第三题用手计算比较困难

用python脚本计算(**使用的所有代码 附在最后**)

得到$\mathbb{Z}_{53}^*$这个群的生成元有以下：

```
[2, 3, 5, 8, 12, 14, 18, 19, 20, 21, 22, 26, 27, 31, 32, 33, 34, 35, 39, 41, 45, 48, 50, 51]
```

定理 8.2.7

假设 G 为一个阶为 n 的有限循环群，α 为对应的生成元，则对整除 n 的每个整数 k，G 都存在一个唯一的阶为 k 的循环子群 H。这个子群是由 $\alpha^{n/k}$ 生成的。H 是由 G 内满足条件 $a^k=1$ 的元素组成的，且 G 不存在其他子群。

已经知道了，子群的基数可能取值只能是：1, 2, 4, 13, 26, 52

选择α=2,对不同的k，先找到子群的生成元a，然后像求阶数那样，迭代k次，每一次的结果就是子群的元素之一

例如k=2时

子群的生成元是2^(26) mod 53 = 52

然后

52 的1次方 mod 53 = 52
52 的2次方 mod 53 = 1

故子群是[1, 52]

以此类推：

子群基数	**子群的一个生成元**	**子群**
1	1	[1]
2	52	[1, 52]
4	30	[1, 23, 30, 52]
13	16	[1, 10, 13, 15, 16, 24, 28, 36, 42, 44, 46, 47, 49]

子群基数	子群的一个生成元	子群
26	4	[1, 4, 6, 7, 9, 10, 11, 13, 15, 16, 17, 24, 25, 28, 29, 36, 37, 38, 40, 42, 43, 44, 46, 47, 49, 52]
52	[2, 3, 5, 8, 12, 14, 18, 19, 20, 21, 22, 26, 27, 31, 32, 33, 34, 35, 39, 41, 45, 48, 50, 51]	自身

```
class Group:
    def __init__(self, set, modulus):
        self.set = tuple(set)
        self.cardinality = len(set)
        self.modulus = modulus
        self.ords = None
        self.generators = None
        self.calculate_ords()

    def calculate_ords(self, display=False):#计算每个元素的阶，顺便算出生成元们
        result = []
        generator = []
        for number in self.set:
            ord_number = self.ord(number)
            temp = [number, ord_number]
            result.append(temp)
            if ord_number == self.cardinality:#如果阶数等于基数
                generator.append(number)
        if display:
            for i in result:
                print('ord(%d) = %d' % tuple(i))
            print(generator)
        self.ords = result
        self.generators = generator




    def ord(self, number, display=False):#计算某个特定元素的阶
        temp = number
        for exp in range(1, self.modulus):
            if temp == 1:
                if display:
                    print('%d 的%d次方 mod %d = 1' % (number, exp, self.modulus))
                    print('ord(%d) = %d' % (number, exp))
                return exp
            else:
                if display:
```

```
                    print('%d 的%d次方 mod %d = %d' % (number, exp, self.modulus,
temp))
            temp = (temp * number) % self.modulus

    def child_group(self, k):#k是子群基数
        rst = []
        alpha = self.generators[0]#alpha是母群的一个生成元
        a = (alpha ** (self.cardinality / k)) % self.modulus #a是子群的一个生成元
        a = int(a)
        print('生成元是', a)
        temp = a
        for i in range(0, k):#迭代k次
            rst.append(temp)
            temp = (temp * a) % self.modulus
        rst.sort()
        return rst

if __name__ == '__main__':
    A = []
    for i in range(1, 53):
        A.append(i)
    GR = Group(set=A, modulus=53)
    print(GR.child_group(26))
    GR.ord(52,True)
```