VK Cup 2016 - Round 1

A. Bear and Displayed Friends

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Limak is a little polar bear. He loves connecting with other bears via social networks. He has n friends and his relation with the i-th of them is described by a unique integer t_i. The bigger this value is, the better the friendship is. No two friends have the same value t_i.

Spring is starting and the Winter sleep is over for bears. Limak has just woken up and logged in. All his friends still sleep and thus none of them is online. Some (maybe all) of them will appear online in the next hours, one at a time.

The system displays friends who are online. On the screen there is space to display at most k friends. If there are more than k friends online then the system displays only k best of them — those with biggest t_i.

Your task is to handle queries of two types:

- "1 id" — Friend id becomes online. It's guaranteed that he wasn't online before.
- "2 id" — Check whether friend id is displayed by the system. Print "YES" or "NO" in a separate line.

Are you able to help Limak and answer all queries of the second type?

Input

The first line contains three integers n, k and q ($1 \le n, q \le 150\,000$, $1 \le k \le min(6, n)$) — the number of friends, the maximum number of displayed online friends and the number of queries, respectively.

The second line contains n integers $t_1, t_2, ..., t_n$ ($1 \le t_i \le 10^9$) where t_i describes how good is Limak's relation with the i-th friend.

The i-th of the following q lines contains two integers $type_i$ and id_i ($1 \le type_i \le 2$, $1 \le id_i \le n$) — the i-th query. If $type_i = 1$ then a friend id_i becomes online. If $type_i = 2$ then you should check whether a friend id_i is displayed.

It's guaranteed that no two queries of the first type will have the same id_i becuase one friend can't become online twice. Also, it's guaranteed that at least one query will be of the second type ($type_i = 2$) so the output won't be empty.

Output

For each query of the second type print one line with the answer — "YES" (without quotes) if the given friend is displayed and "NO" (without quotes) otherwise.

Examples

input

```
4 2 8
300 950 500 200
1 3
2 4
2 3
1 1
1 2
2 1
2 2
2 3
```

output

```
NO
YES
NO
YES
YES
```

input

```
6 3 9
50 20 51 17 99 24
1 3
1 4
1 5
1 2
2 4
2 2
1 1
2 4
2 3
```

output

```
NO
YES
NO
YES
```

Note

In the first sample, Limak has 4 friends who all sleep initially. At first, the system displays nobody because nobody is online. There are the following 8 queries:

1. "1 3" — Friend 3 becomes online.
2. "2 4" — We should check if friend 4 is displayed. He isn't even online and thus we print "NO".
3. "2 3" — We should check if friend 3 is displayed. Right now he is the only friend online and the system displays him. We should print "YES".
4. "1 1" — Friend 1 becomes online. The system now displays both friend 1 and friend 3.
5. "1 2" — Friend 2 becomes online. There are 3 friends online now but we were given $k = 2$ so only two friends can be displayed. Limak has worse relation with friend 1 than with other two online friends ($t_1 < t_2, t_3$) so friend 1 won't be displayed
6. "2 1" — Print "NO".
7. "2 2" — Print "YES".
8. "2 3" — Print "YES".

B. Bear and Forgotten Tree 3

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A tree is a connected undirected graph consisting of n vertices and $n - 1$ edges. Vertices are numbered 1 through n.

Limak is a little polar bear and Radewoosh is his evil enemy. Limak once had a tree but Radewoosh stolen it. Bear is very sad now because he doesn't remember much about the tree — he can tell you only three values n, d and h:

- The tree had exactly n vertices.
- The tree had diameter d. In other words, d was the biggest distance between two vertices.
- Limak also remembers that he once rooted the tree in vertex 1 and after that its height was h. In other words, h was the biggest distance between vertex 1 and some other vertex.

The distance between two vertices of the tree is the number of edges on the simple path between them.

Help Limak to restore his tree. Check whether there exists a tree satisfying the given conditions. Find any such tree and print its edges in any order. It's also possible that Limak made a mistake and there is no suitable tree – in this case print "-1".

Input

The first line contains three integers n, d and h ($2 \le n \le 100\,000$, $1 \le h \le d \le n - 1$) — the number of vertices, diameter, and height after rooting in vertex 1, respectively.

Output

If there is no tree matching what Limak remembers, print the only line with "-1" (without the quotes).

Otherwise, describe any tree matching Limak's description. Print $n - 1$ lines, each with two space-separated integers – indices of vertices connected by an edge. If there are many valid trees, print any of them. You can print edges in any order.

Examples

input
5 3 2
output
1 2 1 3 3 4 3 5

input
8 5 2
output
-1

input
8 4 2
output
4 8 5 7 2 3 8 1 2 1 5 6 1 5

Note

Below you can see trees printed to the output in the first sample and the third sample.

C. Bear and Polynomials

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Limak is a little polar bear. He doesn't have many toys and thus he often plays with polynomials.

He considers a polynomial *valid* if its degree is n and its coefficients are integers not exceeding k by the absolute value. More formally:

Let $a_0, a_1, ..., a_n$ denote the coefficients, so . Then, a polynomial $P(x)$ is valid if all the following conditions are satisfied:

- a_i is integer for every i;
- $|a_i| \leq k$ for every i;
- $a_n \neq 0$.

Limak has recently got a valid polynomial P with coefficients $a_0, a_1, a_2, ..., a_n$. He noticed that $P(2) \neq 0$ and he wants to change it. He is going to change one coefficient to get a **valid** polynomial Q of degree n that $Q(2) = 0$. Count the number of ways to do so. You should count two ways as a distinct if coefficients of target polynoms differ.

Input

The first line contains two integers n and k ($1 \leq n \leq 200\,000, 1 \leq k \leq 10^9$) — the degree of the polynomial and the limit for absolute values of coefficients.

The second line contains $n + 1$ integers $a_0, a_1, ..., a_n$ ($|a_i| \leq k, a_n \neq 0$) — describing a **valid** polynomial . It's guaranteed that $P(2) \neq 0$.

Output

Print the number of ways to change one coefficient to get a valid polynomial Q that $Q(2) = 0$.

Examples

input
3 1000000000 10 -9 -3 5
output
3

input
3 12 10 -9 -3 5
output
2

input
2 20 14 -7 19
output
0

Note

In the first sample, we are given a polynomial $P(x) = 10 - 9x - 3x^2 + 5x^3$.

Limak can change one coefficient in three ways:

1. He can set $a_0 = -10$. Then he would get $Q(x) = -10 - 9x - 3x^2 + 5x^3$ and indeed $Q(2) = -10 - 18 - 12 + 40 = 0$.
2. Or he can set $a_2 = -8$. Then $Q(x) = 10 - 9x - 8x^2 + 5x^3$ and indeed $Q(2) = 10 - 18 - 32 + 40 = 0$.
3. Or he can set $a_1 = -19$. Then $Q(x) = 10 - 19x - 3x^2 + 5x^3$ and indeed $Q(2) = 10 - 38 - 12 + 40 = 0$.

In the second sample, we are given the same polynomial. This time though, k is equal to 12 instead of 10^9. Two first of ways listed above are still valid but in the third way we would get $|a_1| > k$ what is not allowed. Thus, the answer is 2 this time.

D. Bear and Contribution

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Codeforces is a wonderful platform and one its feature shows how much someone contributes to the community. Every registered user has *contribution* — an integer number, not necessarily positive. There are n registered users and the i-th of them has contribution t_i.

Limak is a little polar bear and he's new into competitive programming. He doesn't even have an account in Codeforces but he is able to upvote existing blogs and comments. We assume that every registered user has infinitely many blogs and comments.

- Limak can spend b minutes to read one blog and upvote it. Author's contribution will be increased by 5.
- Limak can spend c minutes to read one comment and upvote it. Author's contribution will be increased by 1.

Note that it's possible that Limak reads blogs faster than comments.

Limak likes ties. He thinks it would be awesome to see a tie between at least k registered users. To make it happen he is going to spend some time on reading and upvoting. After that, there should exist an integer value x that at least k registered users have contribution exactly x.

How much time does Limak need to achieve his goal?

Input

The first line contains four integers n, k, b and c ($2 \le k \le n \le 200\,000$, $1 \le b, c \le 1000$) — the number of registered users, the required minimum number of users with the same contribution, time needed to read and upvote a blog, and time needed to read and upvote a comment, respectively.

The second line contains n integers $t_1, t_2, \ldots, t_n$ ($|t_i| \le 10^9$) where t_i denotes contribution of the i-th registered user.

Output

Print the minimum number of minutes Limak will spend to get a tie between at least k registered users.

Examples

input
4 3 100 30 12 2 6 1
output
220

input
4 3 30 100 12 2 6 1
output
190

input
6 2 987 789 -8 42 -4 -65 -8 -8
output
0

Note

In the first sample, there are 4 registered users and Limak wants a tie between at least 3 of them. Limak should behave as follows.

- He spends 100 minutes to read one blog of the 4-th user and increase his contribution from 1 to 6.
- Then he spends $4 \cdot 30 = 120$ minutes to read four comments of the 2-nd user and increase his contribution from 2 to 6 (four times it was increaded by 1).

In the given scenario, Limak spends $100 + 4 \cdot 30 = 220$ minutes and after that each of users $2, 3, 4$ has contribution 6.

In the second sample, Limak needs 30 minutes to read a blog and 100 minutes to read a comment. This time he can get 3 users with contribution equal to 12 by spending $100 + 3 \cdot 30 = 190$ minutes:

- Spend $2 \cdot 30 = 60$ minutes to read two blogs of the 1-st user to increase his contribution from 2 to 12.
- Spend $30 + 100$ minutes to read one blog and one comment of the 3-rd user. His contribution will change from 6 to $6 + 5 + 1 = 12$.

E. Bear and Paradox

time limit per test: 3.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Limak is a big polar bear. He prepared n problems for an algorithmic contest. The i-th problem has **initial** score p_i. Also, testers said that it takes t_i minutes to solve the i-th problem. Problems aren't necessarily sorted by difficulty and maybe harder problems have smaller initial score but it's too late to change it — Limak has already announced initial scores for problems. Though it's still possible to adjust the speed of losing points, denoted by c in this statement.

Let T denote the total number of minutes needed to solve all problems (so, $T = t_1 + t_2 + ... + t_n$). The contest will last exactly T minutes. So it's just enough to solve all problems.

Points given for solving a problem decrease linearly. Solving the i-th problem after x minutes gives exactly points, where is some real constant that Limak must choose.

Let's assume that c is fixed. During a contest a participant chooses some order in which he or she solves problems. There are $n!$ possible orders and each of them gives some total number of points, not necessarily integer. We say that an order is *optimal* if it gives the maximum number of points. In other words, the total number of points given by this order is greater or equal than the number of points given by any other order. It's obvious that there is at least one optimal order. However, there may be more than one optimal order.

Limak assumes that every participant will properly estimate t_i at the very beginning and will choose some optimal order. He also assumes that testers correctly predicted time needed to solve each problem.

For two distinct problems i and j such that $p_i < p_j$ Limak wouldn't be happy to see a participant with strictly more points for problem i than for problem j. He calls such a situation a *paradox*.

It's not hard to prove that there will be no paradox for $c = 0$. The situation may be worse for bigger c. What is the maximum real value c (remember that) for which there is no paradox possible, that is, there will be no paradox for **any optimal order** of solving problems?

It can be proved that the answer (the maximum c as described) always exists.

Input

The first line contains one integer n $(2 \le n \le 150\,000)$ — the number of problems.

The second line contains n integers $p_1, p_2, ..., p_n$ $(1 \le p_i \le 10^8)$ — initial scores.

The third line contains n integers $t_1, t_2, ..., t_n$ $(1 \le t_i \le 10^8)$ where t_i is the number of minutes needed to solve the i-th problem.

Output

Print one real value on a single line — the maximum value of c that and there is no optimal order with a paradox. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Namely: let's assume that your answer is a, and the answer of the jury is b. The checker program will consider your answer correct if .

Examples

input
3 4 3 10 1 1 8
output
0.62500000000

input
4 7 20 15 10 7 20 15 10
output
0.31901840491

input
2 10 20 10 1
output
1.00000000000

Note

In the first sample, there are 3 problems. The first is $(4, 1)$ (initial score is 4 and required time is 1 minute), the second problem is $(3, 1)$ and the third one is $(10, 8)$. The total time is $T = 1 + 1 + 8 = 10$.

Let's show that there is a paradox for $c = 0.7$. Solving problems in order $1, 2, 3$ turns out to give the best total score, equal to the sum of:

1. solved 1 minute after the start:
2. solved 2 minutes after the start:
3. solved 10 minutes after the start:

So, this order gives $3.72 + 2.58 + 3 = 9.3$ points in total and this is the only optimal order (you can calculate total scores for other 5 possible orders too see that they are lower). You should check points for problems 1 and 3 to see a paradox. There is $4 < 10$ but $3.72 > 3$. It turns out that there is no paradox for $c = 0.625$ but there is a paradox for any bigger c.

In the second sample, all 24 orders are optimal.

In the third sample, even for $c = 1$ there is no paradox.

F. Bear and Chemistry

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Limak is a smart brown bear who loves chemistry, reactions and transforming elements.

In Bearland (Limak's home) there are n elements, numbered 1 through n. There are also special machines, that can transform elements. Each machine is described by two integers a_i, b_i representing two elements, not necessarily distinct. One can use a machine either to transform an element a_i to b_i or to transform b_i to a_i. Machines in Bearland aren't very resistant and each of them can be used **at most once**. It is possible that $a_i = b_i$ and that many machines have the same pair a_i, b_i.

Radewoosh is Limak's biggest enemy and rival. He wants to test Limak in the chemistry. They will meet tomorrow and both of them will bring all their machines. Limak has m machines but he doesn't know much about his enemy. They agreed Radewoosh will choose two distinct elements, let's denote them as x and y. Limak will be allowed to use both his and Radewoosh's machines. He may use zero or more (maybe even all) machines to achieve the goal, each machine at most once. Limak will start from an element x and his task will be to first get an element y and then to again get an element x — **then we say that he succeeds**. After that Radewoosh would agree that Limak knows the chemistry (and Radewoosh would go away).

Radewoosh likes some particular non-empty set of favorite elements and he will choose x, y from that set. Limak doesn't know exactly which elements are in the set and also he doesn't know what machines Radewoosh has. Limak has heard q gossips (queries) though and each of them consists of Radewoosh's machines and favorite elements. For each gossip Limak wonders if he would be able to **succeed** tomorrow for every pair x, y chosen from the set of favorite elements. If yes then print "`YES`" (without the quotes). But if there exists a pair (x, y) from the given set that Limak wouldn't be able to succeed then you should print "`NO`" (without the quotes).

Input

The first line contains three integers n, m and q ($1 \le n, q \le 300\,000$, $0 \le m \le 300\,000$) — the number of elements, the number of Limak's machines and the number of gossips, respectively.

Each of the next m lines contains two integers a_i and b_i ($1 \le a_i, b_i \le n$) describing one of Limak's machines.

Then, the description of q gossips follows.

The first line of the description of the i-th gossip contains two integers n_i and m_i ($1 \le n_i \le 300\,000$, $0 \le m_i \le 300\,000$). The second line contains n_i **distinct** integers $x_{i,1}, x_{i,2}, \ldots, x_{i,n_i}$ ($1 \le x_{i,j} \le n$) — Radewoosh's favorite elements in the i-th gossip. Note that $n_i = 1$ is allowed, in this case there are no pairs of distinct elements, so Limak automatically wins (the answer is "`YES`"). Then m_i lines follow, each containing two integers $a_{i,j}, b_{i,j}$ ($1 \le a_{i,j}, b_{i,j}$) describing one of Radewoosh's machines in the i-th gossip.

The sum of n_i over all gossips won't exceed $300\,000$. Also, the sum of m_i over all gossips won't exceed $300\,000$.

Important: Because we want you to process the gossips online, in order to know the elements in Radewoosh's favorite set and elements that his machines can transform, for on each number that denotes them in the input you should use following function:

```
int rotate(int element)
{
   element=(element+R)%n;

   if (element==0) {
       element=n;
   }

   return element;
}
```

where R is initially equal to 0 and is increased by the number of the query any time the answer is "`YES`". Queries are numbered starting with 1 in the order they appear in the input.

Output

You should print q lines. The i-th of them should contain "`YES`" (without quotes) if for the i-th gossip **for each pair** of elements x and y (in the set $x_{i,1}, x_{i,2}, \ldots, x_{i,n_i}$) Limak is able to succeed. Otherwise you should print "`NO`" (without quotes).

Examples

input
6 5 4 1 2 2 3 3 4 2 4 5 6 2 0 4 2 2 1 6 2 3 4

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

output

```
YES
NO
YES
YES
```

input

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

output

```
NO
YES
```

Note

Lets look at first sample:

In first gossip Radewoosh's favorite set is $\{4, 2\}$ and he has no machines. Limak can tranform element 4 into 2 (so half of a task is complete) and then 2 into 3, and 3 into 4. Answer is "YES", so R is increased by 1.

In second gossip set in the input is denoted by $\{6, 2\}$ and machine by $(3, 4)$, but R is equal to 1, so set is $\{1, 3\}$ and machine is $(4, 5)$. Answer is "NO", so R isn't changed.

In third gossip set $\{6, 4, 3\}$ and machines $(2, 5)$ and $(4, 6)$ are deciphered to be $\{1, 5, 4\}$, $(3, 6)$ and $(5, 1)$.

Consider Radewoosh's choices:

- If he chooses elements 1 and 5, then Limak is able to transform 1 into 5, then 6 into 3, 3 into 2 and 2 into 1.
- If he chooses elements 5 and 4, then Limak is able to transform 5 into 6, 6 into 3, 3 into 4 (half way already behind him), 4 into 2, 2 into 1, 1 into 5.
- If he chooses elements 1 and 4, then Limak is able to transform 1 into 2, 2 into 4, 4 into 3, 3 into 6, 6 into 5 and 5 into 1.

So Limak is able to execute task. Answer is "YES" and R is increased by 3 (it's equal to 4 now).

In last gossip $\{1, 2\}$ and $(1, 2)$ are deciphered to be $\{5, 6\}$ and $(5, 6)$. Now there are 2 machines $(5, 6)$ so Limak is able to execute task again.