Lecture 2 - Arrays 2

Code Binary Search

Send Feedback You have been given a sorted(in ascending order)integer array/list(ARR)of size *N* and an element X.Write a function to search *this* element in the given input array/list using'Binary Search'.Return the index of the element in the input array/list.In case the element is not present in the array/list,then return-1. *Input* format:*The* first line contains an Integer'N'which denotes the size of the array/list.

Second line contains'N'single space separated integers representing the elements in the array/list.

Third line contains an Integer't'which denotes the number of test cases or queries to be run.Then the test cases follow..

All the't'lines henceforth,will take the value of *X* to be searched for in the array/*list*.*Output* Format:*For* each test case,print the index at which *X* is present,-1 otherwise.

Output for every test case will be printed in a separate line.Constraints:1<=t<=10^4 0<=N<=10^6 0<=X<=10^9 *Time* Limit:1 sec *Sample* Input 1:7 1 3 7 9 11 12 45 1 3 *Sample* Output 1:1 *Sample* Input 2:7 1 2 3 4 5 6 7 2 9 7 *Sample* Output 2:-1 6

```java
public class Solution {
    public static int binarySearch(int[] arr, int x) {
        int start = 0, end = arr.length - 1, mid = 0;
        while (start <= end) {
            mid = (start + end) / 2;
            if (x == arr[mid])
                return mid;
            else if (x > arr[mid])
                start = mid + 1;
            else
                end = mid - 1;
        }
        return -1;
    }
}
```

Selection Sort

Send Feedback
Provided with a random integer array/list(ARR) of size N, you have been required to sort *this* array using 'Selection Sort'.
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.
Output format :
For each test case, print the elements of the array/list in sorted order
separated by a single space.

Output for every test case will be printed in a separate line.
Constraints :
1 <= t <= 10^2
0 <= N <= 10^3
Time Limit: 1 sec
Sample Input 1:
1
7
2 13 4 1 3 6 28
Sample Output 1:
1 2 3 4 6 13 28
Sample Input 2:
2
5
9 3 6 2 0
4
4 3 2 1
Sample Output 2:
0 2 3 6 9
1 2 3 4

```java
public class Solution {

    public static void selectionSort(int[] arr) {
        Your code goes here
        int min = 0;
        for(int i=0; i<arr.length-1; i++){
            min = i;
            for(int j=i+1; j<arr.length; j++){
                min = arr[min]>arr[j]?j:min;
                // if(arr[j]<arr[min])
                //     min = j;
            }
            if(min!=i){
                arr[i] = arr[i]+arr[min];
                arr[min] = arr[i]-arr[min];
                arr[i] = arr[i]-arr[min];
            }
            // int temp = arr[min];
            // arr[min] = arr[i];
            // arr[i] = temp;
        }
    }
}
```

Code Bubble Sort

Provided with a random integer array/list(ARR) of size N, you have been required to sort this array using 'Bubble Sort'.

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.

Output format :

For each test case, print the elements of the array/list in sorted order separated by a single space.

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

Constraints :

$1 <= t <= 10^2$

$0 <= N <= 10^3$

Time Limit: 1 sec

Sample Input 1:
```
1
7
2 13 4 1 3 6 28
```
Sample Output 1:
```
1 2 3 4 6 13 28
```
Sample Input 2:
```
2
5
9 3 6 2 0
4
4 3 2 1
```
Sample Output 2:
```
0 2 3 6 9
1 2 3 4
```

```java
public class Solution {
    public static void bubbleSort(int[] a){
        //Your code goes here
        for(int i=0; i<a.length; i++){
            for(int j=0; j<a.length-1-i; j++){
                if(a[j]>a[j+1]){
                    a[j] = a[j]+a[j+1];
                    a[j+1] = a[j]-a[j+1];
                    a[j] = a[j]-a[j+1];
                }
            }
```

```
        }
    }
}
```

Code_Insertion_Sort

Provided with a random integer array/list(ARR) of size N, you have been required to sort this array using 'Insertion Sort'.
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.
Output Format :
For each test case, print the elements of the array/list in sorted order separated by a single space.

Output for every test case will be printed in a separate line.
Constraints :
1 <= t <= 10^2
0 <= N <= 10^3
Time Limit: 1 sec
Sample Input 1:
1
7
2 13 4 1 3 6 28
Sample Output 1:
1 2 3 4 6 13 28
Sample Input 2:
2
5
9 3 6 2 0
4
4 3 2 1
Sample Output 2:
0 2 3 6 9
1 2 3 4

```java
public class Solution {

    public static void swap(int arr[], int a, int b){
        arr[a] = arr[a] + arr[b];
        arr[b] = arr[a] - arr[b];
        arr[a] = arr[a] - arr[b];
    }

    public static void insertionSort(int[] arr) {
```

```
        //Your code goes here
        for(int i=0; i<arr.length-1; i++){
            for(int j=i+1; j>0; j--){
                if(arr[j]<arr[j-1]){
                    swap(arr, j, j-1);
                }
                else
                    break;
            }
        }
    }
}
```

Code_Merge_Two_Sorted_Arrays

You have been given two sorted arrays/lists(*ARR1* and ARR2) of size *N* and *M* respectively, merge them into a third array/list such that the third array is also sorted.

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 of the first array/list.

Third line contains an integer 'M' representing the size of the second array/list.

Fourth line contains 'M' single space separated integers representing the elements of the second array/list.

Output Format :

For each test case, print the sorted array/list(of size N + M) in a single row, separated by a single space.

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

Constraints :

1 <= t <= 10^2

0 <= *N* <= 10^5

0 <= *M* <= 10^5

Time Limit: 1 sec

```
Sample Input 1 :
1
5
1 3 4 7 11
4
2 4 6 13
Sample Output 1 :
1 2 3 4 4 6 7 11 13
Sample Input 2 :
2
3
10 100 500
7
4 7 9 25 30 300 450
4
7 45 89 90
0
Sample Output 2 :
4 7 9 10 25 30 100 300 450 500
7 45 89 90

public class Solution {
    public static int[] merge(int arr1[], int arr2[]) {
        //Your code goes here
        int p1 = 0, p2 = 0;
        int[] arr3 = new int[arr1.length+arr2.length];
        int i=0;
        while(p1<arr1.length && p2<arr2.length){
            if(arr1[p1]<arr2[p2]){
                arr3[i++] = arr1[p1++];
            }
            else if(arr1[p1]>arr2[p2]){
                arr3[i++] = arr2[p2++];
            }
            else{
                arr3[i++] = arr1[p1++];
                arr3[i++] = arr2[p2++];

            }
        }
        while(p2<arr2.length){
            arr3[i++] = arr2[p2++];
        }
        while(p1<arr1.length){
            arr3[i++] = arr1[p1++];
        }
```

```
        return arr3;
    }
}
```

You have been given a random integer array/list(ARR) of size N. You have
been required to push all the zeros that are present in the array/list to
the end of it. Also, make sure to maintain the relative order of the
non-zero elements.
Note:
Change in the input array/list itself. You don't need to return or print
the elements.

You need to do this in one scan of array only. Don't use extra space.

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 elements of the array/list in the desired
order separated by a single space.

Output for every test case will be printed in a separate line.
Constraints :
1 <= t <= 10^2
0 <= N <= 10^5
Time Limit: 1 sec
Sample Input 1:
1
7
2 0 0 1 3 0 0
Sample Output 1:
2 1 3 0 0 0 0
Explanation for the *Sample* Input 1 :
All the zeros have been pushed towards the end of the array/list. Another
important fact is that the order of the non-zero elements have been
maintained as they appear in the input array/list.
Sample Input 2:
2
5
0 3 0 2 0
5
9 0 0 8 2
Sample Output 2:
3 2 0 0 0
9 8 2 0 0

```java
public class Solution {

    // public static void swap(int arr[], int a, int b){
    // arr[a] = arr[a] + arr[b];
    // arr[b] = arr[a] - arr[b];
    // arr[a] = arr[a] - arr[b];
    // }

    public static void pushZerosAtEnd(int[] arr) {
        //Your code goes here
        // int count = 0;
        int[] arr2 = new int[arr.length];
        int k=0;
        for(int i=0; i<arr.length; i++){
            // if(arr[i]==0)
            //     for(int j=i+1; j<arr.length; j++){
            //         if(arr[j]!=0){
            //             swap(arr, i, j);
            //             count++;
            //             break;
            //         }
            //     }
            if(arr[i]!=0)
                arr2[k++] = arr[i];
        }
        while(k<arr.length)
            arr2[k++] = 0;
        k = 0;
        while(k<arr.length)
            arr[k] = arr2[k++];
    }
}
```

Rotate array

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

```java
public class Solution {

    public static void rotate(int[] arr, int d) {
        //Your code goes here
        int[] temp = new int[arr.length];
        int k = 0;
        for(int i=d; i<arr.length; i++){
            temp[k++] = arr[i];
        }
        for(int i=0; i<d; i++){
            temp[k++] = arr[i];
        }
        for(int i=0; i<arr.length; i++){
            arr[i] = temp[i];
        }
    }
}
```

Second Largest in array
Send Feedback
You have been given a random integer array/list(ARR) of size N. You are
required to find and return the second largest element present in the
array/list.

If N <= 1 or all the elements are same in the array/list then return
-2147483648 or -2 ^ 31(It is the smallest value for the range of Integer)
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.

The first line of each test case or query contains an integer 'N'
representing the size of the array/list.

The second line contains 'N' single space separated integers representing
the elements in the array/list.
Output Format :
For each test case, print the second largest in the array/list if exists,
-2147483648 otherwise.

Output for every test case will be printed in a separate line.
Constraints :
1 <= t <= 10^2
0 <= N <= 10^5

Time Limit: 1 sec
Sample Input 1:
1
7
2 13 4 1 3 6 28
Sample Output 1:
13
Sample Input 2:
1
5
9 3 6 2 9
Sample Output 2:
6
Sample Input 3:
2
2
6 6
4
90 8 90 5
Sample Output 3:
-2147483648
8

public class Solution {

    public static int secondLargestElement(int[] arr) {
        //Your code goes here
        int max = Integer.MIN_VALUE;
        int max2 = Integer.MIN_VALUE;

        for(int i=0; i<arr.length; i++){
            if(arr[i]>max)
                max = arr[i];
```

```
        }
        for(int i=0; i<arr.length; i++){
            if(arr[i]!=max && arr[i]>max2)
                max2 = arr[i];
        }
        return max2;
    }
}
```

Check Array Rotation

You have been given an integer array/list(ARR) of size N. It has been sorted(in increasing order) and then rotated by some number 'K' in the right hand direction.
Your task is to write a function that returns the value of 'K', that means, the index from which the array/list has been rotated.
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 value of 'K' or the index from which which the array/list has been rotated.

Output for every test case will be printed in a separate line.
Constraints :
1 <= t <= 10^2
0 <= N <= 10^5
Time Limit: 1 sec
Sample Input 1:
1
6
5 6 1 2 3 4
Sample Output 1:
2
Sample Input 2:
2
5
3 6 8 9 10
4
10 20 30 1
Sample Output 2:
0
3

```
public class Solution {

    public static int arrayRotateCheck(int[] arr){
        //Your code goes here
```

```
        int count = 0;
        for(int i=1; i<arr.length; i++){
            if(arr[i-1]<=arr[i])
                count++;
            else
                break;
        }
        if(arr.length==0 || arr.length == count+1)
            return 0;
        return count+1;
    }

}
```

Sort_0_1_2

You are given an integer array/list(ARR) of size N. It contains only 0s, 1s and 2s. Write a solution to sort *this* array/list in a 'single scan'. 'Single Scan' refers to iterating over the array/list just once or to put it in other words, you will be visiting each element in the array/list just once.
Note:
You need to change in the given array/list itself. Hence, no need to return or print anything.
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(all 0s, 1s and 2s) representing the elements in the array/list.
Output Format :
For each test *case*, print the sorted array/list elements in a row separated by a single space.

Output for every test *case* will be printed in a separate line.
Constraints :
1 <= t <= 10^2
0 <= *N* <= 10^5
Time Limit: 1 sec
Sample Input 1:
1
7
0 1 2 0 2 0 1
Sample Output 1:
0 0 0 1 1 2 2
Sample Input 2:
2
5
2 2 0 1 1
7

```
0 1 2 0 1 2 0
Sample Output 2:
0 1 1 2 2
0 0 0 1 1 2 2

public class Solution {

    public static void sort012(int[] arr){
        //Your code goes here
        int count0=0, count1=0;
        for(int i=0; i<arr.length; i++){
            if(arr[i]==0)
                count0++;
            else if(arr[i]==1)
                count1++;
        }

        int i=0;
        while(i<count0)
            arr[i++] = 0;
        while(i<count1+count0)
            arr[i++] = 1;
        while(i<arr.length)
            arr[i++] = 2;

    }

}
```

Sum of Two Arrays

Two random integer arrays/lists have been given as *ARR1* and *ARR2* of size *N* and *M* respectively. Both the arrays/lists contain numbers from 0 to 9(i.e. single digit integer is present at every index). The idea here is to represent each array/list as an integer in itself of digits *N* and M.
You need to find the sum of both the input arrays/list treating them as two integers and put the result in another array/list i.e. output array/list will also contain only single digit at every index.
Note:
The sizes *N* and *M* can be different.

Output array/list(of all 0s) has been provided as a function argument. Its size will always be one more than the size of the bigger array/list. Place 0 at the 0th index if there is no carry.

No need to print the elements of the output array/list.
Using the function "sumOfTwoArrays", write the solution to the problem and store the answer inside *this* output array/list. The main code will handle the printing of the output on its own.
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 of the first array/list.

Third line contains an integer 'M' representing the size of the second array/list.

Fourth line contains 'M' single space separated integers representing the elements of the second array/list.
Output Format :
For each test case, print the required sum of the arrays/list in a row, separated by a single space.

Output for every test case will be printed in a separate line.
Constraints :
1 <= t <= 10^2
0 <= *N* <= 10^5
0 <= *M* <= 10^5
Time Limit: 1 sec
Sample Input 1:
1
3
6 2 4
3
7 5 6
Sample Output 1:
1 3 8 0
Sample Input 2:
2
3
8 5 2
2
1 3
4
9 7 6 1
3
4 5 9
Sample Output 2:
0 8 6 5
1 0 2 2 0


```java
import java.lang.Math;

public class Solution {

    public static void sumOfTwoArrays(int arr1[], int arr2[], int output[])
    {
        //Your code goes here
        int m = arr1.length-1;
        int n = arr2.length-1;
        int k = arr1.length>=arr2.length?arr1.length:arr2.length;
```

```java
        int num = 0, carry = 0, temp = 0;

        while(m>=0 && n>=0){
            temp = arr1[m--]+arr2[n--]+carry;
            num = (temp)%10;
            carry = temp/10;
            output[k--] = num;
            // System.out.println(carry+""+num);
        }
        while(n>=0){
            temp = arr2[n--]+carry;
            num = temp%10;
            carry = temp/10;
            output[k--] = num;
        }
        while(m>=0){
            temp = arr1[m--]+carry;
            num = temp%10;
            carry = temp/10;
            output[k--] = num;
        }
        while(k>=0){
            output[k--]=carry;
            carry = 0;
        }
    }

}
```