AlgoExpert   Quad Layout   C#   12px   Sublime   Monokai   00:00:00

Prompt   Scratchpad   Our Solution(s)   Video Explanation   Run Code

Solution 1

```csharp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

using System;
using System.Collections.Generic;

public class Program {
  // O(b + s) time | O(b + s) space - where b is the length of the big
  // input string and s is the length of the small input string
  public static string SmallestSubstringContaining(string bigstring, string smallstring) {
    Dictionary<char, int> targetCharCounts = getCharCounts(smallstring);
    List<int> substringBounds = getSubstringBounds(bigstring, targetCharCounts);
    return getstringFromBounds(bigstring, substringBounds);
  }

  public static Dictionary<char, int> getCharCounts(string str) {
    Dictionary<char, int> charCounts = new Dictionary<char, int>();
    for (int i = 0; i < str.Length; i++ ) {
      increaseCharCount(str[i], charCounts);
    }
    return charCounts;
  }

  public static List<int> getSubstringBounds(string str, Dictionary<char,
    int> targetCharCounts) {
    List<int> substringBounds = new List<int>(){
      0, Int32.MaxValue
    };
    Dictionary<char, int> substringCharCounts = new Dictionary<char, int>();
    int numUniqueChars = targetCharCounts.Count;
    int numUniqueCharsDone = 0;
    int leftIdx = 0;
    int rightIdx = 0;
    // Move the rightIdx to the right in the string until you've counted
    // all of the target characters enough times.
    while (rightIdx < str.Length) {
      char rightChar = str[rightIdx];
      if (!targetCharCounts.ContainsKey(rightChar)) {
        rightIdx++;
        continue;
      }
      increaseCharCount(rightChar, substringCharCounts);
      if (substringCharCounts[rightChar] == targetCharCounts[rightChar]) {
        numUniqueCharsDone++;
      }
      // Move the leftIdx to the right in the string until you no longer
      // have enough of the target characters in between the leftIdx and
      // the rightIdx. Update the substringBounds accordingly.
      while (numUniqueCharsDone == numUniqueChars && leftIdx <= rightIdx) {
        substringBounds = getCloserBounds(leftIdx, rightIdx,
            substringBounds[0],
            substringBounds[1]);
        char leftChar = str[leftIdx];
        if (!targetCharCounts.ContainsKey(leftChar)) {
          leftIdx++;
          continue;
        }
        if (substringCharCounts[leftChar] == targetCharCounts[leftChar]) {
          numUniqueCharsDone--;
        }
        decreaseCharCount(leftChar, substringCharCounts);
        leftIdx++;
      }
      rightIdx++;
    }
    return substringBounds;
  }

  public static List<int> getCloserBounds(int idx1, int idx2, int idx3, int idx4) {
    return idx2 - idx1 < idx4 - idx3 ? new List<int>(){
            idx1, idx2
    }
          : new List<int>(){
            idx3, idx4
          };
  }

  public static string getstringFromBounds(string str, List<int> bounds) {
    int start = bounds[0];
    int end = bounds[1];
    if (end == Int32.MaxValue)
      return "";
    return str.Substring(start, end + 1 - start);
  }

  public static void increaseCharCount(char c, Dictionary<char, int> charCounts) {
    if (!charCounts.ContainsKey(c)) {
      charCounts[c] = 1;
    } else {
      charCounts[c] = charCounts[c] + 1;
    }
  }

  public static void decreaseCharCount(char c, Dictionary<char, int> charCounts) {
    charCounts[c]  = charCounts[c] - 1;
  }
}

```