

作业 1

1. 编程实现任意两个正整数的最小公倍数。↵
2. 编程实现求解模素数 p 构成的有限域 $GF(p)$ 上的全部非零元素的加法和乘法逆元。↵

1. 求最大公约数与最小公倍数

```
def gcd(a, b):  
    """计算 a 和 b 的最大公约数"""  
    while b != 0:  
        a, b = b, a % b  
    return a  
  
def lcm(a, b):  
    """计算 a 和 b 的最小公倍数"""  
    return a * b // gcd(a, b)  
  
if __name__ == "__main__":  
    num1 = 18  
    num2 = 15  
    print(f"{num1} 和 {num2} 的最大公约数是 {gcd(num1, num2)}, 最小公倍  
数是 {lcm(num1, num2)}")
```

输出:

```

M+ 信道编码 - 作业.md M  gcd_lcm.py M X
Homework > Information_Theory > class > gcd_lcm.py > ...
You, 2 minutes ago | 1 author (You)
1  def gcd(a, b):
2      """计算 a 和 b 的最大公约数"""
3      while b != 0:
4          a, b = b, a % b
5      return a
6
7  def lcm(a, b):
8      """计算 a 和 b 的最小公倍数"""
9      return a * b // gcd(a, b)
10
11  if __name__ == "__main__":
12      num1 = 18
13      num2 = 15
14      print(f"{num1} 和 {num2} 的最大公约数是 {gcd(num1, num2)}, 最小公倍数是 {lcm(num1, num2)}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SQL CONSOLE

```

/home/kaikai/.pyenv/versions/3.10.12/bin/python /mnt/d/projects/docs/Homework/Information_Theory/class/gcd_lcm.py
→ docs git:(main) X /home/kaikai/.pyenv/versions/3.10.12/bin/python /mnt/d/projects/docs/Homework/Information_Theory/class/gcd_lcm.py
12 和 15 的最小公倍数是 60
→ docs git:(main) X /home/kaikai/.pyenv/versions/3.10.12/bin/python /mnt/d/projects/docs/Homework/Information_Theory/class/gcd_lcm.py
18 和 15 的最小公倍数是 90
→ docs git:(main) X /home/kaikai/.pyenv/versions/3.10.12/bin/python /mnt/d/projects/docs/Homework/Information_Theory/class/gcd_lcm.py
18 和 15 的最大公约数是 3, 最小公倍数是 90
→ docs git:(main) X

```

2. 求乘法逆元与加法逆元

Copy

```

def is_prime(n):
    """检查 n 是否为素数"""
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

def find_inverse_elements(p):
    # 首先检查 p 是否为素数
    if not is_prime(p):
        raise ValueError("输入的数必须是素数")

    # 初始化存储逆元的列表
    additive_inverse = [0] * p
    multiplicative_inverse = [0] * p

```

```

# 计算加法逆元
for i in range(p):
    additive_inverse[i] = (-i) % p

# 计算乘法逆元
for i in range(1, p):
    for j in range(1, p):
        if (i * j) % p == 1:
            multiplicative_inverse[i] = j
            break

# 返回逆元
return additive_inverse, multiplicative_inverse

if __name__ == "__main__":
    # 设置素数
    p = 8

    # 求解逆元
    additive_inverse, multiplicative_inverse =
find_inverse_elements(p)

    # 打印结果
    print(f"模 {p} 的有限域上的全部非零元素的加法逆元是：
{additive_inverse[1:]}")
    print(f"模 {p} 的有限域上的全部非零元素的乘法逆元是：
{multiplicative_inverse[1:]}")

```

输出：

```

docs git:(main) /home/kaikai/.pyenv/versions/3.10.12/bin/python /mnt/d/projects/docs/Homework/In
模 7 的有限域上的全部非零元素的加法逆元是：[6, 5, 4, 3, 2, 1]
模 7 的有限域上的全部非零元素的乘法逆元是：[1, 4, 5, 2, 3, 6]
docs git:(main) X /home/kaikai/.pyenv/versions/3.10.12/bin/python /mnt/d/projects/docs/Homework/I
Traceback (most recent call last):
  File "/mnt/d/projects/docs/Homework/Information Theory/class/inverse.py", line 46, in <module>
    additive_inverse, multiplicative_inverse = find_inverse_elements(p)
  File "/mnt/d/projects/docs/Homework/Information Theory/class/inverse.py", line 20, in find_inverse
    raise ValueError("输入的数必须是素数")
ValueError: 输入的数必须是素数
docs git:(main) X /home/kaikai/.pyenv/versions/3.10.12/bin/python /mnt/d/projects/docs/Homework/I
模 17 的有限域上的全部非零元素的加法逆元是：[16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
模 17 的有限域上的全部非零元素的乘法逆元是：[1, 9, 6, 13, 7, 3, 5, 15, 2, 12, 14, 10, 4, 11, 8, 16]
docs git:(main) X

```