Намиране на елемент

locked

Problem | Submissions | Leaderboard | Discussions

Даден ви е сортиран масив с N целочислени елемента. Ще трябва да отговаряте на много на брой заявки от вида - съдържа ли се елементът X в масива. Ако елемента се съдържа, трябва да изведете първата и последната позиция на елемента, а ако не се съдържа - на коя позиция ще застане ако се добави в сортираната последователност. **Позициите са индексирани от 0.**

Submissions: 91
Max Score: 100
Difficulty: Medium
Rate This Challenge:
More

Input Format

Първия ред съдържа числото **N** - броят на елементите в масива. На вторият ред се намират елементите на масива. На следващия се съдържа числото **Q** - броят на заявките, на които ще трябва да отговаряте. Следват **Q** реда с по едно число на ред - числото, отговарящо на текущата заявка.

Constraints

1 ≤ N ≤ 100 000

1 ≤ Q ≤ 200 000

-10^9 ≤ елементите на масива, числата от заявките ≤ 10^9

Елементите на масива и числата от заявките могат да се повтарят.

Output Format

Изведете **Q** реда със съответно едно или две числа на ред - отговорите на всички заявки.

Sample Input 0

```
6
1 1 3 5 5 7
6
1
0
3
5
6
10
```

Sample Output 0

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

Explanation 0

Отговаряме на 6 заявки - за числата 1, 0, 3, 5, 6, 10.

- 1 се среща в масива като първото му срещане е на позиция 0, а последното - на позиция 1.
- 0 не се среща в масива, ако се добави ще застане точно в началото на масива - на позиция 0.
- 3 се среща 1 път в масива, съответно първото и последното му срещане са на позиция 2.
- 5 се среща в масива, първото и последното срещане са му съответно на позиции 3 и 4
- 6 не се среща и ако се добави ще застане точно след втората петица - на позиция 5
- 10 не се среща и при добавяне ще застане накрая на масива - позиция 6.

Current Buffer (saved locally, editable) C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
void binarySearch(int* arr,int size,int target){
  int left=0;
  int right=size-1;
  bool find=false;

  while(left<=right){
    int mid=(left+right)/2;
    if(target<arr[mid]){
      right=mid-1;
    }
    else if(target>arr[mid]){
      left=mid+1;
    }
    else if(target==arr[mid]){
      find=true;
      int leftLim=mid;
      int rightLim=mid;
      while(arr[rightLim+1]==target){
        rightLim++;
      }
      while(arr[leftLim-1]==target){
        leftLim--;
      }
      cout<<leftLim<<" "<<rightLim;

      break;
    }
  }
  if(!find){
    cout<<left;
  }
}

int main() {
    int size;
    cin>>size;
  int* arr=new int[size];
  for(int i=0;i<size;i++){
    cin>>arr[i];
  }
 int queriesNumber;
  cin>>queriesNumber;
  int number;
  for(int i=0;i<queriesNumber;i++){
    cin>>number;
    binarySearch(arr,size,number);
    cout<<endl;
  }

    return 0;
}

```

Line: 1 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code