



# Pentonix Internship

## Assignment 1:

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

**Input:**  $s = \text{"abcabcbb"}$ , **Output:** 3

### **Approach:**

Solving the problem of finding the length of the longest unique substring involves iterating through the input string while maintaining a sliding window and a data structure (typically a HashSet) to keep track of unique characters within the current window. Here's a step-by-step approach to solving this problem:

**Initialize variables:**

**maxLength** to store the length of the longest unique substring found so far (initially set to 0).

**left** and **right** pointers to define the current sliding window. Initially, both are set to 0.

Create a HashSet (or other suitable data structure) to store unique characters within the current window.

Enter a loop to traverse the input string from left to right:

Check if the character at the right pointer is already in the HashSet:

If it's not in the HashSet, add it to the HashSet, and update **maxLength** if needed (by comparing it to the current window size,  $\text{right} - \text{left} + 1$ ).

Move the right pointer to expand the window to the right.

Repeat this step until a repeated character is encountered.

When a repeated character is found, it means the current window is no longer valid because it contains a duplicate character.

Handle the case when a repeated character is encountered:

Remove the character at the left pointer from the HashSet.

Move the left pointer to slide the window to the right. Continue this process until the repeated character is removed from the current window.

Repeat steps 2 and 3 until the right pointer reaches the end of the string. This way, you will consider all possible substrings.

Finally, return the `maxLength`, which contains the length of the longest unique substring encountered during the traversal.

This sliding window approach with a HashSet allows you to efficiently find the longest unique substring in linear time complexity,  $O(n)$ , where 'n' is the length of the input string, because each character is processed exactly once.

## The Code:

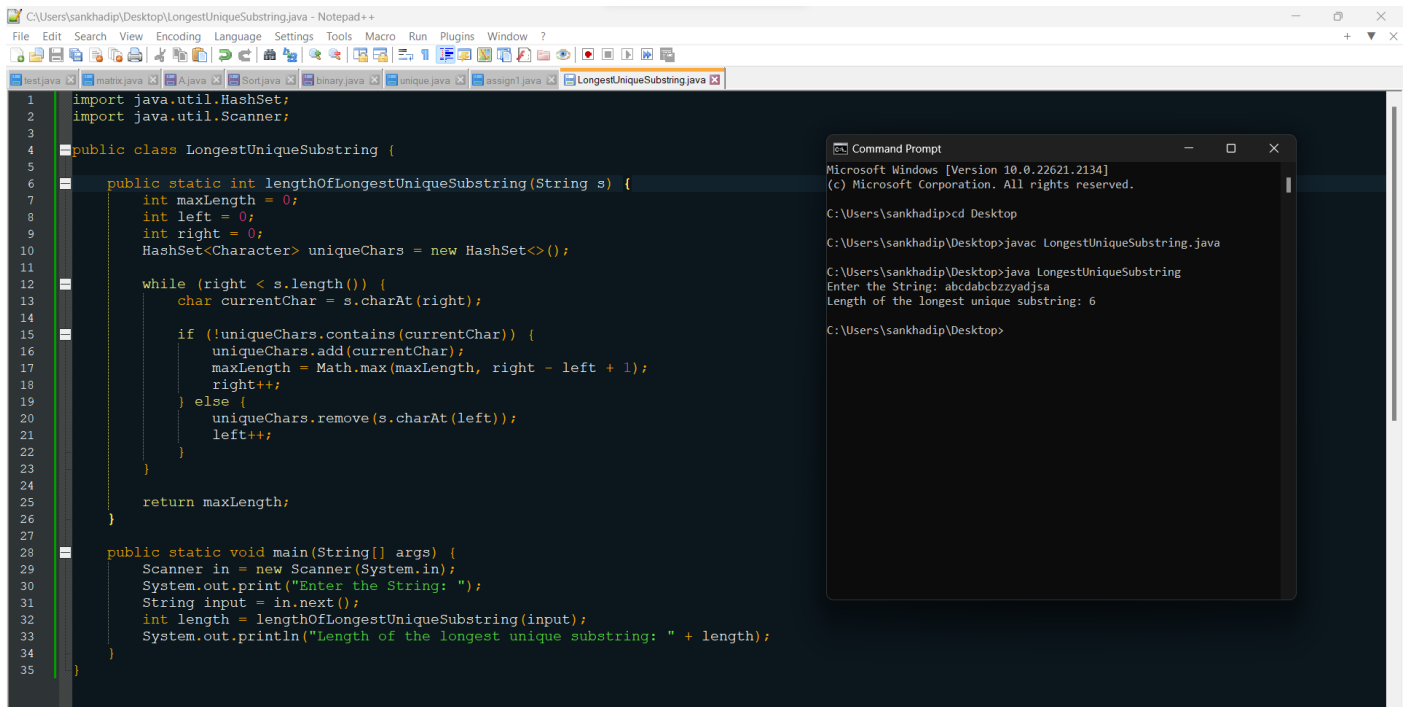
```
import java.util.HashSet;
import java.util.Scanner;

public class LongestUniqueSubstring {

    public static int lengthOfLongestUniqueSubstring(String s) {
        int maxLength = 0;
        int left = 0;
        int right = 0;
        HashSet<Character> uniqueChars = new HashSet<>();
        while (right < s.length()) {
            char currentChar = s.charAt(right);
            if (!uniqueChars.contains(currentChar)) {
                uniqueChars.add(currentChar);
                maxLength = Math.max(maxLength, right - left + 1);
                right++;
            } else {
                uniqueChars.remove(s.charAt(left));
                left++;
            }
        }
        return maxLength;
    }

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the String: ");
        String input = in.next();
        int length = lengthOfLongestUniqueSubstring(input);
        System.out.println("Length of the longest unique substring: " + length);
    }
}
```

# The Code Execution:



The image shows a Notepad++ editor window with the following Java code:

```
1 import java.util.HashSet;
2 import java.util.Scanner;
3
4 public class LongestUniqueSubstring {
5
6     public static int lengthOfLongestUniqueSubstring(String s) {
7         int maxLength = 0;
8         int left = 0;
9         int right = 0;
10        HashSet<Character> uniqueChars = new HashSet<>();
11
12        while (right < s.length()) {
13            char currentChar = s.charAt(right);
14
15            if (!uniqueChars.contains(currentChar)) {
16                uniqueChars.add(currentChar);
17                maxLength = Math.max(maxLength, right - left + 1);
18                right++;
19            } else {
20                uniqueChars.remove(s.charAt(left));
21                left++;
22            }
23        }
24
25        return maxLength;
26    }
27
28    public static void main(String[] args) {
29        Scanner in = new Scanner(System.in);
30        System.out.print("Enter the String: ");
31        String input = in.next();
32        int length = lengthOfLongestUniqueSubstring(input);
33        System.out.println("Length of the longest unique substring: " + length);
34    }
35 }
```

Next to the code editor is a Command Prompt window showing the execution of the program:

```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sankhadip>cd Desktop
C:\Users\sankhadip\Desktop>javac LongestUniqueSubstring.java
C:\Users\sankhadip\Desktop>java LongestUniqueSubstring
Enter the String: abcdabcbzyadjsa
Length of the longest unique substring: 6
C:\Users\sankhadip\Desktop>
```