educative

Longest Substring with K Distinct Characters (medium)

We'll cover the following

Problem Statement

Given a string, find the length of the **longest substring** in it **with no more than K distinct characters**.

Example 1:

```
Input: String="araaci", K=2
Output: 4
Explanation: The longest substring with no more than '2' distinct characters is "araa".
```

Example 2:

```
Input: String="araaci", K=1
Output: 2
Explanation: The longest substring with no more than '1' distinct characters is "aa".
```

Example 3:

```
Input: String="cbbebi", K=3
Output: 5
Explanation: The longest substrings with no more than '3' distinct characters are "cbbeb" & "bbebi"
.
```

Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```
class LongestSubstringKDistinct {
  public static int findLength(String str, int k) {
    // TODO: Write your code here
    int windowstart=0;
    int n=str.length();
    char key=' ';
    int max=0;
    HashMap<Character,Integer> h=new HashMap<>();
    for(int windowend=0;windowend<n;windowend++)
    {
         key=str.charAt(windowend);
          if(h.containsKey(key))
          {
               h.put(key,h.get(key)+1);
          }
          else
            h.put(key,1);

          if(h.size()>k)
          {
            if(h.containsKey(str.charAt(windowstart)))
            {
                  h.put(str.charAt(windowstart),h.get(str
            }
            if(h.get(str.charAt(windowstart))==0)
              h.remove(str.charAt(windowstart));
            windowstart++;
          }
          max=Math.max(max,windowend-windowstart+1);
    }
    return max;
```

Show Results Show Console

3 of 3 Tests Passed

Result	Input	Expected Output	Actual Output	Reason
✓	findLength(araaci, 2)	4	4	Succeeded
✓	findLength(araaci, 1)	2	2	Succeeded
✓	findLength(cbbebi, 3)	5	5	Succeeded

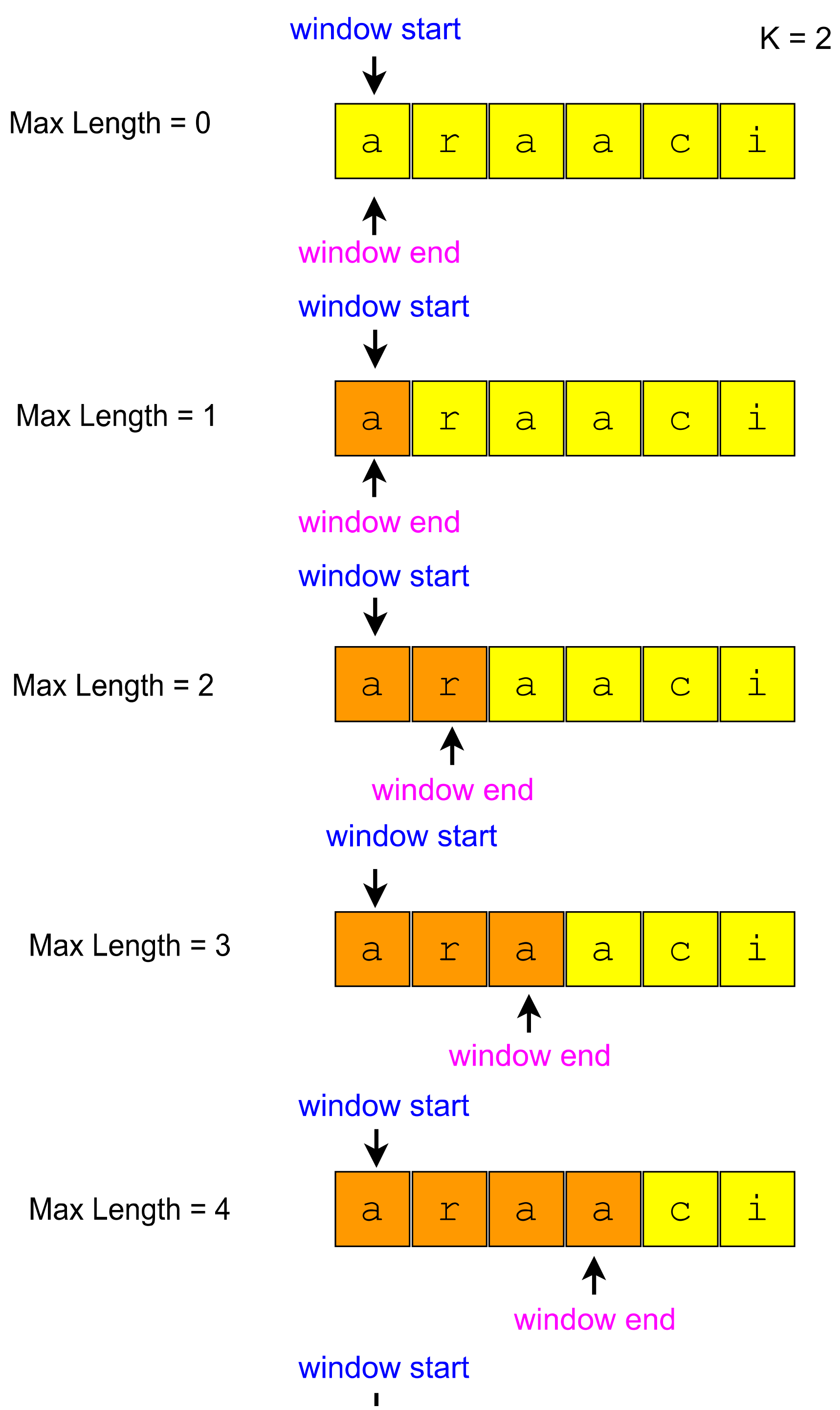
2.870s

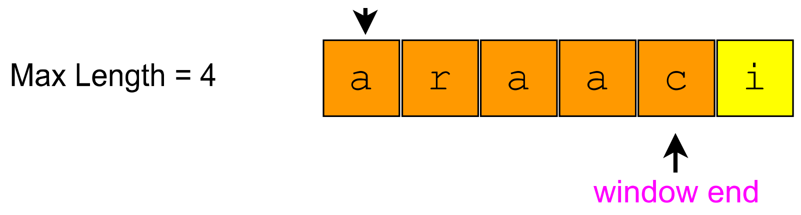
Solution

This problem follows the **Sliding Window** pattern, and we can use a similar dynamic sliding window strategy as discussed in Smallest Subarray with a given sum (https://www.educative.io/collection/page/5668639101419520/5671464854355968/5177043027230720/). We can use a **HashMap** to remember the frequency of each character we have processed. Here is how we will solve this problem:

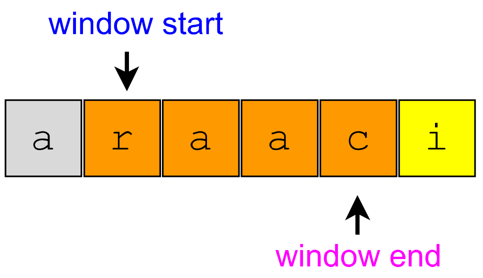
1. First, we will insert characters from the beginning of the string until we have 'K' distinct characters in the **HashMap**.
2. These characters will constitute our sliding window. We are asked to find the longest such window having no more than 'K' distinct characters. We will remember the length of this window as the longest window so far.
3. After this, we will keep adding one character in the sliding window (i.e., slide the window ahead) in a stepwise fashion.
4. In each step, we will try to shrink the window from the beginning if the count of distinct characters in the **HashMap** is larger than 'K.' We will shrink the window until we have no more than 'K' distinct characters in the **HashMap**. This is needed as we intend to find the longest window.
5. While shrinking, we'll decrement the character's frequency going out of the window and remove it from the **HashMap** if its frequency becomes zero.
6. At the end of each step, we'll check if the current window length is the longest so far, and if so, remember its length.

Here is the visual representation of this algorithm for the Example-1:

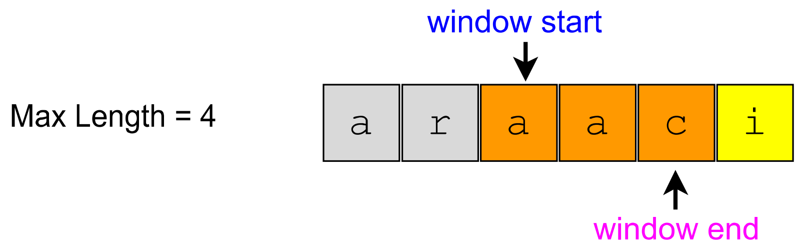

window start
K = 2
Max Length = 0
a r a a c i
window end
window start
Max Length = 1
a r a a c i
window end
window start
Max Length = 2
a r a a c i
window end
window start
Max Length = 3
a r a a c i
window end
window start
Max Length = 4
a r a a c i
window end
window start

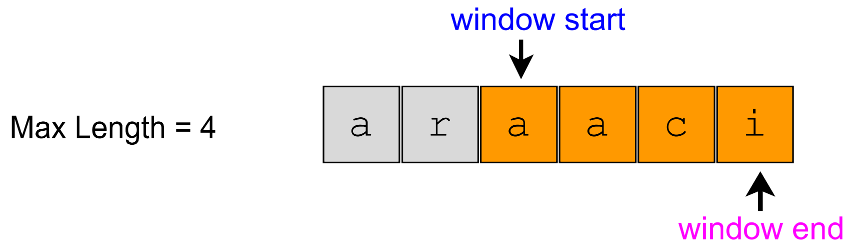


Number of distinct characters > 2, let's shrink the sliding window

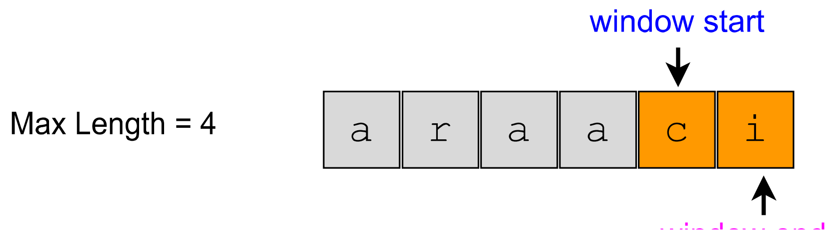


Number of distinct characters are still > 2, let's shrink the sliding window





Number of distinct character > 2, let's shrink the sliding window



window end

Code

Here is how our algorithm will look:

Java | Python3 | C++ | JS

```
class LongestSubstringKDistinct {
  public static int findLength(String str, int k) {
    if (str == null || str.length() == 0 || str.length()
      throw new IllegalArgumentException();

    int windowStart = 0, maxLength = 0;
    Map<Character, Integer> charFrequencyMap = new HashMa
    // in the following loop we'll try to extend the rang
    for (int windowEnd = 0; windowEnd < str.length(); wir
      char rightChar = str.charAt(windowEnd);
      charFrequencyMap.put(rightChar, charFrequencyMap.ge
      // shrink the sliding window, until we are left wit
      while (charFrequencyMap.size() > k) {
        char leftChar = str.charAt(windowStart);
        charFrequencyMap.put(leftChar, charFrequencyMap.g
        if (charFrequencyMap.get(leftChar) == 0) {
          charFrequencyMap.remove(leftChar);
        }
        windowStart++; // shrink the window
      }
      maxLength = Math.max(maxLength, windowEnd - windowS
    }

    return maxLength;
  }

  public static void main(String[] args) {
    System.out.println("Length of the longest substring:
    System.out.println("Length of the longest substring:
```

```
    System.out.println("Length of the longest substring:
  }
```

Output 1.502s

```
Length of the longest substring: 4
Length of the longest substring: 2
Length of the longest substring: 5
```

Time Complexity

The above algorithm's time complexity will be $O(N)$, where 'N' is the number of characters in the input string. The outer `for` loop runs for all characters, and the inner `while` loop processes each character only once; therefore, the time complexity of the algorithm will be $O(N + N)$, which is asymptotically equivalent to $O(N)$.

Space Complexity

The algorithm's space complexity is $O(K)$, as we will be storing a maximum of 'K+1' characters in the HashMap.

← Back
Smallest Subarray with a given sum (e...

Next →
Fruits into Baskets (medium)

✓ Completed

3% completed, meet the criteria and claim your course certificate!

Claim Certificate

Report an Issue

Ask a Question (https://discuss.educative.io/tag/longest-substring-with-k-distinct-characters-medium__pattern-sliding-window__grokking-the-coding-interview-patterns-for-coding-questions)