Solution 1

```python
# Copyright © 2020 AlgoExpert, LLC. All rights reserved.

# 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
def smallestSubstringContaining(bigString, smallString):
    targetCharCounts = getCharCounts(smallString)
    substringBounds = getSubstringBounds(bigString, targetCharCounts)
    return getStringFromBounds(bigString, substringBounds)


def getCharCounts(string):
    charCounts = {}
    for char in string:
        increaseCharCount(char, charCounts)
    return charCounts


def getSubstringBounds(string, targetCharCounts):
    substringBounds = [0, float("inf")]
    substringCharCounts = {}
    numUniqueChars = len(targetCharCounts.keys())
    numUniqueCharsDone = 0
    leftIdx = 0
    rightIdx = 0
    # Move the rightIdx to the right in the string until you've counted
    # all of the target characters enough times.
    while rightIdx < len(string):
        rightChar = string[rightIdx]
        if rightChar not in targetCharCounts:
            rightIdx += 1
            continue
        increaseCharCount(rightChar, substringCharCounts)
        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 and leftIdx <= rightIdx:
            substringBounds = getCloserBounds(leftIdx, rightIdx, substringBounds[0], substringBounds[1])
            leftChar = string[leftIdx]
            if leftChar not in targetCharCounts:
                leftIdx += 1
                continue
            if substringCharCounts[leftChar] == targetCharCounts[leftChar]:
                numUniqueCharsDone -= 1
            decreaseCharCount(leftChar, substringCharCounts)
            leftIdx += 1
        rightIdx += 1
    return substringBounds


def getCloserBounds(idx1, idx2, idx3, idx4):
    return [idx1, idx2] if idx2 - idx1 < idx4 - idx3 else [idx3, idx4]


def getStringFromBounds(string, bounds):
    start, end = bounds
    if end == float("inf"):
        return ""
    return string[start : end + 1]


def increaseCharCount(char, charCounts):
    if char not in charCounts:
        charCounts[char] = 0
    charCounts[char] += 1


def decreaseCharCount(char, charCounts):
    charCounts[char] -= 1

```