JS:

function non\_repeat\_substring(str) {

let windowStart = 0,

maxLength = 0,

charIndexMap = {};

// try to extend the range [windowStart, windowEnd]

for (let windowEnd = 0; windowEnd < str.length; windowEnd++) {

const rightChar = str[windowEnd];

// if the map already contains the 'rightChar', shrink the window from the beginning so that

// we have only one occurrence of 'rightChar'

if (rightChar in charIndexMap) {

// this is tricky; in the current window, we will not have any 'rightChar' after its previous index

// and if 'windowStart' is already ahead of the last index of 'rightChar', we'll keep 'windowStart'

windowStart = Math.max(windowStart, charIndexMap[rightChar] + 1);

}

// insert the 'rightChar' into the map

charIndexMap[rightChar] = windowEnd;

// remember the maximum length so far

maxLength = Math.max(maxLength, windowEnd - windowStart + 1);

}

return maxLength;

}

console.log(`Length of the longest substring: ${non\_repeat\_substring('aabccbb')}`);

console.log(`Length of the longest substring: ${non\_repeat\_substring('abbbb')}`);

console.log(`Length of the longest substring: ${non\_repeat\_substring('abccde')}`);

C++

using namespace std;

#include <iostream>

#include <string>

#include <unordered\_map>

class NoRepeatSubstring {

public:

static int findLength(const string& str) {

int windowStart = 0, maxLength = 0;

unordered\_map<char, int> charIndexMap;

// try to extend the range [windowStart, windowEnd]

for (int windowEnd = 0; windowEnd < str.length(); windowEnd++) {

char rightChar = str[windowEnd];

// if the map already contains the 'rightChar', shrink the window from the beginning so that

// we have only one occurrence of 'rightChar'

if (charIndexMap.find(rightChar) != charIndexMap.end()) {

// this is tricky; in the current window, we will not have any 'rightChar' after its

// previous index and if 'windowStart' is already ahead of the last index of 'rightChar',

// we'll keep 'windowStart'

windowStart = max(windowStart, charIndexMap[rightChar] + 1);

}

charIndexMap[rightChar] = windowEnd; // insert the 'rightChar' into the map

maxLength =

max(maxLength, windowEnd - windowStart + 1); // remember the maximum length so far

}

return maxLength;

}

};

int main(int argc, char\* argv[]) {

cout << "Length of the longest substring: " << NoRepeatSubstring::findLength("aabccbb") << endl;

cout << "Length of the longest substring: " << NoRepeatSubstring::findLength("abbbb") << endl;

cout << "Length of the longest substring: " << NoRepeatSubstring::findLength("abccde") << endl;

}

Python:

def non\_repeat\_substring(str1):

window\_start = 0

max\_length = 0

char\_index\_map = {}

# try to extend the range [windowStart, windowEnd]

for window\_end in range(len(str1)):

right\_char = str1[window\_end]

# if the map already contains the 'right\_char', shrink the window from the beginning so that

# we have only one occurrence of 'right\_char'

if right\_char in char\_index\_map:

# this is tricky; in the current window, we will not have any 'right\_char' after its previous index

# and if 'window\_start' is already ahead of the last index of 'right\_char', we'll keep 'window\_start'

window\_start = max(window\_start, char\_index\_map[right\_char] + 1)

# insert the 'right\_char' into the map

char\_index\_map[right\_char] = window\_end

# remember the maximum length so far

max\_length = max(max\_length, window\_end - window\_start + 1)

return max\_length

def main():

print("Length of the longest substring: " + str(non\_repeat\_substring("aabccbb")))

print("Length of the longest substring: " + str(non\_repeat\_substring("abbbb")))

print("Length of the longest substring: " + str(non\_repeat\_substring("abccde")))

main()

Java:

import java.util.\*;

class NoRepeatSubstring {

public static int findLength(String str) {

int windowStart = 0, maxLength = 0;

Map<Character, Integer> charIndexMap = new HashMap<>();

// try to extend the range [windowStart, windowEnd]

for (int windowEnd = 0; windowEnd < str.length(); windowEnd++) {

char rightChar = str.charAt(windowEnd);

// if the map already contains the 'rightChar', shrink the window from the beginning so that

// we have only one occurrence of 'rightChar'

if (charIndexMap.containsKey(rightChar)) {

// this is tricky; in the current window, we will not have any 'rightChar' after its previous index

// and if 'windowStart' is already ahead of the last index of 'rightChar', we'll keep 'windowStart'

windowStart = Math.max(windowStart, charIndexMap.get(rightChar) + 1);

}

charIndexMap.put(rightChar, windowEnd); // insert the 'rightChar' into the map

maxLength = Math.max(maxLength, windowEnd - windowStart + 1); // remember the maximum length so far

}

return maxLength;

}

public static void main(String[] args) {

System.out.println("Length of the longest substring: " + NoRepeatSubstring.findLength("aabccbb"));

System.out.println("Length of the longest substring: " + NoRepeatSubstring.findLength("abbbb"));

System.out.println("Length of the longest substring: " + NoRepeatSubstring.findLength("abccde"));

}

}