## Good Bye 2014

# A. New Year Transportation

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

New Year is coming in Line World! In this world, there are $n$ cells numbered by integers from $1$ to $n$, as a $1 \times n$ board. People live in cells. However, it was hard to move between distinct cells, because of the difficulty of escaping the cell. People wanted to meet people who live in other cells.

So, user tncks0121 has made a transportation system to move between these cells, to celebrate the New Year. First, he thought of $n - 1$ positive integers $a_1, a_2, \ldots, a_{n-1}$. For every integer $i$ where $1 \le i \le n - 1$ the condition $1 \le a_i \le n - i$ holds. Next, he made $n - 1$ portals, numbered by integers from 1 to $n - 1$. The $i$-th ($1 \le i \le n - 1$) portal connects cell $i$ and cell $(i + a_i)$, and one can travel from cell $i$ to cell $(i + a_i)$ using the $i$-th portal. Unfortunately, one cannot use the portal backwards, which means one cannot move from cell $(i + a_i)$ to cell $i$ using the $i$-th portal. It is easy to see that because of condition $1 \le a_i \le n - i$ one can't leave the Line World using portals.

Currently, I am standing at cell $1$, and I want to go to cell $t$. However, I don't know whether it is possible to go there. Please determine whether I can go to cell $t$ by only using the construted transportation system.

### Input

The first line contains two space-separated integers $n$ ($3 \le n \le 3 \times 10^4$) and $t$ ($2 \le t \le n$) — the number of cells, and the index of the cell which I want to go to.

The second line contains $n - 1$ space-separated integers $a_1, a_2, \ldots, a_{n-1}$ ($1 \le a_i \le n - i$). It is guaranteed, that using the given transportation system, one cannot leave the Line World.

### Output

If I can go to cell $t$ using the transportation system, print "YES". Otherwise, print "NO".

#### Examples

| input |
| --- |
| 8 4<br>1 2 1 2 1 2 1 |
| **output** |
| YES |

| input |
| --- |
| 8 5<br>1 2 1 2 1 1 1 |
| **output** |
| NO |

### Note

In the first sample, the visited cells are: $1, 2, 4$; so we can successfully visit the cell $4$.

In the second sample, the possible cells to visit are: $1, 2, 4, 6, 7, 8$; so we can't visit the cell $5$, which we want to visit.

# B. New Year Permutation

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

User ainta has a permutation $p_1, p_2, \ldots, p_n$. As the New Year is coming, he wants to make his permutation as pretty as possible.

Permutation $a_1, a_2, \ldots, a_n$ is *prettier* than permutation $b_1, b_2, \ldots, b_n$, if and only if there exists an integer $k$ ($1 \le k \le n$) where $a_1 = b_1, a_2 = b_2, \ldots, a_{k-1} = b_{k-1}$ and $a_k < b_k$ all holds.

As known, permutation $p$ is so sensitive that it could be only modified by swapping two distinct elements. But swapping two elements is harder than you think. Given an $n \times n$ binary matrix $A$, user ainta can swap the values of $p_i$ and $p_j$ ($1 \le i, j \le n, i \ne j$) if and only if $A_{i,j} = 1$.

Given the permutation $p$ and the matrix $A$, user ainta wants to know the prettiest permutation that he can obtain.

## Input

The first line contains an integer $n$ ($1 \le n \le 300$) — the size of the permutation $p$.

The second line contains $n$ space-separated integers $p_1, p_2, \ldots, p_n$ — the permutation $p$ that user ainta has. Each integer between $1$ and $n$ occurs exactly once in the given permutation.

Next $n$ lines describe the matrix $A$. The $i$-th line contains $n$ characters '0' or '1' and describes the $i$-th row of $A$. The $j$-th character of the $i$-th line $A_{i,j}$ is the element on the intersection of the $i$-th row and the $j$-th column of A. It is guaranteed that, for all integers $i, j$ where $1 \le i < j \le n$, $A_{i,j} = A_{j,i}$ holds. Also, for all integers $i$ where $1 \le i \le n$, $A_{i,i} = 0$ holds.

## Output

In the first and only line, print $n$ space-separated integers, describing the prettiest permutation that can be obtained.

### Examples

| input |
| --- |
| 7<br>5 2 4 3 6 7 1<br>0001001<br>0000000<br>0000010<br>1000001<br>0000000<br>0010000<br>1001000 |
| **output** |
| 1 2 4 3 6 7 5 |

| input |
| --- |
| 5<br>4 2 1 5 3<br>00100<br>00011<br>10010<br>01101<br>01010 |
| **output** |
| 1 2 3 4 5 |

## Note

In the first sample, the swap needed to obtain the prettiest permutation is: $(p_1, p_7)$.

In the second sample, the swaps needed to obtain the prettiest permutation is $(p_1, p_3), (p_4, p_5), (p_3, p_4)$.

A **permutation** $p$ is a sequence of integers $p_1, p_2, \ldots, p_n$, consisting of $n$ distinct positive integers, each of them doesn't exceed $n$. The $i$-th element of the permutation $p$ is denoted as $p_i$. The size of the permutation $p$ is denoted as $n$.

# C. New Year Book Reading

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

New Year is coming, and Jaehyun decided to read many books during 2015, unlike this year. He has $n$ books numbered by integers from 1 to $n$. The weight of the $i$-th ($1 \le i \le n$) book is $w_i$.

As Jaehyun's house is not large enough to have a bookshelf, he keeps the $n$ books by stacking them vertically. When he wants to read a certain book $x$, he follows the steps described below.

1. He *lifts* all the books above book $x$.
2. He pushes book $x$ out of the stack.
3. He puts down the lifted books without changing their order.
4. After reading book $x$, he puts book $x$ on the top of the stack.

He decided to read books for $m$ days. In the $j$-th ($1 \le j \le m$) day, he will read the book that is numbered with integer $b_j$ ($1 \le b_j \le n$). To read the book, he has to use the process described in the paragraph above. It is possible that he decides to re-read the same book several times.

After making this plan, he realized that the total weight of books he should *lift* during $m$ days would be too heavy. So, he decided to change the order of the stacked books before the New Year comes, and minimize the total weight. You may assume that books can be stacked in any possible order. Note that book that he is going to read on certain step isn't considered as *lifted* on that step. Can you help him?

## Input

The first line contains two space-separated integers $n$ ($2 \le n \le 500$) and $m$ ($1 \le m \le 1000$) — the number of books, and the number of days for which Jaehyun would read books.

The second line contains $n$ space-separated integers $w_1, w_2, \ldots, w_n$ ($1 \le w_i \le 100$) — the weight of each book.

The third line contains $m$ space separated integers $b_1, b_2, \ldots, b_m$ ($1 \le b_j \le n$) — the order of books that he would read. Note that he can read the same book more than once.

## Output

Print the minimum total weight of books he should *lift*, which can be achieved by rearranging the order of stacked books.

### Examples

| input |
| --- |
| 3 5<br>1 2 3<br>1 3 2 3 1 |
| **output** |
| 12 |

## Note

Here's a picture depicting the example. Each vertical column presents the stacked books.

# D. New Year Santa Network

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

New Year is coming in Tree World! In this world, as the name implies, there are $n$ cities connected by $n - 1$ roads, and for any two distinct cities there always exists a path between them. The cities are numbered by integers from 1 to $n$, and the roads are numbered by integers from $1$ to $n - 1$. Let's define $d(u, v)$ as total length of roads on the path between city $u$ and city $v$.

As an annual event, people in Tree World repairs exactly one road per year. As a result, the length of one road decreases. It is already known that in the $i$-th year, the length of the $r_i$-th road is going to become $w_i$, which is shorter than its length before. Assume that the current year is year $1$.

Three Santas are planning to give presents annually to all the children in Tree World. In order to do that, they need some preparation, so they are going to choose three distinct cities $c_1, c_2, c_3$ and make exactly one warehouse in each city. The $k$-th ($1 \le k \le 3$) Santa will take charge of the warehouse in city $c_k$.

It is really boring for the three Santas to keep a warehouse alone. So, they decided to build an only-for-Santa network! The cost needed to build this network equals to $d(c_1, c_2) + d(c_2, c_3) + d(c_3, c_1)$ dollars. Santas are too busy to find the best place, so they decided to choose $c_1, c_2, c_3$ randomly uniformly over all triples of distinct numbers from $1$ to $n$. Santas would like to know the expected value of the cost needed to build the network.

However, as mentioned, each year, the length of exactly one road decreases. So, the Santas want to calculate the expected after each length change. Help them to calculate the value.

## Input

The first line contains an integer $n$ ($3 \le n \le 10^5$) — the number of cities in Tree World.

Next $n - 1$ lines describe the roads. The $i$-th line of them ($1 \le i \le n - 1$) contains three space-separated integers $a_i$, $b_i$, $l_i$ ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $1 \le l_i \le 10^3$), denoting that the $i$-th road connects cities $a_i$ and $b_i$, and the length of $i$-th road is $l_i$.

The next line contains an integer $q$ ($1 \le q \le 10^5$) — the number of road length changes.

Next $q$ lines describe the length changes. The $j$-th line of them ($1 \le j \le q$) contains two space-separated integers $r_j$, $w_j$ ($1 \le r_j \le n - 1$, $1 \le w_j \le 10^3$). It means that in the $j$-th repair, the length of the $r_j$-th road becomes $w_j$. It is guaranteed that $w_j$ is smaller than the current length of the $r_j$-th road. The same road can be repaired several times.

## Output

Output $q$ numbers. For each given change, print a line containing the expected cost needed to build the network in Tree World. The answer will be considered correct if its absolute and relative error doesn't exceed $10^{-6}$.

### Examples

| input |
| --- |
| 3<br>2 3 5<br>1 3 3<br>5<br>1 4<br>2 2<br>1 2<br>2 1<br>1 1 |

| output |
| --- |
| 14.0000000000<br>12.0000000000<br>8.0000000000<br>6.0000000000<br>4.0000000000 |

| input |
| --- |
| 6<br>1 5 3<br>5 3 2<br>6 1 7<br>1 4 4<br>5 2 3<br>5<br>1 2<br>2 1<br>3 5<br>4 1<br>5 2 |

| output |
| --- |
| 19.6000000000<br>18.6000000000<br>16.6000000000<br>13.6000000000<br>12.6000000000 |

### Note

Consider the first sample. There are 6 triples: $(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)$. Because $n = 3$, the cost needed to build the network is always $d(1, 2) + d(2, 3) + d(3, 1)$ for all the triples. So, the expected cost equals to $d(1, 2) + d(2, 3) + d(3, 1)$.

# E. New Year Domino

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

Celebrating the new year, many people post videos of falling dominoes; Here's a list of them:
`https://www.youtube.com/results?search_query=New+Years+Dominos`

User ainta, who lives in a 2D world, is going to post a video as well.

There are $n$ dominoes on a 2D Cartesian plane. $i$-th domino ($1 \le i \le n$) can be represented as a line segment which is parallel to the $y$-axis and whose length is $l_i$. The lower point of the domino is on the $x$-axis. Let's denote the $x$-coordinate of the $i$-th domino as $p_i$. Dominoes are placed one after another, so $p_1 < p_2 < \ldots < p_{n-1} < p_n$ holds.

User ainta wants to take a video of falling dominoes. To make dominoes fall, he can push a single domino to the right. Then, the domino will fall down drawing a circle-shaped orbit until the line segment totally overlaps with the x-axis.

Also, if the $s$-th domino touches the $t$-th domino while falling down, the $t$-th domino will also fall down towards the right, following the same procedure above. Domino $s$ touches domino $t$ if and only if the segment representing $s$ and $t$ intersects.

See the picture above. If he pushes the leftmost domino to the right, it falls down, touching dominoes (A), (B) and (C). As a result, dominoes (A), (B), (C) will also fall towards the right. However, domino (D) won't be affected by pushing the leftmost domino, but eventually it will fall because it is touched by domino (C) for the first time.

The picture above is an example of falling dominoes. Each red circle denotes a touch of two dominoes.

User ainta has $q$ plans of posting the video. $j$-th of them starts with pushing the $x_j$-th domino, and lasts until the $y_j$-th domino falls. But sometimes, it could be impossible to achieve such plan, so he has to lengthen some dominoes. It costs one dollar to increase the length of a single domino by $1$. User ainta wants to know, for each plan, the minimum cost needed to achieve it. Plans are processed independently, i. e. if domino's length is increased in some plan, it doesn't affect its length in other plans. Set of dominos that will fall except $x_j$-th domino and $y_j$-th domino doesn't matter, but the initial push should be on domino $x_j$.

## Input

The first line contains an integer $n$ ($2 \le n \le 2 \times 10^5$)— the number of dominoes.

Next $n$ lines describe the dominoes. The $i$-th line ($1 \le i \le n$) contains two space-separated integers $p_i$, $l_i$ ($1 \le p_i, l_i \le 10^9$)— the $x$-coordinate and the length of the $i$-th domino. It is guaranteed that $p_1 < p_2 < \ldots < p_{n-1} < p_n$.

The next line contains an integer $q$ ($1 \le q \le 2 \times 10^5$) — the number of plans.

Next $q$ lines describe the plans. The $j$-th line ($1 \le j \le q$) contains two space-separated integers $x_j$, $y_j$ ($1 \le x_j < y_j \le n$). It means the $j$-th plan is, to push the $x_j$-th domino, and shoot a video until the $y_j$-th domino falls.

## Output

For each plan, print a line containing the minimum cost needed to achieve it. If no cost is needed, print 0.

### Examples

| input |
| --- |
| 6<br>1 5<br>3 3<br>4 4<br>9 2<br>10 1<br>12 1<br>4<br>1 2<br>2 4<br>2 5<br>2 6 |

| output |
| --- |
| 0<br>1<br>1<br>2 |

## Note

Consider the example. The dominoes are set like the picture below.

Let's take a look at the 4th plan. To make the 6th domino fall by pushing the 2nd domino, the length of the 3rd domino (whose x-

coordinate is 4) should be increased by 1, and the 5th domino (whose x-coordinate is 9) should be increased by 1 (other option is to increase 4th domino instead of 5th also by 1). Then, the dominoes will fall like in the picture below. Each cross denotes a touch between two dominoes.

# F. New Year Shopping

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

Dohyun is running a grocery store. He sells $n$ items numbered by integers from 1 to $n$. The $i$-th ($1 \le i \le n$) of them costs $c_i$ dollars, and if I buy it, my happiness increases by $h_i$. Each item can be displayed only for $p$ units of time because of freshness. As Dohyun displays the $i$-th item at time $t_i$, the customers can buy the $i$-th item only from time $t_i$ to time $t_i + (p - 1)$ inclusively. Also, each customer cannot buy the same item more than once.

I'd like to visit Dohyun's grocery store and buy some items for the New Year Party, and maximize my happiness. Because I am a really busy person, I can visit the store only once, and for very short period of time. In other words, if I visit the store at time $t$, I can only buy the items available at time $t$. But I can buy as many items as possible, if the budget holds. I can't buy same item several times due to store rules. It is not necessary to use the whole budget.

I made a list of $q$ pairs of integers $(a_j, b_j)$, which means I may visit the store at time $a_j$, and spend at most $b_j$ dollars at the store. For each pair, I'd like to know the maximum happiness I can obtain. But there are so many pairs that I can't handle them. Can you help me?

## Input

The first line contains two space-separated integers $n$ and $p$ ($1 \le n \le 4000$, $1 \le p \le 10\,000$) — the number of items, and the display time of each item.

Next $n$ lines describe the items. The $i$-th ($1 \le i \le n$) of them contains three space-separated integers $c_i$, $h_i$, $t_i$ ($1 \le c_i, h_i \le 4000$, $1 \le t_i \le 10\,000$) — the cost of the $i$-th item, the happiness of the $i$-th item, and the time when the $i$-th item starts to be displayed.

The next line contains an integer $q$ ($1 \le q \le 20\,000$)— the number of candidates.

Next $q$ lines describe the candidates. The $j$-th ($1 \le j \le q$) of them contains two space-separated integers $a_j$, $b_j$ ($1 \le a_j \le 20\,000$, $1 \le b_j \le 4000$) — the visit time and the budget for $j$-th visit of store.

## Output

For each candidate, print a single line containing the maximum happiness that I can obtain by buying some items.

### Examples

| input |
| --- |
| 4 4<br>2 3 2<br>3 5 1<br>4 7 2<br>11 15 5<br>4<br>1 3<br>2 5<br>2 6<br>5 14 |

| output |
| --- |
| 5<br>8<br>10<br>18 |

| input |
| --- |
| 5 4<br>3 2 1<br>7 4 4<br>2 1 2<br>6 3 5<br>3 2 2<br>10<br>1 5<br>2 5<br>4 8<br>4 9<br>4 10<br>5 8<br>5 9<br>5 10<br>8 4<br>7 9 |

| output |
| --- |
| 2<br>3 |

```
5
5
6
4
5
6
0
4
```

**Note**

Consider the first sample.

1. At time 1, only the 2nd item is available. I can buy the 2nd item using 3 dollars and my happiness will increase by 5.
2. At time 2, the 1st, 2nd, and 3rd item is available. I can buy the 1st item using 2 dollars, and the 2nd item using 3 dollars. My happiness will increase by 3 + 5 = 8.
3. At time 2, the 1st, 2nd, and 3rd item is available. I can buy the 1st item using 2 dollars, and the 3nd item using 4 dollars. My happiness will increase by 3 + 7 = 10.
4. At time 5, the 1st, 3rd, and 4th item is available. I can buy the 1st item using 2 dollars, and the 4th item using 11 dollars. My happiness will increase by 3 + 15 = 18. Note that I don't need to use the whole budget in this case.

# G. New Year Running

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

New Year is coming in Tree Island! In this island, as the name implies, there are $n$ cities connected by $n - 1$ roads, and for any two distinct cities there always exists exactly one path between them. For every person in Tree Island, it takes exactly one minute to pass by exactly one road.

There is a weird New Year tradition for runnners in Tree Island, which is called "extreme run". This tradition can be done as follows.

A runner chooses two distinct cities $a$ and $b$. For simplicity, let's denote the shortest path from city $a$ to city $b$ as $p_1, p_2, \ldots, p_l$ (here, $p_1 = a$ and $p_l = b$ holds). Then following happens:

1. The runner starts at city $a$.
2. The runner runs from city $a$ to $b$, following the shortest path from city $a$ to city $b$.
3. When the runner arrives at city $b$, he turns his direction immediately (it takes no time), and runs towards city $a$, following the shortest path from city $b$ to city $a$.
4. When the runner arrives at city $a$, he turns his direction immediately (it takes no time), and runs towards city $b$, following the shortest path from city $a$ to city $b$.
5. Repeat step 3 and step 4 forever.

In short, the course of the runner can be denoted as: $p_1 \to p_2 \to \cdots \to p_{l-1} \to p_l \to p_{l-1} \to p_{l-2} \to \cdots \to p_2 \to p_1 \to p_2 \to \cdots \to p_{l-1} \to p_l \to p_{l-1} \to p_{l-2} \to \cdots \to p_2 \to p_1 \to \cdots\cdots$

Two runners JH and JY decided to run "extremely" in order to celebrate the New Year. JH has chosen two cities $u$ and $v$, and JY has chosen two cities $x$ and $y$. They decided to start running at the same moment, and run until they meet at the same city for the first time. Meeting on a road doesn't matter for them. Before running, they want to know the amount of time they will run.

It is too hard for JH and JY to calculate this, so they ask you for help.

## Input

The first line contains a single positive integer $n$ ($5 \le n \le 2 \times 10^5$) — the number of cities in Tree Island.

Next $n - 1$ lines describe the roads of Tree Island. The $i$-th line ($1 \le i \le n - 1$) of them contains two space-separated integers $a_i$ and $b_i$ ($1 \le a_i, b_i \le n, a_i \ne b_i$) — the vertices connected by a single road of the tree.

The next line contains an integer $t$ ($1 \le t \le 2 \times 10^5$) — the number of test cases.

Next $t$ lines describes the test cases. The $j$-th line ($1 \le j \le t$) of them contains four space-separated integers $u_j, v_j, x_j, y_j$ ($1 \le u_j, v_j, x_j, y_j \le n, u_j \ne v_j, x_j \ne y_j$). It means that in this test case, JH has chosen two cities $u_j$ and $v_j$, JY has chosen two cities $x_j$ and $y_j$. JH starts running at city $u_j$, and JY starts running at city $x_j$.

## Output

For each test case, print an integer describing the amount of time they should run in minutes. If they have to run for an infinitely long time (in other words, if they never meet at the same city), print -1 instead. If they meet at the beginning of their run, print 0.

### Examples

| input |
| --- |
| 7<br>1 3<br>3 6<br>7 4<br>3 7<br>5 4<br>7 2<br>4<br>6 5 5 3<br>3 5 4 6<br>1 5 1 3<br>1 5 3 1 |
| **output** |
| 2<br>1<br>0<br>-1 |

## Note

The example looks like: