# Documentation: Happy Number Problem Solution

## Problem Overview:

- The problem is to determine whether a number is a "happy number." A happy number is a number that, through an iterative process of summing the squares of its digits, eventually results in 1. If during this process the number enters a cycle that does not include 1, the number is classified as "unhappy."

## Steps to Solve the Problem:

1. **Initialization:**

- Start with a given positive integer n.
- Initialize a set to track numbers that have already been seen during the process to detect cycles.

2. **Calculating the Sum of the Squares of the Digits:**

- For any number n, extract its individual digits.
- Square each digit and sum them together.
- Replace n with this sum.

3. **Iterative Process:**

- *Repeat the process of summing the squares of the digits until:*
  - The number becomes 1 (indicating that the number is happy).
  - A previously seen number is encountered (indicating a cycle, meaning the number is unhappy).

4. **Cycle Detection:**

- To avoid infinite loops caused by cyclic behavior, maintain a set that stores each intermediate number encountered during the process.
- If the current number has already been encountered (exists in the set), the number is considered unhappy, and the function returns False.

5. **Termination Conditions:**

- If the number becomes 1, return True because it is a happy number.
- If a cycle is detected (i.e., a repeated number appears), return False because the number will never reach 1.

## Example:

1. **Happy Number Example (n = 19):**

- Start with n = 19.
- ***Step 1:*** *Calculate the sum of the squares of the digits of 19:*
  - $1^2 + 9^2 = 1 + 81 = 82$.
- ***Step 2:*** *Calculate the sum of the squares of the digits of 82:*
  - $8^2 + 2^2 = 64 + 4 = 68$.
- ***Step 3:*** *Calculate the sum of the squares of the digits of 68:*
  - $6^2 + 8^2 = 36 + 64 = 100$.
- ***Step 4:*** *Calculate the sum of the squares of the digits of 100:*
  - $1^2 + 0^2 + 0^2 = 1$.
- Since the result is 1, 19 is a happy number.

2. **Unhappy Number Example (n = 2):**

- Start with n = 2.
- ***Step 1:*** *Calculate the sum of the squares of the digits of 2:*
  - $2^2 = 4$.
- ***Step 2:*** *Calculate the sum of the squares of the digits of 4:*
  - $4^2 = 16$.
- ***Step 3:*** *Calculate the sum of the squares of the digits of 16:*
  - $1^2 + 6^2 = 1 + 36 = 37$.
- ***Step 4:*** *Calculate the sum of the squares of the digits of 37:*
  - $3^2 + 7^2 = 9 + 49 = 58$.
- ***Step 5:*** *Calculate the sum of the squares of the digits of 58:*
  - $5^2 + 8^2 = 25 + 64 = 89$.
- This process continues, and the numbers will start repeating, forming a cycle (e.g., 89 → 145 → 42 → 20 → 4 → 16 → 37 → ...). Since the process never reaches 1, 2 is an unhappy number.

## Detailed Breakdown:

1. **Helper Function:**
   - A helper function is used to calculate the sum of the squares of the digits of a number. This is done by iterating over each digit of the number, squaring it, and accumulating the result.

2. **Cycle Prevention:**
   - To prevent infinite loops due to cycles, a set is employed to keep track of numbers that have already been seen during the process.
   - Each new number is checked against this set. If the number is already in the set, a cycle is detected, and the function can safely conclude that the number is unhappy.

### 3. Set Operations:

- The use of a set ensures that checking whether a number has been seen before is efficient, as set operations (like adding elements and checking for membership) have an average time complexity of O(1).

### 4. Edge Cases:

- Numbers like 1 are inherently happy, as the sum of the squares of its digits is 1.
- Very small numbers (such as 2 or 3) that quickly form cycles should be handled by detecting repetitions using the set.

## Time Complexity:

- Each step involves summing the squares of the digits of the current number. The number of digits is proportional to the logarithm of the number (log base 10).
- Therefore, the time complexity of each step is O(log n), where n is the size of the current number.
- The number of steps taken before detecting a cycle or reaching 1 is bounded, as the numbers reduce in size as the process proceeds. Thus, the overall time complexity is approximately O(log n).

## Space Complexity:

- The space complexity is O(k), where k is the number of distinct numbers encountered before detecting a cycle or reaching 1. In practice, this space requirement is small due to the set storing previously seen numbers.

## Constraints:

- The input number n is constrained between 1 and $2^{31}$ 1.
- This means that the number will have at most 10 digits, and thus, the number of iterations before detecting a cycle is reasonably small.

## Conclusion:

- This solution effectively determines whether a number is happy by using an iterative process that calculates the sum of the squares of its digits and employs a set to detect cycles. By following this process, the algorithm can handle all input numbers within the given constraints and return the correct result efficiently.