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.

看到题目里有判重的问题，首先想到hashset。问题的复杂的度就会降为O(2n)

有更优的方法在做题时没有想到，算法代码如下：

public class Solution {
public int lengthOfLongestSubstring(String s) {
int n = s.length(), ans = 0;
Map<Character, Integer> map = new HashMap<>(); // current index of character
// try to extend the range [i, j]
for (int j = 0, i = 0; j < n; j++) {
if (map.containsKey(s.charAt(j))) {
i = Math.max(map.get(s.charAt(j)), i);
}
ans = Math.max(ans, j - i + 1);
map.put(s.charAt(j), j + 1);
}
return ans;
}
}

关键在于 i = Math.max(map.get(s.charAt(j)), i);增加了一个判断，免去了清空hashmap那一步