Happy Number

For this number we must repeatedly replace the number with the sum of squares of its digits and check if we eventually reach 1 (happy) or fall into a cycle (unhappy).

Key Insight:

- If the process enters a cycle, it will never reach 1.
- Known property: The cycle for unhappy numbers always includes 4 (but we can detect cycles generally).
- Two main methods:
  1. Hash set to track visited numbers.
  2. Floyd's Cycle Detection (fast & slow pointers) to detect loops.

Approach 1: Hash Set (Simple)

1. While n is not 1 and not seen before:
   - Add n to visited set.
   - Replace n with sum of squares of its digits.
2. If n == 1, return true; else false

Approach 2: Floyd's Cycle Detection (O(1) space)

- Use two pointers:
  - slow = sumOfSquares(n)
  - fast = sumOfSquares(sumOfSquares(n))
- Loop until slow == fast:
  - If slow == 1 or fast == 1 → happy number.
  - Else continue.
- If loop ends and value isn't 1 → not happy.

Complexity:

- Time: O(log n) per iteration (digit processing), total bounded since numbers decrease fast.
- Space: O(1) for Floyd's method, O(k) for hash set method (k = number of iterations).