# 第4次编程练习报告

姓名：胡博浩 学号：2212998 班级：信息安全

##### 编程练习1——求解最小原根并基于最小原根构造指数表

* **源码部分：**

#include <iomanip>

#include <iostream>

#include <unordered\_map>

using namespace std;

int quick\_mod(int a, int b, int m) {//利用平方乘算法进行模幂运算

int result = 1;

a %= m;

while (b > 0) {

if (b & 1) result = (result \* a) % m;

a = (a \* a) % m;

b >>= 1;

}

return result;

}

int gcd(int a, int b)

{

a = abs(a);//取绝对值

b = abs(b);

int r = b;

while (r != 0) {

r = a % b;

a = b;

b = r;

}

return a;

}

unordered\_map<int, int> primeFactorDecompose(int n) {//利用埃拉托斯特尼筛选法进行质因数分解

unordered\_map<int, int> factors;

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

while (n % i == 0) {

factors[i]++;

n /= i;

}

}

if (n > 1) factors[n]++;

return factors;

}

int phi(int n) {//利用欧拉函数的性质计算欧拉函数值

unordered\_map<int, int> factors = primeFactorDecompose(n);

int result = n;

for (auto& factor : factors) {

result -= result / factor.first;

}

return result;

}

bool hasPrimitiveRoot(int n) {

unordered\_map<int, int> factors = primeFactorDecompose(n);

int count = factors.size();

//如果n只有一个质因数，且这个质因数不是2，或者是2且其指数为1或2，那么返回true

//如果n有两个质因数，其中一个是2，且指数为1，那么也返回true

return count == 1 && (factors.find(2) == factors.end() || factors[2] == 1 || factors[2] == 2) ||

count == 2 && factors.find(2) != factors.end() && factors[2] == 1;

}

int getMin(int n) {//计算最小原根

int phi\_n = phi(n);

unordered\_map<int, int> factors = primeFactorDecompose(phi\_n);

for (int g = 1; g < n; ++g) {

if (gcd(g, n) != 1) continue;//如果g和n不互质，则跳过当前g

bool isPrimitiveRoot = true;

for (auto& factor : factors) {

if (quick\_mod(g, phi\_n / factor.first, n) == 1) {

isPrimitiveRoot = false;

break;

}

}

if (isPrimitiveRoot) return g;

}

return -1;

}

unordered\_map<int, int> getIndexTable(int n, int phi\_n, int g) {//生成指数表

unordered\_map<int, int> ind\_map;

for (int r = 0; r < phi\_n; ++r) {

ind\_map.emplace(quick\_mod(g, r, n), r);

}

return ind\_map;

}

void printIndTable(int n, int phi\_n, int g) {//打印指数表

unordered\_map<int, int> ind\_map = getIndexTable(n, phi\_n, g);

cout << "The ind\_table of " << n << " based on g=" << g << " is: " << endl;

int tens = n / 10;

for (int i = 0; i <= tens + 1; i++) {

for (int j = 0; j <= 10; j++) {

if (i == 0 && j == 0) {

cout << setw(5) << " ";

}

else if (i \* j == 0) {

cout << setw(5) << i + j - 1;

}

else {

int t = 10 \* (i - 1) + j - 1;

if (ind\_map.find(t) != ind\_map.end())

cout << setw(5) << ind\_map.find(t)->second;

else

cout << setw(5) << "-";

}

}

cout << endl;

}

}

int main() {

int n;

cout << "Please input n(n>0): ";

cin >> n;

if (hasPrimitiveRoot(n)) {

int min\_g = getMin(n);

if (min\_g != -1) {

cout << "The min primitive root of " << n << ": g=" << min\_g << endl;

printIndTable(n, phi(n), min\_g);

}

else {

cout << "Error: Unable to find the minimum primitive root of " << n << endl;

}

}

else {

cout << "The number " << n << " has no primitive root!" << endl;

}

system("pause");

return 0;

}

**说明部分：**

这段代码实现了求一个正整数的最小原根以及它的指数表。

1、quick\_mod函数：使用平方乘算法高效地计算模幂运算（a^b % m）。

2、primeFactorDecompose函数：使用埃拉托斯特尼筛法对给定整数n进行质因数分解。它返回一个map，其中键是质因数，而值是它对应的幂次。

3、phi函数：计算欧拉函数φ(n)。利用了欧拉函数的性质。

4、hasPrimitiveRoot函数：**判断存不存在原根**。即判断n是不是2、4、奇素数幂、2倍的奇素数幂。如果n只有一个质因数，且这个质因数不是2，或者是2且其指数为1或2，那么返回true；如果n有两个质因数，其中一个是2，且指数为1，那么也返回true。

5、getMin函数：寻找正整数n的最小原根。首先计算欧拉函数值，然后遍历**1到欧拉函数值之间的与n互质**的所有整数g。对于每个g，检查是否满足原根的条件：对于所有素数因子p，g^((phi)/p) % n≠1。（**注意，由于2的最小原根是1，所以从1开始遍历**）

6、getIndexTable函数：根据原根g生成指数表。它计算g^r % n，其中r从0到φ(n)-1，并将结果存储在map中，其中键是计算得到的值，值是对应的指数r。

7、printIndTable函数：以表格形式打印由getIndexTable生成的指数表。

8、main 函数：首先获取用户输入的正整数 n，然后计算其最小原根g，并在存在原根的情况下打印指数表。

* **运行示例：**



