**实验一 密码学数学基础**

**一、实验目的**

1. 掌握最大公因数的计算方法，理解其在密码学中的重要性。

2. 学习扩展欧几里得算法，能够计算乘法逆元。

3. 熟悉模幂运算的方法，了解其在加密和签名算法中的应用。

**二、实验内容**

1. 求最大公因数

给定两个整数a和b，计算其最大公因数。

2. 扩展的欧几里得算法

输入两个整数a和b，利用扩展欧几里得算法计算其GCD，并输出对应的线性组合系数x和y。

3. 模幂运算

给定整数a，m，n，使用快速幂算法计算。

**三、实验代码（基于Python环境）**

|  |
| --- |
| # 1.暴力穷举法 def gcd\_baoli(a, b):  min\_num = min(a, b)  gcd = 1  for i in range(1, min\_num + 1):  if a % i == 0 and b % i == 0:  gcd = i  return gcd  # 2.更相减损法 def gcd\_xiangjian(a, b):  while a != b:  if a > b:  a = a - b  else:  b = b - a  return a  # 3.辗转相除法 def gcd\_xiangchu(a, b):  while b:  a, b = b, a % b  return a  # 4.扩展的欧几里得算法  def extended\_gcd(a, b):  if a == 0:  return (b, 0, 1)  else:  g, y, x = extended\_gcd(b % a, a)  return (g, x - (b // a) \* y, y)  # 5.模幂运算  def mod\_pow(a, m, n):  result = 1  a = a % n #取余先简略计算  while m > 0:  if m % 2 == 1:  result = (result \* a) % n #提出来依然可以使m变成两倍  m = m >> 1  a = (a \* a) % n  return result  # 主函数  if \_\_name\_\_ == "\_\_main\_\_":  # 获取用户输入  a = int(input("请输入整数a: "))  b = int(input("请输入整数b: "))  m = int(input("请输入整数m: "))  n = int(input("请输入整数n: "))   # 最大公因数  print(f"暴力穷举法求 {a} 和 {b} 的最大公因数: {gcd\_baoli(a, b)}")  print(f"更相减损法求 {a} 和 {b} 的最大公因数: {gcd\_xiangjian(a, b)}")  print(f"辗转相除法求 {a} 和 {b} 的最大公因数: {gcd\_xiangchu(a, b)}")   # 扩展欧几里得算法  g, x, y = extended\_gcd(a, b)  print(f"扩展欧几里得算法求 {a} 和 {b} 的 GCD: {g}，线性组合系数 x: {x}，y: {y}")   # 模幂运算  print(f"快速幂算法计算 {a}^{m} mod {n}: {mod\_pow(a, m, n)}") |

文本

AI 生成的内容可能不正确。**四、实验结果**

实验验证数据选择了例题给出的数据进行验证，经验证采用四种方法均可以求得最大公因数，同时使用拓展欧几里得法也能求出对应的系数。针对快速幂算法的验证

文本

AI 生成的内容可能不正确。

能够快速求得余数，验证成功。

文本

AI 生成的内容可能不正确。**五、实验总结与心得**

在理解拓展欧几里得和模幂运算中，直接的文字理解比较抽象寻找了两个例子进行帮助解读。其例子具体如下：其本质的解决问题的思路都为将一个比较复杂的问题，通过递归进行化简，同时利用了将大数比较占据空间的计算换成先幂的运算从而减少运算时间。通过例子能够很好的理解算法的具体运行方式，从而更好的实现算法，同时不得不提python以元组返回，直接进行变量的传参真的很方便。

文本

AI 生成的内容可能不正确。文本

AI 生成的内容可能不正确。