

Problem A. Ya Sabah El GCD

Input file: **standard input**
Output file: **standard output**
Time limit: 1 second
Memory limit: 256 megabytes

You are given two arrays a and b , each of length n ($1 \leq n \leq 5000$). You may perform the following operation any number of times (indices are 1-based):

- Choose an index i ($1 \leq i \leq n$) and set $a[i] := \gcd(a[i], b[i])$. The cost of performing this operation at index i is $i^2 + i$.

Your task is to compute the minimum total cost required to make the gcd of all elements of array a equal to 1. If it is impossible to achieve $\gcd(a[1], a[2], \dots, a[n]) = 1$ by any sequence of operations, print -1 .

Input

The first line contains a single integer n ($1 \leq n \leq 5000$).

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a[i] \leq 10^{12}$).

The third line contains n integers $b_1, b_2, \dots, b_n$ ($1 \leq b[i] \leq 10^{12}$).

Output

Print a single integer — the minimum total cost to make $\gcd(a_1, a_2, \dots, a_n) = 1$, or -1 if it is impossible.

Examples

standard input	standard output
2 6 10 3 7	2
3 6 30 60 5 10 21	2

Problem B. Bouncing Chaos

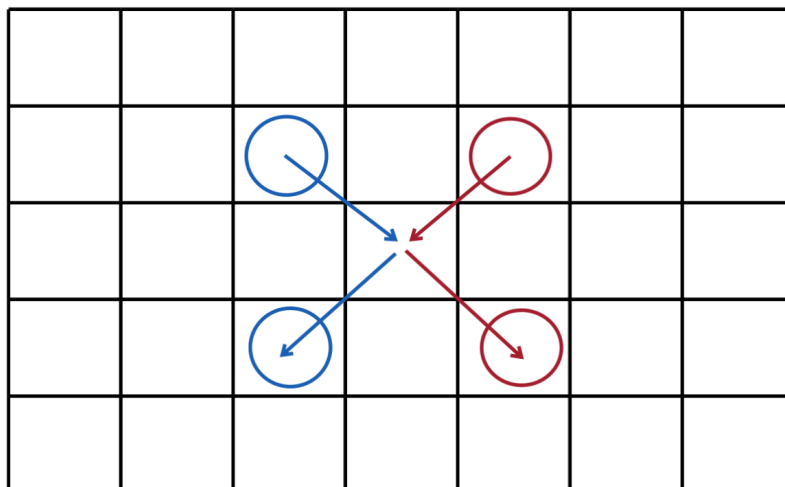
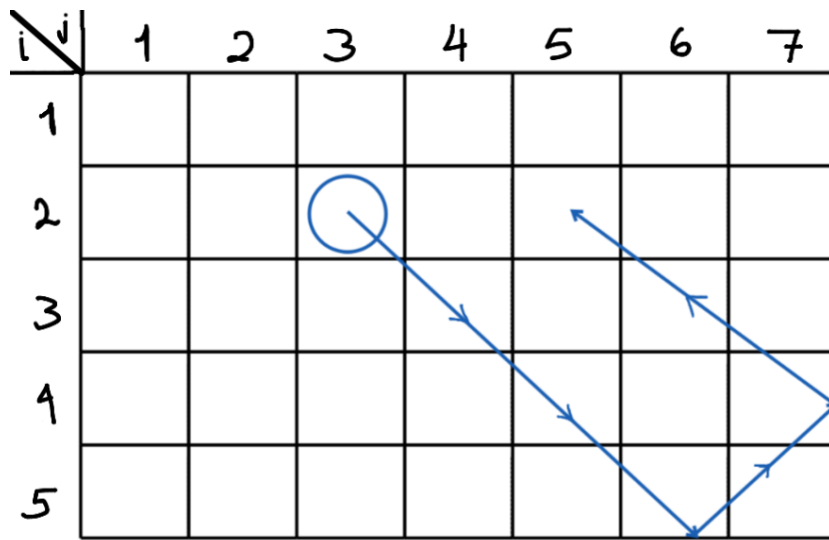
Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

Abo Adnan is experimenting with two balls bouncing inside a rectangular grid of size $n \times m$. The grid cells are numbered from $(1, 1)$ to (n, m) .

Each ball starts at a specific cell with an initial direction of movement: UR (up-right), UL (up-left), DR (down-right), or DL (down-left). These directions correspond to moving $(-1, +1)$, $(-1, -1)$, $(+1, +1)$, and $(+1, -1)$ in terms of $(row, column)$ changes respectively.

When a ball hits a horizontal boundary (top or bottom edge), its row direction reverses. When a ball hits a vertical boundary (left or right edge), its column direction reverses. The balls move simultaneously at each step, and when they collide both balls' directions will exchange (see the photos below for more understanding).

After k steps, determine the positions of both balls and print them in lexicographical order.



Input

The first line contains three integers n , m , and k ($3 \leq n, m$, $n \cdot m \leq 2 \cdot 10^5$, $1 \leq k \leq 10^{12}$) — the grid

dimensions and the number of steps.

The second line contains two integers i_1, j_1 and a string d_1 ($2 \leq i_1 \leq n - 1$, $2 \leq j_1 \leq m - 1$, $d_1 \in \{\text{UR}, \text{UL}, \text{DR}, \text{DL}\}$) — the starting position and direction of the first ball.

The third line contains two integers i_2, j_2 and a string d_2 ($2 \leq i_2 \leq n - 1$, $2 \leq j_2 \leq m - 1$, $d_2 \in \{\text{UR}, \text{UL}, \text{DR}, \text{DL}\}$) — the starting position and direction of the second ball.

It is guaranteed that $(i_1, j_1) \neq (i_2, j_2)$.

Output

Output two lines, each containing two integers — the positions of the two balls after k steps. Output the positions of the balls in lexicographical order.

Examples

standard input	standard output
5 7 10 2 3 UR 4 5 DL	2 1 4 7
6 8 15 3 4 DR 5 6 UL	2 5 4 5

Problem C. Shift And Think

Input file: **standard input**
Output file: **standard output**
Time limit: 1 second
Memory limit: 256 megabytes

Professor Hussein is a renowned researcher in digital signal processing. He models complex, repeating signals as permutations. A permutation of length n is an arrangement of the integers from 1 to n , inclusive, where each integer appears exactly once.

For example, $[2, 1, 4, 3]$ and $[1, 2]$ are permutations, while $[1, 3, 1]$ and $[1, 2, 4]$ are not.

A key operation in the professor's analysis is the **cyclical left rotation**. Given a permutation p , we define $\text{rot}(p)$ as the permutation obtained by shifting p one position to the left, with the first element wrapping around to the end.

For example, if $p = [1, 3, 4, 2]$, then $\text{rot}(p) = [3, 4, 2, 1]$.

More generally, $\text{rot}^i(p)$ is the result of applying the left rotation operation i times. For instance, $\text{rot}^2(p) = [4, 2, 1, 3]$.

Professor Hussein is particularly interested in a property he calls **stable indices**. An index i (where $1 \leq i \leq n$) is a stable index of a permutation p if applying the permutation mapping twice, starting from that index, returns the original index. Formally, i is a stable index if:

$$p[p[i]] = i$$

We denote $S(p)$ as the total number of stable indices in the permutation p .

Example:

Let $p = [3, 6, 1, 4, 2, 5]$.

- Index 1: $p[1] = 3$. $p[p[1]] = p[3] = 1$. Since $p[p[1]] = 1$, index 1 is stable.
- Index 2: $p[2] = 6$. $p[p[2]] = p[6] = 5$. Since $5 \neq 2$, index 2 is not stable.
- Index 3: $p[3] = 1$. $p[p[3]] = p[1] = 3$. Since $p[p[3]] = 3$, index 3 is stable.
- Index 4: $p[4] = 4$. $p[p[4]] = p[4] = 4$. Since $p[p[4]] = 4$, index 4 is stable.
- Indices 5 and 6 are not stable.

Thus, $S(p) = 3$.

Given a permutation p of length n and a total number of rotations k , you must calculate the sum of the stable index counts for the first k rotations (starting from the 0-th rotation, which is p itself).

Your goal is to compute:

$$\sum_{i=0}^{k-1} S(\text{rot}^i(p))$$

Since k can be extremely large, a simple simulation of each rotation is not feasible.

Input

The first line contains two integers n and k ($1 \leq n \leq 2 \cdot 10^5$, $1 \leq k \leq 10^9$) — the length of the permutation and the total number of rotations to sum over.

The second line contains n distinct integers $p_1, p_2, \dots, p_n$ ($1 \leq p_i \leq n$), representing the elements of the initial permutation p .

Output

Print a single integer representing the calculated Total Stability Score.

Examples

standard input	standard output
6 3 3 6 1 4 2 5	6
4 5 1 2 3 4	12

Problem D. Mini-Max Permutation

Input file: `standard input`
Output file: `standard output`
Time limit: 1 second
Memory limit: 256 megabytes

You are given an integer n . Your task is to construct a permutation p of the integers from 1 to n .
Let's define the *cost* of any subarray $p[i \dots j]$ (the elements $p_i, p_{i+1}, \dots, p_j$) as:

$$\text{cost}(i, j) = \max(p_i, \dots, p_j) - \min(p_i, \dots, p_j) - (j - i + 1)$$

The *total cost* of the permutation p is the sum of costs over all possible subarrays:

$$\text{TotalCost}(p) = \sum_{1 \leq i \leq j \leq n} \text{cost}(i, j)$$

You must find a permutation p that minimizes this $\text{TotalCost}(p)$. If there are multiple such permutations that achieve the minimum possible cost, you must output the one which is lexicographically maximum.

Input

The first and only line of input contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$).

Output

Output n space-separated integers, representing the permutation $p_1, p_2, \dots, p_n$ that minimizes the total cost and is lexicographically maximum among all optimal permutations.

Examples

standard input	standard output
1	1
2	2 1

Problem E. Sheesh El Beesh

Input file: **standard input**
Output file: **standard output**
Time limit: 1 second
Memory limit: 256 megabytes

You are given a line of n dominoes, numbered 1 to n from left to right. Each domino i has a height h_i and an associated cost c_i to manually push it.

Your goal is to make all n dominoes fall.

The rules for falling are as follows:

- If you choose to manually push domino i , you incur a cost of c_i , and domino i falls.
- When a domino j falls, it may cause a chain reaction. If domino j has fallen and $j < n$, it will hit domino $j + 1$. If $h_j > h_{j+1}$, then domino $j + 1$ will also fall.
- This chain reaction continues. If $j + 1$ falls, it may cause $j + 2$ to fall, and so on.

Formally, if you manually push domino i , it falls. Then, for any $j \geq i$, if domino j has fallen and $j < n$ and $h_j > h_{j+1}$, then domino $j + 1$ will also fall.

You can choose to manually push any set of dominoes. The total cost is the sum of the costs of all dominoes you manually push. Find the minimum total cost required to make all n dominoes fall.

Input

The first line of input contains a single integer t ($1 \leq t \leq 1000$), the number of test cases. The description of each test case follows.

The first line of each test case contains a single integer n ($1 \leq n \leq 200,000$), the number of dominoes.

The second line contains n integers $h_1, h_2, \dots, h_n$ ($1 \leq h_i \leq 10^9$), the heights of the dominoes.

The third line contains n integers $c_1, c_2, \dots, c_n$ ($1 \leq c_i \leq 10^9$), the costs to push the dominoes.

The sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer on a new line: the minimum total cost to make all n dominoes fall.

Example

standard input	standard output
3	6
5	2
3 1 4 1 5	10
2 1 3 4 1	
10	
10 9 8 7 6 5 4 3 2 1	
2 3 1 4 5 1 2 3 4 5	
4	
1 2 3 4	
1 2 3 4	

Problem F. GCD $\leftrightarrow$ LCM

Input file: **standard input**
Output file: **standard output**
Time limit: 1.5 seconds
Memory limit: 256 megabytes

You are given an array of integers $a_1, a_2, \dots, a_n$.

A quadruple (i, j, k, l) is called *good* if:

- $1 \leq i, j, k, l \leq n$,
- all indices are pairwise distinct, and
- $\gcd(a_i, a_j) = \text{lcm}(a_k, a_l)$.

Your task is to count the number of ordered good quadruples.

Since the answer can be large, print it modulo $10^9 + 7$.

Input

The first line contains a single integer t ($1 \leq t \leq 10^5$) — the number of test cases.

Each test case consists of two lines:

- The first line contains an integer n ($4 \leq n \leq 10^5$).
- The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq n$).

It is guaranteed that the sum of n over all test cases does not exceed 10^5 .

Output

For each test case, output a single integer — the number of ordered good quadruples modulo $10^9 + 7$.

Example

standard input	standard output
2	4
4	24
4 3 1 1	
6	
2 2 2 2 3 5	

Note

In the first test case, the good quadruples of indices are: $(1, 2, 3, 4)$, $(1, 2, 4, 3)$, $(2, 1, 3, 4)$, $(2, 1, 4, 3)$ since $\gcd(4, 3) = 1$ and $\text{lcm}(1, 1) = 1$.

Problem G. Journey Around The World

Input file: **standard input**
Output file: **standard output**
Time limit: 1.5 seconds
Memory limit: 256 megabytes

Arwa is tired of studying. So to clear her head, she decides to go on a journey around the world. Her world consists of n cities, and originally every pair of different cities was connected by a bidirectional road. However, a small number of roads have been removed.

For any number of days k , a valid journey of length k is a sequence of cities $w_0, w_1, \dots, w_k$ such that for every i ($0 \leq i < k$) there exists a direct road between w_i and w_{i+1} . Note that cities and roads may be visited more than once during a journey. Two journeys of length k are considered different if there exists a j ($0 \leq j \leq k$) such that the j -th city is different in both journeys.

Arwa lives in city 1, so she will start her journey from there. She hasn't decided yet how many days her journey will last. So, Arwa wants to know: for every k from 1 up to $n - 1$, how many different journeys of length k can she make in her world? Since these numbers can be very large, output each count modulo 998 244 353.

Input

The first line contain two integers n, m ($2 \leq n \leq 2 \cdot 10^5, 0 \leq m \leq \min(\frac{n(n-1)}{2}, 200)$) — the number of cities, and the number of roads that have been removed.

Each of the next m lines contains two integers u, v ($1 \leq u, v \leq n, u \neq v$) — indicating that the road between u and v has been removed. It is guaranteed that all pairs of u and v are distinct.

Output

Print $n - 1$ space-separated integers $a_1, a_2, \dots, a_{n-1}$, where a_k is the number of journeys of length k that start from city 1, taken modulo 998 244 353.

Example

standard input	standard output
3 1 1 2	1 2

Problem H. Yasser and Arithmetic Sequences

Input file: **standard input**
Output file: **standard output**
Time limit: 1 second
Memory limit: 1024 megabytes

Yasser, a math enthusiast with a particular fondness for arithmetic sequences, has an interesting challenge. He has an array a of length n , and his objective is to determine if this array can be transformed into an arithmetic sequence through a series of allowed operations.

An arithmetic sequence is a list of numbers where the difference between consecutive terms remains constant. For instance, the sequence 3, 7, 11, 15 is an arithmetic sequence because the difference between each term is 4 ($7 - 3 = 11 - 7 = 15 - 11 = 4$).

Yasser can perform some operations on the array. In each operation, he can select an element at any index and either:

- 1 - Add an integer between 1 and 10 (inclusive) to the chosen element.
- 2 - Subtract an integer between 1 and 10 (inclusive) from the chosen element.
- 3 - Leave the element unchanged.

There's a crucial condition: Yasser can only modify each element in the array at most once.

Finally, Yasser is given an array of integers. Can you assist Yasser in determining if it's possible to transform the array into a perfect arithmetic sequence?

Input

The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases.

The first line of each test case contains an integer n ($1 \leq n \leq 10^5$) — the number of elements in a .

The second line of each test case contains n integers ($|a_i| \leq 10^9$).

It is guaranteed that the sum of n over test cases does not exceed 10^5 .

Output

If it is impossible to make an arithmetic progression with the described operations, print "NO" otherwise, print "YES".

You can output "YES" and "NO" in any case (for example, strings "yES", "yes" and "Yes" will be recognized as a positive response).

Example

standard input	standard output
3	YES
4	YES
3 6 8 12	NO
3	
1 10 50	
3	
1 10 70	

Problem I. Er7am El Tree

Input file: `standard input`
Output file: `standard output`
Time limit: 2 seconds
Memory limit: 1024 megabytes

Gohary loved a big tree behind his house, but he did a lot of operations and one day the tree unexpectedly exploded — not the good kind. Every edge fell off and scattered. Gohary and his friends gathered the loose edges and the lonely nodes and laid the edges down in a row on the ground.

To keep things tidy, each friend packed a consecutive run of edges into a box; when they stacked boxes for transport they either nested a box inside another for safe packing or put boxes side-by-side so nothing slipped. (They never made a messy partial overlap — that would have been chaotic.)

Now, Gohary wants to play with the pieces.

You are given the list of node values and the array of discarded edges (in the order they were placed). For a query, friends take the edges from position l to r in the array and reconnect them. After connecting edges l through r , if two nodes u and v are connected, consider the unique simple path between u and v . Along that path you can look at the sequence of node values. The answer to the query is the length of the longest contiguous subsegment of that path in which all node values are equal. If u and v are not connected after adding edges $l \dots r$, the answer is -1 .

Input

The first line contains two integers n and q ($2 \leq n \leq 2 \times 10^5$, $1 \leq q \leq 2 \times 10^5$) — the number of nodes and the number of queries.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^9$) — the values of the nodes.

Each of the next $n - 1$ lines contains two integers u_i and v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$) — the edges in the order they were placed on the ground. The edges form a valid tree (no duplicates, no self-loops).

Each of the next q lines contains four integers l , r , u , and v ($1 \leq l \leq r \leq n - 1$, $1 \leq u, v \leq n$, $u \neq v$) — a query asking about the longest equal-value subsegment on the path from u to v using edges l through r .

Output

For each query, output a single integer:

- If u and v are connected using edges $l \dots r$, output the length of the longest contiguous subsegment of equal values on the path from u to v .
- If u and v are not connected, output -1 .

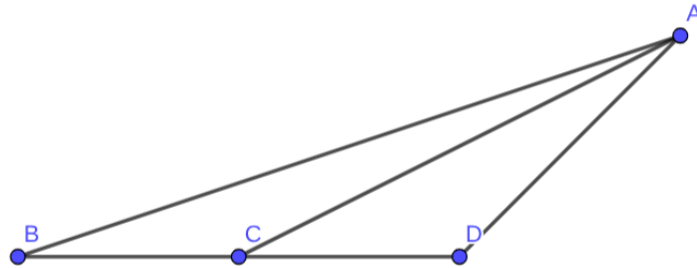
Examples

standard input	standard output
6 5 2 2 2 2 1 1 1 3 1 5 1 2 4 6 2 4 4 4 1 3 1 3 2 5 2 5 5 2 1 4 2 4 5 5 4 5	-1 2 2 -1 -1
10 5 6 5 1 13 9 18 13 8 18 16 1 2 4 9 2 3 4 6 6 8 6 7 8 10 3 5 2 4 7 7 10 5 3 9 10 8 9 9 10 8 8 8 4 2 4 9 1 5	-1 1 -1 -1 -1

Problem J. 7aseb El Triangle

Input file: `standard input`
Output file: `standard output`
Time limit: 1 second
Memory limit: 256 megabytes

Consider a triangle configuration where points B , C , and D lie on a straight line, and point A is above this line forming the triangles ABC and ACD .



Given:

- $BC = CD$ (the segments are equal in length)
- $\angle ABD = \alpha$ (the angle at vertex B in triangle ABD)
- $\angle ACD = \beta$ (the angle at vertex C in triangle ACD)

Your task is to find the measure of $\angle ADC$ (the angle at vertex D in triangle ACD).

Input

The only line contains two integers α and β ($1 \leq \alpha, \beta < 90$) — the measures of angles $\angle ABD$ and $\angle ACD$ in degrees.

It is guaranteed that the input corresponds to a valid geometric configuration.

Output

Print a single real number — the measure of angle $\angle ADC$ in degrees.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6} . Formally, let your answer be a , and the jury's answer be b . Your answer is considered correct if $\frac{|a-b|}{\max(1, |b|)} \leq 10^{-6}$.

Examples

standard input	standard output
2 6	5.9422389569
2 10	3.3094179971

Note

Notice that the triangles need not be obtuse-angled.

Problem K. Good Subarrays

Input file: `standard input`
Output file: `standard output`
Time limit: 1 second
Memory limit: 256 megabytes

You are given an array a consisting of n **distinct** integers.

A array b is called *good* if it can be sorted in non-decreasing order by performing the following operation **at most once**:

- Choose one or more **non-intersecting subarrays** of the array b , and reverse each of them.

In other words, you may select several disjoint segments within the array and reverse each of them, at most once in total. If after these reversals the subarray becomes sorted, it is considered good.

Your task is to determine the number of good subarrays in the given array.

Input

The first line contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the length of the array a .

The second line contains n distinct integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^9$) — the elements of the array.

Output

Print a single integer — the number of good subarrays.

Example

standard input	standard output
3 2 3 1	5

Problem L. Extended Modulo Queries

Input file: `standard input`
Output file: `standard output`
Time limit: 10 seconds
Memory limit: 256 megabytes

You are given an array A of length N .

We define the function:

$$f(l, r, x) = \sum_{i=l}^r (A_i \bmod x),$$

where $A_i \bmod x$ denotes the remainder when A_i is divided by x .

You will be given Q queries. Each query is described by four integers l, r, x_l, x_r ($1 \leq l \leq r \leq N$, $1 \leq x_l \leq x_r \leq 5 \cdot 10^4$).

For each query, compute:

$$\sum_{x=x_l}^{x_r} f(l, r, x).$$

Input

The first line contains two integers N and Q ($1 \leq N, Q \leq 5 \cdot 10^4$) — the length of the array and the number of queries.

The second line contains N integers $A_1, A_2, \dots, A_N$ ($1 \leq A_i \leq 5 \cdot 10^4$) — the elements of the array.

Each of the next Q lines contains four integers l, r, x_l, x_r .

Output

Print Q integers — the answers to the queries in the same order as they are given.

Examples

standard input	standard output
4 1 5 2 8 3 1 3 2 4	10
5 2 10 4 1 7 3 1 5 5 5 2 4 1 3	10 5

Problem M. Ultimate K-Query

Input file: `standard input`
Output file: `standard output`
Time limit: 6 seconds
Memory limit: 1024 megabytes

You are given an array a of n integers. You must process q queries of the form (l, r, k) , where you need to count how many distinct values appear exactly k times in the subarray $a[l..r]$.

The queries are **online**: the actual parameters of each query depend on the answer to the previous query. Specifically, if last is the answer to the previous query (with $\text{last} = 0$ initially), then the actual query parameters are:

$$l' = ((l \oplus \text{last}) \bmod n) + 1$$

$$r' = ((r \oplus \text{last}) \bmod n) + 1$$

where $\oplus$ denotes the XOR operation. If $l' > r'$, swap them.

Input

The first line contains two integers n and q ($1 \leq n, q \leq 2 * 10^5$) — the size of the array and the number of queries.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq n$) — the elements of the array.

Each of the next q lines contains three integers l, r , and k ($1 \leq l, r, k \leq n$) — the encoded query parameters.

Output

For each query, output a single integer — the number of distinct values that appear exactly k times in the subarray $a[l'..r']$.

Examples

standard input	standard output
1 1 1 1 1 1	1
9 8 3 3 3 3 4 4 4 4 5 1 7 3 2 4 1 1 9 3 8 8 1 5 8 4 2 4 2 7 7 1 9 9 1	1 1 0 1 1 1 1 1 1

Problem N. Ma3rofa 2lsra7a

Input file: `standard input`
Output file: `standard output`
Time limit: 1 second
Memory limit: 1024 megabytes

Gohary and Tawfik are arguing about number theory. Gohary claims that any number can be expressed as the sum of three perfect squares, but Tawfik disagrees and challenges him to prove it.

Given a positive integer x , determine whether there exist integers a , b , and c such that:

$$a^2 + b^2 + c^2 = x$$

where $1 \leq a \leq b \leq c$.

Help Gohary settle this debate by determining if such a representation exists!

Input

The input consists of a single integer x ($1 \leq x \leq 10^8$) — the target sum.

Output

Output YES if there exist integers a , b , c with $1 \leq a \leq b \leq c$ such that $a^2 + b^2 + c^2 = x$.

Output NO otherwise.

Note

Are you Waiting for samples !!! Read the title

Problem O. Ya Masa2 El Geometry

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

Hussein, Kamal, Tawfik, and Gohary are setting up a new surveillance system in a large open field. They've marked n important locations on the map, each represented as a point with coordinates (x_i, y_i) . Now, they must install at most two circular watchtowers so that every location is under surveillance — that is, every marked point lies inside or on the boundary of at least one of the circles.

A watchtower can be built anywhere on the map (not necessarily at one of the given locations) and can have any non-negative radius. A watchtower with radius 0 can only cover a single point located exactly at its center.

The team wants to minimize the total coverage effort, defined as the sum of the radii of the two watchtowers used. If one watchtower is enough to cover all locations, the other one may have a radius of 0.

Can you help them determine the minimum possible value of $r_1 + r_2$, where r_1 and r_2 are the radii of the two watchtowers?

Input

The first line contains an integer n ($2 \leq n \leq 300$) — the number of marked locations.

Each of the next n lines contains two integers x_i, y_i ($-10^9 \leq x_i, y_i \leq 10^9$) — the coordinates of a location.

Output

Output a single real number — the minimum possible value of $r_1 + r_2$.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6} . Formally, let your answer be a and the jury's answer be b . Your answer is considered correct if $\frac{|a-b|}{\max(1, |b|)} \leq 10^{-6}$.

Examples

standard input	standard output
10 907 -970 -743 -745 -860 -454 -346 -318 162 366 -650 671 -110 -830 914 864 609 -443 175 -52	1131.0537122524
3 -865 -635 -680 -783 -360 510	118.4577983925