

**计算机与信息学院**

**《信息安全数学基础》学习报告**

|  |  |
| --- | --- |
| **学生姓名：** | 朱铭宇 |
| **学 号：**  **邮 箱：** | 2020210574  975440488@qq.com |
| **专业班级：** | 信息安全20-1班 |
| **指导教师：** | 朱晓玲 |
| **题 目：** | 26.原根存在性判断以及所有原根的求解 |

**2021 年 12 月**

# 题目：

26.模m的原根存在性判断以及求解。判断m原根是否存在，如存在给出一个原根以及所有原根。

# 一、问题描述

首先要判断给定模m是否存在原根，然后需要对其原根进行求解。

# 二、数学基础

①埃氏筛法，将给定范围内的素数以打表的形式求解出来

②判断是否存在原根

③对于给定模m，求解欧拉函数φ(m)

④利用欧几里德除法判断底数a和模m是否互素

⑤利用模平方重复法计算同余式的结果，判断是否同余1

# 算法描述

①埃氏筛法：先将大小为n的数组全部设置为1，从2开始遍历该数组，遇到值为1的下标，便将其下标倍数的元素设置为0（1为素数，0为合数）

②判断给定模m，原根存在性：模m是2、4、pα、2pα中的一种，才会存在原根

③求解欧拉函数，通过欧拉函数公式进行求解

④利用欧几里德除法判断互素：求解两个数的最大公因数，若最大公因数大于1则不互素，若最大公因数等于1则互素

⑤模平方重复法：此处算法实现方式为：answer为模平方重复法中的a，base为b，通过位运算将指数的二进制形式and 1，加上二进制右移运算符，从而实现每次都对指数二进制最低位进行操作，实现二进制位的遍历。当二进制为1和0时，answer和base有不同的求解方式。

# 算法实现

①埃氏筛法：

void Eratosthenes(const int &num, vector<int> &prime)

{

prime.resize(num + 1);

for (auto &i : prime)

i = 1;

prime[0] = 0, prime[1] = 0;

for (int i = 1; i <= num + 1; ++i)

{

if (prime[i] == 0)

continue;

//该数是素数，将他的倍数设置为合数

int composite = i;

while ((composite += i) <= num)

prime[composite] = 0;

}

}

②判断原根存在性：

bool judge\_exist\_root(const int &num)

{

if (num == 2 || num == 4)

return true;

Eratosthenes(num, prime);

for (int element = 3; element < prime.size(); ++element)

{

if (prime[element] == 0)

continue;

for (int i = 1;; ++i)

{

int target = pow(element, i);

if (target > num)

break;

if (target == num || target \* 2 == num)

return true;

}

}

return false;

}

③欧拉函数：

int Euler(const int &num)

{

//求欧拉函数

if (prime[num] == 1)

return num - 1;

int result = num, n = num;

for (int i = 2; i \* i <= n; ++i)

{

if (n % i == 0)

{

//找到一个素因子

result = result / i \* (i - 1);

while (n % i == 0)

//将该素因子全部约去

n /= i;

}

}

if (n > 1)

//检验有无遗漏

result = result / n \* (n - 1);

return result;

}

④判断两个数是否互素：

bool coprime(int i, int num)

{

//欧几里德除法判断互素

if (i == 1 || num == 1)

// 两个正整数中，只有其中一个数值为1，两个正整数为互质数

return true;

while (1)

{

//求出两个数的最大公约数

int temp = i % num;

if (temp == 0)

break;

else

i = num, num = temp;

}

if (num > 1)

// 如果最大公约数大于1，表示两个正整数不互质

return false;

return true;

}

⑤模平方重复法：

int quick\_pow(const int &a, int b, const int &mod)

{

//模平方重复法

int answer = 1, base = a % mod; //answer为模平方重复法中的a，base为b

while (b != 0)

{

//位运算and，计算结果为a的二进制最低一位是否为1,通过右移运算符更新最低位

if (b & 1)

answer = (answer \* base) % mod;

base = (base \* base) % mod;

b >>= 1;

}

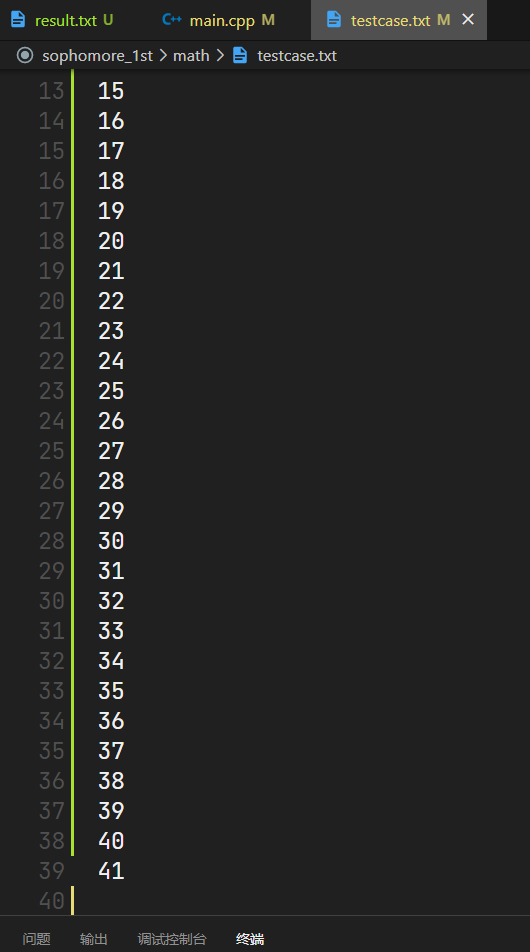
return answer;

}

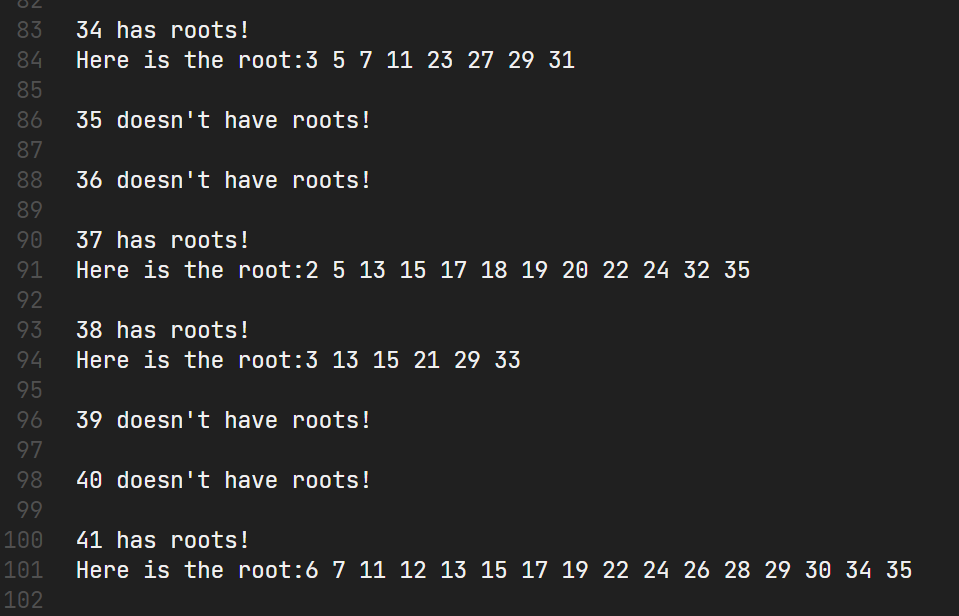
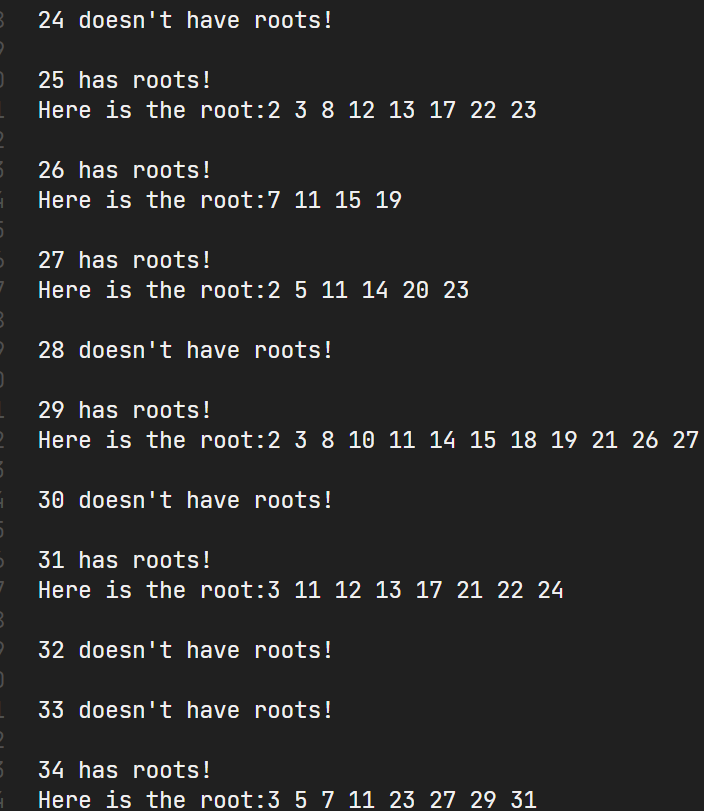
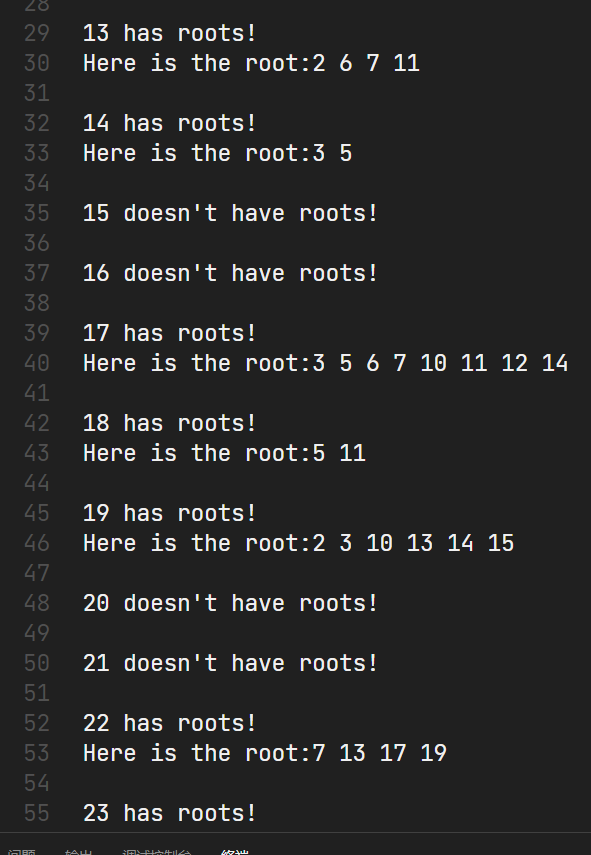
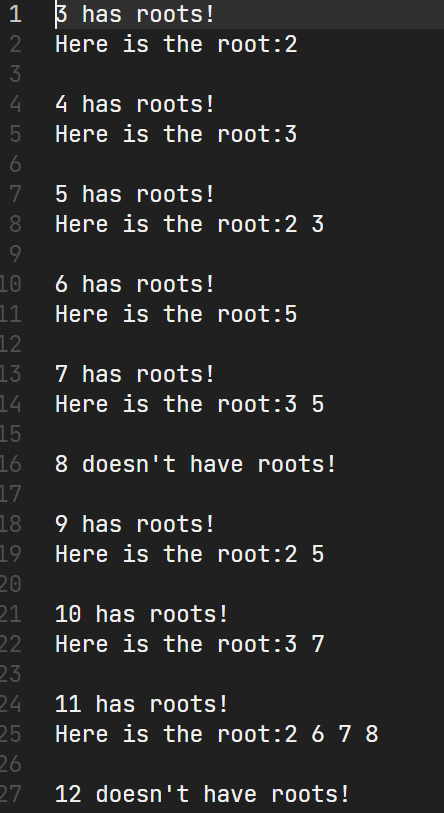
# 运行结果



输入我需要的数据范围上限，给定41，就会在测试用例中生成3~41的所有数



生成测试用例



结果保存在另一个result.txt文件中

# 六、参考文献

无