Codeforces Round #500 (Div. 1) [based on EJOI]

A. Photo of The Sky

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Pavel made a photo of his favourite stars in the sky. His camera takes a photo of all points of the sky that belong to some rectangle with sides parallel to the coordinate axes.

Strictly speaking, it makes a photo of all points with coordinates (x, y), such that $x_1 \le x \le x_2$ and $y_1 \le y \le y_2$, where (x_1, y_1) and (x_2, y_2) are coordinates of the left bottom and the right top corners of the rectangle being photographed. The area of this rectangle can be zero.

After taking the photo, Pavel wrote down coordinates of n of his favourite stars which appeared in the photo. These points are not necessarily distinct, there can be multiple stars in the same point of the sky.

Pavel has lost his camera recently and wants to buy a similar one. Specifically, he wants to know the dimensions of the photo he took earlier. Unfortunately, the photo is also lost. His notes are also of not much help; numbers are written in random order all over his notepad, so it's impossible to tell which numbers specify coordinates of which points.

Pavel asked you to help him to determine what are the possible dimensions of the photo according to his notes. As there are multiple possible answers, find the dimensions with the minimal possible area of the rectangle.

Input

The first line of the input contains an only integer n ($1 \le n \le 100\,000$), the number of points in Pavel's records.

The second line contains $2 \cdot n$ integers a_1, a_2, ..., $a_{2 \cdot n}$ ($1 \le a_i \le 10^9$), coordinates, written by Pavel in some order.

Output

Print the only integer, the minimal area of the rectangle which could have contained all points from Pavel's records.

Examples

input

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

output

```
1
```

input

```
3
5 8 5 5 7 5
```

output

```
0
```

Note

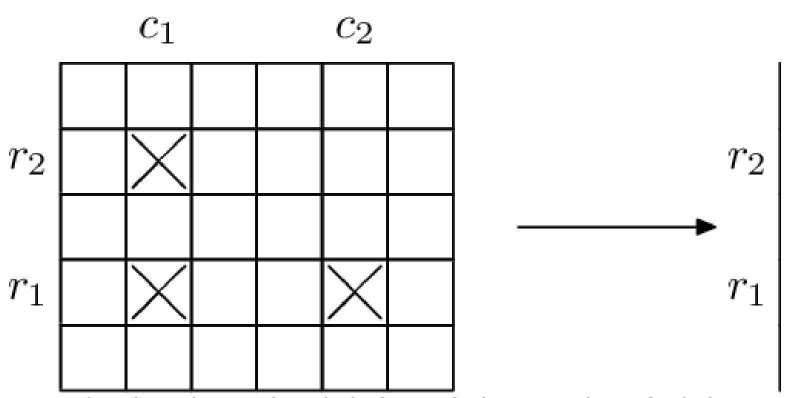
In the first sample stars in Pavel's records can be $(1, 3)$, $(1, 3)$, $(2, 3)$, $(2, 4)$. In this case, the minimal area of the rectangle, which contains all these points is 1 (rectangle with corners at $(1, 3)$ and $(2, 4)$).

B. Chemical table

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Innopolis University scientists continue to investigate the periodic table. There are $n \cdot m$ known elements and they form a periodic table: a rectangle with n rows and m columns. Each element can be described by its coordinates (r, c) ($1 \le r \le n$, $1 \le c \le m$) in the table.

Recently scientists discovered that for every four different elements in this table that form a rectangle with sides parallel to the sides of the table, if they have samples of three of the four elements, they can produce a sample of the fourth element using nuclear fusion. So if we have elements in positions (r_1, c_1), (r_1, c_2), (r_2, c_1), where $r_1 \ne r_2$ and $c_1 \ne c_2$, then we can produce element (r_2, c_2).



Samples used in fusion are not wasted and can be used again in future fusions. Newly crafted elements also can be used in future fusions.

Innopolis University scientists already have samples of q elements. They want to obtain samples of all $n \cdot m$ elements. To achieve that, they will purchase some samples from other laboratories and then produce all remaining elements using an arbitrary number of nuclear fusions in some order. Help them to find the minimal number of elements they need to purchase.

Input

The first line contains three integers n, m, q ($1 \le n, m \le 200\,000$; $0 \le q \le min(n \cdot m, 200\,000)$), the chemical table dimensions and the number of elements scientists already have.

The following q lines contain two integers r_i, c_i ($1 \le r_i \le n$, $1 \le c_i \le m$), each describes an element that scientists already have. All elements in the input are different.

Output

Print the minimal number of elements to be purchased.

Examples

input

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

output

```
0
```

input

```
1 5 3
1 3
1 1
1 5
```

output

```
2
```

input

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

output

```
1
```

Note

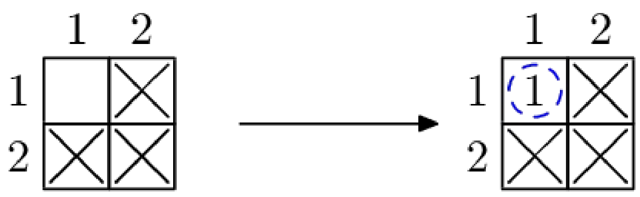
For each example you have a picture which illustrates it.

The first picture for each example describes the initial set of element samples available. Black crosses represent elements available in the lab initially.

The second picture describes how remaining samples can be obtained. Red dashed circles denote elements that should be purchased from other labs (the optimal solution should minimize the number of red circles). Blue dashed circles are elements that can be produced with nuclear fusion. They are numbered in order in which they can be produced.

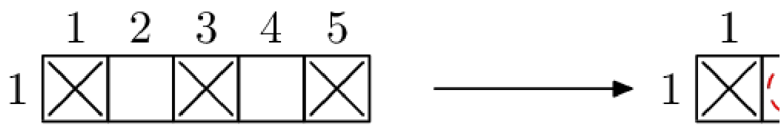
Test 1

We can use nuclear fusion and get the element from three other samples, so we don't need to purchase anything.



Test 2

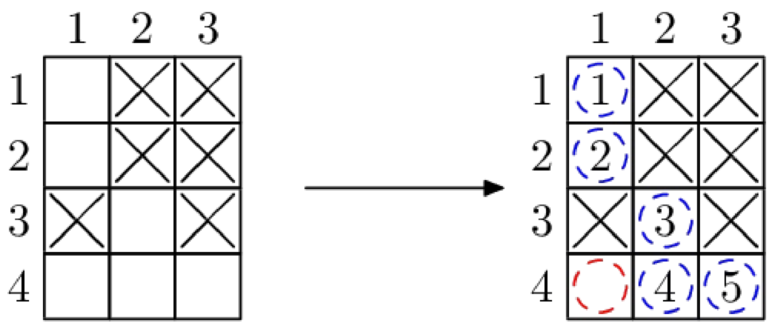
We cannot use any nuclear fusion at all as there is only one row, so we have to purchase all missing elements.



Test 3

There are several possible solutions. One of them is illustrated below.

Note that after purchasing one element marked as red it's still not possible to immidiately produce the middle element in the bottom row (marked as 4). So we produce the element in the left-top corner first (marked as 1), and then use it in future fusions.



C. Hills

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Welcome to Innopolis city. Throughout the whole year, Innopolis citizens suffer from everlasting city construction.

From the window in your room, you see the sequence of n hills, where i-th of them has height a_i. The Innopolis administration wants to build some houses on the hills. However, for the sake of city appearance, a house can be only built on the hill, which is strictly higher than neighbouring hills (if they are present). For example, if the sequence of heights is $5, 4, 6, 2$, then houses could be built on hills with heights 5 and 6 only.

The Innopolis administration has an excavator, that can decrease the height of an arbitrary hill by one in one hour. The excavator can only work on one hill at a time. It is allowed to decrease hills up to zero height, or even to negative values. Increasing height of any hill is impossible. The city administration wants to build k houses, so there must be at least k hills that satisfy the condition above. What is the minimum time required to adjust the hills to achieve the administration's plan?

However, the exact value of k is not yet determined, so could you please calculate answers for all k in range $1 \le k \le \lceil \frac{n}{2} \rceil$? Here $\lceil \frac{n}{2} \rceil$ denotes n divided by two, rounded up.

Input

The first line of input contains the only integer n ($1 \le n \le 5000$)—the number of the hills in the sequence.

Second line contains n integers a_i ($1 \le a_i \le 100\,000$)—the heights of the hills in the sequence.

Output

Print exactly $\lceil \frac{n}{2} \rceil$ numbers separated by spaces. The i-th printed number should be equal to the minimum number of hours required to level hills so it becomes possible to build i houses.

Examples

input

```
5
1 1 1 1 1
```

output

```
1 2 2
```

input

```
3
1 2 3
```

output

```
0 2
```

input

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

output

```
0 1 3
```

Note

In the first example, to get at least one hill suitable for construction, one can decrease the second hill by one in one hour, then the sequence of heights becomes $1, 0, 1, 1, 1$ and the first hill becomes suitable for construction.

In the first example, to get at least two or at least three suitable hills, one can decrease the second and the fourth hills, then the sequence of heights becomes $1, 0, 1, 0, 1$, and hills $1, 3, 5$ become suitable for construction.

D. AB-Strings

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are two strings s and t, consisting only of letters a and b. You can make the following operation several times: choose a prefix of s, a prefix of t and swap them. Prefixes can be empty, also a prefix can coincide with a whole string.

Your task is to find a sequence of operations after which one of the strings consists only of a letters and the other consists only of b letters. The number of operations should be minimized.

Input

The first line contains a string s ($1 \le |s| \le 2\cdot10^5$).

The second line contains a string t ($1 \le |t| \le 2\cdot10^5$).

Here $|s|$ and $|t|$ denote the lengths of s and t, respectively. It is guaranteed that at least one of the strings contains at least one a letter and at least one of the strings contains at least one b letter.

Output

The first line should contain a single integer n ($0 \le n \le 5\cdot10^5$) — the number of operations.

Each of the next n lines should contain two space-separated integers a_i, b_i — the lengths of prefixes of s and t to swap, respectively.

If there are multiple possible solutions, you can print any of them. It's guaranteed that a solution with given constraints exists.

Examples

input

```
bab
bb
```

output

```
2
1 0
1 3
```

input

```
bbbb
aaa
```

output

```
0
```

Note

In the first example, you can solve the problem in two operations:

1. Swap the prefix of the first string with length 1 and the prefix of the second string with length 0. After this swap, you'll have strings ab and bbb.
2. Swap the prefix of the first string with length 1 and the prefix of the second string with length 3. After this swap, you'll have strings bbbb and a.

In the second example, the strings are already appropriate, so no operations are needed.

E. Cycle sort

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

input: standard input
output: standard output

You are given an array of n positive integers $a_1, a_2, \ldots, a_n$. You can perform the following operation any number of times: select several distinct indices $i_1, i_2, \ldots, i_k$ ($1 \le i_j \le n$) and move the number standing at the position i_1 to the position i_2, the number at the position i_2 to the position i_3, ..., the number at the position i_k to the position i_1. In other words, the operation cyclically shifts elements: $i_1 \to i_2 \to \ldots i_k \to i_1$.

For example, if you have $n = 4$, an array $a_1 = 10, a_2 = 20, a_3 = 30, a_4 = 40$, and you choose three indices $i_1 = 2$, $i_2 = 1$, $i_3 = 4$, then the resulting array would become $a_1 = 20, a_2 = 40, a_3 = 30, a_4 = 10$.

Your goal is to make the array sorted in non-decreasing order with the minimum number of operations. The additional constraint is that the sum of cycle lengths over all operations should be less than or equal to a number s. If it's impossible to sort the array while satisfying that constraint, your solution should report that as well.

Input

The first line of the input contains two integers n and s ($1 \le n \le 200\,000$, $0 \le s \le 200\,000$)—the number of elements in the array and the upper bound on the sum of cycle lengths.

The next line contains n integers $a_1, a_2, \ldots, a_n$—elements of the array ($1 \le a_i \le 10^9$).

Output

If it's impossible to sort the array using cycles of total length not exceeding s, print a single number "-1" (quotes for clarity).

Otherwise, print a single number q— the minimum number of operations required to sort the array.

On the next $2 \cdot q$ lines print descriptions of operations in the order they are applied to the array. The description of i-th operation begins with a single line containing one integer k ($1 \le k \le n$)—the length of the cycle (that is, the number of selected indices). The next line should contain k distinct integers $i_1, i_2, \ldots, i_k$ ($1 \le i_j \le n$)—the indices of the cycle.

The sum of lengths of these cycles should be less than or equal to s, and the array should be sorted after applying these q operations.

If there are several possible answers with the optimal q, print any of them.

Examples

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

input
```
4 3
2 1 4 3
```
output
```
-1
```

input
```
2 0
2 2
```
output
```
0
```

Note

In the first example, it's also possible to sort the array with two operations of total length 5: first apply the cycle $1 \to 4 \to 1$ (of length 2), then apply the cycle $2 \to 3 \to 5 \to 2$ (of length 3). However, it would be wrong answer as you're asked to use the minimal possible number of operations, which is 1 in that case.

In the second example, it's possible to the sort the array with two cycles of total length 4 ($1 \to 2 \to 1$ and $3 \to 4 \to 3$). However, it's impossible to achieve the same using shorter cycles, which is required by $s = 3$.

In the third example, the array is already sorted, so no operations are needed. Total length of empty set of cycles is considered to be zero.

F. Passports

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Gleb is a famous competitive programming teacher from Innopolis. He is planning a trip to N programming camps in the nearest future. Each camp will be held in a different country. For each of them, Gleb needs to apply for a visa.

For each of these trips Gleb knows three integers: the number of the first day of the trip s_i, the length of the trip in days len_i, and the number of days t_i this country's consulate will take to process a visa application and stick a visa in a passport. Gleb has P ($1 \le P \le 2$) valid passports and is able to decide which visa he wants to put in which passport.

For each trip, Gleb will have a flight to that country early in the morning of the day s_i and will return back late in the evening of the day $s_i + len_i - 1$.

To apply for a visa on the day d, Gleb needs to be in Innopolis in the middle of this day. So he can't apply for a visa while he is on a trip, including the first and the last days. If a trip starts the next day after the end of the other one, Gleb can't apply for a visa between them as well. The earliest Gleb can apply for a visa is day 1.

After applying for a visa of country i on day d, Gleb will get his passport back in the middle of the day $d + t_i$. Consulates use delivery services, so Gleb can get his passport back even if he is not in Innopolis on this day. Gleb can apply for another visa on the same day he received his passport back, if he is in Innopolis this day.

Gleb will not be able to start his trip on day s_i if he doesn't has a passport with a visa for the corresponding country in the morning of day s_i. In particular, the passport should not be in another country's consulate for visa processing.

Help Gleb to decide which visas he needs to receive in which passport, and when he should apply for each visa.

Input

In the first line of the input there are two integers N ($1 \le N \le 22$) and P ($1 \le P \le 2$)—the number of trips and the number of passports Gleb has, respectively.

The next N lines describe Gleb's trips. Each line contains three positive integers s_i, len_i, t_i ($1 \le s_i, len_i, t_i \le 10^9$)—the first day of the trip, the length of the trip and number of days the consulate of this country needs to process a visa application. It is guaranteed that no two trips intersect.

Output

If it is impossible to get all visas on time, just print "NO" (quotes for clarity). Otherwise, print "YES" and N lines describing trips. For each trip, first print number of the passport Gleb should put this country's visa in, and then print number of the day he should apply for it. Print trips in the same order as they appear in the input. Days are numbered from 1, starting with tomorrow—the first day you can apply for a visa. Passports are numbered from 1 to P.

If there are several possible answers, print any of them.

Examples

input

```
2 1
3 1 1
6 1 1
```

output

```
YES
1 1
1 4
```

input

```
3 1
13 2 2
7 3 1
19 3 4
```

output

```
YES
1 10
1 1
1 2
```

input

```
7 2
15 1 1
14 1 1
18 1 1
21 1 1
9 4 6
22 2 5
5 4 3
```

output

```
YES
2 13
1 1
1 16
1 19
1 2
2 16
```

```
2 1
```

input

```
3 1
7 3 1
13 2 3
19 3 4
```

output

```
NO
```

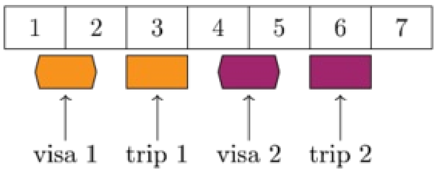
Note

Examples with answer "YES" are depicted below.

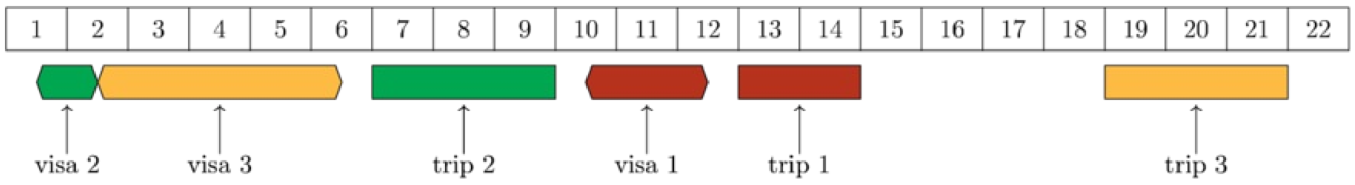
Each cell of the stripe represents a single day. Rectangles represent trips, each trip starts in the morning and ends in the evening. Rectangles with angled corners represent visa applications. Each application starts in the middle of a day and ends t_i days after. The trip and the visa application for the same country have the same color.

In examples with two passports, visa applications and trips depicted above the time stripe are made using the first passport, visa applications and trips depicted below the time stripe are made using the second passport.

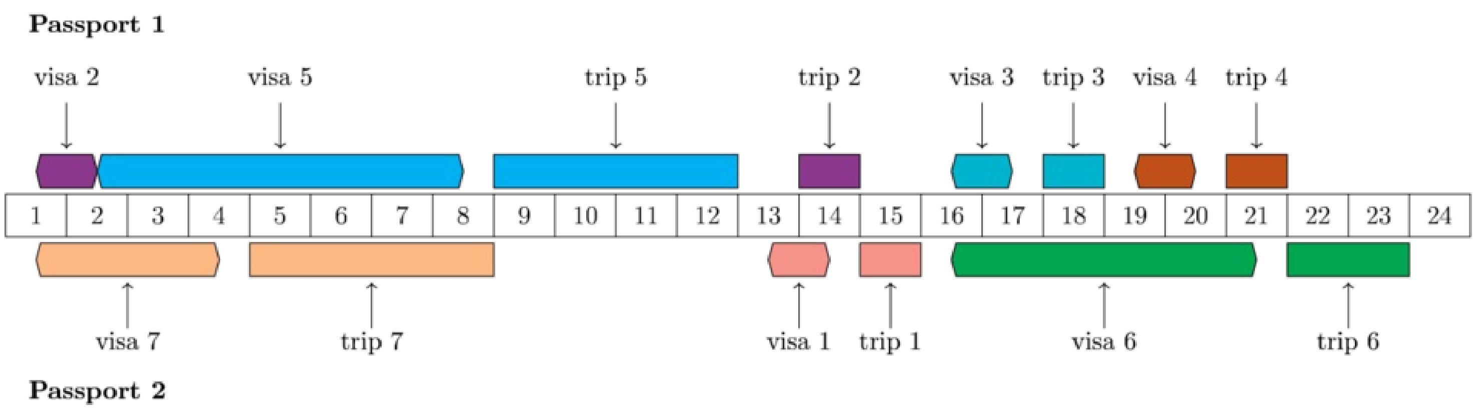
Example 1:



Example 2:



Example 3:


Passport 1
visa 2
visa 5
trip 5
trip 2
visa 3
trip 3
visa 4
trip 4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
visa 7
trip 7
visa 1
trip 1
visa 6
trip 6
Passport 2