求原根

小圆滚滚

1 求素数

python语言用筛法求n以内的素数表

```
#coding=UTF-8

import csv
import os

#1. 创建文件对象
path = 'data' + os.sep + 'SuShu_data1.txt'
f = open(path, 'w', encoding='utf-8', newline='')

n = 10**7
sum1 = 0
prime = []
ls = []
#2. 基于文件对象构建 csv写入对象
csv_writer = csv.writer(f)
for i in range(n + 1):
    prime.append(True)
for i in range(2, n + 1):
    if prime[i]:
        #print(i, end=",",),
        #ls.append(i)
        csv_writer.writerow([i])
        sum1 += 1
        j = i + i
        while j <= n:
            prime[j] = False
            j += i

#print(ls)
#3. 关闭文件
```

```
f.close()
print("总计：{}".format(sum1))
```

2 求原根

chatGPT:求一个给定的数的原根，如果没有原根则打印无，如果有则保存所有原根到一个文本文件，要求算法最优，请给出详细的C++源代码。

C++代码如下：

```
#include <iostream>
#include <vector>
#include <cmath>
#include <algorithm>
#include <fstream>

using namespace std;

typedef long long ll;

ll powmod(ll a, ll b, ll m) {
    ll res = 1;
    while (b) {
        if (b & 1) res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res;
}

vector<ll> factorize(ll n) {
    vector<ll> res;
    for (ll i = 2; i * i <= n; i++) {
        while (n % i == 0) {
            res.push_back(i);
            n /= i;
        }
    }
    if (n > 1) res.push_back(n);
    return res;
}

```

```cpp
bool check_primitive_root(ll g, ll p, const vector<ll>& factors) {
    // 求出phi(p)
    ll phi = p - 1;
    for (auto f : factors) {
        if (powmod(g, phi / f, p) == 1) {
            return false;
        }
    }
    return true;
}

vector<ll> get_primitive_roots(ll p) {
    vector<ll> factors = factorize(p - 1);
    vector<ll> roots;
    for (int g = 2; g < p; g++) {
        if (check_primitive_root(g, p, factors)) {
            roots.push_back(g);
        }
    }
    return roots;
}

int main() {
    ll p;  // p 为一个质数
    cin >> p;
    vector<ll> roots = get_primitive_roots(p);
    if (roots.empty()) {
        cout << "无" << endl;
    } else {
        ofstream out("roots.txt");
        for (auto r : roots) {
            out << r << endl;
        }
        cout << "保存了" << roots.size() << "个原根到 roots.txt 文
            件" << endl;
    }
    return 0;
}
```

实现思路：

1. 定义函数 ‘$powmod(a, b, m)$‘ 对 a 进行快速幂得到 $a^b \bmod m$
2. 定义函数 ‘$factorize(n)$‘ 分解 n 的质因数并返回一个因子数组

3. 定义函数 ‘$check_primitive_root(g, p, factors)$‘，验证 g 是否为 p 的原根。思路是对 $p-1$ 这个数的每个质因子 f，判断 $g^{(p-1)/f} \bmod p$ 是否为 1，如果都不为1，则 g 是原根，否则不是
4. 定义函数 ‘$get_primitive_roots(p)$‘ 求解 p 的所有原根，返回一个原根数组。具体实现是枚举 2 到 $p-1$ 所有数，对于每个数 g，调用 ‘$check_primitive_root(g, p, factors)$‘ 判断是否为原根，是则加入结果数组
5. 主函数中读入 p，调用 ‘$get_primitive_roots(p)$‘ 求出所有原根，如果结果数组为空，则输出“无”，否则将原根保存到文件中，并输出保存的原根数量

可以结合以下几个数检验代码进行测试：

```cpp
bool is_prime(ll n) {
    if (n <= 1) return false;
    for (ll i = 2; i * i <= n; i++) {
        if (n % i == 0) return false;
    }
    return true;
}

bool is_primitive_root(ll g, ll p) {
    vector<ll> factors = factorize(p - 1);
    return check_primitive_root(g, p, factors);
}

void test(ll p) {
    if (!is_prime(p)) {
        cout << p << " 不是质数" << endl;
        return;
    }
    vector<ll> roots = get_primitive_roots(p);
    if (roots.empty()) {
        cout << p << " 没有原根" << endl;
    } else {
        for (auto r : roots) {
            if (!is_primitive_root(r, p)) {
                cout << r << " 不是原根" << endl;
            }
        }
        cout << p << " 有 " << roots.size() << " 个原根" << endl;
    }
}
```