Sum of Array

School Accuracy: 74.98% Submissions: 86K+ Points: 0

Given an integer array **Arr**[] of size **N**. The task is to find sum of it.

Example 1:

Input:
N = 4
Arr[] = {1, 2, 3, 4}
Output: 10
Explanation: 1 + 2 + 3 + 4 = 10.

Example 2:

Input:
N = 3
Arr[] = {1, 3, 3}
Output: 7
Explanation: 1 + 3 + 3 = 7.

Your Task:
Complete the function **sum()** which takes array **arr** and single integer **n**, as input parameters and returns an integer denoting the answer. You don't to print answer or take inputs.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
$1 <= N <= 10^5$
$1 <= Arr[i] <= 10^4$

```cpp
class Solution{
public:
    // function to return sum of elements
    // in an array of size n
    int sum(int arr[], int n) {
        // code here
        int ans=0;
        for(int i=0;i<n;i++)ans+=arr[i];
        return ans;
    }
};
```

Check if array is sorted

Easy Accuracy: 39.37% Submissions: 175K+ Points: 2

Given an array **arr**[] of size **N**, check if it is sorted in non-decreasing order or not.

Example 1:

Input:
N = 5
arr[] = {10, 20, 30, 40, 50}
Output: 1
Explanation: The given array is sorted.

Example 2:

Input:
N = 6
arr[] = {90, 80, 100, 70, 40, 30}
Output: 0
Explanation: The given array is not sorted.

Your Task:
You don't need to read input or print anything. Your task is to complete the function **arraySortedOrNot()** which takes the **arr**[] and N as input parameters and returns a **boolean** value (true if it is sorted otherwise false).

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
$1 \leq N \leq 10^5$
$1 \leq Arr[i] \leq 10^6$

```cpp
class Solution {
  public:
    bool arraySortedOrNot(int arr[], int n) {
        // Array has one or no element
        if (n == 0 || n == 1) return true;

        for (int i = 1; i < n; i++)
            // Unsorted pair found
            if (arr[i - 1] > arr[i]) return false;

        // No unsorted pair found
        return true;
    }
};
```

Find minimum and maximum element in an array

Basic Accuracy: 68.55% Submissions: 238K+ Points: 1

Given an array **A** of size **N** of integers. Your task is to find the **minimum and maximum** elements in the array.

Example 1:

Input:
N = 6
A[] = {3, 2, 1, 56, 10000, 167}
Output: 1 10000
Explanation: minimum and maximum elements of array are 1 and 10000.

Example 2:

Input:
N = 5
A[] = {1, 345, 234, 21, 56789}
Output: 1 56789
Explanation: minimum and maximum element of array are 1 and 56789.

Your Task:
You don't need to read input or print anything. Your task is to complete the function **getMinMax()** which takes the array **A[]** and its size **N** as inputs and returns the **minimum and maximum** element of the array.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
$1 <= N <= 10^5$
$1 <= A_i <= 10^{12}$

```cpp
pair<long long, long long> getMinMax(long long a[], int n) {
    long long mn = 1e18, mx = -1;

    // Iterating over the array
    for (int i = 0; i < n; i++) {

        // Updating the minimum value
        mn = min(a[i], mn);

        // Updating the maximum value
        mx = max(a[i], mx);
    }

    // Returning the minimum and maximum values as a pair
    return {mn, mx};
}
```

Number of occurrence

Medium Accuracy: 59.34% Submissions: 176K+ Points: 4

Given a sorted array **Arr** of size **N** and a number **X**, you need to find the number of occurrences of **X** in **Arr**.

Example 1:

Input:
N = 7, X = 2
Arr[] = {1, 1, 2, 2, 2, 2, 3}
Output: 4
Explanation: 2 occurs 4 times in the given array.

Example 2:

Input:
N = 7, X = 4
Arr[] = {1, 1, 2, 2, 2, 2, 3}
Output: 0
Explanation: 4 is not present in the given array.

Your Task:
You don't need to read input or print anything.
Your task is to complete the function **count()** which takes the array of integers **arr, n,** and **x** as parameters and returns an integer denoting the answer.
If x is not present in the array (arr) then return 0.

Expected Time Complexity: O(logN)
Expected Auxiliary Space: O(1)

Constraints:
$1 \leq N \leq 10^5$
$1 \leq Arr[i] \leq 10^6$
$1 \leq X \leq 10^6$

```cpp
class Solution{
public:
    //Function to count the number of occurrences of a given number in an array.
    int count(int arr[], int n, int x) {
        //get the index of first occurrence of x
        int *low = lower_bound(arr, arr + n, x);

        // If element is not present, return 0
        if (low == (arr + n) || *low != x)
            return 0;

        // Else get the index of last occurrence of x.
        // Note that we are only looking in the subarray after first occurrence
        int *high = upper_bound(low, arr + n, x);

        // return count
        return high - low;
    }
};
```

2798. Number of Employees Who Met the Target

Easy | Topics | Companies | Hint

There are `n` employees in a company, numbered from `0` to `n - 1`. Each employee `i` has worked for `hours[i]` hours in the company.

The company requires each employee to work for **at least** `target` hours.

You are given a **0-indexed** array of non-negative integers `hours` of length `n` and a non-negative integer `target`.

Return *the integer denoting the number of employees who worked at least* `target` *hours*.

Example 1:

```
Input: hours = [0,1,2,3,4], target = 2
Output: 3
Explanation: The company wants each employee to work for at least 2 hours.
- Employee 0 worked for 0 hours and didn't meet the target.
- Employee 1 worked for 1 hours and didn't meet the target.
- Employee 2 worked for 2 hours and met the target.
- Employee 3 worked for 3 hours and met the target.
- Employee 4 worked for 4 hours and met the target.
There are 3 employees who met the target.
```

Example 2:

```
Input: hours = [5,1,4,2,2], target = 6
Output: 0
Explanation: The company wants each employee to work for at least 6 hours.
There are 0 employees who met the target.
```

Constraints:

- `1 <= n == hours.length <= 50`
- `0 <= hours[i], target <= 10^5`

```cpp
class Solution {
public:
    int numberOfEmployeesWhoMetTarget(vector<int>& hours, int target) {
        int cnt = 0;
        for (int i = 0; i < hours.size(); i++){
            if (hours[i] >= target){cnt++;}
        }
        return cnt;
    }
};
```

1512. Number of Good Pairs

Solvec

Easy | Topics | Companies | Hint

Given an array of integers `nums`, return *the number of **good pairs***.

A pair `(i, j)` is called *good* if `nums[i] == nums[j]` and `i` < `j`.

Example 1:

```
Input: nums = [1,2,3,1,1,3]
Output: 4
Explanation: There are 4 good pairs (0,3), (0,4), (3,4), (2,5) 0-indexed.
```

Example 2:

```
Input: nums = [1,1,1,1]
Output: 6
Explanation: Each pair in the array are good.
```

Example 3:

```
Input: nums = [1,2,3]
Output: 0
```

Constraints:

- `1 <= nums.length <= 100`
- `1 <= nums[i] <= 100`

```
/*
Intuition
We can observe that every number that occurs previously that many times new pairs can be made.

Approach
Building upon the above intution while iterating once we'll store the frequency of numbers in unordered map.

keep adding corresponding number frequency to get the result.
*/
```

```cpp
class Solution {
public:
    int numIdenticalPairs(vector<int>& nums) {
        unordered_map<int , int>mp;
        int res = 0;
        for(int i : nums){
            res+=mp[i];
            mp[i]++;
        }
        return res;
    }
};
```