# L5 : Language Tools and Time and Space Complexity Practice Questions

## 1-Ass : Rotate array

Send Feedback

You have been given a random integer array/list(ARR) of size N. Write a function that rotates the given array/list by D elements(towards the left).

**Note:**

Change in the input array/list itself. You don't need to return or print the elements.

**Input format :**

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.
First line of each test case or query contains an integer 'N' representing the size of the array/list.
Second line contains 'N' single space separated integers representing the elements in the array/list.
Third line contains the value of 'D' by which the array/list needs to be rotated.

**Output Format :**

For each test case, print the rotated array/list in a row separated by a single space.
Output for every test case will be printed in a separate line.

**Constraints :**

1 <= t <= 10^4
0 <= N <= 10^6
0 <= D <= N
Time Limit: 1 sec

**Sample Input 1:**

```
1
7
1 2 3 4 5 6 7
2
```

**Sample Output 1:**

```
3 4 5 6 7 1 2
```

**Sample Input 2:**

```
2
7
1 2 3 4 5 6 7
0
4
1 2 3 4
2
```

**Sample Output 2:**

```
1 2 3 4 5 6 7

3 4 1 2
```

```
#include<bits/stdc++.h>
void rotate(int *input, int d, int n)
{
    //Write your code here
    reverse(input,input+n);
    reverse(input,input+(n-d));
    reverse(input+(n-d),input+n);
}
```

## 2-Ass : Pair sum to 0

Send Feedback

Given a random integer array A of size N. Find and print the count of pair of elements in the array which sum up to 0.

Note: Array A can contain duplicate elements as well.

**Input format:**

The first line of input contains an integer, that denotes the value of the size of the array. Let us denote it with the symbol N.
The following line contains N space separated integers, that denote the value of the elements of the array.

**Output format :**

The first and only line of output contains the count of pair of elements in the array which sum up to 0.

**Constraints :**

0 <= N <= 10^4
Time Limit: 1 sec

**Sample Input 1:**

```
5
2 1 -2 2 3
```

**Sample Output 1:**

```
2
```

```
#include<bits/stdc++.h>
int pairSum(int *arr, int n) {
        // Write your code here
    unordered_map<int,int> mymap;
    for(int i = 0; i < n; i++){
        mymap[arr[i]]++;
    }

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

        if(arr[i] == 0){
        continue;
        }

        int comp = -arr[i];
        if(mymap[comp] != 0){
            count += (mymap[arr[i]] * mymap[comp]);
            mymap[arr[i]] = 0;
            mymap[comp] = 0;
        }

    }
    int countzero = mymap[0];
    int result = count + ( (countzero)*(countzero-1) )/2;
    return result;
}
```
Note : if less than 20% test case are failing due to TLE  then try to optimize the check condition's, mostly it works

## 3-Ass : Longest Consecutive Sequence

Send Feedback

You are given an array of unique integers that contain numbers in random order. You have to find the longest possible sequence of consecutive numbers using the numbers from given array.

You need to return the output array which contains starting and ending element. If the length of the longest possible sequence is one, then the output array must contain only single element.

**Note:**

1. Best solution takes O(n) time.
2. If two sequences are of equal length, then return the sequence starting with the number whose occurrence is earlier in the array.

**Input format:**

The first line of input contains an integer, that denotes the value of the size of the array. Let us denote it with the symbol n.

The following line contains n space separated integers, that denote the value of the elements of the array.

**Output format:**

The first and only line of output contains starting and ending element of the longest consecutive sequence. If the length of longest consecutive sequence, then just print the starting element.

**Constraints :**

0 <= n <= 10^6

Time Limit: 1 sec

**Sample Input 1 :**

```
13
2 12 9 16 10 5 3 20 25 11 1 8 6
```

**Sample Output 1 :**

```
8 12
```

**Sample Input 2 :**

```
7
3 7 2 1 9 8 41
```

**Sample Output 2 :**

```
7 9
```

Explanation: Sequence should be of consecutive numbers. Here we have 2 sequences with same length i.e. [1, 2, 3] and [7, 8, 9], but we should select [7, 8, 9] because the starting point of [7, 8, 9] comes first in input array and therefore, the output will be 7 9, as we have to print starting and ending element of the longest consecutive sequence.

**Sample Input 3 :**

```
7
15 24 23 12 19 11 16
```

**Sample Output 3 :**

```
15 16
```

```
#include<bits/stdc++.h>
vector<int> longestConsecutiveIncreasingSequence(int *arr, int n) {
    // Your Code goes here
    int* dp = new int[n]();
    unordered_map<int,int> m;
    int index = 0;

    for(int i=0;i<n;i++){
        m[arr[i]] = index++; //storing rank for first occurance
    }
    m[-1] = n+1;
    for(int i=0;i<n;i++){
        dp[i] = 1;
    }

    sort(arr,arr+n);
    for(int i=1;i<n;i++){
        if(arr[i-1]+1 == arr[i]){
            dp[i] = dp[i-1] + 1; //if prev element is consecutive
        }
    }

    int maxi = INT_MIN; //max length of subsequence possible
    for(int i=0;i<n;i++){
        maxi = max(maxi,dp[i]);
    }

    vector<int> ans;
    for(int i=0;i<n;i++){
        if(dp[i] == maxi){ //all possible subsequences of length "maximum"
            ans.push_back(arr[i]-dp[i]+1);
        }
    }

    int idx = n+1;
    int finalAns; //among all possible answers onw which has first occurance is original array
         for(int i=0;i<ans.size();i++){
        if(m[ans[i]]<idx){
            idx = m[ans[i]];
            finalAns = ans[i];
        }
    }

    vector<int> v;
    v.push_back(finalAns);
    v.push_back(finalAns+maxi-1);

    return v;

}
```

## 4-Ass : Duplicate in array

Send Feedback

You have been given an integer array/list(ARR) of size N which contains numbers from 0 to (N - 2). Each number is present at least once. That is, if N = 5, the array/list constitutes values ranging from 0 to 3, and among these, there is a single integer value that is present twice. You need to find and return that duplicate number present in the array.

**Note :**

Duplicate number is always present in the given array/list.

**Input format :**

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.
First line of each test case or query contains an integer 'N' representing the size of the array/list.
Second line contains 'N' single space separated integers representing the elements in the array/list.

**Output Format :**

For each test case, print the duplicate element in the array/list.
Output for every test case will be printed in a separate line.

**Constraints :**

1 <= t <= 10^2
0 <= N <= 10^6
Time Limit: 1 sec

**Sample Input 1:**

```
1
9
0 7 2 5 4 7 1 3 6
```

**Sample Output 1:**

```
7
```

**Sample Input 2:**

```
2
5
0 2 1 3 1
7
0 3 1 5 4 3 2
```

**Sample Output 2:**

```
1

3
```

```
#include<bits/stdc++.h>
int findDuplicate(int *arr, int n)
{
    //Write your code here
    // we can solve it through both way using map as well as set
    map<int,int> m;
    for(int i = 0; i < n; i++){
        m[arr[i]]++;

        if(m[arr[i]] > 1){
            return arr[i];
        }
    }

}
```

Note : Using map ( BST ) TLE is coming while unordered_map ( Hashmap ) all test cases passed.

```
#include<bits/stdc++.h>
int findDuplicate(int *arr, int n)
{
    //Write your code here
    // we can solve it through both way using map as well as set
    unordered_map<int,int> m;
    for(int i = 0; i < n; i++){
        m[arr[i]]++;

        if(m[arr[i]] > 1){
            return arr[i];
        }
    }

}
```

Using set :

```
#include<bits/stdc++.h>
int findDuplicate(int *arr, int n)
{
    //Write your code here
    // we can solve it through both way using map as well as set
    set<int> s;
    for(int i = 0; i < n; i++){
        if(s.find(arr[i]) != s.end()){
            return arr[i];
        }
        s.insert(arr[i]);
    }

}
```

## 5-Ass : Triplet sum

Send Feedback

You have been given a random integer array/list(ARR) and a number X. Find and return the triplet(s) in the array/list which sum to X.

**Note :**

Given array/list can contain duplicate elements.

**Input format :**

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.
First line of each test case or query contains an integer 'N' representing the size of the first array/list.
Second line contains 'N' single space separated integers representing the elements in the array/list.
Third line contains an integer 'X'.

**Output format :**

For each test case, print the total number of triplets present in the array/list.

Output for every test case will be printed in a separate line.

**Constraints :**

1 <= t <= 10^2
0 <= N <= 10^3
0 <= X <= 10^9

Time Limit: 1 sec

**Sample Input 1:**

```
1
7
1 2 3 4 5 6 7
12
```

**Sample Output 1:**

```
5
```

**Sample Input 2:**

```
2
7
1 2 3 4 5 6 7
19
9
2 -5 8 -6 0 5 10 11 -3
10
```

**Sample Output 2:**

```
0
5
```

**Explanation for Input 2:**

Since there doesn't exist any triplet with sum equal to 19 for the first query, we print 0.

For the second query, we have 5 triplets in total that sum up to 10. They are, (2, 8, 0), (2, 11, -3), (-5, 5, 10), (8, 5, -3) and (-6, 5, 11)

```
int tripletSum(int *arr, int n, int num)
{
             //Write your code here
    sort(arr,arr+n);
    int numTriplets = 0;

    for(int i =0; i < n-2; i++)
    {
        int l = i+1;
        int r = n-1;
        int count = 0;
        while(l < r){
            if(arr[i] + arr[l] + arr[r] == num){

                if (arr[l] == arr[r])
                {
                    int totalElementsFromStartToEnd = (r -l) + 1;
                    count += (totalElementsFromStartToEnd * (totalElementsFromStartToEnd - 1) / 2);
                    break;
                }
                int tempStartIndex = l + 1;
                int tempEndIndex = r - 1;
                while (tempStartIndex <= tempEndIndex && arr[tempStartIndex] == arr[l])
                {
                    tempStartIndex += 1;

                }
                while (tempEndIndex >= tempStartIndex && arr[tempEndIndex] == arr[r])
                {
                    tempEndIndex -= 1;
                }
                int totalElementsFromStart = (tempStartIndex - l);
                int totalElementsFromEnd = (r - tempEndIndex);
                count += (totalElementsFromStart * totalElementsFromEnd);
                l = tempStartIndex; r = tempEndIndex;

            }else if(arr[i] + arr[l] + arr[r] < num){
                l++;
            }
            else{
                r--;
            }
        }
        numTriplets+=count;
    }

  return numTriplets;
}
```

## 6-Ass : **Find the Unique Element**

Send Feedback

You have been given an integer array/list(ARR) of size N. Where N is equal to [2M + 1].

Now, in the given array/list, 'M' numbers are present twice and one number is present only once.

You need to find and return that number which is unique in the array/list.

**Note:**

Unique element is always present in the array/list according to the given condition.

**Input format :**

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.
First line of each test case or query contains an integer 'N' representing the size of the array/list.
Second line contains 'N' single space separated integers representing the elements in the array/list.

**Output Format :**

For each test case, print the unique element present in the array.
Output for every test case will be printed in a separate line.

**Constraints :**

1 <= t <= 10^2
0 <= N <= 10^6
Time Limit: 1 sec

**Sample Input 1:**

```
1
7
2 3 1 6 3 6 2
```

**Sample Output 1:**

```
1
```

**Sample Input 2:**

```
2
5
2 4 7 2 7
9
1 3 1 3 6 6 7 10 7
```

**Sample Output 2:**

```
4

10
```

```
int findUnique(int *arr, int n) {
    // Write your code here
    int res = arr[0];
    for(int i = 1;  i < n; i++){
        res = res^arr[i];
    }
    return res;
}
```