Find the smallest and second smallest element in an array

Basic Accuracy: 24.44% Submissions: 79K+ Points: 1

Given an array of integers, your task is to find the smallest and second smallest element in the array. If smallest and second smallest do not exist, print **-1**.

Example 1:

```
Input :
5
2 4 3 5 6
Output :
2 3
Explanation:
2 and 3 are respectively the smallest
and second smallest elements in the array.
```

Example 2:

```
Input :
6
1 2 1 3 6 7
Output :
1 2
Explanation:
1 and 2 are respectively the smallest
and second smallest elements in the array.
```

Your Task:

You don't need to read input or print anything. Your task is to complete the function **minAnd2ndMin()** which takes the array **A**[] and its size **N** as inputs and returns a **vector** containing the smallest and second smallest element if possible, else return **{-1,-1}**.

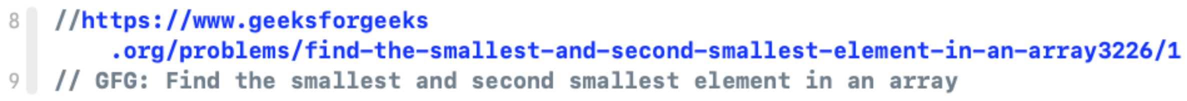
Expected Time Complexity: O(N)

Expected Auxiliary Space: O(1)

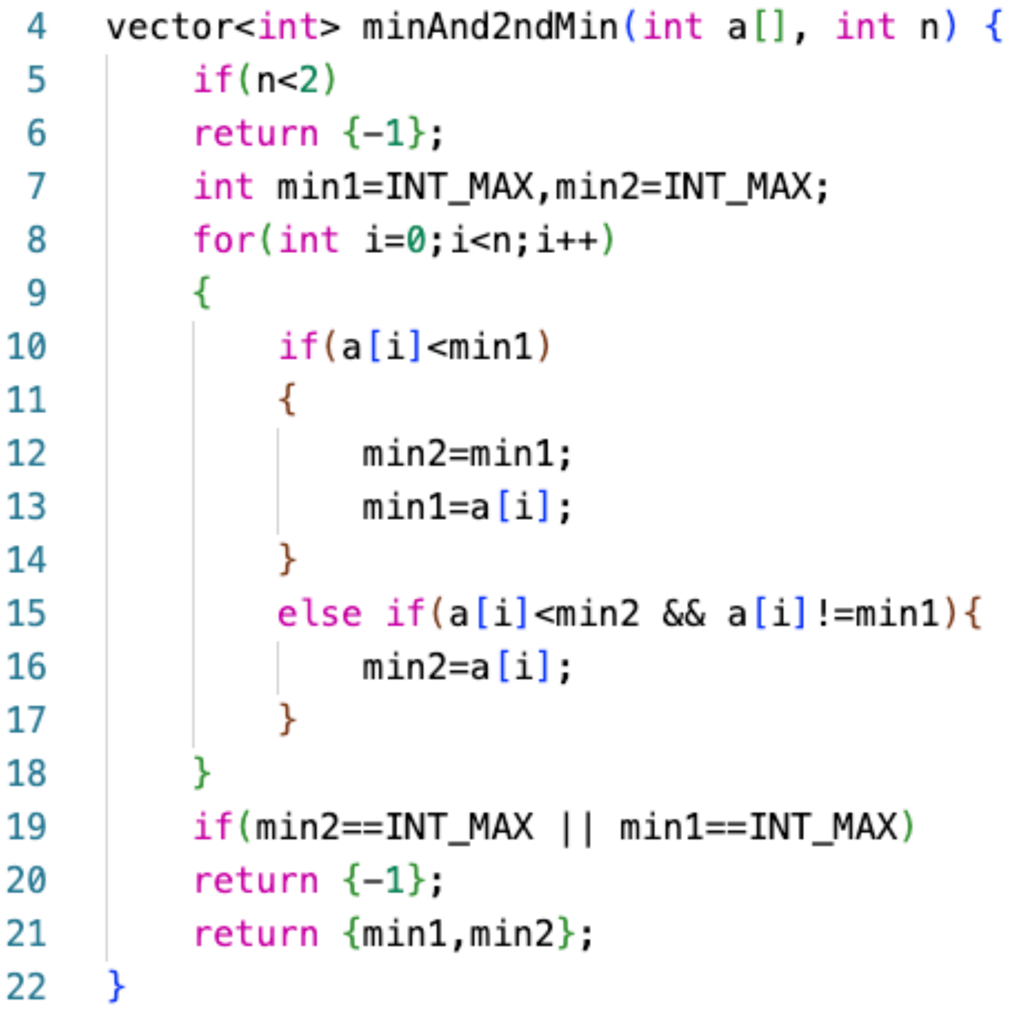
Constraints:

$1<=N<=10^5$

$1<=A[i]<=10^5$



```
//https://www.geeksforgeeks.org/problems/find-the-smallest-and-second-smallest-element-in-an-array3226/1
// GFG: Find the smallest and second smallest element in an array
```



```cpp
vector<int> minAnd2ndMin(int a[], int n) {
    if(n<2)
    return {-1};
    int min1=INT_MAX,min2=INT_MAX;
    for(int i=0;i<n;i++)
    {
        if(a[i]<min1)
        {
            min2=min1;
            min1=a[i];
        }
        else if(a[i]<min2 && a[i]!=min1){
            min2=a[i];
        }
    }
    if(min2==INT_MAX || min1==INT_MAX)
    return {-1};
    return {min1,min2};
}
```

A. Perfect Permutation

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given a positive integer n.

The weight of a permutation $p_1, p_2, \ldots, p_n$ is the number of indices $1 \le i \le n$ such that i divides p_i. Find a permutation $p_1, p_2, \ldots, p_n$ with the minimum possible weight (among all permutations of length n).

A permutation is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The only line of each test case contains a single integer n ($1 \le n \le 10^5$) — the length of permutation.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, print a line containing n integers $p_1, p_2, \ldots, p_n$ so that the permutation p has the minimum possible weight.

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

Example

input

```
2
1
4
```

output

```
1
2 1 4 3
```

Note

In the first test case, the only valid permutation is $p = [1]$. Its weight is 1.

In the second test case, one possible answer is the permutation $p = [2, 1, 4, 3]$. One can check that 1 divides p_1 and i does not divide p_i for $i = 2, 3, 4$, so the weight of this permutation is 1. It is impossible to find a permutation of length 4 with a strictly smaller weight.

```cpp
// https://codeforces.com/problemset/problem/1711/A
// CODEFORCES : PERFECT PERMUTATION

// we just write value (i-1) at ith place.
// So (i-1) will never be divisible by i and
// for i = 1 just write n as 1 will always be
// divisible by every number.
// Hence, the weight of the permutation will always be 1

#include <bits/stdc++.h>
using namespace std;

int main()
{
    int test_cases = 1;
    cin >> test_cases;

    for(int i=1;i<=test_cases;i++)
    {
        int n; cin >> n;
        cout << n << " ";
        for(int i=1; i<n; i++){
            cout << i << " ";
        }
        cout << "\n";
    }

    return 0;
}
```

Count of Maximum

Given an array A of length N, your task is to find the element which repeats in A maximum number of times as well as the corresponding count. In case of ties, choose the smaller element first.

Input

First line of input contains an integer T, denoting the number of test cases. Then follows description of T cases. Each case begins with a single integer N, the length of A. Then follow N space separated integers in next line. Assume that 1 <= T <= 100, 1 <= N <= 100 and for all i in [1..N] : 1 <= A[i] <= 10000

Output

For each test case, output two space separated integers V & C. V is the value which occurs maximum number of times and C is its count.

Sample 1:

Input

```
2
5
1 2 3 2 5
6
1 2 2 1 1 2
```

Output

```
2 2
1 3
```

Explanation:

In first case 2 occurs twice whereas all other elements occur only once. In second case, both 1 and 2 occur 3 times but 1 is smaller than 2.

```cpp
// https://www.codechef.com/problems/MAXCOUNT
// CODECHEF: MAXCOUNT

// Make a count array to count each element with the given value

#include <bits/stdc++.h>
using namespace std;

int main() {
    int test_cases; cin >> test_cases;
    vector <int> count(10001,0);
    while(test_cases--){
        int n,a; cin >> n;
        int max_value(0), max_count(0);
        for(int i=0;i<n; i++){
            cin >> a;
            count[a]++;
        }
        for(int i=10001; i>=1; i--){
            if(count[i]>=max_count){
                max_count = count[i];
                max_value = i;
            }
            count[i] = 0;
        }
        cout << max_value << " " << max_count << "\n";
    }
}
```

Leaders in an array

Easy Accuracy: 29.94% Submissions: 595K+ Points: 2

Given an array A of positive integers. Your task is to find the leaders in the array. An element of array is a leader if it is greater than or equal to all the elements to its right side. The rightmost element is always a leader.

Example 1:

```
Input:
n = 6
A[] = {16,17,4,3,5,2}
Output: 17 5 2
Explanation: The first leader is 17
as it is greater than all the elements
to its right.  Similarly, the next
leader is 5. The right most element
is always a leader so it is also
included.
```

Example 2:

```
Input:
n = 5
A[] = {1,2,3,4,0}
Output: 4 0
Explanation: 0 is the rightmost element
and 4 is the only element which is greater
than all the elements to its right.
```

Your Task:

You don't need to read input or print anything. The task is to complete the function **leader**() which takes array A and n as input parameters and returns an array of leaders in order of their appearance.

Expected Time Complexity: O(n)

Expected Auxiliary Space: O(n)

Constraints:

$1 <= n <= 10^7$

$0 <= A_i <= 10^7$

```cpp
// https://www.geeksforgeeks.org/problems/leaders-in-an-array-1587115620/1
// GFG : Leaders in an array

// Start from the back and track the current maximum of
// numbers from (i+1) till n-1 and check if it is smaller than or
// equal to given number

class Solution{
    public:
    vector<int> leaders(int a[], int n){
        int current_leader = 0;
        vector <int> ans;
        for(int i=n-1; i>=0; i--){
            if(a[i]>=current_leader){
                current_leader = a[i];
                ans.push_back(current_leader);
            }
        }
        reverse(ans.begin(),ans.end());
        return ans;
    }
};
```

3005. Count Elements With Maximum Frequency

Easy   Topics   Companies   Hint

You are given an array `nums` consisting of **positive** integers.

Return *the **total frequencies** of elements in* `nums` *such that those elements all have the **maximum** frequency*.

The **frequency** of an element is the number of occurrences of that element in the array.

Example 1:

```
Input: nums = [1,2,2,3,1,4]
Output: 4
Explanation: The elements 1 and 2 have a frequency of 2 which is the maximum
frequency in the array.
So the number of elements in the array with maximum frequency is 4.
```

Example 2:

```
Input: nums = [1,2,3,4,5]
Output: 5
Explanation: All elements of the array have a frequency of 1 which is the
maximum.
So the number of elements in the array with maximum frequency is 5.
```

Constraints:

- `1 <= nums.length <= 100`
- `1 <= nums[i] <= 100`

```cpp
// https://leetcode.com/problems/count-elements-with-maximum-frequency/description/
// LEETCODE : COUNT ELEMENTS WITH MAXIMUM FREQUENCY

// first find out the frequency of all numbers and track the max.
// frequency. Then just check if the given number is equal to
// max. frequency

class Solution {
public:
    int maxFrequencyElements(vector<int>& nums) {
        vector <int> frequency(101,0);
        int ans = 0;
        int max_freq = 0;
        for(auto x:nums){
            frequency[x]++;
            max_freq = max(max_freq,frequency[x]);
        }

        for(int i=1; i<=100; i++){
            if(frequency[i]==max_freq)
                ans++;
        }
        ans *= max_freq;
        return ans;
    }
};
```

2974. Minimum Number Game

Easy Topics Companies Hint

You are given a **0-indexed** integer array `nums` of **even** length and there is also an empty array `arr`. Alice and Bob decided to play a game where in every round Alice and Bob will do one move. The rules of the game are as follows:

- Every round, first Alice will remove the **minimum** element from `nums`, and then Bob does the same.
- Now, first Bob will append the removed element in the array `arr`, and then Alice does the same.
- The game continues until `nums` becomes empty.

Return *the resulting array* `arr`.

Example 1:

```
Input: nums = [5,4,2,3]
Output: [3,2,5,4]
Explanation: In round one, first Alice removes 2 and then Bob removes 3.
Then in arr firstly Bob appends 3 and then Alice appends 2. So arr = [3,2].
At the begining of round two, nums = [5,4]. Now, first Alice removes 4 and
then Bob removes 5. Then both append in arr which becomes [3,2,5,4].
```

Example 2:

```
Input: nums = [2,5]
Output: [5,2]
Explanation: In round one, first Alice removes 2 and then Bob removes 5.
Then in arr firstly Bob appends and then Alice appends. So arr = [5,2].
```

Constraints:

- `1 <= nums.length <= 100`
- `1 <= nums[i] <= 100`
- `nums.length % 2 == 0`

```cpp
// https://leetcode.com/problems/minimum-number-game/description/
// LEETCODE : MINIMUM NUMBER GAME

// Essentially what is happening is a sorted array will be made
// but the first two moves, the elements will be sorted reverse
// as [bob,alice] will be the order and
// alice < bob for a given move

class Solution {
public:
    vector<int> numberGame(vector<int>& nums) {
        sort(nums.begin(),nums.end());
        for(int i=0; i<nums.size(); i+=2){
            swap(nums[i],nums[i+1]);
        }
        return nums;
    }
};
```

Non-decreasing arrays

26104 90% 20 ★★★★★ 61 votes Arrays, C++, 1-D, Data Structures, Greedy

Details Submissions Discussion Similar Problems Editorial

Problem

You are given an array A consisting of N positive integers. Your task is to find an array B of length N satisfying the following conditions:

- $B_i > 0$ for all $1 \leq i \leq N$
- $B_i \leq B_{i+1}$, for all $1 \leq i < N$
- B_i is divisible by A_i for all $1 \leq i \leq N$
- $\sum_{i=1}^{N} B_i$ is minimum

You are given T test cases.

Input format

- The first line contains a single integer T denoting the number of test cases.
- The first line of each test case contains a single integer N denoting the length of the array.
- The second line of each test case contains N space-separated integers denoting the integer array A.

Output format

For each test case (in a separate line), print N space-separated integers denoting $B_1, B_2, .., B_N$. If there are multiple answers, you can print any of them. It is guaranteed that under the given constraints at least 1 B exists.

Constraints

$1 \leq T \leq 1000$
$1 \leq N \leq 2.5 \times 10^5$
$1 \leq A_i \leq 10^9$
$\text{Sum of N over all test cases does not exceed } 7.5 \times 10^5$

Sample Input	Sample Output

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

Time Limit: 1
Memory Limit: 256
Source Limit:

Explanation

Self explanatory.

```cpp
//https://www.hackerearth.com/practice/data-structures/arrays/1-d/practice-problems/algorithm/make-it-non-decreasing-7d3391fd/
// HACKEREARTH : Non-decreasing arrays

// iterating for a given i, we have
// we have to find b[i]
// which is smallest number divisible by a[i] and greater than b[i-1]
// hence, we can write
// b[i] = f*a[i] and f*a[i] > b[i-1]
// f > b[i-1]/a[i] => f = ceil(b[i-1]/a[i])
// Hence,
// b[i] = arr[i]*ceil(b[i-1]/arr[i])

#include <bits/stdc++.h>
using namespace std;

int main() {
    int test_cases;
    cin >> test_cases;
    while(test_cases--){
        int n;
        cin >> n;
        vector<long long> arr(n);

        for(int i = 0; i < n ; i++)
            cin >> arr[i];

        cout << arr[0];
        for(int i = 1; i < n ; i++){
            arr[i] = arr[i]*((arr[i-1] + arr[i] - 1)/arr[i]);
            cout << " " << arr[i];
        }
        cout << '\n';
    }
}
```