**Smaller Sum :-**

You are given an array **arr** of **n** integers. For each index i, you have to find the sum of all integers present in the array with a value **less** than arr[i].

**Example 1:**

**Input:**

n = 3

arr = {1, 2, 3}

**Output:**

0 1 3

**Explanation:**

For 1, there are no elements lesser than itself.

For 2, only 1 is lesser than 2.

And for 3, 1 and 2 are lesser than 3, so the sum is 3.

**Example 2:**

**Input:**

n = 2

arr = {4, 4}

**Output:**

0 0

**Explanation:**

For 4, there are no elements lesser than itself.

For 4, there are no elements lesser than itself.

There are no smaller elements than 4.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **smallerSum()** which takes an integer **n** and an array **arr** and returns an array of length **n** , the answer for every index.

**Expected Time Complexity**:O(n log n)  
**Expected Space Complexity**:O(n)  
**Constraints:**  
1 <= n <= 105  
1 <= arr[i] <= 109

**CODE :-**

//{ Driver Code Starts

//Initial Template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

//User function Template for C++

class Solution{

public:

vector<long long> smallerSum(int n,vector<int> &arr){

vector<int> help;

for(auto i:arr)

help.push\_back(i);

sort(help.begin(), help.end());

unordered\_map<int, int> freq;

for(auto i:help){

freq[i]++;

}

unordered\_map<int, long long> acu;

acu[help[0]]=0;

for(int i=1; i<help.size(); ++i){

if(help[i]!=help[i-1]){

acu[help[i]] = acu[help[i-1]] + (help[i-1] \* freq[help[i-1]]);

}

}

vector<long long> ans;

for(int i=0; i<arr.size(); ++i){

ans.push\_back(acu[arr[i]]);

}

return ans;

}

};

//{ Driver Code Starts.

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

vector<int> arr(n);

for(int i=0;i<n;i++){

cin>>arr[i];

}

Solution ob;

vector<long long> ans=ob.smallerSum(n,arr);

for(int i=0;i<n;i++){

if(i!=n-1){

cout<<ans[i]<<" ";

}

else{

cout<<ans[i]<<endl;

}

}

}

return 0;

}

// } Driver Code Ends

**Code :- T.C = O(n\*log n), S.C=O(n)**

**Rough Work Analysis :-**

20

0 7 7 9 0 0 5 1 0 6 6 5 8 9 9 9 1 2 4 3

0 0 0 0 1 1 2 3 4 5 5 6 6 7 7 8 9 9 9 9 <-help

0 0 0 0 0 0 2 4 7 11 11 21 21 33 33 47 55 55 55 55 <-acu

5

3 5 1 8 9

1 3 5 8 9 <-help

0 1 4 9 17 <-acu