The SUM problem can be formulated as follows: given four lists A, B, C, D of integer values, compute how many quadruplet (a, b, c, d ) ∈ A x B x C x D are such that a + b + c + d = 0 . In the following, we assume that all lists have the same size n .

Input

The first line of the input file contains the size of the lists n (this value can be as large as 4000). We then have n lines containing four integer values (with absolute value as large as 2 28 ) that belong respectively to A, B, C and D .

Output

For each input file, your program has to write the number quadruplets whose sum is zero.

Sample Input

6

-45 22 42 -16

-41 -27 56 30

-36 53 -37 77

-36 30 -75 -46

26 -38 -10 62

-32 -54 -6 45

Sample Output

5

Hint

Sample Explanation: Indeed, the sum of the five following quadruplets is zero: (-45, -27, 42, 30), (26, 30, -10, -46), (-32, 22, 56, -46),(-32, 30, -75, 77), (-32, -54, 56, 30).

折半枚举

紫书上叫中途相遇法，还有一个名字感觉更加妥帖一点，叫：折半枚举。          有时候，当问题的规模较大时，无法枚举所有元素的组合，但能够枚举一半的元素组合，此时，将问题拆成两半后分别枚举，再合并他们的结果这一方法往往非常有效。

两重循环加二分，总复杂度为n^2logn

这里值得一提的是对集合CD的存储方式，我是用了一个有

#include<iostream>

#include<cstring>

#include<map>

#include<algorithm>

using namespace std;

int z[4005\*4005];

int t[4005][4];

int main()

{

ios::sync\_with\_stdio(false);

int T;

cin>>T;

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

cin>>t[i][0]>>t[i][1]>>t[i][2]>>t[i][3];

int cnt=0;

int len(0);

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

for(int j=0;j<T;j++)

z[i\*T+j]=t[i][0]+t[j][1];

sort(z,z+T\*T);

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

for(int j=0;j<T;j++)

{

int temp=-(t[i][2]+t[j][3]);

cnt+=upper\_bound(z,z+T\*T,temp)-lower\_bound(z,z+T\*T,temp);//排了序之后，upperbound-lowerbound范围内的都是temp

}

cout<<cnt<<endl;

return 0;

}

另解（手写二分）

#include <stdio.h>

#include <string.h>

#include <algorithm>

#include <map>

#include<iostream>

using namespace std;

int n,a[4005],b[4005],c[4005],d[4005],sum1[16000005],sum2[16000005],len;

int main()

{

int i,j,ans,l,r,mid;

while(cin>>n)

{

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

cin>>a[i]>>b[i]>>c[i]>>d[i];

len=0;

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

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

sum1[len++]=a[i]+b[j];//sum1是a,b

len=0;

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

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

sum2[len++]=c[i]+d[j];//sum2是c,d

ans=0;

sort(sum2,sum2+len);

for(int i=0;i<len;i++)//在挨个扫sum1

{

l=0,r=len-1;

while(l<r)//二分的过程

{

mid=(l+r)>>1;

if(sum2[mid]< -sum1[i])//找sum2中最接近sum1[i]绝对值的

l=mid+1;

else

r=mid;

}

while(sum2[l]==-sum1[i] && l<len)

{

ans++;

l++;

}

}

printf("%d\n",ans);

}

return 0;

}