“Previous OA questions asked by Zomato”

Friends:

Given N people. For a pair of people, they have a friend value $A[i][j]$. We want to split these people into N groups, such that each people belong to exactly one group and each group contains at least one person. The partition value of a split is defined as the minimum friend value over all the pairs of different people within the same group. Find the maximum partition value possible.

Suppose the minimum friend value of all pairs of different people from $Group1$ is $min1$ and from $Group2$ is $min2$. Then the partition value is defined as $min(min1, min2)$.

Input

The first line contains an integer N representing the number of people. The next N line contains N integers where $A[i][j]$ represents the friend value between $A[i][j].(3 \leq N \leq 500),(1 \leq A[i][j] \leq 10^9),(A[i][i] = 0),(A[i][j] = A[j][i])$.

Examples

input
4 0 1 2 3 1 0 4 5 2 4 0 9 3 5 9 0
output
3

input
4 0 702209411 496813081 673102149 702209411 0 561219907 730593611 496813081 561219907 0 814024114 673102149 730593611 814024114 0
output
702209411

Solution:

Solution

- If N = 3, then one group will contain only one person hence the partition value will be 0.
- If N >= 4 then each group can have at least 2 people.
- Now to find the optimal value, we will binary search over the answer.
- Let's say the possible partition value is x. To find out whether it is possible or not we will do the following
    - Iterate over the NxN matrix, if for pairs i, j (i not equal to j) the value A[i][j] is less than x then they must be in the opposite group hence make an edge between them.
    - If the value is greater than or equal to x then they can be in the same or opposite group, hence there is no need to make an edge.
    - Now after we have iterated over the matrix, we will check if the graph formed above is bipartite or not. If not then x is not a possible value and we have to reduce the high value in the binary search.
    - If the graph is bipartite and each group contains at least two people, the x is a possible value and we can increase the low value in binary search.
    - In this way at last we will arrive at the most optimal possible value of x.

XOR Operation

Given an array A consisting of N integers. Also given Q queries along with it.In each query, you are given 3 spaced separated integers L, R and X. You need to output the summation of XOR of X with each of the array elements from range L to R both inclusive(1- based indexing).

The array does not change after any query.

Input

The first line contains two integers N and Q. The next line contains N spaced separated integers A_i. The next Q lines contains 3 integers each L,R and X.$(1 \leq N, Q \leq 10^5)(1 \leq A_i, X \leq 10^9)(1 \leq L, R \leq N)$.

Example

input
5 2 2 3 1 4 5 1 1 3 3 5 2
output
1 16

Solution

- For each of the bits i.e from **0_th to 30_th-bit**, **maintain a count(prefix sum)** for each of them, how many times they are set in the range 0 to L.
- Now to find how many times **i_th bit** is set in range L to R we can simply do **pre_sum[i][R]-pre_sum[i][L-1]**.
- Now we will iterate from 0_th bit to 30_th bit and do the following.
    - For each of the bits we will count how many times they are set in the range L to R. Let it be **set_current_bit_number**.
    - Now if the current bit is set in X , then we will add **(1ll<<current_bit)*((R-L+1)-set_current_bit_number)** to the answer.
    - If the current bit is not set in X, then we will add **(1ll<<current_bit)*(set_current_bit_number)** to the answer.
    - At last print the answer.
- This can be done for all the queries.

Numbers Formation:

Given three integers X, Y and Z, your task is to find the sum of all the numbers formed having 4 at most X times, 5 at most Y times, and 6 at most Z times

Output the sum modulo $(10^9 + 7)$

Input

The first line of input contains three integer x,y,z. $(1 \leq x, y, z \leq 60)$

Output

Output the sum of all the numbers formed

Examples

input
1 1 1
output
3675

input
0 0 0
output
0

Note

For test case 1:

Explanation: 4 + 5 + 6 + 45 + 54 + 56 + 65 + 46 + 64 + 456 + 465 + 546 + 564 + 645 + 654 = 3675

Pseudo Code:

```
 num[0][0][0] = 1;
for (int i = 0; i &lt;= x; ++i)
{
    for (int j = 0; j &lt;= y; ++j)
    {
        for (int k = 0; k &lt;= z; ++k)
        {

            if (i &gt; 0)
            {
                sum[i][j][k] += (sum[i - 1][j][k] * 10 + 4 * num[i - 1][j][k]) % mod;
                num[i][j][k] += num[i - 1][j][k] % mod;
            }
            if (j &gt; 0)
            {
                sum[i][j][k] += (sum[i][j - 1][k] * 10 + 5 * num[i][j - 1][k]) % mod;
                num[i][j][k] += num[i][j - 1][k] % mod;
            }
            if (k &gt; 0)
            {
                sum[i][j][k] += (sum[i][j][k - 1] * 10 + 6 * num[i][j][k - 1]) % mod;
                num[i][j][k] += num[i][j][k - 1] % mod;
            }

            ans += sum[i][j][k] % mod;
            ans %= mod;
        }
    }
}
cout&lt;
```

Special Keyboard:

Imagine you have a special keyboard with the following keys:

Key 1: Prints A on screen

Key 2: $(Ctrl - A)$: Select screen

Key 3: $(Ctrl - C)$: Copy selection to buffer

Key 4: $(Ctrl - V)$: Print buffer on screen appending it after what has already been printed.

Find the maximum numbers of $A's$ that can be produced by pressing keys on the special keyboard N times.

Input

The first line contains a single integer $N.(2 \leq N \leq 75)$.

Output

Your task is to return the maximum number of $A's$ that can be on the screen after performing N operations.

Examples

input
7
output
9

input
2
output
2

Solution

- Given different operations, we have to maximize the number of A's that can be produced.
- We basically have 2 ways, either we follow step 1(press key 1) and add a single A or we perform step 2(press key 2) and step 3(press key 3) and then do step 4(press key 4) again and again.
- So this basically boils down to a simple dp problem whose code is explained below.

Given an encoded string, return its decoded string.

The encoding rule is: k[encoded_string], where the encoded_string inside the square brackets is being repeated exactly k times. Note that k is guaranteed to be a positive integer.

You may assume that the input string is always valid; No extra white spaces, square brackets are well-formed, etc.

Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers, k. For example, there won't be input like 3a or 2[4].

```
s = "3[a]2[bc]", return "aaabcbc".
s = "3[a2[c]]", return "accaccacc".
s = "2[abc]3[cd]ef", return "abcabccdcdcdef".
```

Q -> Given an array of n integers, find the sum of all it's possible subsequences (2^n) and return the **bitwise OR of sum** of all the subsequences.

Eg -> arr[2] = { 1, 1 } -----> possible subsequences = 2^n = 4 i.e. {0}, {1}, {1}, {1,1} and sum of each subsequence = 0,1,12 .
Bitwise OR, output = 0 | 1 | 1 | 2 = 3

Beautiful Element

Question 12 **Max. score: 100.00**

Beautiful element

An element in the array *A* at an index *ind* is called beautiful if $\sum_{i=0}^{ind-1} A[i] \leq A[ind]$. In other words, an element is beautiful if it is greater than or equal to the sum of elements before it. The first element of the array is always considered to be beautiful.

You are given an array *A* of *N* integers.

Task

Determine the maximum number of beautiful elements in the array after rearranging the array.

Example

Assumptions

- $N = 4$
- $A = [4, 2, 1, 3]$

Approach

You can rearrange the array as $[1, 2, 4, 3]$.

- 1 becomes beautiful as there is no element before it.
- 2 becomes beautiful as the sum of elements before it is 1.

- $A = [4, 2, 1, 3]$

Approach

You can rearrange the array as $[1, 2, 4, 3]$.

- 1 becomes beautiful as there is no element before it.
- 2 becomes beautiful as the sum of elements before it is 1.
- 4 becomes beautiful as the sum of elements before it is 3.
- 3 is not beautiful as the sum of elements before it is 7 $(1 + 2 + 5)$ which is higher than 3.

Thus the answer is *3*.

Function description

Complete the function *countBeautiful* provided in the editor. This function takes the following *2* parameters and returns the required answer:

- *N*: Represents the size of array *A*
- *A*: Represents the elements of array *A*

Input format

- The first line contains a single integer *T* which denotes the number test cases. *T* also denotes the number of times you have to run the *countBeautiful*function on a different set of inputs..
- For each test case:
  - The first line contains an integer *N* denoting the size of the *A*

Input format

- The first line contains a single integer T which denotes the number of test cases. T also denotes the number of times you have to run the *countBeautiful* function on a different set of inputs..
- For each test case:
  - The first line contains an integer N denoting the size of the array.
  - The second line contains N space-separated integers denoting each element of the array.

Output format For each test case in a new line, print the maximum possible number of beautiful elements.

Constraints $1 \leq T \leq 100$

$1 \leq N \leq 10^5$

$1 \leq A[i] \leq 10^9$

Code snippets (also called starter code/boilerplate code)

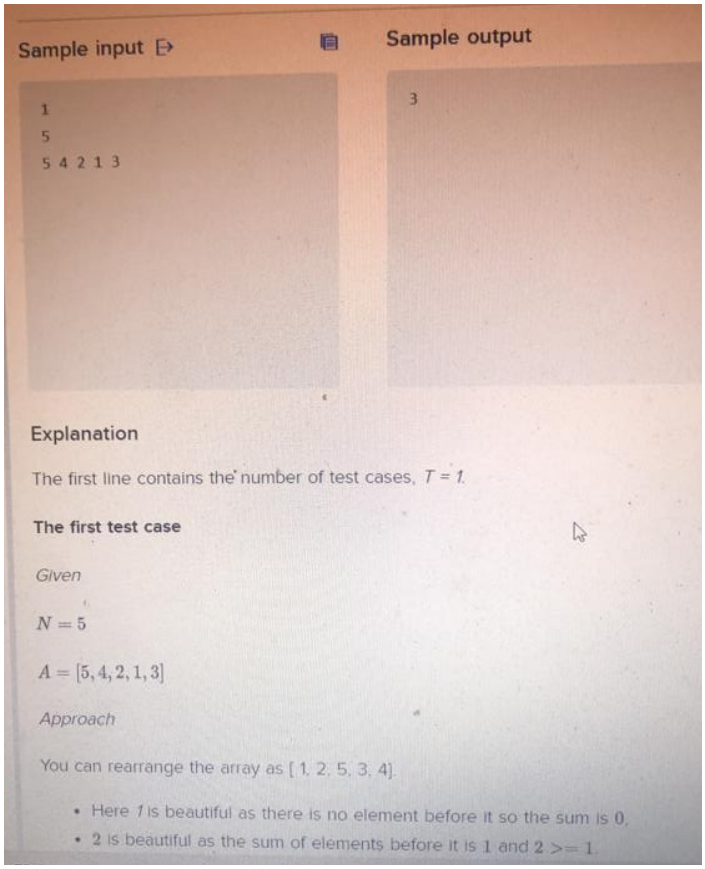
This question has code snippets for C, CPP, Java, and Python.

Sample input

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

Sample output

```
3
```



Explanation

The first line contains the number of test cases, $T = 1$.

The first test case

Given

$N = 5$

$A = [5, 4, 2, 1, 3]$

Approach

You can rearrange the array as [1, 2, 5, 3, 4].

- Here *1* is beautiful as there is no element before it so the sum is 0.
- 2 is beautiful as the sum of elements before it is 1 and $2 >= 1$.

$N = 5$

$A = [5, 4, 2, 1, 3]$

Approach

You can rearrange the array as [1, 2, 5, 3, 4].

- Here *1* is beautiful as there is no element before it so the sum is **0**.
- **2** is beautiful as the sum of elements before it is **1** and **2** >= **1**.
- **5** is beautiful as it's greater than the sum of elements before it which is **3**.
- After that sum of elements before becomes **8** which is higher than any other element.

Thus the answer is *3*.

The following test cases are the actual test cases of this question that may be used to evaluate your submission.

Sample input 1

```
100
4
65329221 12106895 91882089 6471867
5
37699009 57489855 90430685 3293923
1
53287868
4
79866982 3966291 50348020 38379364
2
```

Sample output 1

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

Given an array A of N strings, return all groups of strings that are anagrams.

Represent a group by a list of integers representing the index(1-based) in the original list. Look at the sample case for clarification.

NOTE: Anagram is a word, phrase, or name formed by rearranging the letters, such as 'spar', formed from 'rasp'.

Problem Constraints

$1 <= N <= 10^4$

$1 <= |A[i]| <= 10^4$

Each string consists only of lowercase characters.

The sum of the length of all the strings doesn't exceed 10^7

Input Format

The first and only argument is an integer array A.

Output Format

Return a two-dimensional array where each row describes a group.

Note:

Ordering of the result :
You should not change the relative ordering of the strings within the group suppose within a group containing A[i] and A[j], A[i] comes before A[j] if i < j.

Example Input

Input 1:

```
A = [cat, dog, god, tca]
```

Input 2:

```
A = [rat, tar, art]
```

Example Output

Output 1:

```
[ [1, 4],
  [2, 3] ]
```

Output 2:

```
[ [1, 2, 3] ]
```

Example Explanation

Explanation 1:

```
"cat" and "tca" are anagrams which correspond to index 1 and 4 and "dog" and "god" are another set of anagrams which correspond to index 2 and 3.
The indices are 1 based ( the first element has index 1 instead of index 0).
```

Explanation 2:

```
All three strings are anagrams.
```

Smallest String

Flag Question

You are given a string S which consists of digits from 0 to 9 and two integer R and A. There are two operations which you can do on a string.

1. Rotate
2. Add

You can use Rotate operation to rotate a string clockwise by R positions. For example if R=1 , string "581" will become "158" .
Also you can use Add operation to add number A to all odd indexes of string (0 based indexing). For example if A=3 , then string "781" will become "711". Digits post 9 are cycled back to zero.

You can use any of the two operation any number of times in any order to return the lexicographical smallest string.

Constraints

```
0<N<10

0<=A<100

1<=R<=N
```

For example

```
S="31"
R=1
A=4
Result: "11"
```

Decode String

Flag Question

Given an encoded string **A** consisting of lowercase English alphabets, square parentheses, and digits. The encoding rule is **X[encoded string]**, where the encoded_string inside the square brackets is being repeated exactly **X** times. Note that **X** is guaranteed to be a positive integer.

Find and return the decoded string.

Note:

1. You may assume that the original data does not contain any digits and that digits are only for those repeat numbers, X. For example, there won't be input like 3a or 2[4].
2. You may assume that the input string is always valid. No extra white spaces, square brackets are well-formed, etc.
3. You may assume that the length of the decoded string will not exced 200000.

Input Format

```
The only argument given is string A.
```

Output Format

```
Return the decoded string.
```

Constraints

```
1 <= length of the string <= 100000
```

For Example

```
Input 1:
    A = "3[a]2[bc]"
Output 1:
    "aaabcbc"

Input 2:
    A = "3[a2[c]]"
Output 2:
    accaccacc

Input 3:
    A = "2[abc]3[cd]ef"
Output 2:
    abcabccdcdcdef
```

H-index

Flag Question

Given an array of citations **A** (each citation is a non-negative integer) of a researcher, write a function to compute the researcher's h-index.

According to the definition of h-index: "A scientist has index **h** if **h** of his/her **N** papers have at least h citations each, and the other **N − h** papers have no more than **h** citations each."

Note: If there are several possible values for **h**, the maximum one is taken as the h-index.

Input Format:

```
First and only argument of input contains an integer array A of size N
```

Output Format:

```
return a single integer denoting h-index.
```

Constraints:

```
1 <= N <=3e6
1 <= A[i] <= 1e9
```

For Example:

```
Input 1:
    A = [3, 0, 6, 1, 5]
Output 1:
    3
Explanation 1:
    [3,0,6,1,5] means the researcher has 5 papers in total and each of them had received 3, 0, 6, 1, 5 citations respectively.
    Since the researcher has 3 papers with at least 3 citations each and the remaining two with no more than 3 citations each, her h-index is 3.
Input 2:
    A = [20, 10, 5]
Output 2:
    3
```