1. Happy Number

Write an algorithm to determine if a number n is “happy”.

A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it **loops endlessly in a cycle** which does not include 1. Those numbers for which this process **ends in 1** are happy numbers.

Return True if n is a happy number, and False if not.

**Example:**

Input: 19  
Output: true  
Explanation:   
12 + 92 = 82  
82 + 22 = 68  
62 + 82 = 100  
12 + 02 + 02 = 1

对于happy number的操作，最终的结局可能是：

* 收敛到1
* 陷入循环
* n不断增大直到无穷大

但是第三种情况肯定不会发生

|  |  |  |
| --- | --- | --- |
| Digits | Largest | Next |
| 1 | 9 | 81 |
| 2 | 99 | 162 |
| 3 | 999 | 243 |
| 4 | 9999 | 324 |
| 13 | 9999999999999 | 1053 |

从表中可以看出，三位数最大的下一位是243，四位数最大的下一位小于999，因此最终也会小于243

**解1** hashset判断是否出现环

class Solution {  
public:  
 bool isHappy(int n) {  
 unordered\_map<int, bool>mp;  
 while(!mp[n]){  
 mp[n] = true;  
 n = happy(n);  
 if(n == 1)return true;  
 }  
 return false;  
 }  
 int happy(int n){  
 int res = 0;  
 while(n){  
 int tmp = n % 10;  
 res += tmp \* tmp;  
 n /= 10;  
 }  
 return res;  
 }  
};

**解2** Floyd环检测算法（龟兔赛跑）

class Solution {  
public:  
 unordered\_map<int, int>mp;  
 bool isHappy(int n) {  
 int faster = happy(happy(n)), slower = happy(n);  
 while(faster != 1 && faster != slower){  
 faster = happy(happy(faster));  
 slower = happy(slower);  
   
 }  
 return faster == 1;  
 }  
 int happy(int n){  
 if(mp[n])return mp[n];  
 int res = 0;  
 while(n){  
 int tmp = n % 10;  
 res += tmp \* tmp;  
 n /= 10;  
 }  
 return mp[n] = res;  
 }  
};