一.最大公约数（greatest common divisor)

辗转相除法

$$gcd(a,b)=gcd(b,a\%b)$$

证明

设两数为a、b(a>b)，用gcd(a,b)表示a，b的最大公约数，r=a %b为a除以b的余数，k为a除以b的商，即a÷b=k余r。辗转相除法即是要证明gcd(a,b)=gcd(b,r)。

第一步：令c=gcd(a,b)，则设a=mc，b=nc

第二步：根据前提可知r =a-kb=mc-knc=(m-kn)c，可知c也是r的因数

第三步：证明m-kn与n互质(假设m-kn=xd，n=yd (d>1)，则m=kn+xd=kyd+xd=(ky+x)d，则a=mc=(ky+x)cd，b=nc=ycd，则a与b的一个公约数cd>c，故c非a与b的最大公约数，与前面结论矛盾)，因此c也是b与r的最大公约数。

从而可知gcd(b,r)=c，继而gcd(a,b)=gcd(b,r)。

栗子

求32和18最大公约数

$$gcd(32,18)=gcd(18,14)=gcd(14,4)=gcd(4,2)$$

4是2的倍数，所以停止，最大公约数就是2了

代码

非递归

```
int gcd(int a,int b)
{
    while(a%b!=0)
    {
        int temp;
        temp=a;
        a=b;
        b=temp%b;
    }
    return b;
}
```

递归

```
int gcd(int a,int b)
{
     if(b==0)
         return a;
     else
        return gcd(b,a%b);
}
```

简化

```
int gcd(int a,int b)
{
     return b==0?a:gcd(b,a%b);
}
```

水题

给定平面上的两个格点 $P_1 =（x_1, y_1）$ 和 $P_2 = (x_2, y_2)$，线段 P_1P_2 上，除 P_1 和 P_2 以外一共有几个格点?

PS. $-10^9 \leq x_1, x_2, y_1, y_2 \leq 10^9$

二.素数筛

在一个区间内找出所有素数

1）原始

遍历区间内每个数，判断是否为素数

```
for(int i=2;i<max;i++)
{
    num[i]=1;
    for(int j=2;j*j<=i;j++)
    {
        if(i%j==0)
        {
            num[i]=0;
            break;
        }
    }
}
```

时间复杂度是O(n*sqrt(n))，比爪巴还慢

2）埃氏（埃拉托斯特尼）筛

可以考虑去掉每个数的倍数，剩下的就是素数

```
for(int i=2;i*i<=max;i++)
{
    if(!num[i])
    for(int j=i;i*j<=max;j++)
    {
        num[j*i]=1;
    }
}
```

时间复杂度O(nloglogn)

举一反三

110觉得找素数太菜了，他说他能随便背出 a 以内的所有素数，但是他想知道 a 到 b 里有多少个素数，你能告诉他吗?

PS. $a < b \leq 10^{12}$

$b - a \leq 10^6$

3）线性筛——欧拉（Euler）筛

不难发现埃氏筛中重复判断了不少数字，例如12既会被2又会被3标记，在标记2的倍数时，12=6* 2，在标记3的倍数时，12=4* 3，根本原因是没有找到唯一产生12的方式。

原理

每个合数必有一个最大因子（除它本身），可以利用这个因子筛掉它。

换言之，每个合数必有一个最小素因子，利用这个素因子筛掉这个合数。

过程

假设 i 是合数 t 的最大因数，t 显然可能不唯一（例如 30 和 45 最大因数都是 15）。但是仔细想一想，必然有 $t = i * p$ (p 为小于 i 的质数)。

p为什么比 i 小？因为 i 是 t 的最大因数。

为什么 p 一定是质数？因为如果 p 是合数，那么 i 就一定不是 t 的最大因数，因为 p 可以再拆成若干素数相乘，这些素数再与 i 相乘会使因数更大。 既然如此，我们只需要把所有小于 i 的质数 p 都挨个乘一次好了。可是，真相真的是这样的嘛?

其实不是的，一不小心就忘记了最初的条件。我们要满足 i 是 t 的最大因数。如果 p 大于 i 的最小质因数，那 i 还 是 t 的最大因数嘛？显然不是，任何一个合数都能唯一分解为有限个质数的乘积，除去这其中最小的质因数，其他的都乘起来就是最大因数 i 。所以我们不能让 p 大于 i 的最小质因数。

代码

$v[i]$ 表示 i 的最小质因数。如果i就是质数，那么 $v[i] = i$

$prime[j]$ 表示第 j 个质数。与之前的筛法不同，这个数组是存放质数的，而不是标记质数的

```
int primes[MAX],v[MAX],m;//m表示现在筛出了m个素数
for(int i=2;i<=max;i++)
    {
        if(!v[i])
            primes[m++]=i;
        for(int j=0;j<=m;j++)
        {
            if(primes[j]*i>=MAX) break;
            v[primes[j]*i]=1;
            if(i%primes[j]==0) break; //关键！！如果i已经是primes[j]的倍数，那就无法保证primes是其最小质因数
        }
    }
```

时间复杂度O(n)

三.快速幂

在遇到求一个数的多次方时，我们常规的思路就是利用循环累乘，一般问题这样解决没有问题，但在遇到较大次方时就会出现时间过长的问题。

栗如：2^{16} 怎么求？显然直接计算要16次的乘法，然如果你先计算出 2^8 ，再平方，就只需9次，同理，因为 $16=2^4$ ， 所以 $2^{16}=(((2^2)^2)^2)^2$ ，这样只需要4次乘法。

那么再来看看 2^{13} 。

$13=8+4+1$ （想一想为什么这样拆分），所以 $2^{13}=2^8*2^4*2^1$ ，只需要 $3+3=6$ 次

6次是怎么得来的呢，其实我们在算到 2^8 时已经计算过 2^4 和 2^1 了，只用将它们存起来再相乘就可。

如何更直观的得出一个数的次幂如何拆分呢，不难发现这其实就是二进制。

$13=8+4+1=2^3+2^2+2^0$ ，而13的二进制就是1101，我们可以以此推出代码

```
int fast_power(int a, int b)
{
    int ans=1, res=a;
    while(b)
    {
        if(b%2==1) ans=ans*res;
        res=res*res;
        b/=2;
    }
    return ans;
}
```

位运算：

& 位与 1&1=1 1&0=0 0&0=0

>> 右移 相当于除2

```cpp
typedef long long ll;
const ll mod=1e9+7;
ll fast_power(ll a, int b)
{
    ll ans=1, res=a;
    while(b)
    {
        if(b&1) ans=ans*res%mod;
        res=res*res%mod;
        b>>=1;
    }
    return ans%mod;
}
```

时间复杂度O(logn)

ww的幸运数

每个人都有一个幸运数字，但是ww的幸运数字很特殊，对于任意的 $1 < x < n$ 都有 $x^n \equiv x(mod\ n)$ 成立的合数 n 是ww的幸运数字，现在ww给你一个数字，你要判断它是不是幸运数字，说错了ww会很生气的哦

PS. $2 < n < 65000$

四.博弈论

博弈论是二人在平等的对局中各自利用对方的策略变换自己的对抗策略，达到取胜的目的。

Bash博弈

yd和zzy在玩一个AK比赛的博弈，有总数为 N 的题目 ，每个人轮流交题，一次能交 1~ M 道题目，由于他们实力强劲，交了必AC，并且不能交对方AC过的题目，两人都聪明绝顶，最终谁能AK比赛呢?

若 N % （M+1）！ = 0 , 则先手胜；只要先手开始交题，让剩下的题数一直为(M+1)的整倍数，为先手的必胜局面； 若N % （M+1) == 0 ， 则后手胜；同理的，只要开始先手打破平衡，先交题，后手再交，让剩下的题数一直为（N+1）的整数倍，则为后手的必胜局面。

Nim博弈

现在yd和zzy厌倦了AK一场比赛，他们现在准备比赛谁能AK所有区域赛，每场比赛的题数都是有限的，他们每次可以AK任意一场比赛或者AC任一比赛的任意道题（跨越时间地点的AK），如果轮到某个人时所有的比赛都已经被AK了，则他就要女装（因为他此刻没有题做了），你能计算出谁女装吗?

显然，只剩一场的时候必然先手胜利

当有两场相同题数的比赛，无论先手是什么操作，后手在另一场比赛执行相同操作，就可以使后手必然胜利

推论： 1）当a1 ^ a2 ^ a3 ^ ..... ^ an = 0时，为P-position，即先手必败。

2）当a1^ a2 ^ a3^ …… ^ an! = 0时，一定存在某个移动，使得局面变成一个P-position，即此局面为N-position。

所以我们只用将每场比赛的题目数异或就能得到答案，代码不做赘述。