A Lake Camp

There is a camp by the lake of Ohrid, just near the hotel Mizo. The camp consists of N huts (houses) numbered with distinct integers from 1 to N. There are $N-1$ direct paths between $N-1$ pairs of huts. The network of paths is organised in such a way that you may travel between any two huts by traversing a sequence of (one or more) distinct direct paths. The distance between a pair of huts is defined as the number of direct paths in the sequence of direct paths between them. The huts reachable by direct path from a hut are called its neighbouring huts. Shortly, the huts are connected as a tree structure.

In each hut, one or more Wi-Fi devices may be installed. If a device with frequency f and range d is installed in some hut v, it will affect other huts that are at distance less or equal to d from v, at frequency f.

You are to determine the highest frequency that affects some hut v after installing some Wi-Fi devices in some of the huts.

Q operations will be performed. The operations can be of two types:

- **update** v f d - A device with frequency f is placed at hut v, which affects all huts at a distance less or equal to d from hut v.
- **query** v - Ask what the highest frequency is by which hut v is affected, i.e., there is a device that affects hut v with that frequency. If no device affects v, the answer should be -1.

Implementation Details

Include the header file `camp.h`.

Implement the following functions:

```
void init(int N, int Q, vector < vector < int > > A);
```

The `init` function is called once at the start of the program with the number of huts N, the number of operations Q, and a vector A containing the direct paths between pairs of huts such that for each i ($0 \le i < N-1$) there is a direct path between $A[i][0]$ and $A[i][1]$ ($1 \le A[i][j] \le N$).

```
void update(int v, int f, int d);
```

The `update` function is called every time there is an operation of type update with the parameters v, f, d whose meaning is described in the problem statement.

```
int query(int v);
```

The `query` function is called every time there is an operation of type query with the parameter v whose meaning is described in the problem statement. The function should return an answer to the query.

Constraints

- $3 \leq N \leq 10^5$
- $3 \leq Q \leq 10^5$
- $1 \leq v, d \leq N$
- $1 \leq f \leq 10^9$

Sample Grader

The sample grader has the following input format:

- **Line 1**: An integer N (the number of huts).
- **Line 2 + i** (for $0 \leq i < N - 1$): Two space-separated integers $A[i][0]$ and $A[i][1]$.
- **Line N + 1**: An integer Q (the number of operations to be performed).
- **Line N + 2 + i** (for $0 \leq i < Q$):
  - An integer $t[i]$ signifying the type of operation to be performed.
    - If $t[i]$ is 1, the operation is of type update and 3 integers follow (parameters used to call the `update` function): $v[i]$, $f[i]$, $d[i]$.
    - If $t[i]$ is 2, the operation is of type query and 1 integer follows (parameter used to call the `query` function): $v[i]$.

Subtasks

- **(7 pts)** $N \leq 1000$, $Q \leq 1000$.
- **(11 pts)** There is a pair of huts where the path between them contains all $N - 1$ direct paths, i.e. all the huts belong to one line of huts (the camp layout is only one long line of huts).
- **(17 pts)** If a hut has 3 or more neighboring huts, only two of those neighboring huts can have more than 1 neighbor.
- **(25 pts)** The camp contains exactly one hut with more than 2 neighbors.
- **(18 pts)** The distance between any two huts is at most 30.
- **(22 pts)** No other constraints.

Example

Call	Return value
init(9, 7, [[1, 2] [1, 3] [2, 4] [2, 5] [3, 6] [5, 7] [5, 8] [6, 9]])	
query(1)	-1
update(3, 5, 2)	
update(5, 2, 9)	
update(4, 8, 2)	
query(1)	8
query(6)	5
query(8)	2