# **[容斥原理专辑](https://www.cnblogs.com/nefu929831238/p/6042645.html)**

## **对容斥原理的描述**

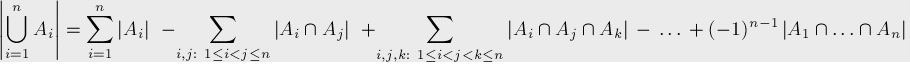
容斥原理是一种重要的组合数学方法，可以让你求解任意大小的集合，或者计算复合事件的概率。

### **描述**

       容斥原理可以描述如下：

         要计算几个集合并集的大小，我们要先将所有****单个集合****的大小计算出来，然后减去所有****两个集合相交****的部分，再加回所有****三个集合相交****的部分，再减去所有****四个集合相交****的部分，依此类推，一直计算到****所有集合相交****的部分。

### **关于集合的原理公式**

      上述描述的公式形式可以表示如下：********

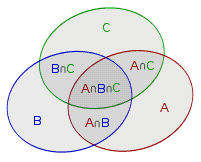
      它可以写得更简洁一些，我们将B作为所有Ai的集合，那么容斥原理就变成了：

IMG_257

         这个公式是由 De Moivre (Abraham de Moivre)提出的。

### **关于维恩图的原理**

       用维恩图来表示集合A、B和C：



         那么IMG_259的面积就是集合A、B、C各自面积之和减去IMG_260 , IMG_261, IMG_262 的面积，再加上IMG_263的面积。

IMG_264

### **1、hdu 1796  How many integers can you find**

容斥典型问题：求指定区间内与n互素的数的个数

题目大意：给定n和一个大小为m的集合，集合元素为非负整数。为1...n内能被集合里任意一个数整除的数字个数。n<=2^31,m<=10

**Sample Input**

12 2

2 3

**Sample Output**

7

解题思路：容斥原理地简单应用。先找出1...n内能被集合中任意一个元素整除的个数，再减去能被集合中任意两个整除的个数，即能被它们两只的最小公倍数整除的个数，因为这部分被计算了两次，然后又加上三个时候的个数，然后又减去四个时候的倍数...所以深搜，最后判断下集合元素的个数为奇还是偶，奇加偶减。

#include<iostream>

#include<cstdio>

#include<cstring>

#include<vector>

using namespace std;

int n,m,cnt;

long long ans,a[30];

long long gcd(long long a,long long b){

return b==0?a:gcd(b,a%b);

}

void DFS(int cur,long long lcm,int id){

lcm=a[cur]/gcd(a[cur],lcm)\*lcm;

if(id&1)

ans+=(n-1)/lcm; //因为这题并不包含n本身，所以用n-1

else

ans-=(n-1)/lcm;

for(int i=cur+1;i<cnt;i++)

DFS(i,lcm,id+1);

}

int main(){

//freopen("input.txt","r",stdin);

while(~scanf("%d%d",&n,&m)){

cnt=0;

int x;

while(m--){

scanf("%d",&x);

if(x!=0) //除0

a[cnt++]=x;

}

ans=0;

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

DFS(i,a[i],1);

cout<<ans<<endl;

}

return 0;

}

#include <iostream>

#define ll long long

using namespace std;

ll gcd(ll a,ll b){

if(b==0) return a;

else return gcd(b,a%b);

}

ll lcm(ll a,ll b){

return a\*b/gcd(a,b);

}

int main(){

int n,m,k;

ll data[15],a[15];

while(cin>>n>>m){

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

cin>>data[i];

k=0;

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

if(data[i])

a[k++]=data[i];

}

ll ans=0;

for(int i=1;i<(1<<k);i++){

int cnt=0;

ll mul=1;

for(int j=0;j<k;j++){

if(i&(1<<j)){

cnt++;

mul=lcm(mul,a[j]);

}

}

if(cnt&1) ans+=(n-1)/mul;

else ans-=(n-1)/mul;

}

cout<<ans<<endl;

}

return 0;

}

### **2、hdu 4135 Co-prime**

容斥典型问题：求指定区间内与n互素的数的个数

题意：给定a、b、c，求a到b区间内与c互质的数。

**Sample Input**

2

1 10 2

3 15 5

**Sample Output**

Case #1: 5

Case #2: 10

思路：通常我们求１～ｎ中与ｎ互质的数的个数都是用欧拉函数！ 但如果ｎ比较大或者是求１～ｍ中与ｎ互质的数的个数等等问题， 要想时间效率高的话还是用容斥原理！

容斥、先对n分解质因数，分别记录每个质因数， 那么所求区间内与某个质因数不互质的个数就是n / r(i)，假设r(i)是r的某个质因子 假设只有三个质因子， 总的不互质的个数应该为p1+p2+p3-p1\*p2-p1\*p3-p2\*p3+p1\*p2\*p3, 及容斥原理，可以转向百度百科查看相关内容 pi代表n/r(i),即与某个质因子不互质的数的个数 ，当有更多个质因子的时候， 可以用状态压缩解决，二进制位上是1表示这个质因子被取进去了。 如果有奇数个1，就相加，反之则相减

#include<iostream>

#include<cstdio>

#include<cstring>

#include<vector>

using namespace std;

long long a,b,n;

vector<long long> vt;

long long solve(long long x,long long n){

vt.clear();

long long i,j;

for(i=2;i\*i<=n;i++) //对n进行素数分解

if(n%i==0){

vt.push\_back(i);

while(n%i==0)

n/=i;

}

if(n>1)

vt.push\_back(n);

long long sum=0,val,cnt;

for(i=1;i<(1<<vt.size());i++){ //用二进制来1,0来表示第几个素因子是否被用到,如m=3，//三个因子是2,3,5，则i=3时二进制是011，表示第2、3个因子被用到

val=1;

cnt=0;

for(j=0;j<vt.size();j++)

if(i&(1<<j)){ //判断第几个因子目前被用到

val\*=vt[j];

cnt++;

}

if(cnt&1) //容斥原理，奇加偶减

sum+=x/val;

else

sum-=x/val;

}

return x-sum;

}

int main(){

//freopen("input.txt","r",stdin);

int t,cases=0;

scanf("%d",&t);

while(t--){

cin>>a>>b>>n;

cout<<"Case #"<<++cases<<": "<<solve(b,n)-solve(a-1,n)<<endl;

}

return 0;

}

容斥典型问题：求指定区间内与n互素的数的个数

方法：首先利用逆向思维，求出与n不互质的个数

考虑n的所有素因子pi(i=1…k)

在[l,r]中有多少数能被pi整除呢？它就是：

r/pi-(l-1)/pi

由于有些数可能被统计多次（被好几个素因子整除）。所以，我们要运用容斥原理来解决。

我们可以用2^k的算法求出所有的pi组合，然后计算每种组合的pi乘积，通过容斥原理来对结果进行加减处理。

结果返回 （l-r+1)-个数

#include <bits/stdc++.h>#define ll long long

using namespace std;

ll solve(ll a,ll b,ll n)

{

vector <ll> p;

for(int i=2;i\*i<=n;i++)

{

if(n%i==0)

{

p.push\_back(i);

while(n%i==0)

n=n/i;

}

}

if(n>1) p.push\_back(n);

ll sum=0;

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

{

ll cnt=0;

ll mult=1;

for(int j=0;j<p.size();j++)

{

if(i&(1<<j))

{

cnt++;

mult\*=p[j];

}

}

if(cnt&1) sum+=b/mult-(a-1)/mult;

else sum-=b/mult-(a-1)/mult;

}

return (b-a+1)-sum;

}

int main()

{

int t,i=0;

ll a,b,n;

cin>>t;

while(t--)

{

cin>>a>>b>>n;

i++;

cout<<"Case #"<<i<<": ";

cout<<solve(a,b,n)<<endl;

}

return 0;

}

### **3、hdu 1695 GCD**

容斥典型问题：求指定区间内与n互素的数的个数

# HDU 1695 GCD 【数论，容斥原理】

题目大意：

给你5个整数a、b、c、d、k，在区间[a，b]中选一个数x，在区间[c，d]中选一个数y，使得x和y的公约数为k，即gcd(x，y) = k。现在问题来了：这样的整数对共有多少对。

**Sample Input**

2

1 3 1 5 1

1 11014 1 14409 9

**Sample Output**

Case 1: 9

Case 2: 736427

求（1，b）区间和（1，d）区间里面gcd(x, y) = k的数的对数（1<=x<=b , 1<= y <= d）。

b和d分别除以k之后的区间里面，只需要求gcd(x, y) = 1就可以了，这样子求出的数的对数不变。

这道题目还要求1-3 和 3-1 这种情况算成一种，因此只需要限制x<y就可以了

只需要枚举x，然后确定另一个区间里面有多少个y就可以了。因此问题转化成为区间（1， d）里面与x互素的数的个数。

先求出x的所有质因数，因此（1，d）区间里面是x的质因数倍数的数都不会与x互素，因此，只需要求出这些数的个数，减掉就可以了。

如果w是x的素因子，则（1，d）中是w倍数的数共有d/w个。

容斥原理：

所有不与x互素的数的个数= 1个因子倍数的个数 - 2个因子乘积的倍数的个数 + 3个……-……

答案很大，用long long。

所有数的素因子，预先处理保存一下，不然会超时的。

#include <cstdio>

#include <cstring>

#include <vector>

#include <algorithm>

using namespace std;

#define N 100005

typedef long long ll;

vector<int> x[N];

bool is[N];

void prime() {

memset(is, false, sizeof(is));

for (int i=0; i<N; i++) x[i].clear();

for (int j=2; j<N; j+=2) x[j].push\_back(2);

for (int i=3; i<N; i+=2)

if (!is[i]) {

for (int j=i; j<N; j+=i) {

is[j] = true;

x[j].push\_back(i);

}

}

}

int work(int u, int s, int w) {

int cnt = 0, v = 1;

for (int i=0; i<x[w].size(); i++) {

if ((1<<i) & s) {

cnt++;

v \*= x[w][i];

}

}

int all = u/v;

if (cnt % 2 == 0) return -all;

else return all;

}

int main() {

prime();

int T, a, b, c, d, k;

scanf("%d", &T);

for (int cas=1; cas<=T; cas++) {

scanf("%d%d%d%d%d", &a, &b, &c, &d, &k);

if (k == 0) {

printf("Case %d: 0\n", cas);

continue;

}

b /= k, d /= k;

if (b > d) { a = b; b = d; d = a; }

long long ans = 0;

for (int i=1; i<=d; i++) {

k = min(i, b);

ans += k;

for (int j=1; j<(1<<x[i].size()); j++)

ans -= work(k, j, i);

}

printf("Case %d: %I64d\n", cas, ans);

}

return 0;

}

### **4、hdu 2841 Visible Trees**

容斥典型问题：求指定区间内与n互素的数的个数

题目：大概就是有个m\*n个点的矩形从(1,1)到(m,n)，问从(0,0)出发直线看过去最多能看到几个点。

**Sample Input**

2

1 1

2 3

**Sample Output**

1

5

题意：你站在（0,0）点给你一个从（1,1）开始放置的 m\*n 的点阵图，问你能看到几个点（与自己位置相连，在一条直线上的点只能看到离自己最近的那一个）。

其实就是求点（x,y）中 x与y 互质的点的个数

所以只需要会求 1-m 中与一个数n互质的数的个数即可，我们可以通过求 1-m 中与n不互质的数的个数，再让总数减去这个数求得

这时就用到了容斥原理，首先要处理出n的所有质因子{p0,p1,p2,...,pn}。

而[1,X]能被pi整除的数有⌊Xpi⌋个，再用二进制容斥处理即可（奇数次加上，偶数次减去）

#include <bits/stdc++.h>

using namespace std;

int get\_tmp(int m,int n)

{

int cnt=0,prime[7];

for(int i=2;i\*i<=n;i++)

{

if(n%i) continue;

while(n%i==0) n/=i;

prime[cnt++]=i;

}

if(n!=1)

{

prime[cnt++]=n;

}

int res=0;

for(int i=1;i<(1<<cnt);i++)

{

int tmp=1,k=0;

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

{

if(i&(1<<j))

{

tmp\*=prime[j];

k++;

}

}

if(k&1)

{

res+=m/tmp;

}

else

{

res-=m/tmp;

}

}

return m-res;

}

int main()

{

int t;

cin >> t;

while(t--)

{

int m,n;

cin >> m >> n;

long long ans=0;

for(int i=1;i<=n;i++)

{

ans+=get\_tmp(m,i);

}

cout << ans << endl;

}

return 0;

}

### **5、hdu 3501 Calculation 2**

容斥典型问题：求指定区间内与n互素的数的个数

题意：求解1~n与n不互质的数的和。

****Sample Input****

3

4

0

****Sample Output****

0

2

方法：该问题是求小于n且与n不互质数的和为多少

        我们可以先求出n的素因子， p1 p2 p3 …… pn

        那么对于 p1 在小于 n 的数的范围内有

        1\*p1  2\*p1  3\*p1  …… x\*p1 与n不互质

        再根据等差数列的性质计算出它们的和，再利用容斥原理计算

#include <iostream>

#include <vector>#define ll long long

using namespace std;

const ll mod=1000000007;

int main()

{

ll n;

while(cin>>n&&n)

{

if(n==1)

{

cout<<"0"<<endl;

continue;

}

int a=n;

vector <ll> p;

for(int i=2;i\*i<=a;i++)

{

if(a%i==0)

{

p.push\_back(i);

while(a%i==0)

a=a/i;

}

}

if(a>1) p.push\_back(a);

ll ans=0;

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

{

int cnt=0;

int mul=1;

for(int j=0;j<p.size();j++)

{

if(i&(1<<j))

{

cnt++;

mul=mul\*p[j];

}

}

ll num=(n-1)/mul;

ll tmp=(mul+mul\*num)\*num/2;

if(cnt&1) ans+=tmp;

else ans-=tmp;

}

cout<<ans%mod<<endl;

}

return 0;

}

/\*

\*Time: 0 ms

\*题目大意：

\* 求1~n里面比n小，但是与n不互素的数的总和。

\*解题思路：

\* 利用欧拉函数即可求解，1~n比n小且与n互素的数的总和为

\* sum(n) = n \* phi(n) / 2;那么可以先求出1~n-1的总和，然后

\* 减去sum(n)即可。

\*/

#include <iostream>

#include <cmath>

using namespace std;

int jisuan(int x)

{

int i,res=x;

for(i=2;i<(int)sqrt(x\*1.0)+1;i++)

if(x%i==0)

{

res=res/i\*(i-1);

while(x%i==0)

x/=i;

}

if(x>1)

res=res/x\*(x-1);

return res;

}

int main(void)

{

unsigned \_\_int64 n;

while(scanf("%I64d", &n), n)

{

unsigned \_\_int64 sum = n \* (n + 1) / 2 - n, res;

res = sum - (n \* jisuan(n) / 2);

printf("%I64d\n", res % 1000000007);

}

return 0;

}

### **6、UVA 11806 **Cheerleaders****

# **[UVA11806【拉拉队】Cheerleaders-------2015年1月24日](https://www.cnblogs.com/khbcsu/p/4245943.html)**

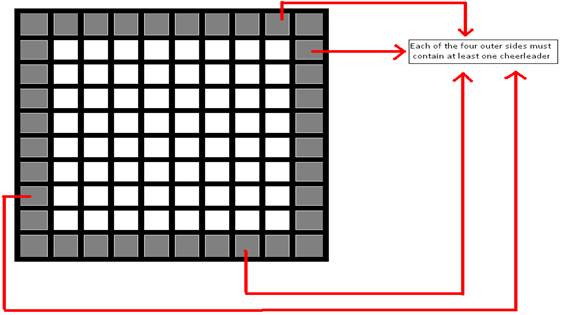
1.题意描述

本题大致意思是讲：给定一个广场，把它分为M行N列的正方形小框。现在给定有K个拉拉队员，每一个拉拉队员需要站在小框内进行表演。但是表演过程中有如下要求：

（1）每一个小框只能站立一个拉拉队员；

（2）广场的第一行，最后一行，第一列，最后一列都至少站有一个拉拉队员；

（3）站在广场的四个角落的拉拉队员可以认为是同时占据了一行和一列。



Sample Input   
2   
2 2 1   
2 3 2   
Sample Output   
Case 1: 0   
Case 2: 2

2.思路分析：

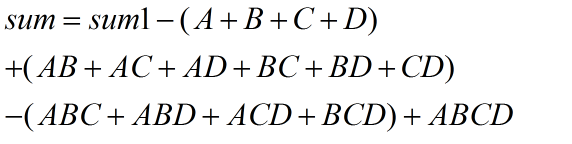
本题如果直接枚举的话难度很大并且会无从下手。那么我们是否可以采取逆向思考的方法来解决问题呢？我们可以用总的情况把不符合要求的减掉就行了。

首先我们如果不考虑任何约束条件，我们可以得出如下结论：

IMG_258

下面我们假定第一行不站拉拉队员的所有的站立方法有A种。最后一行不站拉拉队员的所有的方法有B种。第一列不站拉拉队员的所有的站立方法有C种。最后一列不站拉拉队员的站立方法有D种。

下面我们可以得出最后结果：



下面问题来了我们如何利用代码实现容斥原理呢？我们可以借用离散数学的最大项和最小项知识结合与运算来判断每一项的特征。比如说，含A的和1进行与运算。含B的与2进行与运算。含C的和4进行与运算。含D的和8进行与运算。

然后对于每一种状态，我们利用数字0-15来代替。

在进行这些工作之前，我们还要进行基础性工作，数据初始化和打表。

对于如何打表，我们可以采取组合数公式的递推式进行。打表过程中一定要注意边界问题的处理，要不极容易出错。

3.AC代码

#include<iostream>

#include<cstdio>

#include<algorithm>//注意头文件的使用

#include<string.h>

using namespace std;

#define maxn 500+5

const int mod=1000007;

int c[maxn][maxn];//数组c[i][j]表示在i个可用的点中抽取j个点的情况总数

void process()//函数预处理过程相当于打表这样节省时间 {

memset(c,0,sizeof(c));

c[0][0]=1;

for(int i=1;i<maxn;i++)

{

c[i][0]=c[i][i]=1;//初始化边界

for(int j=1;j<i;j++)//注意是j<i不能超多这是由于C[i][j]的定义得来的

c[i][j]=(c[i-1][j-1]+c[i-1][j])%mod;//组合数公式的递推式 }

}

int main(){

process();

int T;

cin>>T;

for(int Case=1;Case<=T;Case++)

{

int sum=0;

int n,m,k;

cin>>n>>m>>k;

for(int s=0;s<16;s++)//处理容斥原理的方式，与运算 {

//当s=0时就相当于我们在分析过程中的sum1

int r=n,c1=m;int b=0;//注意理解其中的b

if(s&1) {r--;b++;}

if(s&2){r--;b++;}

if(s&4){c1--;b++;}

if(s&8){c1--;b++;}

if(b&1){sum=(sum+mod-c[r\*c1][k])%mod;}//说明含有奇数个项如ABD，A等

else sum=(sum+c[r\*c1][k])%mod;//说明含有偶数个项如ABCD,AB等 }

printf("Case %d: ",Case);

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

}

return 0;

}

<https://blog.csdn.net/a273868471/article/details/19713589>

## **2005: [Noi2010]能量采集**

Time Limit: 10 Sec  Memory Limit: 552 MB  
Submit: 4379  Solved: 2615  
[[Submit](http://www.lydsy.com/JudgeOnline/submitpage.php?id=2005)][[Status](http://www.lydsy.com/JudgeOnline/problemstatus.php?id=2005)][[Discuss](http://www.lydsy.com/JudgeOnline/bbs.php?id=2005)]

## **Description**

栋栋有一块长方形的地，他在地上种了一种能量植物，这种植物可以采集太阳光的能量。在这些植物采集能量后，栋栋再使用一个能量汇集机器把这些植物采集到的能量汇集到一起。 栋栋的植物种得非常整齐，一共有n列，每列有m棵，植物的横竖间距都一样，因此对于每一棵植物，栋栋可以用一个坐标(x, y)来表示，其中x的范围是1至n，表示是在第x列，y的范围是1至m，表示是在第x列的第y棵。 由于能量汇集机器较大，不便移动，栋栋将它放在了一个角上，坐标正好是(0, 0)。 能量汇集机器在汇集的过程中有一定的能量损失。如果一棵植物与能量汇集机器连接而成的线段上有k棵植物，则能量的损失为2k + 1。例如，当能量汇集机器收集坐标为(2, 4)的植物时，由于连接线段上存在一棵植物(1, 2)，会产生3的能量损失。注意，如果一棵植物与能量汇集机器连接的线段上没有植物，则能量损失为1。现在要计算总的能量损失。 下面给出了一个能量采集的例子，其中n = 5，m = 4，一共有20棵植物，在每棵植物上标明了能量汇集机器收集它的能量时产生的能量损失。 在这个例子中，总共产生了36的能量损失。

## **Input**

仅包含一行，为两个整数n和m。

## **Output**

仅包含一个整数，表示总共产生的能量损失。

## **Sample Input**

【样例输入1】  
5 4  
【样例输入2】  
3 4

## **Sample Output**

【样例输出1】  
36  
【样例输出2】  
20  
对于100%的数据：1 ≤ n, m ≤ 100,000。

## **HINT**

## **Source**

代码：

//会发现(x,y)到(0,0)路上有gcd(x,y)-1棵植物，即求sum(2\*gcd(i,j)-1),(1<=i<=n,1<=j<=m),然后枚举每一个gcd算

//贡献度，n以内以i为因子的数有(n/i)个，还要减去以i的倍数作为因子的数的个数就不会算重了。这到这算记录一下

//就行了

#include<iostream>

#include<cstdio>

#include<cstring>using namespace std;

typedef long long ll;

ll n,m,f[100009];

int main()

{

memset(f,0,sizeof(f));

scanf("%lld%lld",&n,&m);

ll ans=0;

if(n>m) swap(n,m);

for(int i=n;i>=1;i--){

f[i]=(n/i)\*(m/i);

for(int j=i\*2;j<=n;j+=i)

f[i]-=f[j];

ans+=f[i]\*(i+i-1);

}

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

return 0;

}

# 【BZOJ1042】硬币购物（动态规划，容斥原理）

## **题面**

[BZOJ](http://www.lydsy.com/JudgeOnline/problem.php?id=1042" \t "https://blog.csdn.net/qq_30974369/article/details/_blank)

### **Description**

[硬币](https://www.baidu.com/s?wd=%E7%A1%AC%E5%B8%81&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/qq_30974369/article/details/_blank)购物一共有4种硬币。面值分别为c1,c2,c3,c4。某人去商店买东西，去了tot次。每次带di枚ci硬币，买si的价值的东西。请问每次有多少种付款方法。

### **Input**

　　第一行 c1,c2,c3,c4,tot 下面tot行 d1,d2,d3,d4,s,其中di,s<=100000,tot<=1000

### **Output**

　　每次的方法数

### **Sample Input**

1 2 5 10 2

3 2 3 1 10

1000 2 2 2 900

### **Sample Output**

4

27

## **题解**

[真题](https://www.baidu.com/s?wd=%E7%9C%9F%E9%A2%98&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/qq_30974369/article/details/_blank)真好啊。

先不考虑任何有关于硬币个数的限制   
设f[i]表示没有任何限制的情况下，价格为n的方案数   
直接做一个背包就行了。

现在加上限制来看，我们用总方案减去不合法。   
总方案是f[n]，不合法呢？   
某一个硬币如果不合法，那么它就要用d+1个   
剩下的随便选，也就是f[n−c∗(d+1)]   
这样直接容斥计算即可。

首先，计算出购买面值为i的物品的方案数f(i),这一步强制有序就可以了。

然后，每一次查询时ans=(f(s)-d1溢出方案-d2…+d1d2+d2d3+…-d1d2d3….+d1d2d3d4)，即容斥原理。

注意到d1溢出时，至少使用了d1+1个物品，于是剩下S-(d1+1)c1都可以随意分配，于是d1溢出的方案就是f(S-(d1+1)c1)

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<cmath>

#include<algorithm>

#include<set>

#include<map>

#include<vector>

#include<queue>

using namespace std;

#define ll long long

#define RG register

inline int read()

{

RG int x=0,t=1;RG char ch=getchar();

while((ch<'0'||ch>'9')&&ch!='-')ch=getchar();

if(ch=='-')t=-1,ch=getchar();

while(ch<='9'&&ch>='0')x=x\*10+ch-48,ch=getchar();

return x\*t;

}

int c[4],d[4],S;

ll f[111111];

int main()

{

for(int i=0;i<4;++i)c[i]=read();

f[0]=1;

for(int k=0;k<4;++k)

for(int j=c[k];j<=100000;++j)

f[j]+=f[j-c[k]];

int Q=read();

while(Q--)

{

for(int i=0;i<4;++i)d[i]=read();S=read();

ll ss,ans=0;

for(int i=0,tt;i<16;++i)

{

ss=tt=0;

for(int j=0;j<4;++j)

if(i&(1<<j))++tt,ss+=(d[j]+1)\*c[j];

if(ss>S)continue;

(tt&1)?ans-=f[S-ss]:ans+=f[S-ss];

}

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

}

return 0;

}