Problem A. Ring Road

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	1024 mebibytes

KOI City consists of N intersections and $N-1$ two-way roads. You can travel between two different intersections using only the given roads. In other words, the city's road network forms a tree structure. Roads are on a two-dimensional plane, and two roads do not intersect at locations other than the endpoints. Each road has an non-negative integer weight. This weight represents the time it takes to use the road.

KOI City was a small town until a few decades ago but began to expand rapidly as people arrived. In the midst of rapid expansion, the mayor had numbered the intersections between 1 and N for administrative convenience. The number system satisfies the following properties.

- Intersection 1 is the center of the city and is incident to at least 2 roads.
- The numbers assigned to intersections form one of the pre-orders of the tree rooted at intersection 1: for any subtree, the number of its root is the least number in that subtree.
- For each intersection, consider the lowest-numbered intersection among all adjacent (directly connected by road) intersections. When you list all adjacent intersections in a counterclockwise order starting from this intersection, the numbers go in increasing order.

With a large influx of people to KOI City, the traffic congestion problem has intensified. To solve this problem, the mayor connected the outermost cities with the *outer ring road*. Let $\{v_1, v_2, \ldots, v_k\}$ be the increasing sequence of numbers of all the intersections incident to exactly one road. For each $1 \le i \le k$, the mayor builds a two-way road between intersection v_i and intersection $v_{(i \bmod k)+1}$. The weight of each road is a nonnegative integer w_i. Due to the nature of the numbering system, you can observe that the outer ring road can be added in a two-dimensional plane in a way such that two roads do not intersect at any location except at the endpoint.

You are trying to build a navigation system for KOI city. The navigation system should answer Q queries of the form (u, v). For each query, the navigation system should return the shortest time it takes to move from intersection u to intersection v. The time to move through a path equals the sum of the weights of edges in the path.

Given a road network structure, write a program that efficiently answers Q queries.

Input

The first line contains the number of intersections N in the KOI City ($4 \le N \le 100\,000$).

Each of the next $N-1$ lines contains two space-separated integers p_i and c_i. They indicate that there is a two-way road with weight c_i connecting intersection p_i and intersection $i+1$ ($1 \le p_i \le i$, $0 \le c_i \le 10^{12}$).

Let k be the number of intersections incident to exactly one road in the original tree, and let $\{v_1, v_2, \ldots, v_k\}$ be the increasing sequence of their numbers. On the next line, k space-separated integers $w_1, w_2, \ldots, w_k$ are given. This indicates that the weight of the outer ring road connecting the intersection v_i and intersection $v_{(i \bmod k)+1}$ is w_i ($0 \le w_i \le 10^{12}$).

The next line contains the number of queries Q ($1 \le Q \le 250\,000$).

Each of the next Q lines contains two integers u and v denoting the intersections of interest ($1 \le u, v \le N$ and $u \ne v$).

Output

For each query, print a line with a single integer: the shortest time to move from u to v.

Examples

standard input	*standard output*
4	9
1 9	8
1 8	0
1 0	9
9 9 9	9
6	8
1 2	
1 3	
1 4	
2 3	
2 4	
3 4	

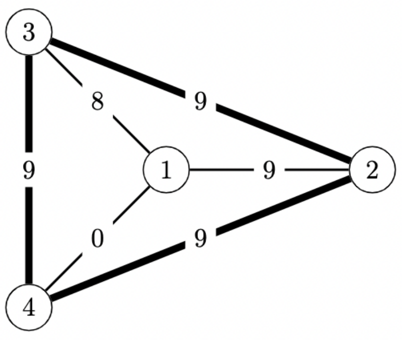
standard input	*standard output*
11	7
1 9	8
1 8	8
3 0	7
4 7	7
4 1	7
3 6	0
1 0	7
8 7	1
8 1	7
10 6	7
0 0 0 0 0 0	7
21	1
1 2	7
1 3	0
1 4	7
1 5	0
1 6	8
1 7	1
1 8	6
1 9	0
1 10	
1 11	
7 1	
8 2	
9 3	
10 4	
11 5	
1 6	
2 7	
3 8	
4 9	
5 10	
6 11	

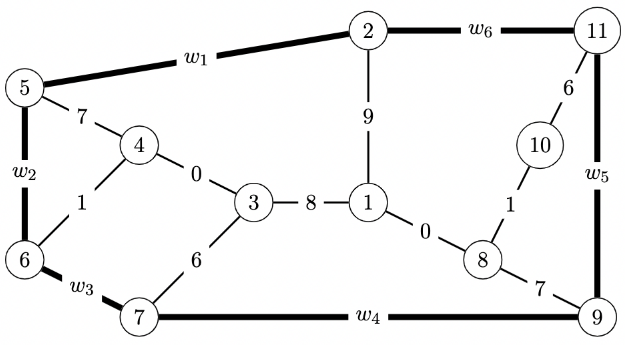
standard input	*standard output*

11
1 9
1 8
3 0
4 7
4 1
3 6
1 0
8 7
8 1
10 6
1000000000000 1000000000000
1000000000000 1000000000000
1000000000000 1000000000000
21
1 2
1 3
1 4
1 5
1 6
1 7
1 8
1 9
1 10
1 11
7 1
8 2
9 3
10 4
11 5
1 6
2 7
3 8
4 9
5 10
6 11
	9
	8
	8
	15
	9
	14
	0
	7
	1
	7
	14
	9
	15
	9
	22
	9
	23
	8
	15
	16
	16

Note

In the third sample, the line with $w_1, w_2, \ldots, w_k$ (in red) is split into several lines for readability.





The picture on the left corresponds to the first sample. The picture on the right corresponds to the second and third samples.

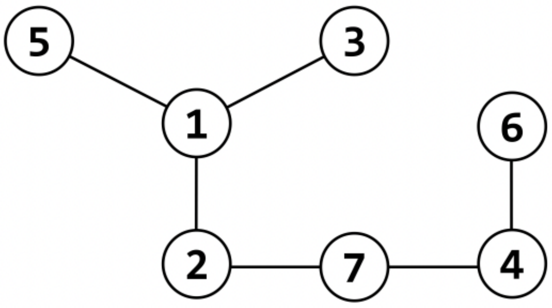
Problem B. Query on a Tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 mebibytes

You are given a tree where vertices are labeled with integers $1, 2, \ldots, N$.

For a subset of vertices $S \subseteq \{1, 2, \ldots, N\}$, we say two vertices (u, v) are *connected under* S if there exists a path that only passes through the vertices in S. Note that this includes endpoints of the path, so $u, v \in S$ should hold.

For example, consider the following tree and the set $S = \{1, 2, 3, 4, 5, 6\}$.



In this case, $(1, 2)$, $(3, 5)$ and $(4, 6)$ are connected under S, while $(1, 6)$ and $(2, 7)$ are not connected under S.

Let *strength*(S) be the number of pairs of vertices (u, v) such that $u \neq v$ and (u, v) are connected under S. You are given Q queries, where each query contains a set S. For each query, you should compute the quantity *strength*(S).

Input

The first line contains a single integer N, the number of vertices ($2 \leq N \leq 250\,000$).

Each of the next $N - 1$ lines contains two space-separated integers a and b: the vertices connected by an edge ($1 \leq a, b \leq N$). Together, the edges form a tree.

The next line contains a single integer Q, the number of queries ($1 \leq Q \leq 100\,000$).

Each of the next Q lines contains a query, denoted by space-separated integers. A query starts with an integer K, the size of the set ($1 \leq K \leq N$). It is followed by K distinct integers from 1 to N in arbitrary order: the vertices of set S.

The sum of K in each test case is at most $1\,000\,000$.

Output

For each of the Q queries, print a single line with the integer *strength*(S) as defined above.

Example

standard input	*standard output*

7
1 2
1 3
1 5
2 7
4 6
4 7
6
1 1
2 1 2
4 1 2 3 4
5 1 2 4 6 7
6 1 2 3 4 5 6
7 1 2 3 4 5 6 7
	0
	1
	3
	10
	7
	21

Problem C. A+B Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 mebibytes

In the era of constructives and ad-hocs, what could be more sacrilegious than combining two query problems into one?

KOI City consists of N intersections and $N - 1$ two-way roads. You can travel between two different intersections using only the given roads. In other words, the city's road network forms a tree structure. Roads are on a two-dimensional plane, and two roads do not intersect at locations other than the endpoints. Each road has an non-negative integer weight. This weight represents the time it takes to use the road.

KOI City was a small town until a few decades ago but began to expand rapidly as people arrived. In the midst of rapid expansion, the mayor had numbered the intersections between 1 and N for administrative convenience. The number system satisfies the following properties.

- Intersection 1 is the center of the city and is incident to at least 2 roads.
- The numbers assigned to intersections form one of the pre-orders of the tree rooted at intersection 1: for any subtree, the number of its root is the least number in that subtree.
- For each intersection, consider the lowest-numbered intersection among all adjacent (directly connected by road) intersections. When you list all adjacent intersections in a counterclockwise order starting from this intersection, the numbers go in increasing order.

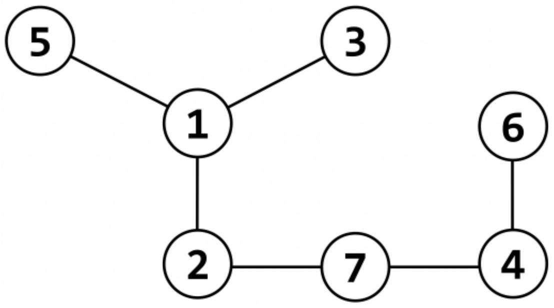
With a large influx of people to KOI City, the traffic congestion problem has intensified. To solve this problem, the mayor connected the outermost cities with the *outer ring road*. Let $\{v_1, v_2, \ldots, v_k\}$ be the increasing sequence of numbers of all the intersections incident to exactly one road. For each $1 \leq i \leq k$, the mayor builds a two-way road between intersection v_i and intersection $v_{(i \bmod k)+1}$. The weight of each road is a nonnegative integer w_i. Due to the nature of the numbering system, you can observe that the outer ring road can be added in a two-dimensional plane in a way such that two roads do not intersect at any location except at the endpoint.

However, resolving traffic congestion only reduces commute times, making it easier for capitalists to exploit workers. Workers would not fall for the capitalists' disgusting plot — they want to go back to the good old days when they could apply heavy-light and centroid decomposition in KOI City! The workers successfully carried out the socialist revolution and overthrew the capitalist regime. Now they want to rebuild the structure of the existing KOI city by creating a **new tree**, which satisfies the following:

- Let K be the number of vertices in the new tree; $K \leq 4N$ should hold. From now on, we will label vertices of the new tree as $1, 2, \ldots, K$.
- For each vertex i of the new tree, there is a corresponding set X_i which is a subset of $\{1, 2, \ldots, N\}$.
- For all roads (u, v) in the KOI City (both tree and outer ring roads), there exists a set X_i where $\{u, v\} \subseteq X_i$.
- For all $1 \leq j \leq N$, let S_j be the set of vertices $1 \leq i \leq K$ such that $j \in X_i$. Then S_j must be **non-empty**, and should be a **revolutionary set** on the new tree.
- For all $1 \leq i \leq K$, it is true that $|X_i| \leq 4$.

For a tree T and a set S which is a subset of vertices of T, the set S is **revolutionary** on T if for all vertices $u, v \in S$ it is connected under S. Two vertices (u, v) are *connected under* S if there exists a path in T that only passes through the vertices in S.

For example, consider the following tree and the set $S = \{1, 2, 3, 4, 5, 6\}$.



In this case, $(1, 2)$, $(3, 5)$ and $(4, 6)$ are connected under S, while $(1, 6)$ and $(2, 7)$ are not connected under S.

Input

The first line contains the number of intersections N in the KOI City ($4 \leq N \leq 100\,000$).

Each of the next $N - 1$ lines contains a single integer p_i. This indicates that there is a two-way road connecting intersection p_i and intersection $i + 1$ ($1 \leq p_i \leq i$). Note that these are not outer ring roads.

Output

On the first line, print the number of vertices in the new tree K. Your answer should satisfy $1 \leq K \leq 4N$.

Then print K lines. On i-th of these lines, print $|X_i| + 1$ space-separated integers. The first integer should be the size of set X_i. The next $|X_i|$ integers should be elements of X_i in any order.

In each of the next $K - 1$ lines, print two space-separated integers a and b, denoting that there exists an edge connecting a and b in the new tree.

It can be proved that the answer always exists.

Example

standard input	*standard output*
4 1 1 1	1 4 1 2 3 4

Problem D. Building Bombing

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

KAIST has a series of N buildings in a row, numbered from 1 to N, from left to right. Building i has a height of h_i. Building i is visible from the left if and only if every building on its left has a height strictly less than h_i.

Your lab is located in building number L. Since your favorite number is K, you want to make your lab building the K-th tallest building visible from the left. To achieve your goal, you will blow up some of the buildings.

For example, suppose there are $N = 7$ buildings in a row and their heights are $[10, 30, 90, 40, 60, 60, 80]$. Your lab is located at building number $L = 2$ and your favorite number is $K = 3$. After blowing up buildings 3 and 7, the buildings visible from the left will be buildings 1, 2, 4, and 5. Then your lab becomes the 3rd tallest building visible from the left, as desired.

What is the minimum number of buildings to blow up to make your lab building the K-th tallest building visible from the left?

Input

The first line contains three space-separated integers N, L, and K.

The second line contains N space-separated integers $h_1, \ldots, h_N$.

- $1 \le L \le N \le 100\,000$
- $1 \le K \le 10$
- $1 \le h_i \le 10^9$ $(1 \le i \le N)$

Output

Output the minimum number of buildings to blow up to make your lab building the K-th tallest building visible from the left. If it is impossible to do so, output -1 instead.

Examples

standard input	*standard output*

`7 2 3`
`10 30 90 40 60 60 80`
	`2`

`3 2 2`
`30 20 10`
	`-1`

Problem E. Double-Colored Papers

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

In your factory, you are making two kinds of colored paper, one colored red, and the other colored blue.

Each red-colored paper has a string S written on it: it is made of $|S|$ unit squares in a row, and S_i is written on the i-th square from the left.

Each blue-colored paper has a string T written on it: it is made of $|T|$ unit squares in a row, and T_i is written on the i-th square from the left.

You plan to make a new kind of paper called *double-colored paper* out of red and blue paper. To do so, you will cut a piece of red paper to leave a continuous part with positive integer length, then do the same with a piece of blue paper. After that, you will glue the end of the red piece to the start of the blue piece.

For example, suppose S is `abcde` and T is `fghij`. You can make a *double-colored paper* with string `bcdfg` or `abcij` written on it. However, you cannot make a *double-colored paper* with string `acdghij` or `fghij` written on it. (Here the underlined string denotes a red piece, and the rest denotes a blue piece.) Two pieces of *double-colored paper* are considered the same if they have the same red string and the same blue string written on them.

Among all different pieces of *double-colored paper* that can be made, you want to know the one with the lexicographically K-th smallest string written on it. Note that there may be papers with the same strings written on them, but with different lengths of red paper: in this case, you may order them arbitrarily.

Input

The first line contains the string S.

The second line contains the string T.

The third line contains the integer K.

- $1 \le |S| \le 75\,000$
- $1 \le |T| \le 75\,000$
- S and T consist of lowercase English letters
- $1 \le K \le 8 \cdot 10^{18}$

Output

If the total number of possible *double-colored papers* is strictly less than K, output -1.

Otherwise, output the lexicographically K-th smallest string of all possible *double-colored papers* that can be made.

Example

standard input	*standard output*

`tww`
`wtw`
`21`
	`wwtw`

Problem F. Making Number

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	1024 mebibytes

You are given two positive integers X and Y of the same length in base 10. Let us define Z as the positive integer in base 10 satisfying the following conditions.

- The digits of Z should be a rearrangement of the digits of X. Leading zeros in Z are not allowed. For example, if $X = 1103$, Z can be 1103 or 3101, but Z cannot be 2110, 301, nor 0131.
- $Y \le Z$.
- Z is the minimum value satisfying the above conditions.

You have to perform Q queries. Each query is one of the following:

- Given i and x, change the i-th digit of Y into x.
- Given i, output the i-th digit of Z. If there is no such Z, print -1.

The digits of an integer are numbered from left to right starting from 1. For example, The third digit of 1234 is 3.

Input

The first line contains two space-separated integers, X and Y.

The second line contains a single integer Q, the number of queries.

Each of the following Q lines contains space-separated integers describing the queries. Each line has one of the following forms, where the first integer represents the type of the query:

- "`1` i x": Change the i-th digit of Y to x.
- "`2` i": Output the i-th digit of Z. If there is no such Z, print -1.

It is guaranteed that there is at least one query of type `2`.

Let $\text{len}(A)$ be the number of digits in a positive integer A.

- $1 \le X, Y < 10^{100\,000}$
- $1 \le Q \le 100\,000$
- $\text{len}(X) = \text{len}(Y)$
- The first digits of X and Y are not 0.
- For a query of type `1`, $1 \le i \le \text{len}(Y)$, $0 \le x \le 9$, and if $i = 1$, then $x \ne 0$.
- For a query of type `2`, $1 \le i \le \text{len}(Y)$.

Output

For each query of type `2`, output a single line with the answer to the query.

Examples

standard input	*standard output*
3304 1615	3
6	4
2 3	0
2 4	3
1 1 3	
2 2	
1 2 4	
2 1	
838046 780357	8
10	0
2 1	3
2 2	4
1 2 4	6
2 3	8
2 4	-1
1 4 5	
2 5	
2 6	
1 1 9	
2 2	
2950 9052	9
4	0
2 1	5
2 2	2
2 3	
2 4	

Problem G. One Path

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given a tree T consisting of N vertices. Each edge has a positive integer weight.

You can perform the following operation on the given tree.

- Delete an edge from the graph, then add a new edge between any two distinct vertices. The weight of the new edge must be the same as the weight of the deleted edge. The resulting graph need not be a tree.

We define the weight of a path as the sum of the weights of the edges on the path. The distance between two vertices u and v is defined as the weight of the *shortest path* from u to v — having the minimum weight. If there is no such path, we define the distance as 0.

The weight of a graph is the maximum of the weights between any two vertices.

Your task is to find the largest weight of the graph that can be obtained by performing the operation exactly i times, for $i = 0, 1, \ldots, K$.

Input

The first line contains two space-separated integers, N and K.

The i-th of the following $N - 1$ lines contains three space-separated integers, u_i, v_i, and w_i, representing an undirected edge that connects two different vertices u_i and v_i with a weight of w_i.

It is guaranteed that the edges form a tree.

- $2 \leq N \leq 2000$
- $0 \leq K \leq 2000$
- $1 \leq u_i < v_i \leq N$ $(1 \leq i \leq N - 1)$
- $1 \leq w_i \leq 10^9$ $(1 \leq i \leq N - 1)$

Output

Output $K+1$ space-separated integers. The i-th integer should be equal to the largest weight of the graph that can be obtained by performing the operation exactly $i - 1$ times.

Examples

standard input	*standard output*

`5 1`
`1 3 2`
`4 5 4`
`3 4 3`
`2 3 7`
	`14 16`

`7 2`
`1 2 4`
`2 3 6`
`2 4 2`
`4 5 5`
`2 6 1`
`4 7 3`
	`13 20 21`

Problem H. Permutation Arrangement

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given an array a of length N. Each element of a is either -1 or an integer between 1 and N. Each number between 1 and N appears at most once in a. Also, no two adjacent elements of a differ by exactly 1.

You are to find the lexicographically smallest permutation p of $\{1, 2, \ldots, N\}$ satisfying the following.

- if $a_i \neq -1$, then $a_i = p_i$ $(1 \leq i \leq N)$;
- $|p_i - p_{i+1}| \neq 1$ $(1 \leq i \leq N-1)$.

Input

The first line contains one integer, N.

The second line contains space-separated N integers: elements of the array a.

- $1 \leq N \leq 200\,000$
- $1 \leq a_i \leq N$ or $a_i = -1$ $(1 \leq i \leq N)$
- $a_i \neq a_j$ or $a_i = -1$ $(1 \leq i < j \leq N)$
- $|a_i - a_{i+1}| \neq 1$ $(1 \leq i \leq N-1)$

Output

If there is no permutation p satisfying the condition, then output a single integer -1.

Otherwise, output the lexicographically smallest permutation p.

Examples

standard input	*standard output*

`10`
`3 -1 10 -1 8 -1 -1 -1 -1 -1`
	`3 1 10 2 8 4 6 9 5 7`

`2`
`-1 -1`
	`-1`

Problem I. Similarity Graph

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Let p and q be two permutations of $\{1, 2, \ldots, N\}$.

A *similarity graph* of p and q, $S(p, q)$, is defined as follows:

- $S(p, q)$ has N labeled vertices, numbered from 1 to N.
- There is a edge between vertices i and j $(1 \le i < j \le N)$ if and only if $p_i < p_j$ and $q_i < q_j$ are both true, or both false.

You are given a simple undirected graph G with N labeled vertices, numbered from 1 to N.

Find a pair (p, q) of permutations of $\{1, 2, \ldots, N\}$ satisfying $S(p, q) = G$.

Input

The first line contains one integer, N.

Each of the next N lines contains N space-separated integers. The j-th integer of the i-th line, $E(i, j)$, is 1 if there is an edge between vertices i and j, or 0 otherwise.

- $1 \le N \le 100$
- $0 \le E(i, j) \le 1$ $(1 \le i, j \le N)$
- $E(i, j) = E(j, i)$ $(1 \le i < j \le N)$
- $E(i, i) = 0$ $(1 \le i \le N)$

Output

If it is impossible to find p and q satisfying the condition, output `NO`.

Otherwise, output `YES` on the first line. On the following two lines, output p and q. If there are multiple answers, output any one of them.

Examples

standard input	*standard output*

4
0 1 0 1
1 0 0 0
0 0 0 1
1 0 1 0
	YES
	1 2 3 4
	2 4 1 3

6
0 1 0 1 0 1
1 0 0 0 1 0
0 0 0 1 1 1
1 0 1 0 0 0
0 1 1 0 0 0
1 0 1 0 0 0
	NO

Problem J. Squirrel Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

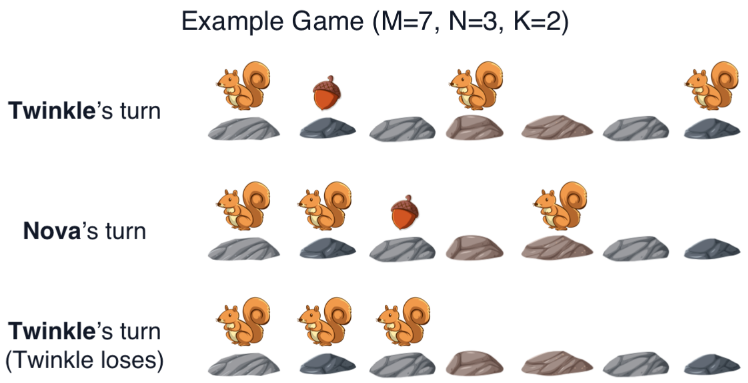
Twinkle and Nova are walking in a national park. There are M stones laid out in the park at positions $1, \ldots, M$, from left to right. There are also N squirrels on the stones at $x_1, \ldots, x_N$, from left to right. The squirrels are on different stones from each other, and they are all facing left.

Twinkle suggests the following game to Nova. Twinkle and Nova take turns alternately. On each turn, a player has to place an acorn on one of the stones without a squirrel. Also, there must be at least one squirrel to the right of the acorn.

After placing an acorn, the leftmost K squirrels among the squirrels to the right of the acorn start running towards the acorn at the same time. (If there are less than K squirrels to the right of the acorn, all of them start running.) All the squirrels run at the same speed. Once any of the squirrels reaches the acorn, all the squirrels immediately stop. The squirrel who has reached the acorn puts the acorn into its cheek pouch, effectively removing the acorn from the stone.

If there is no valid stone to place an acorn on, the player currently taking the turn immediately loses.

Twinkle goes first. Determine who will win if Twinkle and Nova both play optimally.



Input

The first line contains three space-separated integers, M, N, and K.

The second line contains N space-separated integers $x_1, \ldots, x_N$.

- $1 \le N \le M \le 100\,000$
- $1 \le K \le 10$
- $1 \le x_1 < \ldots < x_N \le M$

Output

If Twinkle wins, output `Twinkle`. Otherwise, output `Nova`.

Examples

standard input	*standard output*
7 3 2 1 4 7	Nova
7 3 1 1 4 7	Twinkle

Problem K. Two Paths

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	1024 mebibytes

You are given a tree T consisting of N vertices. Each edge has a positive integer weight. The weight of a path P in T is defined as the sum of weights of edges in P, denoted by $W(P)$.

You are given a total of Q queries, each containing two vertices, u and v, and two integers, A and B. For each query, you are to find two simple paths P_1 and P_2 in T satisfying the following requirements.

- P_1 and P_2 don't share a vertex.
- P_1 starts from u, and P_2 starts from v.
- Among all P_1 and P_2 satisfying the conditions above, the value of $A \cdot W(P_1) + B \cdot W(P_2)$ should be maximized.

You should output the value of $A \cdot W(P_1) + B \cdot W(P_2)$ for each query.

Input

The first line contains two space-separated integers N and Q.

Each of the following $N - 1$ lines contains three space-separated integers u, v, w. This means that there is an edge in T connecting vertices u and v with weight w. Together these edges form a tree.

Each of the following Q lines contains four space-separated integers u, v, A, B, denoting a single query.

- $2 \le N \le 200\,000$
- $1 \le Q \le 500\,000$
- $1 \le u < v \le N$ for both edges and queries
- $1 \le w \le 10\,000$
- $1 \le A, B \le 2 \cdot 10^9$

Output

For each query, output a single line with an integer: the maximum possible value of $A \cdot W(P_1) + B \cdot W(P_2)$.

Example

standard input	standard output

6 4
1 2 4
2 5 5
2 3 7
3 6 5
3 4 4
1 4 1 1
1 4 2 1
5 6 1 1
5 6 1 10
	18
	32
	18
	160

Problem L. Village Planning

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 mebibytes

As the mayor of the RUN town, you are planning to build a new village. The village consists of houses and bidirectional roads connecting two different houses. Roads are organized in such a way that no two roads connect the same pair of houses. In other words, the village can be treated as a simple graph where houses corresponds to vertices and roads corresponds to bidirectional edges. Note that the village may be disconnected.

You want your village to be as simple as possible. Therefore, for any distinct houses i and j, there should be at most K simple paths from house i to house j.

Let N be the number of houses. The score of the village is

$$\prod_{1 \le i < j \le N} A_{f(i,j)},$$

where $f(i,j)$ is the number of simple paths from house i to house j.

While the number of houses is not determined yet, you know that it will be an integer between 2 and M. You should calculate the sum of the scores for all possible villages with N houses for each N from 2 to M.

Since the answers can be large, output them modulo 998 244 353.

Input

The first line contains two space-separated integers M and K.

The second line contains $K+1$ space-separated integers $A_0, \ldots, A_K$.

- $2 \le M \le 100\,000$
- $0 \le K \le 3$
- $1 \le A_i < 998\,244\,353$ $(0 \le i \le K)$

Output

For each N from 2 to M, output the sum of the scores for all possible villages with N houses, modulo 998 244 353. The answers should be separated by single spaces. Note that $998\,244\,353 = 119 \cdot 2^{23} + 1$ is a prime number.

Examples

standard input	*standard output*
4 0 2	2 8 64
5 1 3 4	7 327 96721 169832849
6 2 5 6 7	11 1566 3000672 306031599 466869291
7 3 8 9 10 11	17 5427 31856976 326774674 449014006 997476587

Problem M. Window Arrangement

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

KAIST is running out of budget — they need some money! They thought the dormitories were way too luxurious compared to the other buildings of KAIST; they planned to sell all the dormitory buildings and build a new completely non-aesthetic one.

The new dormitory will have grid shape — it can't be more boring than this — of size $N \times M$, each cell being a room for the students to live in. We are going to add some windows, because we want students to get some sunlight during the daytime!

We plan to have exactly $w_{i,j}$ windows in the room (i, j). Each room is surrounded by 4 unit edges of the grid. A window can be built on the side of a unit edge of the grid, and at most one window can be built on each side of a unit edge — so each cell has between 0 and 4 windows. A window is one-sided: a window on the opposite side of a unit edge does not count as a window in the room.

Unfortunately, students will experience huge discomfort when their private space is watched by someone else through the window. The **total discomfort** is the number of pairs of students $\{a, b\}$ such that a and b can see each other's private space through the window.

In other words, if a unit edge has windows on both sides, the total discomfort increases by the product of the numbers of people living in the two rooms sharing the window.

You are given $w_{i,j}$, the number of windows planned for the room (i, j), and $p_{i,j}$, the number of people living in the room (i, j). Your task is to find the minimum total discomfort that can be achieved by arranging the windows properly.

Input

The first line contains two integers N and M: the dimensions of the grid.

Each of the following N lines contains M space-separated integers $p_{i,j}$.

Each of the following N lines contains M space-separated integers $w_{i,j}$.

- $1 \le N \le 50$
- $1 \le M \le 50$
- $1 \le p_{i,j} \le 1000$ ($1 \le i \le N$, $1 \le j \le M$)
- $0 \le w_{i,j} \le 4$ ($1 \le i \le N$, $1 \le j \le M$)

Output

Output one integer: the minimum total discomfort.

Examples

standard input	*standard output*
`4 3`	`178`
`1 7 10`	
`7 2 8`	
`7 9 10`	
`4 6 4`	
`3 3 3`	
`3 2 4`	
`4 3 4`	
`2 2 3`	
`4 3`	`0`
`2 2 9`	
`9 8 4`	
`8 4 5`	
`7 5 2`	
`0 1 0`	
`1 0 1`	
`0 0 1`	
`0 1 0`	