202-快乐数

题述

202. 快乐数

难度 简单 793

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。
- 如果这个过程 **结果为** 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` 。

示例 1：

输入： n = 19
输出： true
解释：
$1^2 + 9^2 = 82$
$8^2 + 2^2 = 68$
$6^2 + 8^2 = 100$
$1^2 + 0^2 + 0^2 = 1$

示例 2：

输入： n = 2
输出： false

思路

这道题目看上去貌似一道数学问题，其实并不是！

题目中说了会 **无限循环**，那么也就是说**求和的过程中，sum会重复出现，这对解题很重要！**

当我们遇到了要快速判断一个元素是否出现集合里的时候，就要考虑哈希法了。

所以这道题目使用哈希法，来判断这个sum是否重复出现，如果重复了就是return false， 否则一直找到sum为1为止。

判断sum是否重复出现就可以使用unordered_set。

还有一个难点就是求和的过程，如果对取数值各个位上的单数操作不熟悉的话，做这道题也会比较艰难。

题解

```
class Solution {
public:
    //取各个位置上的数字之和
    int getSum(int n)
    {
        int sum=0;
        while(n)
        {
            //依次取各个位置上的数字平方和
            sum+=(n%10)*(n%10);
            n/=10;
        }
        return sum;
    }

    bool isHappy(int n)
    {
        //快乐数  重复置为每个位置上的数字的平方和，无限循环该过程，直至变为1
        //例子：19就是一个快乐数

        //这道题的求和过程中，sum会重复出现，这是破题依据！
        //前面说过，当我们遇到了要快速判断一个元素是否出现在集合里面的时候，我们需要考虑哈希算法
        //判断sum是否重复出现时，我们可以使用unordered_set

        unordered_set<int> set;
        while(1)
        {
            int sum=getSum(n);
            if(sum==1)
            {
                return true;
            }

            //如果这个sum曾经出现过，说明已经陷入了无限循环了，这时我们就可以return false
            if(set.find(sum)!=set.end())
            {
                return false;
            }
            else
            {
                set.insert(sum);
            }
            n=sum;
        }

    }
};
```

提交结果	执行用时	内存消耗	语言	提交时间	备注
通过	4 ms	6.1 MB	C++	2022/01/27 11:25	
编译出错	N/A	N/A	C++	2022/01/27 11:23	

思考

unordered_set的知识我们前面提到过

find如果找不到的话会返回end地址