1000 100 501 900 501 500 251 250

output
0

Note

Explanation for the sample input/output #1



The figure above shows the P from the sample output. Note that there exists a feasible P with only one point in this sample, although you are not required to find such P.

J. Tiling Terrace

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Talia has just bought an abandoned house in the outskirt of Jakarta. The house has a nice and long yard which can be represented as a one-dimensional grid containing $1 \times N$ cells. To beautify the house, Talia is going to build a terrace on the yard by tiling the cells. Each cell on the yard contains either soil (represented by the character '`.`') or rock (represented by the character '`#`'), and there are at most 50 cells containing rocks.

Being a superstitious person, Talia wants to tile the terrace with mystical tiles that have the power to repel ghosts. There are three types of mystical tiles:

- Type-1: Covers 1×1 cell and can only be placed on a soil cell ("`.`").
- Type-2: Covers 1×2 cells and can only be placed on two consecutive soil cells ("`..`").
- Type-3: Covers 1×3 cells and can only be placed on consecutive soil-rock-soil cells ("`.#.`").

Each tile of Type-1, Type-2, and Type-3 has the power to repel G_1, G_2, and G_3 ghosts per day, respectively. There are also some mystical rules which must be followed for the power to be effective:

- There should be no overlapping tiles, i.e. each cell is covered by at most one tile.
- There should be at most K tiles of Type-1, while there are no limitations for tiles of Type-2 and Type-3.

Talia is scared of ghosts, thus, the terrace (which is tiled by mystical tiles) should be able to repel as many ghosts as possible. Help Talia to find the maximum number of ghosts that can be repelled per day by the terrace. Note that Talia **does not need** to tile all the cells on the yard as long as the number of ghosts that can be repelled by the terrace is maximum.

Input
Input begins with a line containing five integers: N K G_1 G_2 G_3 ($1 \le N \le 100\,000$; $0 \le K \le N$; $0 \le G_1, G_2, G_3 \le 1000$) representing the number of cells, the maximum number of tiles of Type-1, the number of ghosts repelled per day by a tile of Type-1, the number of ghosts repelled per day by a tile of Type-2, and the number of ghosts repelled by a tile of Type-3, respectively. The next line contains a string of N characters representing the yard. Each character in the string is either '`.`' which represents a soil cell or '`#`' which represents a rock cell. There are at most 50 rock cells.

Output
Output in a line an integer representing the maximum number of ghosts that can be repelled per day.

Examples

input
6 4 10 25 40 ..#...
output
75

input
6 4 10 100 40 ..#...
output
210

input
7 2 30 10 100 ..#...#
output
160

Note
Explanation for the sample input/output #1

Let "`A`" be a tile of Type-1, "`BB`" be a tile of Type-2, and "`CCC`" be a tile of Type-3. The tiling "`ACCCBB`" in this case produces the maximum number of ghosts that can be repelled, i.e. $10 + 40 + 25 = 75$

Explanation for the sample input/output #2

This sample input has the same yard with the previous sample input, but each tile of Type-2 can repel more ghosts per day. The tiling "`BB#BBA`" or "`BB#ABB`" produces the maximum number of ghosts that can be repelled, i.e. $100 + 100 + 10 = 210$. Observe that the third cell is left untiled.

Explanation for the sample input/output #3

The tiling "`ACCCA.#`", "`ACCC.A#`", or "`.CCCAA#`" produces the maximum number of ghosts that can be repelled, i.e. $30 + 100 + 30 = 160$. Observe that there is no way to tile the last cell.

K. Addition Robot

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Adding two numbers several times is a time-consuming task, so you want to build a robot. The robot should have a string $S = S_1 S_2 \ldots S_N$ of N characters on its memory that represents addition instructions. Each character of the string, S_i, is either '`A`' or '`B`'.

You want to be able to give Q commands to the robot, each command is either of the following types:

- 1 L R. The robot should toggle all the characters of S_i where $L \le i \le R$. Toggling a character means changing it to '`A`' if it was previously '`B`', or changing it to '`B`' if it was previously '`A`'.
- 2 L R A B. The robot should call $f(L, R, A, B)$ and return two integers as defined in the following pseudocode:

```
function f(L, R, A, B):
  FOR i from L to R
    if S[i] = 'A'
      A = A + B
    else
      B = A + B
  return (A, B)
```

You want to implement the robot's expected behavior.

Input
Input begins with a line containing two integers: N Q ($1 \le N, Q \le 100\,000$) representing the number of characters in the robot's memory and the number of commands, respectively. The next line contains a string S containing N characters (each either '`A`' or '`B`') representing the initial string in the robot's memory. The next Q lines each contains a command of the following types.

- 1 L R ($1 \le L \le R \le N$)
- 2 L R A B ($1 \le L \le R \le N$; $0 \le A, B \le 10^9$)

There is at least one command of the second type.

Output
For each command of the second type in the same order as input, output in a line two integers (separated by a single space), the value of A and B returned by $f(L, R, A, B)$, respectively. As this output can be large, you need to modulo the output by $1\,000\,000\,007$.

Example

input

```
5 3
ABAAA
2 1 5 1 1
1 3 5
2 2 5 0 1000000000
```

output

```
11 3
0 1000000000
```

Note

Explanation for the sample input/output #1

For the first command, calling $f(L, R, A, B)$ causes the following:

- Initially, $A = 1$ and $B = 1$.
- At the end of $i = 1$, $A = 2$ and $B = 1$.
- At the end of $i = 2$, $A = 2$ and $B = 3$.
- At the end of $i = 3$, $A = 5$ and $B = 3$.
- At the end of $i = 4$, $A = 8$ and $B = 3$.
- At the end of $i = 5$, $A = 11$ and $B = 3$.

Therefore, $f(L, R, A, B)$ will return $(11, 3)$.
For the second command, string S will be updated to "ABBBB".

For the third command, the value of A will always be 0 and the value of B will always be $1\,000\,000\,000$. Therefore, $f(L, R, A, B)$ will return $(0, 1\,000\,000\,000)$.

L. Road Construction

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are N cities in the country of Numbata, numbered from 1 to N. Currently, there is no road connecting them. Therefore, each of these N cities proposes a road candidate to be constructed.

City i likes to connect with city A_i, so city i proposes to add a direct bidirectional road connecting city i and city A_i. It is guaranteed that no two cities like to connect with each other. In other words, there is no pair of integers i and j where $A_i = j$ and $A_j = i$. It is also guaranteed that any pair of cities are connected by a sequence of road proposals. In other words, if all proposed roads are constructed, then any pair of cities are connected by a sequence of constructed road.

City i also prefers the road to be constructed using a specific material. Each material can be represented by an integer (for example, 0 for asphalt, 1 for wood, etc.). The material that can be used for the road connecting city i and city A_i is represented by an array B_i containing M_i integers: $[(B_i)_1, (B_i)_2, \ldots, (B_i)_{M_i}]$. This means that the road connecting city i and city A_i can be constructed with either of the material in B_i.

There are K workers to construct the roads. Each worker is only familiar with one material, thus can only construct a road with a specific material. In particular, the i^{th} worker can only construct a road with material C_i. Each worker can only construct at most one road. You want to assign each worker to construct a road such that any pair of cities are connected by a sequence of constructed road.

Input

Input begins with a line containing two integers: N K ($3 \le N \le 2000$; $1 \le K \le 2000$) representing the number of cities and the number of workers, respectively. The next N lines each contains several integers: A_i M_i $(B_i)_1$, $(B_i)_2$, $\cdots$, $(B_i)_{M_i}$ ($1 \le A_i \le N$; $A_i \ne i$; $1 \le M_i \le 10\,000$; $0 \le (B_i)_1 < (B_i)_2 < \cdots < (B_i)_{M_i} \le 10^9$) representing the bidirectional road that city i likes to construct. It is guaranteed that the sum of M_i does not exceed $10\,000$. It is also guaranteed that no two cities like to connect with each other and any pair of cities are connected by a sequence of road proposals. The next line contains K integers: C_i ($0 \le C_i \le 10^9$) representing the material that is familiarized by the workers.

Output

If it is not possible to assign each worker to construct a road such that any pair of cities are connected by a sequence of constructed road, simply output `-1` in a line. Otherwise, for each worker in the same order as input, output in a line two integers (separated by a single space): u and v in any order. This means that the worker constructs a direct bidirectional road connecting city u and v. If the worker does not construct any road, output "`0 0`" (without quotes) instead. Each pair of cities can only be assigned to at most one worker. You may output any assignment as long as any pair of cities are connected by a sequence of constructed road.

Examples

input

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

output

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

input

```
4 5
2 2 10 20
3 2 2 3
4 2 3 4
2 2 4 5
1 2 3 4 5
```

output

```
-1
```

Note

Explanation for the sample input/output #1

We can assign the workers to construct the following roads:

- The first worker constructs a road connecting city 1 and city 2.
- The second worker constructs a road connecting city 2 and city 3.
- The third worker constructs a road connecting city 3 and city 4.
- The fourth worker does not construct any road.
- The fifth worker constructs a road connecting city 4 and city 2.

Therefore, any pair of cities are now connected by a sequence of constructed road.
Explanation for the sample input/output #2

There is no worker that can construct a road connecting city 1, thus city 1 is certainly isolated.