ABB

`abb.c`, `abb.cpp`, `Abb.java`, `abb.py`

Fernando was hired by the University of Waterloo to finish a development project the university started some time ago. Outside the campus, the university wanted to build its representative bungalow street for important foreign visitors and collaborators.

Currently, the street is built only partially, it begins at the lake shore and continues into the forests, where it currently ends. Fernando's task is to complete the street at its forest end by building more bungalows there. All existing bungalows stand on one side of the street and the new ones should be built on the same side. The bungalows are of various types and painted in various colors.

The whole disposition of the street looks a bit chaotic to Fernando. He is afraid that it will look even more chaotic when he adds new bungalows of his own design. To counterbalance the chaos of all bungalow shapes, he wants to add some order to the arrangement by choosing suitable colors for the new bungalows. When the project is finished, the whole sequence of bungalow colors will be symmetric, that is, the sequence of colors is the same when observed from either end of the street.

Among other questions, Fernando wonders what is the minimum number of new bungalows he needs to build and paint appropriately to complete the project while respecting his self-imposed bungalow color constraint.

Input Specification

The first line contains one integer N ($1 \le N \le 4 \cdot 10^5$), the number of existing bungalows in the street. The next line describes the sequence of colors of the existing bungalows, from the beginning of the street at the lake. The line contains one string composed of N lowercase letters ("`a`" through "`z`"), where different letters represent different colors.

Output Specification

Output the minimum number of bungalows which must be added to the forest end of the street and painted appropriately to satisfy Fernando's color symmetry demand.

Sample Input 1

```
3
abb
```

Output for Sample Input 1

```
1
```

Sample Input 2

```
12
recakjenecep
```

Output for Sample Input 2

```
11
```

Sample Input 3

```
15
murderforajarof
```

Output for Sample Input 3

```
6
```

Be Geeks!

`begeeks.c`, `begeeks.cpp`, `Begeeks.java`, `begeeks.py`

The musical band Be Geeks! got its name by no accident, as all the members are genuine math geeks. Among others, they love examining various properties of number sequences. Let's see an example of their subject of interest.

Let A be a nonempty sequence of positive integers, $A = (a_1, a_2, ..., a_N)$.
Let $G(i, j) = \gcd(a_i, a_{i+1}, \ldots, a_j)$, where $1 \le i \le j \le N$.
Let $M(i, j) = \max(a_i, a_{i+1}, \ldots, a_j)$, where $1 \le i \le j \le N$.
Let $P(i, j) = G(i, j) \cdot M(i, j)$, where $1 \le i \le j \le N$.
Let $F(A) = \sum P(i, j)$ over all pairs of integers $1 \le i \le j \le N$.

The function gcd stands for the greatest common divisor of the given values. The greatest common divisor of a nonempty sequence of integers is the biggest integer which divides each integer in the sequence evenly.

Input Specification

The first line contains one integer N ($1 \le N \le 2 \cdot 10^5$). The next line contains N integers $a_1, a_2, \ldots, a_N$ ($1 \le a_i \le 10^9$).

Output Specification

Print the value of $F(A)$ modulo 1 000 000 007.

Sample Input 1

```
4
1 2 3 4
```

Output for Sample Input 1

```
50
```

Sample Input 2

```
5
2 4 6 12 3
```

Output for Sample Input 2

```
457
```

Bob in Wonderland

`bob.c`, `bob.cpp`, `Bob.java`, `bob.py`

A chain, as everybody knows, is made of connected links. Typically, all links are of the same shape and size. Bob is a blacksmith apprentice, and he is making his own first iridium chain. He follows the traditional formula of chain-making. It says:

- If there is no chain yet, make a link and it will be a piece of your chain.
- If there is a piece of chain, make another link and connect it to one other link in the piece of chain you already have.

Bob made the first link. Then, each time he made another link, he connected it to some other link in his piece of chain, exactly as the formula told him to do.

When he finished, he realized that the object he created did not resemble a usual chain at all. In an effort to straighten the chain, he repeatedly took two links which seemed to be at the ends of the chain and tried to pull them apart as far as he could. But there were some more pieces of the "chain" dangling down from the straightened piece at various places.

It was obvious to Bob that his work is not finished yet and he decided to call the object he produced the unfinished chain. After some more pondering, Bob came to a conclusion that he has to break some links and reconnect them to the rest of the unfinished chain more cautiously to obtain a straight chain he aims to produce. In a straight chain, each link is connected to at most two other links and a straight chain cannot be separated into more pieces without breaking a link.

Being now more careful, Bob is going to progress in simple steps. In one step he will choose a link, say A, connected to another link, say B, in the unfinished chain. He will then break A, disconnect it from B and reconnect A to some other link, say C, in the unfinished chain. If there are more links other than B originally connected to A, Bob will keep them connected to A during the whole step.

What is the minimum number of steps Bob has to perform to get a straight chain?

Input Specification

The first line contains one integer N ($1 \le N \le 3 \cdot 10^5$), the number of links in the unfinished chain. The links are labeled $1, 2, \ldots, N$. Each of the next $N-1$ lines has two integers denoting the labels of two connected links in the unfinished chain. The connections are listed in arbitrary order. The unfinished chain is guaranteed to form only one piece.

Output Specification

Output the minimum number of steps which will turn Bob's unfinished chain into a straight chain.

Sample Input 1

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

Output for Sample Input 1

```
0
```

Sample Input 2

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

Output for Sample Input 2

```
2
```

Sample Input 3

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

Output for Sample Input 3

```
1
```

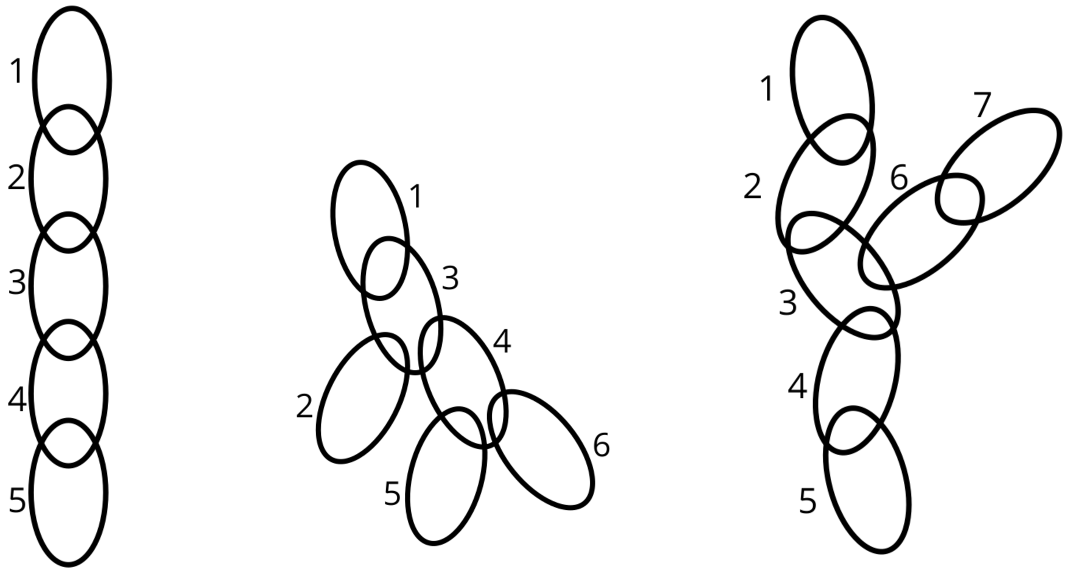

Figure 1: Illustration of Sample Input 1, Sample Input 2 and Sample Input 3

Crimson Sexy Jalapeños

`crimson.c`, `crimson.cpp`, `Crimson.java`, `crimson.py`

The central piece of the Tainted chocolate game is a classic chocolate bar divided into square pieces by a rectangular grid of grooves parallel to the sides of the bar. Some squares have been tainted with extremely bitter substance that makes the tainted square (nearly) indigestible.

The game is played by two players who alternate in their moves. In a valid move, one player is obliged to consume some part of the chocolate bar. It is allowed to divide the current chocolate bar along one of the grooves into two smaller bars and then eat just one of them. The player who consumes a bar containing at least one tainted square loses the game.

The positions of all tainted squares are known at the beginning of the game. All other squares are safe to eat. Each player tries to avoid eating a bar of chocolate containing one or more tainted squares, because when this happens, the player involuntarily makes their personal most disgusted grimace becoming a source of great amusement not only to the other player but also to other people watching the game.

In this problem, you are to write a program to play the Tainted chocolate game. We neglect the part of the code which simulates players grimaces and chocolate consumption, and instead focus only on the winning moves.

A valid move is described by a directional string and a positive integer X. The directional string is one of the four strings "`top`", "`bottom`", "`left`", or "`right`". The description means that the bar is divided by the X-th groove, counted from that side of the currently remaining bar which is specified by the given directional string. The player then consumes the part on that side.

Input Specification

The first line of input contains integers R, C, K ($1 \le R, C \le 10^4; 1 \le K \le 100$). R is the number of rows, C is the number of columns, and K is the number of tainted squares in the chocolate bar. Each of the next K lines contains two integer values A and B ($1 \le A \le R, 1 \le B \le C$), the coordinates of one tainted square. The coordinates of the top-left corner square are $(1, 1)$.

The rest of the input depends on your output. For each of your valid moves, there will appear one input line containing either an opponent's valid move description or the string "`yuck!`" which indicates the opponent lost the game. In the latter case your program should terminate.

Output Specification

After reading the chocolate bar description, you may decide whether you want to start the game. If you want to play second, print one line containing the string "`pass`". This special string may only appear on the very first line of your output.

Then, for each of your moves, print one line containing a description of a valid move. Your program will be given a *Wrong Answer* if it produces anything else than a sequence of valid moves leading to the victory.

After printing each move description, flush the output buffer. For example, you may use `fflush(stdout)` or `cout.flush()` in C++, `System.out.flush()` in Java, or `stdout.flush()` in Python.

Sample Input 1 / Output for Sample Input 1

```
4 6 2
2 3
4 4
                                top 1
right 2
                                left 2
yuck!
```

For the purpose of clarity, the above data are interleaved to illustrate the order of interaction between your program and the system. Note that there will be no empty lines in real data and there must not be any empty lines in your output.

Sample Input 2 / Output for Sample Input 2

```
3 5 1                           pass
2 3                             right 1
left 1                          top 1
left 1                          bottom 1
right 1
yuck!
```

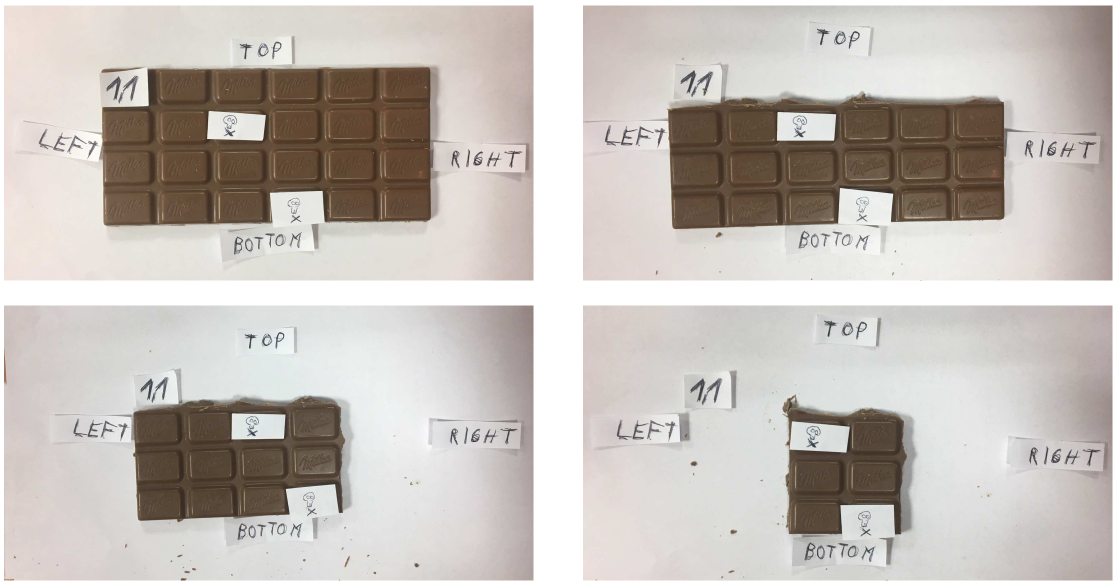

Figure 1: Illustration of the game from Sample Input 1

Deep800080

`deep.c`, `deep.cpp`, `Deep.java`, `deep.py`

Ian, the owner of the boat called Deep800080 is going to make a barbecue on the lake this evening. There is a narrow straight pier running from the shore into the lake. Ian thinks it will be a good idea to make the barbecue somewhere on the pier.

He wants to test his new special barbecue charcoal. When it burns it produces a distinctive thick purple smoke which spreads over the water. The cloud of smoke is guaranteed to spread in a perfect circle around the barbecue site. Eventually, the cloud will reach some maximum radius and will stay unchanged for the rest of the evening until it dissipates later in the night.

There are multiple boats anchored in the lake in various distances from the pier. Ian wants to inform their crews about the barbecue. Being a little lazy, he hopes that the barbecue smoke will do this job for him. Ian hopes that when the cloud reaches a boat, the crew can smell the smoke, and so they immediately know there is a barbecue nearby.

Ian wants to maximize the number of alerted crews and thus he wants to choose a place of the barbecue grill on the pier which would maximize the number of the boats reached by the purple cloud.

You may assume that the boats on the lake are firmly anchored and that they do not move while the cloud is on the lake. Also, once the barbecue site is chosen, its position remains fixed for the whole event.

Input Specification

The first line contains four integers N, R, A, B ($0 \leq N \leq 3 \cdot 10^5; 1 \leq R \leq 10^6$). N is the number of boats on the lake, R is the maximal radius of the purple barbecue cloud. You may assume that the pier is so narrow and so long that it may be perceived as a straight line passing through two distinct points with coordinates $(0, 0)$ and (A, B).

Each of the next N lines contains two integers X and Y describing the coordinates of one boat on the lake. No two boats share the same coordinates.

All coordinates are common Cartesian coordinates in the plane, their absolute value will be at most 10^6.

Output Specification

Output the maximum possible number of boats that can be reached by the purple barbecue cloud.

A boat is considered reached if its position is in the circle formed by the cloud, including its boundary. You may assume that increasing the maximal radius of the smoke cloud by 10^{-3} would not change the solution.

Sample Input 1

```
7 5 0 1
-1 -1
1 -1
0 0
2 3
3 4
10 10
2 12
```

Output for Sample Input 1

```
5
```

Sample Input 2

```
3 1 1 0
0 0
2 0
4 0
```

Output for Sample Input 2

```
2
```

Sample Input 3

```
4 1 1 0
0 0
1 1
1 -1
2 0
```

Output for Sample Input 3

```
4
```

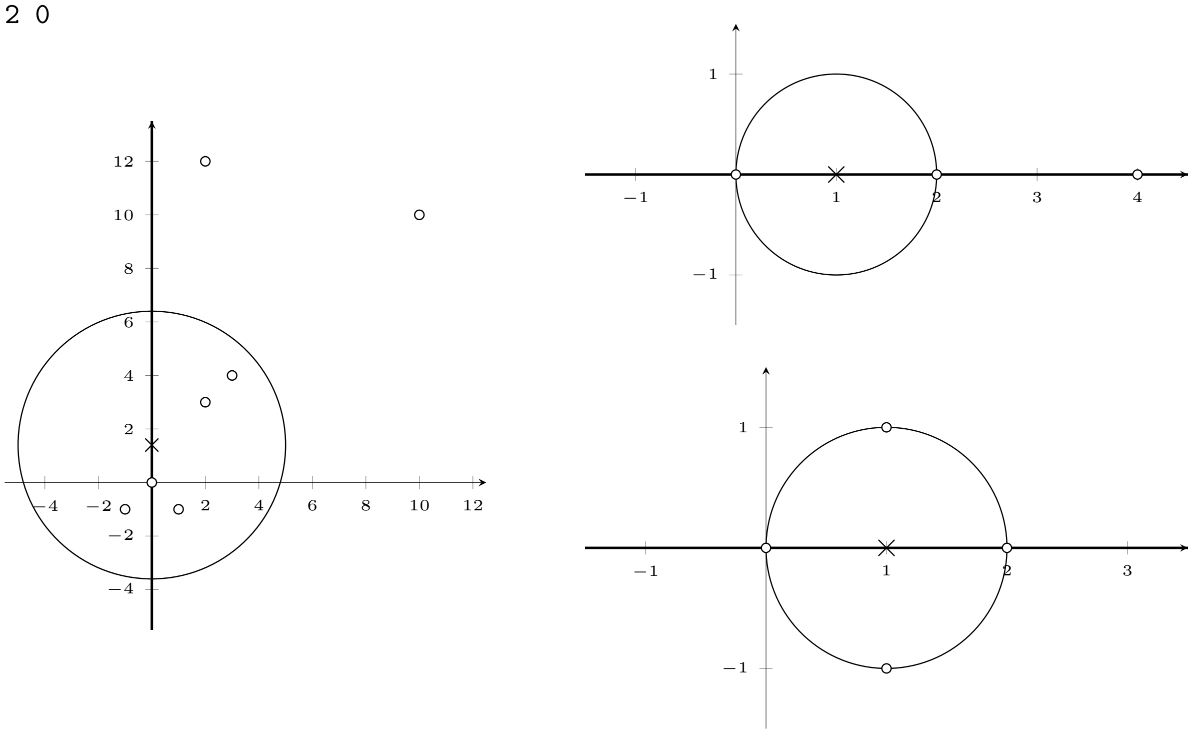

Figure 1: Possible solution of Sample Input 1 (left), Sample Input 2 (top right) and Sample Input 3 (bottom right)

Zeldain Garden

`garden.c`, `garden.cpp`, `Garden.java`, `garden.py`

Boris is the chief executive officer of Rock Anywhere Transport (RAT) company which specializes in supporting music industry. In particular, they provide discount transport for many popular rock bands. This time Boris has to move a large collection of quality Mexican concert loudspeakers from the port on the North Sea to the far inland capital. As the collection is expected to be big, Boris has to organize a number of lorries to assure smooth transport. The multitude of lorries carrying the cargo through the country is called a convoy.

Boris wants to transport the whole collection in one go by a single convoy and without leaving even a single loudspeaker behind. Strict E.U. regulations demand that in the case of large transport of audio technology, all lorries in the convoy must carry exactly the same number of pieces of the equipment.

To meet all the regulations, Boris would like to do some planning in advance, despite the fact that he does not yet know the exact number of loudspeakers, which has a very significant influence on the choices of the number and the size of the lorries in the convoy. To examine various scenarios, for each possible collection size, Boris calculates the so-called "variability", which is the number of different convoys that may be created for that collection size without violating the regulations. Two convoys are different if they consist of a different number of lorries.

For instance, the variability of the collection of 6 loudspeakers is 4, because they may be evenly divided into 1, 2, 3, or 6 lorries.

Input Specification

The input contains one text line with two integers N, M ($1 \leq N \leq M \leq 10^{12}$), the minimum and the maximum number of loudspeakers in the collection.

Output Specification

Print a single integer, the sum of variabilities of all possible collection sizes between N and M, inclusive.

Sample Input 1	Output for Sample Input 1
`2 5`	`9`

Sample Input 2	Output for Sample Input 2
`12 12`	`6`

Sample Input 3

```
555 666
```

Output for Sample Input 3

```
852
```

Light Emitting Hindenburg

`hindenburg.c`, `hindenburg.cpp`, `Hindenburg.java`, `hindenburg.py`

Lothar is organizing a concert tour of his friends' rock band. The tour will take place in November and each day there will be at most one concert. The tour will be very representative and many musicians are willing to take part in it. The number of musicians in the tour is strictly prescribed and cannot be changed. Each concert on the tour must be attended by all the musicians taking part in the tour.

The good news for Lothar is that the number of candidate musicians is at least as big as the prescribed number of musicians in the tour. The bad news is that a typical musician is not available during the whole month and that various musicians' schedules differ a lot from each other.

Long ago, Lothar wrote a core of a computer scheduling system, and he is exploiting it now to organize the tour. He repeatedly and somewhat randomly chooses a group of musicians of prescribed size, and lets the system calculate an acceptable tour schedule. The system depends on a very specific data format. The schedules of musicians and the tour schedules are represented as numerical codes. The days in November are labeled by their numbers in the month: $1, 2, \ldots, 30$.

For a given musician, each November day is assigned a particular numerical code. A day with label L is coded by integer 2^{30-L} if the musician is available on that day. Otherwise, the day is coded by 0. The musician schedule code is the sum of all his or her day codes.

For a given group of musicians, each November day is assigned a particular numerical code. A day with label L is coded by integer 2^{30-L} if all musicians in the group are available on that day. Otherwise, the day is coded by 0. The group availability code is the sum of all day codes of the group.

For many additional subtle reasons, Lothar thinks that the best tour would be the one with the highest possible value of the availability code of the group of musicians taking part in it.

Input Specification

The first line contains two integers N, K ($1 \leq K \leq N \leq 2 \cdot 10^5$). N is the number of available musicians, K is the prescribed number of musicians taking part in the tour. The next line contains a sequence of N positive integers. Each integer in the sequence represents a code of a schedule of one musician. The codes are listed in arbitrary order.

Output Specification

Print the best possible availability code of any group of K musicians.

Sample Input 1

```
5 2
6 15 9 666 1
```

Output for Sample Input 1

```
10
```

Sample Input 2

```
8 4
13 30 27 20 11 30 19 10
```

Output for Sample Input 2

```
18
```

K==S

`kequalss.c`, `kequalss.cpp`, `Kequalss.java`, `kequalss.py`

Progressive hard octave rock tunes (so-called “phorts”) are written using a specific music notation. This flavor of rock is built on just 13 different note pitches, other pitches (in other octaves) are considered to be an outdated musical ballast. Each note can be either a long one or a short one. Consequently, there are exactly 26 different notes in the rock.

You are going to compose a phort tune on the occasion of your friend’s birthday and perform it with your band on the main town square. While composing the phort, you need to avoid certain musical phrases, which are heavily copyrighted as a result of long research sponsored by big record companies. It has been established that these phrases are very catchy, easy to remember, and could be exploited to bind the listeners subconsciously to a particular music company which would utilize these phrases in their production.

The tune is a sequence of notes. A musical phrase is also a sequence of notes and it is considered to be contained in a tune if its notes form a contiguous subsequence of the tune, which means the same notes appear in the tune right after each other in the same order.

Fortunately, only a few forbidden phrases have been patented so far. Thus, you have a relative freedom in composing your own tunes. In particular, you are interested in the number of acceptable tunes of some length. An acceptable tune is any tune which does not contain a forbidden phrase. The length of the tune is equal to the number of notes it contains.

Input Specification

The first line contains two integers N, Q ($1 \leq N \leq 10^9, 1 \leq Q \leq 100$). N is the length of the tune, Q is the number of forbidden musical phrases. Each of the Q following lines describes one forbidden phrase. A description of a forbidden phrase starts with a positive integer L, indicating its length, followed by a string of L lowercase English letters. Each letter represents one rock note, different letters represent different notes.

The sum of lengths of all forbidden phrases does not exceed 100.

Output Specification

Output the number of different acceptable tunes of length N. Print the result modulo 1 000 000 007.

Sample Input 1

```
2 3
1 a
1 b
1 c
```

Output for Sample Input 1

```
529
```

Sample Input 2

```
3 3
2 aa
1 a
1 a
```

Output for Sample Input 2

```
15625
```

Sample Input 3

```
3 1
2 ab
```

Output for Sample Input 3

```
17524
```

Ponk Warshall

`ponk.c`, `ponk.cpp`, `Ponk.java`, `ponk.py`

Listening to the rock music permutes your nuclear DNA. This astonishing and unbelievable fact was recently published in the Rock Nature Weekly, one of the top scientific journals on the planet. Part of the research was to take DNA samples from volunteers, both before and after the rock concerts season. The samples were processed and various genes isolated from the samples. For each person, each gene was isolated twice: The variant before the rock season and the variant after the season. These two variants were paired and in many cases one variant was found to be some permutation of the other variant in the pair.

The next step in the research is to determine how the permutations happen. The prevalent hypothesis suggests that a permutation is composed of a sequence of transpositions, so-called swaps. A swap is an event (its chemistry is not fully understood yet) in which exactly two nucleobases in the gene exchange their places in the gene. No other nucleobases in the gene are affected by the swap. The positions of the two swapped nucleobases might be completely arbitrary.

To predict and observe the movement of the molecules in the permutation process, the researchers need to know the theoretical minimum number of swaps which can produce a particular permutation of nucleobases in a gene. We remind you that the nuclear DNA gene is a sequence of nucleobases cytosine, guanine, adenine, and thymine, which are coded as C, G, A, and T, respectively.

Input Specification

The input contains two text lines. Each line contains a string of N capital letters “`A`”, “`C`”, “`G`”, or “`T`”, ($1 \le N \le 10^6$). The two strings represent one pair of a particular gene versions. The first line represents the gene before the rock season, the second line represents the same gene from the same person after the rock season. The number of occurrences of each nucleobase is the same in both strings.

Output Specification

Output the minimum number of swaps that transform the first gene version into the second one.

Sample Input 1

```
CGATA
ATAGC
```

Output for Sample Input 1

```
2
```

Sample Input 2

```
CTAGAGTCTA
TACCGTATAG
```

Output for Sample Input 2

```
7
```

Saba1000kg

`saba.c`, `saba.cpp`, `Saba.java`, `saba.py`

There are many different streams in Viking rock movement. Old Icelandic granite rock, Middle Danish dusty Viking rock, Late Finngail dark green rock, Fjord boulder avalanche rock, and many others, a complete list of all popular streams would overflow this page many times. The Scandinavian Ministry of Higher Education studies various ways the streams influence each other. They are currently planning a huge experiment, when a number of suitably chosen volunteers will be distributed over an archipelago of uninhabited small islands, and the researchers want to observe the mutual influence of their rock styles and preferences over a relatively long period of time.

The inhabitants on one island will always influence each other. Some pairs of the islands are situated close enough for their inhabitants to influence each other, while the distances between other pairs prevent any direct influence. In the latter case, the inhabitants of such islands may still influence each other, but only indirectly, if there are one or more other islands that are inhabited and relay the influence.

There are several proposals on the distribution of the volunteers among the islands. For each of these distributions, the Ministry would like to know the number of independent groups of inhabitants that will form in the archipelago. Two groups of island inhabitants, each occupying one or more islands, are considered independent, if there is no possibility of their mutual influence, not even in the indirect way.

Help the Ministry to evaluate their proposals.

Input Specification

The first input line contains three integers, N, E, P ($1 \leq N \leq 10^5, 0 \leq E \leq 10^5, 1 \leq P \leq 10^5$). N is the number of islands in the archipelago, E is the number of pairs of islands that allow direct influence, and P is the number of proposals to be evaluated. The islands are labeled from 1 to N.

The next E lines specify the pairs of islands that allow direct mutual influence. Each of these lines contains two integers A and B denoting the labels of two different islands. No pair of islands occurs more than once.

Each of the next P lines describes one proposal. Each line starts with a number of islands inhabited under that proposal M ($1 \leq M \leq N$) and then contains pairwise distinct labels of M inhabited islands. No other island will be inhabited under the respective proposal.

The sum of the sizes of all proposals (all numbers M) does not exceed 10^5.

Output Specification

For each proposal, print a line with the number of independent groups that will form in the archipelago.

Sample Input 1

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

Output for Sample Input 1

```
2
1
1
```

Sample Input 2

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

Output for Sample Input 2

```
4
```

Screamers in the Storm

`screamers.c`, `screamers.cpp`, `Screamers.java`, `screamers.py`

You may not know that the official name of the “common” urban pigeon you see every day in a city is in fact “rock pigeon” or “rock dove”. Rock pigeon Rocky Dave is in love with Columba Livia, a young female rock pigeon who has keen interest in rock, pigeons, and all that relates to this kind of things. By sheer coincidence, one sunny autumn afternoon, we find them in the city suburbs sitting on the roof of the same building.

To show off his self-confident athletic gait, Rocky Dave decided not to fly over the roof to his object of affection, but to walk towards her instead. The roof is not a modern flat roof, it is a classical roof made of slopes. As the building consists of more smaller buildings joined together, its floor plan is a bit complex. Thus, the “straight line” in which Dave intends to move towards Columba Livia, appears straight only when observed from above. Rocky Dave might have to climb up, over the ridge, down on the other side into a roof valley and then up again, and so on.

We know the exact floor plan of the building, the angle of the slopes on the roof (all angles are the same), and also the original positions of both our hero and heroine on the roof. Given these data, compute the exact length of Rocky Dave’s journey, provided that Columba Livia does not leave her position before Rocky Dave reaches her.

If the path leaves the building boundary, it is Dave’s plan that whenever he meets a rain gutter on the roof edge, he would fly straight and in the same height in the direction towards Columba Livia, until he reaches the rain gutter on some other side of the building, where he would further continue his journey on foot.

To make the situation unambiguous, we present you with the geometric model of a roof. Let Z be a simple polygon in the x-y plane. A simple polygon is a polygon, whose boundary is a closed curve which does not touch or intersect itself. An acceptable pyramid is a pyramid whose base is an axis-parallel square in the x-y plane and its apex (top vertex) is located directly above the base center. The pyramid height is exactly one half of the length of its base side. Moreover, no part of the base of an acceptable pyramid lies outside Z. Note that an acceptable pyramid may formally have zero base side length and zero height, it then consists of a single node, which is also its apex.

A point X in or above the x-y plane is a point of the roof of Z if and only if it is an apex of an acceptable pyramid and no point directly above X is an apex of an acceptable pyramid.

Input Specification

The first line contains five integers. The first of them, N ($4 \leq N \leq 400$), is the number of corner points of the polygon formed in the x-y plane by the edges of the floor plan of the building. The next two are the x and y coordinates of Rocky Dave and the last two are the x and y coordinates of Columba Livia.

Each of the next N lines contains the x and y coordinates of one corner point of the polygon. The points are listed in the counter-clockwise order. Every side of the building is parallel with one of the axes.

Neither of the two pigeons is located outside of the given polygon. All input coordinates are integers with the absolute value of at most 10^5.

Output Specification

Output the length of Rocky Dave's journey. The answer must have an absolute or relative error less than 10^{-7}.

Sample Input 1

```
4 3 0 3 4
0 0
4 0
4 4
0 4
```

Output for Sample Input 1

```
4.828427124746
```

Sample Input 2

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

Output for Sample Input 2

```
6.292528739884
```

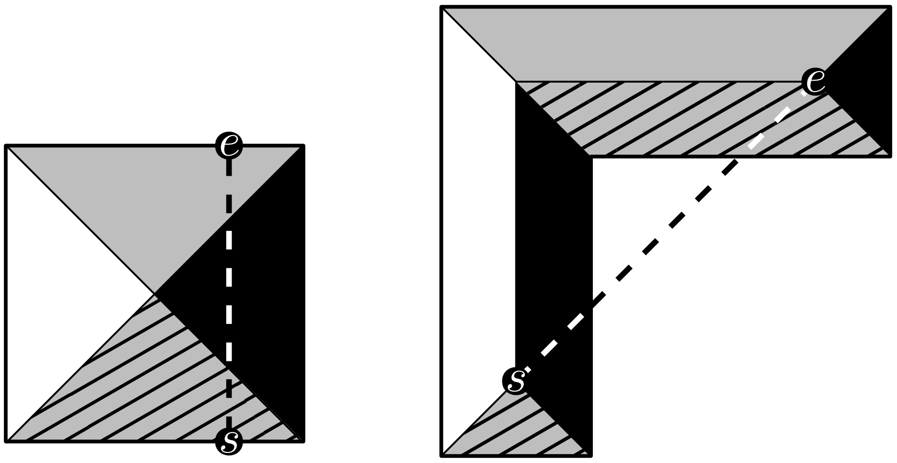

Figure 1: Illustration of Sample Input 1 and Sample Input 2

The Bugs

`thebugs.c`, `thebugs.cpp`, `Thebugs.java`, `thebugs.py`

A submarine painted in bright plastic lemon color is investigating ocean depths and measuring the concentration of plastic yoctoparticles in the water. Each measurement (due to ingenious choice of the units) is a positive integer.

The measurement system is very recent, poorly tested and prone to errors. The project management suspects it is full of bugs. They find themselves in a time of trouble. Mother Mary also suspects it is full of bugs. Everybody cries: Help!

To identify and correct the bugs, they came together and decided to take the sequence of measured concentrations and analyze all triplets that occur in the sequence of measurements.

- A triplet is a sequence of three integers.
- Each triplet is associated with its characteristic.
- The characteristic of a triplet (x, y, z) is a triplet $(\mathrm{sgn}(y - x), \mathrm{sgn}(z - y), \mathrm{sgn}(z - x))$. The value of the $\mathrm{sgn}(x)$ function is -1, 0 or 1 for negative, zero or positive value of x, respectively.
- A triplet (x_1, y_1, z_1) is smaller than triplet (x_2, y_2, z_2) if and only if the first nonzero value in the triplet $(x_1 - x_2, y_1 - y_2, z_1 - z_2)$ is negative.
- The label of a triplet T is the smallest triplet whose values are all positive and whose characteristic is equal to the characteristic of T.

A measured triplet is a triplet which is a subsequence of the measurement sequence. That is, the elements of the triplet appear in the measurement sequence in the same order they appear in the triplet, but in the sequence they do not necessarily follow each other.

Before they are analyzed, the measured triplets are grouped according to their labels. The management wants to know in advance the set of labels of all measured triplets.

Input Specification

The first line contains one integer N ($3 \le N \le 2 \cdot 10^5$), the number of measurements. The next line contains N integers $x_1, x_2, \ldots, x_N$ ($1 \le x_i \le 10^9$), each representing one measurement.

Output Specification

Print, in the increasing order, all different labels of all measured triplets which can be obtained from the given sequence of measurements, one per line. A label should be printed with no spaces between its consecutive elements.

Sample Input 1

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

Output for Sample Input 1

```
123
```

Sample Input 2

```
6
5 4 9 1 7 4
```

Output for Sample Input 2

```
121
132
211
212
213
231
312
321
```