# LeetCode 3: Longest Substring Without Repeating Characters

## Problem Statement:

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

## Intuition

* Use a **sliding window** to represent the current substring without duplicates.
* Use a **frequency array or a HashSet** to track characters in the window.
* When you hit a duplicate, **shrink the window** from the left.

## Java Solution using Frequency Array (ASCII 128)

public class Solution {

public int lengthOfLongestSubstring(String s) {

int[] freq = new int[128]; // ASCII character frequency

int left = 0, right = 0;

int maxLen = 0;

while (right < s.length()) {

char c = s.charAt(right);

freq[c]++;

// If duplicate found, move left pointer to remove it

while (freq[c] > 1) {

freq[s.charAt(left)]--;

left++;

}

maxLen = Math.max(maxLen, right - left + 1);

right++;

}

return maxLen;

}

}

## Dry Run Example: s = "abcabcbb"

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | left | right | char | freq[char] | Window | Action | maxLen |
| 1 | 0 | 0 | 'a' | 1 | "a" | Expand window | 1 |
| 2 | 0 | 1 | 'b' | 1 | "ab" | Expand window | 2 |
| 3 | 0 | 2 | 'c' | 1 | "abc" | Expand window | 3 |
| 4 | 0 | 3 | 'a' | 2 | "abca" | Duplicate 'a' → shrink | 3 |
| 5 | 1 | 3 |  |  | "bca" |  | 3 |
| 6 | 1 | 4 | 'b' | 2 | "bcab" | Duplicate 'b' → shrink | 3 |
| 7 | 2 | 4 |  |  | "cab" |  | 3 |
| 8 | 2 | 5 | 'c' | 2 | "cabc" | Duplicate 'c' → shrink | 3 |
| 9 | 3 | 5 |  |  | "abc" |  | 3 |
| 10 | 3 | 6 | 'b' | 2 | "abcb" | Duplicate 'b' → shrink | 3 |
| 11 | 4 | 6 |  |  | "cb" |  | 3 |
| 12 | 4 | 7 | 'b' | 2 | "cbb" | Duplicate 'b' → shrink | 3 |
| 13 | 5 | 7 |  |  | "b" |  | 3 |

✅ Final maxLen = 3 → Substrings: "abc", "bca", "cab"

## Time & Space Complexity

|  |  |
| --- | --- |
| Metric | Value |
| Time | O(n) |
| Space | O(1) (fixed 128-size array for ASCII) |

## Edge Test Cases:

Input: s = "" → Output: 0

Input: s = "bbbbbb" → Output: 1

Input: s = "abcabcbb" → Output: 3

Input: s = "pwwkew" → Output: 3 ("wke")

Input: s = "dvdf" → Output: 3 ("vdf")

## Alternatives

### HashSet-based version (good for Unicode):

public int lengthOfLongestSubstring(String s) {

Set<Character> set = new HashSet<>();

int left = 0, maxLen = 0;

for (int right = 0; right < s.length(); right++) {

while (set.contains(s.charAt(right))) {

set.remove(s.charAt(left));

left++;

}

set.add(s.charAt(right));

maxLen = Math.max(maxLen, right - left + 1);

}

return maxLen;

}