Problem A. Album of Numbers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	128 mebibytes

Johnny is a collector of numbers — he has a large album with them, which he keeps updating by inserting new numbers and deleting old ones.

This album is Johnny's treasure most prized, so Johnny wants to be sure that no one will ever replace his numbers with different ones. Therefore, he maintains a checksum for his album, determined as follows. He considers all distinct nonempty subsets of the numbers in his album, in each such subset he determines the minimum, and finally he calculates the arithmetic mean of these minima. As any number may appear multiple times in Johnny's album, the "subsets" are multisets, i.e., each number may appear in any subset multiple times (but no more than in the whole album). Two such multisets are equal if and only if all numbers have the same multiplicity in both. For instance, an album $\{1, 2, 2, 5\}$ has the following 11 distinct subsets: $\{1\}$, $\{1, 2\}$, $\{1, 2, 2\}$, $\{1, 2, 2, 5\}$, $\{1, 2, 5\}$, $\{1, 5\}$, $\{2\}$, $\{2, 2\}$, $\{2, 2, 5\}$, $\{2, 5\}$ and $\{5\}$.

Unfortunately, Johnny's album changes quite often, making maintaining the checksum manually rather tedious. Help him out by writing a program that will: read the number of updates to be applied to the album and descriptions of these operations, determines the checksum after each successive operation, and prints the checksums to the standard output.

Input

In the first line of input, there is a single positive integer n ($1 \le n \le 250\,000$) that specifies the number of updates to the album. The n lines that follow specify these operations, one per line, as follows. Each update is given as a `+` or `-` sign, a single space, and a number a_i ($1 \le a_i \le 1\,000\,000\,000$). These describe inserting the number a_i to the album and removing a appearance of the number a_i from the album, respectively.

You may assume that the album is initially empty, afterwards it is always nonempty, and that only numbers present in the album are ever deleted.

Output

Your program should print exactly n lines to the output. The i-th such line should contain the checksum after the i-th update, as prescribed by Johnny's formula.

An answer is deemed correct if, for each checksum, the absolute or relative error with respect to the correct value does not exceed 10^{-6}.

Examples

standard input	standard output
6 + 10 + 13 - 10 + 7 + 2 + 5	10.000000000 11.000000000 13.000000000 9.000000000 5.000000000 4.200000000
4 + 1 + 2 + 2 + 5	1.000000000 1.333333333 1.400000000 1.727272727

Note

In Sample 1, after all the insertions, the album's contents are $\{1, 2, 2, 5\}$. At that point, Johnny considers the following 11 subsets: $\{1\}$, $\{1, 2\}$, $\{1, 2, 2\}$, $\{1, 2, 2, 5\}$, $\{1, 2, 5\}$, $\{1, 5\}$, $\{2\}$, $\{2, 2\}$, $\{2, 2, 5\}$, $\{2, 5\}$ and $\{5\}$. The respective minima of these subsets are: $1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 5$. Hence, the corresponding checksum is $\frac{19}{11} \approx 1.727272727$.

Problem B. Well Off

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	128 mebibytes

Thanks to his hard work, John, formerly Johnny, has become a member of the respectable Well Off Club. The club has n members, numbered from 1 to n; the i-th club member has a fortune worth x_i. These can be arbitrary numbers, including negative ones. The reason for this is that the club membership is for life, i.e., neither impoverishment nor even bankruptcy results in an ousting.

The club members take pleasure in comparing their asset values, but they shun telling the number directly; it is their custom to engage in a conversation that lets them establish that their fortunes x_i and x_j satisfy one of the inequalities $x_i + x_j > 0$, $x_i - x_j > 0$, $-x_i + x_j > 0$, or $-x_i - x_j > 0$.

Johnny (over-)heard many such inequalities today at the club, but he suspects that some of the members are lying. Help him determine if all the inequalities could possibly be true, i.e., if they can be simultaneously satisfied by some set of fortunes. Write a program that: reads the inequalities heard by Johnny, determines if they are satisfiable, and prints the answer to the standard output.

Input

In the first line of input there are two integers, n and m ($1 \leq n \leq 100\,000$, $1 \leq m \leq 500\,000$), separated by a single space. These specify the number of club members and the number of inequalities relating their fortunes, respectively.

Each of the m lines that follow describes one inequality, encoded by a `+` or `-` sign, an integer i ($1 \leq i \leq n$), another `+` or `-` sign, and an integer j ($1 \leq j \leq n$), separated by single spaces. These correspond to the inequality $\pm x_i \pm x_j > 0$ (with the signs as preceding i and j in the description). It might happen that $i = j$.

Output

The first and only line of the output should contain a single word: "`TAK`" if the system of inequalities is satisfiable, and "`NIE`" otherwise. (Naturally, these are the Polish words for *yes* and *no*.)

Examples

standard input	standard output

`3 3`
`+ 1 - 2`
`- 3 + 1`
`+ 2 - 3`
	`TAK`

`3 3`
`+ 1 - 2`
`+ 3 - 1`
`+ 2 - 3`
	`NIE`

Note

In Sample 1, system of inequalities is satisfiable, e.g. by $x_1 = 3$, $x_2 = 2$, $x_3 = 1$.

In Sample 2, system of inequalities is not satisfiable.

Problem C. Accurate Shots (8Mb TL!)

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	8 mebibytes

Sophie plans to take over control of the world. Currently a central computer holds it, but its software has critical bugs. All that Sophie needs to do to overthrow the computer's rule is to set the value of its variable z to any integer divisible by m.

At the moment, the variable z is set to an integer n, encoded in binary without leading zeros. In her basement, Sophie has constructed an ion cannon, whose shot can flip a bit of Sophie's choice in the computer's memory. Naturally, she is going to target the register holding the variable z, and in fact can choose to flip any of its bits with an accurate shot. The cannon can only flip existing bits in the binary representation (which has no leading zeros!), and cannot be used to insert new bits anywhere in the representation, including either of its ends. However, any leading ones can be flipped to zeros. Sophie can shoot as many times as she pleases, but each shot increases the chance of her plot being detected. Thus, she wants to minimize the number of shots.

Help Sophie in seizing power. Write a program that will read the current value n stored in the variable z and a number m from the standard input, determines the minimum number of shots required to overthrow the computer, the number of distinct integers divisible by m that Sophie can transform n into with the minimum number of shots, and the smallest such integer and prints those three numbers.

Input

In the first and only line of input there are two positive integers separated by a single space: n and m ($1 \leq n, m \leq 10^{15}$). The former is the initial value of the variable z, whereas the latter is the one that the final value of z should be a multiple of.

Output

Three numbers should be printed to the standard output, separated by single spaces: the minimum number of shots to overthrow the computer's rule, the number of distinct n' that ensure overthrowing the computer with the minimum number of shots fired, and the minimum one out of those n'.

Examples

standard input	standard output
`30 7`	`1 2 14`
`69 41`	`3 1 0`

Note

In first example, the variable has value 30, i.e. $11110_{(2)}$ in binary. To make it a multiple of 7, it suffices to shoot once. There are two ways to attain that: by transforming to either $\underline{0}1110_{(2)} = 14$ or $111\underline{0}0_{(2)} = 28$. The first value is the minimum.

In second example, the variable has value $69 = 1000101_{(2)}$. The only way to make it a multiple of 41 is by transforming it to $0 = \underline{0}000\underline{0}0\underline{0}_{(2)}$, which requires three shots.

Problem D. Prom

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	128 mebibytes

Johnny is getting ready for prom, which traditionally begins with a polonnaise dance. Any mixed pair (boy and girl) can lead the polonnaise, as long as they do not differ too much in height. More precisely, the difference in their heights cannot exceed k byteometers. Jimmy wants to find out how many ways there are to choose the leading pair.

Write a program which reads the heights of all girls and boys, computes the number of possible leading pairs, and writes the result to standard output.

Input

The first line of input contains three space-separated integers n, m and k ($1 \leq n, m \leq 250\,000$, $0 \leq k \leq 1\,000\,000\,000$), denoting the number of girls, the number of boys and the maximum possible difference of height in the leading pair, respectively.

The second line of input contains a sequence of n space-separated integers a_i ($1 \leq a_i \leq 1\,000\,000\,000$): the heights of girls, given in byteometers.

The third line of input contains a sequence of m space-separated integers b_i ($1 \leq b_i \leq 1\,000\,000\,000$): the heights of boys, given in byteometers.

Output

The first and only line of output should contain a single integer — the number of possible leading pairs.

Example

standard input	standard output

`4 5 5`
`15 2 5 7`
`1 5 10 15 1`
	`11`

Note

There are 11 possible leading pairs: $(15, 10)$, $(15, 15)$, $(1, 1)$, $(1, 5)$, $(1, 1)$, $(5, 1)$, $(5, 5)$, $(5, 10)$, $(5, 1)$, $(7, 5)$ and $(7, 10)$.

Problem E. Impressive Graphs

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

One of Sophie's tasks as a sales department employee is creating graphs of sales volumes. She has access to all the data, namely the integer sales volumes from the last n months. Having glanced at the figures already, she knows that these numbers are pairwise different. The snag is that Sophie's boss expects impressive graphs, and not one but k of them! A graph is impressive if the sequence of sales volumes in it is increasing. While Sophie can create empty graphs, she cannot get too creative. Namely, she has to conform with the following rules:

- any month's sales volume can be used in at most one graph;
- each graph must be in chronological order, i.e., for each pair of months in the graph, the earlier of the two months (and its corresponding sales volume) comes first.

The more sales volumes are used in a set of graphs, the more impressive it is.

Help Sophie in finding the most impressive set of k graphs. That is, write a program that will: read the number of sales volumes, the number of sales graphs to be created, and the monthly sales volumes themselves, determines the most impressive set of sales graphs, and prints it to the standard output. If the most impressive set of sales graphs is not unique, your program may choose any of them.

Input

In the first line of input, there are two integers n and k ($1 \le n \le 200\,000$, $1 \le k \le 20$), separated by a single space. These specify the number of sales volumes and the number of graphs, respectively. In the second (and last) line of input, there are n pairwise different integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$), separated by single spaces. These are the sales volumes from successive months.

Output

Your program should print exactly $k+1$ lines. The first line should contain a single integer: the maximum number of sales volumes that can be used in a set of k impressive graphs. The next k lines should describe those graphs, one per line. A single description should consist of an integer ℓ ($\ell \ge 0$) — the number of sales volumes in the graph — followed by these volumes, i.e., ℓ integers $a_{i_1}, a_{i_2}, \ldots, a_{i_\ell}$ such that $i_1 < i_2 < \ldots < i_\ell$ and $a_{i_1} < a_{i_2} < \ldots < a_{i_\ell}$. All these numbers are separated by single spaces.

Examples

standard input	standard output

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

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

Note

In first sample, the plots in the most impressive set of plots both have 2 sales volumes: $1, 2$ in one plot and $4, 5$ in the other.

Problem F. Pen

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	128 mebibytes

Johnny loves order and is very upset by untidy parentheses. Lately he saw a string "`){(`

", which annoyed him since it was not a correct bracket sequence. Fortunately, Johnny had a pen in his pocket and he could add '`(`' at the beginning and '`)}`' at the end. Then he could sleep soundly again, since the string "`(){(`

`)}` is a correct bracket sequence.

Johnny took on repairing bracket sequences to be his life's mission. He will do this by adding new parentheses at the beginning and at the end of sequences he finds, so as to make them correct. Since his pen is running out of ink, Johnny would like to add the least possible number of brackets which will accomplish this. Help him! Write a program which will read a bracket sequence and either compute a shortest possible corrected bracket sequence for it and write it to standard output, or say that there is no way to correct it.

Correct bracket sequences are defined recursively as follows:

- the empty sequence is a correct bracket sequence,
- if S and T are correct bracket sequences, then their concatenation ST is also a correct bracket sequence,
- if S is a correct bracket sequence, then (S), $[S]$ and $\{S\}$ are all also correct bracket sequences.

Input

The first and only line of input contains a nonempty sequence consisting of characters `(`, `)`, `[`, `]`, `{`, `}` and having length at most 1 000 000; this is the bracket sequence that Johnny wants corrected.

Output

The output should contain a corrected input sequence. If there are many possible ways to correct the input sequence that have the same minimum length, output any of them.

If it is impossible to correct the input sequence, output the word `NIE`.

Examples

standard input	standard output

`){(`
 	
	`(){(`
	
	`)}`

`(`
 	
`}`
	`NIE`

Note

First example is the one from the problem statement.

In the second example, sequence cannot be repaired.

Problem G. Board Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 mebibytes

Today, Sophie received a new board game, whose key element is establishing trade links between cities. There are n cities with coordinates $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ on the board, no two of them sharing both coordinates. Each city produces a good of one type. Trade links can be established only between cities producing goods of different kinds. For establishing such a link between a pair of cities with coordinates (x_i, y_i) and (x_j, y_j), the player receives

$$(x_i - x_j)^2 + (y_i - y_j)^2$$

points.

Sophie would like to know which trade link would grant her the most points. Write a program that: reads the board description, determines the trade link that maximizes the number of points, and prints this number to the standard output.

Input

In the first line of input, there is an integer n ($2 \le n \le 250\,000$) that specifies the number of cities on the board. Each of the n lines that follow describes a single city. A city's description consists of a triple of integers x_i, y_i, t_i ($-1\,000\,000\,000 \le x_i, y_i \le 1\,000\,000\,000$, $1 \le t_i \le n$) separated by single spaces; the numbers x_i, y_i are the i-th city's coordinates, whereas t_i is the type of good that it produces.

More than one type of good is produced overall, and no two cities share both coordinates.

Output

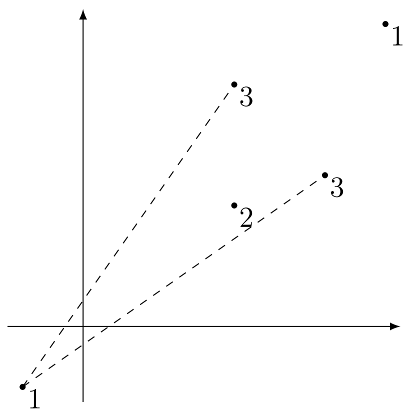
The first and only line of output should contain the maximum number of points that Sophie can receive for a single trade link.

Example

standard input	standard output

5
5 4 2
-2 -2 1
10 10 1
8 5 3
5 8 3
	149

Note



The trade link that maximize the number of points links the cities with coordinates $(-2, -2)$ and $(8, 5)$. This link yields $149 = 10^2 + 7^2$ points. There is also another link that yields the same amount of points: between the cities with coordinates $(-2, -2)$ and $(5, 8)$.

Problem H. Scouts

Input file:	`standard input`
Output file:	`standard output`
Time limit:	9 seconds
Memory limit:	256 mebibytes

Johnny's boy scout team is facing a change of its command structure. The team has n scouts, numbered from 1 to n. The command structure is defined as follows:

- The command structure has a commander.
- Scouts with numbers lower than their commander's comprise the first subteam, and those with numbers higher than their commander's comprise the second subteam; one or both subteams may be empty.
- Nonempty subteams have their own command structures.
- The commander of a team is the supervisor of commanders of its subteams.

The command structure will be used to convey one, very important message: the message is communicated from outside to one of the scouts, who reads it and then passes it on to his supervisor; this procedure is repeated until the message reaches the supervisor of the whole team. Different scouts take different amounts of time to read the message: the i-th scout takes a_i time. The time it takes to handle the message is the sum of times it takes for the message to be read by all the scouts who passed it (including the commander of the entire team).

Unfortunately, it is not known which scout is going to receive the message. Johnny should reorganize the command structure of the team so as to minimize the maximum possible time of handling the message. Help him. Write a program which reads the number of scouts and the time it takes each of them to read the message, computes the minimum possible maximum time of handling the message and writes the result to standard output.

Input

The first line of input contains one integer n ($1 \le n \le 2\,000$) — the number of scouts. The second line of input contains a description of the team. It consists of n space-separated integers a_i ($1 \le a_i \le 1\,000\,000\,000$), which are the times of reading the message by the respective scouts.

Output

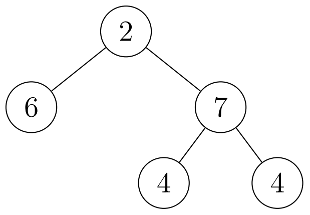
Your program should write one integer — the minimum possible maximum time it can take to handle the message.

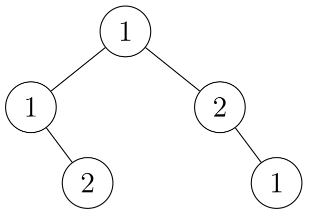
Example

standard input	standard output
5 6 2 4 7 4	13
5 1 2 1 2 1	4

Note

In sample 1, optimum command structure is shown below at the left. The numbers in vertices correspond to reading times for the respective scouts. The maximum time of handling the message is 13.





In sample 2, an optimum command structure is shown above at the right. The numbers in vertices correspond to reading times for the respective scouts. The maximum time of handling the message is 4.

Problem I. Insects

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	128 mebibytes

Sophie works in a company that produces insect baits. Each bait has three ingredients: attractor, feed, and poison. Moreover, for each out of n insect types, there is a unique such triplet that eliminates it. To streamline the process, the attractors, feeds, and poisons are numbered.

Sophie is designing a brand new insect bait. It need not kill all the insects, and is rather supposed to maximize the company's profits. The productions cost depends on the number of different attractors, feeds, and poisons that the bait contains. Specifically, it is the sum of all their unit costs. Conveniently, the unit cost of all attractors is the same, and the same holds for the feeds and for the poisons. When it comes to market price, it is the company's policy to charge per insect type that the bait can kill. Specifically, the price of the bait is this number multiplied by p, the so called bang-per-bug factor. The profit per unit of such insect bait is the difference between its price and its production cost.

Help Sophie design the most profitable insect bait. Write a program that will read the insect number, the bang-per-bug factor, the costs of attractors, feeds and poisons, determines the maximum possible profit per unit of bait, and prints this number to the standard output.

Input

In the first line of input, the integers n, p, c_a, c_k, c_t ($1 \leq n, p, c_a, c_k, c_t \leq 1000$) are given, separated by single spaces. These are, respectively: the number of insect types, the bang-per-bug factor, and the unit costs of attractors, feeds, and poisons. The n lines that follow describe the insect types, one per line. A single description consists of three integers a_i, k_i, t_i ($0 \leq a_i, k_i, t_i < 256$), separated by single spaces; these indicate that the i-th insect can be eliminated by combining the attractor a_i, the feed k_i, and the poison t_i.

It is guaranteed that the triplets (a_i, k_i, t_i) given on input are pairwise different.

Output

Your program should print a single integer: the maximum possible profit per unit of an insect bait.

Example

standard input	standard output

5 10 3 10 1
127 127 127
0 0 127
255 127 0
127 127 0
64 64 64
	12

Note

The bait should consist of the attractors 127 and 255 (which cost $2 \cdot 3 = 6$), the feed 127 (which costs $1 \cdot 10 = 10$), and the poisons 0 and 127 (which cost $2 \cdot 1 = 2$), with the total production cost of 18. This bait kills the insects of types 1, 3, and 4, so its price is 30. Therefore, the profit per unit of this bait is 12.

Problem J. Caves

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	256 mebibytes

Sophie is a treasure seeker. Right now she's looking for a treasure hidden in a system of n caves, which are numbered from 1 to n. She knows that the cave system contains exactly one treasure in one of the caves. Sophie has done some measurements using a georadar; this has not told her where the treasure is, but she has obtained, for each cave, a probability that the treasure is in that cave.

Sophie will continue her search using a precise remote probe: after the probe enters a cave, it instantly finds out whether the cave contains the treasure. The probe starts its search in cave number 1, and it is known that this cave does not contain the treasure. Sophie knows (from her georadar measurements) which pairs of caves are connected using bidirectional tunnels which the probe can traverse. Such a traversal takes the probe one minute. Unfortunately, there is no guarantee that the probe can make its way from cave 1 to every other cave (even in many steps).

As an alternative to traversing the tunnels, Sophie can choose to abandon her probe and insert a new probe into the system, which takes t minutes. If she does so, communication with the old probe is lost and Sophie can no longer use it. Under these field conditions, inserting a new probe is very imprecise: it lands with uniform $\frac{1}{n}$ probability in any of the n caves. Sophie instantly knows where the probe has landed. Sophie has infinitely many probes at her disposal.

Help Sophie to find the treasure fast. Find a program which reads the size of the cave system, a description of the tunnels, the time t of introducing a new probe, as well as the probability distribution of where the treasure is, computes the smallest (over all Sophie's strategies) expected time of finding the treasure, and writes the computed value to standard output. By finding the treasure we mean reaching the treasure cell with a probe.

Input

The first line of input contains three space-separated integers n, m, t ($2 \le n \le 20$, $0 \le m \le \frac{n(n+1)}{2}$, $1 \le t \le 100$). They denote the number of caves, the number of tunnels and the time needed to insert a new probe. The second line contains n space-separated real numbers $p_1, p_2, \ldots, p_n$ ($p_1 = 0$, $0 \le p_i \le 1$, $\sum_{i=1}^{n} p_i = 1$); the number p_i is the probability that the treasure is in the i-th cave. These values are given with two decimal digits of precision. The next m lines describe the tunnels. The description of each tunnel consists of two space-separated integers i, j ($1 \le i, j \le n$ and $i \ne j$), which denote a tunnel between caves i and j that the probe can traverse in either direction. You can assume that all tunnels are distinct.

Output

The first and only line of output should contain a real number — the minimum possible expected time of finding the treasure.

The answer is considered correct if its absolute or relative error is at most 10^{-6}.

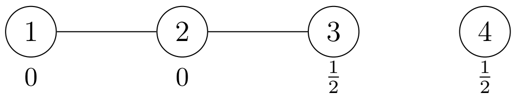
Example

standard input	standard output

4 2 10
0.00 0.00 0.50 0.50
1 2
2 3
	22.000000000

6 5 2
0.00 0.00 0.00 0.00 0.50 0.50
1 2
2 3
3 4
4 5
5 6
	4.100000000

Note



In sample 1, Sophie should first move the probe to cave 2 and then to cave 3. With probability $\frac{1}{2}$ it has found the treasure in time 2. If it has not, she should now start inserting new probes, until one gets inserted into the cave 4. The expected number of insertions is 4, so the expected total time in this case is 42. On average, it takes Sophie 22 seconds to find the treasure. (Note that after 2 minutes, Sophie already knows where the treasure is, but she still needs to reach it with a probe.)

Problem K. Blocks

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Sophie is playing with n cuboid blocks. The i-th of her blocks has size $1 \times 1 \times i$.

Her favorite game is arranging all the blocks in parallel in a row in such a way that they all stand on their 1×1 faces (i.e., these are their bases). After arranging the blocks, Sophie stands at a large distance and watches the block arrangement from both ends of the row, which we shall refer to as the left end and the right end. She walks so far away that each block obstructs visibility of all lower blocks behind it. Sophie is satisfied if she sees exactly ℓ blocks from the left end and p blocks from the right end.

She is wondering how many satisfying block arrangements exist. Help Sophie by writing a program that will: read the number of blocks and the numbers of blocks that she wants to see from the left and the right end, determines the number of satisfying arrangements, and prints the result to the standard output.

Input

In the first line of input, there is the number of data sets m ($1 \le m \le 100\,000$). Each of the m lines that follow describes a single data set. Each description consists of three integers n, ℓ, p ($1 \le n \le 50\,000$, $1 \le \ell, p \le 100$), separated by single spaces. These specify the number of given blocks, the number of blocks Sophie wants to see from the left end, and the number of blocks she wants to see from the right end.

Output

For each data set, your program should print a single line with only a single integer: the remainder modulo $10^9 + 7$ of the number of satisfying block arrangements.

Example

standard input	standard output

2
4 3 2
5 3 3
	3
	6

Note

In the first data set, Sophie is satisfied with the following block arrangements: $(1, 2, 4, 3)$, $(1, 3, 4, 2)$, $(2, 3, 4, 1)$. In the $(1, 2, 4, 3)$ arrangement, the blocks 1, 2, and 4 are visible from the left, whereas 4 and 3 are visible from the right.

Problem L. Postman

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	256 mebibytes

Johnny has just become a postman. Since he's new in the company, he got the worst assignment — delivering urgent mail to a new neighborhood. The urgency is no joke: the delivery deadline is in exactly one hour. Moreover, just recently the mail company has introduced a new customer-friendly set of rules: for each letter delivered past its deadline, the postal company issues a compensation of 1 bythaler for every hour of delay.

The neighborhood consists of n houses, numbered from 1 to n. Johnny has to deliver one letter to each house. The houses are connected by $n - 1$ two-way streets in such a way that, for every pair of houses, it is possible to get from one to the other by following the streets.

Johnny has to deliver the letters as soon possible, but due to company driving regulations, the company car will only get him to one house of his choosing, and afterward Johnny has to walk. The ride takes exactly one hour, so he will deliver exactly one letter on time. Walking each street also takes exactly one hour. The time to drop the letter in the mailbox is negligible. Needless to say, Johnny has to deliver all the letters.

Well-aware though he is that timely delivery is impossible, Johny would still like to at least minimize the amount of compensation that the postal company will have to pay out. Help him by writing a program that: reads the number of houses in the neighborhood and the description of the streets connecting them, determines the minimum compensation for late deliveries, and prints this number to the standard output.

Input

The first line of input contains a single integer n ($1 \le n \le 1\,000\,000$) that specifies the number of houses in the neighborhood. Each of the $n - 1$ lines that follow describes one street in the neighborhood. Such description consists of two integers a and b ($1 \le a, b \le n$), separated by a single space; these are the numbers of houses directly linked by a two-way street.

Output

One number is to be printed — the minimum compensation (in bythalers) that the postal company will have to pay.

Example

standard input	standard output

5
1 2
2 3
2 4
2 5
	13

4
1 2
1 3
3 4
	6

Note

Johnny can be dropped off at the house no. 1, deliver the letter (on time, so 0 compensation), then walk to the house no. 2, deliver the letter (at compensation of 1), then walk to the house no. 3 and deliver the letter (at compensation of 2). Next, he can return to the house no. 2, then walk further to the house no. 4 and deliver a letter there (at compensation of 4), again return to the house no. 2, walk to the house no. 5 and deliver the final letter there (at compensation of 6).