

扩展欧几里德算法

小圆滚滚

1 源代码

```
1 X = [0, 1]
2 Y = [1, 0]
3 count = 0 #记录计算次数
4
5
6 def extended\_euclid(x, y):
7     """求模的逆元
8     an
9     :param x:
10    :param y:
11    :return: 无返回, 直接打印
12    """
13    if y[2] == 0:
14        print("无逆元, gcd(%d, %d) = %d" % (a, n, x[2]))
15    elif y[2] == 1:
16        print("gcd(%d, %d) = %d \n a 的逆元: %d" % (a, n, y[2], y[1])) #
17        Y[1]即所求模逆元
18    else:
19        global count
20        Q = x[2] // y[2]
21        T1, T2, T3 = [x[0] - Q * y[0], (x[1] - Q * y[1]), (x[2] - Q * y
22        [2])]
23        x[0], x[1], x[2] = y[0], y[1], y[2]
24        y[0], y[1], y[2] = T1, T2, T3
25        count += 1
26        extended\_euclid(x, y)
27
28 if __name__ == '__main__':
29     while True:
30         a = int(input("请输入的值: a"))
31         n = int(input("请输入的值: n"))
```

```

31     # a = 38
32     # n = 47
33     X.append(a)  # a要求模逆元, a要加在X上, 因为X列表的第一个元素是1。
34     Y.append(n)
35     extended\_euclid(X, Y)
36     # print("迭代次数=".format(count))

```

2 解释

这是一段用 Python 编写的扩展欧几里得算法程序，以下是对图片内容的详细介绍：

2.1 文件信息

* 这段代码是在名为“扩展欧几里得算法 - 20220730 - pre3.py”的文件中编写的，使用的是 Python 3.8 语言版本，文件编码为 UTF - 8，缩进方式是 4 个空格。

2.2 代码结构

2.2.1 变量初始化:

- * ‘X = [0, 1]’：初始化一个列表 X，其中包含两个元素 0 和 1。
- * ‘Y = [1, 0]’：初始化一个列表 Y，其中包含两个元素 1 和 0。
- * ‘count = 0’：定义一个变量 count，并初始化为 0，用于记录递归调用的次数。

2.2.2 函数定义:

定义了一个名为 ‘extended_euclid’ 的函数，用于实现扩展欧几里得算法，该函数的参数是两个列表 X 和 Y。

函数文档字符串： 使用 ‘"""..."""’ 描述了函数的功能，包括求 a 模 n 的逆元，以及参数 a（要求模逆元的数）和 n（模）的作用，同时说明该函数无返回值，直接打印结果。

递归终止条件： * 如果 ‘Y[2] == 0’，则表示没有逆元，此时打印出“无逆元，gcd(”
* 如果 ‘Y2[] == 1’，则表示找到了逆元，打印出“gcd(

递归计算过程： * 定义全局变量 count，用于记录递归调用的次数。

- * 计算 Q 为 ‘X[2]’ 整除 ‘Y[2]’ 的商。
- * 计算 T1、T2、T3 分别为 ‘X[0] - Q * Y[0]’、‘X[1] - Q * Y[1]’ 和 ‘X[2] - Q * Y[2]’。
- * 更新 X 列表的元素为 Y 列表的元素，更新 Y 列表的元素为 T1T、2、T3。 * count 自增 1，表示递归调用次数加 1。
- * 调用 ‘extended_euclid(X, Y)’ 函数进行递归计算。

2.2.3 主程序部分:

- * 判断是否为 ‘_main_’ 模块, 如果是, 则进入无限循环, 不断接收用户输入。
- * 在循环中, 提示用户输入 a 和 n 的值, 并将其转换为整数类型。
- * 将 a 添加到 X 列表的末尾, 因为 X 列表的第二个元素是 1, 这是扩展欧几里得算法的一个要求。
- * 将 n 添加到 Y 列表的末尾。
- * 调用 ‘extended_euclid(X, Y)’ 函数, 开始进行扩展欧几里得算法的计算。
- * 注释掉的 ‘print’ 语句原本用于打印迭代次数。

2.3 代码功能

* 该程序通过扩展欧几里得算法, 计算给定整数 a 在模 n 下的逆元 (如果存在), 即求解满足 $a \times x \equiv 1 \pmod{n}$ 的 x 值。如果没有逆元, 则输出相应的提示信息。

* 通过递归的方式不断更新 X 和 Y 列表中的值, 直到找到满足条件的逆元或者确定没有逆元为止, 同时记录了递归调用的次数。

3 什么叫模逆元

在数学中, 特别是在数论和抽象代数领域, 逆元是指一个数在某个特定运算下的“逆”。具体来说, 对于给定的数 a 和 n , 如果存在一个整数 x 使得:

$$a \times x \equiv 1 \pmod{n}$$

这里, \pmod{n} 表示取模 n 的运算, 即求余数。上述等式说明 a 和 x 相乘后除以 n 的余数为 1。在这种情况下, 我们称 x 是 a 在模 n 下的乘法逆元。

3.1 模逆元的性质

1. ****唯一性****: 如果逆元存在, 那么它是唯一的。即对于给定的 a 和 n , 最多只有一个整数 x 满足上述等式。2. ****存在性****: 并不是所有的 a 在模 n 下都有逆元。逆元存在的条件是 a 和 n 必须互质, 即它们的最大公约数 (GCD) 为 1。这可以通过欧几里得算法来验证。

3.2 应用

逆元在密码学、编码理论、计算机科学等领域有广泛的应用。例如, 在 RSA 加密算法中, 逆元用于解密过程中, 将加密后的密文转换回原始明文。

3.3 扩展欧几里得算法

扩展欧几里得算法不仅可以用来计算两个整数的最大公约数, 还可以找到满足 $a \times x + n \times y = \text{GCD}(a, n)$ 的整数 x 和 y 。当 $\text{GCD}(a, n) = 1$ 时, x 就是 a 在模 n 下的逆元。扩展欧几里得算法是求解逆元的一种有效方法。

4 举个例子

当然可以。让我们通过一个具体的例子来说明什么是逆元以及如何找到逆元。

假设我们有整数 $a = 3$ 和 $n = 11$ ，我们想要找到 a 在模 n 下的乘法逆元，即找到一个整数 x 使得：

$$3 \times x \equiv 1 \pmod{11}$$

这意味着我们需要找到一个整数 x ，使得 $3x$ 除以 11 的余数是 1。

我们可以通过扩展欧几里得算法来找到这个逆元。首先，我们使用欧几里得算法来找到 3 和 11 的最大公约数 (GCD)：

1. $11 = 3 \times 3 + 2$

2. $3 = 2 \times 1 + 1$

3. $2 = 1 \times 2 + 0$

最大公约数是最后一个非零余数，这里是 1。因为 3 和 11 的最大公约数是 1，所以 3 在模 11 下有逆元。

接下来，我们使用扩展欧几里得算法回代找到系数 x 和 y ：

从第二步开始回代：

- $1 = 3 - 2 \times 1$

- 将第一步的 2 替换进去： $1 = 3 - (11 - 3 \times 3) = 3 \times 4 - 11$

所以，我们得到 $3 \times 4 \equiv 1 \pmod{11}$ 。这意味着 4 是 3 在模 11 下的乘法逆元。

为了验证，我们可以计算 3×4 并检查它除以 11 的余数是否为 1：

$$3 \times 4 = 12$$

$$12 \pmod{11} = 1$$

确实，12 除以 11 的余数是 1，这验证了 4 是 3 在模 11 下的逆元。

因此，对于给定的例子，3 在模 11 下的逆元是 4。

5 模逆元和原根有什么关系

逆元和原根是数论中的两个重要概念，它们在模运算中都有重要的应用。下面分别对这两个概念进行解释，并说明它们之间的关系。

5.1 模逆元

对于一个整数 a 和一个正整数 n ，如果存在一个整数 b 使得：

$$a \cdot b \equiv 1 \pmod{n}$$

那么 b 就是 a 模 n 的逆元。逆元存在的条件是 a 和 n 互质，即 $\gcd(a, n) = 1$ 。

5.2 原根

对于一个正整数 n ，如果存在一个整数 g 使得 g 的所有幂次模 n 的结果互不相同，并且可以生成所有与 n 互质的整数，那么 g 就是 n 的一个原根。原根存在的条件是 n 必须是 $2, 4, p^k, 2p^k$ 其中之一，其中 p 是奇素数， k 是正整数。

5.3 逆元和原根的关系

1. **逆元的计算**：如果 g 是 n 的原根，那么 g 的逆元可以通过计算 $g^{\phi(n)-1} \pmod n$ 得到，其中 $\phi(n)$ 是欧拉函数，表示小于 n 且与 n 互质的整数个数。这是因为：

$$g \cdot g^{\phi(n)-1} \equiv g^{\phi(n)} \equiv 1 \pmod n$$

所以 $g^{\phi(n)-1}$ 是 g 的逆元。

2. **原根的性质**：如果 g 是 n 的原根，那么 g 的所有幂次模 n 的结果互不相同，并且可以生成所有与 n 互质的整数。这意味着 g 的逆元也是 n 的一个原根，因为 g 的逆元的幂次模 n 的结果也是互不相同的，并且可以生成所有与 n 互质的整数。

综上所述，逆元和原根之间的关系是：如果 g 是 n 的原根，那么 g 的逆元也是 n 的一个原根，且可以通过计算 $g^{\phi(n)-1} \pmod n$ 得到。

最终答案是：

如果 g 是 n 的原根，那么 g 的逆元也是 n 的一个原根，且可以通过计算 $g^{\phi(n)-1} \pmod n$ 得到。