ICPC Europe Regionals

icpc global sponsor
programming tools

icpc.foundation

icpc diamond
multi-regional sponsor

The 2021 ICPC Central Europe Regional Contest

Official Problem Set

A – Airline

Time limit: 15 s *Memory limit: 512 MiB*

An airline company offers regular flights involving n different airports. Each flight links two airports directly (i.e. without stopping at any other airport) and allows travel in both directions. The flights are arranged such that for any choice of starting airport s and destination airport t, there exists exactly one sequence of flights between the two airports without visiting any airport more than once. The number of flights in this sequence is called the *distance* between s and t.

Were the airline to add another flight, say between airports x and y, it is possible that for some pairs (s, t), another, shorter sequence of flights from s to t would form. The more pairs affected, the more promising the new connection between x and y is considered to be. The airline is asking you to help them evaluate several possible additions (x, y) with respect to this criterion.

Input data

The first line contains two integers, n, the number of airports, and q, the number of possible additions (x, y) that are to be evaluated.

The next $n-1$ lines describe the original flights before any additions. The i-th of these lines contains two integers u_i and v_i, indicating that there is a direct flight connection between airports u_i and v_i.

The remaining q lines describe the possible additional flights that are being considered. The i-th of these lines contains two integers x_i and y_i, indicating that in the i-th scenario the original $n-1$ flights would be supplemented by a new direct flight connection between airports x_i and y_i.

Input limits

- $2 \leq n \leq 10^6$
- $1 \leq q \leq 10^5$
- $1 \leq u_i \leq n$; $1 \leq v_i \leq n$; $u_i \neq v_i$
- $1 \leq x_i \leq n$; $1 \leq y_i \leq n$; $x_i \neq y_i$
- $\sum_{i=1}^{q} d_i \leq 10^7$, where d_i is the distance between x_i and y_i in the original flight network.

Output data

Output q lines; in the i-th line, output the number of pairs (s, t) such that $1 \leq s < t \leq n$ and the distance between airports s and t would decrease if the original network of $n-1$ flights were supplemented by a direct flight connection between the airports x_i and y_i.

Example

Input	Output
8 2 1 5 5 2 7 3 3 8 6 4 4 5 6 3 5 7 2 6	10 4

B – Building on the Moon

Time limit: 1 s *Memory limit: 256 MiB*

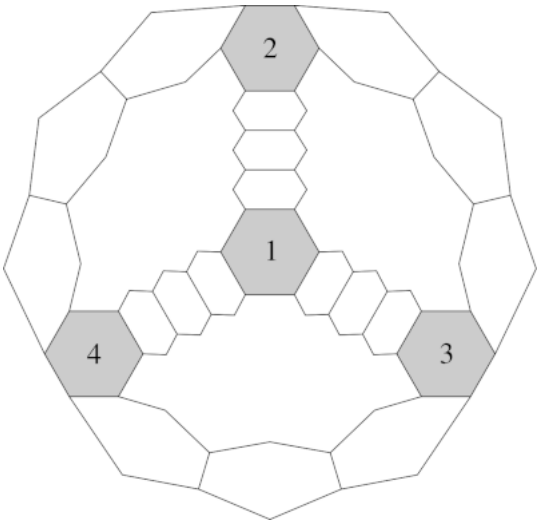
ICPC coaches never properly retire. When they announce their "retirement", they actually start working for a secret agency (we are not allowed to disclose any further details) that builds monumental structures on the dark side of the Moon. There is currently one such project in progress.

To construct that monumental building they can use *hexagonal* building blocks of two different types:

- A *chamber* has three openings that are located on three of its mutually non-incident sides.
- A *link* has two openings that are located on two opposite sides.

Two links (or a link and a chamber) can be attached together along those sides that contain an opening; then those structures are welded together, so that they become airtight.

The plan is to construct a building that will comprise N chambers on the Moon's surface. Each of these chambers will be connected to exactly three other chambers via *passages*. Each passage is built by attaching together L links. Each end (i.e. where the opening is located) of a passage is attached to a chamber. For example, suppose that there are $N = 4$ chambers labeled 1 to 4 and suppose that $L = 3$. A possible structure is shown in the figure (chambers are shaded gray):



It is guaranteed that any pair of chambers is connected by at most one passage and no passage is connecting a chamber to itself. Also, a person inside the building may reach any chamber from any other chamber via passages. Moreover, the plan was prepared by a former CERC coach, so it is guaranteed that passages do not intersect each other (remember that the structure will be built on the surface of the Moon). Such a plan may be described by a sequence of triples

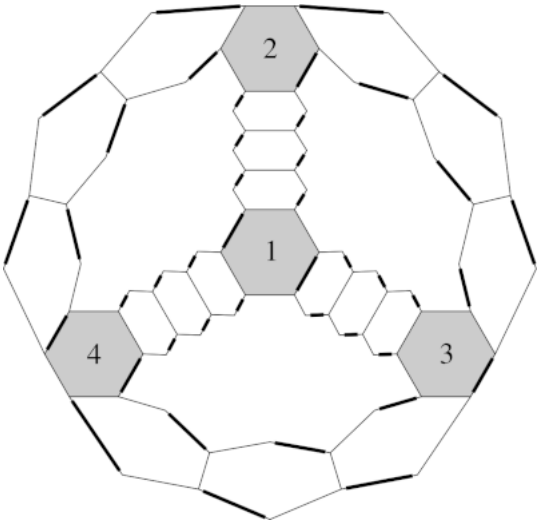
$$(c_1^{(1)}, c_2^{(1)}, c_3^{(1)}), (c_1^{(2)}, c_2^{(2)}, c_3^{(2)}), \ldots, (c_1^{(n)}, c_2^{(n)}, c_3^{(n)}).$$

This means that chamber i is connected to chambers $c_1^{(i)}, c_2^{(i)}$ and $c_3^{(i)}$. If a person is standing in chamber number i and does a pirouette in the clockwise direction, that person

will see the passage to chamber $c_1^{(i)}$, followed by the passage to chamber $c_2^{(i)}$, and finally the passage to chamber $c_3^{(i)}$. The above plan can be described by the following sequence:

$$(2, 3, 4), (1, 4, 3), (1, 2, 4), (1, 3, 2).$$

Since the the dark side of the Moon is dark (as the name conveniently suggests), a neon tube will be attached on each side of each building block (either a chamber or a link). Of course, a side where two building blocks are welded together will only have one neon tube. Because the structure will be on the Moon, we should not waste too much energy, so no two incident neon tubes should be lit at the same time. The coaches have decided that in order to provide sufficient lighting, they will light the *maximum* number of neon lights (with respect to the energy-saving constraint). Such a lighting will be called a *valid lighting* and may even be obtained in several ways. One possibility is shown in the following figure:



The coaches feel that there are many more ways to achieve this. They are now wondering what is the total number of valid lightings. Because they are too lazy to code it, they will prepare a task for a contest, so that students will come up with efficient solutions. Write a program that will read in the description of a monumental structure and determine the total number of valid lightings. Because the solution may be a huge number, output the answer modulo $10^6 + 3$.

Input data

The first line contains space-separated integers N and L, where N is the number of chambers and L is the number of links in each passage. This is followed by N lines; the i-th line contains space-separated integers $c_1^{(i)}, c_2^{(i)}$ and $c_3^{(i)}$.

Input limits

- $4 \leq N \leq 16$
- $1 \leq L \leq 100$
- $1 \leq c_j^{(i)} \leq N$ for $j = 1, 2, 3$ and $i = 1, 2, \ldots, N$

Output data

Print a single integer: the total number of valid lightings modulo $10^6 + 3$.

Example

Input	Output
4 3 2 3 4 1 4 3 1 2 4 1 3 2	4400

C – Cactus cutting

Time limit: 15 s *Memory limit: 256 MiB*

Mr Malnar has given up on his tree obsession and found something even more interesting, cacti! Formally, a cactus is a connected graph where each edge is contained in at most one cycle. A cycle is defined as a sequence of more than one distinct edge in which every two consecutive edges share a common endpoint, and the first and last edge share a common endpoint as well.

Unfortunately, the cactus that Mr Malnar bought is rather big, so he would like to cut it up into disjoint sticks. One stick is defined as a pair of edges that share a common endpoint. Mr Malnar is a pedantic individual, so he wants to know the exact number of ways he can cut up his cactus into sticks.

Input data

The first line contains the number of vertices N and the number of edges M. This is followed by M lines, each containing two distinct integers A_i and B_i denoting an edge between vertices A_i and B_i. Each edge will be listed exactly once.

Input limits

- $1 \leq N, M \leq 100\,000$
- $1 \leq A_i, B_i \leq N$

Output data

Compute the number of distinct ways Mr Malnar can cut his cactus up into sticks. Since this number can get quite large, output the result modulo $10^6 + 3$.

Example

Input	Output
10 12 1 6 2 5 7 2 8 9 8 1 2 6 4 3 4 10 3 10 3 9 1 3 5 7	8

D – DJ Darko

Time limit: 4 s *Memory limit: 256 MiB*

A new DJ is in town. DJ Darko needs to set up his speakers. He has N speakers in a row with the i-th speaker volume set to A_i. Changing the volume is rather difficult so the i-th speaker requires B_i units of energy to increase or decrease the volume by the value of 1.

Unforutanently, Darko's evil twin brother Karko likes to mess with him. There are Q events that will be happening.

```
1 l r x
2 l r
```

In an event of type 1, Karko changes the volume of all speakers from the l-th to the r-th by x. In an event of type 2, Darko sets all the speakers from the l-th to the r-th to the same volume in a way that uses up the minimal amount of energy. If there are multiple ways of doing that, he chooses the one which minimizes the final volume.

As a bystander, you would like to know the volume that Darko set for each event of type 2.

Input data

The first line contains the number of speakers N and the number of events Q. In the second line, there are N numbers A_i indicating the current volume of the speakers. In the third line, there are N numbers B_i, indicating the energy needed to change the volume of the i-th speaker by one. In the next Q lines there are Q events, formatted in the way described above. All numbers in the input are integers.

Input limits

- $1 \leq N, Q \leq 200\,000$
- $0 \leq A_i, B_i \leq 10^9$
- $1 \leq l \leq r \leq N$
- $-10^{-9} \leq x \leq 10^9$

Output data

For each event of type 2, output the volume to which Darko set the speakers.

Examples

Input	Output
5 5 8 1 6 4 9 3 6 4 1 7 2 2 4 1 1 4 -8 2 1 1 2 1 3 2 4 5	1 0 -7 9

Input	Output
8 3 4 3 9 3 7 6 4 8 9 5 8 5 2 2 1 8 1 1 7 -10 2 5 5 2 4 7	-3 -7

E – Fishing

Time limit: 10 s *Memory limit: 512 MiB*

There is a small village situated on the coast of the Adriatic Sea. The fishermen map the sea as a grid of $N \times M$ cells such that the first row is adjacent to the coast and the last row is the furthest away. They track the movement of fish and other items floating in the sea. The sea is mostly empty, but there are K grid cells of interest. The location of each such point is denoted by row R_i and column C_i. The fishermen estimate that their catch from fishing in the i-th cell is going to be worth V_i. Note that V_i can be zero or negative if the corresponding area is predominantly occupied by undesired items. All other cells are considered to have a value of 0.

Every day, the local council approves a rectangular fishing area that includes columns from X to Y and extends H rows from the shore into the sea. To fish in the selected area, the fishermen will prepare a fishing net that is exactly H units long. Although the net has a fixed length, it can be rolled out to an arbitrary width W that doesn't exceed $Y - X + 1$. Based on their information about the sea, they will drop the net somewhere within the approved fishing area to maximize the catch defined as the sum of cell values covered by the net.

The fishermen aim to choose the optimal fishing location every day. Write a program that will find the best value of their catch for the approved fishing areas for the next Q days. You may assume that the cell values are constant; they are not depleted from fishing on previous days.

Input data

The first line contains the number of rows N, the number of columns M and the number of non-empty cells K. These cells are described in the following K lines with their row R_i, column C_i and value V_i, separated by a space. Rows are numbered from 1 to N and columns from 1 to M. All values V_i are integers.

The next line contains the number of queries Q. The j-th query is described by three integers A'_j, X'_j and Y'_j. To ensure that your solution answers queries in the given order, the queries are given in an encoded form. The actual query can be computed as

$$\begin{aligned} H_j &= H'_j \oplus A_{j-3}, \\ X_j &= X'_j \oplus A_{j-2}, \\ Y_j &= Y'_j \oplus A_{j-1}, \end{aligned}$$

where A_j denotes the answer to the j-th query (or 0 if $j \leq 0$) and $\oplus$ denotes a bitwise xor operation. Your program should find the region with the maximum catch value that spans the first H_j rows and some subrange of columns from X_j to Y_j.

Input limits

- $1 \leq N, M, K, Q \leq 300\,000$
- $|V_i| \leq 1000$

Output data

For each query, output a single line with the maximum value of the catch. Note that the fishermen can always choose to keep an empty net with the value of 0.

Example

Input

```
10 7
12
2 6 -5
3 3 3
4 2 -2
4 6 2
5 3 -1
5 5 5
7 1 8
7 7 4
8 4 -3
8 5 1
9 6 -4
10 3 2
6
5 1 5
10 1 0
7 1 11
15 15 6
9 1 0
3 7 1
```

Output

```
7
13
0
6
3
0
```

Comment

The decoded list of queries:

```
5 1 5
10 1 7
7 6 6
8 2 6
4 1 6
3 1 2
```

F – Letters

Time limit: 2 s *Memory limit: 256 MiB*

Martin is attending a lecture on linear algebra. It is needless to say that the professor who is giving the lecture is the most boring person in the entire universe. There is a $N \times M$ matrix written on the blackboard. Some of the entries in the matrix are letters (of the English alphabet) while some other entries are blank. Here is an example of such a matrix of size 6×8:

$$\begin{bmatrix} k & l & & n & d & i & & \\ & & & & c & & & \\ & & & & & i & h & \\ j & & a & & & & & \\ & c & b & & & & & \\ & c & & & & e & f & \end{bmatrix}.$$

Martin has absolutely no idea what this matrix represents. He is so bored that he has not been following the lecture anymore for the last 30 minutes. However, Martin has an extremely vivid imagination. He is imagining that the matrix is suddenly influenced by gravity and all the letters in it are sliding downwards until each letter either 'reaches the bottom' or 'hits the letter that is below it'. In the first phase, the above matrix becomes:

$$\begin{bmatrix} & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & l & & & & i & & \\ k & c & a & & d & i & h & \\ j & c & b & n & c & e & f & \end{bmatrix}.$$

After that, gravity changes direction and is now pulling the letters to the left. We are now in the second phase. Again, all the letters are sliding to the left until each letter either 'reaches the left bracket' or 'hits the letter on its left'. The previous matrix thus becomes:

$$\begin{bmatrix} & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ l & i & & & & & & \\ k & c & a & d & i & h & & \\ j & c & b & n & c & e & f & \end{bmatrix}.$$

Martin is carrying out this procedure in his head until the very end of the boring lecture. Of course, after each phase, i.e. after all the letters land at their respective destinations, gravity may change its direction (there are four possibilities for the direction: left, right, up and down).

Task

Write a program that determines the final positions of all letters in the matrix given the precise sequence of the gravity direction changes.

Input data

The first line contains three integers N, M and K where $N \times M$ is the size of the matrix and K is the number of phases.

The second line contains a string of length K that consists of letters `L`, `R`, `U` and `D` that represent the direction of gravity in each phase (left, right, up and down, respectively).

The final N lines represent the matrix. Each of the lines contains M characters. The characters are lowercase letters of the English alphabet and '`.`' (dot) which represents a blank entry.

Input limits

- $1 \leq N, M \leq 100$
- $0 \leq K \leq 100$

Output data

Output the matrix which Martin obtained at the end of the lecture. The format of the matrix is identic to the one in the input data.

Examples

Input

```
6 8 5
DLURD
k.l.ndi.
.....c..
......ih
j..a....
..cb....
..c...ef
```

Output

```
........
........
........
......hf
..iadice
.lkcbnjc
```

Input

```
3 3 0

a..
.b.
..x
```

Output

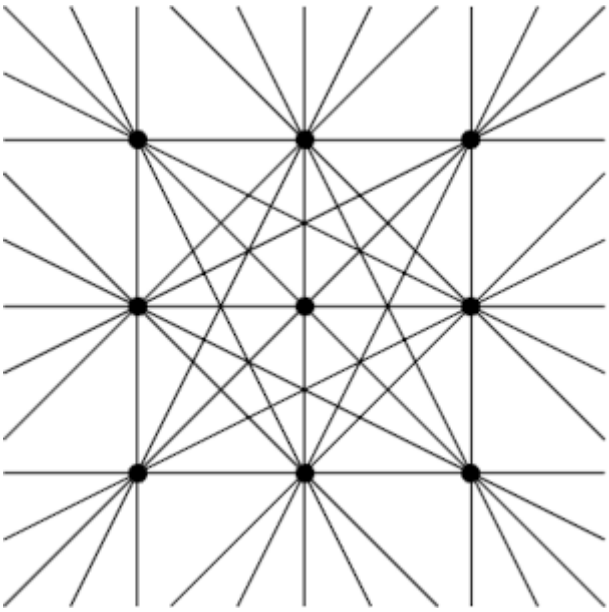
```
a..
.b.
..x
```

G – Lines in a grid

Time limit: 8 s *Memory limit: 1024 MiB*

Suppose that we are given a $n \times n$ integer grid, e.g. $\{(i, j)\}_{i=0,j=0}^{n-1,n-1}$. Let l_n be the number of different lines that intersect with at least two points on the grid.

For $n = 3$, there are exactly 20 such lines, as drawn on the image below.



Compute l_n for all given n.

Input data

First line contains an integer Q – the number of queries. The second line contains Q space-separated integers $n_1, \ldots, n_Q$.

Input limits

- $1 \leq Q \leq 1000$
- $1 \leq n_i \leq 10^7$

Output data

Print Q numbers $l_{n_1}, \ldots, l_{n_N}$, each in its own line. Since l_k can be large, print them modulo $10^6 + 3$.

Example

Input	Output
3 1 3 2	0 20 6

H – Radar

Time limit: 2 s *Memory limit: 256 MiB*

We are using a special radar to scan an area. The radar accepts a list of distances, e.g. $2, 4, 1$, and a list of angles, e.g. $100°, 270°, 180°, 10°, 300°$, and scans the points across all the given distances and angles. How close to some other points of interest will we be able to scan?

Input data

The first line of the input gives three space-separated integers: R, F, N, representing the number of radii, the number of angles, and the number of points of interest, respectively. Then R lines follow, i-th of which contains an integer r_i, representing the distance from the radar that will be scanned. Then, F lines follow, each containing two space-separated integers $(f_x)_i$, $(f_y)_i$, that represent Cartesian coordinates of a point, defining the i-th angle. Then, N lines follow, each containing two space-separated integers x_i, y_i, that represent the Cartesian coordinates of the i-th point.

The angle, defined by the point $(f_x)_i$, $(f_y)_i$ is the angle from the x-axis to the ray from the origin through $(f_x)_i$, $(f_y)_i$.

Input limits

- $1 \leq R, F, N \leq 10^5$
- $|x_i|, |y_i|, |(f_x)_i|, |(f_y)_i|, r_i < 10^6$
- $(f_x)_i^2 + (f_y)_i^2, r_i > 0$
- All r_i are pairwise distinct.
- Rays, defined by $(f_x)_i, (f_y)_i$, are pairwise distinct.

Output data

Output N lines, i-th of which should contain the distance from the point (x_i, y_i) to the closest scanned point. The result will be considered correct if it is within the 10^{-6} of absolute or relative precision.

Example

Input

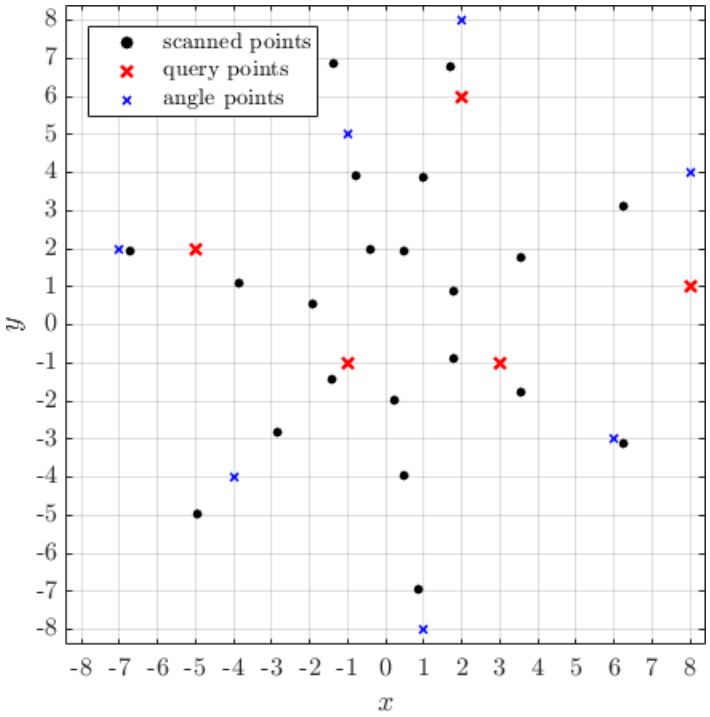
```
3 7 5
2
4
7
8 4
2 8
-1 5
-7 2
-4 -4
1 -8
6 -3
3 -1
8 1
2 6
-5 2
-1 -1
```

Output

```
0.977772290466
2.750120773895
0.846777708005
1.464071052924
0.585786437627
```

Comment

Illustration of sample case:



I – Regional development

Time limit: 4 s *Memory limit: 256 MiB*

The King has received several complaints that certain regions of his kingdom are economically neglected. The citizens have not seen a single merchant travelling on certain roads between villages in a very long time. To fix this problem and return wealth and prosperity to his kingdom, the King has appointed his royal mathematician to come up with a viable plan of merchant's routes.

The plan will consist of a positive number of merchants travelling along each road in one of the directions. The number of merchants entering a village along the roads should be equal to the number of merchants exiting it. To ensure a somewhat even distribution of merchants throughout the kingdom, the King has requested that the number of merchants travelling along each road should be at least one and less than M.

The royal mathematician has been summoned by the King to present his findings. His future is uncertain as he has not been able to solve the problem. However, he did make some progress. He found a plan with a valid number of merchants travelling along each road. The only problem is that the incoming and outgoing merchants in the villages do not add up (at least not exactly). Their difference might not be zero for every village, but it is equal to zero modulo M. He is willing to share his findings with you, if you can write a program that finds a valid plan or reports that it doesn't exist.

Input data

The first line contains N, the number of villages, R, the number of roads and the number M.

The following R lines describe the roads with numbers A_i, B_i and C_i that indicate a road between villages A_i and B_i with C_i merchants travelling from A_i to B_i. Cities are numbered from 1 to N. There is at most one road between each pair of villages and no road connects a village with itself. The difference between incoming and outgoing merchants in each village is equal to 0 modulo M.

Input limits

- $1 \leq N \leq 1000$
- $0 \leq R \leq 10\,000$
- $2 \leq M \leq 1000$
- $1 \leq A_i, B_i \leq N$
- $0 < C_i < M$

Output data

Print the number of merchants travelling along each road. Print them in the same order as they were given in the input and on separate lines. If the merchants travel in the opposite direction with respect to the order of cities that defined a road in the input, indicate this with a negative value (e.g. if there are X merchants travelling from B_i to A_i, indicate this with $-X$ in the i-th line of output).

If there are multiple solutions, you can output any of them. If no solution exists, print the word `"IMPOSSIBLE"` in a single line (without the quotes).

Example

Input	Output
4 5 4 1 2 1 2 3 2 4 1 1 2 4 3 3 4 2	2 3 2 -1 3

J – Repetitions

Time limit: 10 s *Memory limit: 512 MiB*

Bob is an aspiring avant-garde writer. He disdains the use of spaces, punctuation, capital letters and the like; hence, his stories are nothing but long strings of lowercase letters of the English alphabet. Critics have also noted that his style is marked by a certain fondness for *repetitions*, in the sense that it sometimes happens that two instances of the same substring appear in his story twice in a row, without any other intervening characters.

Bob has submitted his latest masterpiece, a string which happens to be n characters long, to q different literary magazines in the hopes that at least one of them might be willing to publish it. The response was more favourable than he had dared to hope. The editors of all q magazines have expressed willingness to publish some part (i.e. a substring) of his story, but on the condition that he identify the longest repetition (i.e. a shorter substring appearing twice in a row) within that part of the story. The editors intend to remove that part to prevent the story from being too boring. Now Bob needs your help to answer these queries from the editors.

Write a program that, given a string of n letters, $s[1]s[2]\ldots s[n]$, answers q queries of the form "given a_i and b_i, how long is the longest string t for which tt appears as a substring of $s[a_i]s[a_i+1]\ldots s[b_i-1]s[b_i]$, and where does the leftmost such occurrence begin?"

Input data

The first line contains two integers, n and q. The second line contains the string s, which is n characters long; all these characters are lowercase letters of the English alphabet. The remaining q lines describe the queries; the i-th of these lines contains the integers a_i and b_i, separated by a space.

Input limits

- $1 \leq n \leq 10^6$
- $1 \leq q \leq 100$
- $1 \leq a_i \leq b_i \leq n$ for each $i = 1, 2, \ldots, q$

Output data

Output q lines; the i-th of these lines must contain two space-separated integers ℓ_i and c_i. ℓ_i should be the length of the longest string t for which tt appears as a substring in $s[a_i]s[a_i+1]\ldots s[b_i-1]s[b_i]$, and c_i should be the index at which the leftmost repetition of this length begins, i.e. the smallest integer such that $a_i \leq c_i$, $c_i + 2\ell_i - 1 \leq b_i$ and $s[c_i]\ldots s[c_i+\ell_i-1] = s[c_i+\ell_i]\ldots s[c_i+2\ell_i-1]$. (If $\ell_i = 0$, then $c_i = a_i$ by definition.)

Example

Input	Output
10 4 cabaabaaca 4 8 1 9 5 9 8 10	1 4 3 2 1 7 0 8

Comment

The four queries in the above example refer to the substrings **a**abaa, c**aba**abaac, ab**a**ac, and aca; the part shown in bold is the substring referred to by the result of that query (a substring of length ℓ_i, beginning at index c_i). In the last query there is no repetition, so $\ell_4 = 0$.

K – Single-track railway

Time limit: 4 s *Memory limit: 512 MiB*

Trains running on a single-track railway can only meet at the stations. Suppose that a pair of trains simultaneously leave in the opposite directions, one from the initial and the other from the final station, i.e. the initial station in the opposite direction. It is likely that one of the trains will have to wait for the other one at one of the stations along the railway. To minimize the delays, the trains meet at the station such that the waiting time is minimized.

We know the travel time between each two adjacent stations, equal in both directions. Unfortunately, the travel times constantly change because of the works along the railway. You are given the initial travel times and an updated travel time for the affected section after each change. Write a program that computes the shortest possible waiting time for a pair of trains leaving from the opposite ends of the railway after each of the changes.

Input data

The first line specifies the number of stations, n. In the second line, $n-1$ numbers are given, corresponding to the initial travel times between the adjacent stations (the i-th number is the travel time between stations i and $i+1$). The third line specifies the number of changes, k. This is followed by k lines, each containing two numbers: the first one, $j \in [1, n-1]$, specifies the station, and the second gives the updated travel time between stations j and $j+1$. Keep in mind that the first station is numbered 1 rather than 0.

Input limits

- $2 \leq n \leq 200\,000$
- $0 \leq k \leq 200\,000$
- All travel times (both the initial and the updated ones) are integers from the interval $[1, 10^6]$.

Output data

Output $n+1$ lines, where the i-th line will contain the shortest possible waiting time after $i-1$ changes (the first one should correspond to the situation before any changes).

Example

Input	Output
6 20 70 40 10 50 2 4 80 2 30	10 0 40

Comment

At the beginning, the trains leaving in the opposite directions should meet at station 3. The first train will reach that station in 90 minutes, and the second will arrive there in 100 minutes; the waiting time will thus be 10 minutes. Following the first change, the optimal meeting point becomes station 4. Both trains will take 130 minutes to get there, so neither will have to wait. After the second change, they will also meet at station 4. This time, however, the train that arrives first will have to wait for 40 minutes.

L – Systematic salesman

Time limit: 6 s *Memory limit: 256 MiB*

A travelling salesman has received a list of cities that he must visit in a single journey. He may start and end his tour in any city as long as he visits each city at least once. He doesn't have to start and finish in the same city. The travelling salesman noticed that his fellow travelling salesmen are spending way too much time planning and finding an optimal route. Therefore, he has decided to take a different, more systematic approach to planning his route.

He will first split all the cities into the left half and the right half. If the number of cities is odd, the right half will contain one city more than the left. He will then pick one of the halves and visit all the cities in that half before visiting any of the cities in the other half.

To visit the cities in the selected left/right half, he will split this set of cities into the lower and upper half. In case of an odd number of cities in this set, the upper half will contain an extra city. Again, he will first visit all cities in one of these halves before visiting any of the cities in the other half.

He will continue applying the same logic of alternatingly splitting the cities in half horizontally and vertically until he obtains the complete path of the journey. Compute the shortest path that visits all cities that the salesman can obtain in this way.

Input data

The first line contains the number of cities N that the salesman wants to visit. Locations of these cities are given in the N lines that follow. The location of each city is defined by a space-separated pair of integer coordinates X_i and Y_i, which describe its location on a plane. It is guaranteed that X_i coordinates of all cities are distinct. The same holds for all Y_i coordinates.

Input limits

- $1 \leq N \leq 1000$
- $0 \leq X_i, Y_i \leq 10^6$

Output data

In the first line, print the minimum length of the salesman's path. The length will be considered correct if it differs from the official solution by at most 10^{-4}. In the second line output a space-separated list of cities as visited by the salesman. The cities are numbered from 1 to N in order as given in the input. If there are several solutions, you can output any of them.

Example

Input	Output
6 5 1 9 6 2 5 3 3 10 4 7 2	13.142182 3 4 1 6 5 2

Comment

The salesman first visits the left half (cities 1, 3 and 4) and then the right half (cities 2, 5 and 6).

To visit cities 1, 3 and 4, he first visits the upper half (cities 3 and 4) before the lower half (city 1). The cities in the upper half are split into the left half (city 3), visited first, and the right half (city 4), visited afterwards.

Cities 2, 5 and 6 are split into the lower half (city 6) and the upper half (cities 2 and 5). Here, the salesman first visits the lower half. He continues his route with the upper half, where he first visits the right half (city 5) and concludes with the left half (city 2).