1. 题目
2. 题干

编写一个算法来判断一个数 n 是不是快乐数。

「快乐数」定义为：对于一个正整数，每一次将该数替换为它每个位置上的数字的平方和，然后重复这个过程直到这个数变为 1，也可能是 **无限循环** 但始终变不到 1。如果 **可以变为**  1，那么这个数就是快乐数。

如果 n 是快乐数就返回 True ；不是，则返回 False 。

1. 示例

**输入：**19

**输出：**true

**解释：**

12 + 92 = 82

82 + 22 = 68

62 + 82 = 100

12 + 02 + 02 = 1

1. 题解
2. 思路

判断一个正整数是否为“快乐数”，按照题目的说法，数中每个位数的平方相加，不断迭代，直到和为1，则为快乐数；如果持续地迭代，结果依然不为1，则不是快乐数。

那么，当结果持续不为1时，它会一直迭代下去，是不会停止的，怎么判断这种情况呢？

可以通过简单地找规律得出，当结果持续不为1时，它会陷入一个循环之中，它会重复地出现之前的数，像一个闭环一样。因此，我们通过判断之前已经出现过的数是否再次出现即可得知，该数是否陷入了死循环之中。如果陷入了死循环中，就一定不是快乐数。

1. 代码实现

Java：

class Solution {

    public boolean isHappy(int n) {

        int sum = n;

        // sum 不为1，进入循环， 为1，直接输出

        Set<Integer> set = new HashSet<>();

        while(sum != 1){

            int digit = 0;

            while(sum != 0){

                digit += (sum % 10) \* (sum % 10);

                sum /= 10;

            }

            sum = digit;

            //如果一个数字不是快乐数，那么它一定会处于一个循环之中，我们需要判断该数循环链中的数是否已经出现过

            if(set.contains(sum)){

                return false;

            }else{

                set.add(sum);

            }

        }

        return sum == 1 ? true: false;

    }

}