**Fraction pairs with sum 1 :-**

Medium Accuracy: 54.94% Submissions: 4K+ Points: 4

Given a list of **N** fractions, represented as two lists **numerator**and **denominator**, the task is to determine the count of pairs of fractions whose sum equals 1.

**Example 1:**

**Input:**  
N = 4  
numerator = [1, 2, 2, 8]  
denominator = [2, 4, 6, 12]  
**Output:**

2

**Explanation:**Fractions 1/2 and 2/4 sum to 1. Similarly fractions 2/6 and 8/12 sum to 1. So there are 2 pairs of fractions which sum to 1.

**Example 2:**

**Input:**N = 5  
numerator = [3, 1, 12, 81, 2]  
denominator = [9, 10, 18, 90, 5]  
**Output:**

2

**Explanation:**Fractions 3/9 and 12/18 sum to 1. Similarly fractions 1/10 and 81/90 sum to 1. So there are 2 pairs of fractions which sum to 1.

**Your task:**  
You don't need to read input or print anything. Your task is to complete the function **countFractions()** which take integer **N** and arrays **numerator**and **denominator** of size N as arguments, and returns an integer.

**Expected Time Complexity:** O(N\*log(N))  
**Expected Auxiliary Space:** O(N)

**Constraints:**  
1 <= N <=105  
1 <= numerator[i] <= denominator[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++

bool cmp(long double &num1, long double &num2){

return num1<num2;

}

class Solution

{

public:

int lower(long double &item, vector<long double> &temp, int start){

int low=start, high=temp.size()-1, ans=-1;

//cout<<"#";

while(low <= high){

int mid = low + (high - low) / 2;

//cout<<"{"<<item<<","<<temp[mid]<<"}";

if(item+temp[mid] == (long double)1){

ans = mid;

high = mid - 1;

}

else if(item+temp[mid] < (long double)1)

low = mid + 1;

else

high = mid - 1;

}

//cout<<"#"<<ans;

return ans;

}

int higher(long double &item, vector<long double> &temp, int start){

int low=start, high=temp.size()-1, ans=-1;

//cout<<"$";

while(low <= high){

int mid = low + (high - low) / 2;

//cout<<"{"<<item<<","<<temp[mid]<<"}";

if(item+temp[mid] == (long double)1){

ans = mid;

low = mid + 1;

}

else if(item+temp[mid] < (long double)1)

low = mid + 1;

else

high = mid - 1;

}

//cout<<"$"<<ans;

return ans;

}

int countFractions(int n, int num[], int den[]){

vector<long double> temp(n);

for(auto i=0; i<n; i++)

temp[i] = (long double)num[i] / den[i];

sort(temp.begin(), temp.end(), cmp);

int count=0;

for(auto i=0; i<n-1; i++){

//cout<<"("<<temp[i];

int start = lower(temp[i], temp, i+1), end=-1;

if(start > 1){

end = higher(temp[i], temp, start);

//cout<<"-"<<start<<","<<end;

count += (end - start + 1);

}

//cout<<")";

}

return count;

}

};

//{ Driver Code Starts.

int main()

{

int t;

cin>>t;

while(t--)

{

int n;

cin>>n;

int numerator[n],denominator[n];

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

cin>>numerator[i];

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

cin>>denominator[i];

Solution ob;

int ans=ob.countFractions(n,numerator,denominator);

cout<<ans<<endl;

}

}

// } Driver Code Ends

**T.C :- O(N \* log N)**

**S.C :- O(N)**