A. Replication - Part 1

Limits: 2 sec., 512 MiB

Replication - the process of creating two daughter DNA molecules based on the parent DNA molecule. DNA replication is performed by a various of complex operations, consisting of 15-20 different protein-enzymes. In order to replicate, viruses are constantly dividing their DNA - thereby shrinking it. For viruses, replication of the longest gene is the optimal division.

Also, each virus has a genetic tree.

A genetic tree is a binary tree that satisfies the property: if gene B is a descendant node of gene A, then the length of gene A is more or equal length of gene B. When the gene is removed from a tree, the virus replaces it with the most recent gene in the hierarchy and performs the replacement operation. Substitution is the process of exchange in the tree of a short gene with a longer one, provided that the shorter gene is higher in the hierarchy. If the shorter gene is higher in the hierarchy, it is exchanged with the direct (neighboring) in the hierarchy, the longer gene. Replacement operations occur independently until the entire hierarchy satisfies the property of the genetic tree. When the gene is goind to be inserted, it is added as the last one in the hierarchy, and the replacement operation also occurs.

Your task is to emulate the behavior of the virus genotype. There are two requests for the addition of a gene and the removal of the top of the genetic tree. After each request, print a new vertex and the number of replacement operations performed during the operation.

Input

The first line of input contains an integer **N**, **P** - the number of requests and first element in the tree. Then **N** queries follow, where the addition operation is:

"$+A_i$", where A_i is the length of the i-th added gene.

"-" - delete operation.

Guaranteed that each added gene will be shorter in length than the previous $A_i > A_j$ for all $j > i$ as well as $P > A_i$. It is also guaranteed that at any given time the genetic tree will not be empty, and all the elements of the tree are unique.

Output

After each operation, a line of the form "C T" is displayed, where **C** is the number of substitution operations during the operation, **T** is the new top item in the hierarchy

Constraints

$1 \le \mathbf{N} \le 10^5$, $1 \le \mathbf{P} \le 10^9$, $1 \le \mathbf{A_i} \le 10^9$

Samples

Input (*stdin*)	Output (*stdout*)
5 43 + 23 + 15 - + 4 -	0 43 0 43 1 23 0 23 1 15

B. Replication - Part 2

Limits: 2 sec., 512 MiB

To increase their lifespan, viruses use corevaluation. The process by which a virus adds an arbitrary gene to its genotype. Your task is to emulate the behavior of the virus genotype in the absence of the ability to predict the added gene.

Input

The first line of input contains an integer **N**, **P** - the number of requests and first element in the tree. Then **N** queries follow, where the addition operation is:

'+A_i**'**, where A_i is the length of the i-th added gene.

'-' - delete operation.

Also guaranteed that at any given time the genetic tree will not be empty.

Output

After each operation, a line of the form "C T" is displayed, where C is the number of substitution operations during the operation, T is the new top item in the hierarchy

Constraints

$1 \leq \mathbf{N} \leq 10^7$, $1 \leq \mathbf{P} \leq 10^9$, $1 \leq \mathbf{A_i} \leq 10^9$

Samples

Input (*stdin*)	Output (*stdout*)
5 43 + 23 + 15 - + 4 -	0 43 0 43 1 23 0 23 1 15

C. Infection

Limits: 2 sec., 512 MiB

Viruses need carriers to successfully divide. The more infected surfaces, the more infected.

Each contaminated item is linked directly to another. However, at different distances. All surfaces can be represented as an acyclic graph, aka tree, where each surface is numbered with a non-negative integer. Each virus can move from one surface to another with V_i time, the virus cannot return to the infected surface due to possible transmutations. Also, viruses cannot co-exist on the same surface. There are **N** viruses of a single group, where ai is the lifetime of the i-th virus. Virus spread rate = sum of residual life on the last surface. Transition time - from surface to surface.

If it is not possible to find the final surface, the virus dies. Also virus can die moving from one surface to another if lifespan is over.

Input

The first line of input contains an integer **E**, **N**, **S**, where **E** is the number of surface connections, **N** is the number of viruses, **S** is the number the surface from which viruses start spreading. The initial surface cannot be after the surface.

Next is a line with **N** elements where A_i is the lifetime of the i-th virus. This is followed by **E** lines of the form "V_i T_i C_i", the surface with the number V_i is linked to the surface with the number T_i and the travel time C_i.

It is guaranteed that the graph is connected.

Output

Print a single number - the maximum possible spread of the virus.

Constraints

$1 \leq \mathbf{E} \leq 3 * 10^4$, $1 \leq \mathbf{N} \leq 10^5$, $\mathbf{S} \leq \mathbf{E}$, $1 \leq \mathbf{A_i} \leq 10^9$, $1 \leq \mathbf{C_i} \leq 10^9$

Samples

Input (*stdin*)	Output (*stdout*)
10 7 1 17 14 12 14 16 15 11 1 2 3 1 4 5 1 3 2 3 5 6 3 6 10 3 7 8 5 9 1 3 8 3 2 10 4 10 11 6	35

D. Coevolution

Limits: 2 sec., 512 MiB

After scientists studied the process of replication itself, they decided to reproduce it.

Each virus uses DNA-polymerase and DNA-ligase for replication. They tend to absorb each other and repeat themselves due to this. Let's take a look at the whole process. At the very beginning, the gene contains **N** polymerases and **M** ligases. In the entire genome, polymerase and ligase genes are in descending order in length. At the very beginning, in case of an overweight of one of the chains, it absorbs the other and returns to the structure, occupying the last non-decreasing position in it. In the case of the coincidence of the lengths of the ligase and polymerase chains, scientists believed that the absorption was accidental.

However, it has been observed that every odd time of coincidence, the polymerase absorbs lipase. This is enough for scientists. To estimate density, scientists calculate how far the chain has moved in the genome after being absorbed.

Input

The first line of the input file contains two integers separated by one space, these are numbers **N** and **M** respectively.

The second line contains **N** integers $\mathbf{A_i}$ in non-decreasing order, separated by single spaces - the length of the DNA polymerase in non-decreasing order.

The third line contains **M** integers $\mathbf{B_j}$ - the lengths of the DNA ligase in non-decreasing order.

Output

In the only line of the output file print one integer - the total density of the genome.

Constraints

$1 \leq \mathbf{N}, \mathbf{M} \leq 10^5$, $1 \leq \mathbf{A_i}, \mathbf{B_j} \leq 10^9$

Samples

Input (*stdin*)	Output (*stdout*)
3 2 1 2 5 2 4	5
2 2 3 7 3 6	3

E. Synthesis and struggle

Limits: 2 sec., 512 MiB

Finding and synthesizing a drug is extremely difficult.

To destroy the **RNA** of the virus, targeted modulation of the genome is used, and this procedure is multi-stage. A network of medical centers is used for complete synthesis. Medical transport is forced to move between centers. You can get from every center to every other in exactly one way, and the lengths of the roads are the same everywhere.

However, to speed up the search for a cure, it was decided to reform. It was decided to select a number of centers and designate them as **MDDSC** (Main Directorate of the Drug Search Center).

You have to spend a lot of money on servicing the MDDSC - **K** dalyars per quarter. It is believed that MDDSC always has the ability to produce synthesis. Each center, which is not a MDDSC, it was decided to appoint a MDDSC, which will be responsible for the synthesis in this center. In this case, cost to maintenance the med. center with appointed MDDSC is equal to Dl dollars per quarter, where l is the distance from med. Center to the corresponding MDDSC, measured in the number of roads that must be traveled.

Your task is to minimize the costs of renovation.

Input

The first line contains two integers **N**, **K**.

The second line contains **N - 1** integers D_i, numbered from one. Cost you need to maintain med. center with i roads to **MDDSC**.

The next **(N - 1)** lines contain pairs of numbers of cities connected by a road.

Output

In a line print the minimum amount of dollars spent on servicing for the quarter.

Constraints

$1 \leq \mathbf{N} \leq 100$, $1 \leq \mathbf{K} \leq 10^5$, $1 \leq \mathbf{V_i} \leq 10^5$

Samples

Input (*stdin*)	Output (*stdout*)
6 10 1 6 11 16 20 4 6 2 3 1 2 2 4 2 5	20

F. Interesting proteins

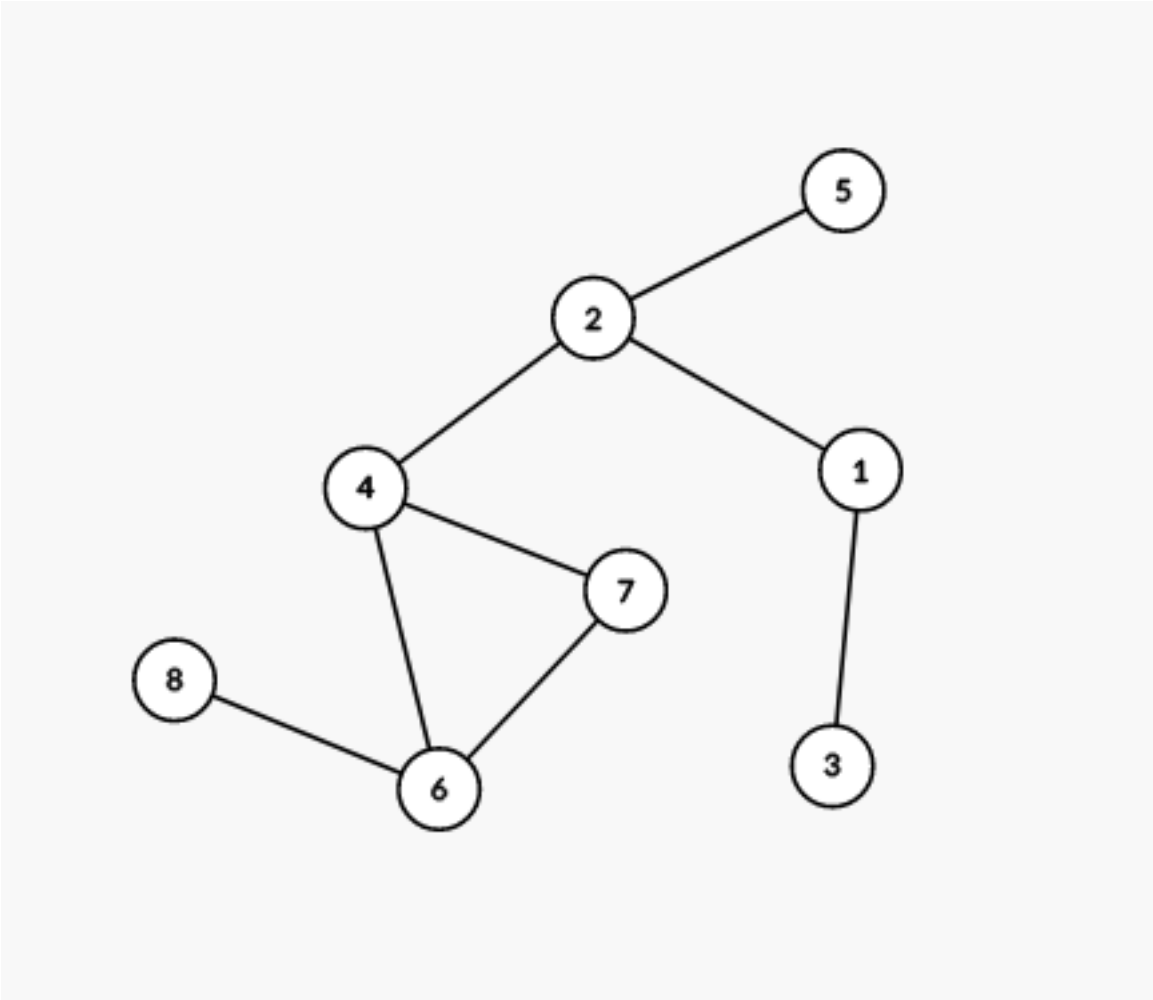
Limits: 2 sec., 512 MiB

Its a known fact that the most important component of human body, it is basically the building material of every cell in your body. They are usually divided into two groups: simple and complex. But few know that there is one more but not popular classification, that divides proteins into interesting and uninteresting one.

Proteins can be represented as a set of **N** nodes, where **M** pairs of nodes are connected with aminoacids. The proteins are always connected, that means there is at least one sequence of connections between any two distinct pairs of nodes.

A ring in a protein is a sequence of no less than 3 unique nodes, in which adjacent nodes connected with aminoacids and ends of the sequence connected too.

Protein is called interesting if any node lies lies on at most one ring. An example of interesing protein is shown in the picture below:



To help with better understanding of interesting proteins, you were asked to answer several queries – will the original protein be interesing after adding connection between u_i and v_i.

Input

The first line of input contains two integers **N** and **M**, where **N** is the number of nodes in the protein and **M** is the number of connections between nodes.

Next **M** lines contains description of connections. i^{th} connections is represented by two distinct integers u_i and v_i, which is nodes that it joins.

This is followed by line with a single integers **Q**, which is the number of queries.

Following **Q** lines describe queries. Each query contains two integers a_j and b_j – nodes that need to be connected to obtain graph for j^{th} query.

It is guaranteed that input represents connected protein.

Output

For each query output "Yes", if, when adding an connection between nodes a_j and b_j, the protein remains interesting, and "No" otherwise.

Constraints

$3 \leq \mathbf{N} \leq 2 * 10^5$, $N - 1 \leq \mathbf{M} \leq 2 * 10^5$, $0 \leq \mathbf{Q} \leq 2 * 10^5$, $1 \leq a_j, b_j \leq 2 * 10^5$, $1 \leq v_j, u_j \leq 2 * 10^5$

Samples

Input (*stdin*)	Output (*stdout*)
8 8 1 2 1 3 2 4 2 5 4 6 4 7 6 8 6 7 5 3 5 2 3 5 6 1 4 7 8	Yes Yes No No No

G. Ring decryptor

Limits: 2 sec., 512 MiB

As all know, most expensive and valuable equipment for experiments is stored behind a lock with a strong password. But what if the password has been lost? The answer is, of course, the Universal Device for Password Recoovery (UDPR). It allows to recover password for any lock.

However the device was present in a single copy and was lost a long time ago. Only the drawings of the device itself remains, but not the program for it. In order to avoid such occurences of password lost you was asked to write firmware for the device, that should conform to the Universal Password Recovery Interface (UPRI).

According to the documentation for the interface, a binary tree rooted at vertex 1 is fed to the input of the device with weight w_i on each vertex. Further, in order to acquire the password, the device should output answers to Q queries.

Queries are a pairs of vertices u_i and v_i. Answer to the query is the minimum sum of weights on the path between u_i and v_i which you can achieve after applying at most one rotation of the path beween any two vertices a and b which don't lie on the path from u_i to v_i.

In the documentation to the UDPR was a note that rotation of the path from a to b is called a proccess of right cyclic shift of vertices weights. Namely, w_{path_2} becomes equal to w_{path_1}, w_{path_3} changes to w_{path_2} and so on to w_{path_k}, and finally w_{path_k} goes to w_{path_1}, where k is length of the path. Also, $path_i$ is the i^{th} vertex on the path from vertex a to b.

Since there are too many queries and you need to handle them in fractions of a seconds, you were asked to help with writing the firmware to handle this queries quickly.

Input

The first line of the input contains one integer **N**, number of vertices in the tree

Next **N** lines describes vertices. i^{th} vertex is represented by three integers u_i, v_i and w_i, which are the left and the right childs of vertex i and its weight. If left or right child doesn't exist u_i or v_i will be equal to 0.

Following line contains single integer **Q**, which is the number of queries.

Next **Q** lines describes queries to the device. Each query contains two distinct integers a_j and b_j, vertices of the path for which you need to find minimum sum of weights after applying one path rotation.

It is guaranteed that input represents correct binary tree.

Output

For each query output single integer – minimum sum of weights that can be achieved by applying no more than one path rotation of the path which start and end vertices doesn't belong to the path from u_j to v_j for j^{th} query.

Constraints

$3 \le \mathbf{N} \le 5000$, $0 \le \mathbf{Q} \le 5000$, $0 \le v_j, u_j \le \mathbf{N}$, $0 \le w_j \le 10^9$, $1 \le a_j, b_j \le \mathbf{N}$,

Samples

Input (*stdin*)	Output (*stdout*)
10 2 3 10 7 6 4 4 5 1 0 0 9 0 0 7 8 9 7 0 0 1 10 0 6 0 0 0 0 0 5 3 7 10 5 9 1 4	19 23 17

H. Cyclic nucleotides

Limits: 2 sec., 512 MiB

The launch of well-known space probe "Traveler-3" gave scientists an enormous number of samples from several planets. During their research one of the scientists noticed that samples contains virus of alien origin, which has very odd kind DNA.

After analysis of alien DNA, it turned out that its structure is not linear, but resembles connected structure in which n nodes are clearly distinguished and some of them are connected with two types of nucleobases. During the study, scientists decided to hang DNA by the node 1 to simplify further experiments, but this revealed some previously undetected properties of DNA. One of them states that DNA not contains cycles.

To connect their research with already existed science of molecular biology scientists decided to name nucleotide path between two nodes, consisting of nodes connected in series by nucleobases. They also begin to name nucleotide "straight" if every subsequent node in nucleotide are located lower than previous one in the order of traversal from the starting node to end of nucleotide.

Further research discovered "cyclic" nucleotides. Nucleotide is named "cyclic" if nucleobases, which connects nodes in nucleotide in the order from its start to end, forms "cyclic" pattern. "Cyclic" pattern is called such sequence of DNA nucleobases consisting of positive number of same segments, each of which can be represented as k nucleobases of type 1 followed by k nucleobases of type 0, for some $k > 0$.

Your task is to help scientists with classification of DNAs. For this they asked you to find number of "straight" and "cyclic" nucleotides in given DNA.

Input

The first line contains single integer $\mathbf{N}$, number of nodes alien DNA.

Then there are $\mathbf{N} - 1$ lines describing connections between nodes in DNA. Line describing i^{th} connection contains three integers three integers u_i, v_i and c_i, which are starting and ending nodes of i^{th} connection and also nucleobase that links them.

It is guaranteed that input describes correct tree.

Output

Output one integer – number of "straight" "cyclic" nucleotides.

Constraints

$1 \leq \mathbf{N} \leq 10^5$, $1 \leq v_i, u_i \leq \mathbf{N}$, $0 \leq c_i \leq 1$

Samples

Input (*stdin*)	Output (*stdout*)
5 1 2 1 2 3 0 3 4 1 4 5 0	3

I. Cyclic nucleotides 2

Limits: 2 sec., 512 MiB

After the end of research of alien DNA brought by famous space probe "Traveler-3" it is not a secret that DNA structure of alien beings is not linear, it looks more like a tree.

This time scientists decides to recreate alien species using only samples of their DNA. This obviously takes years of research and practice. However, research center "Laboratory No. 1" knows how to radically speed up decoding alien DNA for upcoming experiment to recreate aliens. The plan is to automate tests with DNA under control of artificial intelligence (AI) and Automatic DNA Manipulation Station (ADMS). As an experiments developer, you were asked to write an emulator of ADMS device for test run under AI control.

As already mentioned, DNA consists of a set of **N** numbered nodes from 1 to **N**, some of which are connected by one of two nucleobase types. It is known that scientists has chosen one of the primitive species for first time, so the DNA's nodes forms a binary tree with a root in vertex 1. It means that:

- there is only one path between any two nodes;
- any node of DNA contains at most two nodes connected to it, which are located below.

Scientists already have been accustomed to calling nucleotide a path between two nodes, consisting of nodes connected in series by nucleobases. Also, a nucleotide is called "cyclic" if the nucleobase types, placed in order from its start to end, form a "cyclic" pattern. "Cyclic" pattern is called such sequence of DNA nucleobases consisting of positive number of same segments, each of which can be represented as k nucleobases of type 0 followed by k nucleobases of type 1, for some $k > 0$.

To satisfy the conditions of the experiments emulator should be able to process two types of queries:

- 1 u_j v_j – query to check if path from u_j to v_j "cyclic" nucleotide;
- 2 u_j v_j k_j – change of nucleobases lying on nucleotide from node u_j to v_j so that it becomes "cyclic" and line of each segment equal to $2 * k_j$.

For simplification of tests emulator always starts with empty DNA, which means before starting queries processing all nuclebases on existing connections are of type 0.

Help scientists with their research and write emulator for ADMS device.

Input

The first line contatins only one integer **N**, number of nodes in alien DNA.

Following **N** lines contains two integers each – l_i and r_i, which are left and right nodes connected to i^{th} node and located below it. If left or right adjacent nodes don't exists l_i or/and r_i will be equal to 0.

Next line contains one integer **Q**, number of queries to the device.

Following **Q** lines describes queries. j^{th} query can be in one of two formats:

- 1 u_j v_j – for descibe query of the first type;
- 2 u_j v_j k_j – for describe query of the second type.

It is guaranteed that input describes correct binary tree, and for queries of the second type $2 * k_j$ divides length of the path from u_j to v_j.

Output

For every query of the first type output "Yes", if nucleotide from u_j to v_j node is "cyclic", and "No" otherwise.

Constraints

$1 \leq \mathbf{N} \leq 3 * 10^4$, $0 \leq l_i, r_i \leq \mathbf{N}$,
$0 \leq u_i, v_i \leq \mathbf{N}$, $u_i \neq v_i$, $1 \leq k_i \leq \lfloor \frac{\mathbf{N}-1}{2} \rfloor$
$0 \leq Q \leq 3 * 10^4$

Samples

Input (*stdin*)	Output (*stdout*)
10 0 2 4 3 9 7 5 6 0 0 0 0 8 0 0 0 0 10 0 0 10 2 7 2 1 2 3 1 1 2 7 6 2 2 5 9 1 1 4 8 1 7 9 2 1 3 1 2 9 7 1 1 2 4 2 4 1 1	No Yes No

J. Centrifuge 2.0

Limits: 2 sec., 512 MiB

One day laboratory-assistant Petya during one of the usage of device "Centrifuge 2.0" for extracting DNA from the solution decided not to configure it before turning on, which was clearly described in the instruction. As the result settings of the main mechanism of the device was reset, so now the usage time has become huge, and one lauch can take weeks according to device estimates.

Reading through instruction Petya decided to adjust configuration of the main mechanism so that operating time of the centrifuge be as less as possible. The documentation says that mechanism is a binary tree consisting of **N** numbered nodes from 1 to **N**, each of which has attached numbers w_i – weights of the nodes. Petya recalled from the first year of university that a binary tree has the following properties:

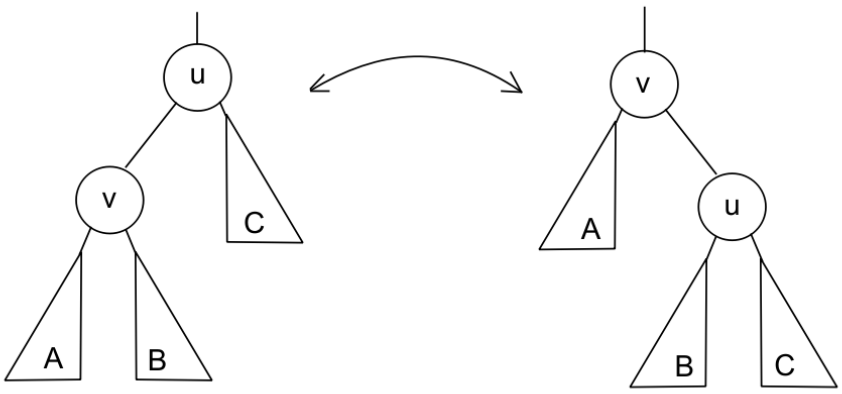
- between any two nodes there is a path consisting of connected in series nodes;
- each node has no ore than two child – left and right, linked by edges to the original node;
- there is an node 1, which is the root of the tree.

The instruction shows that operating time **T** of the centrifuge depends on the configuration of the main mechanism, namely, it is calculated by the formula:

$$T = w_1 * h_1 + w_2 * h_2 + ... + w_{\mathbf{N}} * h_{\mathbf{N}} = \sum_{n=1}^{\mathbf{N}} w_n * h_n$$

– where w_i is the weight of i^{th} vertex, h_i is number of vertices from root to node 1, counting ends of the path.

Configuration algorithm contains of several sequential rotations of the edges. Edge rotation for the edge $u - v$ to $v - u$ or vice versa depicted on the scheme below:



It can be seen seen from the figure that one of the vertices becomes higher than other, but the order of the Euler traversal of the entire tree does not change, i.e. order of subtrees A, B and C remains the same.

Help Petya fix the device by adjusting the main mechanism and performing a certain number of rotations so that the operating time is minimal.

Input

The first line contains integer **N**, number of nodes in the main mechanism.

Next **N** lines describes nodes of the mechanism. i^{th} node is described by three integers l_i, r_i and w_i – left and right childs of the i^{th} node, and its weight.

Its guaranteed that each tests describes correct binary tree.

Output

In the first line output one integer **Q** – number of the edge rotation, required to achieve minimal operating time of the device.

Next **Q** lines should contains two integers separated by spaces in any order v_i and u_i – vertices of the edge over which the rotation is performed, while an edge between these vertices must exist.

If there are several ways to configure mechanism, you may output any of them.

Constraints

$1 \leq \mathbf{N} \leq 500$, $0 \leq l_i, r_i \leq \mathbf{N}$, $0 \leq w_i \leq 10^9$
$0 \leq \mathbf{Q} \leq 3 * 10^5$

Samples

Input (*stdin*)	Output (*stdout*)
5 5 0 3 3 4 5 0 0 4 0 0 1 2 0 2	2 5 1 2 5