Our Solution(s) Run Code

Solution 1

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

package main

import "math"

// 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, smallString string) string
  targetCharCounts := getCharCounts(smallString)
  substringBounds := getSubstringBounds(bigString, targetCharCounts)
  return getStringFromBounds(bigString, substringBounds)
}

func getCharCounts(str string) map[byte]int {
  charCounts := map[byte]int{}
  for _, char := range str {
    increaseCharCount(byte(char), charCounts)
  }
  return charCounts
}

func getSubstringBounds(str string, targetCharCounts map[byte]int) []i
  substringBounds := []int{0, math.MaxInt32}
  substringCharCounts := map[byte]int{}
  numUniqueChars := len(targetCharCounts)
  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.
  for rightIdx < len(str) {
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

func SmallestSubstringContaining(bigString, smallString string) string
  // Write your code here.
  return ""
}

```

Our Tests

Custom Output Submit Code

Run or submit code when you're ready.