第三章　数学问题

第1节　简单数论

在20世纪初之前，数论被称为算术，之后人们才开始使用数论的名称。数论主要研究整数的性质，正整数分为素数、合数和1。素数产生了很多人们能理解，但是又难以证明的问题，比如孪生素数、哥德巴赫猜想等，在解决这些问题的过程中产生了很多新的数学思想和方法，推动了数学学科的发展。

算数基本定理（也叫唯一分解定理），即任意一个大于1的自然数都可分解成若干素数的乘积的形式就是数论的内容，那什么是素数，素数又有哪些性质，我们怎样得到素数呢？带着这些问题，让我们开始学习本节的内容。

一、整数性质

（一）（带余除法，即欧几里德除法）设 a,b 为整数，$b\neq 0$，则存在整数 q 和 r，使得 $a=bq+r$，其中 $0\leqslant r<|b|$，并且 q 和 r 由上述条件唯一确定；整数 q 被称为 a 被 b 除得的（不完全）商，数 r 称为 a 被 b 除得的余数。

注意：r 共有 $|b|$ 种可能的取值：$0,1,\cdots,|b|-1$。若 $r=0$，即为 a 被 b 整除的情形，因此，证明 $b|a$ 的基本手法是将 a 分解为 b 与一个整数之积。

带余除法的核心是关于余数 r 的不等式：$0\leqslant r<|b|$。

（二）任何一个正整数 n，都可以写成 $n=2^m l$ 的形式，其中 m 为非负整数，l 为奇数。

（三）若 $a\in Z,a>1$，则 a 的除1以外的最小正因数 q 是一个质（素）数．如果 $q\neq a$，则 $q\leqslant\sqrt{a}$。

推论：如果不超过 $\sqrt{a}$ 的所有质数均不是 a 的约数，则 a 必为质数。

（四）算术基本定理（素数唯一分解定理）任何一个大于1的正整数 a，能唯一地表示成质（素）因数的乘积（不计较因数的排列顺序）；即

任何大于1的整数 a 能唯一地写成下面

$$a=p_1^{a_1}p_2^{a_2}\cdots p_k^{a_k} \tag{1}$$

的形式，其中 p_i 为素数（$p_i<p_j(i<j)$），$a_i\in N_+,i=1,2,\cdots,k$。

上式叫作整数 a 的标准分解式。

推论1：若 a 的标准分解式是(1)式，则 d 是 a 的正因数的充要条件是：

$$d=p_1^{\beta_1}p_2^{\beta_2}\cdots p_k^{\beta_k},0\leqslant\beta_i\leqslant a_i,i=1,2,\cdots,k。 \tag{2}$$

应注意(2)不能称为是 d 的标准分解式，其原因是其中的某些 β_i 可能取零值（当 d 不含某个素因数 p_i 时，$\beta_i=0$）。

推论2：设 $a=bc$，且 $(b,c)=1$，若 a 是整数的 $k(k\geqslant 2)$ 次方，则 b,c 也是整数的 k 次方。特别地，若 a 是整数的平方，则 b,c 也是整数的平方。

一般地，设正整数 $a,b,\cdots,c$ 之积是一个正整数的 k 次方幂（$k\geqslant 2$），若 $a,b,\cdots,c$ 两两互素，则 $a,b,\cdots,c$ 都是正整数的 k 次方幂。

二、整除

(一)常见数字的整除判定方法

1.一个数的末位能被2或5整除,这个数就能被2或5整除;一个数的末两位能被4或25整除,这个数就能被4或25整除;一个数的末三位能被8或125整除,这个数就能被8或125整除。

2.一个数各位数字和能被3整除,这个数就能被3整除;一个数各位数字和能被9整除,这个数就能被9整除。

3.如果一个整数的奇数位上的数字之和与偶数位上的数字之和的差能被11整除,那么这个数能被11整除。

4.如果一个整数的末三位与末三位以前的数字组成的数之差能被7、11或13整除,那么这个数能被7、11或13整除。

5.如果一个数能被99整除,这个数从后两位开始两位一截所得的所有数(如果有偶数位,则拆出的数都是两位数;如果是奇数位,则拆出的数中有若干个两位数,还有一个是一位数)的和是99的倍数,这个数一定是99的倍数。

(二)整除的性质

1.性质1　如果数a和数b都能被数c整除,那么它们的和或差也能被c整除。即如果c|a,c|b,那么c|(a±b)。

2.性质2　如果数a能被数b整除,b又能被数c整除,那么a也能被c整除。即如果b|a,c|b.那么c|a。

3.性质3　如果数a能被数b与数c的积整除,那么a也能被b或c整除。即如果bc|a,那么b|a或c|a。

4.性质4　如果数a能被数b整除,也能被数c整除,且数b和数c互质,那么a一定能被b与c的乘积整除。即如果b|a,c|a,且(b,c)=1,那么bc|a。

例如:如果3|12,4|12,且(3,4)=1,那么(3×4)|12。

5.性质5　如果数a能被数b整除,那么am也能被bm整除。如果b|a,那么bm|am(m为非0整数)。

6.性质6　如果数a能被数b整除,且数c能被数d整除,那么ac也能被bd整除。如果b|a,且d|c,那么bd|ac

三、余数

(一)余数三大余数定理

1.余数的加法定理。

a与b的和除以c的余数,等于a、b分别除以c的余数之和。

例如:23、16除以5的余数分别是3和1,所以23+16=39除以5的余数等于4,即两个余数的和为(3+1)。

当余数的和比除数大时,所求的余数等于余数之和再除以c的余数。

例如:23、19除以5的余数分别是3和4,所以23+19=42除以5的余数等于3+4=7除以5的余数,即为2。

2.余数的减法定理。

a与b的差除以c的余数,等于a、b分别除以c的余数之差。

例如：23、16 除以 5 的余数分别是 3 和 1，所以 23－16＝7 除以 5 的余数等于 2，即两个余数差 3－1＝2。

当余数的差不够减时，补上除数再减。

例如：23、14 除以 5 的余数分别是 3 和 4，23－14＝9 除以 5 的余数等于 4，即两个余数差为(3＋5)－4＝4.

3.余数的乘法定理。

a 与 b 的乘积除以 c 的余数，等于 a、b 分别除以 c 的余数的积。

例如：23、16 除以 5 的余数分别是 3 和 1，所以 23×16 除以 5 的余数等于 3×1＝3。

当余数的积比除数大时，所求的余数等于余数之积再除以 c 的余数。

例如：23、19 除以 5 的余数分别是 3 和 4，所以 23×19 除以 5 的余数等于 3×4 除以 5 的余数，即 2。

乘方：如果 a 与 b 除以 m 的余数相同，那么 a^n 与 b^n 除以 m 的余数也相同。

(二)同余定理

1.同余的定义

若两个整数 a、b 被自然数 m 除有相同的余数，那么称 a、b 对于模 m 同余，用式子表示为：a≡b(mod m)左边的式子叫作同余式(读作：a 同余于 b，模 m)。

2.同余的重要性质及推论

若两个数 a，b 除以同一个数 m 得到的余数相同，则 a，b 的差定能被 m 整除。

例如：17 与 11 除以 3 的余数都是 2，所以 17－11 能被 3 整除。用式子表示为：如果有 a≡b(mod m)，那么一定有 a－b＝mk，k 是整数，即 m|(a－b)。

3.余数判别法

当一个数不能被另一个数整除时，虽然可以用长除法求得余数，但当被除位数较多时，计算是很麻烦的。建立余数判别法的基本思想是：为了求出“N 被 m 除的余数”，我们希望找到一个较简单的数 R，使得 N 与 R 对于除数 m 同余.由于 R 是一个较简单的数，所以可以通过计算 R 被 m 除的余数来求得 N 被 m 除的余数。

(1)整数 N 被 2 或 5 除的余数等于 N 的个位数被 2 或 5 除的余数；

(2)整数 N 被 4 或 25 除的余数等于 N 的末两位数被 4 或 25 除的余数；

(3)整数 N 被 8 或 125 除的余数等于 N 的末三位数被 8 或 125 除的余数；

(4)整数 N 被 3 或 9 除的余数等于其各位数字之和被 3 或 9 除的余数；

(5)整数 N 被 11 除的余数等于 N 的奇数位数之和与偶数位数之和的差被 11 除的余数(不够减的话先适当加 11 的倍数再减)；

(6)整数 N 被 7、11 或 13 除的余数等于先将整数 N 从个位起按右往左每三位分节，奇数节的数之和与偶数节的数之和的差被 7、11 或 13 除的余数就是原数被 7、11 或 13 除的余数。

四、素数与约数

素数是指在大于 1 的自然数中，除了 1 和它本身以外不再有其他因数的自然数。

(一)素数的判定

1.方法一：对于一个数 N，可以从 2 枚举到 N－1，从而判断这个数是不是素数，时间复杂度为 O(N)。

```
bool ck_prime (int n) {
```

```
    if (n==1)  return false;
    if (n==2)  return true;
    for (int i=2;i<n;i++)
        if (n%i==0)
          return false;
    return true;
}
```

2.方法二：不难发现 N 的因数是成对出现的，所以对于任何一个整数 N，只需要从 1 枚举到$\sqrt{N}$，时间复杂度为 O($\sqrt{N}$)。

```
bool ck_prime (int n) {
if (n==1)     return false;
if (n==2)     return true;
for (int i=2;i*i<=n;i++)//或者 i<=sqrt(n)
    if (n% i==0)
        return false;
return true;
}
```

3.方法三：Miller-Rabin 素数判定

有时候我们想快速的知道一个数是不是素数，而这个数又特别的大，导致 O($\sqrt{N}$)的算法不能通过，这时我们可以对其进行 Miller-Rabin 素数测试，来判断其是否为素数。

需要说明的两个基本定理：

费马小定理：当 P 为素数，有 $a^{P-1}\equiv 1 \bmod P$，反过来不一定成立，也就是说，如果 a，P 互质，且 $a^{P-1}\equiv 1 \bmod P$，不能推出 P 是素数。

二次探测：如果 P 是一个素数，$0<x<P$，则方程 $x^2\equiv 1 \bmod P$ 的解为 x=1 或 x=P−1。

具体实现代码如下：

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef long double ld;
inline ll mul(ll a,ll b,ll p){//快速乘板子，防止 a*b 爆 long long
    ll ans=(ld)a*b/p;
    return((a*b-ans*p)% p+p)% p;
}
inline ll power(ll a,ll b,ll p){//快速幂板子
    ll ans=1;
    for (;b;b> > =1){
        if (b&1)ans=mul(ans,a,p);
        a=mul(a,a,p);
    }
    return ans;
}
ll pri[10]={2,3,5,7,11,13,17,19,23}; //选取的质数，选取九个正确率已经很高
```

```
inline bool judge(ll x){                    //判断 x 是否是质数
    for (register int i=0;i<=8;i++) //选取质数
    {
        if (pri[i]==x)return true;  //如果是一个质数
        else if (pri[i]> x)return false;//如果是一个合数
        ll cnt=power(pri[i],x-1,x);
        if (cnt!=1)return false;//根据费马小定理,如果 x 是素数,则 pri[i]
的 x-1 次方在模 x 意义下和 1 同余
        else {//接着用二次探测定理检验//有二次探测定理,得知在 x 为质数时,关
于 a 的方程  a 的 p-1 次方在模 x 意义下与一同余的 a 只有唯二解分别为 1 和 p-1
        //如果方程的解不是 x-1 或 1,那么 x 就不是质数
        //而通过上面费马小定理得到则 pri[i]的 x-1 次方在模 x 意义下和 1 同余
        //如果 x-1 是偶数,可以 pri[i]的 x-1 次方变为 pri[i]的(x-1)/2 二次方
        //这样如果 pri[i]的(x-1)/2 与 1 在模 x 意义下同余,则 x 不是质数
            ll now=x-1;
            while (now% 2==0)
            {//如果目前的次幂仍然是二的倍数,可以模仿之前的操作,接着检验
              now/=2;
              cnt=power(pri[i],now,x);
              if (cnt!=1&&cnt!=x-1)
                return false;//cnt 不为 1,方程的解不是 x-1,则 x 不是质数
            }
        }
    }
    return true;//多次无法判错,则 x 几乎一定是质数
}
int main(){
    ll n;
    scanf("%d",&n);
    if (judge(n))cout<<"Yes"<<endl;
    else cout<<"No"<<endl;
    return 0;
}
```

(二)素数的筛选

求 1～n 之间的所有素数,称为素数的筛选问题。

筛选素数的方法有很多种,一般都是基于“素数的倍数一定不是素数”的思想。公元前 250 年古希腊数学家埃拉托斯特尼发明了一种寻找素数的方法,我们称之为“埃拉托斯特尼筛法”。

1.方法一:埃拉托斯特尼筛法/Eratosthenes 筛选,也叫埃氏筛法。

基本思想:素数的倍数一定不是素数。

实现方法:用一个长度为 N+1 的数组保存信息(0 表示素数,1 表示非素数),先假设所有的数都是素数(初始化为 0),从小到大枚举每一个素数 x,把 x 的倍数都标记为非素数(置

为 1)。实际上小于 x^2 的 x 的倍数在扫描更小的数时就已经被标记过,因此可对埃筛法优化,对于每一个 x,把大于等于 x^2 的 x 的倍数标记为合数即可,时间复杂度为 O(nloglogn)。

```
void primes(int n){
    memset(vis,0,sizeof(vis));
    vis[1]=1;              //1 不是素数标记为 1
    for (int i=2;i<=n;i++){
        if (vis [i])      //vis [i]=1 代表 i 不是素数
            continue;
        cout<<i<<endl;//否则输出素数 i
        for (int j=i;j<=n/i;j++)
            vis[i*j]=1; //大于等于 i²的 i 的倍数标记为合数
    }
}
```

2.方法二:线性筛法(欧拉筛)

通过上述代码,发现此方法还可以继续优化,因为上述的方法中,每一个数有多组因数可能会被筛多次,例如:30 会被 2,3,5 各筛一次导致计算冗余,可以对此方法进行改进,得到更加高效的筛法。

基本思想:每个合数只被它的最小的素因子筛一次。

实现方法:数组 v 记录每个数的最小素因子,每个合数只会被它的最小的素因子筛一次,时间复杂度为 O(n)。

```
int v[MAXN],prime[MAXN];
void primes(int n){
memset(v,0,sizeof(v));  //数组 v 保存最小素因子 m=0;//素数数量
for (int i=2;i<=n;i++){
    if (v [i]==0){   //i 是素数
        v [i]=i;     //素数 i 的最小素因子是其本身
        prime[++m]=i;
    }
    for (int j=1;j<=m;j++){      //给当前的数 i 乘一个素因子
        if (prime[j]> v [i]||prime[j]> n/i)
            break;          //i 有比 prime[j]更小的素因子或者超出 n 的范围
        v[i*prime[j]]=prime[j];//prime[j]是合数 i*prime[j]的最小素因子
    }
}
for (int i=1;i<=m;i++)
    cout<<prime [i]<<endl;     //输出
}
```

(三)素数的性质

1.素数 p 的约数只有两个:1 和 p。

2.初等数学基本定理:任一大于 1 的自然数,要么本身是素数,要么可以分解为几个素数之积,且这种分解是唯一的。

3.素数的个数是无限的。

4.素数的个数公式 $\pi(x)$ 是不减函数。

5.若 n 为正整数，在 n^3 到 $(n+1)^3$ 之间至少有一个素数。

6.若 n 为大于或等于 2 的正整数，在 n 到 n! 之间至少有一个素数。

7.若素数 p 为不超过 $n(n\geqslant 4)$ 的最大素数，则 $p>\frac{n}{2}$。

(四)质因数分解

把一个合数分解成若干个质因数乘积的形式(即求质因数的过程)叫做分解质因数(也叫分解素因数)。

对 n 分解质因数为质因数的乘积，要从最小的素数开始，我们从 2 开始试除，能整除，就输出 2，再对商继续试除，直到把包含 2 的因子"除干净"；再用下一个素数试除，然后再下一个素数试除……直到商为 1 结束。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int n,i=2;
    scanf("%d",&n);
    cout<<n<<'=';
    do {
        while (n% i==0){
            cout<<i;
            n=n/i;
            if (n!=1)cout<<'*';
        }
        i++;
    }
    while (n!=1);//n 没有除尽，继续操作
  return 0;
}
```

(五)约数的判定

若 $d\geqslant\sqrt{N}$ 是 N 的约数，则 $N/d\leqslant\sqrt{N}$ 也是 N 的约数，也就是说约数总数成对出现。因此，只需扫描 $d=1\sim\sqrt{N}$，看 d 能否整除 N，若能整除，则 N/d 也是 N 的约数，算法时间复杂度为 $O(\sqrt{N})$。

```
#include<bits/stdc++.h>
using namespace std;
int p[100100];
void deal(int n){
        int m=0;
        for (int i=1;i*i<=n;i++){
                if (n% i==0){
```

```
            p[++m]=i;
            if (i!=n/i)p[++m]=n/i;
        }
    }
    sort (p+1,p+1+m);
    for (int i=1;i<=m;i++)cout<<p[i]<<'  ';
}
int main(){
    int x;
    cin>>x;
    deal(x);
    return 0;
}
```

五、最大公约数与最小公倍数

(一)最大公约数:设整数 a,b,…,c 不全为零,同时整除 a,b,…,c 的整数(如±1)称为它们的公约数。因为 a,b,…,c 不全为零,故同时整除 a,b,…,c 的整数只有有限多个,其中最大的一个称为 a,b,…,c 的最大公约数,用符号(a,b,…,c)表示。

当(a,b,…,c)=1(即 a,b,…,c 的公约数只有±1)时,称 a,b,…,c 互素(互质)。注意,(如涉及三个及三个以上的整数)此时不能推出 a,b,…,c 两两互素;但反过来,若 a,b,…,c 两两互素,则显然有(a,b,…,c)=1。

(二)最小公倍数:设 a,b,…,c 均是非零整数,一个同时为 a,b,…,c 倍数的数称为它们的公倍数,a,b,…,c 的公倍数有无穷多个,其中最小的一个正整数称为 a,b,…,c 的最小公倍数,记作[a,b,…,c]。

(三)由最大公约数、最小公倍数的定义,不难得出它们的一些简单性质:

1.a,b 的任何一个公约数都是它们最大公约数的约数,即若 m|a,m|b,则 m|(a,b);

2.a,b 的任何一个公倍数都是它们最小公倍数的倍数,即若 a|m,b|m,则[a,b]|m;

3.若 $b\in N_+$,则(0,b)=b,[1,b]=b;

4.对于任意的整数 x,有(a,b)=(a,b+ax);

5.(辗转相除法)设 $a,b\in Z,b\neq 0$,由带余除法知 $a=bq+r,0\leqslant r<|b|,q,r\in Z$。

故(a,b)=(b,r)。

设 $a>b>0$,作带余除法如下:

$a=bq_1+r_1,0\leqslant r_1<b,q_1,r_1\in Z$;

$b=r_1q_2+r_2,0\leqslant r_2<r_1,q_2,r_2\in Z$;

……

由于 $b>r_1>r_2>\cdots$,故必存在 $n\in N_+$,使得 $r_n=0$,而 $r_{n-1}\neq 0$(约定 $r_0=b$).

则有 $(a,b)=(b,r_1)=(r_1,r_2)=\cdots=(r_{n-1},r_n)=r_{n-1}$。

由此不仅得到 a,b 的最大公约数,而且将上述过程反过来,即得:

6.裴蜀定理:设 a,b 是不全为 0 的整数,则存在整数 x,y,使得 ax+by=(a,b);

(其中 x,y 不唯一。如加上等式 ab−ab=0,有 a(b+x)+b(y−a)=1,即裴蜀等式除通

过辗转相除得到外，也可通过“凑”、恒等变形等其他方法得到）。

特别是，两个整数 a，b 互素的充要条件是存在整数 x，y，使得 $ax+by=1$。

事实上，条件的必要性是裴蜀定理的一个特例。反过来，若有 x，y 使等式成立，不妨设 $(a,b)=d$，则 $d|a$，$d|b$，故 $d|ax$ 及 $d|by$，于是 $d|(ax+by)$，即 $d|1$，从而 $d=1$.

7.设两个正整数 a，b 的分解式分别为

$a=p_1^{\alpha_1}p_2^{\alpha_2}\cdots p_k^{\alpha_k}$，$\alpha_i\geqslant 0$，$i=1,2,,\cdots,k$；

$b=p_1^{\beta_1}p_2^{\beta_2}\cdots p_k^{\beta_k}$，$\beta_i\geqslant 0$，$i=1,2,,\cdots,k$；

取 $\gamma_i=\min\{\alpha_i,\beta_i\}$，$\delta_i=\max\{\alpha_i,\beta_i\}$，

则 $(a,b)=p_1^{\gamma_1}p_2^{\gamma_2}\cdots p_k^{\gamma_k}$，$[a,b]=p_1^{\delta_1}p_2^{\delta_2}\cdots p_k^{\delta_k}$。

注：此结论对多个正整数 $a_1,a_2,\cdots,a_n$ 仍成立。

8.两个整数的最大公约数与最小公倍数满足：$(a,b)\cdot[a,b]=|ab|$；

（但注意，这只限于两个整数的情形，对于多于两个整数的情形，类似结论不成立）。

（四）最大公约数和最小公倍数的求法

假设 x 和 y 的最大公约数是 m，最小公倍数是 n，由性质(8)可知，$xy=mn$。故只需求出最大公约数即可求得最小公倍数。

1.方法一：枚举法

对两个整数 a 和 b，共同的约数在 1 和 $\min(a,b)$ 之间，从大到小枚举，若判断它能同时整除 a 和 b，即为两个数的最大公约数。该算法的最坏时间复杂度为 $O(\min(a,b))$。

```
#include<bits/stdc++.h>
using namespace std;
int a,b;
int main(){
    cin>>a>>b;
    for (int i=min(a,b);i>0;i--){
        if (a% i==0&&b% i==0){
            cout<<i<<''<<a*b/i;
            break;
        }
    }
    return 0;
}
```

2.方法二：更相减损算法

若 c 是 a 和 b 的公约数（$a>b$），c 亦是 b 和 $a-b$ 的公约数，可以表示成 $\gcd(a,b)=\gcd(b,a-b)$。采用函数进行迭代，当迭代到 $a=b$ 时，a 为 a 和 b 的最大公约数。该算法时间复杂度为 $O(\max(a,b))$。

(1)具体代码递归实现如下：

```
#include<bits/stdc++.h>
using namespace std;
int gcd (int a,int b){
    if (a==b) return a;
```

```
    if (a<b) {
        swap (a,b);
    }
    return gcd(b,a-b);
}
int a,b;
int main(){
    cin>>a>>b;
    cout<<gcd(a,b)<<''<<a*b/gcd(a,b)<<endl;
    return 0;
}
```

(2)具体代码非递归实现如下：

```
#include<bits/stdc++.h>
using namespace std;
int gcd (int a,int b){//非递归
    while (a!=b){
        if (a>b)a=a-b;
        else b=b-a;
    }
    return a;
}
int a,b;
int main(){
    cin>>a>>b;
    cout<<gcd (a,b)<<''<<a*b/gcd (a,b)<<endl;
    return 0;
}
```

3.方法三：辗转相除法(欧几里得算法)

可以表示成 gcd(a,b)==gcd(b,a%b)(a>b)。因此可以采用函数进行迭代(即递归)，当迭代到 b==0 时，a 为 a 和 b 的最大公约数。若 b<=a/2，gcd(a,b)变为 gcd(b,a%b)，规模至少缩小一半，若 b>a/2，a%b 也将规模最少缩小为一半。即经过一次迭代，gcd(a,b)数据规模至少变为原来的一半，所以该算法的时间复杂度为 $O(\log_2(\max(a,b)))$。

输入 a,b，输出它们的最大公约数和最小公倍数代码如下：

```
#include<bits/stdc++.h>
using namespace std;
int a,b;
int gcd (int a,int b){
    if (b==0)
    return a;
    else return gcd(b,a% b);
}
int main(){
```

```
    cin>>a>>b;
    cout<<gcd(a,b)<<''<<a*b/gcd(a,b);
//为了防止 a*b 爆 long long,经常写成 a/gcd(a,b)*b
    return 0;
}
```

课 堂 练 目

1.【NOIP2013】下面是根据欧几里得算法编写的函数，它所计算的是 a 和 b 的（ ）。

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

A.最大公共素因子 B.最小公共素因子 C.最大公约数 D.最小公倍数

2.【NOIP2018】10000 以内，与 10000 互质的正整数有（）个。

A.2000 B.4000 C.6000D.8000

3.【NOIP2018】从 1 到 2018 这 2018 个数中，共有（ ）个包含数字 8 的数。