Codeforces Round #200 (Div. 1)

A. Rational Resistance

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

Mad scientist Mike is building a time machine in his spare time. To finish the work, he needs a resistor with a certain resistance value.

However, all Mike has is lots of identical resistors with unit resistance $R_0 = 1$. Elements with other resistance can be constructed from these resistors. In this problem, we will consider the following as elements:

1. one resistor;
2. an element and **one** resistor plugged in sequence;
3. an element and **one** resistor plugged in parallel.

With the consecutive connection the resistance of the new element equals $R = R_e + R_0$. With the parallel connection the resistance of the new element equals $R = \frac{1}{\frac{1}{R_e} + \frac{1}{R_0}}$. In this case R_e equals the resistance of the element being connected.

Mike needs to assemble an element with a resistance equal to the fraction $\frac{a}{b}$. Determine the smallest possible number of resistors he needs to make such an element.

Input

The single input line contains two space-separated integers a and b ($1 \le a, b \le 10^{18}$). It is guaranteed that the fraction $\frac{a}{b}$ is irreducible. It is guaranteed that a solution always exists.

Output

Print a single number — the answer to the problem.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is recommended to use the `cin`, `cout` streams or the `%I64d` specifier.

Examples

input
1 1
output
1

input
3 2
output
3

input
199 200
output
200

Note

In the first sample, one resistor is enough.

In the second sample one can connect the resistors in parallel, take the resulting element and connect it to a third resistor consecutively. Then, we get an element with resistance $\frac{1}{\frac{1}{1}+1} = \frac{3}{2}$. We cannot make this element using two resistors.

B. Alternating Current

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

Mad scientist Mike has just finished constructing a new device to search for extraterrestrial intelligence! He was in such a hurry to launch it for the first time that he plugged in the power wires without giving it a proper glance and started experimenting right away. After a while Mike observed that the wires ended up entangled and now have to be untangled again.

The device is powered by two wires "plus" and "minus". The wires run along the floor from the wall (on the left) to the device (on the right). Both the wall and the device have two contacts in them on the same level, into which the wires are plugged in some order. The wires are considered entangled if there are one or more places where one wire runs above the other one. For example, the picture below has four such places (top view):

Mike knows the sequence in which the wires run above each other. Mike also noticed that on the left side, the "plus" wire is always plugged into the top contact (as seen on the picture). He would like to untangle the wires without unplugging them and **without moving** the device. Determine if it is possible to do that. A wire can be freely moved and stretched on the floor, but cannot be cut.

To understand the problem better please read the notes to the test samples.

Input

The single line of the input contains a sequence of characters "+" and "-" of length n ($1 \le n \le 100000$). The i-th ($1 \le i \le n$) position of the sequence contains the character "+", if on the i-th step from the wall the "plus" wire runs above the "minus" wire, and the character "-" otherwise.

Output

Print either "Yes" (without the quotes) if the wires can be untangled or "No" (without the quotes) if the wires cannot be untangled.

Examples

input
-++-
output
Yes

input
+-
output
No

input
++
output
Yes

input
-
output
No

Note

The first testcase corresponds to the picture in the statement. To untangle the wires, one can first move the "plus" wire lower, thus eliminating the two crosses in the middle, and then draw it under the "minus" wire, eliminating also the remaining two crosses.

In the second testcase the "plus" wire makes one full revolution around the "minus" wire. Thus the wires cannot be untangled:

In the third testcase the "plus" wire simply runs above the "minus" wire twice in sequence. The wires can be untangled by lifting "plus" and moving it higher:

In the fourth testcase the "minus" wire runs above the "plus" wire once. The wires cannot be untangled without moving the device itself:

C. Read Time

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

Mad scientist Mike does not use slow hard disks. His modification of a hard drive has not one, but n different heads that can read data in parallel.

When viewed from the side, Mike's hard drive is an endless array of tracks. The tracks of the array are numbered from left to right with integers, starting with 1. In the initial state the i-th reading head is above the track number h_i. For each of the reading heads, the hard drive's firmware can move the head exactly one track to the right or to the left, or leave it on the current track. During the operation each head's movement does not affect the movement of the other heads: the heads can change their relative order; there can be multiple reading heads above any of the tracks. A track is considered *read* if at least one head has visited this track. In particular, all of the tracks numbered h_1, h_2, ..., h_n have been read at the beginning of the operation.

Mike needs to read the data on m distinct tracks with numbers p_1, p_2, ..., p_m. Determine the minimum time the hard drive firmware needs to move the heads and read all the given tracks. Note that an arbitrary number of other tracks can also be read.

Input

The first line of the input contains two space-separated integers n, m ($1 \le n, m \le 10^5$) — the number of disk heads and the number of tracks to read, accordingly. The second line contains n distinct integers h_i in ascending order ($1 \le h_i \le 10^{10}$, $h_i < h_{i+1}$) — the initial positions of the heads. The third line contains m distinct integers p_i in ascending order ($1 \le p_i \le 10^{10}$, $p_i < p_{i+1}$) - the numbers of tracks to read.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is recommended to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

Print a single number — the minimum time required, in seconds, to read all the needed tracks.

Examples

input
3 4 2 5 6 1 3 6 8
output
2

input
3 3 1 2 3 1 2 3
output
0

input
1 2 165 142 200
output
81

Note

The first test coincides with the figure. In this case the given tracks can be read in 2 seconds in the following way:

1. during the first second move the 1-st head to the left and let it stay there;
2. move the second head to the left twice;
3. move the third head to the right twice (note that the 6-th track has already been read at the beginning).

One cannot read the tracks in 1 second as the 3-rd head is at distance 2 from the 8-th track.

D. Water Tree

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Mad scientist Mike has constructed a rooted tree, which consists of n vertices. Each vertex is a reservoir which can be either empty or filled with water.

The vertices of the tree are numbered from 1 to n with the root at vertex 1. For each vertex, the reservoirs of its children are located below the reservoir of this vertex, and the vertex is connected with each of the children by a pipe through which water can flow downwards.

Mike wants to do the following operations with the tree:

1. Fill vertex v with water. Then v and all its children are filled with water.
2. Empty vertex v. Then v and all its ancestors are emptied.
3. Determine whether vertex v is filled with water at the moment.

Initially all vertices of the tree are empty.
Mike has already compiled a full list of operations that he wants to perform in order. Before experimenting with the tree Mike decided to run the list through a simulation. Help Mike determine what results will he get after performing all the operations.

Input

The first line of the input contains an integer n ($1 \le n \le 500000$) — the number of vertices in the tree. Each of the following $n - 1$ lines contains two space-separated numbers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$) — the edges of the tree.

The next line contains a number q ($1 \le q \le 500000$) — the number of operations to perform. Each of the following q lines contains two space-separated numbers c_i ($1 \le c_i \le 3$), v_i ($1 \le v_i \le n$), where c_i is the operation type (according to the numbering given in the statement), and v_i is the vertex on which the operation is performed.

It is guaranteed that the given graph is a tree.

Output

For each type 3 operation print 1 on a separate line if the vertex is full, and 0 if the vertex is empty. Print the answers to queries in the order in which the queries are given in the input.

Examples

input
5 1 2 5 1 2 3 4 2 12 1 1 2 3 3 1 3 2 3 3 3 4 1 2 2 4 3 1 3 3 3 4 3 5

output
0 0 0 1 0 1 0 1

E. Pumping Stations

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

Mad scientist Mike has applied for a job. His task is to manage a system of water pumping stations.

The system consists of n pumping stations, which are numbered by integers from 1 to n. Some pairs of stations are connected by bidirectional pipes through which water can flow in either direction (but only in one at a time). For each pipe you know its bandwidth — the maximum number of liters of water that can flow through it in one hour. Each pumping station can pump incoming water from some stations to other stations through the pipes, provided that in one hour the total influx of water to the station is equal to the total outflux of water from the station.

It is Mike's responsibility to pump water between stations. From station a to station b through the pipes (possibly through other stations) within one hour one can transmit a certain number of liters of water according to the rules described above. During this time, water from other stations can not flow into station a, and can not flow out of the station b. However, any amount of water can flow out of station a or in station b. If a total of x litres of water flows out of the station a in an hour, then Mike gets x bollars more to his salary.

To get paid, Mike needs to work for $n - 1$ days, according to the contract. On the first day he selects two stations v_1 and v_2, and within one hour he pumps a certain amount of water from v_1 to v_2. Next, on the i-th day Mike chooses a station v_{i+1} that has been never selected before, and pumps a certain amount of water out of the station v_i to station v_{i+1} for one hour. The quantity of water he pumps on the i-th day **does not depend** on the amount of water pumped on the $(i - 1)$-th day.

Mike needs to earn as much bollars as he can for his projects. Help Mike find such a permutation of station numbers $v_1, v_2, \ldots, v_n$ so Mike will be able to earn the highest possible salary.

Input

The first line of the input contains two space-separated integers n and m ($2 \le n \le 200, 1 \le m \le 1000$) — the number of stations and pipes in the system, accordingly. The i-th of the next m lines contains three space-separated integers a_i, b_i and c_i ($1 \le a_i, b_i \le n, a_i \ne b_i, 1 \le c_i \le 100$) — the numbers of stations connected by the i-th pipe and the pipe's bandwidth, accordingly. It is guaranteed that any two stations are connected by at most one pipe and that there is a pipe path between any two stations.

Output

On the first line print a single integer — the maximum salary Mike can earn.

On the second line print a space-separated permutation of n numbers from 1 to n — the numbers of stations in the sequence $v_1, v_2, \ldots, v_n$. If there are multiple answers, print any of them.

Examples

input
6 11 1 2 10 1 6 8 2 3 4 2 5 2 2 6 3 3 4 5 3 5 4 3 6 2 4 5 7 4 6 2 5 6 3
output
77 6 2 1 5 3 4