3. Longest substring without repeating characters:

I/P: s = "abcabcbb"

O/P: 3

Explanation: The answer is "abc", with length of 3.

Need to find the longest substring that has no repeated characters.

A substring means characters must be continuous (not skipping in between).

Brute force:

1. use two loops to generate all substrings
2. for each substring, check if all characters are unique.
3. Track the maximum length.

```
class Solution {
    public int lengthOfLongestSubstring(String s) {
        int maxLen = 0;
        for (int i = 0; i < s.length(); i++) {
            boolean[] seen = new boolean[256];
            int currentLen = 0;
            for (int j = i; j < s.length(); j++) {
                char ch = s.charAt(j);
                if (seen[ch]) {
                    break;
                } else {
                    seen[ch] = true;
                    currentLen++;
                    maxLen = Math.max(maxLen, currentLen);
                }
            }
        }
        return maxLen;
    }
}
```

Time → $O(n^2)$ space → $O(1)$.

optimized:

```
import java.util.HashSet;
class Solution {
    public int lengthOfLongestSubstring(String s) {
        HashSet<Character> set = new HashSet<>();
        int left = 0, maxLen = 0;
    for(int right = 0; right < s.length(); right++) {
        char ch = s.charAt(right);
    while(set.contains(ch)) {
        set.remove(s.charAt(left));
        left++;
    }
        set.add(ch);
        maxLen = Math.max(maxLen, right - left + 1);
    }
    return maxLen;
    }
}.
```

Time → O(n) space → O(1).