Happy Number (202)

Use Floyd's cycle detection algorithm. If there is a cycle which does not terminate in 1 ~~[illegible]~~ when we try to move one ~~[illegible]~~ variable one step at a time and another variable two steps at a time, this mean ~~[illegible]~~ that number is not happy.

For an unhappy number, there would surely exist a cycle. because even if n = 999,999,999, the sum of squares is 729. which would then go down rapidly.

CODE

```
int digitSquareSum (int n) {
        int sum = 0;
        while (n > 0) {
              int x = n % 10;
              sum = sum + x * x;
              n = n/10;
        }
       return sum.
```

⇒

```
bool isHappy (int n) {
    int slow = digitSquareSum(n)
    int fast = digitSquareSum(digitSquareSum(n))

    while (slow != fast){
        slow = digitSquareSum(slow);
        fast = digitSquareSum(digitSquareSum(fast
    }
    return slow == 1;
}
```

Time complexity → can't say, depends on numbers.

space complexity → O(1).