Problem A. Solar

Input:	stdin
Output:	stdout
Time Limit:	2 seconds
Memory Limit:	1024 MB

A country has N cities, numbered 1 to N, the capital is numbered 1. The electricity network in this country consists of $N-1$ transmission lines numbered from 1 to $N-1$. The i-th line connects city u_i with city v_i and has the length of w_i. The network ensures that electricity can be transferred between any pair of cities directly or through other immediate cities. According to initial estimation, the city u has the demand of a_u power units. The power loss on the lines is calculated by the formula $A = C \times d \times X$, where X is the amount of power units transferred, d is the distance transferred and C is a positive constant depends on the resistance of the line, in this problem, we consider $C = 1$.

The government is planning to build a solar power plant in one of the N cities. This power plant is expected to serve the needs of all N cities. Gennady is assigned this task of finding city u to build the power plant so that the total loss S is minimized where: $S = \sum_{v=1}^{N} a_v \times d(u, v)$ and $d(u, v)$ is the distance from city u to city v.

To find the optimal city to build the solar plant, Gennady is observing the demand in power consumption of N cities. There is a sequence of Q fluctuations, each is in 1 of 4 types:

- 1 u k: the demand of city u (a_u) is increased by k units;
- 2 u k: the demand of city u (a_u) is decreased by k units;
- 3 u k: With each city v in the area managed by u, the demand of city v (a_v) is increased by k units;
- 4 u k: With each city v in the area managed by u, the demand of city v (a_v) is decreased by k units.

We consider city v is in the area managed by u if the shortest distance from v to the capital (city 1) must go through u (city u is also in the area managed by itself).

After each fluctuation you need to find the optimal city to build the solar plant. In case there are multiple optimal answers, you can answer any of them.

Input

- The first line consists of 2 positive integers N and Q ($1 \le N \le 2 \times 10^5, 1 \le Q \le 2 \times 10^5$);
- The next $N-1$ lines describe $N-1$ electric lines by 3 positive integers u_i, v_i, w_i ($w_i \le 10^7$);
- The j-th line of the last Q lines contains information about j-th fluctuations consists of 3 positive integers t_j, u_j, k_j ($k_j \le 10^7$). The data ensures that the demand of all cities are always greater or equal to 0.

Output

For each fluctuations, print the optimal city in one line. If there are multiple optimal cities, you can print any of them.

Examples

stdin	stdout
5 3 1 2 1 1 4 2 1 5 1 2 3 3 1 1 3 1 1 1 1 2 1 2 3 4 2 2	1 2 1

Explanation

In the sample, after the first fluctuation, the demand of city 1 is 3. If we build the solar plant in city 1, the total loss will be:

$$a_1 \times d(1,1) + a_2 \times d(1,2) + a_3 \times d(1,3) + a_4 \times d(1,4) = 3 \times 0 + 1 \times 1 + 3 \times 4 + 1 \times 2 + 1 \times 1 = 16$$

If we build the solar plant in city 2 or city 3 or city 4 or city 5, the total loss will be: 17, 26, 30, 23, respectively. Thus, the optimal city is city 1.

After the second fluctuation, the demand of 5 cities are 3, 4, 3, 1, 1. So the total loss if we build the power plant will be at each city will be 19, 17, 35, 39, 29 respectively. Thus, the optimal city is city 2.

Subtask 1 (6 points)

$N \leq 100, Q \leq 5\,000$

Subtask 2 (10 points)

$N \leq 5\,000, Q \leq 5\,000$

Subtask 3 (12 points)

The electricity network forms a line

Subtask 4 (16 points)

Fluctuations contain only type 1

Subtask 5 (20 points)

Fluctuations contain only type 1 and 2

Subtask 6 (36 points)

No additional constraints

Problem B. Maximal Binary Search Tree

Input:	`stdin`
Output:	`stdout`
Time Limit:	3 seconds
Memory Limit:	1024 MB

A binary tree is a rooted tree where every vertex has at most two children. These children are called left child and right child. A binary search tree is a binary tree where each vertex has a comparable key and all vertices' keys satisfy all below conditions:

- For each vertex, all keys in its left sub-tree are strictly less than the vertex's key.
- For each vertex, all keys in its right sub-tree are strictly greater than the vertex's key.
- For each vertex, its left sub-tree and its right sub-tree are also valid binary search trees.

Given a rooted binary tree where each vertex is assigned a value, find the largest set of connected vertices in this tree which forms a valid binary search tree. Note that tree rotation is not allowed, meaning that the vertices' relations (left child, right child and parent) can not be changed. The tree contains n vertices which are numbered from 1 to n, inclusive. The root of the tree is the 1st vertex.

Input

- The first line contains a single integer θ ($1 \leq \theta \leq 6$) — the index of the subtask containing this test.
- The second line contains a single integer n ($1 \leq n \leq 2 \cdot 10^6$) — the number of vertices of the given tree.
- Among the last n lines, the i-th one contains three integers l_i, r_i and v_i ($0 \leq l_i, r_i \leq n, 1 \leq v_i \leq 10^9$), where l_i and r_i are the indices of the left child and the right child of the i-th vertex, respectively; and v_i is the value assigned to the i-th vertex. If the i-th vertex does not have a left child, $l_i = 0$. If the i-th vertex does not have a right child, $r_i = 0$.

It is guaranteed that the input represents a binary tree rooted at the 1st vertex.

Output

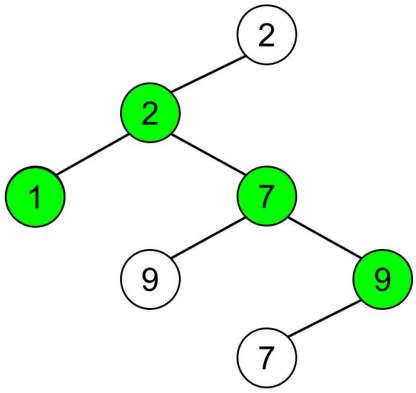
Print a single integer denoting the size of the maximum connected set of vertices which forms a valid binary search tree.

Examples

stdin	stdout
1 7 2 0 2 4 3 2 5 6 7 0 0 1 0 0 9 7 0 9 0 0 7	4

Explanation

The below figure demonstrates the above example, whose vertices in green represent the maximum connected set forming a binary search tree.



Subtask 1 (11 points)

$n \leq 20$

Subtask 2 (13 points)

It is guaranteed that:

- Every vertex which is the left child of some other vertex does not have a right child.
- Every vertex which is the right child of some other vertex does not have a left child.

Subtask 3 (17 points)

$n \leq 20\,000$ and the value assigned to every vertex does not exceed $1\,000$.

Subtask 4 (17 points)

$n \leq 2\,000$

Subtask 5 (19 points)

$n \leq 200\,000$

Subtask 6 (23 points)

No extra conditions.

Problem C. Half and half

Time Limit: 5 seconds
Memory Limit: 1024 MB

Gennady wants to play a game with you. He took all integers from 1 to n inclusive, shuffled them and then put all even numbers into array e and all odd numbers into array o. Your task is to find arrays e and o. You can ask Gennady questions of certain kind. Each question consists of two integers i and j. For each question Gennady says whether $e_i < o_j$ or not. You can ask at most 300 000 questions.

Interaction Protocol

First, the testing system writes two integers θ and n ($1 \le \theta \le 4, 1 \le n \le 10\,000$) — the index of the subtask containing this test and the number of integers Gennady used. Your solution shall print requests of two types:

- `compare i j` where $1 \le i \le \lfloor \frac{n}{2} \rfloor, 1 \le j \le \lceil \frac{n}{2} \rceil$. The testing system responds with the symbol "<" if $e_i < o_j$ or with the symbol ">" otherwise.
- `answer` $e_1\ e_2\ \cdots\ e_{\lfloor \frac{n}{2} \rfloor}\ o_1\ o_2\ \cdots\ o_{\lceil \frac{n}{2} \rceil}$ tells the values of e and o that your program has determined.

Don't forget to flush the output after each request!

Your solution must issue exactly one request of the second type, which must be the last request, and the solution must terminate gracefully after issuing it. Your solution is allowed to issue at most 300 000 requests of the first type.

For each test case the number n, the array e and the array o are fixed (the responses are **not** adaptive).

Examples

stdin	stdout
1 5 > > < > < <	compare 1 1 compare 1 2 compare 1 3 compare 2 1 compare 2 2 compare 2 3 answer 4 2 1 3 5

Subtask 1 (5 points)

$n \le 1\,000$

Subtask 2 (14 points)

It is guaranteed that at least one array is sorted in the increasing order. In other words, at least one of the two conditions hold: $e_1 < e_2 < \cdots < e_{\lfloor \frac{n}{2} \rfloor}$; $o_1 < o_2 < \cdots < o_{\lceil \frac{n}{2} \rceil}$

Subtask 3 (30 points)

$n \le 4\,000$

Subtask 4 (51 points)

No additional constraints