**西安电子科技大学网信院**

**信息安全基础与密码学**

**综合实验**

**实 验 报 告（一）**

**Fermat素性检测算法**

**班级：2118021**

**姓名： 盖乐**

**学号：21009200991**

**日期：2024.9.22**

一、实验目的

1. 实验环境：
   1. Windows11
   2. Python 3.12
2. 实现目标：

给定奇整数m大于等于3和安全参数k，判断m是否为合数，或者有(1-1/2^k)概率为素数。

二、方案设计

1. 实验原理
   1. Fermat小定理：给定素数p,a∈Z，则有a^(p-1)≡1(mod p)；
   2. 奇整数m，若任取一整数2≤a≤m-2，(a,m)= 1，使得 a^(m-1)=1 (mod m)，则m至少有1/2的概率为素数。
2. 算法流程
   1. 给定奇整数m>=3和安全参数k
   2. 随机选取a，2<=a<=m-2
   3. 计算g=(a,m),若g=1，则转d)，否则m为合数
   4. 计算r=a^m-1(mod m), 若r=1，转b)，否则m为合数
   5. 重复计算k次，则m为素数的概率为1-1/2^k

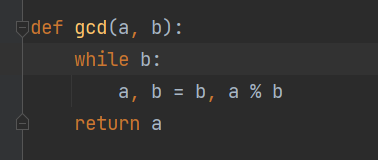
三、方案实现

1. 算法流程图：

图示

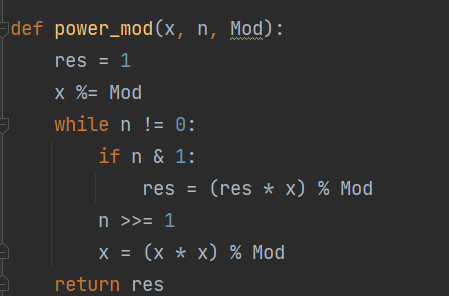
描述已自动生成

1. 主要函数：
   1. 求最大公约数gcd(a, b)



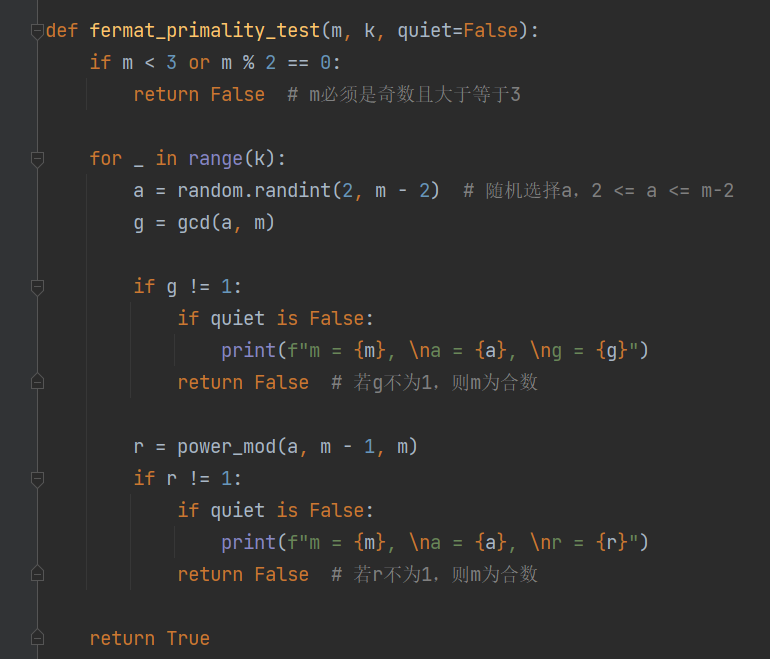
辗转相除法

* 1. 计算模幂运算power\_mod(base, exponent, modulus)

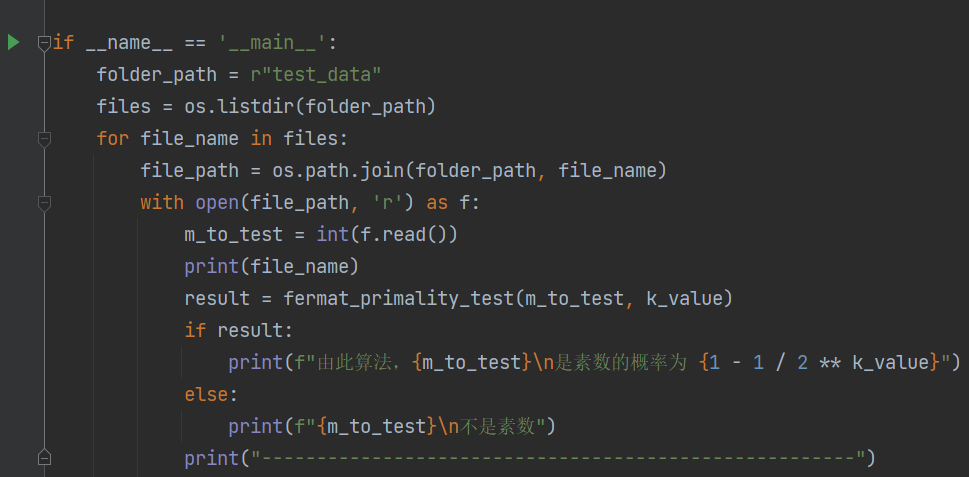


函数使用了快速幂的算法，通过迭代减小指数的同时，根据指数的奇偶性来更新计算结果。这有助于降低计算复杂度。

* 1. Fermat素性检验函数fermat\_primality\_test(m, k)



* 1. 主函数



1. 代码：

import os  
import random  
  
# 测试次数  
k\_value = 5  
  
  
def gcd(a, b):  
 while b:  
 a, b = b, a % b  
 return a  
  
  
def power\_mod(x, n, Mod):  
 res = 1  
 x %= Mod  
 while n != 0:  
 if n & 1:  
 res = (res \* x) % Mod  
 n >>= 1  
 x = (x \* x) % Mod  
 return res  
  
  
def fermat\_primality\_test(m, k, quiet=False):  
 if m < 3 or m % 2 == 0:  
 return False # m必须是奇数且大于等于3  
  
 for \_ in range(k):  
 a = random.randint(2, m - 2) # 随机选择a，2 <= a <= m-2  
 g = gcd(a, m)  
  
 if g != 1:  
 if quiet is False:  
 print(f"m = {m}, \na = {a}, \ng = {g}")  
 return False # 若g不为1，则m为合数  
  
 r = power\_mod(a, m - 1, m)  
 if r != 1:  
 if quiet is False:  
 print(f"m = {m}, \na = {a}, \nr = {r}")  
 return False # 若r不为1，则m为合数  
  
 return True  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 folder\_path = r"test\_data"  
 files = os.listdir(folder\_path)  
 for file\_name in files:  
 file\_path = os.path.join(folder\_path, file\_name)  
 with open(file\_path, 'r') as f:  
 m\_to\_test = int(f.read())  
 print(file\_name)  
 result = fermat\_primality\_test(m\_to\_test, k\_value)  
 if result:  
 print(f"由此算法，{m\_to\_test}\n是素数的概率为 {1 - 1 / 2 \*\* k\_value}")  
 else:  
 print(f"{m\_to\_test}\n不是素数")  
 print("------------------------------------------------------")

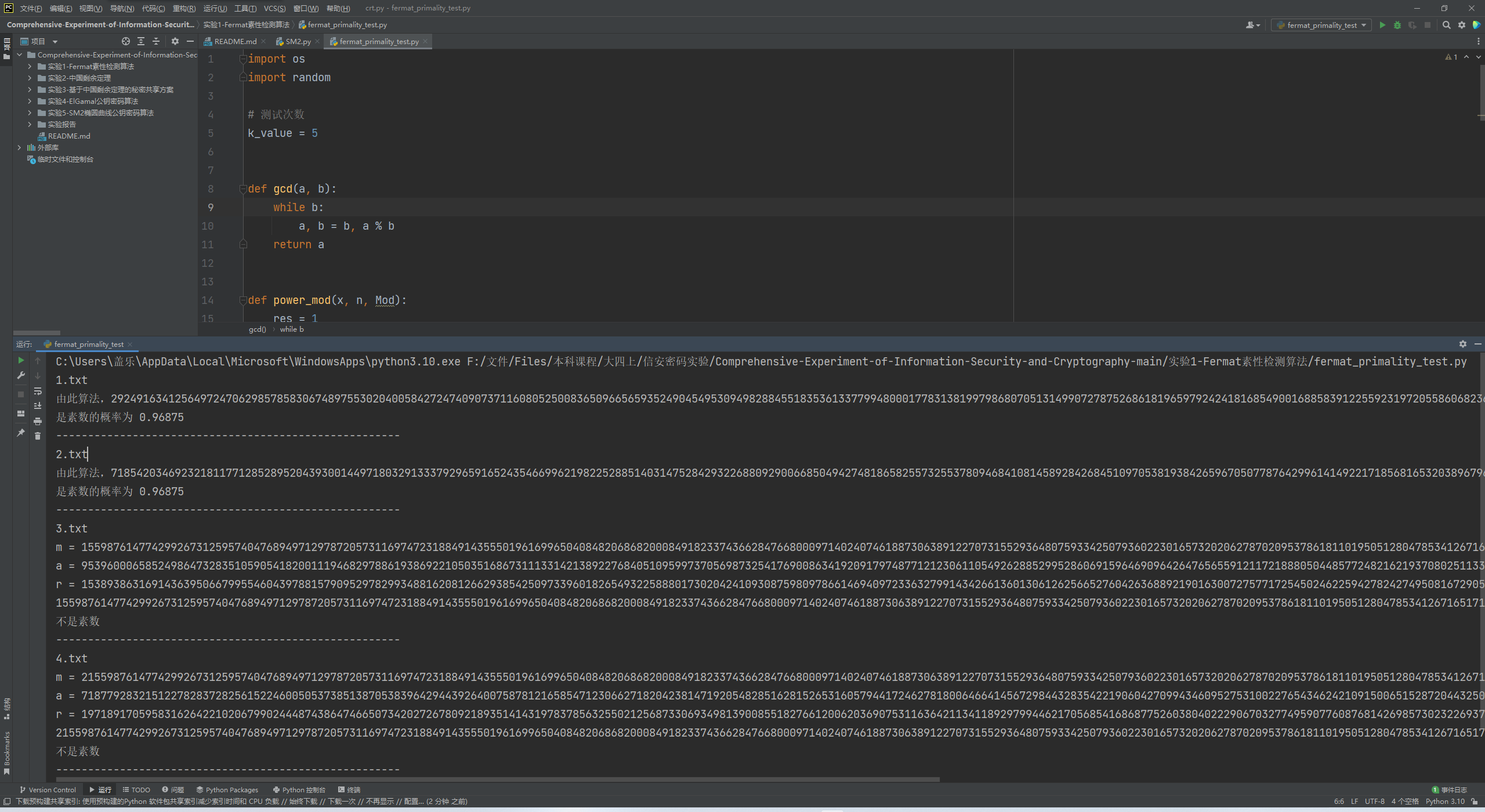
四、数据分析(包括算法测试数据的分析，运行结果截图等等)

测试：

使用如下代码分别读入四个文档的数据，并进行Fermat素性检验

if \_\_name\_\_ == '\_\_main\_\_':  
 folder\_path = r"test\_data"  
 files = os.listdir(folder\_path)  
 for file\_name in files:  
 file\_path = os.path.join(folder\_path, file\_name)  
 with open(file\_path, 'r') as f:  
 m\_to\_test = int(f.read())

检验结果：



五、思考与总结

1. 如果有一个整数𝒂，(𝒂,𝒎)=𝟏，使得𝒂𝒎−𝟏≡𝟏 𝒎𝒐𝒅 𝒎 则𝒎一定是一个素数吗？为什么？

不一定。例如：a=8,m=63;显然8^63-1≡8^62≡(8^2)^31≡1(mod63)

但是63=3×3×7，显然不是素数。

1. Fermat中都用到了哪些运算？分别实现什么功能？请简述。

Euclid算法,辗转相除法求a和m的最大公约数；

快速幂算法，计算模幂运算，即a^b mod c

1. 你还了解哪种素性检测算法？请简述，并分析其与Fermat素性检测算法的区别与联系。
2. Miller-Rabin素性检测:

基本原理：Miller-Rabin算法基于费马小定理的扩展，引入随机选取的底数。

算法特点：相较于费马素性检测更为强大，通常更准确，尤其对于大数。

1. Solovay-Strassen素性检测:

基本原理：类似于费马素性检测，但引入了雅可比符号。

算法特点：也是一种概率性算法，与费马素性检测相似。

他们都是概率算法，结果可能出错。

1. 实验过程中还遇到了什么问题，如何解决的？通过该实验有何收获？

实验相对简单，没有遇到什么大的问题。

收获：概率性素性检测算法，在一些情况下，可以用概率性算法来判断素性，虽然结果可能是错误的，但是可以通过多次运行来提高准确性。