Problem A: Building Roads

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

After finishing your studies, you moved to the little known island kingdom of *Tsu-Nami*, whose infrastructure gets obliterated by a flood every couple of years. Every time this happens, the king orders his subjects to rebuild the network of roads between the island's villages by themselves and report to him whenever they finish a road. Not to waste too much of their time on this, he wants them to stop once any village can be reached from any other village via the new roads. This hasn't worked very well in the past though, since the mathematicians he ordered to oversee the process were having a hard time keeping track of the network's connectivity.

As he recently learned that you, a computer scientist, had moved to *Tsu-Nami*, he contacted you and ordered you to write a program that reads the reports of newly built roads and instantly informs the king when the desired connectivity is reached.

Input

The input consists of

- one line containing N ($2 \leq N \leq 10^5$) – the number of villages to be connected, and M ($1 \leq M \leq 10 \cdot N$), the number of roads that the king's subjects are *willing* to build following his initial order (cities and roads are numerated $1, ..., N$ and $1, ..., M$ respectively)
- M lines containing descriptions of the new roads in the order in which they are (or *would be*) finished, with the i-th road description consisting of two integers a_i and b_i ($1 \leq a_i, b_i \leq N$), describing that road i connects villages a_i and b_i (bidirectionally). Note that due to the lack of organization the subjects may occasionally build roads between a city and itself or between cities that were already connected by another road.

Output

If road i makes the last connections that were missing, output i. If there are still cities not connected by the M roads you are given, output "Build some more!".

Sample input and output

Input	Output
4 6 1 2 2 3 3 1 4 2 3 4 4 1	4
6 6 1 3 6 4 3 5 4 2 5 1 2 6	Build some more!

Problem B: Bacteria

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

An old friend of you is currently doing research on the real time evolution of a certain species of bacteria for his PhD. As bacteria are generally very fast evolving, he is having trouble to keep track of their phylogenetic relationships. Since he can't really code himself, he asked you to write him a little program that would help with that.

To make the task a little easier, he decided to numerate the newly evolving species himself, so that the program only needs to work with integer identifiers of species, not the cryptic names they would receive by the conventional naming procedure.

Initially there is only the species he starts with, s.t. at this point the phylogenetic tree consists only of its root, vertex 1. In the following, all its descending species are added to the tree.

The program should be able to process the following kinds of queries:

- `ADD` a b
  denoting that species b has evolved from species a, i.e. the program needs to suspend a vertex b on vertex a (it is guaranteed that a already exists in the tree).
- `GET` a b
  requesting the index of the latest evolved species of which both a and b descended

The species will be numerated from 1 to N.

Input

The input consists of

- one line containing K ($0 \leq K \leq 10^6$) – the amount of queries
- K lines each containing a query of one of the types described above.

Output

For every `GET` query output the correct response on a separate line.

Sample input and output

Input	Output
9 ADD 1 2 ADD 1 3 ADD 2 4 GET 1 3 GET 2 3 GET 3 4 ADD 2 5 GET 4 5 GET 5 5	1 1 1 2 5

Problem C: Contaminated Rivers

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You recently moved to a Chinese province with lots and lots of rivers in it. Unfortunately, even though numerous regulations against their pollution have been put in place in recent years, those are still largely being ignored by the industries they were targeting. As a result, most of the rivers are highly contaminated with toxic waste.

Of course this by itself causes lots of issues for the people living there. Nonetheless, one aspect of the pollution is particularly problematic: Some pollutants can drastically increase their harmfulness to humans when they come into contact with certain other pollutants. Naturally, when all ingredients for such a catastrophe are being disposed of into the same network of rivers, they eventually meet.

After some fatal cases of this phenomenon occurred in recent months, the local government has vowed to do a better job in monitoring the pollution and warning or even evacuating those parts of the population that – due to their proximity to the mixing spot – would be most affected by it. However, they still need some help to find out in which part of the river network given pollutants will mix. As you were already worried about getting your visa extended, you decided to offer your service to them.

Input

The input consists of

- one line containing N – the number of rivers ($2 \leq N \leq 10^5$)
- one line containing all connections between rivers, given via integers $a_2, ..., a_n$, with a_i being the index of the river that river i flows into ($1 \leq a_i \leq N$) and 1 being the index of the large stream everything ultimately flows into before it reaches the ocean
- several lines containing the queries: Each query begins with the number k of pollutants involved ($2 \leq k \leq 10$) in the anticipated reaction,

followed by k numbers $r_1, ..., r_k$, representing the indices of the rivers the pollutants are dumped into ($1 \leq r_i \leq N$). There will be between 1 and 10^5 such queries. The program is supposed to terminate when given a line containing only 0.

Output

For every non-terminating query, output the index of the river in which the given pollutants meet (assuming that each pollutant contaminates all rivers that carry it further downstream), in a separate line.

Sample input and output

Input	**Output**
12 1 1 2 2 3 3 4 5 5 5 8 2 7 8 3 12 9 10 2 4 12 5 9 10 11 10 10 3 6 6 6 0	1 2 4 5 6

Problem D: LCA minimum request

Advanced Algorithms for Programming Contests

Restrictions

Time: 5 seconds
Memory: 512 MB

Problem description

Given a rooted tree with N vertices, numerated from 0 to $N-1$ with 0 being the root. You are to respond to M requests of LCA for a pair of vertices.

The requests are generated as follows. Given a_1, a_2 and numbers x, y and z, the numbers $a_3, ..., a_{2M}$ generated as $a_i = (xa_{i-2} + ya_{i-1} + z) \mod N$. The first request is $lca(a_1, a_2)$. If the response of the $(i-1)$-th request is v, then the i-th request is $lca((a_{2i-1} + v) \mod N, a_{2i})$.

Input

The input consists of

- one line containing N and M ($1 \leq N \leq 10^5, 1 \leq M \leq 10^7$) – the numbers of vertices and requests
- one line containing $N-1$ integers, the i-th of which being the ancestor of vertex i
- one line containing the parameters a_1 and a_2 ($0 \leq a_1, a_2 \leq N-1$)
- one line containing the parameters x, y and z ($0 \leq x, y, z \leq 10^9$).

Output

Output the sum over the correct responses to all requests.

Sample input and output

Input	Output
3 2 0 1 2 1 1 1 0	2

Problem E: Mountain Rescue

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

The mountain rescue responsible for overseeing Mount Eret has a problem: As more and more people are climbing the legendary mountain, it is getting increasingly difficult for the rescue team to monitor all of their locations – but doing this is kind of necessary, as the mountain's surface is way too large to quickly find mountaineers just known to be somewhere on it in case of an emergency.

Luckily, they don't have to constantly track all mountaineers to be able to roughly determine their locations: It is known that the mountaineering routes on Mount Eret can be modeled by a tree, where the root models the peak, the internal vertices model the permanent bases used by the mountaineers, the leaves model the starting points of routes and the edges correspond available summits from starting point to base, from base to base, from base to peak or in some cases even from starting point to peak. Furthermore, each vertex has a known altitude and all routes from starting point to peak are strictly ascending (including the segments between adjacent vertices). Thus the rescue team only needs to know the starting point each mountaineer chooses for his summit and the highest ascent (height difference between start and end point) he is able to accomplish, in order to determine the route he will take (each mountaineer will start at a leaf, ascend towards the root as far as he can and then go back down on the same route). However, they are having some difficulties automating this analysis. Can you help them?

Input

The input consists of

- one line containing N ($5 \leq N \leq 10^5$) and r ($1 \leq r \leq N$) – the number of vertices in the tree model (which are numbered $1, ..., N$) and the index of the root

- one line containing N numbers $p_1, ..., p_N$ $(0 \leq p_i \leq N)$, where p_i is the index of the parent of vertex i for all vertices except the root and 0 for the root
- one line containing N numbers $a_1, ..., a_N$ $(0 \leq a_i \leq 10^9)$, where a_i is the altitude (above sea level) of the starting point / base station / peak modeled by vertex i
- one line containing M $(1 \leq M \leq 10^5)$ – the number of mountaineers
- M lines each containing numbers s_i and m_i $(1 \leq s_i \leq N, 0 \leq m_i \leq 10^9)$ – the starting point the respective mountaineer wants to use and the maximum ascent he is able to accomplish.

Output

For each mountaineer, output the index of the highest situated vertex he will reach. The rescue team will then figure out the rest.

Sample input and output

Input	**Output**
5 1 0 1 2 2 1 3980 2747 950 842 1231 5 3 1800 3 3030 5 2000 5 3000 4 1905	2 1 5 1 2

Problem F: Color Wizards

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

The magical country Dreamland consists entirely of cities, some of which are connected by both-way roads. Moreover, every city is reachable from every other one (directly or through other cities), while no road connects a city with itself and there is at most one road between two cities.

Among the countless magical creatures living in Dreamland there are two color wizards, a yellow one and a blue one. Whenever either of them travels through the country, the path it takes gets dyed in its respective color. It's well known that if the yellow and blue paths they leave behind overlap, this results in the common path becoming green, which is the color most detested by both wizards.

In a couple of days, for the first time ever, a conference of all the country's wizards will be held in its capital (one of the cities it consists of). Since they will have to start their respective journeys very soon in order to make it to the conference in time, the color wizards want to know at least how many roads will have to be green when both of them have arrived in the capital. As the traces they left with their previous journeys have faded away over time, currently all roads in the country are without any color.

Unfortunately, the color wizards are not so sure about which cities they are currently in, therefore the problem needs to be solved for a number of different cases representing their guesses about where they could be.

Input

The input consists of

- one line containing integers N and M ($1 \leq N \leq 10^5,\ 1 \leq M \leq 5 \cdot 10^5$) – the numbers of cities and roads in Dreamland
- one line containing a single integer c – the index of the country's capital ($1 \leq c \leq N$)

- M lines describing Dreamland's road system, each of them containing two integers a_i and b_i ($1 \leq a_i, b_i \leq N$), denoting a road between cities a_i and b_i
- one line containing a single integer K ($1 \leq K \leq 10^5$) – the number of joint guesses the two wizards have about their positions
- K lines, each containing one of the guesses, i.e. two integers in the range $1, ..., N$, representing the indices of the cities the wizards guess to be in.

Output

For each guess, output the minimum number of roads that would need to be left behind green by the wizards' journeys to the capital in case the guess was right.

Sample input and output

Input	**Output**
6 6 1 1 2 2 3 3 4 4 2 4 5 3 6 2 5 6 6 6	1 2