A. Ilya and a Colorful Walk

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

Ilya lives in a beautiful city of Chordalsk.

There are n houses on the street Ilya lives, they are numerated from 1 to n from left to right; the distance between every two neighboring houses is equal to 1 unit. The neighboring houses are 1 and 2, 2 and 3, ..., $n-1$ and n. The houses n and 1 are not neighboring.

The houses are colored in colors $c_1, c_2, \ldots, c_n$ so that the i-th house is colored in the color c_i. Everyone knows that Chordalsk is not boring, so there are at least two houses colored in different colors.

Ilya wants to select two houses i and j so that $1 \le i < j \le n$, and they have different colors: $c_i \neq c_j$. He will then walk from the house i to the house j the distance of $(j-i)$ units.

Ilya loves long walks, so he wants to choose the houses so that the distance between them is the maximum possible.

Help Ilya, find this maximum possible distance.

Input
The first line contains a single integer n ($3 \le n \le 300\,000$) — the number of cities on the street.

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le n$) — the colors of the houses.

It is guaranteed that there is at least one pair of indices i and j so that $1 \le i < j \le n$ and $c_i \neq c_j$.

Output
Print a single integer — the maximum possible distance Ilya can walk.

Examples

input
5 1 2 3 2 3
output
4

input
3 1 2 1
output
1

input
7 1 1 3 1 1 1 1
output
4

Note
In the first example the optimal way is to walk from the first house to the last one, where Ilya can walk the distance of $5-1=4$ units.

In the second example the optimal way is to either walk from the first house to the second or from the second to the third. Both these ways have the distance of 1 unit.

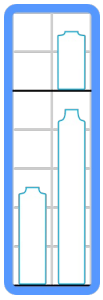
In the third example the optimal way is to walk from the third house to the last one, where Ilya can walk the distance of $7-3=4$ units.

B. Alyona and a Narrow Fridge

time limit per test: 1 second
memory limit per test: 256 megabytes

input: standard input
output: standard output

Alyona has recently bought a miniature fridge that can be represented as a matrix with h rows and 2 columns. Initially there is only one shelf at the bottom of the fridge, but Alyona can install arbitrary number of shelves inside the fridge between any two rows. A shelf is two cells wide, does not occupy any space but separates the inside of the fridge to the lower and upper part.



An example of a fridge with $h = 7$ and two shelves. The shelves are shown in black. The picture corresponds to the first example.

Alyona has n bottles of milk that she wants to put in the fridge. The i-th bottle is a_i cells tall and 1 cell wide. She can put a bottle on some shelf if the corresponding space above the shelf is at least as tall as the bottle. She can **not** put a bottle on top of another bottle (if there is no shelf between them). Two bottles can not share a cell.

Alyona is interested in the largest integer k such that she can put bottles 1, 2, ..., k in the fridge at the same time. Find this largest k.

Input

The first line contains two integers n and h ($1 \le n \le 10^3$, $1 \le h \le 10^9$) — the number of bottles and the height of the fridge.

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le h$) — the heights of the bottles.

Output

Print the single integer k — the maximum integer such that Alyona can put the bottles 1, 2, ..., k in the fridge at the same time. If Alyona can put all bottles in the fridge, print n. It is easy to see that Alyona can always put at least one bottle in the fridge.

Examples

input
5 7 2 3 5 4 1
output
3

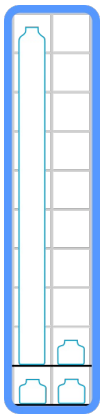
input
10 10 9 1 1 1 1 1 1 1 1 1
output
4

input
5 10 3 1 4 2 4
output
5

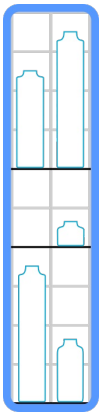
Note

One of optimal locations in the first example is shown on the picture in the statement.

One of optimal locations in the second example is shown on the picture below.



One of optimal locations in the third example is shown on the picture below.

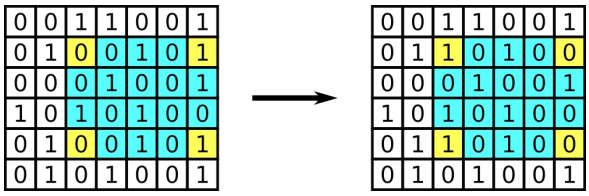


C. Ramesses and Corner Inversion

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

Ramesses came to university to algorithms practice, and his professor, who is a fairly known programmer, gave him the following task.

You are given two matrices A and B of size $n \times m$, each of which consists of 0 and 1 only. You can apply the following operation to the matrix A arbitrary number of times: take any ***submatrix*** of the matrix A that has at least two rows and two columns, and invert the values in its corners (i.e. all corners of the submatrix that contain 0, will be replaced by 1, and all corners of the submatrix that contain 1, will be replaced by 0). You have to answer whether you can obtain the matrix B from the matrix A.



An example of the operation. The chosen submatrix is shown in blue and yellow, its corners are shown in yellow.

Ramesses don't want to perform these operations by himself, so he asks you to answer this question.

A *submatrix* of matrix M is a matrix which consist of all elements which come from one of the rows with indices $x_1, x_1 + 1, \ldots, x_2$ of matrix M and one of the columns with indices $y_1, y_1 + 1, \ldots, y_2$ of matrix M, where x_1, x_2, y_1, y_2 are the edge rows and columns of the submatrix. In other words, a *submatrix* is a set of elements of source matrix which form a solid rectangle (i.e. without holes) with sides parallel to the sides of the original matrix. The corners of the submatrix are cells (x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2), where the cell (i, j) denotes the cell on the intersection of the i-th row and the j-th column.

Input

The first line contains two integers n and m ($1 \le n, m \le 500$) — the number of rows and the number of columns in matrices A and B.

Each of the next n lines contain m integers: the j-th integer in the i-th line is the j-th element of the i-th row of the matrix A ($0 \le A_{ij} \le 1$).

Each of the next n lines contain m integers: the j-th integer in the i-th line is the j-th element of the i-th row of the matrix B ($0 \le B_{ij} \le 1$).

Output

Print "Yes" (without quotes) if it is possible to transform the matrix A to the matrix B using the operations described above, and "No" (without quotes), if it is not possible. You can print each letter in any case (upper or lower).

Examples

input
3 3 0 1 0 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0
output
Yes

input
6 7 0 0 1 1 0 0 1 0 1 0 0 1 0 1

```
0 0 0 1 0 0 1
1 0 1 0 1 0 0
0 1 0 0 1 0 1
0 1 0 1 0 0 1
1 1 0 1 0 1 1
0 1 1 0 1 0 0
1 1 0 1 0 0 1
1 0 1 0 0 1 0
0 1 1 0 1 0 0
0 1 1 1 1 0 1
```

output

Yes

input

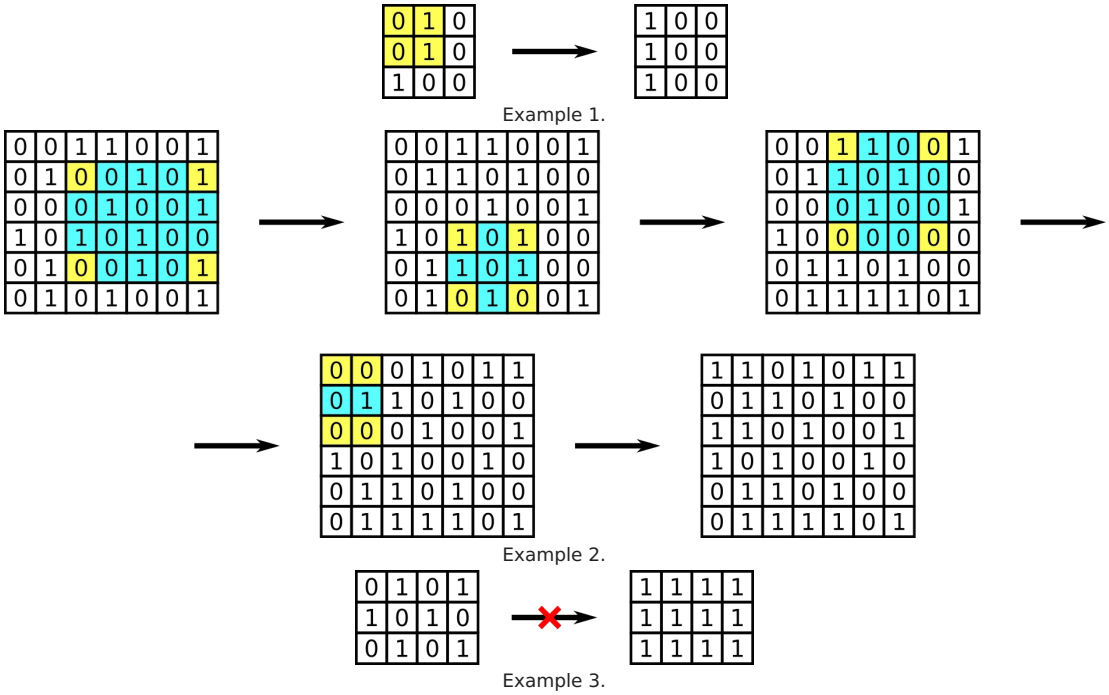
```
3 4
0 1 0 1
1 0 1 0
0 1 0 1
1 1 1 1
1 1 1 1
1 1 1 1
```

output

No

Note

The examples are explained below.



Example 1.

Example 2.

Example 3.

D. Frets On Fire

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Miyako came to the flea kingdom with a ukulele. She became good friends with local flea residents and played beautiful music for them every day.

In return, the fleas made a bigger ukulele for her: it has n *strings*, and each string has $(10^{18}+1)$ *frets* numerated from 0 to 10^{18}. The fleas use the array $s_1, s_2, \ldots, s_n$ to describe the ukulele's tuning, that is, the *pitch* of the j-th fret on the i-th string is the integer $s_i + j$.

Miyako is about to leave the kingdom, but the fleas hope that Miyako will answer some last questions for them.

Each question is in the form of: "How many different pitches are there, if we consider frets between l and r (inclusive) on all strings?"

Miyako is about to visit the cricket kingdom and has no time to answer all the questions. Please help her with this task!

Formally, you are given a matrix with n rows and $(10^{18}+1)$ columns, where the cell in the i-th row and j-th column ($0 \le j \le 10^{18}$) contains the integer $s_i + j$. You are to answer q queries, in the k-th query you have to answer the number of distinct integers in the matrix from the l_k-th to the r_k-th columns, inclusive.

Input
The first line contains an integer n ($1 \le n \le 100\,000$) — the number of strings.

The second line contains n integers $s_1, s_2, \ldots, s_n$ ($0 \le s_i \le 10^{18}$) — the tuning of the ukulele.

The third line contains an integer q ($1 \le q \le 100\,000$) — the number of questions.

The k-th among the following q lines contains two integers $l_k r_k$ ($0 \le l_k \le r_k \le 10^{18}$) — a question from the fleas.

Output
Output one number for each question, separated by spaces — the number of different pitches.

Examples

input
6 3 1 4 1 5 9 3 7 7 0 2 8 17
output
5 10 18

input
2 1 500000000000000000 2 1000000000000000000 1000000000000000000 0 1000000000000000000
output
2 1500000000000000000

Note
For the first example, the pitches on the 6 strings are as follows.

Fret	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	...
s_1 :	3	4	5	6	7	8	9	10	...
s_2 :	1	2	3	4	5	6	7	8	...
s_3 :	4	5	6	7	8	9	10	11	...
s_4 :	1	2	3	4	5	6	7	8	...
s_5 :	5	6	7	8	9	10	11	12	...
s_6 :	9	10	11	12	13	14	15	16	...

There are 5 different pitches on fret 7 — $8, 10, 11, 12, 16$.

There are 10 different pitches on frets $0, 1, 2$ — $1, 2, 3, 4, 5, 6, 7, 9, 10, 11$.

E. Pavel and Triangles

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

Pavel has several sticks with lengths equal to powers of two.

He has a_0 sticks of length $2^0 = 1$, a_1 sticks of length $2^1 = 2$, ..., a_{n-1} sticks of length 2^{n-1}.

Pavel wants to make the maximum possible number of triangles using these sticks. The triangles should have strictly positive area, each stick can be used in at most one triangle.

It is forbidden to break sticks, and each triangle should consist of exactly three sticks.

Find the maximum possible number of triangles.

Input
The first line contains a single integer n ($1 \le n \le 300\,000$) — the number of different lengths of sticks.

The second line contains n integers a_0, a_1, ..., a_{n-1} ($1 \le a_i \le 10^9$), where a_i is the number of sticks with the length equal to 2^i.

Output
Print a single integer — the maximum possible number of non-degenerate triangles that Pavel can make.

Examples

input
5 1 2 2 2
output
3

input
3 1 1 1
output
0

input
3 3 3 3
output
3

Note

In the first example, Pavel can, for example, make this set of triangles (the lengths of the sides of the triangles are listed): $(2^0, 2^4, 2^4)$, $(2^1, 2^3, 2^3)$, $(2^1, 2^2, 2^2)$.

In the second example, Pavel cannot make a single triangle.

In the third example, Pavel can, for example, create this set of triangles (the lengths of the sides of the triangles are listed): $(2^0, 2^0, 2^0)$, $(2^1, 2^1, 2^1)$, $(2^2, 2^2, 2^2)$.

F. Niyaz and Small Degrees

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Niyaz has a tree with n vertices numerated from 1 to n. A tree is a connected graph without cycles.

Each edge in this tree has strictly positive integer weight. A degree of a vertex is the number of edges adjacent to this vertex.

Niyaz does not like when vertices in the tree have too large degrees. For each x from 0 to $(n-1)$, he wants to find the smallest total weight of a set of edges to be deleted so that degrees of all vertices become at most x.

Input

The first line contains a single integer n ($2 \le n \le 250\,000$) — the number of vertices in Niyaz's tree.

Each of the next $(n-1)$ lines contains three integers a, b, c ($1 \le a, b \le n$, $1 \le c \le 10^6$) — the indices of the vertices connected by this edge and its weight, respectively. It is guaranteed that the given edges form a tree.

Output

Print n integers: for each $x = 0, 1, \ldots, (n-1)$ print the smallest total weight of such a set of edges that after one deletes the edges from the set, the degrees of all vertices become less than or equal to x.

Examples

input
5 1 2 1 1 3 2 1 4 3 1 5 4
output
10 6 3 1 0

input
5 1 2 1 2 3 2 3 4 5 4 5 14
output
22 6 0 0 0

Note

In the first example, the vertex 1 is connected with all other vertices. So for each x you should delete the $(4 - x)$ lightest edges outgoing from vertex 1, so the answers are $1 + 2 + 3 + 4$, $1 + 2 + 3$, $1 + 2$, 1 and 0.

In the second example, for $x = 0$ you need to delete all the edges, for $x = 1$ you can delete two edges with weights 1 and 5, and for $x \geq 2$ it is not necessary to delete edges, so the answers are $1 + 2 + 5 + 14$, $1 + 5$, 0, 0 and 0.

G. Get Ready for the Battle

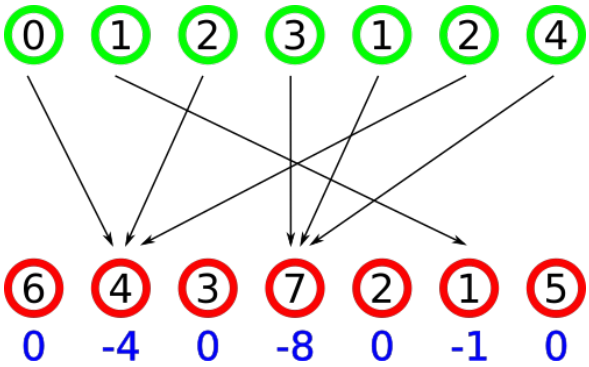
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Recently Evlampy installed one interesting computer game, one of aspects of which is to split army into several groups and then fight with enemy's groups. Let us consider a simplified version of the battle.

In the nearest battle Evlampy should fight an enemy army that consists of m groups, the i-th of which has hp_i health points.

Evlampy's army consists of n equal soldiers. Before each battle he should split his army in exactly m groups (possibly, empty) so that the total size of the groups is n. The battle is played step-by-step. On each step each of Evlampy's groups attacks exactly one enemy group. Thus, each step is described with an array of m integers $a_1, a_2, \ldots, a_m$, meaning that the i-th Evlampy's group attacks the a_i-th enemy group. Different groups can attack same group, on each step the array a is chosen independently.

After each step the health points of each enemy group decreases by the total number of soldiers in Evlampy's groups that attacked this enemy group at this step. Enemy group is destroyed once its health points are zero or negative. Evlampy's soldiers do not lose health.



An example of a step. The numbers in green circles represent the number of soldiers in Evlampy's groups, the arrows represent attacks, the numbers in red circles represent health points of enemy groups, the blue numbers represent how much the health points will decrease after the step.

Evlampy understands that the upcoming battle will take the whole night. He became sad because this way he won't have enough time to finish his homework. Now Evlampy wants you to write a program that will help him win in the smallest possible number of steps. Can you help him?

In other words, find the smallest number of steps needed to destroy all enemy groups and show a possible way of doing this. Find the requires splitting of the army into m groups and the arrays a for each step.

Input

The first line contains two integers n and m ($1 \leq m \leq n \leq 10^6$) — the number of soldiers in Evlampy's army and the number of groups in enemy army. m is also equal to the maximum possible number of groups Evlampy can split the army to.

The second line contains m integers $hp_1, hp_2, \ldots, hp_m$ ($1 \leq hp_i \leq 10^6$) — the health points of each of the enemy groups.

It is guaranteed that the sum of hp_i does not exceed 10^6.

Output

Print a single integer t — the minimum possible number of steps needed to win the battle.

After that print m integers $s_1, s_2, \ldots, s_m$ ($s_i \geq 0$, $s_1 + s_2 + \ldots + s_m = n$), meaning that the i-th group of Evlampy's army should contain s_i soldiers.

In each of the next t lines print m integers $a_1, a_2, \ldots, a_m$ ($1 \leq a_i \leq m$) — the description of one step. The integers mean that on the corresponding step the i-th Evlampy's group should attack the a_i-th enemy group. It is allowed to attack an already destroyed group.

Examples

input
13 7 6 4 3 7 2 1 5
output
3 0 1 2 3 1 2 4 2 6 2 4 4 2 4 3 1 7 1 7 7 1 3 1 5 3 7 5 1

input
6 5 3 3 3 3 3

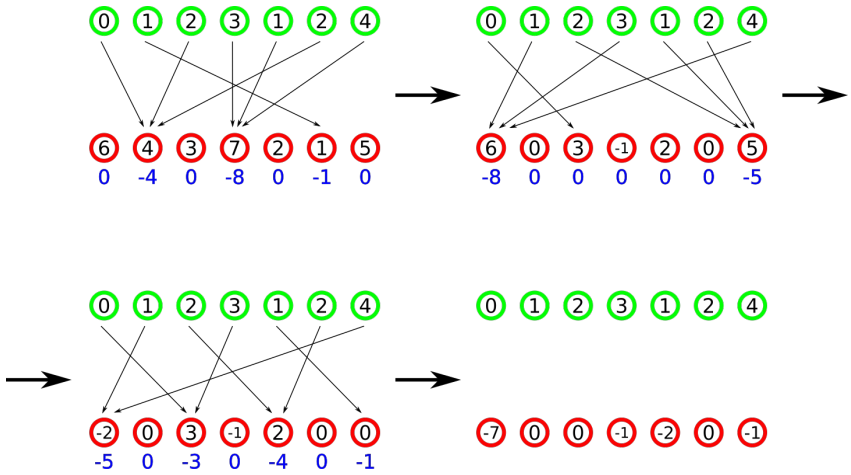
output
3 3 3 0 0 0 1 2 3 4 5 3 4 5 5 5 5 5 5 5 5

input
7 4 1 5 9 2

output
3 1 2 4 0 1 4 2 3 2 3 3 3 3 3 3 3

Note

The first example is shown below.



H. Triple

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have received your birthday gifts — n triples of integers! The i-th of them is $\{a_i, b_i, c_i\}$. All numbers are greater than or equal to 0, and strictly smaller than 2^k, where k is a fixed integer.

One day, you felt tired playing with triples. So you came up with three new integers x, y, z, and then formed n arrays. The i-th array consists of a_i repeated x times, b_i repeated y times and c_i repeated z times. Thus, each array has length $(x + y + z)$.

You want to choose exactly one integer from each array such that the XOR (bitwise exclusive or) of them is equal to t. Output the number of ways to choose the numbers for each t between 0 and $2^k - 1$, inclusive, modulo 998244353.

Input

The first line contains two integers n and k ($1 \le n \le 10^5$, $1 \le k \le 17$) — the number of arrays and the binary length of all numbers.

The second line contains three integers x, y, z ($0 \le x, y, z \le 10^9$) — the integers you chose.

Then n lines follow. The i-th of them contains three integers a_i, b_i and c_i ($0 \le a_i, b_i, c_i \le 2^k - 1$) — the integers forming the i-th array.

Output

Print a single line containing 2^k integers. The i-th of them should be the number of ways to choose exactly one integer from each array so that their XOR is equal to $t = i - 1$ modulo 998244353.

Examples

input
1 1 1 2 3 1 0 1

output
2 4

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

input
4 3 1 2 3 1 3 7 0 2 5 1 0 6 3 3 2
output
198 198 126 126 126 126 198 198

Note

In the first example, the array we formed is $(1, 0, 0, 1, 1, 1)$, we have two choices to get 0 as the XOR and four choices to get 1.

In the second example, two arrays are $(0, 1, 1, 2)$ and $(1, 2, 2, 3)$. There are sixteen $(4 \cdot 4)$ choices in total, 4 of them ($1 \oplus 1$ and $2 \oplus 2$, two options for each) give 0, 2 of them ($0 \oplus 1$ and $2 \oplus 3$) give 1, 4 of them ($0 \oplus 2$ and $1 \oplus 3$, two options for each) give 2, and finally 6 of them ($0 \oplus 3$, $2 \oplus 1$ and four options for $1 \oplus 2$) give 3.