Number of occurrence

Difficulty: **Easy** Accuracy: **59.34%** Submissions: **356K+** Points: **2**

Average Time: **20m**

Given a **sorted** array, **arr[]** and a number **target**, you need to find the number of occurrences of **target** in **arr[]**.

Examples :

Input: arr[] = [1, 1, 2, 2, 2, 2, 3], target = 2
Output: 4
Explanation: target = 2 occurs 4 times in the given array so the output is 4.

Input: arr[] = [1, 1, 2, 2, 2, 2, 3], target = 4
Output: 0
Explanation: target = 4 is not present in the given array so the output is 0.

C++ (12) Start Timer

```cpp
class Solution {
  public:
    int countFreq(vector<int>& arr, int target) {
        // code here
        int n = arr.size();
        int c=0;
        for(int i=0;i<n;i++)
        {
            if(arr[i]==target)
            {
                c++;
            }
        }
        return c;
    }
};
```

Custom Input Compile & Run Su

Sorted and Rotated Minimum

Difficulty: **Easy** Accuracy: **40.57%** Submissions: **161K+** Points: **2**

A sorted array of distinct elements **arr[]** is rotated at some unknown point, the task is to find the minimum element in it.

Examples:

Input: arr[] = [5, 6, 1, 2, 3, 4]
Output: 1
Explanation: 1 is the minimum element in the array.

Input: arr[] = [3, 1, 2]
Output: 1
Explanation: Here 1 is the minimum element.

Input: arr[] = [4, 2, 3]
Output: 2
Explanation: Here 2 is the minimum element.

C++ (12) Start Timer

```cpp
class Solution {
  public:
    int findMin(vector<int>& arr) {
        // complete the function here
        sort(arr.begin(),arr.end());
        return arr[0];
    }
};
```

Search in Rotated Sorted Array

Difficulty: **Medium** Accuracy: **37.64%** Submissions: **280K+** Points: **4**

Given a **sorted** and **rotated** array **arr[]** of **distinct** elements, the task is to find the index of a target **key**. If the key is not present in the array, return **-1**.

Examples :

Input: arr[] = [5, 6, 7, 8, 9, 10, 1, 2, 3], key = 3
Output: 8
Explanation: 3 is found at index 8.

Input: arr[] = [3, 5, 1, 2], key = 6
Output: -1
Explanation: There is no element that has value 6.

Input: arr[] = [33, 42, 72, 99], key = 42
Output: 1
Explanation: 42 is found at index 1.

C++ (12) Start Timer

```cpp
class Solution {
  public:
    int search(vector<int>& arr, int key) {
        // Code Here
        int n= arr.size();
        int res=-1;
        for(int i=0;i<n;i++)
        {
            if(arr[i]==key)
            {
                res=i;
            }
        }
        return res;
    }
};
```

K-th element of two Arrays

Difficulty: **Medium** Accuracy: **37.4%** Submissions: **368K+** Points: **4**

Average Time: **15m**

Given two sorted arrays **a[]** and **b[]** and an element **k**, the task is to find the element that would be at the k^{th} position of the combined sorted array.

Examples :

Input: a[] = [2, 3, 6, 7, 9], b[] = [1, 4, 8, 10], k = 5
Output: 6
Explanation: The final combined sorted array would be [1, 2, 3, 4, 6, 7, 8, 9, 10]. The 5th element of this array is 6.

Input: a[] = [1, 4, 8, 10, 12], b[] = [5, 7, 11, 15, 17], k = 6
Output: 10
Explanation: Combined sorted array is [1, 4, 5, 7, 8, 10, 11, 12, 15, 17]. The 6th element of this array is 10.

Constraints:

C++ (12) Start Timer

```cpp
class Solution {
  public:
    int kthElement(vector<int> &a, vector<int> &b, int k) {
        // code here
        int n=a.size();
        int m= b.size();
        vector<int> temp(n+m);

        for(int i=0;i<n;i++){temp[i]=a[i];}
        for(int i=0;i<m;i++){temp[i+n]=b[i];}

        sort(temp.begin(),temp.end());
        return temp[k-1];

    }
};
```

Custom Input Compile & Run