Happy Numbers

202. Happy Number

Easy 9016 1174 Add to List Share

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.
- 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.

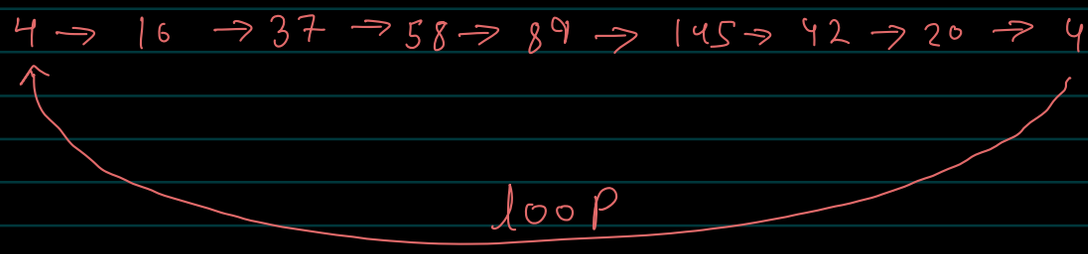
Return `true` *if* `n` *is a happy number, and* `false` *if not.*

Example 1:

```
Input: n = 19
Output: true
Explanation:
1^2 + 9^2 = 82
8^2 + 2^2 = 68
6^2 + 8^2 = 100
1^2 + 0^2 + 0^2 = 1
```

If number is not happy number then there will be endless loop

like



so we there are 2 possibities

1) N will end up with 1

2) N will end up with N again

First approch:

store elements in set and if we found any repeating element we will return false

```
n = 19
loop(n != 1)
        if (st.find(n) != st.end())
                return false
        st.insert(n)
```

```
int sum
while (n != 0){
    sum = sum + pow(n%10, 2)
    n = n/10;
}
n = sum;
}
return true;
```

Dry run:

n = 19

while(n != 1

st.find(i) != st.end() False

insert(19) False

68
82
19

sum = 0
while (n != 0) {
sum = 0 + 9*9 = 81 ⇒ sum = 81 + 1 = 82
n = 19/10 => 1 ⇒ n = 0

n = sum

n = 82
while (n != 0)

sum = 0 + 4 ⇒ ⇒ sum = 64 + 4 = 68
n = 82/10 => 8

n = sum // 68

n = 68

sum = $8^2 + 8^2$ ⇒ sum = $1^2 + 0^2 + 0^2$
= 36 + 64 = 1
= 100
n = 0

while (n != 1)
n = 1

return true

Dry run:

n = 4

20
42
145
58
37
16
4

while (n != 1)

if (st.find(n) != st.end())

return False

insert(n)

while (n != 0)

sum = 0 + 4^2 = 16, n = 0

n = 16

sum = 1^2 + 6^2 = 37

n = 37

sum = 3^2 + 7^2 = 9 + 49 = 58

n = 58

sum = 5^2 + 8^2 = 25 + 64 = 89

n = 89

sum = 64 + 81 = 145

n = 145

n = 1^2 + 4^2 + 5^2 = 1 + 16 + 25 = 42

n = 42

n = 4^2 + 2 = 16 + 4 = 20

n = 20

n = 2^2 + 0^2 = 4

n = 4 Found in Set

return false

Second Approch :-

Floyd cycle detection algorithm :-

we know that there is a cycle in code if it is not happy number

so we will have 2 pointer slow & fast

n = 19

slow = n, fast = n.

First sum

slow = 1^2+9^2

fast = (1^2+9^2)

fast = (8^2+2^2)

while (slow != Fast) ? How

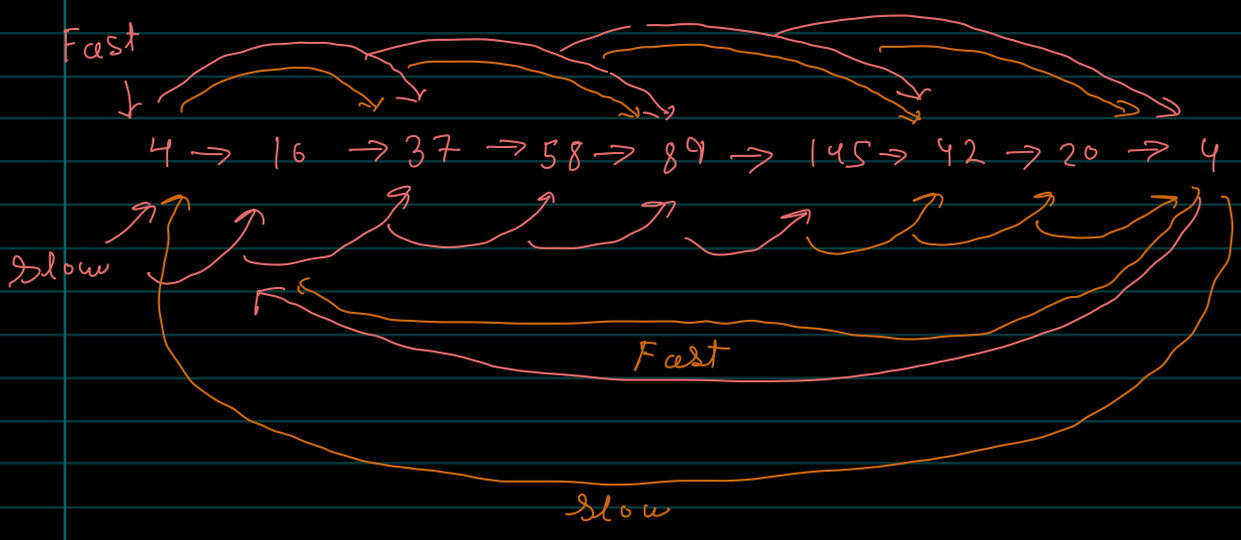
if we got 1 then squaring 1 will always be one

we will use helper / sum function

n = 4

slow = 16	slow = 37
fast = 37	fast = 89

145
58
37
16
4



fast == slow at 4

but slow is != 1

hence return false

```cpp
class Solution {
public:
    bool isHappy(int n) {
        unordered_set<int> st;
        while(n!=1)
        {
            if(st.find(n) != st.end())
            {
                return false;
            }

            st.insert(n);
            int sum=0;
            while(n!=0){
                sum=sum+pow(n%10,2);
                n=n/10;
            }
            n=sum;
        }
        return true;

    }
};
```

```cpp
class Solution {
public:
    int help(int n){
        int ans=0;
        while(n){
            int temp=n%10;
            ans=ans+(temp*temp);
            n/=10;
        }
        return ans;
    }
    bool isHappy(int n) {
        int slow=n,fast=n;
        do{
            slow=help(slow);
            fast=help(help(fast));
        }while(slow!=fast);
        return slow==1;

    }
};
```