JOI Tour

In IOI country, there are N towns which are numbered from 0 to $N-1$, and $N-1$ roads which are numbered from 0 to $N-2$. Road j ($0 \leq j \leq N-2$) connects town U_j and town V_j bidirectionally. You can move between any pair of towns by traversing some roads.

There is a restaurant in each town of IOI country. The type of restaurant at town i ($0 \leq i \leq N-1$) is represented by an integer F_i, which corresponds to:

- $F_i = 0$: Juice restaurant
- $F_i = 1$: Omelette restaurant
- $F_i = 2$: Ice cream restaurant

Rie is a tour guide in IOI country, who plans a sightseeing tour named **JOI Tour**. JOI Tour is a tour to visit 3 types of restaurants in a following way:

1. Choose a town i_0 with juice restaurant ($0 \leq i_0 \leq N-1$), and start the tour at town i_0.
2. Visit the juice restaurant at town i_0.
3. Choose a town i_1 with omelette restaurant ($0 \leq i_1 \leq N-1$), and move from town i_0 to town i_1 along the roads by bus, using the shortest route.
4. Visit the omelette restaurant at town i_1.
5. Choose a town i_2 with ice cream restaurant ($0 \leq i_2 \leq N-1$), and move from town i_1 to town i_2 along the roads by bus, using the shortest route.
6. Visit the ice cream restaurant at town i_2.
7. Finish the tour at town i_2.

To avoid customers getting bored, Rie decided to choose three towns i_0, i_1, i_2 so that they don't pass the same road twice. We call such JOI tour **good**. In order to help her finding the ideal tour plan, you are asked to compute the number of good JOI tours. In other words, you should find the number of tuples (i_0, i_1, i_2) which meets all of the following conditions:

- The restaurant at town i_0 is a juice restaurant.
- The restaurant at town i_1 is an omelette restaurant.
- The restaurant at town i_2 is an ice cream restaurant.
- When we move from town i_0 to town i_1 then from town i_1 to town i_2, both using the shortest routes, we don't pass the same road twice.

In IOI country, there will be Q events involving change of restaurant type. When the $(k+1)$-th event ($0 \leq k \leq$

$Q-1$) happens, two integers X_k and Y_k will be given to you, which holds $0 \leq X_k \leq N-1$ and $0 \leq Y_k \leq 2$. Then, the type of the restaurant at town X_k is changed to the type represented by integer Y_k. That is, when $Y_k = 0, 1, 2$, it is changed to juice, omelette, ice cream restaurant, respectively. After each event, you should immediately compute the up-to-date number of good JOI tours and tell the result to Rie.

Write a program which, given information of roads and restaurants, computes the number of good JOI tours after each event of change of restaurant type.

Implementation Details

You need to submit one file.

The file is `joitour.cpp`. The program should include `joitour.h` using the preprocessing directive `#include`.

In `joitour.cpp`, the following functions should be implemented.

- `void init(int N, std::vector<int> F, std::vector<int> U, std::vector<int> V, int Q)`
  - With this function call, the information of roads and restaurants is given.
  - This function is called only once, at the beginning.
  - The parameter `N` is the number of towns N.
  - The parameter `F` is an array of length N. `F[i]` ($0 \leq i \leq N-1$) represents the type of restaurant at town i, that is, F_i.
  - The parameters `U` and `V` are arrays of length $N-1$. `U[j]` and `V[j]` ($0 \leq j \leq N-2$) are two towns connected by road j, that is, U_j and V_j.
  - The parameter `Q` is the number of events of change of restaurant type, that is, Q.
- `void change(int X, int Y)`
  - With this function call, the information of change of restaurant type is given.
  - This function is called Q times.
  - In $(k+1)$-th call of this function ($0 \leq k \leq Q-1$), the parameter `X` is the index of town that change of restaurant type happens, that is, X_k.
  - In $(k+1)$-th call of this function ($0 \leq k \leq Q-1$), the parameter `Y` represents the new type of restaurant, that is, Y_k. It is guaranteed that new type is different from the old type.
- `long long num_tours()`
  - This function is called in the following occasions, totalling $Q+1$ times.
    - Right after the execution of function `init`
    - Right after the execution of function `change`
  - This function should return the up-to-date number of good JOI tours.

Important Notices

- Your program can implement other functions for internal use, or use global variables.
- Your program must not use the standard input and the standard output. Your program must not communicate with other files by any methods. However, your program may output debugging information to the standard error.

Compilation and Test Run

You can download an archive file from the contest webpage which contains the sample grader to test your program. The archive file also contains a sample source file of your program.

The sample grader is the file `grader.cpp`. In order to test your program, put `grader.cpp`, `joitour.cpp`, `joitour.h` in the same directory, and run the following command to compile your programs. Instead, you may run `compile.sh` contained in the archive file.

```
g++ -std=gnu++20 -O2 -o grader grader.cpp joitour.cpp
```

When the compilation succeeds, the executable file `grader` is generated.

Note that the actual grader is different from the sample grader. The sample grader will be executed as a single process, which will read input data from the standard input and write the results to the standard output.

Input for the Sample Grader

The sample grader reads the following data from the standard input.

N
$F_0\ F_1\ \cdots\ F_{N-1}$
$U_0\ V_0$
$U_1\ V_1$
$\vdots$
$U_{N-2}\ V_{N-2}$
Q
$X_0\ Y_0$
$X_1\ Y_1$
$\vdots$

X_{Q-1} Y_{Q-1}

Output of the Sample Grader

The sample grader outputs the return value of function `num_tours` in one line to the standard output, after every call of this function.

Constraints

- $3 \leqq N \leqq 200\,000$.
- $0 \leqq F_i \leqq 2$ $(0 \leqq i \leqq N-1)$.
- $0 \leqq U_j < V_j \leqq N-1$ $(0 \leqq j \leqq N-2)$.
- You can move between any pair of towns by traversing some roads.
- $0 \leqq Q \leqq 50\,000$.
- $0 \leqq X_k \leqq N-1$ $(0 \leqq k \leqq Q-1)$.
- $0 \leqq Y_k \leqq 2$ $(0 \leqq k \leqq Q-1)$.
- For each call to the function `change`, the new type is different from the old type.
- Given values are all integers.

Subtasks

1. (6 points) $N \leqq 400$, $Q \leqq 100$.
2. (8 points) $N \leqq 4\,000$, $Q \leqq 1\,000$.
3. (6 points) $Q = 0$.
4. (16 points) $U_j = j$, $V_j = j+1$ $(0 \leqq j \leqq N-2)$.
5. (16 points) $U_j = \left\lfloor \frac{j}{2} \right\rfloor$, $V_j = j+1$ $(0 \leqq j \leqq N-2)$. ($\left\lfloor \frac{j}{2} \right\rfloor$ denotes the largest integer not exceeding $\frac{j}{2}$.)
6. (34 points) $N \leqq 100\,000$, $Q \leqq 25\,000$.
7. (14 points) No additional constraints.

Sample Communication

Here is a sample input for the sample grader and corresponding function calls.

Sample Input 1	Function Calls	Return Values	Sample Output 1
3	`init(3, [0, 1, 2], [0, 1], [1, 2], 0)`		1
0 1 2	`num_tours()`	1	
0 1			
1 2			
0			

There are only one good JOI tour, which is represented by $(i_0, i_1, i_2) = (0, 1, 2)$. The following is the explanation that it meets the conditions of good JOI tours.

- Since $F_0 = 0$, the restaurant at town 0 is a juice restaurant.
- Since $F_1 = 1$, the restaurant at town 1 is an omelette restaurant.
- Since $F_2 = 2$, the restaurant at town 2 is an ice cream restaurant.
- When we move from town 0 to town 1 and then from town 1 to town 2, both using the shortest routes, we don't pass the same road twice.

Therefore, the return value in the first call of `num_tours` should be 1.

This sample input satisfies the constraints of Subtasks 1, 2, 3, 4, 6, 7.

Sample Input 2	Function Calls	Return Values	Sample Output 2
3	`init(3, [0, 1, 2], [0, 1], [1, 2], 2)`		1
0 1 2	`num_tours()`	1	0
0 1	`change(2, 0)`		1
1 2	`num_tours()`	0	
2	`change(0, 2)`		
2 0	`num_tours()`	1	
0 2			

Initially, there is one good JOI tour, which is represented by $(i_0, i_1, i_2) = (0, 1, 2)$. Thus, the return value in the first call of `num_tours` should be 1.

In the first event, the restaurant at town 2 is changed from ice cream restaurant to juice restaurant. After the change, ice cream restaurants disappear from IOI country, and there are no good JOI tours. Thus, the return value in the second call of `num_tours` should be 0.

In the second event, the restaurant at town 0 is changed from juice restaurant to ice cream restaurant. After the change, there is one good JOI tour, which is represented by $(i_0, i_1, i_2) = (2, 1, 0)$. Thus, the return value in the third call of `num_tours` should be 1.

This sample input satisfies the constraints of Subtasks 1, 2, 4, 6, 7.

Sample Input 3	Sample Output 3
7 1 0 2 2 0 1 0 0 1 0 2 1 3 1 4 2 5 2 6 7 0 0 1 1 2 0 3 0 4 2 5 2 6 2	3 0 4 4 0 4 5 5

This sample input satisfies the constraints of Subtasks 1, 2, 5, 6, 7.