Exercise 1: Distances in a Network with Exponential Edge Lengths

The Province of Exponential Distances is an interesting area of the country of Algorithms. Among other things, it is characterized by a very special road network. It consists of N cities that are connected by M two-way streets. The length of each road is a different power of 2, with the shortest road having a length of 2^0 km and the longest road having a length of 2^{M-1} km. Ever since you learned about the Provincial Road Network, you cannot stop thinking about it! You want to understand the structure of the shortest routes between cities, how they can be calculated efficiently, what is the easiest way to represent them, and how many edges will be used in total.

As a first step, you want to write a program that calculates the sum of the lengths of the shortest routes for all pairs of cities. Since the length of each path is a different force of 2, the length of each shortest path can normally be written in the binary system. So, it is better for your program to calculate the (possibly very large!) binary representation of the total length of the shortest routes for all pairs of cities.

Input Data: Your program will read from the standard input two positive integers N and M that correspond to the number of cities and the number of roads of the road network. The nodes of the network are numbered from 1 to N. In each of the following M lines, there will be three natural numbers α_e, b_e, c_e denoting that the path e = {α_e, b_e}, which connects the cities α_e and b_e, has a length of 2^{c_e}. The road network is undirected, will be connected and will not contain cycles or multiple edges. The c_e will be different from each other.

Output Data: Your program should output at standard output, in only one line, the binary representation of the total length of the smallest paths for all pairs of cities. Note that for large values of N and M, the result might have a length of several thousand binary bits. One way to represent the result is to follow a sequence of (x_0, c_0), …, (x_k, c_k), …, where the pair (x_k, c_k) declares that the edge with length 2^{c_k} is included x_k times in the shortest paths between the cities. Finally, your program should output the binary representation of the sum $\sum_k x_k 2^{c_k}$.

Restrictions

$3 \leq N \leq 10^5$

$N - 1 \leq M \leq 2 \cdot 10^5$

$c_e \in \{0,1,\dots,M-1\}$

Execution time limit: 1 sec.

Memory limit: 64MB.

Input Examples

```
5 6
1 3 5
4 5 0
2 1 3
3 2 1
4 3 4
4 2 2
```

Output Example

```
1000100
```

Exercise 2: Train Schedules

The pandemic has led many countries of the world to take restrictive measures, with a rather unexpected consequence to (also) the train schedules. Due of the measures, every train is forced to skip some stations of its route, without stopping at them. More specifically, a train is allowed to stop at a station s_j, only if its distance from a previous train station belongs to a set of permissible distances $\{d_1,\ldots,d_N\}$. There are 10^9 stations in total and the distance between each pair of consecutive stations is 1. Thus, a train can stop at some station s_j, only if its distance from the starting point can be expressed as a sum of the multiples of the permissible distances {d1,…,dN}. The trainline company has perceived the difficulties in travelling due to the measures and has made sure that there is at least one distance $d_k \leq 10^4$ in the total of permissible distances, so the trains can stop relatively often for the passengers to be served.
You are a student at the town with the station number 0 and you want to travel with the train at your hometown for the holidays. Since you know that it might not be achievable, you have started to work out alternatives (possibly with a combination of public transport). For that reason you want to be able to answer questions like "Can I reach the station s_j with the train if I begin from station 0?". You need to answer Q of these type of questions in total, so you prefer to automate the answering process, to not waste time.
Write a program that answers your questions, given the N permissible distances for the stations. You can assume that the first train station will be at station 0, where is also the start of your route.

Input Data: Your program will read from the standard input two positive integers N and Q, that correspond to the number of permissible distances and the number of Q questions that you want to answer. At the next line there will be N positive integers d_1, …, d_N that correspond to the permissible distances between two stations. Q lines will follow. At the j^{th} line, there will be a positive integer s_j that corresponds to the distance of the station regarding the j^{th} question from the starting point.

Output Data: The program should output at the standard output Q lines in total. The j^{th} line has to include the word "YES", if the train is allowed to stop at stop s_j or "NO" otherwise.

Restrictions
$1 \leq N \leq 10^3$
$1 \leq Q \leq 10^5$
$1 \leq d_i \leq 10^9$
$1 \leq min_i\{d_i\} \leq 10^3$
$1 \leq s_j \leq 10^9$
Execution time limit: 1 sec.
Memory limit: 64MB.

Input Example

```
3 3
6 7 5
5
8
13
```

Output Example

```
YES
NO
YES
```