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。

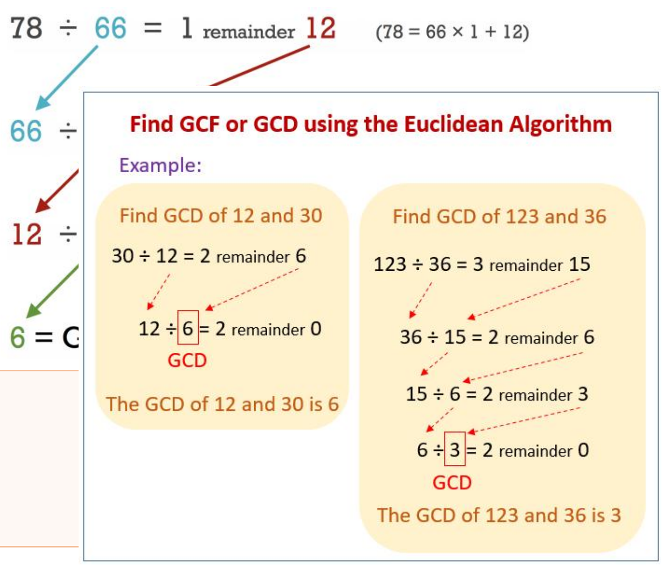
- 最大公约数是能**同时整除两个数的最大整数**。
- 辗转相除法通过不断用较大数除以较小数，再用除数和余数替换，直到余数为 0。
- 最后得到的除数就是最大公约数。

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：

- 18 ÷ 12 = 1 余 6
- 12 ÷ 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;
```

```
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

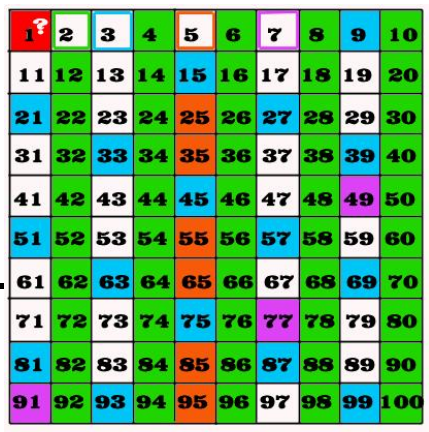
易错点	正确说明
余数为 0 时未及时停止	余数为 0 时，当前除数即最大公约数
参数顺序写反	调用时顺序无影响，算法内部会处理
误把最大公约数当最大公倍数	最大公约数是“共同的最大因数”，非倍数

（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）



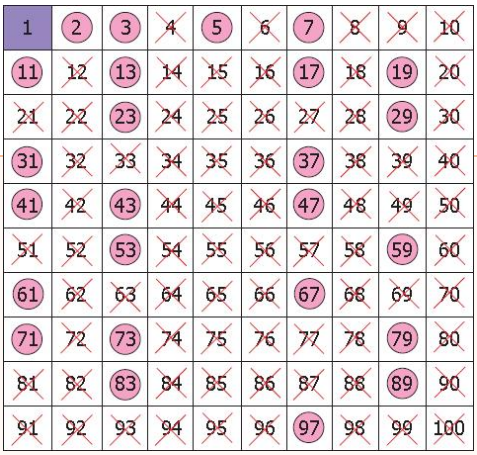
埃氏筛法是一种古老且简单的找质数的方法：

- 从 2 开始，把 2 **的倍数**都标记为非质数。

- 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 << " ";
    }
```

```
        return 0;
    }
```

2、线性筛法（Linear Sieve）

线性筛法是一种改进的筛法：

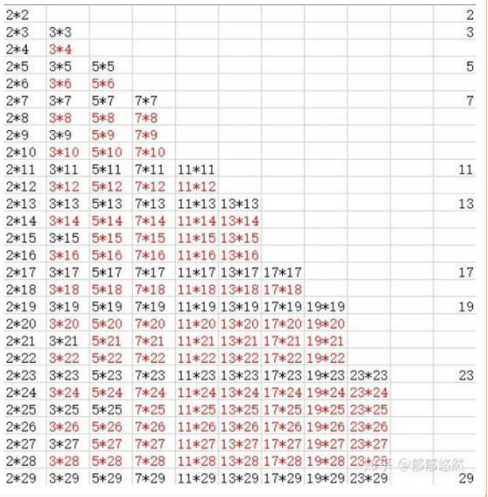
- 每个合数只会被**它的最小质因数筛一次**，保证算法时间更快（线性时间）。
- 维护一个质数列表，同时标记合数的最小质因数。

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;
        }
```



```
        }
        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 > 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