Code

Here is how our algorithm will look:

Java | Python3 | C++ | JS

```cpp
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;
}
```

Output 0.730s

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

Time Complexity

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