

### 三、初等数论 (Elementary Number Theory)

#### 【大纲内容】

- 【3】整除、因数、倍数、指数、质(素)数、合数
- 【3】取整
- 【3】模运算与取余
- 【3】整数唯一分解定理
- 【3】辗转相除法(欧几里得算法)
- 【4】素数筛法：埃氏筛法与线性筛法

#### (十) 辗转相除法 (Euclidean Algorithm)

辗转相除法是一种用来求两个整数**最大公约数** (Greatest Common Divisor, GCD) 的快速算法。

- 最大公约数是能**同时整除两个数的最大整数**。
- 辗转相除法通过不断用较大数除以较小数，再用除数和余数替换，直到余数为 0。
- 最后得到的除数就是最大公约数。

#### 举例说明

- 想象你有两堆糖果，要找出最大的能同时分给两堆的“组数”，比如你有 12 和 18 颗糖，最大的共同分组数是 6 颗。
- 算法像玩游戏，一直用除法把数字变小，直到不能再分了。

**举例：**求 12 和 18 的最大公约数：  $78 \div 66 = 1 \text{ remainder } 12$  ( $78 = 66 \times 1 + 12$ )

- $18 \div 12 = 1$  余 6
- $12 \div 6 = 2$  余 0
- 余数为 0，最大公约数是 6。

#### C++示例代码：计算最大公约数

```
#include <iostream>
using namespace std;
```

Find GCF or GCD using the Euclidean Algorithm

Example:

Find GCD of 12 and 30  
 $30 \div 12 = 2 \text{ remainder } 6$   
 $12 \div 6 = 2 \text{ remainder } 0$   
**GCD**  
 The GCD of 12 and 30 is 6

Find GCD of 123 and 36  
 $123 \div 36 = 3 \text{ remainder } 15$   
 $36 \div 15 = 2 \text{ remainder } 6$   
 $15 \div 6 = 2 \text{ remainder } 3$   
 $6 \div 3 = 2 \text{ remainder } 0$   
**GCD**  
 The GCD of 123 and 36 is 3

```

int gcd(int a, int b) {
    while (b != 0) {
        int r = a % b; // 求余数
        a = b;
        b = r;
    }
    return a;
}

int main() {
    int x = 12, y = 18;
    cout << x << " 和 " << y << " 的最大公约数是: " << gcd(x, y)
    << endl;
    return 0;
}

```

### 考试注意事项 & 易错点

易错点	正确说明
余数为 0 时未及时停止	余数为 0 时，当前除数即最大公约数
参数顺序写反	调用时顺序无影响，算法内部会处理
误把最大公约数当最大公倍数	最大公约数是“共同的最大因数”，非倍数

## (十一) 素数筛法 (Prime Sieve Methods)

### 什么是素数筛法? (What is Prime Sieve?)

素数筛法是用来**快速找出某个范围内所有质数**的方法。

#### 1、埃氏筛法 (Sieve of Eratosthenes)

埃氏筛法是一种古老且简单的找质数的方法：

- 从 2 开始，把 2 的倍数都标记为非质数。

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

- 再找到下一个没标记的数 3，把 **3 的倍数**都标记。
- 重复这个过程直到处理完范围内的数。
- 没标记的数就是质数。

### 举例说明

- 想象有一大堆数字卡片，从 2 开始，先把所有 2 的倍数涂红（剔除），
- 接着找到下一个没涂红的数字（3），把它的倍数也涂红，
- 依次这样做，剩下的数字就是质数。

### C++示例代码（埃氏筛法）：

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n = 30;

    vector<bool> isPrime(n + 1, true);

    isPrime[0] = isPrime[1] = false;

    for (int i = 2; i * i <= n; i++) { //i<=sqrt(n)

        if (isPrime[i]) {

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

                isPrime[j] = false;
            }
        }
    }

    cout << "2 到 " << n << " 之间的质数有: ";

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

        if (isPrime[i]) cout << i << " ";
    }
}
```

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

```
    return 0;
```

```
}
```

## 2、线性筛法 (Linear Sieve)

线性筛法是一种改进的筛法：

- 每个合数只会被**它的最小质因数筛一次**，保证算法时间更快（线性时间）。
- 维护一个质数列表，同时标记合数的最小质因数。

### 举例说明

- 就像把每个合数只涂一次红，不像埃氏筛可能重复涂，节省时间更快找到质数。

### C++示例代码（线性筛法）：

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n = 30;

    vector<int> primes;
    vector<int> minPrime(n + 1, 0);

    for (int i = 2; i <= n; i++) {
        if (minPrime[i] == 0) {
            minPrime[i] = i;
            primes.push_back(i);
        }

        for (int p : primes) {
            if (p > minPrime[i] || i * p > n) break;
            minPrime[i * p] = p;
        }
    }
}
```

2*2									2
2*3	3*3								3
2*4	3*4								
2*5	3*5	5*5							5
2*6	3*6	5*6							
2*7	3*7	5*7	7*7						7
2*8	3*8	5*8	7*8						
2*9	3*9	5*9	7*9						
2*10	3*10	5*10	7*10						
2*11	3*11	5*11	7*11	11*11					11
2*12	3*12	5*12	7*12	11*12					
2*13	3*13	5*13	7*13	11*13	13*13				13
2*14	3*14	5*14	7*14	11*14	13*14				
2*15	3*15	5*15	7*15	11*15	13*15				
2*16	3*16	5*16	7*16	11*16	13*16				
2*17	3*17	5*17	7*17	11*17	13*17	17*17			17
2*18	3*18	5*18	7*18	11*18	13*18	17*18			
2*19	3*19	5*19	7*19	11*19	13*19	17*19	19*19		19
2*20	3*20	5*20	7*20	11*20	13*20	17*20	19*20		
2*21	3*21	5*21	7*21	11*21	13*21	17*21	19*21		
2*22	3*22	5*22	7*22	11*22	13*22	17*22	19*22		
2*23	3*23	5*23	7*23	11*23	13*23	17*23	19*23	23*23	23
2*24	3*24	5*24	7*24	11*24	13*24	17*24	19*24	23*24	
2*25	3*25	5*25	7*25	11*25	13*25	17*25	19*25	23*25	
2*26	3*26	5*26	7*26	11*26	13*26	17*26	19*26	23*26	
2*27	3*27	5*27	7*27	11*27	13*27	17*27	19*27	23*27	
2*28	3*28	5*28	7*28	11*28	13*28	17*28	19*28	23*28	
2*29	3*29	5*29	7*29	11*29	13*29	17*29	19*29	23*29	29

```
    }  
  
    cout << "2 到 " << n << " 之间的质数有: ";  
  
    for (int p : primes) cout << p << " ";  
  
    return 0;  
  
}
```

### 考试注意事项 & 易错点

易错点	正确说明
埃氏筛起始位置错误	内层循环从 $i*i$ 开始，非从 $2*i$
线性筛判断条件写错	$p > \text{minPrime}[i]$ 是剪枝条件，不能漏写
标记数组初始化错误	素数数组或标记数组要初始化正确
数组越界	确保循环变量不超出数组范围