LongestSubstringWithNonRepeatingCharacters in C++

```cpp
#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 = "aabcbcdbca";
    std::cout << LongestSubstringWithNonRepeatingCharacters::solution(str) << std::endl;
    return 0;
}
```

Step-by-Step Dry Run:

Initial state:

- str = "aabcbcdbca"
- ans = 0
- i = -1, j = -1
- map = {}

First pass:

1. **Expand window (while (i < str.length() - 1)):**
   - i = 0, character is a, map = {a: 1}
   - i = 1, character is a, map = {a: 2}
   - Since map[a] == 2, break the loop.

2. **Shrink window (while (j < i)):**
   - j = 0, character is a, map = {a: 1}
   - Now, map[a] == 1, break the loop.

At this point:

- ans = 1 because we found the substring "a" (length 1).

Second pass:

1. **Expand window (while (i < str.length() - 1)):**
   - i = 2, character is b, map = {a: 1, b: 1}
   - i = 3, character is c, map = {a: 1, b: 1, c: 1}
   - i = 4, character is b, map = {a: 1, b: 2, c: 1}
   - Since map[b] == 2, break the loop.

2. **Shrink window (while (j < i)):**
   - j = 1, character is a, map = {a: 0, b: 2, c: 1}
   - j = 2, character is b, map = {b: 1, c: 1}
   - map.size() = 2 so continue shrinking.
   - j = 3, character is c, map = {b: 1, c: 0}
   - Now map.size() = 1 and j = 3, break the loop.

At this point:

- ans = 3 because the substring "abc" (length 3) was found.

Third pass:

1. **Expand window (while (i < str.length() - 1)):**
   - i = 4, character is d, map = {b: 1, c: 1, d: 1}
   - i = 5, character is c, map = {b: 1, c: 2, d: 1}
   - Since map[c] == 2, break the loop.

2. **Shrink window (while (j < i)):**
   - j = 4, character is b, map = {b: 0, c: 2, d: 1}
   - j = 5, character is c, map = {c: 1, d: 1}
   - map.size() = 2 so continue shrinking.
   - j = 6, character is d, map = {d: 0, c: 1}
   - Now map.size() = 1 and j = 6, break the loop.

At this point:

- ans = 3 because the substring "bcd" (length 3) was found.

Fourth pass:

1. **Expand window (while (i < str.length() - 1)):**
   - i = 7, character is b, map = {d: 0, c: 1, b: 1}
   - i = 8, character is c, map = {d: 0, c: 2, b: 1}
   - Since map[c] == 2, break the loop.

2. **Shrink window (while (j < i)):**
   - j = 6, character is d, map = {d: 0, c: 1, b: 1}
   - Now map.size() = 3 and we have found the largest substring "bcd" (length 3).

At this point:

- The function finishes and ans = 4.

Output:-4