Task: ARI
Arithmetic progressions

Maratona de Verão, Day 1. Source file `ari.*` Available memory: 256 MB. *20.01.2014*

Young Andrew loves arithmetic progressions[*]. His even younger sister Anya has written a sequence of numbers on a blackboard. Now Andrew wants to split this sequence into several arithmetic progressions. For example, a sequence `1 2 5 7 9 11 3` can be split into three arithmetic progressions: `1 2`, `5 7 9 11` and `3`. There is another way to split it into three arithmetic progressions (`1 2`, `5 7 9` and `11 3`), but there is no way to split it into two or less progressions.

Obviously, Andrew would love to split the sequence into as little progressions as possible. He's even willing to change some of the numbers (possibly to non-integers) so that the resulting number of arithmetic progressions is smaller. But it would be boring to just change all numbers to `1 2 3 ...`.

So Andrew has decided to assign a score of c to each change operation, and a score of p to each resulting arithmetic progression. If he changes n_c numbers and splits the result into n_p progressions, his total score is $cn_c + pn_p$. He wants his total score to be as small as possible.

Input

The first line of the input contains three integers n, $1 \leq n \leq 3\,000$ (the length of Anya's sequence), c and p, $1 \leq c, p \leq 10\,000$. The second line of the input file contains n integers between -1000 and 1000, inclusive — Anya's sequence.

Output

In the first and only line of the output print the minimal total score.

Example

For the input data:

```
11 2 5
-100 -100 -100 1 1 2 2 3 100 100 100
```

the correct result is:

```
19
```

Explanation. The result corresponds to the following sequence which can be split into three arithmetic progressions. `-100 -100 -100 1 3/2 2 5/2 3 100 100 100`.

[*]Arithmetic progression is a sequence in which the difference between any two consecutive elements is the same.

Task: BEA
Bear

Maratona de Verão, Day 1. Source file `bea.*` Available memory: 64 MB. *20.01.2014*

The Bear was very happy to see the winter come early. He had hoped that he would no longer be forced to wade through the forest in mud up to his knees while watching out for annoying mosquitoes. Unfortunately, the winter decided to make the Bear's life even harder and it snowed so much that our protagonist can barely move through the woods.

The Bear does not want to give up. He resolved to clean up the forest, or at least remove enough snow to uncover paths leading to his three favorite glades. The forest consists of n glades connected with m (bidirectional) roads. The glades are numbered from 1 to n. It seems that some of the roads might go through tunnels or bridges, which is a bit surprising to the Bear, but he currently has greater issues to deal with. For each road the Bear knows how much time is needed to remove the snow from it.

How much time does the Bear need to clean up enough roads to be able to move between his den (located at the glade 1) and his favorite glades, walking only through cleaned roads?

Input

The first line of the input contains two integers n and m ($4 \leq n \leq 50\,000$, $3 \leq m \leq 300\,000$). They denote respectively the number of glades in the forest and the number of roads connecting them. The second line contains three distinct integers p_1, p_2, p_3 ($2 \leq p_i \leq n$) – the indices of the Bear's favorite glades. Each of the next m lines describes a single road. The i-th of these lines contains three integers a_i, b_i, d_i ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$, $1 \leq d_i \leq 1\,000\,000$). They describe a road connecting glades a_i and b_i that takes d_i seconds to clean. Each pair of glades is connected by at most one road. Any two glades from the set $\{1, p_1, p_2, p_3\}$ can be reached from each other by walking through the roads.

Output

Output the minimum number of seconds required to remove snow from roads that allow moving between glades $1, p_1, p_2$ and p_3.

Example

For the input data:

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

the correct result is:

```
18
```

Task: BLI
Blizzard

Maratona de Verão, Day 1. Source file `bli.*` Available memory: 512 MB. *20.01.2014*

As usual winter has surprised the Byteburg city's work crews. There was a blizzard and the main road requires urgent plowing.

The responsibility is on the Department of Disaster Management, equipped with m plows numbered 1 to m. Each plow is associated with a connected fragment of the main road. Two fragments may intersect, however no fragment is contained in another one. The fragments do not have to cover the whole road (the road may consist also of tunnels, which clearly do not need to be plowed).

Unfortunately a strike of all the snow plow men is right about to begin. The head of the Department of Disaster Management convinced only one snow plow man not to take part in the strike. For this reason the only working snow plow man is assigned to handle all the plows. Now it is the time to determine the order in which the plows are going to be used. The Department of Disaster Management ordered the snow plow man to always select a plow, which has the least total length of yet unplowed parts of the fragment associated with it (after plowing a part of the road it is considered to be plowed till the end of the process). In case of a tie the snow plow man should select the plow of minimum number.

Input

The first line of the input contains two integers n and m ($1 \leq n \leq 10^9$, $1 \leq m \leq 300\,000$), denoting the length of the road and the number of plows. The following m lines describe subsequent plows, in the order of increasing numbers. The i-th of those lines contains two integers a_i, b_i ($1 \leq a_i < b_i \leq n$), meaning that the fragment of the road associated with the i-th plow starts at a_i and ends at b_i. You may assume that $a_i < a_{i+1}$, i.e. the plows are sorted according to the leftmost point of the associated fragment of the road. Furthermore no fragment is contained in a fragment associated with another plow.

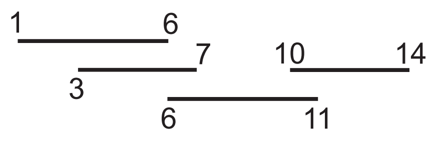
Output

Your program should output the numbers of plows in the order they shall be used. The output should consist of m lines, each containing a single integer.

Example

For the input data:

```
15 4
1 6
3 7
6 11
10 14
```



the correct result is:

```
2
1
3
4
```

Task: EVI
Evil and Good

Maratona de Verão, Day 1. Source file `evi.*` Available memory: 64 MB. *20.01.2014*

It should not be a surprise that Byteland is a completely normal country, namely an ordinary three-dimensional Euclidean space. What may be surprising is that there is a Plane in Byteland that separates Evil from Good. Everything on one side of the Plane is good, while everything else (on the other side and on the plane) is evil.

Byteasar is a well-known good citizen of Byteland, or at least he *was* good when he was at point U and lost his consciousness. He has just woken up at point W. Is he still good?

Input

The first line contains a single integer t ($1 \le t \le 50\,000$) that specifies the number of test cases that follow. Each of the following t lines contains 15 integers from range $[-500\,000, 500\,000]$. The numbers give the coordinates of five points: A, B, C, U, W. Points A, B, C lie on the Plane and are not colinear. Points U and W are the positions of Byteasar before and after he lost his consciousness.

Output

Output t lines with answers to the consecutive test cases. Each answer is either a word `TAK` (Polish for *yes*) if Byteasar is good after he woke up, or `NIE` (Polish for *no*), otherwise.

Example

For the input data:

```
2
-1 -45 0 10 6543 0 6543 -65 0 0 0 -1 0 0 1
-1 -45 0 10 6543 0 6543 -65 0 0 0 -1 100 1000 -145
```

the correct result is:

```
NIE
TAK
```

Task: MIS
Space Mission

Maratona de Verão, Day 1. Source file `mis.*` Available memory: 128 MB. *01.01.2014*

Byteland preparing to launch its first rocket into space. Byteasar is one of the space program employees and he is responsible for the process of boarding the rocket by the astronauts. The interior of the rocket consists of n cabins connected by two-way passages in such a manner that we can move between each two cabins in exactly one way (if we do not turn back). Crossing each corridor takes one Byteotian second. Cabins are numbered from 1 to n. The entrance to the rocket leads directly into the cabin number 1.

There will be n astronauts boarding the rocket, also numbered from 1 to n. For each $1 \leq i \leq n$ astronaut number i will take quarters in cabin numbered i. Astronauts enter the rocket one after the other in one second intervals (Byteotian seconds) and use the shortest way to reach their cabins. Astronaut number i starts to unpack his gear after reaching his cabin, which takes him exactly a_i Byteotian seconds.

The order of residents boarding the ship must be such, that no one should have to go through the cabin which is already occupied by its resident (regardless whether that occupant has just finished unpacking, or not).

Byteasar's task is to plan the rocket boarding process to run as quickly as possible, meaning that between the first astronaut entering the ship and the moment at which all astronauts finish unpacking, the least amount of time passes.

Input

The first line of input contains an integer n ($2 \leq n \leq 500\,000$) denoting the number of astronauts and the number of cabins. The second line contains a sequence of n integers a_i ($1 \leq a_i \leq 10^9$). Number a_i determines how much time astronaut number i needs to unpack. Subsequent $n-1$ rows describe the ship's cabin system. Each of them contains two integers a i b ($1 \leq a, b \leq n$), which indicate that cabins numbered a and b have direct corridor connection.

Output

Output the minimum time required by all astronauts to board and accommodate in their cabins.

Example

For the input data:

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

the correct result is:

```
7
```

Task: NIM
Nim3

Maratona de Verão, Day 1. Source file `nim.*` Available memory: 64 MB. *20.01.2014*

The three brothers Antek, Bartek and Cezary really like playing Nim. At some point, they came up with the idea to play this game between the three of them. At the beginning they lined up many pebble stacks. Then they moved in turns, taking any amount of pebbles (a positive number, obviously) from the chosen stack. Antek, the youngest of the brothers, always performed the first move, the next move belonged to Bartek, another to Cezary, then again moved Antek, and so on. Loser was the player who could not make a move. Unfortunately, it was usually Antek.

His older brothers did not compete with each other and did everything to make him lose. The youngest of the brothers decided to change it and suggested that the first place be taken by a person who will perform the last move; *third*— by a player who can not move, and the second place — by the last of the three brothers. Brothers analyzed the new rules and came to the conclusion that the way in which a move is performed will be guided by the following algorithm defined recursively:

1. Consider all the moves that you can do and for each of them calculate what will be the result of the game (applying the algorithm recursively).
2. Select the move which gives the best result (in the case of many equally good opportunities select any one of them).

Knowing the initial situation, decide who will win the game in case all three brothers play in line with the above description.

Input

The first line of input contains an integer t ($1 \le t \le 1000$) denoting the number of test cases. Subsequent t lines describe one test case each.

A single test case description starts with a positive integer n_i, which indicates the number of stacks of pebbles in the game. Next follows a string of n_i integers a_{ij} ($1 \le a_{ij} \le 10^{18}$), which determines the initial number of pebbles in each stake. The sum of all n_i does not exceed 1 000 000.

Output

The output should produce t lines with answers for subsequent test cases.

The answer for one case is the letter A, B, C, which is the first letter of the winner's name of the corresponding game.

Example

For the input data:

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

the correct result is:

```
B
C
A
```

Task: SOR
Sorting Machine

Maratona de Verão, Day 1. Source file `sor.*` Available memory: 128 MB. *20.01.2014*

Byteasar works at the Sorting Machines Institute. For several months he has been busy inventing new and efficiently working machine, and finally he succeeded: the invention devised by Byteasar can easily sort any string consisting of not more than k different natural numbers.

Byteasar's boss approached him today with an urgent order to sort a certain permutation of numbers ranging from 1 to n, where $n > k$. Byteasar quickly realised that the permutation is too long for his machine, however he failed to persuade the boss to drop this idea. Presently he had just enough time left to run the machine once, selecting an k-element fragment (of consecutive numbers) of boss's permutation. He decided to do it in such a way, that the resulting permutation is lexicographically as small as possible. But how to choose the right fragment to perform the sorting?

Input

The first line contains two integers n and k ($2 \leq k < n \leq 1\,000\,000$), indicating the length of the boss's permutation and the maximum fragment length, which is still possible to be sorted by Byteasar's machine. The second line contains a permutation of the numbers $\{1, \ldots, n\}$, ie, the sequence of n pairwise distinct integers ranging from 1 to n.

Output

The first and only line of output should contain a sequence of n numbers denoting the lexicographically minimum permutation Byteasar can obtain running his machine not more than once.

Example

For the input data:

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

the correct result is:

```
2 1 3 4 5
```

Task: SUB
Substitution

Maratona de Verão, Day 1. Source file `sub.*` Available memory: 64 MB. *20.01.2014*

A *substitution* operation works on a source string S, composed of letters `a` and `b` and two strings T_a and T_b. Each occurrence of `a` in S is replaced with T_a, whereas each occurrence of `b` is replaced with T_b. For example, if $S =$ `aab`, $T_a =$ `aba`, $T_b =$ `bbba`, the result of the operation is `abaababbbba`.

You are given the string S and the resulting string W. Find sequences T_a and T_b that were used in the substitution or check that a solution does not exist.

Input

The first line of input contains a single integer denoting the number of test cases that follow. Each test case consists of two lines that contain nonempty strings S and W. They contain only letters `a` and `b` and the length of each of them is at most 200 000. The total size of each input file is at most 2.5 MB.

Output

Output answers for the consecutive test cases.

If there exist two nonempty sequences T_a and T_b, for which the substitution performed on S gives the string W, output them in two separate lines (in the first one the string that is substituted for `a`, and in the second one — the string substituted for `b`). If one of the letters is not present in S at all, output `-` (minus, ASCII code 45) in the line corresponding to this letter. If the solution does not exist at all, output a single line containing the word `NIE` (Polish for *no*). If there are multiple solutions, your program may output any of them.

Example

For the input data:

```
5
abaababa
abaababaabaab
ab
ba
aba
aabbb
aaa
aabaabaab
ba
aabbaabab
```

one of the correct answers is:

```
ab
a
b
a
NIE
aab
-
b
aabbaaba
```

Task: SWE
Sweets

Maratona de Verão, Day 1. Source file `swe.*` Available memory: 256 MB. *20.01.2014*

Bytie loves to make plans. He has already planned his k future birthday parties. For his i-th birthday he is going to order n_i types of sweets:

- a_i sweets of the first kind,
- $a_i + 1$ sweets of the second kind,
- $a_i + 2$ sweets of the third kind,
- ...
- $a_i + (n_i - 1)$ sweets of the n_i-th kind.

Now, Bytie is wondering how many friends he should invite for each of the parties. His plan is to distribute sweets of each kind evenly among some group of at least two guests. On each party, each friend will get at most one kind of sweets. What is the minimal number of children which can be invited to every birthday party?

Input

The first line contains a single integer k specifying the number of parties that Bytie has planned ($1 \leq k \leq 10\,000$). Each of the following k lines contains two integers n_i, a_i that describe the sweets ordered for i-th birthday ($1 \leq n_i \leq 3\,000\,000$, $2 \leq a_i \leq 3\,000\,000$).

Output

Output k lines. The i-th of these lines should contain the minimal number of children that have to be invited to i-th birthday.

Example

For the input data:

```
2
2 3
3 4
```

the correct result is:

```
5
9
```

Task: TES
Oar Tester

Maratona de Verão, Day 1. Source file `tes.*` Available memory: 512 MB. *20.01.2014*

A Byteotian company SSO (Super Strong Oars) has produced n new types of rowing oars. The company does not have a precise method of measuring the jet force that can be obtained using the blades of oars of different types. For this reason SSO hired a professional oar tester, Byteasar. Unfortunately the Byteasar's method used to determine the force of an oar's blade is also imprecise — Byteasar takes every possible pair of oar types and rows some distance using those. As a result he concludes: *I am sure that the sum of forces of the two oar blades is not greater than x, and at the same time at least one of the oar blades provides force of at least y.*

When all the tests were performed Byteasar delivered all the results to Byteen, the head of SSO. Byteen thinks that the volume of gathered data is too large, and moreover the output seems meaningless. She would appreciate any sequence of oar blade's forces, consistent with the tests performed by Byteasar. Your task is to deliver such a sequence.

Input

The first line of the input contains a single integer n ($1 \leq n \leq 300$), denoting the number of oar types. Oar types are numbered 1 to n. Each of the following n lines contains exactly n integers: the j-th integer in the i-th of those lines is x_{ij} ($1 \leq x_{ij} \leq 10^9$), which denotes an upper bound on the sum of the forces of oar blades of types i and j. The next line is empty. Finally there are n lines of n integers each. The j-th integer in the i-th of those lines is y_{ij} ($1 \leq y_{ij} \leq 10^9$), meaning that the blade of an oar of type i *or* the blade of an oar of type j enables to generate force of at least y_{ij}.

Output

Your program should output n positive integers: the i-th integer being the force that can be produced using the i-th oar's blade in some sequence of values consistent with all the information gathered by Byteasar. You may assume that there exists at least one such sequence.

Example

For the input data:

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

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

the correct result is:

```
2 3 3
```

Task: UNS
Unsure

Maratona de Verão, Day 1. Source file `uns.*` Available memory: 128 MB. *20.01.2014*

You're given an undirected bipartite graph*, which is k-regular (there are exactly k edges starting at every vertex). Unfortunately, we're not sure what the expected output is. Thus, it suffices if you solve *any* of the following tasks:

- find the maximum matching, or
- find a simple path that consists of $2k - 1$ edges.

Output

The first line contains three integers n, m and k ($2 \le n \le 10\,000$, $1 \le m \le 50\,000$, $1 \le k < n$). They denote, respectively, the number of vertices, edges, and the degree of every vertex. The vertices are numbered 1 to n. Each of the following m lines contains the description of one edge: a pair of integers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$). They denote the numbers of vertices connected with the edge.

There are exactly k edges incident to any vertex. Each pair of vertices may be connected with at most one edge. The vertices of the graph may be colored with two colors, in such a way that any two vertices connected with an edge have distinct colors.

Output

The first line should contain a single word `sciezka` (Polish for *path*) or `skojarzenie` (Polish for *matching*).

If your program outputs `sciezka` in the first line, the second line should contain a sequence of $2k$ integers from range $[1, n]$ that denote the numbers of vertices on the path. The path may go through any vertex at most once.

If your program outputs `skojarzenie` in the first line, the second line should contain an integer s denoting the number of edges in the maximum matching. Each of the following s lines should contain a pair of integers denoting the numbers two vertices connected with an edge from the matching.

Example

For the input data:

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

the correct result is:

```
sciezka
1 3 2 4
```

as well as:

```
skojarzenie
2
1 3
2 4
```

*A graph is called bipartite if its vertices can be partitioned into two sets, in such a way that the edges only connect vertices from different sets.