

## LongestSubstringWithNonRepeatingCharacters in C++

```
#include <iostream>
#include <string>
#include <unordered_map>

class LongestSubstringWithNonRepeatingCharacters {
public:
    static int solution(const std::string& str) {
        int ans = 0;
        int i = -1;
        int j = -1;

        std::unordered_map<char, int> map;
        while (true) {
            bool f1 = false;
            bool f2 = false;

            while (i < static_cast<int>(str.length()) - 1) {
                f1 = true;
                i++;
                char ch = str[i];
                map[ch]++;

                if (map[ch] == 2) {
                    break;
                } else {
                    int len = i - j;
                    if (len > ans) {
                        ans = len;
                    }
                }
            }

            while (j < i) {
                f2 = true;
                j++;
                char ch = str[j];
                map[ch]--;
                if (map[ch] == 1) {
                    break;
                }
            }

            if (!f1 && !f2) {
                break;
            }
        }

        return ans;
    }
};

int main() {
    std::string str = "aabcbcd bca";
    std::cout <<
    LongestSubstringWithNonRepeatingCharacters::solution(str)
    << std::endl;
    return 0;
}
```

## Understanding the Problem

- The function `solution(str)` finds the **length of the longest substring with all distinct (non-repeating) characters**.
- Uses **two-pointer sliding window** (i and j) with an **unordered\_map** to track character frequencies.
- Expands the window until a duplicate character is found, then contracts the window to remove duplicates.

## Example Input

```
string str = "aabcbcd bca";
```

**Expected Output:** 4 (longest substring = "bcd b")

## Step-by-Step Dry Run

Step	i	j	Window (str[j+1] to str[i])	Map	Max Length (ans)
1	0	-1	a	{a:1}	1
2	1	-1	aa	{a:2} (duplicate)	1
3	1	0	a	{a:1}	1
4	2	0	ab	{a:1, b:1}	2
5	3	0	abc	{a:1, b:1, c:1}	3
6	4	0	abcb	{a:1, b:2, c:1}	3
7	4	1	bcb	{b:2, c:1}	3
8	4	2	cb	{b:1, c:1}	3
9	5	2	c b c	{b:1, c:2}	3
10	5	3	b c	{b:1, c:1}	3
11	6	3	b c d	{b:1, c:1, d:1}	3
12	7	3	b c d b	{b:2, c:1, d:1}	4 ✓
13	7	4	c d b	{b:1, c:1, d:1}	4

				c:1, d:1}	
	14	8	4	cdbc	{b:1, c:2, d:1} 4
	15	8	5	dbc	{b:1, c:1, d:1} 4
	16	9	5	dbca	{b:1, c:1, d:1, a:1} 4 ✓
	17	10	6	bca	{b:1, c:1, a:1} 4
<b>Final Output</b>					
✓ <b>Longest substring without repeating characters: 4</b> ("bcd b" or "dbca")					
Output:-4					