October 15

Problem 1: Happy Number

Problem Statement:

Write an algorithm to determine if a number n is "happy."
A **happy number** is 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 that 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.

Link to problem:

https://leetcode.com/problems/happy-number/description/

Example 1:

- Input: n = 19
- Output: true
- Explanation:

  $1^2 + 9^2 = 82$

  $8^2 + 2^2 = 68$

  $6^2 + 8^2 = 100$

  $1^2 + 0^2 + 0^2 = 1$

Example 2:

- Input: n = 2
- Output: false

Solution:

```
import java.util.HashSet;
import java.util.Set;

class Solution {
   public boolean isHappy(int n) {
```

```
        Set<Integer> seen = new HashSet<>();

        while (n != 1 && !seen.contains(n)) {
            seen.add(n);  // Track numbers to detect cycles
            n = getNext(n);  // Get the sum of squares of digits
        }

        return n == 1;
    }

    private int getNext(int n) {
        int totalSum = 0;
        while (n > 0) {
            int digit = n % 10;
            totalSum += digit * digit;
            n /= 10;
        }
        return totalSum;
    }
}
```

Explanation:

- We use a set to track numbers we have already seen. If we see the same number again, it means we are in a cycle, and the number is not happy.
- For each number, we calculate the sum of the squares of its digits. If we reach 1, the number is happy; otherwise, we return false.

Time Complexity: O(log n), as the number of digits reduces in each step**.**

Space Complexity: O(log n), to store the set of seen numbers.

Problem 2: Longest Consecutive Sequence

Problem Statement:

Given an unsorted array of integers nums, return the length of the longest consecutive elements sequence.

You must write an algorithm that runs in O(n) time.

You must write an algorithm that runs in O(log n) time.

Link to problem:

https://leetcode.com/problems/longest-consecutive-sequence/description/

Example 1:

- Input: nums = [100, 4, 200, 1, 3, 2]
- Output: 4
- Explanation: The longest consecutive elements sequence is [1, 2, 3, 4]. Therefore its length is 4.

Example 2:

- Input: nums = [0,3,7,2,5,8,4,6,0,1]
- Output: 9

Solution:

```
import java.util.HashSet;
import java.util.Set;

class Solution {
    public int longestConsecutive(int[] nums) {
        Set<Integer> numSet = new HashSet<>();
        for (int num : nums) {
            numSet.add(num);
        }

        int longestStreak = 0;

        for (int num : numSet) {
            // Start of a sequence (num-1 is not in the set)
            if (!numSet.contains(num - 1)) {
                int currentNum = num;
                int currentStreak = 1;

                // Count the length of the streak
                while (numSet.contains(currentNum + 1)) {
                    currentNum += 1;
                    currentStreak += 1;
                }

                longestStreak = Math.max(longestStreak, currentStreak);
            }
        }

        return longestStreak;
    }
}
```

Explanation:

- We first add all elements to a set for O(1) lookups.
- For each number, if it is the start of a sequence (i.e., num - 1 is not present), we count the length of the consecutive sequence starting from that number.
- We keep track of the longest streak and return it at the end.

Time Complexity: O(n), as we traverse the array once and perform constant-time operations.

Space Complexity: O(n), for storing elements in the set.
