Problem A: Aliases

Time limit: 40s, memory limit: 1GB.

The Individual Fast Programming Contest (IFPC) has seen a record number of registrations this year – n competitors. Your task is to set up accounts for them. Each account will have an *alias*, i.e. a username. The IFPC system only accepts aliases composed of lowercase English letters and numbers.

In order not to mess things up, you create aliases based on the first and last names of the participants, following the same algorithm: the first a letters of the first name, then the first b letters of the last name, and finally c digits of your choice. If the first name has fewer than a letters or the last name has fewer than b letters, you just take all the letters (then the alias is shorter, but it is fine). For example, for $a = b = c = 3$ James Bond could get the alias `jambon007` and Lady Di could get `laddi123`.

You want the aliases to be as short as possible – that saves a few bytes of memory and you like saving memory. On the other hand, all the participants must have unique aliases. For a given list of participants (their names and surnames), find a, b and c that will allow you to create different aliases for all participants, and for which $a+b+c$ will be as small as possible. Numbers a, b and c can equal 0 (but not all at the same time – the aliases must be non-empty).

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 6$). The descriptions of the test cases follow.

The first line of the test case contains one integer n ($1 \leqslant n \leqslant 200\,000$) – the number of participants. In the following n lines there are the first and last names of the contestants. Both first and last name are sequences consisting only of lowercase letters of the English alphabet.

The sum of the lengths of all names and surnames of the competitors in one test case does not exceed 1 500 000. It is possible that two participants have the same name and surname (and they still need to get different aliases).

Output

For each test case, output three non-negative integers a, b and c that allow you to create unique aliases and have the lowest possible sum.

If there are several possible solutions with the minimal sum, you can output any of them.

Example

For an example input:	A possible correct output is:
1 11 sven eriksson erik svensson sven svensson erik eriksson bjorn eriksson bjorn svensson bjorn bjornsson erik bjornsson sven bjornsson thor odinsson odin thorsson	1 1 0

Explanation

Unique aliases can be obtained by taking the initials of all participants (the first letters of the first name and surname: `se`, `es`, `ss`, `ee`, ...). The answer `1 0 1` is also correct as you can create unique aliases by adding one number to the first letter of each person's name.

Problem B: Bars

Time limit: 6s, memory limit: 1GB.

Being the mayor of the village of Straightlineham is a real challenge. Admittedly, the expenditure on road infrastructure is minimal - houses of all n inhabitants, numbered from 1 to n, lie in order along one straight road that runs through the entire village. Still, sometimes you have to make difficult decisions, such as issuing permits to open a bar.

It turns out that all Straightlineham residents dream of opening their own bar. There have been n permit forms submitted, one per inhabitant. Each of the residents presented their business plan, from which you are most interested in the proposed tax amount if the bar gets opened: i-th resident promises to pay the village p_i gold coins from each customer.

You plan to grant permission to open bars to a certain (non-empty) subset of the inhabitants (maybe even everyone). Each resident, regardless of receiving permission to open their own bar, will become a client of two others: the nearest one strictly to the left of his house and the nearest one strictly to the right (as long as such bars exist – otherwise a given person will be a customer of fewer bars). When determining the nearest bars, we do not take into account the one that is run by the resident in question – after all, even the best bartender should not serve themselves. After deciding which of the bars get opened, each one will start to generate income to the village budget of p_i gold coins for each client. For example, if $n = 5$ and the third and fifth bar get opened, the first one will have 4 clients and the other 2 clients, generating total tax revenue of $4 \cdot p_3 + 2 \cdot p_5$.

Knowing the promised amount of tax to be paid for each of the hypothetical bars, determine the maximum profit that can be achieved by issuing permits in an optimal way.

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 10\,000$). The descriptions of the test cases follow.

The first line of a test case contains one integer n ($2 \leqslant n \leqslant 500\,000$) - the number of inhabitants of Straightlineham.

The second line of a test case contains n integers p_i ($1 \leqslant p_i \leqslant 10^9$) - the proposed amount of the customer tax for each of n hypothetical bars.

The total number of inhabitants in all test cases will not exceed $3 \cdot 10^6$.

Output

For each test case, output one integer representing the maximum total profit you can make by issuing permits optimally.

Example

For an example input:	The correct output is:
2 4 5 2 2 6 5 1 5 4 4 1	33 29

Explanation

In the first example test, the optimal solution is to allow opening the first and the last bar. Each of them will have 3 clients, which generates total revenue of $3 \cdot 5 + 3 \cdot 6 = 33$ coins.

In the second sample test, it is optimal to allow opening all the bars except the third. In this case, the profit is $1 \cdot 1 + 3 \cdot 5 + 3 \cdot 4 + 1 \cdot 1 = 29$. If instead all the bars were allowed to open, the profit would be smaller: 28 coins.

Problem C: Ctrl+C Ctrl+V

Time limit: 5s, memory limit: 1GB.

– *How are you doing with that autobiography?*
– *Er, Ania, what? Was there any homework for the Polish class?*
– *Come on, we were supposed to write an autobiography. Did you forget?*
– *I forgot. Can I copy it from you?*
– *You want to copy an autobiography!? Fine, but at least make some changes...*

You are given a string s consisting of lowercase English letters. This is an autobiography written by Ania, which means it may contain the word `ania` as a contiguous substring, perhaps even multiple times. Determine the minimum number of characters that need to be changed in s so that it does not contain the word `ania` as a contiguous substring.

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 10\,000$). The descriptions of the test cases follow.

The first and only line of each test case contains the string s - Ania's autobiography. The string's length is l ($1 \leqslant l \leqslant 10^6$). It solely consists of lowercase letters of the English alphabet.

The total length of the strings in all test cases does not exceed $5\,000\,000$.

Output

For each test case, in a separate line, write one integer, representing the minimum number of changes that have to be made to the autobiography so that it does not contain a substring `ania`.

Example

For an example input:	The correct output is:
3 aniasieurodzilaapotemnicsieniedzialo nicciekawegouanianiagnieszkianialicji jeszczekrotszaautobiografiaani	1 2 0

Problem D: Dazzling Mountain

Time limit: 10s, Memory limit: 1GB.

The highest peak of Bytelandshire is Mount Stackframe - a picturesque mountain famous for its beautiful views. On Mount Stackframe there are n shelters for tourists (numbered from 1 to n), and $n-1$ trails, each connecting two shelters. Shelter number 1 is located at the very top of the mountain and it is known that from any other shelter you can get to 1 in exactly one way (as long as you do not turn back on the way). The shelters with exactly one trail adjacent to them are located at the foot of the mountain (except for shelter 1 which, needless to say, is not a ground shelter even if there is only one trail adjacent to it). The traditional Mount Stackframe hike begins at some ground shelter and ends at the top.

If a tourist stops at a certain shelter x and looks down, they see a number of shelters – these are exactly the ones from which the path to 1 passes through x. A new idea of the National Park's leadership is to build a viewpoint wherever the number of shelters visible from x (including x) is exactly d.

Find all values of d for which every tourist hiking from the bottom to the top of the mountain will visit at least one viewpoint, regardless of which ground shelter they have started from.

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 10\,000$). The descriptions of the test cases follow.

The first line of the test case contains one integer n ($2 \leqslant n \leqslant 1\,000\,000$) – the number of shelters. Each of the following $n-1$ lines consists of two integers a_i and b_i ($1 \leqslant a_i, b_i \leqslant n$) denoting that the i-th trail connects shelters a_i and b_i. The trails do not intersect outside the shelters.

The total number of shelters in all test cases does not exceed $3 \cdot 10^6$.

Output

For each test case, output two lines: the first should contain the number of different values d, for which each visitor always passes a viewpoint. The second line should contain these values in ascending order.

Example

For an example input:	The correct output is:
1 9 1 2 2 3 3 4 3 5 2 6 6 7 7 8 7 9	4 1 3 8 9

Problem E: Euclidean Algorithm

Time limit: 30s, memory limit: 8MB.

You sit an exam and keep wondering how to calculate the greatest common divisor. With utmost effort you manage to recall that $\texttt{gcd}(x, y)$ is the greatest integer $d \geqslant 1$ that divides both x and y. You vaguely remember something along these lines having appeared at one of the lectures. Alas, even if your mind has managed to register any shreds of information during the class, they are now long gone, displaced by the adventures of Gebyte the Witcher as well as the newest episodes of Peaky Byters.

You start with x and y on the blackboard... then you compute some results of subtraction of the previously written numbers...? You try to chase away an unpleasant thought that it is going to end up exactly like the median problem last year... Or was it just one of your nightmares?

Based on your foggy memories from the lecture you came up with the following algorithm. You start with numbers x and y written on the blackboard. In one step you can choose any two numbers a and b that are already written on the blackboard, and write down $c = 2a - b$, conditional upon $c > 0$. The result for the pair (x, y) is the lowest number that can be written on the blackboard after repeating the above procedure some number of times (maybe zero).

After checking a few test cases, your algorithm looked promising. For example, for a pair $(10, 14)$, one of the numbers that you can write down in the first step is 6 (because $2 \cdot 10 - 14 = 6$), while in the next step you can write 2 (because $2 \cdot 6 - 10 = 2$). On the other hand, one can prove that it is impossible to obtain 1, so the algorithm's answer is correct. Unfortunately, for $(10, 16)$ the lowest number you managed to get was 4, while the correct answer is 2. Something is amiss...

For a given n determine the number of pairs (x, y) $(1 \leqslant x < y \leqslant n)$ for which your algorithm returns the correct value of $\texttt{gcd}(x, y)$ [1].

Observe that this task has a low memory limit – 8MB.

Input

The first line of input contains the number of test cases z ($z \geqslant 1$). The descriptions of the test cases follow.

The first and only line of a test case contains one integer n ($n \geqslant 2$) with the meaning explained in the problem statement.

Test cases

Each input file will belong to one of the three groups:

- $z \leqslant 3000$, $n \leqslant 10^6$
- $z = 30$, $n \leqslant 10^9$
- $z = 3$, $n \leqslant 10^{11}$

Output

For each test case, output one integer representing the number of pairs that we are looking for.

[1] In particular, if $\texttt{gcd}(x, y) = x$, then we consider the algorithm to work correctly, because x is already written on the blackboard at the start.

Example

For an example input:	The correct output is:
3 2 5 14	1 9 62

Problem F: Flower Garden

Time limit: 30s, memory limit: 1GB.

In front of the Bytingham Palace there is a beautiful garden. Every year masses of travellers wish to see this wonder with their own eyes. The King Intles III has been investing for years mainly in its length, so as many as $3n$ flowers can be planted in a row.

The current gardener, who invested a lot of energy in this majestic subject of the royal pride, has recently decided to retire early, even before turning forty years old. You have just arrived at the palace to take over his role and although the glimpse at the face of your predecessor made you question your abilities to estimate age of other people, you eagerly accepted the job offer. Now the first task awaits you!

King Intles has decided that two kinds of flowers will be planted in the garden this year: violets and roses. They have to conform to the scheme, which is defined by the multi-page royal decree. The first page states as follows:

Flower beds for violets and roses are numbered from 1 *to* $3n$.

The next pages state:

At least one of the following conditions must be satisfied:

- *All flowers planted on beds from* a_i *to* b_i *inclusive must be roses.*
- *All flowers planted on beds from* c_i *to* d_i *inclusive must be violets.*

You are astonished to find q pages with almost the same instructions, differing only by values of a_i, b_i, c_i, d_i. It is not that bad so far, but there is also one terrifying statement on the last page:

There are $2n$ *roses and* $2n$ *violets available.*

Suddenly you remember the face of the gardener you saw when coming to the royal palace and you start regretting your decision. Is this task even solvable? Find an assignment of the flowers that meets the conditions, or determine that it does not exist (and start considering how to avoid the king's fury).

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 10^5$). The descriptions of the test cases follow.

The first line of each test case contains two integers n and q ($1 \leqslant n \leqslant 33\,333, 1 \leqslant q \leqslant 10^5$).

Next q lines contain the description of the royal decree. Each i-th line consists of four numbers a_i, b_i, c_i, d_i ($1 \leqslant a_i \leqslant b_i \leqslant 3n, 1 \leqslant c_i \leqslant d_i \leqslant 3n$) with the meaning described in the problem statement.

Sum of n in all test cases does not exceed 333 333. Sum of q in all test cases does not exceed 10^6.

Output

For each test case, in the first line print a single word `TAK` if it is possible to plant all the flowers according to the rules, or `NIE` otherwise.

If the answer is affirmative, in the second line output a string of length $3n$, consisting of characters `F` and `R`. Character `F` at the i-th position of this string means that the i-th flower planted in the row should be violet, and character `R` means the rose. This string must not contain more than $2n$ characters `F` nor more than $2n$ characters `R`.

Example

For an example input:	A possible correct output is:
2 1 3 1 1 2 2 1 2 3 3 1 1 3 3 1 3 1 1 2 2 2 2 3 3 3 3 1 1	TAK RFF NIE

Problem G: Great Chase

Time limit: 5s, memory limit: 1GB.

Game of *Cops and Robber* is played on a long street, which for the purpose of this problem can be identified with the number axis. One player (the robber) places himself on the 0 position on the axis, and the remaining n players (cops) stand on both of his sides (at least one on each side). At the start of the game, each cop starts running towards the robber at a set speed, and the robber starts escaping at the speed of v, which is greater than the speed of any cop, going rightwards (along the increasing values on the axis). Every time the robber reaches the first cop rushing at him, he turns back in negligibly short time and continues running in the opposite direction. This situation repeats until two cops running in the opposite directions meet, with the robber between them.

For a given set of initial positions of the robber and the cops, determine the total distance the robber will travel throughout the game.

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 10\,000$). The descriptions of the test cases follow.

The first line of the test case contains two integers n ($2 \leqslant n \leqslant 400\,000$) and v ($1 < v \leqslant 10^6$) - number of cops and robber's speed.

Each of the following n lines of the test case contains two integers p_i ($-10^{12} \leqslant p_i \leqslant 10^{12}$, $p_i \neq 0$) and v_i ($1 \leqslant v_i < v$) - the starting position and the speed of the i-th cop.

The total number of cops in all test cases will not exceed $2 \cdot 10^6$.

Output

For each test case, on a separate line, write one real number in decimal format (not in scientific notation), representing the distance travelled by the robber. For the answer to be considered correct, the relative or absolute error should not exceed 10^{-8}. In other words, if your algorithm answers a and the correct answer is b, then your answer will be accepted if $\frac{|a-b|}{\max(1,b)} \leqslant 10^{-8}$.

Your output number should have no more than 20 digits after the decimal point.

Example

For an example input:	A possible correct output is:
3 4 9 10 2 -7 2 -6 1 7 1 2 8 -1 7 1 6 2 3 -1000000000000 1 1000000000000 1	38.25 1.23076923 3000000000000

Empty lines in the sample input have been added for readability. They are not present in the test files on which your solution will be run.

Remark

Note that in this problem we consider both absolute and relative error. In particular, this means that when the correct answer is large, your program's allowed error is also large. In the last sample test case, the correct answer is $3 \cdot 10^{12}$, so any number that differs from it by no more than 30 000 would be accepted.

Problem H: Hyperloop

Time limit: 50s, memory limit: 64MB.

The year is 2077. Artificial Intelligence has just solved the last problem of colouring hypergraphs, and the newly unemployed researchers from the Theoretical Computer Science department have been commissioned by an eccentric billionaire Melon Usk to develop software for the Hyperloop complex – a system of super-fast intercity connections.

Due to rising water levels, the only land mass left on Earth is a single island on whose shore there lie n surviving cities, which we will name for simplicity 1, 2, ..., n in the order of occurrence along the shore. Melon owns ferries that run around the island, i.e. between cities i and $i+1$ for $1 \leqslant i < n$ and between cities n and 1. He also owns Hyperloop connections between certain pairs of *non-neighbouring* cities [1]. The only thing left is to design the software for computing the shortest routes between cities; in the beta version it will only support travel from 1 to n.

The task might seem trivial: after all, the software is supposed to take into account both types of connections, and between the cities 1 and n there is already a *direct* ferry connection, but it doesn't have to be the fastest route! Even worse, there may be several shortest paths. In such situation the system should prefer paths which incorporate the longest possible connections between cities (the longer a single journey leg lasts, the more likely the passengers are to order Melon's coffee at inflated prices during the ride). Formally, in order to compare paths **of the same total length**, one should list the lengths of the connections in them (including repetitions), sort each sequence in non-increasing order, and select the lexicographically largest [2].

Given the description of the ferries around the island and the Hyperloop connections, find the optimal path from city 1 to city n. When comparing paths, both kinds of connections are treated equally. All the connections are bidirectional.

Observe that this task has a low memory limit – 64MB.

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 600$). The descriptions of the test cases follow.

The first line of the test case contains two integers n, m ($3 \leqslant n \leqslant 100\,000$, $n \leqslant m \leqslant 300\,000$) – the number of cities and the number of connections.

Each of the following m lines contains three integers u_i, v_i, d_i ($1 \leqslant u_i \neq v_i \leqslant n$, $1 \leqslant d_i \leqslant 50\,000$), meaning in order: cities connected by a given bidirectional connection, and the time needed to travel it.

Among the connections given at the input $(1,2)$, $(2,3)$, ..., $(n,1)$ will appear, corresponding to the ferry connections.

There will be at most one connection between any pair of cities.

The total number of cities in all test cases will not exceed 400 000. The total number of connections in all test cases will not exceed 800 000.

[1] Hyperloop connections use underground tunnels. The tunnels that could cross were simply laid at different depths.

[2] The string $a_1, ..., a_p$ is lexicographically smaller than the string $b_1, ..., b_q$ if it is its prefix or there exists $i \leqslant \min(p,q)$ such that $a_j = b_j$ for $j < i$ and $a_i < b_i$. In this task, the sequences to be compared always have the same total sum, so neither of them will be a prefix of the other.

Output

For each test case, on the first line print one integer k, representing the number of cities in the optimal path. On the second line, output k different integers – the identifiers of subsequent cities on the path. The chosen path should start in the city 1, end in the city n, and be optimal as defined in the problem statement.

If there are multiple optimal paths, you can output any of them.

Example

For an example input:	A possible correct output is:
2 4 6 1 2 1 1 3 2 2 3 1 2 4 2 3 4 1 1 4 4 6 11 1 2 9 2 3 12 3 4 3 4 5 5 5 6 10 6 1 22 2 4 9 3 6 1 4 6 5 2 5 2 3 5 8	3 1 2 4 5 1 2 5 3 6

Explanation

In the first sample test, the minimum distance from city 1 to city 4 is 3, and it is achieved by three paths: $1 \to 2 \to 4$, $1 \to 3 \to 4$ and $1 \to 2 \to 3 \to 4$. As defined in the problem statement, the first two paths are better than the third, because lexicographically $(2, 1) > (1, 1, 1)$. Either of the first two paths is a correct answer. The direct path $1 \to 4$ is not considered as its total length is 4.

In the second example test, the paths with the minimum distance are $1 \to 2 \to 5 \to 3 \to 6$ and $1 \to 2 \to 5 \to 4 \to 3 \to 6$. The first one is optimal, as comparing their sorted lengths we obtain $(9, 8, 2, 1) > (9, 5, 3, 2, 1)$.

Problem I: Investors

Time limit: 4s, memory limit: 1GB.

The year is 2087. Artificial Intelligence has solved even those hypergraph colouring problems that had never been posed, while the completed Hyperloop network turned out to be the slowest subway in the world. The software developed in Problem H has been working flawlessly for the past 10 years, but (as always) the sociological aspect turned out to be the weakest link of the plan: whenever the system proposed a Hyperloop route involving the longest possible connections, the blood pressure of a passenger looking at the itinerary was rising so much that they no longer had to buy Melon's coffee at inflated prices.

In order to calm down the investors, Usk decided to present the company's financial results. Unfortunately, they were not too impressive because of low ticket revenues and high coffee purchasing costs. The eccentric billionaire decided to "tweak" them a bit.

The company's results can be represented as a sequence of n numbers denoting the revenue for the following months. In order not to arouse suspicions, Usk decided to increase the numbers on a chosen contiguous interval of the sequence by some positive integer. Performing this operation once fell short of the visionary's expectations. So he tried a second time, and a third... Ultimately, he decided that he would like to perform at most k operations of increasing the results (possibly by different values).

Usk knows that investors are looking primarily at the growth of the company's revenues, while they do not like when the revenues drop compared to previously achieved results. Therefore, he would like to minimise the number of inversions [1] among his financial results. Are you able to save the future of his company?

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 400$). The descriptions of the test cases follow.

In the first line of the test case there are two integers n, k ($1 \leqslant n \leqslant 6000$, $0 \leqslant k \leqslant n$).

In the second line there is a sequence of numbers $a_1, a_2, ..., a_n$ ($0 \leqslant a_i \leqslant 10^9$), denoting the financial results of Melon's company.

The sum of n among all test cases does not exceed 6000.

Output

For each test case, output exactly one integer – the minimum number of inversions in the sequence after applying an operation of increasing a contiguous interval no more than k times.

[1] As a reminder, given a sequence $a_1, a_2, ..., a_n$, an inversion is a pair of indices i, j such that $i < j$ and $a_i > a_j$.

Example

For an example input:	The correct output is:
2 6 1 4 5 6 2 2 1 6 2 4 5 6 2 2 1	2 0

Problem J: Job for a Hobbit

Time limit: 2s, memory limit: 1GB.

One ring to rule them all, one ring to find them, One ring to bring them all, and in the darkness bind them...? Truly, Byteon the dark sorcerer could never wrap his head around this. How could have Sauron been so reckless as to let his fate depend on just one ring, and then brag about it in the inscription?

Byteon also draws his power from magic rings, but – to avoid Sauron's fate – he doesn't bet everything on one small piece of jewellery. In front of his Black Tower there is a row of n magical poles, and on each pole there are exactly k coloured rings.

The good wizard Bitalf has prophesied that strength has its roots in diversity, and Byteon will lose it when each pole is single-coloured (or empty). Bitalf himself is reluctant to move the rings, but he conjured up two additional poles – one on the right and the other on the left side of the structure (so now there are $n + 2$ poles, the outermost ones are empty, and there are still exactly k rings on all the other poles).

The job of moving the rings fell (as always) on a hobbit, Bitbo Byteins. Each day, Bitbo will climb unnoticed onto one of the poles and move the topmost ring from that pole onto the top of one of the adjacent poles. Fortunately, Byteon's evil eye has lost its vigilance over the years and will not notice the tiny hobbit nor the difference in the rings. What complicates the task, though, is the fact that no pole can fit more than k rings.

Nobody is sure whether Bitalf's prophecy makes sense and the rings can be moved accordingly. Help the brave hobbit prepare a plan! It should be no longer than a million days so that Bitbo has a chance to live until the end of the story.

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 25$). The descriptions of the test cases follow.

In the first line of the test case there are two integers n, k ($1 \leqslant n \leqslant 50$, $1 \leqslant k \leqslant 10$).

In each of the following n lines there is a sequence of numbers: $a_{i1}, a_{i2}, ..., a_{ik}$ ($0 \leqslant a_{ij} \leqslant 10^9$), denoting rings on the i-th pole given in order from the bottom to the top. The poles marked with numbers 0 and $n + 1$ exist, but are initially empty.

Sum of n among all test cases does not exceed 50.

Output

For each test case, write `TAK` if it is possible to defeat Byteon by placing the rings so that each pole has rings of at most one colour, or `NIE` otherwise.

If the answer is affirmative, in the following lines output a plan to accomplish the goal. In the first line, output the number of moves p ($0 \leqslant p \leqslant 10^6$). In each of the following p lines, output two integers a_i, b_i ($0 \leqslant a_i, b_i \leqslant n + 1, |a_i - b_i| = 1$), which means that Bitbo should move the top ring from the a_i-th pole to the top of the b_i-th pole. Before performing this operation, pole a_i must not be empty, while pole b_i must contain less than k rings.

If there are many possible plans, you can print any of them, as long as you fit within the limit of 10^6 moves.

Example

For an example input:	A possible correct output is:
2 2 2 1 2 2 1 1 4 1 2 3 4	TAK 2 1 0 2 1 NIE

Problem K: Kooky Tic-Tac-Toe

Time limit: 6s, memory limit: 1GB.

Every player in "Tic-Tac-Toe" sooner or later realises that there exists a simple strategy to guarantee a draw, which makes this game a bit uninteresting. In this problem, instead of making the optimal moves in the classic game of "Tic-Tac-Toe", we will consider its more general variant and solve the reversed problem: for a given state of the board, is it possible that this is the state in which some correct game ended?

For the purpose of this problem, we will assume that the game is played on a board $n \times n$, consisting of (initially empty) unit squares. Players alternate to make moves, placing their symbol with each move (for one of the players it is a circle, for the other a cross) into the selected empty field. **The first move can be made by any player.** After a player has performed their move, if a sequence of identical symbols has appeared on k consequtive sequares in one of the four possible directions (horizontally, vertically, diagonally, or antidiagonally [1]), the corresponding player wins and the game is over. Finally, if neither player wins and the entire board is full, the game is declared a draw.

The input contains a board with some circles, crosses and free squares. Decide if there exists a valid game that could end with the given board. Note that in this hypothetical game, players do not have to play optimally, they just need to follow the game rules described above.

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 10\,000$). The descriptions of the test cases follow.

The first line of each test case contains two integers n, k ($3 \leqslant n \leqslant 6$, $2 \leqslant k \leqslant n$) – the board size and the number of adjacent symbols needed to win.

Each of the following n lines contains n characters, describing the board. Possible signs in the board description are `.` (blank space), `x` (cross), and `o` (circle).

Output

For each test case, output `TAK` on the first line if the given board can be a final correct position in Tic-Tac-Toe, or `NIE` otherwise.

If the answer is affirmative, output the description of an example game in the following lines, giving the possible order of filling the fields. In the i-th line of the game description, output two integers x_i, y_i ($1 \leqslant x_i, y_i \leqslant n$), meaning that the field in the x_i-th row (counting from the top) and the y_i-th column (counting from the left) was filled in the i-th move.

If there are multiple correct answers, you can output any of them.

[1] The "diagonal" direction means the oblique direction to the right and down, and "antidiagonal" direction means the oblique direction to the left and down.

Example

For an example input:	A possible correct output is:
7 3 3 x.o xxx o.o 4 3 xx.x ...o ..o. .o.. 3 3 xoo oxx xoo 3 2 xoo oxx xoo 3 3 xox .o. xox 3 2 xo. ..x xo. 3 3 x.. .x. ..x	TAK 2 2 1 3 1 1 3 3 2 1 3 1 2 3 TAK 1 1 3 3 1 2 4 2 1 4 2 4 TAK 1 2 1 1 1 3 3 1 2 1 2 2 3 2 2 3 3 3 NIE NIE NIE NIE

Explanation

Test 1: The game was won by cross.

Test 2: Circle won, taking three consecutive squares in the antidiagonal direction.

Test 3: The game ended in a draw.

Test 4: Any sequence of moves leading to this board would have resulted in a win for one of the players earlier than in the ninth move.

Test 5: Cross had to make the last move, which contradicts the fact that circle won.

Test 6: The board is in a valid state, but the game is not finished.

Test 7: Cross made three moves while the circle made none.

Problem L: Line Replacements

Time limit: 10s, memory limit: 1GB.

Ordnungrad is a city whose inhabitants take pride in maintaining perfect order in all areas of life: the streets are always clean, trams always come according to the timetables, and the police department got closed when a city council ordinance declared that carrying out criminal activities within the city borders is no longer possible. This last titbit has caught the attention of Byteasar, the scrap collector.

The city tram network consists of n intersections, near p of them there are tram loops located, and the intersections are connected by $n-1$ two-way tracks such that it is possible to reach any intersection from any other in exactly one way. Byteasar has identified k poorly lit tracks. Each of the following k nights he plans to dismantle one of them and transport the obtained scrap material outside the city.

The city council will surely prefer to pretend that the tracks are being closed due to renovations rather than admit that such a serious disorder took place during their tenure... or at least that's what Byteasar is counting on. However, the Ordnungradians are very much attached to the fact that if there have been 10 tram runs per hour available on their street for years (counting all the lines going through this street), then next day there will also be exactly 10 runs – no more, no less. If any morning the city council failed to satisfy this requirement (on any active track), it would become obvious to everyone that the closures had not been planned at all! This is what Byteasar would like to avoid.

More precisely, on the j-th day (for each $j \leqslant k$) it should be possible to create a grid of replacement lines satisfying the following conditions:

- Each line must start and end its course at some tram loops, making a certain number of runs per hour (the same in both directions), and its route must be a *simple path* [1] because trams cannot just turn back.
- No line can run through a track that Byteasar has stolen since the 1st until the j-th night.
- If the i-th track has not been stolen, then on this day all the lines running through this track must make c_i runs (each way) per hour **in total**.

Of course, no local programmer would be willing to help Byteasar, so he turned to you. Find the correct order of track thefts or state that no such order exists. Observe that the grid of replacement lines for each day can be created independently of the grids for other days, it only needs to meet the conditions described.

Note: It is possible that the values c_i measured by Byteasar do not describe a correct grid of lines (after all, anyone can make a mistake while counting the passing trams) – in such a situation, you also need to inform Byteasar about his error. In other words, you need to verify the conditions described above also on the day number $j = 0$, i.e. before the first theft.

[1] This means that if the route of a given line leads along the track from the intersection u to the intersection v (and it does not end in v), then it must further lead through a track from v to an intersection other than u.

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 15\,000$). The descriptions of the test cases follow.

The first line of the test case contains two integers n, p ($2 \leqslant p \leqslant n \leqslant 500\,000$) denoting the number of intersections and tram loops, respectively.

The intersections are numbered from 1 to n, while the tracks are numbered from 1 to $n-1$.

The next line contains p different integers p_i ($1 \leqslant p_i \leqslant n$) in ascending order, describing the intersections at which the loops are located. You can assume that if there is only one track leaving an intersection, then there is definitely a loop at that intersection (but loops can also be located at other intersections).

The following $n-1$ lines describe the tracks. Each of them contains three integers u_i, v_i, c_i ($1 \leqslant u_i, v_i \leqslant n$, $u_i \neq v_i$, $1 \leqslant c_i \leqslant 10^9$) meaning that track number i connects the intersections u_i and v_i, and according to Byteasar's measurements c_i trams per hour pass this track in each direction.

The next line contains one integer k ($1 \leqslant k \leqslant n-1$).

The last line of the test case contains k different integers k_i ($1 \leqslant k_i \leqslant n-1$) in ascending order, representing the tracks Byteasar would like to steal.

Sum of n among all test cases does not exceed 2 000 000.

Output

For each data set, if there exists an order of track thefts fulfilling the conditions described, on the first line of the output write the single word `TAK`. In the next line write k integers r_i ($1 \leqslant r_i \leqslant n-1$), where r_i is the identifier of the track that Byteasar should dismantle during the i-th night. Each of the k tracks chosen by Byteasar should appear on this list exactly once. If there is more than one correct answer, you can output any of them.

If the order you are looking for does not exist, or **if the input values c_i do not describe any valid grid of lines** even before the thefts begin [2], print the word `NIE`.

[2] In such a situation, Byteasar will have to repeat his research in order to collect correct data on the frequencies of the trams.

Example

For an example input:	A possible correct output is:
2 7 5 1 2 3 4 6 7 1 3 7 2 4 7 3 4 7 4 3 5 3 1 5 6 1 2 2 3 7 5 1 2 3 4 6 7 1 3 7 2 4 7 3 4 7 4 3 5 3 1 5 6 1 4 2 3 5 6	TAK 2 3 NIE

Explanation

In the first test case, the theft of first track 2 and then track 3 will allow city officials to develop the correct network of replacement lines after each change:

- Before the first theft, the following grid of lines fulfils the objectives of the task: line going through intersections 1-7-2 performing 3 runs [per hour, each way], line 2-7-3-5-6 performing 1 run and line 3-7-4 doing 3 runs.
- After the theft of track number 2 (connecting vertices 2 and 7), it is possible to develop the following grid of substitute lines: line 1-7-3 performing 1 run, line 1-7-3-5-6 performing 1 run, line 1-7-4 performing 1 run, line 3-7-4 performing 2 runs.
- After the theft of tracks number 2 (connecting vertices 2 and 7) and number 3 (connecting vertices 3 and 7), it is possible to develop the following grid of substitute lines: line 1-7-4 performing 3 runs, line 3-5-6 performing 1 run.

The opposite order (3 2) is also a correct answer.

In the second test case, Byteasar would also like to steal tracks 5 and 6. Regardless of which of the two tracks he would dismantle earlier, after this theft it would not be possible to develop a line grid that would properly serve the other track. Therefore, the answer is `NIE`.

Problem M: Minor Evil

Time limit: 15s, memory limit: 1GB.

Evil is Evil. Lesser, greater, middling. Makes no difference. The degree is arbitrary, the definition's blurred... but not for Gebyte. In-depth reflection on moral ambiguities is not easy while being permanently hangover [1], so whenever two people ask Gebyte to kill the other person, in a blink of an eye he knows which choice is the lesser evil.

As a powerful witch, you can see what awaits Gebyte on his trail: each of the following k days he is going meet a pair of people in conflict. You are also able to foresee which person he is going to kill in each encounter. You decided to use this insight for your own advantage: you want to ensure that a few people whom you don't particularly like end up among the victims.

Unfortunately, your magic is not powerful enough to force Gebyte to make a decision he considers the greater evil. However, you are well versed in charms that can drive the hangover away. If you cast such a spell before a given encounter, the Witcher – suddenly having regained extraordinary clarity of reasoning – will start to reflect more deeply on the dilemma he is facing, not killing either person as the result. (In the evening, though, he is sure to visit a local inn, so the next morning he will be back to his usual hangover-self.)

You can cast spells as many times as you like. Are you able to ensure that all your enemies perish at the hand of Gebyte?

Note that a person can participate in multiple conflicts. **Any given encounter only occurs if both people are still alive** – otherwise, nothing happens that day.

Input

The first line of input contains the number of test cases z ($1 \leqslant z \leqslant 1\,000$). The descriptions of the test cases follow.

The first line of the test case contains two integers n, k ($2 \leqslant n \leqslant 10^6$, $1 \leqslant k \leqslant 10^6$), denoting the number of people and the number of meetings on Gebyte's trail, respectively. People are numbered from 1 to n, and meetings from 1 to k.

The following k lines describe the meetings. Each i-th of them contains two integers a_i, b_i ($1 \leqslant a_i, b_i \leqslant n$, $a_i \neq b_i$), meaning that on the i-th day Gebyte will encounter a conflict between persons a_i and b_i and he will decide that killing the person b_i is the lesser evil.

The next line contains one integer s ($1 \leqslant s \leqslant n$) – the number of your enemies.

The last line of the test case contains s distinct integers s_j ($1 \leqslant s_j \leqslant n$) in ascending order. All the people appearing on this list should die at Gebyte's hand. All the other people **may or may not** survive.

Sums of values of n and k over all test cases do not exceed 4 000 000 each.

Remark

If Gebyte meets the same pair of people more than once, then you **cannot** assume that his choice is the same every time. Perhaps in the meantime he has gained some new information that altered his judgement. In other words, it is possible that $\exists_{i,j}(a_i = b_j \wedge a_j = b_i)$.

[1] The earlier tale of Gebyte fighting hangover and monsters alike can be found in the problem "Gebyte's Grind" from AMPPZ 2021.

Output

For each test case, print the solution in the following format.

If a solution exists, in the first line of output write a single word `TAK`. In the next line, write a string composed of k letters `T` and `N`. If on the i-th day you want to let Gebyte kill person b_i, then the i-th letter of the sequence should be `T`. On the other hand, if you want Gebyte not to kill anyone on the i-th day, then the i-th letter of the sequence should be `N`. (If in your solution Gebyte has already killed person a_i and/or person b_i, then it does not matter whether you print `T` or `N` on the i-th position – the conflict will not take place, and Gebyte will not kill anyone on that day.) Your solution will be accepted if none of your s enemies listed on the input survives until the end.

If there is no correct solution, in a single line write `NIE`.

Example

For an example input:	A possible correct output is:
2 5 6 1 2 2 1 2 5 2 3 2 4 4 2 3 1 2 3 3 2 1 2 2 3 2 2 3	TAK NTTTNT NIE

Explanation

In the first test case, if we allowed Gebyte to choose the lesser evil each time, then on the first day he would kill person number 2 and none of the subsequent conflicts would take place. Therefore, persons 1, 3, 4 and 5 would survive.

On the other hand, if we stop Gebyte on the first and fifth days (the `NTTTNT` solution), then Gebyte will kill person 1 on the second day, person 5 on the third day, person 3 on the fourth day, and person 2 on the sixth day. Therefore, only person 4 will survive. Another correct solution is `NTNTNT`, which lets persons 4 and 5 survive.

In the second test case, it is impossible to get both person 2 and person 3 killed.