Hello World ! ---数论

/\*

数论 本原勾股数组

题意：求解N以内有多少个本原勾股数组

本原勾股数组 : 满足a^2 + b^2 = c^2且a,b,c的最大公约数为1

思路：根据下面的叙述实现，枚举s,t,判断一下

看下面的证明有两点疑惑：

(1)证明a、b奇偶性不同；

(2)证c-b与c+b互质？

解惑：http://hi.baidu.com/lvhuyh/item/5e9aecce861f0a130ad93a9b

摘: http://blog.csdn.net/magicnumber/article/details/6410043

本章主要讨论的是勾股数组，也就是关于满足a^2+b^2=c^2的三元组（a,b,c）的问题。

其实，对于勾股数组的个数进行讨论并没有多大意义，因为已知a，b，c为勾股数组，那么显然有da，db，dc(d>0)也为勾股数组，因为(da)^2+(db)^2=d^2(a^+b^2)=d^2c^2=(dc)^2。

因此着重研究的是关于本原勾股数组的问题，本原勾股数组即为a，b，c为勾股数组且满足a，b，c的最大公约数为1。

对于本原勾股数组，显然a和b奇偶性不同只需要将a=2x+1,b=2y+1,c=2z代入a^2+b^2=c^2即可推出有公约数2。由于a和b奇偶性不同，那么显然c为奇数。

那么最关心的是如何求出所有的本原勾股数组。

如果将公式转化一下，得到a^2=c^2-b^2=(c+b)(c-b)，那么显然有，c+b和c-b没有公因子，用反证法证明：

如果d|(c+b)且d|(c-b)，那么显然有d|((c+b)+(c-b))和d|((c+b)-(c-b))即d|2b且d|2c，因为在定义本原勾股数组的时候已经有了b和c的最大公约数是1的约定(虽然定义是a，b，c最大公约数是1，但是如果gcd(b,c)=d>1，显然有d|c不满足定义)。所以d要么为1，要么为2。但是如果d=2时，那么显然a，b，c均为偶数，不满足定义，那么d只能为1，证明了c+b和c-b没有公因子。

因此，如果将a^2进行质因数分解，那么会有a=p1^t1\*p2^t2\*p3^t3...pn^tn，其中指数t1,t2,t3...tn为偶数（因为这样才能保证a^2开根号后为整数），又因为c+b和c-b没有公因数，所以c+b和c-b各取a分解后的其中某一些pk^tk，因此，c+b和c-b均为平方数。那么假设c+b=s^2,c-b=t^2，则有c=(s^2+t^2)/2,b=(s^2-t^2)/2，a=st，因此形如(st,(s^2-t^2)/2,(s^2+t^2)/2)的三元组为本原勾股数组。(其中s和t都是奇数，因为如果s和t中有且只有一个为奇数，那么显然(s^2+t^2)/2不会是整数，而如果两个数都是偶数，那么显然该三元组有公因子2，与本原勾股数组定义矛盾。)

\*/

bool vis[MAXN];

bool check(int a, int b, int c)

{

return a\*a+b\*b==c\*c && gcd(c, gcd(a, b)) == 1;

}

int main()

{

while(scanf("%d", &n) != EOF) {

int p = 0, ans = 0, a, b, c;

memset(vis, false, sizeof(bool)\*(n+5));

for(int oo = 2\*(int)(sqrt(n+0.5)), s = 1; s <= oo; s += 2)

for(int t = 1; t < s; t += 2) {

int a = s\*t, b = (s\*s-t\*t)/2, c = (s\*s+t\*t)/2;

if(c > n) break;

// if(gcd(s, t) != 1) continue;

if(check(a,b,c)) {

ans++; int k = 1;

while(c\*k <= n) {

vis[a\*k] = vis[b\*k] = vis[c\*k] = true;

k++;

}

}

}

for(int i = 1; i <= n; i++) p += !vis[i];

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

}

return 0;

}

/\*

数论 分解质因子 Minimum Sum LCM 好题

题意：给出最小公倍数n，要求找出至少由两个数组成的数集，该集合的LCM等于n，且该数集的总和最小

思路：一开始想到了xxx不等式，分析觉得要找的数集大小肯定为2，就想找出符合a\*b = n, a与b互质的最小a+b...然后就堕落进了万物罪恶之源...一开始的思路和推断都是不正确的...

正解是分解质因子。将最小公倍数n分解成n = a1^p1 \* a2^p2 \* ... \* ak^pk，发现LCM(a1^p1, a2^p2, ..., ak^pk) == n，易知我们找的数集里面肯定两两互质，不然不满足总和最小。所以算法就很明显了，要“两两互质，又总和最小”，那就是各个质因子的相应次方数之和

特殊地：n = 1时，ans = 2

n为素数，ans = n + 1

n只有单质因子，ans = n + 1

n = 2147483647，它是素数，此时输出2147483648，但超了int，应考虑用long long

实现里面分解因子时，每改变一次n的大小，可重新计算一遍sqrt(n)，减少冗余计算量

\*/

ll solve(ll n)

{

ll flag = 0, sum = 0;

for(ll oo = (ll)sqrt((double) n), i = 2; i <= oo; i++) if(n % i == 0) {

ll tmp = 1;

while(n % i == 0) {

//sum += i;

tmp \*= i;

n /= i;

}

sum += tmp;

flag++;

oo = (ll)sqrt((double) n); //加这句快了一倍

}

if(n > 1) sum += n, flag ++;

if(flag == 1) sum++;

return sum;

}

/\*

组合数学 多项式定理

(n, k1+k2+...+km) = n!/(k1!k2!...km!)

m = 2时就是二项式定理

\*/

/\*

概率论 三门问题 好题 （Monty Hall problem）亦称为蒙提霍尔问题、蒙特霍问题或蒙提霍尔悖论

题意：参赛者会看见三扇关闭了的门，其中一扇的后面有一辆汽车，选中后面有车的那扇门可赢得该汽车，另外两扇门后面则各藏有一只山羊。当参赛者选定了一扇门，但未去开启它的时候，节目主持人开启剩下两扇门的其中一扇，露出其中一只山羊。主持人其后会问参赛者要不要换另一扇仍然关上的门。问题是：换另一扇门会否增加参赛者赢得汽车的概率？

现在题目变成了：给你NCOWS只牛，NCARS辆车，NSHOW扇开启的门（ NSHOW < NCOWS） 问：换门后赢得车的概率是？

思路：题目明确要求“是换门后赢车”的概率，不是“赢车”的概率

分类讨论，设A = {最开始选的那扇门是山羊}，B = {第二次选的门后是车}

由全概率公式得 P(B) = P(A)P(B|A) + P(A')P(B|A') , A'表A的补

http://blog.csdn.net/synapse7/article/details/9771107

\*/

int main()

{

double cow, car, show;

while(scanf("%lf %lf %lf", &cow, &car, &show) != EOF) {

double N = cow + car;

double ans = (N - 1) \* car / (N \* (N - show - 1));

printf("%.5f\n", ans);

}

return 0;

}

/\*

概率论 分组背包 母函数？

题意：扔N颗骰子，问总和不少于x的概率

思路：这不是赤果果的分组背包么 怎么会在数论场？难道说还能用母函数做？or else ？

就是“每组必须取一个且仅取一个”的分组背包，hdu\_3303是“每组至少取一个”的分组背包

\*/

int main()

{

int n, x;

while(scanf("%d %d", &n, &x), n || x) {

int cur = 1;

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

f[0][0] = 1;

for(int i = 1; i <= n; i++) { //group

for(int j = 0; j <= 6\*n; j++) //

for(int k = 1; k <= 6; k++) //items

f[cur][j+k] += f[1-cur][j];

cur = 1 - cur;

memset(f[cur], 0, sizeof(f[cur]));

}

ll ans = 0;

for(int i = x; i <= 6\*n; i++) ans += f[1-cur][i];

ll t = 1, d = 1;

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

t \*= 6;

if(( d = GCD(t, ans) ) != 1) t /= d, ans /= d;

}

if(ans == 0) puts("0");

else if(t == 1 && ans == 1) puts("1");

else printf("%lld/%lld\n", ans, t);

}

return 0;

}

/\*

概率论 实现题

题意：给出n\*n的n个人相互比赛的胜率，来log(n, 2)场淘汰赛，问最后i夺冠的概率。

思路：感觉就是靠实现的=\_=

我是对于每一轮，计算每个人能在该轮出线的概率，则夺冠概率就是在第4轮出线的概率，初始概率都为1

然后就说实现吧，我用set来搞，每一轮结束我都合并一次set

某轮比赛set[i]只和set[i+1](奇)或set[i-1](偶)有关

个人感觉实现还是比较漂亮的~~

\*/

set<int> Set[n+1];

int main()

{

for(int i = 1; i <= n; i++) scanf("%s", name[i]);

for(int i = 1; i <= n; i++) for(int j = 1; j <= n; j++) scanf("%lf", &p[i][j]), p[i][j] \*= 0.01;

for(int i = 1; i <= n; i++) Set[i].insert(i);

for(int i = 1; i <= n; i++) f[0][i] = 1;

int top = 16;

for(int step = 1; step <= 4; step++) {

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

int j = (i & 1) ? i + 1 : i - 1; //i,j两个set里的人物PK

for(set<int> :: iterator u = Set[i].begin(); u != Set[i].end(); u++) {

double sum = 0;

for(set<int> :: iterator v = Set[j].begin(); v != Set[j].end(); v++) {

sum += f[step-1][\*v] \* p[\*u][\*v];

}

f[step][\*u] = f[step-1][\*u] \* sum;

}

}

for(int i = 1; i <= top; i ++) { //合并set，有个if是因为有i=1时

if((i-1)/2+1 != i) Set[(i-1)/2+1].insert(Set[i].begin(), Set[i].end()), Set[i].clear();

}

top /= 2; //合并之后set个数少一半

}

for(int i = 1; i <= n; i++) printf("%-10s p=%.2f\%\n", name[i], f[4][i]\*100);

return 0;

}

/\*

循环冗余校验 Cyclic reduandancy check(CRC)

题意：给一字符串，求CRC。 题意挺难读懂的，我是看题解的。

思路：我只想记录一下实现上的问题：

"

我们一般写程序和编译的机器环境（包括OJ系统运行的环境）都是x86架构的，也就意味着字节序是little-endian， 即在存储器中的所有整型数值都是高位在前低位在后。比如32位16进制数：AF045Bh，在内存中的顺序应该是：

5B 04 AF 00

如果我们直接将字符串的指针转为int型指针进行计算就会算错。必须先对每一组字节（整型变量大小的一组）先进行反转再作运算。

"

一开始我是直接反转到unsigned int数组里面，但这样做是错的，错因归于我还是不理解little-endian。必须得先反转到char数组，然后再转为int，反转可以直接reverse\_copy。还要注意要copy过去以后，因为是char->unsigned int，字节由1->4，我们要把末尾后一位赋为0，直接a[len] = 0有可能会错，还是那个问题，a是char\*，字长1字节，我们计算的时候是用unsigned int的4字节扫，所以得应该要\*((unsigned int\*)(&a[len])) = 0，或者用我的笨办法，直接把后面几位都赋为0，更安全memset(a+len, 0, sizeof(unsigned int))

要求输出两位16进制，直接printf("%02X");

\*/

//uva 128

char a[2048];

int main()

{

while(getline( cin, str ), str != "#") {

int len = str.length();

reverse\_copy(str.begin(), str.end(), a); //NB

//a[len] = '\0';

//\*((unsigned int \*)(&a[len])) = 0;

memset(a+len, 0, sizeof(unsigned int));

unsigned int \* p = (unsigned int \*)a;

len = len / 4 + (len % 4 != 0);

ll rmder = 0;

for(int i = len-1; i >= 0; i--)

rmder = ((rmder << 32) + p[i]) % MOD;

int ans = 0;

if(rmder) ans = (int) (MOD - (rmder << 16) % MOD);

printf("%02X %02X\n", ans >> 8, ans % (1<<8));

}

return 0;

}

/\*

数论 概率 叫什么唯一分解定理?

题意：给p,q，表示概率p/q。 有n双红袜子，m双黑袜子，从这m+n双袜子里取出两双，两双都是红袜子的概率为p/q，求满足条件的最小n和m，不存在输出impossible（袜子总数最多为50000）

思路：记N = n+m。

首先对p/q进行化简，可得到p / q = n\*(n-1) / (N\*(N-1))

==> p \* N \* (N-1) = q \* n \* (n-1)

因为p,q互质，所以q | N\*(N-1), p | n\*(n-1)

然后这个时候就可以枚举N，搞出n，判断一下~~

知道思路就挺简单了....虽说思路也挺简单的，话说为什么我卡了两天？？？！！我是大沙茶...

\*/

//uva 10277

int main()

{

ll p, q;

while( scanf("%lld %lld", &p, &q) , p || q ) {

if(p == 0) { puts("0 2"); continue; }

ll d = GCD(p, q), N = 2;

p /= d, q /= d;

for(N = 2; N <= 50000; N++) if( N\*(N-1) % q == 0 ) {

ll x = N\*(N-1) / q \* p;

ll n = (ll) sqrt(x + 0.5) ;

if(n\*(n+1) == x) {

n++;

printf("%lld %lld\n", n, N-n);

break;

}

}

if(N > 50000) puts("impossible");

}

return 0;

}

/\*

数论 好题

题意：要求输出n^k的前3位和后3位(数据保证答案不少于6位数字)

思路：两数乘法的后三位自然是由各自后三位决定的，因此对于后三位我们完全可以用快速幂取模来做，但前三位就不行了。

但对于n^k我们可以做变形10^(k\*logn)，这样又由于10^x只是改变小数点的位置我们自然就不用去考虑了，因此前3位只取决于10^(2+fmod(k\*logn,1))，当输入最后结果的时候，由于要的是精确的前3位，所以应用强制转型忽略掉小数位，而不能直接用%.0f输出。

我一开始前后3位都是仅保留后4位来搞快速幂，WA了，还是不明白为何会WA。

\*/

//uva 11029

/\*

条件概率 好题

题意：n个人去商店，每个人买东西的概率为ai。问在确定有r个人买东西的情况下第i个人包括在内的概率

思路：直接根据条件概率的定义来

P(A|B) = P(AB)/P(B)

在这道题中B就是r个人买了东西

A就是某个人买了东西

然后考虑所有情况的概率，累加起来求的各个事件的概率

实现就是类似于二进制暴力枚举。

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*

\* 重点是P(AB)怎么表示 \*

\* 这里表示得很经典，值得学习 \*

\* \*

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*

疑问：我的第一个wa的程序(W\_WA.cpp)为什么不对？好累，看W\_WA.cpp，内有写。

\*/

void rec(double pre, ll stat)

{

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

if((1LL << i) & stat) pro\_a[i] += pre;

pro\_b += pre;

}

void dfs(int cur, int rest, double pre, ll stat)

{

if(rest < 0 || n - cur < rest) return ; //漏了r < 0这个判断郁闷了好久

if(cur == n) {

rec(pre, stat);

return ;

}

dfs(cur+1, rest-1, pre\*a[cur], stat|(1LL << cur));

dfs(cur+1, rest , pre\*(1-a[cur]), stat);

// if(n - cur < rest) return ;

// if(rest == 0) {

// double tmp = 1;

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

// tmp \*= ( (1LL << i) & stat ? a[i] : (1 - a[i]));

// pro\_b += tmp;

// for(int i = 0; i < n; i++) if((1LL << i) & stat) pro\_a[i] += tmp;

// return ;

// }

// for(int i = cur; i < n; i++) dfs(i+1, rest-1, pre, stat|(1LL<<i));

}

int main()

{

int r, Cas = 0;

while(scanf("%d %d", &n, &r) , n || r) {

printf("Case %d:\n", ++Cas);

for(int i = 0; i < n; i++) scanf("%lf", &a[i]), pro\_a[i] = 0;

pro\_b = 0;

dfs(0, r, 1, 0);

for(int i = 0; i < n; i++) pro\_a[i] = pro\_a[i] / pro\_b, printf("%.6f\n", pro\_a[i]);

}

return 0;

}

/\*

概率论 组合数学 推公式 好题

题意：n/2个汉堡，n/2个起司，分给n个人，按顺序投硬币决定是要汉堡还是要起司。问最后两个人得到同样食物的概率

思路：我们直接算所求概率是比较麻烦的，要考虑硬币抛到谁之后就不再抛了。但我们反过来求不合要求的概率就比较简单了，即最后两个人得到不同食物的概率，因为必然硬币到了最后时刻还在抛，那么对于任意一种最终不符合要求的情况，都是等概率事件。

那么我们只要把不合要求的情况找到，再乘上每种情况的概率即可：

一共有2^(n-2)种情况（最后两人是不计在内的），所以每种情况的概率为2^-(n-2)

不合要求的情况，必然是前n-2个人分别那道n/2-1个汉堡，n/2-1个起司，所以共有C( n/2-1, n-2 )种情况，即在前n-2个位置选出n/2-1个位置放置汉堡。

得f(n) = 2^(2-n) \* C( (n/2)/2, n-2 )

直接计算据说精度会挂掉，或者TLE。“受组合数的递推计算式的启发，对于f(n)的计算我们是否也可以用递推的方式计算呢？显然可以，因为我们有了f(n)的通项公式嘛，而且化成递推式后，我们会发现它出乎意料得简练 ”

组合数递推公式: C(n,k+1) = C(n,k) \* (n-k)/(k+1)

f(n) = 2^(2-n) \* C((n-2)/2, n-2)

f(n+2) = 2^(-n) \* C(n/2, n)

2^2 \* C( (n-2)/2, n-2 ) (n-2)! n/2! \* n/2! (n/2) \* (n/2) n

f(n) / f(n+2) = ------------------------ = ------------------ \* ------------------ = ------------------ = -----

C( n/2 , n ) (n-2)/2! \* (n-2)/2! n! (n-1) \* n 4\*(n-1)

so, f(n+2) = f(n) \* (n-1)/n

http://www.cnblogs.com/staginner/archive/2011/12/13/2286151.html

http://blog.csdn.net/sdust\_dc/article/details/9616173

\*/

void init()

{

f[2] = 1;

for(int i = 2; i < 100000; i += 2)

f[i+2] = f[i] \* (i-1) / i;

}

int main()

{

int n, cases;

init();

scanf("%d", &cases);

while(cases--) {

scanf("%d", &n);

printf("%.4f\n", 1 - f[n]);

}

return 0;

}

/\*

判断n能否被7/11/13整除

正整数n能被7/11/13整除，充要条件是n = ak\*1000^k + ak-1 \* 1000^(k-1) + ... + a0，7/11/13能整除(a0+a2+a4) - (a1+a3+..)的总和

证明是用同余的性质证明的，《初等数论》P51

\*/

/\*

数论 Lucas定理 + 扩展欧几里得(模逆) + 组合数学

题意：给出N, M, P, 求C(N+M, M) mod P. 1 <= n,m <= 1000000000,1 < p < 100000, P保证是素数

思路：Lucas定理: Lucas(N, M, P) 定义为C(N, M) % P, P一定要是素数

则:Lucas(N, M, P) = C(N%P, M%P) \* Lucas(N/P, M/P, P);

证明摘：Lucas 定理：A、B是非负整数，p是质数。AB写成p进制：A=a[n]a[n-1]...a[0]，B=b[n]b[n-1]...b[0]。

则组合数C(A,B)与C(a[n],b[n])\*C(a[n-1],b[n-1])\*...\*C(a[0],b[0]) modp同

即：Lucas(n,m,p)=c(n%p,m%p)\*Lucas(n/p,m/p,p)

首先我们注意到 n=(ak...a2,a1,a0)p = (ak...a2,a1)p \* p + a0

= [n/p]\*p+a0

且m=[m/p]+b0

只要我们更够证明 C(n,m)=C([n/p],[m/p]) \* C(a0,b0) (mod p)

剩下的工作由归纳法即可完成

我们知道对任意质数p: (1+x)^p == 1+(x^p) (mod p)

注意!这里一定要是质数 ................(为什么)

对 模p 而言

上式左右两边的x^m的系数对模p而言一定同余(为什么),其中左边的x^m的系数是 C(n,m) 而由于a0和b0都小于p

右边的x^m ( = x^(([m/p]\*p)+b0)) 一定是由 x^([m/p]\*p) 和 x^b0 相乘而得 (即发生于 i=[m/p] , j=b0 时) 因此我们就有了

C(n,m)=C([n/p],[m/p]) \* C(a0,b0) (mod p)

\*/

ll C(ll N, ll M, ll P)

{

if(N < M) return 0;

return A[N] \* mod\_inv(A[M], P) % P \* mod\_inv(A[N-M], P) % P;

}

ll Lucas(ll N, ll M, ll P)

{

if(M == 0) return 1;

return C(N % P, M % P, P) \* Lucas(N/P, M/P, P) % P;

}

int main()

{

while(cases--) {

scanf("%I64d %I64d %I64d", &N, &M, &P);

init\_fac(P);

printf("%I64d\n", Lucas(N+M, M, P));

}

return 0;

}

/\*

卡特兰数递推公式

我这个是n^2的，应该有O(n)的

\*/

void catalan()

{

f[0] = f[1] = 1;

for(int i = 2; i <= 10000; i++) {

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

f[i] += f[j]\*f[i-j], f[i] %= MOD;

}

}