Given a string, find the length of the **longest substring** without repeating characters.

**Examples:**

Given "abcabcbb", the answer is "abc", which the length is 3.

Given "bbbbb", the answer is "b", with the length of 1.

Given "pwwkew", the answer is "wke", with the length of 3. Note that the answer must be a **substring**, "pwke" is a *subsequence* and not a substring.

Solution:

Solution to this problems is done in O(n) run time and O(1) space. This can be done in a greedy approach. Start with the basic brute force; for every i ≤ j, check if the substring between i and j don’t have repeating characters (by using maps to count the character occurrences) and return the maximum of those lengths. This is O(n^2) solution. How to optimize this ? let’s start i = j = 0. Increment j till any of the character occurrence is > 1. Unlike brute force solution, we stop at this point because it makes no sense to increment j further as there won’t be any non-repeating substring anyway. Now, we’ll increment i (accordingly decrement the character occurrences from map) till the character at j position’s count becomes 1 again. At this stage, we’re sure that none of the characters between i and j have more than one occurrence (meaning non-repeating substring). We’ll repeat the process of incrementing j (and accordingly incrementing the character occurrence count) till the occurrence is < 2. We’ll stop once j I reaches n.