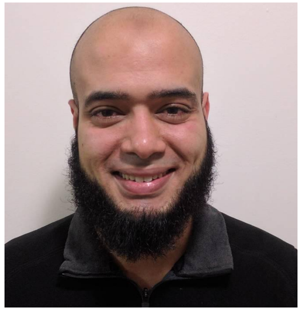
Cracking Coding Interviews

Longest Consecutive Sequence

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Leetcode 128 - Longest Consecutive Sequence

- Given an **unsorted** array of integers, return the length of the longest consecutive elements sequence.
- Input ⇒ Output
  - [100, 4, 200, 1, 3, 2] ⇒ 4
    - Sequence [4, 1, 3, 2] when sorted [1, 2, 3, 4] ⇒ consecutive elements
  - [0, 3, 7, 2, 5, 8, 4, 6, 1] ⇒ 9
    - The whole array when sorted is consecutive elements [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
- Signature
  - C++: int longestConsecutive(vector<int>& nums)
  - Python: def longestConsecutive(self, nums: List[int]) -> int
  - Java: public int longestConsecutive(int[] nums)

Your turn

- Ask the right questions, if any, and state your assumptions
- Develop some test cases

Questions & test cases

- The problem seems clear
  - A sequence => if sorted => consecutive items
  - If you feel confused, take your time to confirm what is being requested by the problem
- Several good cases are given already!

Your turn

- Can we approach it with brute-force? If so, how?

Your turn

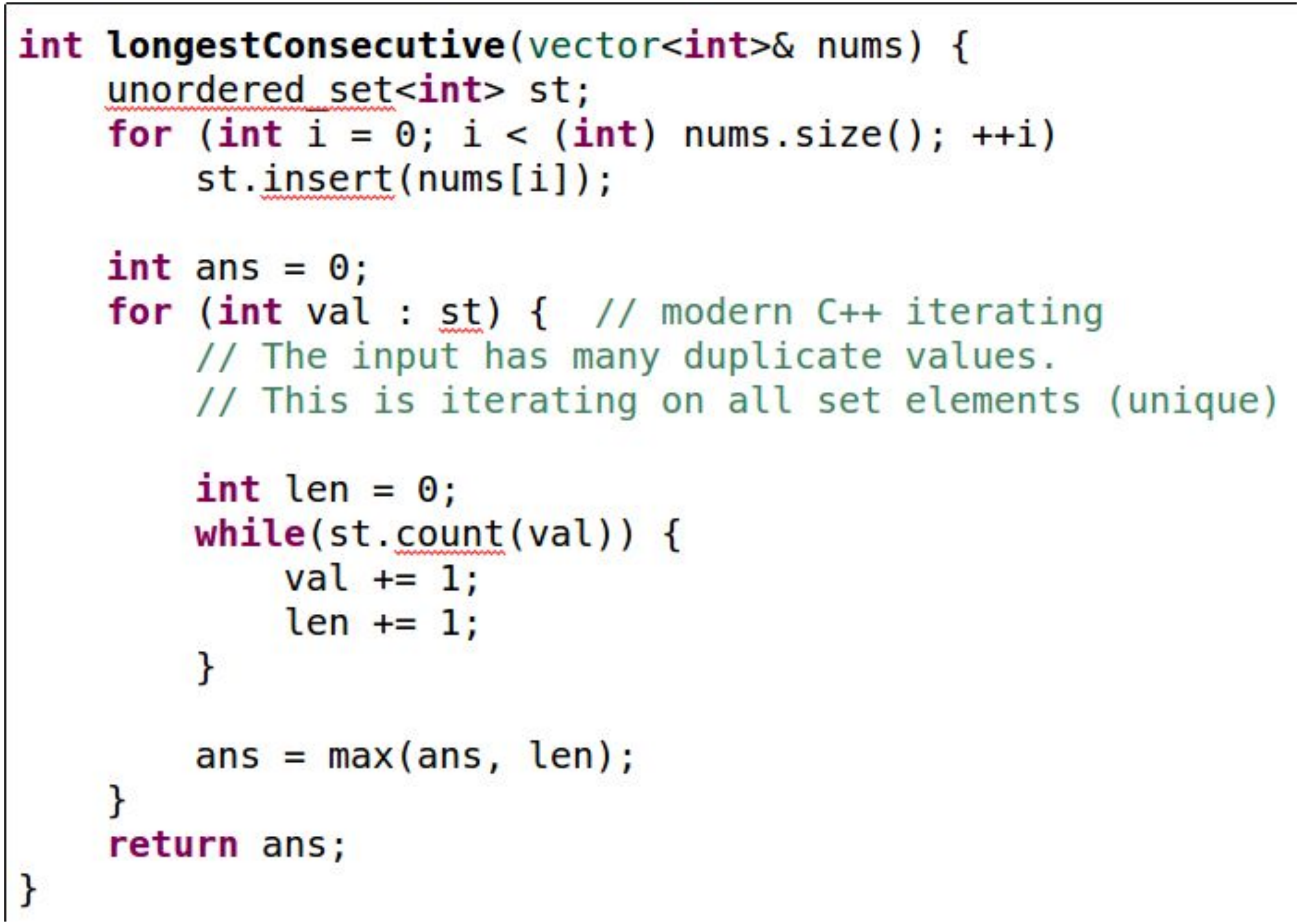
- Can we approach it with brute-force? If so, how?
    - Not direct
- Hint: Given a number, can you use brute force to deduce the longest consecutive sequence from that number?
- E.g. for [100, 4, 200, 1, 3, 2] and value 2: can you get the answer, which is 3 [2, 3, 4]?

- Start with the initial value: keep checking if the increment to it exists in the array. Stop once you can no longer find it.
    - 2 ⇒ there. Increment
    - 3 ⇒ there. Increment
    - 4 ⇒ there. Increment
    - 5 ⇒ not there. Stop

Your turn

- Now: for every number in the list compute its longest sequence and maximize
- How to check if a number is there?
  - Slow: loop to search in O(n). Or, as we learned, we can do this quickly with a hash table

Brute-force it!



```cpp
int longestConsecutive(vector<int>& nums) {
    unordered_set<int> st;
    for (int i = 0; i < (int) nums.size(); ++i)
        st.insert(nums[i]);

    int ans = 0;
    for (int val : st) {  // modern C++ iterating
        // The input has many duplicate values.
        // This is iterating on all set elements (unique)

        int len = 0;
        while(st.count(val)) {
            val += 1;
            len += 1;
        }

        ans = max(ans, len);
    }
    return ans;
}
```

Brute-force it: Optimizations

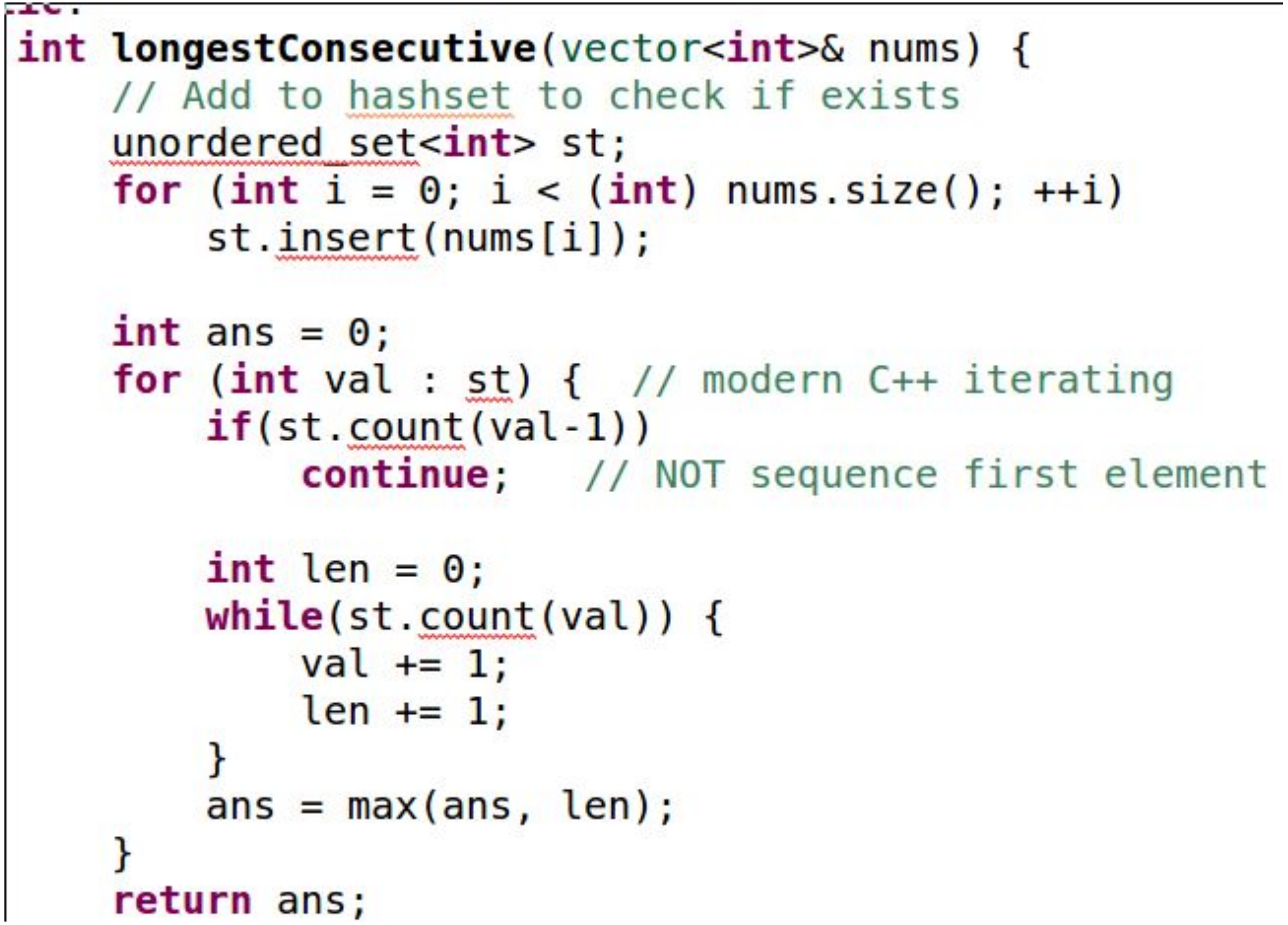
- The last code was O(n^2).
- We try every number
  - Compute the longest sequence
- Can we optimize more?
- We waste time trying every number in the sequence
- Assume input is [0, 1, 2, 3, 4, 5]
  - We get for 0 [0, 1, 2, 3, 4, 5]
  - We get for 1 [1, 2, 3, 4, 5]
  - We get for 2 [2, 3, 4, 5]
- But all are part of the SAME sequence. We need to process the whole sequence once!

Brute-force it: Optimizations

- There are different ways to do that
- Here is an elegant way
- How can we recognize whether this number is the first number in the sequence?
- Always analyze the test cases
  - For [100, 4, 200, 1, 3, 2]
  - How to know in O(1) that 1 is the begin of a sequence but 2 is not?
  - If X is the first element in the sequence, then X-1 won’t be in the sequence
  - What is before 1? 0
  - Is 0 in the list? No. Then 1 is the first number in its sequence
  - What about 3? Before it is 2. Is 2 in the list? Yes. 3 is in the middle of a sequence!

Optimized

- O(n) time & memory



```
int longestConsecutive(vector<int>& nums) {
    // Add to hashset to check if exists
    unordered_set<int> st;
    for (int i = 0; i < (int) nums.size(); ++i)
        st.insert(nums[i]);

    int ans = 0;
    for (int val : st) {  // modern C++ iterating
        if(st.count(val-1))
            continue;   // NOT sequence first element

        int len = 0;
        while(st.count(val)) {
            val += 1;
            len += 1;
        }
        ans = max(ans, len);
    }
    return ans;
```

Another direction

- Another trivial solution is to sort numbers in O(nlogn)
    - E.g. [1, 2, 3, 4, 100, 200, 201, 202, 500]
    - Now process them, and group consecutive numbers together
        - 1, 2, 3, 4
        - 100, 200
        - 200, 201, 202
        - 500
- This can't be improved to O(n). Take care to avoid getting trapped yourself with a method that doesn't improve upon your previous solution
    - Consider other directions when you are stuck
    - Ask for a few hints if needed

Interesting idea

- How can u formulate this problem as a graph problem?

Interesting idea

- How can u formulate this problem as a graph problem?
- Each consecutive sequence is actually a chain of nodes
- So we can build the tree and then find the maximum connected component, e.g. using union-find
- How to build the graph Edges?
    - For number X
        - If X+1 exists then we have edge (X, X+1)
        - If X-1 exists then we have edge (X-1, X)
- Find solutions in your languages in the discussion tab. C++ Code
- *It is rare in interviews when you reinterpret the problem in another domain*

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."