课程项目报告——线性同余方程求解

源码

样例

输入

ps: 下列输入依次是参数**a m b**，即求解 **ax = m (mod b)**

```
13 6 34
```

输出

```
x=24+34k
```

project.py

```python
def get(): # 输入函数
    f = input()
    value = f.split(" ")
    v = [int(i) for i in value]
    return v


def gcd(a, b): # 获取最大公约数
    if b == 0:
        return a
    return gcd(b, a % b)


def solve(a, b, m): # 判断是否有解
    g = gcd(a, b)
    # print(g)
    if m % g == 0:
        return True
    return False


def liner_mod_function(a, b): # 利用扩展欧几里得算法求解
    if b == 0:
        return [1, 0]
    y = liner_mod_function(b, a % b)
    x = [1, 1]
    x[0] = y[1]
    x[1] = y[0] - a // b * y[1]
```

```
    return x

# 求解 ax = m (mod b)
values = get()  # value[0]: a  value[1]: m  value[2]: b
a = values[0]
b = values[2]
m = values[1]
if not solve(a, b, m):
    print("无解")
else:
    x = liner_mod_function(a, b) # 求特解
    ans = x[0]
    ans *= m / gcd(a, b)
    ans %= b
    print("x={}+{}k".format(ans, b)) # 将特解表现成通解形式
```

算法空间效率分析

本程序主要采用扩展欧几里得算法，易得其算法递归层数为**O(log max(a, b))**（下面在时间复杂度里证明之）

而每一层所用空间为**O(1)**，同时在主程序里定义变量个数亦是**O(1)**

故空间复杂度为**O(log n)**

算法时间效率分析

在递归里可以看到，欧几里得算法与扩展欧几里得算法时间复杂度是一样的，故下面证明欧几里得算法时间复杂度即为原算法时间复杂度

简略证明欧几里得算法时间复杂度为O(log n)

gcd(a, b) -> gcd(b, a % b) -> gcd(a % b, b % (a % b))

不妨设a > b

1. b > a / 2 时， a % b = a - b < a / 2
2. b < a / 2 时， a % b < b < a / 2

综上，仅使用两次辗转相除法，第一个参数已经小于原来的一半了，故其复杂度在**O(log n)**以内

故其复杂度为**O(log n)**