A. Bear and Displayed Friends

time limit per test: 2 seconds
memory limit per test: 256 megabytes

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".

Problem B. Laundro, Matt

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	512 megabytes

The original problem tests were splitted to contain at most 5 testcases.

Matt Laundro is about to engage in his favourite activity — doing laundry! He's brought L indistinguishable loads of laundry to his local laundromat, which has N washing machines and M dryers. The i-th washing machine takes W_i minutes to wash one load of laundry, and each dryer takes D minutes to dry a load of laundry. At any point in time, each machine may only be processing at most one load of laundry.

As one might expect, Matt wants to wash and then dry each of his L loads of laundry. Each load of laundry will go through the following steps in order:

1. A non-negative amount of time after Matt arrives at the laundromat, Matt places the load in an unoccupied washing machine i
2. W_i minutes later, he removes the load from the washing machine, placing it in a temporary holding basket (which has unlimited space)
3. A non-negative amount of time later, he places the load in an unoccupied dryer
4. D minutes later, he removes the load from the dryer

Matt can instantaneously add laundry to or remove laundry from a machine. Help Matt minimize the amount of time (in minutes after he arrives at the laundromat) after which he can be done drying all L loads of laundry!

Input

Input begins with an integer T ($1 \le T \le 5$), the number of times Matt goes to the laundromat. For each trip to the laundromat, there is first a line containing the space-separated integers L ($1 \le L \le 10^6$), N ($1 \le N \le 10^5$), M ($1 \le M \le 10^9$), and D ($1 \le D \le 10^9$) in that order. After that is a line containing N space-separated integers, the i-th of which is W_i ($1 \le Wi \le 10^9$).

Output

For the i-th trip, print a line containing "`Case #i: `" followed by the minimum time it will take Matt to finish his laundry.

Examples

standard input	standard output

5
1 1 1 34
1200
2 3 2 10
100 10 1
3 3 3 3
1 2 3
4 2 2 7
5 8
999 1 999 6
3
	Case #1: 1234
	Case #2: 12
	Case #3: 5
	Case #4: 22
	Case #5: 3003

Note

In the first case, Matt has just one load of laundry. He washes it for 1200 minutes, and dries it for 34 minutes. In the second case, Matt uses the 1-minute washer for both loads of laundry. The second load finishes at the 2-minute mark, so it finishes drying at the 12-minute mark.

C. Generating Sets

time limit per test: 2 seconds
memory limit per test: 256 megabytes

You are given a set Y of n **distinct** positive integers $y_1, y_2, ..., y_n$.

Set X of n **distinct** positive integers $x_1, x_2, ..., x_n$ is said to *generate* set Y if one can transform X to Y by applying some number of the following two operation to integers in X:

1. Take any integer x_i and multiply it by two, i.e. replace x_i with $2 \cdot x_i$.
2. Take any integer x_i, multiply it by two and add one, i.e. replace x_i with $2 \cdot x_i + 1$.

Note that integers in X are not required to be distinct after each operation.

Two sets of distinct integers X and Y are equal if they are equal as sets. In other words, if we write elements of the sets in the array in the increasing order, these arrays would be equal.

Note, that any set of integers (or its permutation) generates itself.

You are given a set Y and have to find a set X that generates Y and the **maximum element of X is mininum possible**.

Input

The first line of the input contains a single integer n ($1 \le n \le 50\,000$) — the number of elements in Y.

The second line contains n integers $y_1, ..., y_n$ ($1 \le y_i \le 10^9$), that are guaranteed to be distinct.

Output

Print n integers — set of distinct integers that generate Y and the maximum element of which is minimum possible. If there are several such sets, print any of them.

Examples

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

input
```
6
15 14 3 13 1 12
```
output
```
12 13 14 7 3 1
```

input
```
6
9 7 13 17 5 11
```
output
```
4 5 2 6 3 1
```

D. Destroying Array

time limit per test: 1 second

memory limit per test: 256 megabytes

You are given an array consisting of n non-negative integers $a_1, a_2, ..., a_n$.

You are going to destroy integers in the array one by one. Thus, you are given the permutation of integers from 1 to n defining the order elements of the array are destroyed.

After each element is destroyed you have to find out the segment of the array, such that it contains no destroyed elements and the sum of its elements is maximum possible. The sum of elements in the empty segment is considered to be 0.

Input

The first line of the input contains a single integer n ($1 \le n \le 100\,000$) — the length of the array.

The second line contains n integers $a_1, a_2, ..., a_n$ ($0 \le a_i \le 10^9$).

The third line contains a permutation of integers from 1 to n — the order used to destroy elements.

Output

Print n lines. The i-th line should contain a single integer — the maximum possible sum of elements on the segment containing no destroyed elements, after first i operations are performed.

Examples

input
```
4
1 3 2 5
3 4 1 2
```
output
```
5
4
3
0
```

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

input
```
8
5 5 4 4 6 6 5 5
5 2 8 7 1 3 4 6
```
output
```
18
16
11
8
8
6
6
0
```

Note

Consider the first sample:

1. Third element is destroyed. Array is now $1\ 3\ *\ 5$. Segment with maximum sum 5 consists of one integer 5.
2. Fourth element is destroyed. Array is now $1\ 3\ *\ *$. Segment with maximum sum 4 consists of two integers $1\ 3$.
3. First element is destroyed. Array is now $*\ 3\ *\ *$. Segment with maximum sum 3 consists of one integer 3.
4. Last element is destroyed. At this moment there are no valid nonempty segments left in this array, so the answer is equal to 0.

E. Cubes

time limit per test: 3 seconds

memory limit per test: 256 megabytes

Once Vasya and Petya assembled a figure of m cubes, each of them is associated with a number between 0 and $m - 1$ (inclusive, each number appeared exactly once). Let's consider a coordinate system such that the OX is the ground, and the OY is directed upwards. Each cube is associated with the coordinates of its lower left corner, these coordinates are integers for each cube.

The figure turned out to be stable. This means that for any cube that is not on the ground, there is at least one cube under it such that those two cubes touch **by a side or a corner**. More formally, this means that for the cube with coordinates (x, y) either $y = 0$, or there is a cube with coordinates $(x - 1, y - 1)$, $(x, y - 1)$ or $(x + 1, y - 1)$.

Now the boys want to disassemble the figure and put all the cubes in a row. In one step the cube is removed from the figure and being put to the right of the blocks that have already been laid. The guys remove the cubes in such order that the figure remains stable. To make the process more interesting, the guys decided to play the following game. The guys take out the cubes from the figure in turns. It is easy to see that after the figure is disassembled, the integers written on the cubes form a number, written in the m-ary positional numerical system (possibly, with a leading zero). Vasya wants the resulting number to be maximum possible, and Petya, on the contrary, tries to make it as small as possible. Vasya starts the game.

Your task is to determine what number is formed after the figure is disassembled, if the boys play optimally. Determine the remainder of the answer modulo $10^9 + 9$.

Input

The first line contains number m ($2 \le m \le 10^5$).

The following m lines contain the coordinates of the cubes x_i, y_i ($-10^9 \le x_i \le 10^9$, $0 \le y_i \le 10^9$) in ascending order of numbers written on them. It is guaranteed that the original figure is stable.

No two cubes occupy the same place.

Output

In the only line print the answer to the problem.

Examples

input

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

output

```
19
```

input

```
5
0 0
0 1
0 2
0 3
0 4
```

output

```
2930
```

F. Alphabet Permutations

time limit per test: 1 second

memory limit per test: 512 megabytes

You are given a string s of length n, consisting of first k lowercase English letters.

We define a c-*repeat* of some string q as a string, consisting of c copies of the string q. For example, string "`acbacbacbacb`" is a 4-repeat of the string "`acb`".

Let's say that string a contains string b as a subsequence, if string b can be obtained from a by erasing some symbols.

Let p be a string that represents some permutation of the first k lowercase English letters. We define function $d(p)$ as the smallest integer such that a $d(p)$-repeat of the string p contains string s as a subsequence.

There are m operations of one of two types that can be applied to string s:

1. Replace all characters at positions from l_i to r_i by a character c_i.
2. For the given p, that is a permutation of first k lowercase English letters, find the value of function $d(p)$.

All operations are performed sequentially, in the order they appear in the input. Your task is to determine the values of function $d(p)$ for all operations of the second type.

Input

The first line contains three positive integers n, m and k ($1 \le n \le 200\,000$, $1 \le m \le 20000$, $1 \le k \le 10$) — the length of the string s, the number of operations and the size of the alphabet respectively. The second line contains the string s itself.

Each of the following lines m contains a description of some operation:

1. Operation of the first type starts with 1 followed by a triple l_i, r_i and c_i, that denotes replacement of all characters at positions from l_i to r_i by character c_i ($1 \le l_i \le r_i \le n$, c_i is one of the first k lowercase English letters).
2. Operation of the second type starts with 2 followed by a permutation of the first k lowercase English letters.

Output

For each query of the second type the value of function $d(p)$.

Examples

input

```
7 4 3
abacaba
1 3 5 b
2 abc
1 4 4 c
2 cba
```

output

```
6
5
```

Note

After the first operation the string s will be `abbbbba`.

In the second operation the answer is 6-repeat of `abc`: `ABcaBcaBcaBcaBcAbc`.

After the third operation the string s will be `abbcbba`.

In the fourth operation the answer is 5-repeat of `cba`: `cbAcBacBaCBacBA`.

Uppercase letters means the occurrences of symbols from the string s.