L9: Searching Sorting Practice Questions

1-Tut : 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
```

```
1.  int binarySearch(int *input, int n, int val)
2.  {
3.      //Write your code here
4.      int s = 0;
5.      int end = n-1;
6.      int mid = (s + end) / 2;
7.
```

```
8.
9.      while(s <= end){
10.
11.         if(input[mid] == val){
12.             return mid;
13.         }
14.
15.         if(input[mid] < val){
16.             s = mid + 1;
17.             mid = (s + end) / 2;
18.             continue;
19.
20.         }
21.
22.         if(input[mid] > val){
23.             end = mid -1;
24.             mid = (s + end) / 2;
25.             continue;
26.         }
27.
28.     }
29.
30.     if( s > end ){
31.
32.         return -1;
33.     }
34.
35. }
```

2-Tut : Code Bubble Sort

Send Feedback

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

```
1.  void bubbleSort(int *input, int size)
2.  {
3.      //Write your code here
4.      for(int i = 0; i < size-1; i++){
5.
6.          for(int j = 1; j < size - i; j++){
7.
8.              if(input[j] < input[j-1]){
9.                  int temp = input[j-1];
10.                 input[j-1] = input[j];
11.                     input[j] = temp;
12.             }
13.         }
14.     }
15. }
```

3-Tut : Code Insertion Sort

Send Feedback

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

```
1.  void insertionSort(int *input, int size)
2.  {
3.      //Write your code here
4.      int i , j ,key;
5.
6.      for( i = 1; i < size; i++)
7.      {
8.          key = input[i];
9.          j = i-1;
10.
11.         while(j >= 0 && key < input[j]){
12.             input[j+1] = input[j];
13.             j--;
14.
15.         }
16.         input[j+1] = key;
17.         //correct index
18.        // shifting
19.     }
20. }
```

4-Tut : Code Merge Two Sorted Arrays

Send Feedback

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
```

```
void merge(int *arr1, int size1, int *arr2, int size2, int *ans)
{
    //Write your code here
    int i = 0, j=0, k = 0;
    while(i < size1 && j < size2){
        if(arr1[i] < arr2[j]){
            ans[k] = arr1[i];
            i++;
            k++;
            continue;
        }
        if(arr1[i] >= arr2[j]){
            ans[k] = arr2[j];
            j++;
            k++;
            continue;
        }

    }

    while(i < size1){
        ans[k]= arr1[i];
        k++;
        i++;
    }

    while(j < size2){
        ans[k] = arr2[j];
        k++;
        j++;
    }
}
```

5-Ass : Push Zeros to end

Send Feedback

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

```
1.  void pushZeroesEnd(int *input, int size)
2.  {
3.      //Write your code here
4.      int i = 0 , k = 0;
5.      while(i < size){
6.
7.          if (input[i] != 0){
8.              input[k] = input[i];
9.              k++;
10.         }
11.         i++;
12.     }
13.
14.     while(k < size){
15.         input[k] = 0;
16.         k++;
17.     }
18. }
```

6-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
```

```
1.  void rotate(int *input, int d, int n)
2.  {
3.      //Write your code here
4.      int index = d;
5.
6.      int a[n];
```

```
7.      int k = 0;
8.
9.      if(index > -1){
10.
11.
12.         for(int i = index; i < n ; i++){
13.             a[k] = input[i];
14.             k++;
15.         }
16.
17.         for(int i = 0; i < index; i++){
18.             a[k]= input[i];
19.             k++;
20.         }
21.
22.         for(int i = 0 ; i < n; i++){
23.             input[i] = a[i];
24.         }
25.
26.     }
27.
28. }
```

7-Ass : 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
```

```
#include <bits/stdc++.h>
int findSecondLargest(int *input, int n)
{
    //Write your code here
    if(n <= 1){
        return INT_MIN;
    }

    long int sl = INT_MIN;
    int fl = input[0];

    int i = 1;
    while(i < n){
        if(input[i] > fl ){
            fl = input[i];
        }
        i++;
    }
     i = 0;
    while(i < n){
        int key = input[i];
        if(key > sl && key != fl){
            sl = key;
        }

        i++;
    }


    return sl;

}
```

8-Ass : Check Array Rotation

Send Feedback

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
```

```
1.  int arrayRotateCheck(int *input, int size)
2.  {
3.      //Write your code here
4.      int ans = 0;
5.
6.      for(int i = 1; i < size; i++){
7.
8.          if(input[i] < input[i-1]){
9.              ans = i;
10.             break;
11.
```

```
        }
    }

    return ans;
}
```

9-Ass : **Sort 0 1 2**

Send Feedback

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
```

```
1.  void sort012(int *arr, int n)
2.  {
3.      //Write your code here
4.      int nz = 0;
5.      int nt = n-1;
6.      int i = 0;
7.
8.      while(i <= nt){
9.
10.         if(arr[i] == 0){
11.            int temp = arr[nz];
12.             arr[nz] = 0;
13.             arr[i] = temp;
14.             nz ++;
15.             i++;
16.            continue;
17.         }
18.
19.         if(arr[i] == 1)
20.         {
21.             i++;
22.             continue;
23.         }
24.
25.
26.         if (arr[i] == 2){
27.             int temp = arr[nt];
28.             arr[nt] = 2;
29.             arr[i] = temp;
30.             nt--;
31.            continue;
32.         }
33.     }
34. }
```

10-Ass : Sum of Two Arrays

Send Feedback

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
```

```
1.  void sumOfTwoArrays(int *input1, int size1, int *input2, int size2, int *output)
2.  {
3.      //Write your code here
4.      int i = size1-1, j = size2-1;
5.      int k = 0;
6.
7.      if(size1 < size2){
8.          k = size2;
9.      }
10.     else {
11.         k = size1;
12.     }
13.
14.
15.     while(i >= 0 && j >= 0){
16.
17.         int sum = input1[i] + input2[j] + output[k];
18.
19.         if(sum <= 9){
20.             output[k] = sum;
21.             k--;
22.         }
23.         else
24.         {
25.             int carry = sum / 10;
26.             output[k] = sum % 10;
27.             output[k-1] = carry;
28.             k--;
29.
30.         }
31.         i--;
32.         j--;
33.
34.
35.     }
36.
37.      while(i >= 0){
38.
39.         int sum = input1[i] + output[k];
40.
41.         if(sum < 9){
42.             output[k] = sum;
43.             k--;
44.         }
45.         else
46.         {
47.             int carry = sum / 10;
48.             output[k] = sum % 10;
49.             output[k-1] = carry;
```

```
            k--;

        }
        i--;
    }

     while(j >= 0){

        int sum = input2[i] + output[k];

        if(sum < 9){
            output[k] = sum;
            k--;
        }
        else
        {
            int carry = sum / 10;
            output[k] = sum % 10;
            output[k-1] = carry;
            k--;

        }
        j--;
    }

}
```