**Smallest subarray with all occurrences of a most frequent element**

Submissions: [628](https://practice.geeksforgeeks.org/problem_submissions.php?pid=2863)  Accuracy:

25.03%

   Difficulty: [Easy](https://practice.geeksforgeeks.org/Easy/0/0/)  Marks: 2

Show Topic Tags   

Given an array A. Let X be an element in the array which has the maximum frequency. The task is to find the smallest sub segment of the array which also has X as the maximum frequency element.

**Note:** if two or more elements have the same frequency (i.e., maximum frequency) and the same sub segment size then print the sub segmentwhich occurs first in the array.

**Input:**  
The first line of input contains an integer T denoting the number of test cases. Then T test cases follow. Each test case consists of two lines. First line of each test case contains an Integer N denoting size of array and the second line contains N space separated elements.

**Output:**  
For each test case, print the required sub segment of array in new line.

**Constraints:**  
1<=T<=100  
1<=N<=105  
1<=A[i]<=105

**Example:  
Input:**  
3  
8  
4 1 1 2 2 1 3 3  
5  
1 2 2 3 1  
6  
1 4 3 3 5 5  
**Output:**  
1 1 2 2 1  
2 2  
3 3

**Explanation:**

Input : A[] = {1, 2, 2, 3, 1}

Output : 2, 2

Note that there are two elements that appear

two times, 1 and 2. The smallest window for

1 is whole array and smallest window for 2 is

{2, 2}. Since window for 2 is smaller, this is

our output.

\*\* For More Input/Output Examples Use ['Expected Output'](https://practice.geeksforgeeks.org/problems/smallest-subarray-with-all-occurrences-of-a-most-frequent-element/0#ExpectOP) option \*\*

[Author: arun03](https://auth.geeksforgeeks.org/user/arun03/practice/)

<https://practice.geeksforgeeks.org/problems/smallest-subarray-with-all-occurrences-of-a-most-frequent-element/0>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

public class GFG

{

static string FindMaxFrec(int[] a, int n)

{

int max\_val = 1, max\_key = a[0];

Dictionary<int, int> diccio = new Dictionary<int, int>();

for (int i = 0; i < n; i++)

{

if (diccio.ContainsKey(a[i]))

{

diccio[a[i]]++;

if (diccio[a[i]] > max\_val)

{

max\_val = diccio[a[i]];

max\_key = a[i];

}

}

else

{

diccio[a[i]] = 1;

}

}

//Console.WriteLine(max\_key + " " + max\_val);

Dictionary<int, int> distMaxVal = new Dictionary<int, int>();

//buscar el subarray mas corto

Dictionary<int, int[]> distanciasMinimasDelMaximo = new Dictionary<int, int[]>();

int max = 0;

for (int i = 0; i < a.Length; i++)

{

if (diccio[a[i]] == max\_val)

{

if (distMaxVal.ContainsKey(a[i]))

{

// distMaxVal[a[i]] = Array.LastIndexOf(a, a[i]) - distMaxVal[a[i]];

max = Math.Max(max, i - distMaxVal[a[i]]);

distanciasMinimasDelMaximo[a[i]][1] = i;

distanciasMinimasDelMaximo[a[i]][2] = i - distanciasMinimasDelMaximo[a[i]][0];

}

else

{

distMaxVal[a[i]] = i;

distanciasMinimasDelMaximo[a[i]] = new int[3];

distanciasMinimasDelMaximo[a[i]][0] = i;

}

}

}

int[] min = new int[3];

min[2] = int.MaxValue;

foreach (KeyValuePair<int, int[]> kvp in distanciasMinimasDelMaximo)

{

//Console.WriteLine(kvp.Key + " " + kvp.Value[0] + " " + kvp.Value[1] + " " + kvp.Value[2]);

if (kvp.Value[2] < min[2])

{

min[0] = kvp.Value[0];

min[1] = kvp.Value[1];

min[2] = kvp.Value[2];

}

}

StringBuilder sb = new StringBuilder();

for (int i = min[0]; i <= min[1]; i++)

{

//Console.Write(a[i] + " ");

sb.Append(a[i]);

sb.Append(" ");

}

return sb.ToString().Trim();

}

static void Main(string[] args)

{

//Input:

//84

//int[] arr = Array.ConvertAll("87 78 16 94 36 87 93 50 22 63 28 91 60 64 27 41 27 73 37 12 69 68 30 83 31 63 24 68 36 30 3 23 59 70 68 94 57 12 43 30 74 22 20 85 38 99 25 16 71 14 27 92 81 57 74 63 71 97 82 6 26 85 28 37 6 47 30 14 58 25 96 83 46 15 68 35 65 44 51 88 9 77 79 89".Split(' '), e => int.Parse(e));

//FindMaxFrec(arr, arr.Length);

//int[] arrp = Array.ConvertAll("30 83 31 63 24 68 36 30 3 23 59 70 68 94 57 12 43 30 74 22 20 85 38 99 25 16 71 14 27 92 81 57 74 63 71 97 82 6 26 85 28 37 6 47 30".Split(' '), e => int.Parse(e));

//Console.WriteLine("\n" + arrp.Length);

//Its Correct output is:

//30 83 31 63 24 68 36 30 3 23 59 70 68 94 57 12 43 30 74 22 20 85 38 99 25 16 71 14 27 92 81 57 74 63 71 97 82 6 26 85 28 37 6 47 30

int t = int.Parse(Console.ReadLine());

while (t-- > 0)

{

int n = int.Parse(Console.ReadLine());

int[] arr = Array.ConvertAll(Console.ReadLine().Trim().Split(' '), e => int.Parse(e));

Console.WriteLine(FindMaxFrec(arr, arr.Length));

}

// int[] arr = { 4, 1, 1, 2, 2, 1, 3, 3,3 };

////FindMaxFrec(arr, arr.Length);

//int[] arr = { 1, 5, 3, 2, 3, 1, 2, 3 };

//FindMaxFrec(arr, arr.Length);

//int[] arr = Array.ConvertAll("30 83 31 63 24 68 36 30 3 23 59 70 68 94 57 12 43 30 74 22 20 85 38 99 25 16 71 14 27 92 81 57 74 63 71 97 82 6 26 85 28 37 6 47 30 98 3 18 93 53 57 2 81 87 42 66 90 45 20 41 30 32".Split(' '), e => int.Parse(e));

//int[] arr = Array.ConvertAll("87 78 16 94 36 87 93 50 22 63 28 91 60 64 27 41 27 73 37 12 69 68 30 83 31 63 24 68 36 30 3 23 59 70 68 94 57 12 43 30 74 22 20 85 38 99 25 16 71 14 27 92 81 57 74 63 71 97 82 6 26 85 28 37 6 47 30 14 58 25 96 83 46 15 68 35 65 44 51 88 9 77 79 89".Trim().Split(' '), e => int.Parse(e));

// int[] arr = { 100000, 200000, 200000, 300000, 100000 };

//int[] arr = { 4, 4, 1, 2, 3, 1, 2, 3, 1, 2, 3 };

// Console.WriteLine( FindMaxFrec(arr, arr.Length));

//Console.WriteLine(arr.Length);

Console.ReadLine();

}

}

<https://www.geeksforgeeks.org/smallest-subarray-with-all-occurrences-of-a-most-frequent-element/>

// C++ implementation to find smallest

// subarray with all occurrences of

// a most frequent element

#include <bits/stdc++.h>

using namespace std;

void smallestSubsegment(int a[], int n)

{

    // To store left most occurrence of elements

    unordered\_map<int, int> left;

    // To store counts of elements

    unordered\_map<int, int> count;

    // To store maximum frequency

    int mx = 0;

    // To store length and starting index of

    // smallest result window

    int mn, strindex;

    for (int i = 0; i < n; i++) {

        int x = a[i];

        // First occurrence of an element,

        // store the index

        if (count[x] == 0) {

            left[x] = i;

            count[x] = 1;

        }

        // increase the frequency of elements

        else

            count[x]++;

        // Find maximum repeated element and

        // store its last occurrence and first

        // occurrence

        if (count[x] > mx) {

            mx = count[x];

            mn = i - left[x] + 1; // length of subsegment

            strindex = left[x];

        }

        // select subsegment of smallest size

        else if (count[x] == mx && i - left[x] + 1 < mn) {

            mn = i - left[x] + 1;

            strindex = left[x];

        }

    }

    // Print the subsegment with all occurrences of

    // a most frequent element

    for (int i = strindex; i < strindex + mn; i++)

        cout << a[i] << " ";

}

// Driver code

int main()

{

    int A[] = { 1, 2, 2, 2, 1 };

    int n = sizeof(A) / sizeof(A[0]);

    smallestSubsegment(A, n);

    return 0;

}