
Problem Statement:

The function longest_consecutive_sequence takes a list of integers as input and returns the length of the longest consecutive sequence found within that list. For example, if the input is [100, 4, 200, 1, 3, 2], the function identifies the consecutive sequence [1, 2, 3, 4] and returns 4.

Solution 1

The provided solution works by first converting the list into a set to enable O(1) lookups. It then iterates over each number in the original list. For each number, if the preceding number (`num - 1`) is not in the set, it marks the start of a new sequence. A while loop is then used to count how many consecutive numbers follow, updating the longest sequence length accordingly.

```python
def longest_consecutive_sequence(nums: List[int]) -> int:
    if not nums: return 0
    longest = 0
    num_set = set(nums)

    for num in nums:
        if num - 1 not in num_set: # no prior number means start of sequence
            streak = 1

            while num + 1 in num_set:
                streak += 1
                num += 1

        longest = max(longest, streak)

    return longest
```

Step-by-Step Breakdown

1. **Input:**

   A list of integers (e.g., [100, 4, 200, 1, 3, 2]).

2. **Initial Check & Set Creation:**

   Step: If the list is empty, the function returns 0 immediately. Otherwise, convert the list to a set for fast lookups.

```
1. if not nums: return 0
2.
3. num_set = {100, 4, 200, 1, 3, 2}   →   {1, 2, 3, 4, 100, 200}
4.
```

   Explanation:

   Converting to a set ensures that checking if a number exists (num in num_set) is done in constant time.

3. **Iteration Over Each Number:**

- **Step:** The function loops over every number in the original list.

```
1. For each number in [100, 4, 200, 1, 3, 2]:
2.    └── Process the number
```

4. **Identifying the Start of a Sequence:**

   - **Step:**
     For each number, check if num - 1 is not in the set. This indicates the number is the beginning of a consecutive sequence.
     - For 100: Check if 99 is in the set. It is not, so start a new sequence.
     - For 4: Check if 3 is in the set. Since 3 is present, skip starting a sequence here.

```
1. Number = 100
2.    └── 100 - 1 = 99 → 99 not in set → Start sequence [100]
3.
4. Number = 4
5.    └── 4 - 1 = 3 → 3 in set → Skip (Not the start)
6.
```

5. **Counting the Consecutive Sequence:**

   - **Step:**
     If the number is the start of a sequence, initialize a counter (streak = 1) and then use a while loop to check for consecutive numbers.
       - num = 1 → Check 1 - 1 = 0 → Not in set → Start a new sequence.
       - Set streak = 1.

```
1. While (num + 1) is in num_set:
2.    1. num = 1: Check if 2 is in set? Yes → streak becomes 2, update num to 2.
3.    2. num = 2: Check if 3 is in set? Yes → streak becomes 3, update num to 3.
4.    3. num = 3: Check if 4 is in set? Yes → streak becomes 4, update num to 4.
5.    4. num = 4: Check if 5 is in set? No → End loop.
6.
7. Sequence starting at 1: [1 → 2 → 3 → 4]
8. Final streak for this sequence: 4
9.
```

- **Other Numbers:**
  - For 200:
    - 200 - 1 = 199 is not in the set, so sequence starts at 200.
    - Check 201 in set → Not found → Sequence length = 1.
  - For 100 (as seen earlier):
    - Sequence is just [100] → Length = 1.

6. **Updating the Longest Sequence:**

- **Step:**
  After processing each starting number, update the variable longest with the maximum streak found so far.

```
1. longest = max(longest, streak)
```

- **Flow Example:**
  - After processing 100: longest = max(0, 1) = 1
  - After processing 1: longest = max(1, 4) = 4
  - Other numbers do not change the longest value.

7. **Output:**

- **Final Step:**
  Return the longest streak found.

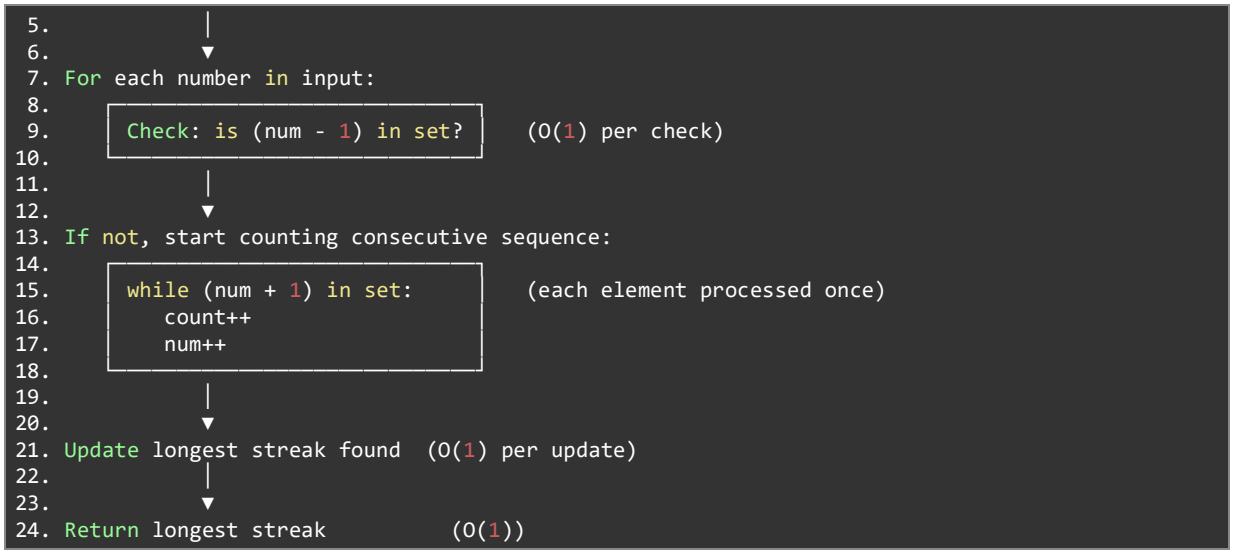
```
1. return longest → returns 4
```

Efficiency Details for Solution 1:

- **Time Complexity:**
  - **Step 1:** Converting the input list to a set takes O(n) time.
  - **Step 2:** Iterating over each number in the list is O(n).
  - **Step 3:** For each number that starts a sequence, the while loop only processes each number once across all iterations.
  - **Overall:** The algorithm runs in O(n) time on average.
- **Space Complexity:**
  - The set created from the input list uses O(n) extra space.

Visual Flow Diagram

```
1. Input: [100, 4, 200, 1, 3, 2]
2.            |
3.            ▼
4. Convert List → Set: {1, 2, 3, 4, 100, 200}   [O(n) time, O(n) space]
```


For each number in input:
Check: is (num - 1) in set?
(O(1) per check)
If not, start counting consecutive sequence:
while (num + 1) in set:
count++
num++
(each element processed once)
Update longest streak found (O(1) per update)
Return longest streak (O(1))