约数

1 什么是约数

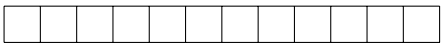
如果正整数 n 能被正整数 d 整除，就称 d 是 n 的约数，也叫因数或正因子。

例如，12 的所有正约数为

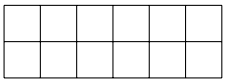
$$1,\ 2,\ 3,\ 4,\ 6,\ 12.$$

这几个数都可以整除 12。而 $12 \div 5$ 的余数不为 0，所以 5 不是 12 的约数。

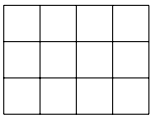
从图形上看，12 个小方格可以排成下面几种长方形：



1×12



2×6



3×4

每一种排法的行数与列数，都是 12 的约数。交换行和列，仍然对应同一对约数。

1.1 代码中的判断

```
if(n%d==0){
    // d是n的约数
}
```

本讲义只讨论正整数的正约数，除数不能为 0。

2 约数的基本性质

- 1 是每个正整数的约数。
- 每个正整数都是它自己的约数。
- 正整数 n 的约数都在 1 到 n 之间。
- 如果 d 是 n 的约数，那么 n/d 也是 n 的约数。

例如，3 是 12 的约数，而 $12/3 = 4$，因此 4 也是 12 的约数。

2.1 特殊的数 1

1 只有一个正约数，就是 1 自己。这里“1 和它本身”是同一个数，不能计算两次。

2.2 质数与合数

大于 1 的正整数，如果只有 1 和它本身两个正约数，就称为质数。例如 $2, 3, 5, 7$。

大于 1 的正整数，如果除了 1 和它本身，还有其他正约数，就称为合数。例如 $4, 6, 8, 9$。

1 既不是质数，也不是合数。

3 枚举约数

从 1 到 n 逐个判断，能整除 n 的数就输出。

```
for(int i=1;i<=n;i++){
    if(n%i==0){
        cout<<i<<"\n";
    }
}
```

由于 i 从小到大枚举，输出的约数也从小到大排列。

这种方法需要判断 n 次，时间复杂度为 $O(n)$，不需要额外数组。数据较小时，直接枚举即可。

4 约数成对出现

以 36 为例：

$$\boxed{1} \times \boxed{36} = 36$$

$$\boxed{2} \times \boxed{18} = 36$$

$$\boxed{3} \times \boxed{12} = 36$$

$$\boxed{4} \times \boxed{9} = 36$$

$$\boxed{6} \times \boxed{6} = 36$$

对于一对约数，较小的数不超过 $\sqrt{n}$。否则两个数都大于 $\sqrt{n}$，它们的乘积就会超过 n。

因此，只需枚举不超过 $\sqrt{n}$ 的数。找到约数 i 后，另一半就是 n/i。

```
for(int i=1;i<=n/i;i++){
    if(n%i==0){
        int j=n/i;
        // 找到一对约数i和j
    }
}
```

`i<=n/i` 与这里的 `i*i<=n` 判断的是同一个范围，但前者避免了 `i*i` 可能产生的乘法溢出。

枚举次数由 n 次减少到约 $\sqrt{n}$ 次，时间复杂度为 $O(\sqrt{n})$。例如 $n = 10^8$ 时，只需枚举到 10000。

5 约数的个数

找到一对不同的约数，数量增加 2。如果这一对相等，则只增加 1。

```
int cnt=0;
for(int i=1;i<=n/i;i++){
    if(n%i==0){
        int j=n/i;
        if(i==j) cnt++;
        else cnt+=2;
    }
}
```

5.1 完全平方数

如果一个正整数可以写成某个整数的平方，就称为完全平方数，例如 $1, 4, 9, 16, 25, 36$。

对于 36，最后一对约数是 6 和 6。它们是同一个约数，所以只算一次。

一对约数	这次增加	累计个数
$1, 36$	2	2
$2, 18$	2	4
$3, 12$	2	6
$4, 9$	2	8
$6, 6$	1	9

不是完全平方数时，所有约数都能两两配对，约数个数为偶数。是完全平方数时，平方根这个约数单独出现，约数个数为奇数。

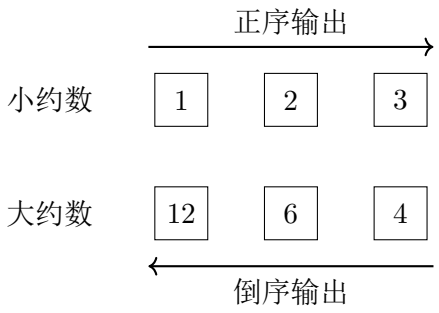
所以，**一个正整数的正约数个数为奇数，当且仅当它是完全平方数**。1 也符合这个结论。

6 从小到大输出约数

成对枚举时，发现约数的顺序不一定是从小到大。例如对 12 直接输出每一对，会得到

$$1,\ 12,\ 2,\ 6,\ 3,\ 4.$$

可以用两个普通数组分别保存每一对中较小、较大的约数。



小约数依次增大，大约数依次减小。所以先正序输出小约数，再倒序输出大约数，不必排序。

下面的数组大小适用于 $1 \le n \le 10^9$。每个数组最多记录 $\lfloor \sqrt{n} \rfloor$ 个数。

```
const int N=100010;
int small[N],large[N];
int c1=0,c2=0;
for(int i=1;i<=n/i;i++){
    if(n%i==0){
        small[++c1]=i;
        if(i!=n/i) large[++c2]=n/i;
    }
}
for(int i=1;i<=c1;i++) cout<<small[i]<<"\n";
for(int i=c2;i>=1;i--) cout<<large[i]<<"\n";
```

当两个约数相等时，只存入 small，防止重复输出。

7 约数的和

求和时，将找到的约数加入答案。成对枚举仍然需要判断两者是否相等。

```
long long sum=0;
for(int i=1;i<=n/i;i++){
    if(n%i==0){
        sum+=i;
        if(i!=n/i) sum+=n/i;
    }
}
```

例如，12 的所有正约数之和为

$$1+2+3+4+6+12=28.$$

约数相加的结果可能超过原来的数，模板使用 `long long` 保存总和。

7.1 真因数

真因数是一个正整数除它本身以外的正约数。例如，12 的真因数为

$$1,\ 2,\ 3,\ 4,\ 6.$$

所有正约数中，n 本身恰好出现一次。因此可以先计算全部正约数之和，再减去 n。

```
long long properSum=sum-n;
```

1 没有正的真因数，它的真因数之和为 0。

7.2 约数个数与约数和的区别

对象	约数个数	全部正约数的和
1	1	1
6	4	12
12	6	28

计数是每找到一个约数增加 1，求和是把约数的值加进去。两种操作使用相同的枚举范围。

8 完全数

如果一个正整数的所有真因数之和等于它本身，就称这个数为完全数。

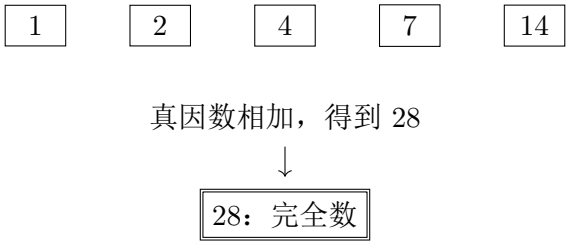
例如，6 的真因数为 $1, 2, 3$，而

$$1 + 2 + 3 = 6,$$

所以 6 是完全数。28 的真因数为 $1, 2, 4, 7, 14$，它们的和为

$$1 + 2 + 4 + 7 + 14 = 28,$$

所以 28 也是完全数。



8.1 完全数的判断

先计算全部正约数的和，再减去自身，与原数比较。

```
bool isPerfect(int n){
    long long sum=0;
    for(int i=1;i<=n/i;i++){
        if(n%i==0){
            sum+=i;
            if(i!=n/i) sum+=n/i;
        }
    }
    return sum-n==n;
}
```

1 的真因数之和为 0，不等于 1，所以 1 不是完全数。

完全平方数和完全数是两种不同的概念。前者看能否写成整数的平方，后者看真因数之和是否等于自身。

9 基础模板

下面给出约数个数与约数和的函数写法，均适用于正整数。函数只计算传入的一个数，不涉及具体题目的输入输出。

9.1 求约数个数

```
int divisorCount(int n){
    int cnt=0;
    for(int i=1;i<=n/i;i++){
        if(n%i==0){
            cnt++;
            if(i!=n/i) cnt++;
        }
    }
    return cnt;
}
```

9.2 求全部正约数的和

```
long long divisorSum(int n){
    long long sum=0;
    for(int i=1;i<=n/i;i++){
        if(n%i==0){
            sum+=i;
            if(i!=n/i) sum+=n/i;
        }
    }
    return sum;
}
```

两个函数都是每发现一对约数就处理一次，时间复杂度为 $O(\sqrt{n})$，额外空间为 $O(1)$。如果需要保存并输出约数，再使用前面的普通数组模板。

10 配套训练

算法竞赛入门指南：第四章“约数”

题号	题目
P1841	找因数
P1673	因子个数
P1320	求正整数 2 和 n 之间的完全数
B10022	八个因数的奇数