Solution 1

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

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
    func smallestSubstringContaining(_ bigString: String, _ smallString: String) -> String {
        let targetCharCounts = getCharCounts(smallString)
        let substringBounds = getSubstringBounds(bigString, targetCharCounts)
        return getStringFromBounds(bigString, substringBounds)
    }

    func getCharCounts(_ str: String) -> [Character: Int] {
        var charCounts = [Character: Int]()
        for char in str {
            changeCharCount(char, &charCounts, 1)
        }
        return charCounts
    }

    func changeCharCount(_ char: Character, _ charCounts: inout [Character: Int], _ change: Int) {
        if let count = charCounts[char] {
            charCounts.updateValue(count + change, forKey: char)
            return
        }
        charCounts[char] = change
    }

    func getSubstringBounds(_ str: String, _ targetCharCounts: [Character: Int]) -> [Int] {
        var substringBounds = [0, Int.max]
        var substringCharCounts = [Character: Int]()
        var numUniqueChars = targetCharCounts.count
        var numUniqueCharsDone = 0
        var leftIdx = 0
        var 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 {
            let rightStringIndex = str.index(str.startIndex, offsetBy: rightIdx)

            let rightChar = str[rightStringIndex]
            if targetCharCounts[rightChar] == nil {
                rightIdx += 1
                continue
            }
            changeCharCount(rightChar, &substringCharCounts, 1)
            if substringCharCounts[rightChar] == targetCharCounts[rightChar] {
                numUniqueCharsDone += 1
            }

            // 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 {
                let leftStringIndex = str.index(str.startIndex, offsetBy: leftIdx)
                substringBounds = getCloserBounds(leftIdx, rightIdx,
                                                  substringBounds[0], substringBounds[1])
                let leftChar = str[leftStringIndex]
                if substringCharCounts[leftChar] == nil {
                    leftIdx += 1
                    continue
                }
                if substringCharCounts[leftChar] == targetCharCounts[leftChar] {
                    numUniqueCharsDone -= 1
                }
                changeCharCount(leftChar, &substringCharCounts, -1)
                leftIdx += 1
            }
            rightIdx += 1
        }

        return substringBounds
    }

    func getCloserBounds(_ idx1: Int, _ idx2: Int, _ idx3: Int, _ idx4: Int) -> [Int] {
        if idx2 - idx1 < idx4 - idx3 {
            return [idx1, idx2]
        }
        return [idx3, idx4]
    }

    func getStringFromBounds(_ str: String, _ bounds: [Int]) -> String {
        let start = bounds[0]
        let end = bounds[1]
        if end == Int.max {
            return ""
        }

        let startIdx = str.index(str.startIndex, offsetBy: start)
        let endIdx = str.index(str.startIndex, offsetBy: end + 1)
        let newStr = str[startIdx ..< endIdx]
        return String(newStr)
    }
}

```