## 公约数和公倍数

<http://acm.nyist.net/JudgeOnline/problem.php?pid=40>

## 这道题目是非常基础的题目，在学习了欧几里德算法之后，应该很好就能做的出来了。注意两个数的最小公倍数和最大公约数之间有关系为：a\*b=gcd(a,b)\*lcm(a,b);

## 代码如下：

#include<iostream>

using namespace std;

inline int Gcd(int m,int n) //求最大公约数

{

if (m==0) return n;

return Gcd(n%m,m);

}

int main()

{

int n,a,b,g;

cin>>n;

while(n--)

{

cin>>a>>b;

g=Gcd(a,b);

cout<<g<<" "<<a\*b/g<<endl;

}

}

## Biorhythms

<http://acm.nyist.net/JudgeOnline/problem.php?pid=151>

## 很经典的中国剩余问题。

## 题目大意是：人的体力、情绪和智力都是有周期的，分别是23天、28天和33天。分别给出你体力、情绪和智力最高值过后的天数p、e、d，然后让你计算下一次三者同时达到最高值的时间。

## 那么再次达到最高值时间为n，则有：

## 

## 那么找到k1、k2、k3满足：

k1：k1%23==0&&k1%28==0&&k1%33==1

k2：k2%33==0&&k2%28==0&&k2%23==1

k3：k3%33==0&&k3%23==0&&k3%28==1

则n=k2\*p+k3\*e+k1\*i+k\*21252；

代码如下：

#include <stdio.h>  
int main()  
{  
int n,a,b,c,t;  
while(scanf("%d%d%d%d",&a,&b,&c,&t),~a)  
{  
n=(5544\*a+14421\*b+1288\*c)%21252-t;  
if(n<=0)  
n+=21252;  
printf("%d\n",n);  
}  
}

## 韩信点兵

<http://acm.nyist.net/JudgeOnline/problem.php?pid=34>

这个题目也是很经典的中国剩余问题类的题目，思路跟上面差不多这道题目因为数据范围很小实际上暴力就可以过，但是这个题目不失为练习中国剩余的很好题目，所以建议大家用中国剩余定理做一下。

直接给出代码：

暴力求解代码：

#include <stdio.h>

main()

{

int a,b,c,n;

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

for(n=11;n<100;n++)

if(n%3==a&&n%5==b&&n%7==c)

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

}

中国剩余定理思路代码：

#include<stdio.h>

int main()

{

int a,b,c,m;

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

m=(70\*a+21\*b+15\*c)%105;

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

return 0;

}

## 次方求模

<http://acm.nyist.net/JudgeOnline/problem.php?pid=102>

这个题目是要求出a的b次方对c取余的值。

显然我们不能求出a^b后再对c取余，a^b可能是一个很大的数，而且这样做肯定很慢。那么我们利用同余定理来解决这个问题。

当然最简单的我们会想到：a^b%c=((a^(b-1)%c)\*(a%c))%c；

但是这样效率依然是很低的。

那么我们考虑一种叫做二分快速幂的思路。

关键改进就是：a^b%c=((a^(b/2)%c)\* (a^(b/2)%c))%c

比如我们要算3^25%4的值，由于25=1+8+16，那么我们就只需要知道3^1,3^8,3^16的值。这样复杂度就从O(n)降低为O(log(n))了。

代码实现如下：

#include <iostream>

using namespace std;

int mod(int k,int x,int c)

{

int a=1;

long long r=k;

while(x)

{

if(x&1)

a=(a\*r)%c;

r=((r%c)\*(r%c))%c;

x=x>>1;

}

return a;

}

int main()

{

int n,a,b,c;

cin>>n;

while(n--)

{

cin>>a>>b>>c;

cout<<mod(a,b,c)<<endl;

}

}

## 青蛙的约会

<http://poj.org/problem?id=1061>

这道题目用到了扩展欧几里德算法。

设过s步后两青蛙相遇，则必满足以下等式：(x+m\*s)-(y+n\*s)=k\*l(k=0,1,2....)

稍微变一下形得：(n-m)\*s+k\*l=x-y

令n-m=a,k=b,x-y=c,即a\*s+b\*l=c

只要上式存在整数解，则两青蛙能相遇，否则不能。

这样问题就转化为了扩展欧几里德问题了。

代码如下：

# include <stdio.h>

\_\_int64 gcd(\_\_int64 a,\_\_int64 b)

{

       if(b==0)

              return a;

       return gcd(b,a%b);

}

void exgcd(\_\_int64 a,\_\_int64 b,\_\_int64 &m,\_\_int64 &n)

{

       if(b==0)

       {

              m=1;

              n=0;

              return ;

       }

       exgcd(b,a%b,m,n);

       \_\_int64 t;

       t=m;

       m=n;

       n=t-a/b\*n;

}

int main()

{

       \_\_int64 x,y,m,n,l,a,b,c,k1,k2,r,t;

       while(scanf("%I64d%I64d%I64d%I64d%I64d",&x,&y,&m,&n,&l)!=EOF)

       {

              a=n-m;

              b=l;

              c=x-y;

              r=gcd(a,b);

              if(c%r)

              {

                     printf("Impossible\n");

                     continue;

              }

              a/=r;

              b/=r;

              c/=r;

              exgcd(a,b,k1,k2);

              t=c\*k1/b;//注1

              k1=c\*k1-t\*b;

              if(k1<0)

                     k1+=b;

              printf("%I64d\n",k1);

       }

       return 0;

}

## Least Common Multiple

<http://acm.hdu.edu.cn/showproblem.php?pid=1019>

这道题目是要求出多个数的最小公倍数，求n个数的最小公倍数我们最容易想到的思路就是求出两个数的最小公倍数，然后再用这个最小公倍数与第三个数球最小公倍数，依次下去就可以求出n个数的最小公倍数了。至于两个数的最小公倍数我们从上面的习题中已经可以知道方法了。

a\*b=gcd(a,b)\*lcm(a,b);

那么我们考虑这种方法的复杂度，会发现我们要求出n个数的gcd，那么我们又更好的选择吗？是的，想想我们小学时间最开始学习最小公倍数小、最大公约数的时候，那时间我们是怎么求的呢？我们采用了一种叫做短除法的算法。

示例如下：

2|\_\_12\_\_\_\_\_\_16\_\_\_\_\_\_\_24\_\_

2| 6 8 12

2| 3 4 6

此时我们发现3 、4、 6是互质的，所以12、16、24的最小公倍数就是

2\*2\*2\*3\*4\*6=576

当然这种方法的代码难度会略高于上一种思路。

附上代码（第一种思路）：

#include <stdio.h>

int gcd(int x,int y)

{

return x?gcd(y%x,x):y;

}

int main()

{

int i,j,n,m,ret,tem;

scanf("%d",&n);

while(n--)

{

scanf("%d",&m);

scanf("%d",&ret);

m--;

while(m--)

{

scanf("%d",&tem);

ret=ret/gcd(ret,tem)\*tem;

}

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

}

}

## C Looooops

<http://poj.org/problem?id=2115>

扩展欧几里德，不是太裸的题目：

意思是输入4个数：a, b, c, k;(0<=a, b, c<2^k, 1=<k<=32)

计算是否存在x,使（a+c\*x )≡ b(mod 2^k)；如果存在输出x,否则输出FOREVER;

思路：首先知道存在temp使(a+temp)≡ b(mod 2^k)；可得，temp = (b-a)%(2^k);现在问题是计算是否存在x,使x\*c ≡ temp(mod 2^k)；也就是求x\*c-(x\*c/(2^k)) \* 2^k = temp - (temp/2^k) \* 2^k;

既x\*c-(x\*c/(2^k) - temp / (2^k))\*(2^k) = temp;既x\*c-y\*(2^k) = temp;因为c, k, temp都以知，（就是二元一次方程组（a\*x0+b\*y0=c的形式,知道如果（GCD(a, b)若不能整除c,可以判断方程无解））

现在问题是计算x，计算gcd = gcd(c, 2^k),先判断是否有解；

若有解，方程两边除以gcd：因为gcd（c/gcd, 2^k/gcd）= 1,所以用扩展欧几里德算法可求出：

x0\*(c/gcd) + y0\*(2^k/gcd) = 1;因此两边乘以temp/gcd得x\*(c/gcd) + y\*(2^k/gcd) = temp/gcd.

此x就是满足题目（x\*c ≡ temp(mod 2^k)）的解。

代码：

#include <cstdio>

long long extended\_gcd(long long a, long long b, long long &x, long long &y)

{

long long ret, tmp;

if (!b)

{

x = 1, y = 0;

return a;

}

ret = extended\_gcd(b, a % b, x, y);

tmp = x;

x = y;

y = tmp - a / b \* y;

return ret;

}

long long modular\_linear\_equation(long long a, long long b, long long n)

{

long long x, y, e;

long long d = extended\_gcd(a, n, x, y);

if (b % d) return -1;

e = b / d \* x % n + n;

return e % (n / d);

}

int main()

{

long long a, b, c, ans;

int k;

while (scanf("%lld %lld %lld %d", &a, &b, &c, &k) == 4)

{

if (a == 0 && b == 0 && c == 0 && k == 0)

break;

ans = modular\_linear\_equation(c, b - a, 1LL << k);

if (ans == -1)

puts("FOREVER");

else

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

}

return 0;

}

## 兔子的烦恼(一）

<http://acm.nyist.net/JudgeOnline/problem.php?pid=189>

这个题目其实也是很简单的。模拟就能过，但是用些数论的知识会很快的解决掉的。

分析：当n,m的最大公约数大于1，设最大公约数是k,不妨设m=a\*k,n=b\*k,那么第一轮进入的洞是0，m,2m,...(x-1)\*m(不妨假设x\*m>n,(x-1)\*m<n.)。那么下一轮第一个是 x\*m%n, 而x\*m%n=x\*a\*k-b\*k=（x\*a-b）\*k ,即说明下一轮开始的第一个也是k的倍数，而 k>1，所以狼没办法遍历所有的洞，于是兔子躲过一劫！

当n,m的最大公约数等于1，即互质，刚开始的证明同上，而k=1,说明狼可以遍历所有的洞，说得更白话点：只要洞口号是1的倍数，狼就可以进去。于是兔子就成了狼的美味了！

代码：

#include<stdio.h>

int gcd(int y,int x)

{

return x?gcd(x,y%x):y;

}

int main()

{

int m,n;

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

{

if(gcd(m,n)==1)

printf("NO\n");

else

printf("YES\n");

}

}

## 兔子的烦恼(二）

## <http://acm.nyist.net/JudgeOnline/problem.php?pid=317>

这道题目同上面的题目差不多，只是比上面的略微复杂一点，思路是一样的所以直接上代码了：

#include<stdio.h>

int gcd(int y,int x)

{

return x?gcd(x,y%x):y;

}

int main()

{

int m,n,k,i;

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

{

if(gcd(m,n)==1)

printf("NO\n");

else

{

k=gcd(m,n);

printf("%d ",n-(n-1)/k-1);

i=0;

while(i<n)

{

if(i%k)

printf("%d ",i);

++i;

}

printf("\n");

}

}

}

## 10、A problem is easy

<http://acm.nyist.net/JudgeOnline/problem.php?pid=216>

这道题目的本质实际上是要你找出一个数的因子的个数的。题目大致意思是说给你一数字n然后要你找出n可以有多少种表示为i\*j+i+j的形式(0<i<=j)，那么我们对这个问题稍作变换就可以看出，实际上是要找i\*j+i+j+1=(i+1)\*(j+1)=n+1，有多少整数解，那么问题也就转化为了将n+1分解为两个整数的乘积的形式。

对于这样的问题，最简单的思路就是拿1到之间的数去一一试除了。当然可行，但是这里介绍一种更为快捷的办法。

基于数论的一个基本定理：我们知道任何一个大于1的整数n我们都可以写作这样的形式n= 其中为n的素因子。

那么这个n的因子个数就是，利用这个结论这个题目就可以很快算出来了。

代码如下（采用第一种思路）：

#include <stdio.h>

#include <math.h>

int main()

{

int i,j,k,l,m,n;

scanf("%d",&m);

while(m--)

{

scanf("%d",&n);

n=n+1;

l=sqrt(double(n));

for(i=2,k=0;i<=l;i++)

{

if(n%i==0)

k++;

}

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

}

}

采用第二种思路：

#include <stdio.h>

#include <math.h>

#include <string.h>

int prime[100006];

int visit[100006];

int pos=0;

int getprime(int n)

{

memset(visit,0,sizeof(visit));

for(long long i=2;i<n;i++)

{

if(!visit[i])

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

{

visit[j]=1;

}

}

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

{

if(!visit[i])

prime[pos++]=i;

}

return pos;

}

int getfs(int n)

{

int ans=1;

for(int i=0;i<pos && prime[i]\*prime[i]<=n;i++)

{

int d=0;

while(n%prime[i] == 0)

{

d++;

n/=prime[i];

}

ans\*=(d+1);

}

if(n!=1) ans\*=2;

return ans;

}

int main()

{

getprime(100005);

int t;

scanf("%d",&t);

while(t--)

{

int n;

scanf("%d",&n);

printf("%d\n",(getfs(n+1)-2+1)/2);

//printf("%d\n",(getfactionsum(n+1)-2+1)/2);

}

return 0;

}