JS:

function longest\_substring\_with\_k\_distinct(str, k) {

let windowStart = 0,

maxLength = 0,

charFrequency = {};

// in the following loop we'll try to extend the range [window\_start, window\_end]

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

const rightChar = str[windowEnd];

if (!(rightChar in charFrequency)) {

charFrequency[rightChar] = 0;

}

charFrequency[rightChar] += 1;

// shrink the sliding window, until we are left with 'k' distinct characters in the char\_frequency

while (Object.keys(charFrequency).length > k) {

const leftChar = str[windowStart];

charFrequency[leftChar] -= 1;

if (charFrequency[leftChar] === 0) {

delete charFrequency[leftChar];

}

windowStart += 1; // shrink the window

}

// remember the maximum length so far

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

}

return maxLength;

}

C++:

using namespace std;

#include <iostream>

#include <string>

#include <unordered\_map>

class LongestSubstringKDistinct {

public:

static int findLength(const string &str, int k) {

int windowStart = 0, maxLength = 0;

unordered\_map<char, int> charFrequencyMap;

// in the following loop we'll try to extend the range [windowStart, windowEnd]

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

char rightChar = str[windowEnd];

charFrequencyMap[rightChar]++;

// shrink the sliding window, until we are left with 'k' distinct characters in the frequency

// map

while ((int)charFrequencyMap.size() > k) {

char leftChar = str[windowStart];

charFrequencyMap[leftChar]--;

if (charFrequencyMap[leftChar] == 0) {

charFrequencyMap.erase(leftChar);

}

windowStart++; // shrink the window

}

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: " << LongestSubstringKDistinct::findLength("araaci", 2)

<< endl;

cout << "Length of the longest substring: " << LongestSubstringKDistinct::findLength("araaci", 1)

<< endl;

cout << "Length of the longest substring: " << LongestSubstringKDistinct::findLength("cbbebi", 3)

<< endl;

}

Python:

def longest\_substring\_with\_k\_distinct(str1, k):

window\_start = 0

max\_length = 0

char\_frequency = {}

# in the following loop we'll try to extend the range [window\_start, window\_end]

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

right\_char = str1[window\_end]

if right\_char not in char\_frequency:

char\_frequency[right\_char] = 0

char\_frequency[right\_char] += 1

# shrink the sliding window, until we are left with 'k' distinct characters in the char\_frequency

while len(char\_frequency) > k:

left\_char = str1[window\_start]

char\_frequency[left\_char] -= 1

if char\_frequency[left\_char] == 0:

del char\_frequency[left\_char]

window\_start += 1 # shrink the window

# 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 substr1ing: " + str(longest\_substring\_with\_k\_distinct("araaci", 2)))

print("Length of the longest substr1ing: " + str(longest\_substring\_with\_k\_distinct("araaci", 1)))

print("Length of the longest substr1ing: " + str(longest\_substring\_with\_k\_distinct("cbbebi", 3)))

main()

JAVA:

import java.util.\*;

class LongestSubstringKDistinct {

public static int findLength(String str, int k) {

if (str == null || str.length() == 0 || str.length() < k)

throw new IllegalArgumentException();

int windowStart = 0, maxLength = 0;

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

// in the following loop we'll try to extend the range [windowStart, windowEnd]

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

char rightChar = str.charAt(windowEnd);

charFrequencyMap.put(rightChar, charFrequencyMap.getOrDefault(rightChar, 0) + 1);

// shrink the sliding window, until we are left with 'k' distinct characters in the frequency map

while (charFrequencyMap.size() > k) {

char leftChar = str.charAt(windowStart);

charFrequencyMap.put(leftChar, charFrequencyMap.get(leftChar) - 1);

if (charFrequencyMap.get(leftChar) == 0) {

charFrequencyMap.remove(leftChar);

}

windowStart++; // shrink the window

}

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: " + LongestSubstringKDistinct.findLength("araaci", 2));

System.out.println("Length of the longest substring: " + LongestSubstringKDistinct.findLength("araaci", 1));

System.out.println("Length of the longest substring: " + LongestSubstringKDistinct.findLength("cbbebi", 3));

}

}