数论倒数，又称逆元（因为我说习惯逆元了，下面我都说逆元）

先来引入求余概念

(a +  b) % p = (a%p +  b%p) %p  （对）

(a  -  b) % p = (a%p  -  b%p) %p  （对）

(a  \*  b) % p = (a%p \*  b%p) %p  （对）

(a  /  b) % p = (a%p  /  b%p) %p  （错）

对于一些题目，我们必须在中间过程中进行求余，否则数字太大，电脑存不下，那如果这个算式中出现除法，我们是不是对这个算式就无法计算了呢？

答案当然是 NO (>o<)

比如2 \* 3 % 5 = 1，那么3就是2关于5的逆元，或者说2和3关于5互为逆元

a的逆元，我们用inv(a)来表示

那么(a  /  b) % p = (a \* inv(b) ) % p = (a % p \* inv(b) % p) % p

这样就把除法，完全转换为乘法了

忘了说，a和p互质，a才有关于p的逆元

方法一：

费马小定理

a^(p-1) ≡1 (mod p)

两边同除以a

a^(p-2) ≡1/a (mod p)

什么(,,• ₃ •,,)，这可是数论，还敢写1/a

应该写a^(p-2) ≡ inv(a) (mod p)

所以inv(a) = a^(p-2) (mod p)

这个用快速幂求一下，复杂度O(logn)(ง •̀\_•́)ง

[复制代码](javascript:void(0);)

1 LL pow\_mod(LL a, LL b, LL p){//a的b次方求余p

2 LL ret = 1;

3 while(b){

4 if(b & 1) ret = (ret \* a) % p;

5 a = (a \* a) % p;

6 b >>= 1;

7 }

8 return ret;

9 }

10 LL Fermat(LL a, LL p){//费马求a关于b的逆元

11 return pow\_mod(a, p-2, p);

12 }

方法二：

a\*x + b\*y = 1

如果ab互质，有解

这个解的x就是a关于b的逆元

y就是b关于a的逆元

a\*x % b + b\*y % b = 1 % b

a\*x % b = 1 % b

a\*x = 1 (mod b)

所以x是a关于b的逆元

反之可证明y

附上代码：

[复制代码](javascript:void(0);)

1 #include<cstdio>

2 typedef long long LL;

3 void ex\_gcd(LL a, LL b, LL &x, LL &y, LL &d){

4 if (!b) {d = a, x = 1, y = 0;}

5 else{

6 ex\_gcd(b, a % b, y, x, d);

7 y -= x \* (a / b);

8 }

9 }

10 LL inv(LL t, LL p){//如果不存在，返回-1

11 LL d, x, y;

12 ex\_gcd(t, p, x, y, d);

13 return d == 1 ? (x % p + p) % p : -1;

14 }

15 int main(){

16 LL a, p;

17 while(~scanf("%lld%lld", &a, &p)){

18 printf("%lld\n", inv(a, p));

19 }

20 }

[复制代码](javascript:void(0);)

方法三：

当p是个质数的时候有  
inv(a) = (p - p / a) \* inv(p % a) % p

证明：  
设x = p % a,y = p / a  
于是有 x + y \* a = p  
(x + y \* a) % p = 0  
移项得 x % p = (-y) \* a % p  
x \* inv(a) % p = (-y) % p  
inv(a) = (p - y) \* inv(x) % p  
于是 inv(a) = (p - p / a) \* inv(p % a) % p

然后一直递归到1为止，因为1的逆元就是1

代码：

[复制代码](javascript:void(0);)

1 #include<cstdio>

2 typedef long long LL;

3 LL inv(LL t, LL p) {//求t关于p的逆元，注意:t要小于p，最好传参前先把t%p一下

4 return t == 1 ? 1 : (p - p / t) \* inv(p % t, p) % p;

5 }

6 int main(){

7 LL a, p;

8 while(~scanf("%lld%lld", &a, &p)){

9 printf("%lld\n", inv(a%p, p));

10 }

11 }

[复制代码](javascript:void(0);)

这个方法不限于求单个逆元，比前两个好，它可以在O(n)的复杂度内算出n个数的逆元

递归就是上面的写法，加一个记忆性递归，就可以了

递推这么写

[复制代码](javascript:void(0);)

1 #include<cstdio>

2 const int N = 200000 + 5;

3 const int MOD = (int)1e9 + 7;

4 int inv[N];

5 int init(){

6 inv[1] = 1;

7 for(int i = 2; i < N; i ++){

8 inv[i] = (MOD - MOD / i) \* 1ll \* inv[MOD % i] % MOD;

9 }

10 }

11 int main(){

12 init();

13 }

欧拉函数，用φ(n)表示

欧拉函数是求小于等于n的数中与n互质的数的数目

代码如下：

[复制代码](javascript:void(0);)

1 #include<cstdio>

2 using namespace std;

3 const int N = 1e6+10 ;

4 int phi[N], prime[N];

5 int tot;//tot计数，表示prime[N]中有多少质数

6 void Euler(){

7 phi[1] = 1;

8 for(int i = 2; i < N; i ++){

9 if(!phi[i]){

10 phi[i] = i-1;

11 prime[tot ++] = i;

12 }

13 for(int j = 0; j < tot && 1ll\*i\*prime[j] < N; j ++){

14 if(i % prime[j]) phi[i \* prime[j]] = phi[i] \* (prime[j]-1);

15 else{

16 phi[i \* prime[j] ] = phi[i] \* prime[j];

17 break;

18 }

19 }

20 }

21 }

22

23 int main(){

24 Euler();

25 }

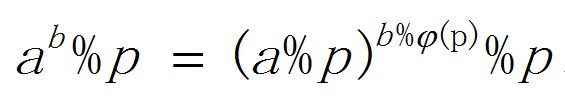
[复制代码](javascript:void(0);)

a^φ(p) ≡ 1 (mod p)

所以

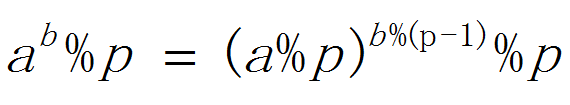
a^b % p  =  (a%p)^(b%φ(p)) % p

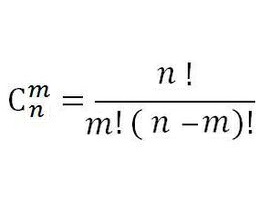
（欧拉函数前提是a和p互质）



如果p是质数

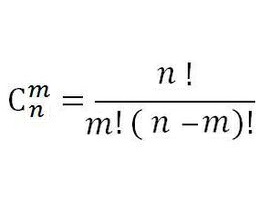
直接用这个公式



组合数并不陌生(´・ω・`)

因为大部分题都有求余，所以我们大可利用逆元的原理（没求余的题目，其实你也可以把MOD自己开的大一点，这样一样可以用逆元做）

根据这个公式



我们需要求阶乘和逆元阶乘

 我们就用1e9+7来求余吧

代码如下：

[复制代码](javascript:void(0);)

1 #include<cstdio>

2 const int N = 200000 + 5;

3 const int MOD = (int)1e9 + 7;

4 int F[N], Finv[N], inv[N];//F是阶乘，Finv是逆元的阶乘

5 void init(){

6 inv[1] = 1;

7 for(int i = 2; i < N; i ++){

8 inv[i] = (MOD - MOD / i) \* 1ll \* inv[MOD % i] % MOD;

9 }

10 F[0] = Finv[0] = 1;

11 for(int i = 1; i < N; i ++){

12 F[i] = F[i-1] \* 1ll \* i % MOD;

13 Finv[i] = Finv[i-1] \* 1ll \* inv[i] % MOD;

14 }

15 }

16 int comb(int n, int m){//comb(n, m)就是C(n, m)

17 if(m < 0 || m > n) return 0;

18 return F[n] \* 1ll \* Finv[n - m] % MOD \* Finv[m] % MOD;

19 }

20 int main(){

21 init();

22 }

中国剩余定理，又名孙子定理o(\*≧▽≦)ツ

问题：一堆物品3个3个分剩2个，5个5个分剩3个，7个7个分剩2个问这个物品有多少个

[复制代码](javascript:void(0);)

1 //n个方程：x=a[i](mod m[i]) (0<=i<n)

2 LL china(int n, LL \*a, LL \*m){

3 LL M = 1, ret = 0;

4 for(int i = 0; i < n; i ++) M \*= m[i];

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

6 LL w = M / m[i];

7 ret = (ret + w \* inv(w, m[i]) \* a[i]) % M;

8 }

9 return (ret + M) % M;

10 }

[复制代码](javascript:void(0);)

**问题描述：**

     人自出生起就有体力，情感和智力三个生理周期，分别为23，28和33天。一个周期内有一天为峰值，在这一天，人在对应的方面（体力，情感或智力）表现最好。通常这三个周期的峰值不会是同一天。现在给出三个日期，分别对应于体力，情感，智力出现峰值的日期。然后再给出一个起始日期，要求从这一天开始，算出最少再过多少天后三个峰值同时出现。

**分析：**

因为23 = 23

28 = 2\*2\*7

33 = 3\*11

满足两两互质关系，所以直接套模板就好了

：

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 #include<cstdio>

2 typedef long long LL;

3 const int N = 100000 + 5;

4 void ex\_gcd(LL a, LL b, LL &x, LL &y, LL &d){

5 if (!b) {d = a, x = 1, y = 0;}

6 else{

7 ex\_gcd(b, a % b, y, x, d);

8 y -= x \* (a / b);

9 }

10 }

11 LL inv(LL t, LL p){//如果不存在，返回-1

12 LL d, x, y;

13 ex\_gcd(t, p, x, y, d);

14 return d == 1 ? (x % p + p) % p : -1;

15 }

16 LL china(int n, LL \*a, LL \*m){//中国剩余定理

17 LL M = 1, ret = 0;

18 for(int i = 0; i < n; i ++) M \*= m[i];

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

20 LL w = M / m[i];

21 ret = (ret + w \* inv(w, m[i]) \* a[i]) % M;

22 }

23 return (ret + M) % M;

24 }

25 int main(){

26 LL p[3], r[3], d, ans, MOD = 21252;

27 int cas = 0;

28 p[0] = 23; p[1] = 28; p[2] = 33;

29 while(~scanf("%I64d%I64d%I64d%I64d", &r[0], &r[1], &r[2], &d) && (~r[0] || ~r[1] || ~r[2] || ~d)){

30 ans = ((china(3, r, p) - d) % MOD + MOD) % MOD;

31 printf("Case %d: the next triple peak occurs in %I64d days.\n", ++cas, ans ? ans : 21252);

32 }

33

34 }

[复制代码](javascript:void(0);)

*m*1,*m*2, ... ,*m*n两两不保证互质，辣怎么办(っ °Д °)っ

别怕，看我接着抛代码

[复制代码](javascript:void(0);)

1 #include<cstdio>

2 #include<algorithm>

3 using namespace std;

4 typedef long long LL;

5 typedef pair<LL, LL> PLL;

6 PLL linear(LL A[], LL B[], LL M[], int n) {//求解A[i]x = B[i] (mod M[i]),总共n个线性方程组

7 LL x = 0, m = 1;

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

9 LL a = A[i] \* m, b = B[i] - A[i]\*x, d = gcd(M[i], a);

10 if(b % d != 0) return PLL(0, -1);//答案不存在，返回-1

11 LL t = b/d \* inv(a/d, M[i]/d)%(M[i]/d);

12 x = x + m\*t;

13 m \*= M[i]/d;

14 }

15 x = (x % m + m ) % m;

16 return PLL(x, m);//返回的x就是答案，m是最后的lcm值

17 }

[复制代码](javascript:void(0);)

大组合数

我们学了O(n^2)的做法，加上逆元，我们又会了O(n)的做法

现在来了新问题，如果n和m很大呢，

比如求C(n, m) % p  ， n<=1e18,m<=1e18,p<=1e5

看到没有，n和m这么大，但是p却很小，我们要利用这个p

卢卡斯说：

C(n, m) % p  =  C(n / p, m / p) \* C(n%p, m%p) % p

对于C(n / p, m / p)，如果n / p 还是很大，可以递归下去，一直到世界的尽头

1 LL Lucas(LL n, LL m, int p){

2 return m ? Lucas(n/p, m/p, p) \* comb(n%p, m%p, p) % p : 1;

3 }

hdu 5446

http://acm.hdu.edu.cn/showproblem.php?pid=5446

题意：

给你三个数n, m, k

第二行是k个数，p1,p2,p3...pk

所有p的值不相同且p都是质数

求C(n, m) % (p1\*p2\*p3\*...\*pk)的值

范围：1≤m≤n≤1e18，1≤k≤10，pi≤1e5，保证p1\*p2\*p3\*...\*pk≤1e18

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 #include<cstdio>

2 typedef long long LL;

3 const int N = 100000 + 5;

4 LL mul(LL a, LL b, LL p){//快速乘，计算a\*b%p

5 LL ret = 0;

6 while(b){

7 if(b & 1) ret = (ret + a) % p;

8 a = (a + a) % p;

9 b >>= 1;

10 }

11 return ret;

12 }

13 LL fact(int n, LL p){//n的阶乘求余p

14 LL ret = 1;

15 for (int i = 1; i <= n ; i ++) ret = ret \* i % p ;

16 return ret ;

17 }

18 void ex\_gcd(LL a, LL b, LL &x, LL &y, LL &d){

19 if (!b) {d = a, x = 1, y = 0;}

20 else{

21 ex\_gcd(b, a % b, y, x, d);

22 y -= x \* (a / b);

23 }

24 }

25 LL inv(LL t, LL p){//如果不存在，返回-1

26 LL d, x, y;

27 ex\_gcd(t, p, x, y, d);

28 return d == 1 ? (x % p + p) % p : -1;

29 }

30 LL comb(int n, int m, LL p){//C(n, m) % p

31 if (m < 0 || m > n) return 0;

32 return fact(n, p) \* inv(fact(m, p), p) % p \* inv(fact(n-m, p), p) % p;

33 }

34 LL Lucas(LL n, LL m, int p){

35 return m ? Lucas(n/p, m/p, p) \* comb(n%p, m%p, p) % p : 1;

36 }

37 LL china(int n, LL \*a, LL \*m){//中国剩余定理

38 LL M = 1, ret = 0;

39 for(int i = 0; i < n; i ++) M \*= m[i];

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

41 LL w = M / m[i];

42 //ret = (ret + w \* inv(w, m[i]) \* a[i]) % M;//这句写了会WA，用下面那句

43 ret = (ret + mul(w \* inv(w, m[i]), a[i], M)) % M;

44 //因为这里直接乘会爆long long ,所以我用快速乘(unsigned long long也是爆掉，除非用高精度)

45 }

46 return (ret + M) % M;

47 }

48 int main(){

49 int T, k;

50 LL n, m, p[15], r[15];

51 scanf("%d", &T);

52 while(T--){

53 scanf("%I64d%I64d%d", &n, &m, &k);

54 for(int i = 0; i < k; i ++){

55 scanf("%I64d", &p[i]);

56 r[i] = Lucas(n, m, p[i]);

57 }

58 printf("%I64d\n", china(k, r, p));

59 }

60 }

[复制代码](javascript:void(0);)

我们知道题目要求C(n, m) % (p1\*p2\*p3\*...\*pk)的值

其实这个就是中国剩余定理最后算出结果后的最后一步求余

那C(n, m)相当于以前我们需要用中国剩余定理求的值

然而C(n, m)太大，我们只好先算出

C(n, m) % p1 = r1

C(n, m) % p2 = r2

C(n, m) % p3 = r3

C(n, m) % pk = rk

用Lucas，这些r1,r2,r3...rk可以算出来

然后又是用中国剩余定理求答

三分！！！！！！！！！！！！！！单峰函数求极值

# 题意：给出n个点的初始坐标以及每秒移动方向，让你计算哪个时刻距离最远的点的距离最小

点的距离是单峰函数，那么n（n-1）个单峰函数的max也是个单峰函数，直接套三分

这道题wa了一发，原因是我求距离的时候没有求平方根！！！！啊啊啊

感觉自己写题目都是思路没问题实现一堆问题啊阿西吧呜呜呜 （ノ\*-\_-\*）ノ

上代码

#include<iostream>

#include<cstdio>

#include<cstring>

#include<stack>

#include<queue>

#include<vector>

#include<cmath>

#include<algorithm>

#define ll long long

#define eps 1e-6

using namespace std;

struct point{

int x,y,dx,dy;

point(){}

point(int xx,int yy,int ddx,int ddy){x = xx;y = yy;dx = ddx;dy = ddy;}

}poi[310];

int n;

double dist(int i,int j,double t){

double cx1 = poi[i].x + poi[i].dx\*t;

double cy1 = poi[i].y + poi[i].dy\*t;

double cx2 = poi[j].x + poi[j].dx\*t;

double cy2 = poi[j].y + poi[j].dy\*t;

return (cx2 - cx1)\*(cx2 - cx1) + (cy2 - cy1)\*(cy2 - cy1);

}

double cal(double t){

double max\_dis = 0;;

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

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

if(dist(i,j,t) > max\_dis)

max\_dis = dist(i,j,t);

}

}

return max\_dis;

}

double san\_fen(double l0,double r0){

double l = l0,r = r0,lmid,rmid,v1,v2;

while(r - l > eps){//浮点数的比较,eps为误差

lmid = l + (r-l)/3;

rmid = r - (r-l)/3;

//或者 lmid = (2\*l + r)/3,rmid = (l + r\*2 + 2)/3;

v1 = cal(lmid);

v2 = cal(rmid);

if(v1 > v2) l = lmid;//寻找的是最小值

else r = rmid;

}

return r;

}

int main(){

int t0;

scanf("%d",&t0);

double t;

int a,b,c,d;

for(int tt = 1;tt <= t0;tt++){

scanf("%d",&n);

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

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

poi[i] = point(a,b,c,d);

}

t = san\_fen(0,1e5);

printf("Case #%d: %.2f %.2f\n",tt,t,sqrt(cal(t)));

}

return 0;

}

[复制代码](javascript:void(0);)

Sample Input

2 2 0 0 1 0 2 0 -1 0 2 0 0 1 0 2 1 -1 0

Sample Output

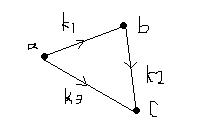
Case #1: 1.00 0.00 Case #2: 1.00 1.00

差分约束！

一直不知道差分约束是什么类型题目，最近在写最短路问题就顺带看了下，原来就是给出一些形如x-y<=b不等式的约束，问你是否满足有解的问题

好神奇的是这类问题竟然可以转换成图论里的最短路径问题，下面开始详细介绍下

比如给出三个不等式,b-a<=k1,c-b<=k2,c-a<=k3,求出c-a的最大值,我们可以把a,b,c转换成三个点，k1，k2，k3是边上的权，如图



由题我们可以得知，这个有向图中，由题b-a<=k1,c-b<=k2,得出c-a<=k1+k2,因此比较k1+k2和k3的大小，求出最小的就是c-a的最大值了

根据以上的解法，我们可能会猜到求解过程实际就是求从a到c的最短路径，没错的....简单的说就是从a到c沿着某条路径后把所有权值和k求出就是c -a<=k的一个

推广的不等式约束，既然这样，满足题目的肯定是最小的k，也就是从a到c最短距离...

理解了这里之后，想做题还是比较有困难的，因为题目需要变形一下，不能单纯的算..

首先以poj3159为例,这个比较简单，就是给出两个点的最大差，然后让你求1到n的最大差，直接建图后用bellman或者spfa就可以过了

稍微难点的就是poj1364，因为他给出的不等式不是x-y<=k形式，有时候是大于号，这样需要我们去变形一下，并且给出的还是>,<没有等于，都要变形

再有就是poj1201，他要求出的是最长距离，那就要把形式变换成x-y>=k的标准形式

注意点:

1. 如果要求最大值想办法把每个不等式变为标准x-y<=k的形式,然后建立一条从y到x权值为k的边,变得时候注意x-y<k =>x-y<=k-1

   如果要求最小值的话,变为x-y>=k的标准形式，然后建立一条从y到x的k边，求出最长路径即可

2.如果权值为正，用dj，spfa，bellman都可以，如果为负不能用dj，并且需要判断是否有负环，有的话就不存在

博弈

有一种很有意思的游戏，就是有物体若干堆，可以是火柴棍或是围棋子等等均可。两个  
人轮流从堆中取物体若干，规定最后取光物体者取胜。这是我国民间很古老的一个游戏  
，别看这游戏极其简单，却蕴含着深刻的数学原理。下面我们来分析一下要如何才能够  
取胜。

（一）**巴什博奕（Bash Game）：**只有一堆n个物品，两个人轮流从这堆物品中取物，规  
定每次至少取一个，最多取m个。最后取光者得胜。

    显然，如果n=m+1，那么由于一次最多只能取m个，所以，无论先取者拿走多少个，  
后取者都能够一次拿走剩余的物品，后者取胜。因此我们发现了如何取胜的法则：如果  
n=（m+1）r+s，（r为任意自然数，s≤m),那么先取者要拿走s个物品，如果后取者拿走  
k（≤m)个，那么先取者再拿走m+1-k个，结果剩下（m+1）（r-1）个，以后保持这样的  
取法，那么先取者肯定获胜。总之，要保持给对手留下（m+1）的倍数，就能最后获胜。  
    这个游戏还可以有一种变相的玩法：两个人轮流报数，每次至少报一个，最多报十  
个，谁能报到100者胜。  
（二）**威佐夫博奕（Wythoff Game）：**有两堆各若干个物品，两个人轮流从某一堆或同  
时从两堆中取同样多的物品，规定每次至少取一个，多者不限，最后取光者得胜。  
  
    这种情况下是颇为复杂的。我们用（ak，bk）（ak ≤ bk ,k=0，1，2，…,n)表示  
两堆物品的数量并称其为**局势**，如果甲面对（0，0），那么甲已经输了，这种局势我们  
称为**奇异局势**。前几个奇异局势是：（0，0）、（1，2）、（3，5）、（4，7）、（6，  
10）、（8，13）、（9，15）、（11，18）、（12，20）。

    可以看出,a0=b0=0,ak是未在前面出现过的最小自然数,而 bk= ak + k，奇异局势有  
如下三条性质：

    1。任何自然数都包含在一个且仅有一个奇异局势中。  
    由于ak是未在前面出现过的最小自然数，所以有ak > ak-1 ，而 bk= ak + k > ak  
-1 + k-1 = bk-1 > ak-1 。所以性质1。成立。  
    2。任意操作都可将奇异局势变为非奇异局势。  
    事实上，若只改变奇异局势（ak，bk）的某一个分量，那么另一个分量不可能在其  
他奇异局势中，所以必然是非奇异局势。如果使（ak，bk）的两个分量同时减少，则由  
于其差不变，且不可能是其他奇异局势的差，因此也是非奇异局势。  
    3。采用适当的方法，可以将非奇异局势变为奇异局势。

    假设面对的局势是（a,b），若 b = a，则同时从两堆中取走 a 个物体，就变为了  
奇异局势（0，0）；如果a = ak ，b > bk，那么，取走b  – bk个物体，即变为奇异局  
势；如果 a = ak ，  b < bk ,则同时从两堆中拿走 ak – ab + ak个物体,变为奇异局  
势（ ab – ak , ab – ak+ b – ak）；如果a > ak ，b= ak + k,则从第一堆中拿走多余  
的数量a – ak 即可；如果a < ak ，b= ak + k,分两种情况，第一种，a=aj （j < k）  
,从第二堆里面拿走 b – bj 即可；第二种，a=bj （j < k）,从第二堆里面拿走 b – a  
j 即可。

    从如上性质可知，两个人如果都采用正确操作，那么面对非奇异局势，先拿者必胜  
；反之，则后拿者取胜。

    那么任给一个局势（a，b），怎样判断它是不是奇异局势呢？我们有如下公式：

    ak =[k（1+√5）/2]，bk= ak + k  （k=0，1，2，…,n 方括号表示取整函数)  
  
奇妙的是其中出现了黄金分割数（1+√5）/2 = 1。618…,因此,由ak，bk组成的矩形近  
似为黄金矩形，由于2/（1+√5）=（√5-1）/2，可以先求出j=[a（√5-1）/2]，若a=[  
j（1+√5）/2]，那么a = aj，bj = aj + j，若不等于，那么a = aj+1，bj+1 = aj+1  
+ j + 1，若都不是，那么就不是奇异局势。然后再按照上述法则进行，一定会遇到奇异  
局势。

（三）**尼姆博奕（Nimm Game）：**有三堆各若干个物品，两个人轮流从某一堆取任意多的  
物品，规定每次至少取一个，多者不限，最后取光者得胜。

    这种情况最有意思，它与二进制有密切关系，我们用（a，b，c）表示某种局势，首  
先（0，0，0）显然是奇异局势，无论谁面对奇异局势，都必然失败。第二种奇异局势是  
（0，n，n），只要与对手拿走一样多的物品，最后都将导致（0，0，0）。仔细分析一  
下，（1，2，3）也是奇异局势，无论对手如何拿，接下来都可以变为（0，n，n）的情  
形。

    计算机算法里面有一种叫做按位模2加，也叫做异或的运算，我们用符号（+）表示  
这种运算。这种运算和一般加法不同的一点是1+1=0。先看（1，2，3）的按位模2加的结  
果：

1 =二进制01  
2 =二进制10  
3 =二进制11 （+）  
———————  
0 =二进制00 （注意不进位）

    对于奇异局势（0，n，n）也一样，结果也是0。

    任何奇异局势（a，b，c）都有a（+）b（+）c =0。

如果我们面对的是一个非奇异局势（a，b，c），要如何变为奇异局势呢？假设 a < b  
< c,我们只要将 c 变为 a（+）b,即可,因为有如下的运算结果: a（+）b（+）(a（+）  
b)=(a（+）a)（+）(b（+）b)=0（+）0=0。要将c 变为a（+）b，只要从 c中减去 c-（  
a（+）b）即可。  
  
    例1。（14，21，39），14（+）21=27，39-27=12，所以从39中拿走12个物体即可达  
到奇异局势（14，21，27）。

    例2。（55，81，121），55（+）81=102，121-102=19，所以从121中拿走19个物品  
就形成了奇异局势（55，81，102）。

    例3。（29，45，58），29（+）45=48，58-48=10，从58中拿走10个，变为（29，4  
5，48）。

    例4。我们来实际进行一盘比赛看看：  
        甲:(7,8,9)->(1,8,9)奇异局势  
        乙:(1,8,9)->(1,8,4)  
        甲:(1,8,4)->(1,5,4)奇异局势  
        乙:(1,5,4)->(1,4,4)  
        甲:(1,4,4)->(0,4,4)奇异局势  
        乙:(0,4,4)->(0,4,2)  
        甲:(0.4,2)->(0,2,2)奇异局势  
        乙:(0,2,2)->(0,2,1)  
        甲:(0,2,1)->(0,1,1)奇异局势  
        乙:(0,1,1)->(0,1,0)  
        甲:(0,1,0)->(0,0,0)奇异局势  
        甲胜。

取火柴的游戏  
题目1：今有若干堆火柴，两人依次从中拿取，规定每次只能从一堆中取若干根，   
可将一堆全取走，但不可不取，最后取完者为胜，求必胜的方法。   
题目2：今有若干堆火柴，两人依次从中拿取，规定每次只能从一堆中取若干根，   
可将一堆全取走，但不可不取，最后取完者为负，求必胜的方法。  
嘿嘿，这个游戏我早就见识过了。小时候用珠算玩这个游戏：第一档拨一个，第二档拨两个，依次直到第五档拨五个。然后两个人就轮流再把棋子拨下来，谁要是最后一个拨谁就赢。有一次暑假看见两个小孩子在玩这个游戏，我就在想有没有一个定论呢。下面就来试着证明一下吧  
先解决第一个问题吧。  
定义：若所有火柴数异或为0，则该状态被称为利他态，用字母T表示；否则，   
为利己态，用S表示。  
[定理1]：对于任何一个S态，总能从一堆火柴中取出若干个使之成为T态。  
证明：  
    若有n堆火柴，每堆火柴有A(i)根火柴数，那么既然现在处于S态，  
      c = A(1) xor A(2) xor … xor A(n) > 0;  
    把c表示成二进制，记它的二进制数的最高位为第p位，则必然存在一个A(t),它二进制的第p位也是1。（否则，若所有的A(i)的第p位都是0，这与c的第p位就也为0矛盾）。  
    那么我们把x = A(t) xor c,则得到x < A(t).这是因为既然A(t)的第p位与c的第p位同为1,那么x的第p位变为0,而高于p的位并没有改变。所以x < A(t).而  
    A(1) xor A(2) xor … xor x xor … xor A(n)  
  = A(1) xor A(2) xor … xor A(t) xor c xor … xor A(n)  
  = A(1) xor A(2) xor… xor A(n) xor A(1) xor A(2) xor … xor A(n)  
  = 0  
这就是说从A(t)堆中取出 A(t) – x 根火柴后状态就会从S态变为T态。证毕  
[定理2]：T态，取任何一堆的若干根，都将成为S态。  
证明：用反证法试试。  
      若  
      c = A(1) xor A(2) xor … xor A(i) xor … xor A(n) = 0；  
      c’ = A(1) xor A(2) xor … xor A(i’) xor c xor … xor A(n) = 0;  
      则有  
c xor c’ = A(1) xor A(2) xor … xor A(i) xor … xor A(n) xor A(1) xor A(2) xor … xor A(i’) xor c xor … xor A(n) = A(i) xor A(i’) =0  
      进而推出A(i) = A(i’)，这与已知矛盾。所以命题得证。  
[定理 3]：S态，只要方法正确，必赢。   
  最终胜利即由S态转变为T态，任何一个S态，只要把它变为T态，（由定理1，可以把它变成T态。）对方只能把T态转变为S态(定理2)。这样，所有S态向T态的转变都可以有己方控制，对方只能被动地实现由T态转变为S态。故S态必赢。  
[定理4]：T态，只要对方法正确，必败。   
  由定理3易得。   
接着来解决第二个问题。  
定义：若一堆中仅有1根火柴，则被称为孤单堆。若大于1根，则称为充裕堆。  
定义：T态中，若充裕堆的堆数大于等于2，则称为完全利他态，用T2表示；若充裕堆的堆数等于0，则称为部分利他态，用T0表示。  
   
孤单堆的根数异或只会影响二进制的最后一位，但充裕堆会影响高位（非最后一位）。一个充裕堆，高位必有一位不为0，则所有根数异或不为0。故不会是T态。  
[定理5]：S0态，即仅有奇数个孤单堆，必败。T0态必胜。   
证明：  
S0态，其实就是每次只能取一根。每次第奇数根都由己取，第偶数根都由对   
方取，所以最后一根必己取。败。同理,  T0态必胜#  
[定理6]：S1态，只要方法正确，必胜。   
证明：  
若此时孤单堆堆数为奇数，把充裕堆取完；否则，取成一根。这样，就变成奇数个孤单堆，由对方取。由定理5，对方必输。己必胜。  #   
[定理7]：S2态不可转一次变为T0态。   
证明：  
充裕堆数不可能一次由2变为0。得证。  #

[定理8]：S2态可一次转变为T2态。   
证明：  
由定理1，S态可转变为T态，态可一次转变为T态，又由定理6，S2态不可转一次变为T0态，所以转变的T态为T2态。  #   
[定理9]：T2态，只能转变为S2态或S1态。   
证明：  
由定理2，T态必然变为S态。由于充裕堆数不可能一次由2变为0，所以此时的S态不可能为S0态。命题得证。   
[定理10]：S2态，只要方法正确，必胜.   
证明：  
方法如下：   
      1）  S2态，就把它变为T2态。（由定理8）   
      2）  对方只能T2转变成S2态或S1态（定理9）  
    若转变为S2,  转向1）   
    若转变为S1,  这己必胜。（定理5）   
[定理11]：T2态必输。   
证明：同10。   
综上所述，必输态有：  T2,S0   
          必胜态：    S2,S1,T0.   
两题比较：   
第一题的全过程其实如下：   
S2->T2->S2->T2->  ……  ->T2->S1->T0->S0->T0->……->S0->T0(全0)   
第二题的全过程其实如下：   
S2->T2->S2->T2->  ……  ->T2->S1->S0->T0->S0->……->S0->T0(全0)   
下划线表示胜利一方的取法。  是否发现了他们的惊人相似之处。   
我们不难发现(见加黑部分)，S1态可以转变为S0态（第二题做法），也可以转变为   
T0（第一题做法）。哪一方控制了S1态，他即可以有办法使自己得到最后一根（转变为   
T0）,也可以使对方得到最后一根（转变为S0）。   
  所以，抢夺S1是制胜的关键！   
  为此，始终把T2态让给对方，将使对方处于被动状态，他早晚将把状态变为S1.

Hdu 1536 nim博弈加上sg函数

#include<cstdio>

#include<algorithm>

#include<iostream>

#include<cmath>

#include<set>

#include<cstring>

using namespace std;

const int N = 10005;//每堆个数的最大值

int sg[N],a[105];

bool Hash[N];

void sg\_solve(int \*s,int t){ ///N求解范围 S[]数组是可以每次取的值，t是s的长度。下标从0开始

int i,j;

memset(sg,0,sizeof(sg));

for(i=1; i<N; i++){

memset(Hash,0,sizeof(Hash));

for(j=0; j<t; j++)

if(i - s[j] >= 0)

Hash[sg[i-s[j]]] = 1;

for(j=0; j<N; j++)

if(!Hash[j])

break;

sg[i] = j;

}

}

int main(){

int n,n2,n3,res,cur;

int s[110];

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

if(n == 0)

break;

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

scanf("%d",&s[i]);

}

sg\_solve(s,n);

scanf("%d",&n2);

while(n2--){

scanf("%d",&n3);

res = 0;

while(n3--){

scanf("%d",&cur);

res = res^sg[cur];

}

if(res == 0)printf("L");

else printf("W");

}

printf("\n");

}

return 0;

}

**扩展欧几里德算法**

首先， ax+by = gcd(a, b) 这个公式肯定有解 （( •̀∀•́ )她说根据数论中的相关定理可以证明，反正我信了）

所以 ax+by = gcd(a, b) \* k 也肯定有解 （废话，把x和y乘k倍就好了）

所以，这个公式我们写作ax+by = d，(gcd(a, b) | d)

gcd(a, b) | d，表示d能整除gcd，这个符号在数学上经常见

那么已知 a，b 求 一组解 x，y 满足 ax+by = gcd(a, b) 这个公式

[复制代码](javascript:void(0);)

1 #include<cstdio>

2 typedef long long LL;

3 void extend\_Eulid(LL a, LL b, LL &x, LL &y, LL &d){

4 if (!b) {d = a, x = 1, y = 0;}

5 else{

6 extend\_Eulid(b, a % b, y, x, d);

7 y -= x \* (a / b);

8 }

9 }

10 int main(){

11 LL a, b, d, x, y;

12 while(~scanf("%lld%lld", &a, &b)){

13 extend\_Eulid(a, b, x, y, d);

14 printf("%lld\*a + %lld\*b = %lld\n", x, y, d);

15 }

16 }