Longest Consecutive Sequence

2nd Review	@January 18, 2026
Date Solved	@January 14, 2026
Difficulty	Medium
# Review	1
Status	Revise
Topic/Pattern	Array and String

Link → https://neetcode.io/problems/longest-consecutive-sequence/history

Problem

- You are given an unsorted integer array `nums`.
- Return the **length of the longest consecutive sequence**.
- A consecutive sequence means numbers increase by `+1`.
- The sequence **does not need to be contiguous** in the array.
- You must write an algorithm that runs in **O(n)** time.

Example

Input	Output	Reason
`[100,4,200,1,3,2]`	`4`	Longest sequence is `[1,2,3,4]`
`[2,20,4,10,3,4,5]`	`4`	Longest sequence is `[2,3,4,5]`
`[0,3,2,5,4,6,1,1]`	`7`	Longest sequence is `[0,1,2,3,4,5,6]`

Approaches

1. Brute Force — Check Every Possible Sequence

```
deflongestConsecutive(nums):
    longest =0

for numin nums:
        current = num
        length =1
while current +1in nums:
            current +=1
            length +=1
        longest =max(longest, length)

return longest
```

- **Time:** O(n²) — checking existence inside a loop
- **Space:** O(1)
- **Notes:** Works logically but **too slow** for large inputs

---

2. Sorting Based Approach

```
deflongestConsecutive(nums):
ifnot nums:
return0

    nums =sorted(set(nums))
    longest =1
    current =1

for iinrange(1,len(nums)):
if nums[i] == nums[i -1] +1:
            current +=1
else:
            current =1
        longest =max(longest, current)
```

```
return longest
```

- **Time:** O(n log n) — sorting required
- **Space:** O(n)
- **Notes:** Cleaner logic but **does not meet O(n) requirement**

---

3. Optimal — HashSet + Sequence Start Detection (Recommended)

```
deflongestConsecutive(nums):
    numSet =set(nums)
    longest =0

for numin numSet:
# only start counting if num is the start of a sequence
if num -1 notin numSet:
            current = num
            length =1

while current +1in numSet:
                current +=1
                length +=1

            longest =max(longest, length)

return longest
```

- **Time: O(n)** — each number visited once
- **Space:** O(n) — hash set for fast lookup
- **Notes:**

  ✅ Handles duplicates

  ✅ Handles unordered input

  ✅ Interview-preferred solution

Summary

Approach	Time	Space	Notes
Brute Force	O(n^2)	O(1)	Too slow
Sorting	O(n log n)	O(n)	Breaks time constraint
HashSet + Start Detection	**O(n)**	**O(n)**	✅ Optimal

Edge Cases

Input	Output	Reason
`[]`	`0`	Empty array
`[5]`	`1`	Single number
`[1,1,1]`	`1`	Duplicates only
`[1,3,5]`	`1`	No consecutive numbers

Tip

- This problem is a **sequence problem**, not a subarray problem.
- Always detect the **start of a sequence** using:

```
if num -1notinset
```