Daily Coding Problem #99

Problem

This problem was asked by Microsoft.

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

For example, given `[100, 4, 200, 1, 3, 2]`, the longest consecutive element sequence is `[1, 2, 3, 4]`. Return its length: `4`.

Your algorithm should run in `O(n)` complexity.

Solution

We can see that if the array of integers was sorted, we'd be able to find the longest consecutive sequence fairly easily. First, we sort the array in ascending order. Then, we traverse the array, keeping the count of the current sequence. If the next element is one higher than the current element, then we increment the count. Otherwise, we start the count over at 1. We simply return the maximum count we've seen overall. The overall run-time of this solution would be `O(nlogn)`, since we have to sort the input array. Depending on the sorting implementation and whether or not we are able to modify the array, the space complexity would be `O(n)` or `O(1)`.

We can improve our solution by using extra space to cache the bounds of sequences we've seen so far. If we get a new number, then we start with a sequence length of 1, If we get a number we've seen already, we can ignore it since none of the bounds should change.

```
def longest_consecutive(self, nums):
```

```
    max_len = 0
    bounds = dict()
    for num in nums:
        if num in bounds:
            continue
        left_bound, right_bound = num, num
        if num - 1 in bounds:
            left_bound = bounds[num - 1][0]
        if num + 1 in bounds:
            right_bound = bounds[num + 1][1]
        bounds[num] = left_bound, right_bound
        bounds[left_bound] = left_bound, right_bound
        bounds[right_bound] = left_bound, right_bound
        max_len = max(right_bound - left_bound + 1, max_len)

    return max_len
```

This solution has a time complexity of O(N), and a space complexity of O(N).