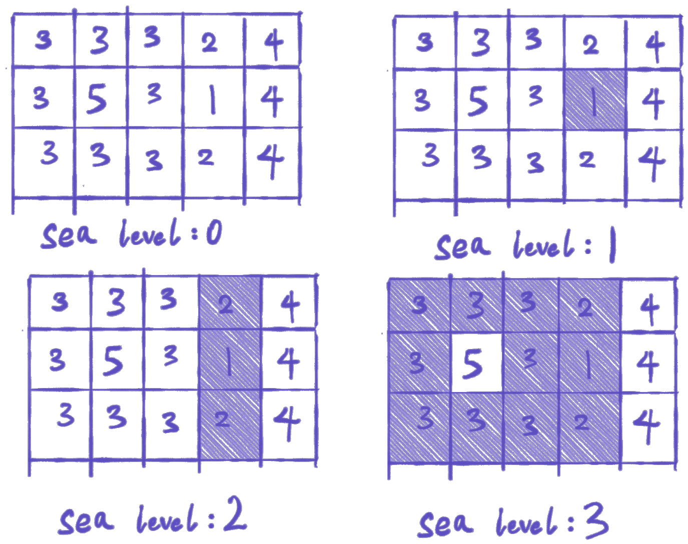
Mr. Panda

Mr. Panda is a polar bear who lives at the Bamboo Island. He is worried about flooding. Some low lying parts of Bamboo Island will be under water if the sea levels continue to rise. Mr. Panda hates swimming so it is a bad news for him.



Mr. Panda has a topographic map of the Bamboo Island. Bamboo Island has a rectangular shape, and can be divided into a n by m grid. The elevation of the each field at grid point (i, j) is A_{ij}. The soil at Bamboo Island is porous, and the water can freely flow through it. Thus, if the sea level is no less than A_{ij}, then the field at grid point (i, j) will be flooded.

Adjacent un-flooded fields (i.e., sharing common edge) create safe areas. Mr. Panda is interested in the number of distinct safe areas for a given sea level.

An example of 3x5 map is given on the right. Numbers denote the elevation of the A_{ij} as described above. Flooded fields are shaded. There will be

- 1 safe area when the sea level is 0,
- 1 safe area when the sea level is 1,
- 2 safe areas when the sea level is 2,
- 2 safe areas when the sea level is 3,
- 1 safe area when the sea level is 4 (not shown in the image),
- and no safe areas when the sea level is 5 (not shown in the image).

Input

The first line contains two numbers n and m separated by a single space, the dimensions of the Bamboo Island, where $1 \leq n, m \leq 1000$.

Next n lines contain m integers from the range $[1, 10^9]$ separated by single spaces, denoting the elevations of the respective fields.

Next line contains an integer T, $1 \leq T \leq 10^5$, the number of sea levels that Mr. Panda is interested in.

The last line contains T integers t_i, separated by single spaces, such that $0 \leq t_1 \leq t_2 \leq ... \leq t_{T-1} \leq t_T \leq 10^9$. The i-th integer denotes the sea level of the i-th query.

Output

Output a single line consisting of T numbers c_i separated by single spaces, where c_i is the number of safe areas when the sea level is equal to t_i. The output ends with a newline.

Example

```
Input:
3 5
3 3 3 2 4
3 5 3 1 4
3 3 3 2 4
5
1 2 3 4 5

Output:
1 2 2 1 0
```

Kou Sort

Kou just finished her first algorithm class and designed a new sorting algorithm: given an array of N distinct numbers, the algorithm sorts them in ascending order by performing the minimum number of swaps required. A *swap* is an exchange of two adjacent elements in the array.

For example, if the array is [9, 1, 0, 5, 4], the smallest number of required swaps to produce the array [0, 1, 4, 5, 9] is 6.

```
0: [9, 1, 0, 5, 4] (original array)
1: [1, 9, 0, 5, 4] (swap 1)
2: [1, 0, 9, 5, 4]
3: [1, 0, 5, 9, 5]
4: [1, 0, 5, 4, 9]
5: [0, 1, 5, 4, 5]
6: [0, 1, 4, 5, 9] (swap 6)
```

(Note that there may be other sequences of swaps that lead to the minimum number of swaps.)

Now Kou wants to determine if her algorithm always performs as she claims it does. Given an array of N distinct integers, your task is to figure out the minimum number of swaps required to sort it. Kou will then make sure that her algorithm performs exactly that number of swaps.

Input

The first line of the input contains a single integer N (1 <= N <= 500,000), the size of that array. The following N lines represent the content of the input array. Each of these N lines contains a single integer between 0 and 999,999,999. It is guaranteed that the array contains no duplicates.

Output

You should print one line containing a single integer: the minimum number of swaps required to sort the given array followed by a new line character.

Example 1

```
Input:
5
9
1
0
5
4

Output:
6
```

Example 2

```
Input:
3
1
2
3

Output:
0
```

APS Homework

Daru is worried about a very time-consuming APS homework. As a super hacker, he develops an AI called Amadeus to help him solve his homework problems automatically.

There are N problems for this week. It takes a_i seconds for Amadeus to solve the i-th problem. Since Daru is a super hacker, he has broken into Gradescope to figure out how long it takes Gradescope to evaluate each submission. He knows that it takes b_i seconds for Gradescope autograder to evaluate and accept the correct solution for the i-th problem. (Daru is not interested in cheating by also downloading all the test cases from Gradescope since it's not cool and he knows that this would violate the academic integrity policies).

Amadeus can not work on multiple problems simultaneously. It will follow an order given by Daru to solve problems. Once Amadeus solves a problem, it submits the solution to Gradescope and continues to the next problem.

It takes no time to submit a solution and Gradescope is able to evaluate multiple solutions at the same time. The solution for the i-th problem will be accepted in exactly b_i seconds after it is submitted to Gradescope. Amadeus always produces a correct solution, so Gradescope always accepts after the first submission.

Daru wants to find an order of problems to minimize the time necessary for all problems to be accepted on Gradescope. Can you help him to figure out the optimal order?

Input

The first line contains one integer N. N is the number of problems in homework, 1 <= N <= 1000.

The i-th line of the following N lines contains a pair of integers a_i and b_i, $1 <= a_i, b_i <= 10000$, as described above.

Output

Output one number followed by a newline. The total number of seconds that it takes Amadeus to solve, submit and get accept verdict for all the problems when the order of the problems is optimal. (Note, that there may be multiple orders that result in the same time. Since you only report on the time, the actual order is irrelevant.) The output ends with a newline.

Example 1

```
Input:
3
2 1
2 5
3 2

Output:
8
```

The optimal order is to do the 2nd problem first, then solve the 3rd problem and leave the 1st problem to solve at the end.

The solution for the 1st problem will be accepted at time $5 + 2 + 1 = 8$ (Amadeus starts working on this problem at time 5). The solution for the 2nd problem will be accepted at time $0 + 2 + 5 = 7$. The evaluation for the 3rd problem will be accepted at time $2 + 3 + 2 = 7$,

Hence the total number of seconds consumed from start to end is 8.

Grade Sort

You are given the midterm exam scores for all students in all of NYU campuses, 0 <= score <= 100.

To report the midterm grades, you need to display all the scores in ascending order.

Input

The first line contains one integer N. N is the total number of grades, 1 <= N <= 10,000,000.

In the next line, there are N integers indicating the grades.

Output

Print a line with N space separated integers. These integers are the grades of the midterm sorted in ascending order. The output ends with a newline.

Warning: This problem may have potentially very large inputs/outputs (see the value for N above). Use fast IO operations.

Example 1

```
Input:
4
50 60 100 90

Output:
50 60 90 100
```

Example 2

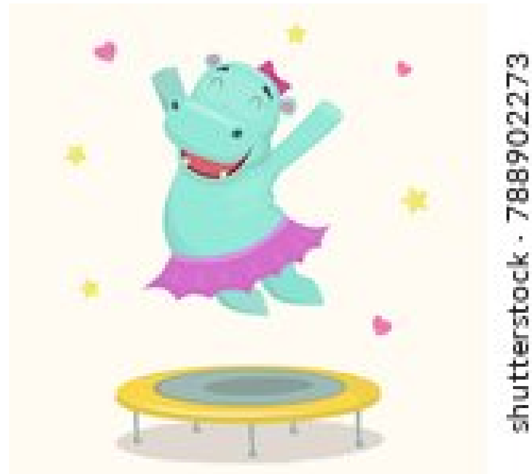
```
Input:
6
99 92 93 92 93 91

Output:
91 92 92 93 93 99
```

Dancing Hippos

NYU is preparing a new attraction for the graduation ceremonies: A Dancing Hippos Act. The new president of atractions was able to locate a supplier who can provide a large number of dancing hippos. The idea for the act is to have them dancing on a platform that will be raised above the graduates.

The hope is to have as many hippos dancing up on the elevated platform as possible. But the platform has limited weight capcity and we would not want to have new graduates squashed by the falling hippos.



Given the weight capacity `W` of the platforms, the number of dancing hippos available `N`, and the weights of all the hippos, `w_1, w_2, ..., w_N`, we need to figure out the largest number of dancing hippos that can be placed on the platform without endangering the graduates.

Input

First two lines contains `0<=W<=10^8` and `0<=N<=10^5` as described above. The net line contains `N` weights of the dancing hippos, `0<=w_i<=10^8`.

Output

Maximum number of hippos that can be safely placed on the platform.

Example 1

```
Input
50
4
76 56 64 69


Output
0
```

Example 2

```
Input
150
7
45 32 78 56 22 63 37

Output
4
```

Example 3

```
Input
10000
10
87 21 15 43 60 61 71 57 79 20

Output
10
```