VK Cup 2015 - Finals

A. Logistical Questions

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

Some country consists of n cities, connected by a railroad network. The transport communication of the country is so advanced that the network consists of a minimum required number of $(n - 1)$ bidirectional roads (in the other words, the graph of roads is a tree). The i-th road that directly connects cities a_i and b_i, has the length of l_i kilometers.

The transport network is served by a state transporting company FRR (Fabulous Rail Roads). In order to simplify the price policy, it offers a single ride fare on the train. In order to follow the route of length t kilometers, you need to pay $t^{\frac{3}{2}}$ burles. Note that it is forbidden to split a long route into short segments and pay them separately (a special railroad police, or RRP, controls that the law doesn't get violated).

A Large Software Company decided to organize a programming tournament. Having conducted several online rounds, the company employees determined a list of finalists and sent it to the logistical department to find a place where to conduct finals. The Large Software Company can easily organize the tournament finals in any of the n cities of the country, so the the main factor in choosing the city for the last stage of the tournament is the total cost of buying tickets for all the finalists. We know that the i-th city of the country has w_i cup finalists living there.

Help the company employees find the city such that the total cost of travel of all the participants to it is minimum.

Input

The first line of the input contains number n $(1 \le n \le 200\,000)$ — the number of cities in the country.

The next line contains n integers $w_1, w_2, ..., w_n$ $(0 \le w_i \le 10^8)$ — the number of finalists living in each city of the country.

Next $(n - 1)$ lines contain the descriptions of the railroad, the i-th line contains three integers, a_i, b_i, l_i $(1 \le a_i, b_i \le n, a_i \ne b_i, 1 \le l_i \le 1000)$.

Output

Print two numbers — an integer f that is the number of the optimal city to conduct the competition, and the real number c, equal to the minimum total cost of transporting all the finalists to the competition. Your answer will be considered correct if two conditions are fulfilled at the same time:

1. The absolute or relative error of the printed number c in comparison with the cost of setting up a final in city f doesn't exceed 10^{-6};
2. Absolute or relative error of the printed number c in comparison to the answer of the jury doesn't exceed 10^{-6}.

If there are multiple answers, you are allowed to print any of them.

Sample test(s)

input
5 3 1 2 6 5 1 2 3 2 3 1 4 3 9 5 3 1
output
3 192.0

input
2 5 5 1 2 2
output
1 14.142135623730951000

Note

In the sample test an optimal variant of choosing a city to conduct the finals of the competition is 3. At such choice the cost of conducting is $3 \cdot 4^{\frac{3}{2}} + 1 \cdot 1^{\frac{3}{2}} + 2 \cdot 0^{\frac{3}{2}} + 6 \cdot 9^{\frac{3}{2}} + 5 \cdot 1^{\frac{3}{2}} = 192$ burles.

In the second sample test, whatever city you would choose, you will need to pay for the transport for five participants, so you will need to pay $2\sqrt{2}$ burles for each one of them.

B. Clique in the Divisibility Graph

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

As you must know, the maximum clique problem in an arbitrary graph is NP-hard. Nevertheless, for some graphs of specific kinds it can be solved effectively.

Just in case, let us remind you that a clique in a non-directed graph is a subset of the vertices of a graph, such that any two vertices of this subset are connected by an edge. In particular, an empty set of vertexes and a set consisting of a single vertex, are cliques.

Let's define a divisibility graph for a set of positive integers $A = \{a_1, a_2, ..., a_n\}$ as follows. The vertices of the given graph are numbers from set A, and two numbers a_i and a_j $(i \neq j)$ are connected by an edge if and only if either a_i is divisible by a_j, or a_j is divisible by a_i.

You are given a set of non-negative integers A. Determine the size of a maximum clique in a divisibility graph for set A.

Input

The first line contains integer n $(1 \leq n \leq 10^6)$, that sets the size of set A.

The second line contains n distinct positive integers $a_1, a_2, ..., a_n$ $(1 \leq a_i \leq 10^6)$ — elements of subset A. The numbers in the line follow in the ascending order.

Output

Print a single number — the maximum size of a clique in a divisibility graph for set A.

Sample test(s)

input
8 3 4 6 8 10 18 21 24
output
3

Note

In the first sample test a clique of size 3 is, for example, a subset of vertexes $\{3, 6, 18\}$. A clique of a larger size doesn't exist in this graph.

C. Restoring Map

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

Archaeologists found some information about an ancient land of Treeland. We know for sure that the Treeland consisted of n cities connected by the n - 1 road, such that you can get from each city to any other one along the roads. However, the information about the specific design of roads in Treeland has been lost. The only thing that the archaeologists can use is the preserved information about *near* cities.

Two cities of Treeland were called *near*, if it were possible to move from one city to the other one by moving through at most two roads. Also, a city is considered *near* to itself. During the recent excavations archaeologists found a set of n notes, each of them represents a list of cities, *near* to some of the n cities of the country. However, unfortunately, none of the found records lets you understand in what order the cities go in the list and for which city in the list the *near* to it cities were listed.

Help the archaeologists and restore any variant of the map of Treeland that meets the found information.

Input

The first line contains integer n $(2 \le n \le 1000)$ — the number of cities in the country.

Next n lines describe the found lists of *near* cities. Each list starts from number k $(1 \le k \le n)$, representing the number of cities in the list followed by k city numbers. All numbers in each list are distinct.

It is guaranteed that the given information determines at least one possible road map.

Output

Print n - 1 pairs of numbers representing the roads of the country. The i-th line must contain two integers a_i, b_i $(1 \le a_i, b_i \le n, a_i \ne b_i)$, showing that there is a road between cities a_i and b_i.

The answer you print must satisfy the description of close cities from the input. You may print the roads of the countries in any order. The cities that are connected by a road may also be printed in any order.

If there are multiple good answers, you may print any of them.

Sample test(s)

input

```
5
4 3 2 4 1
5 5 3 2 4 1
5 4 2 1 5 3
4 2 1 4 3
3 1 4 5
```

output

```
1 4
1 2
1 3
4 5
```

input

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

output

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

D. Restructuring Company

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

Even the most successful company can go through a crisis period when you have to make a hard decision — to restructure, discard and merge departments, fire employees and do other unpleasant stuff. Let's consider the following model of a company.

There are n people working for the Large Software Company. Each person belongs to some *department*. Initially, each person works on his own project in his own department (thus, each company initially consists of n departments, one person in each).

However, harsh times have come to the company and the management had to hire a crisis manager who would rebuild the working process in order to boost efficiency. Let's use $team(person)$ to represent a team where person $person$ works. A crisis manager can make decisions of two types:

1. Merge departments $team(x)$ and $team(y)$ into one large department containing all the employees of $team(x)$ and $team(y)$, where x and y $(1 \le x, y \le n)$ — are numbers of two of some company employees. If $team(x)$ matches $team(y)$, then nothing happens.
2. Merge departments $team(x), team(x+1), ..., team(y)$, where x and y $(1 \le x \le y \le n)$ — the numbers of some two employees of the company.

At that the crisis manager can sometimes wonder whether employees x and y $(1 \le x, y \le n)$ work at the same department.

Help the crisis manager and answer all of his queries.

Input

The first line of the input contains two integers n and q $(1 \le n \le 200\,000, 1 \le q \le 500\,000)$ — the number of the employees of the company and the number of queries the crisis manager has.

Next q lines contain the queries of the crisis manager. Each query looks like $type\ x\ y$, where $type \in \{1, 2, 3\}$. If $type = 1$ or $type = 2$, then the query represents the decision of a crisis manager about merging departments of the first and second types respectively. If $type = 3$, then your task is to determine whether employees x and y work at the same department. Note that x can be equal to y in the query of any type.

Output

For each question of type 3 print "YES" or "NO" (without the quotes), depending on whether the corresponding people work in the same department.

Sample test(s)

input
8 6 3 2 5 1 2 5 3 2 5 2 4 7 2 1 2 3 1 7

output
NO YES YES

E. Max and Min

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

Two kittens, Max and Min, play with a pair of non-negative integers x and y. As you can guess from their names, kitten Max loves to maximize and kitten Min loves to minimize. As part of this game Min wants to make sure that both numbers, x and y became negative at the same time, and kitten Max tries to prevent him from doing so.

Each kitten has a set of pairs of integers available to it. Kitten Max has n pairs of non-negative integers (a_i, b_i) $(1 \le i \le n)$, and kitten Min has m pairs of non-negative integers (c_j, d_j) $(1 \le j \le m)$. As kitten Max makes a move, it can take any available pair (a_i, b_i) and add a_i to x and b_i to y, and kitten Min can take any available pair (c_j, d_j) and subtract c_j from x and d_j from y. Each kitten can use each pair multiple times during distinct moves.

Max moves first. Kitten Min is winning if at some moment both numbers a, b are negative **simultaneously**. Otherwise, the winner of the game is kitten Max. Determine which kitten wins if both of them play optimally.

Input

The first line contains two integers, n and m $(1 \le n, m \le 100\,000)$ — the number of pairs of numbers available to Max and Min, correspondingly.

The second line contains two integers x, y $(1 \le x, y \le 10^9)$ — the initial values of numbers with which the kittens are playing.

Next n lines contain the pairs of numbers a_i, b_i $(1 \le a_i, b_i \le 10^9)$ — the pairs available to Max.

The last m lines contain pairs of numbers c_j, d_j $(1 \le c_j, d_j \le 10^9)$ — the pairs available to Min.

Output

Print «`Max`» (without the quotes), if kitten Max wins, or "`Min`" (without the quotes), if kitten Min wins.

Sample test(s)

input
2 2 42 43 2 3 3 2 3 10 10 3
output
Min

input
1 1 1 1 3 4 1 1
output
Max

Note

In the first test from the statement Min can respond to move $(2, 3)$ by move $(3, 10)$, and to move $(3, 2)$ by move $(10, 3)$. Thus, for each pair of Max and Min's moves the values of both numbers x and y will strictly decrease, ergo, Min will win sooner or later.

In the second sample test after each pair of Max and Min's moves both numbers x and y only increase, thus none of them will become negative.

F. Matching Names

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

Teachers of one programming summer school decided to make a surprise for the students by giving them names in the style of the "Hobbit" movie. Each student must get a pseudonym maximally similar to his own name. The pseudonym must be a name of some character of the popular saga and now the teachers are busy matching pseudonyms to student names.

There are n students in a summer school. Teachers chose exactly n pseudonyms for them. Each student must get exactly one pseudonym corresponding to him. Let us determine the relevance of a pseudonym b to a student with name a as the length of the largest common prefix a and b. We will represent such value as $\mathrm{lcp}(a, b)$. Then we can determine the *quality* of matching of the pseudonyms to students as a sum of relevances of all pseudonyms to the corresponding students.

Find the matching between students and pseudonyms with the maximum *quality*.

Input

The first line contains number n ($1 \le n \le 100\,000$) — the number of students in the summer school.

Next n lines contain the name of the students. Each name is a non-empty word consisting of lowercase English letters. Some names can be repeating.

The last n lines contain the given pseudonyms. Each pseudonym is a non-empty word consisting of small English letters. Some pseudonyms can be repeating.

The total length of all the names and pseudonyms doesn't exceed $800\,000$ characters.

Output

In the first line print the maximum possible *quality* of matching pseudonyms to students.

In the next n lines describe the optimal matching. Each line must have the form $a\ b$ ($1 \le a, b \le n$), that means that the student who was number a in the input, must match to the pseudonym number b in the input.

The matching should be a one-to-one correspondence, that is, each student and each pseudonym should occur exactly once in your output. If there are several optimal answers, output any.

Sample test(s)

input
5 gennady galya boris bill toshik bilbo torin gendalf smaug galadriel
output
11 4 1 2 5 1 3 5 2 3 4

Note

The first test from the statement the match looks as follows:

- **bil**l ⟶ **bil**bo (lcp = 3)
- **gal**ya ⟶ **gal**adriel (lcp = 3)
- **gen**nady ⟶ **gen**dalf (lcp = 3)
- **to**shik ⟶ **to**rin (lcp = 2)
- boris ⟶ smaug (lcp = 0)

G. Replicating Processes

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

A Large Software Company develops its own social network. Analysts have found that during the holidays, major sporting events and other significant events users begin to enter the network more frequently, resulting in great load increase on the infrastructure.

As part of this task, we assume that the social network is $4n$ processes running on the n servers. All servers are absolutely identical machines, each of which has a volume of RAM of 1 GB = 1024 MB [1]. Each process takes 100 MB of RAM on the server. At the same time, the needs of maintaining the viability of the server takes about 100 more megabytes of RAM. Thus, each server may have up to 9 different processes of social network.

Now each of the n servers is running exactly 4 processes. However, at the moment of peak load it is sometimes necessary to replicate the existing $4n$ processes by creating $8n$ new processes instead of the old ones. More formally, there is a set of replication rules, the i-th ($1 \le i \le 4n$) of which has the form of $a_i \rightarrow (b_i, c_i)$, where a_i, b_i and c_i ($1 \le a_i, b_i, c_i \le n$) are the numbers of servers. This means that instead of an old process running on server a_i, there should appear two new copies of the process running on servers b_i and c_i. The two new replicated processes can be on the same server (i.e., b_i may be equal to c_i) or even on the same server where the original process was (i.e. a_i may be equal to b_i or c_i). During the implementation of the rule $a_i \rightarrow (b_i, c_i)$ first the process from the server a_i is destroyed, then appears a process on the server b_i, then appears a process on the server c_i.

There is a set of $4n$ rules, destroying all the original $4n$ processes from n servers, and creating after their application $8n$ replicated processes, besides, on each of the n servers will be exactly 8 processes. However, the rules can only be applied consecutively, and therefore the amount of RAM of the servers imposes limitations on the procedure for the application of the rules.

According to this set of rules determine the order in which you want to apply all the $4n$ rules so that at any given time the memory of each of the servers contained at most 9 processes (old and new together), or tell that it is impossible.

Input

The first line of the input contains integer n ($1 \le n \le 30\,000$) — the number of servers of the social network.

Next $4n$ lines contain the rules of replicating processes, the i-th ($1 \le i \le 4n$) of these lines as form a_i, b_i, c_i ($1 \le a_i, b_i, c_i \le n$) and describes rule $a_i \rightarrow (b_i, c_i)$.

It is guaranteed that each number of a server from 1 to n occurs four times in the set of all a_i, and eight times among a set that unites all b_i and c_i.

Output

If the required order of performing rules does not exist, print "NO" (without the quotes).

Otherwise, print in the first line "YES" (without the quotes), and in the second line — a sequence of $4n$ numbers from 1 to $4n$, giving the numbers of the rules in the order they are applied. The sequence should be a permutation, that is, include each number from 1 to $4n$ exactly once.

If there are multiple possible variants, you are allowed to print any of them.

Sample test(s)

input

```
2
1 2 2
1 2 2
1 2 2
1 2 2
2 1 1
2 1 1
2 1 1
2 1 1
```

output

```
YES
1 2 5 6 3 7 4 8
```

input

```
3
1 2 3
1 1 1
1 1 1
1 1 1
2 1 3
2 2 2
2 2 2
2 2 2
3 1 2
3 3 3
3 3 3
3 3 3
```

output

```
YES
2 3 4 6 7 8 10 11 12 1 5 9
```

Note

(1) To be extremely accurate, we should note that the amount of server memory is 1 GiB = 1024 MiB and processes require 100 MiB RAM where a gibibyte (GiB) is the amount of RAM of 2^{30} bytes and a mebibyte (MiB) is the amount of RAM of 2^{20} bytes.

In the first sample test the network uses two servers, each of which initially has four launched processes. In accordance with the rules of replication, each of the processes must be destroyed and twice run on another server. One of the possible answers is given in the statement: after applying rules 1 and 2 the first server will have 2 old running processes, and the second server will have 8 (4 old and 4 new) processes. After we apply rules 5 and 6, both servers will have 6 running processes (2 old and 4 new). After we apply rules 3 and 7, both servers will have 7 running processes (1 old and 6 new), and after we apply rules 4 and 8, each server will have 8 running processes. At no time the number of processes on a single server exceeds 9.

In the second sample test the network uses three servers. On each server, three processes are replicated into two processes on the same server, and the fourth one is replicated in one process for each of the two remaining servers. As a result of applying rules $2, 3, 4, 6, 7, 8, 10, 11, 12$ each server would have 7 processes (6 old and 1 new), as a result of applying rules $1, 5, 9$ each server will have 8 processes. At no time the number of processes on a single server exceeds 9.