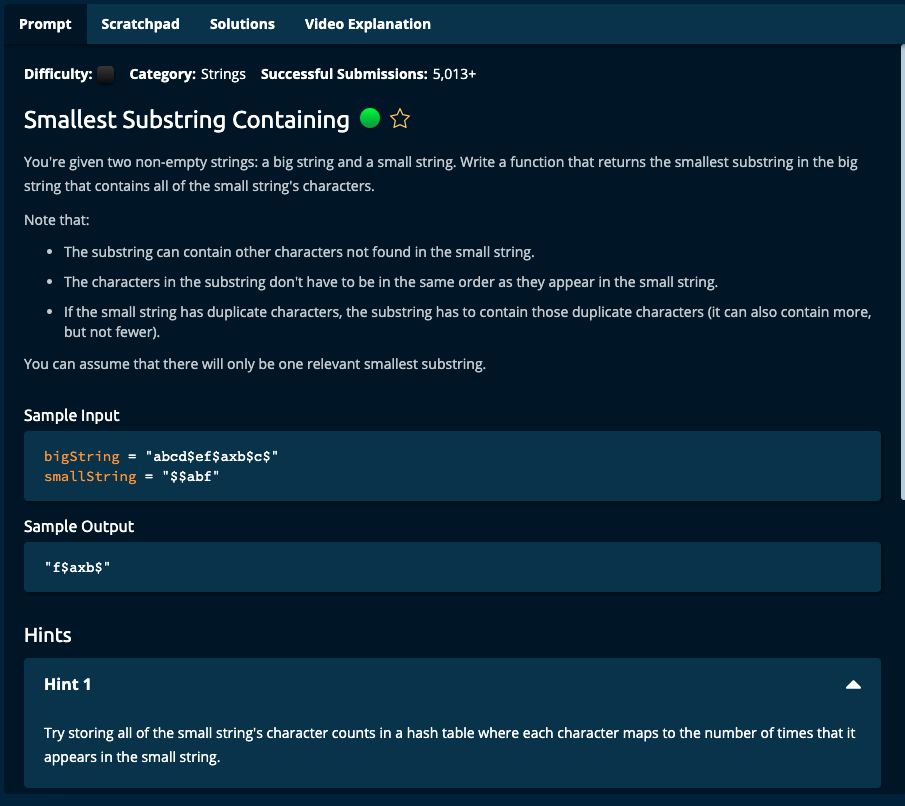
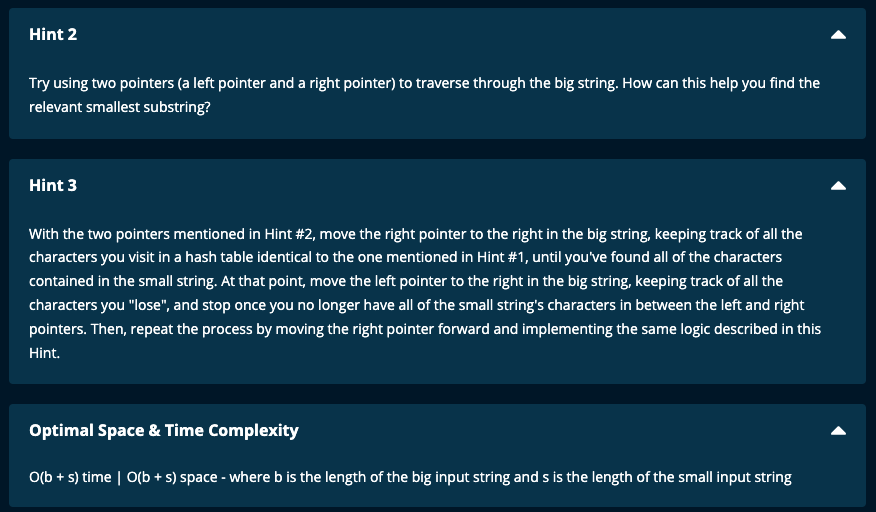
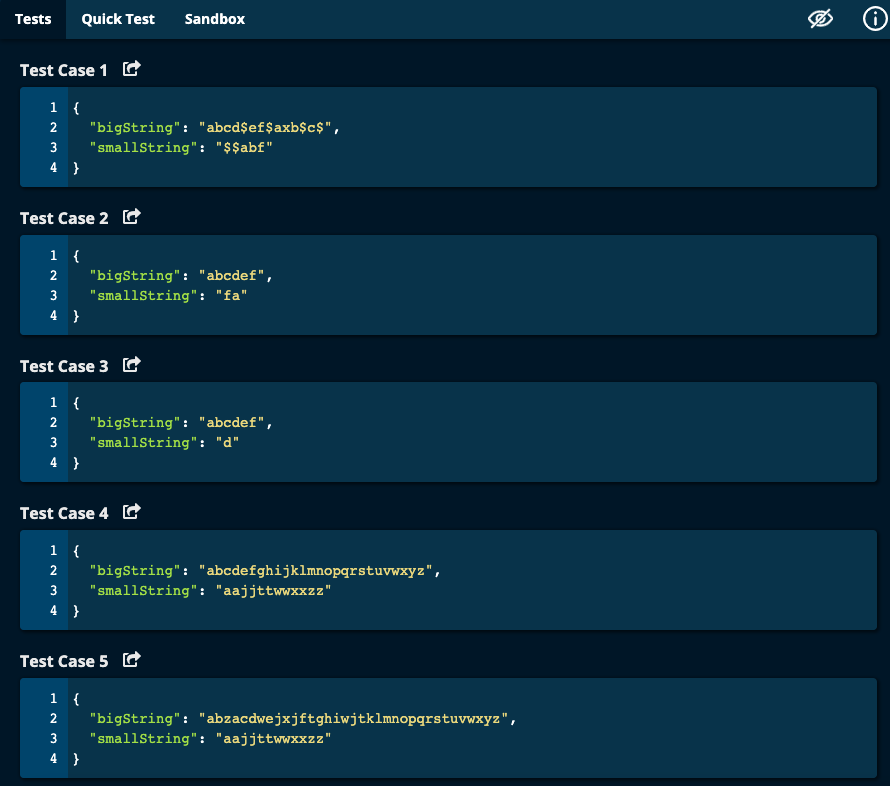
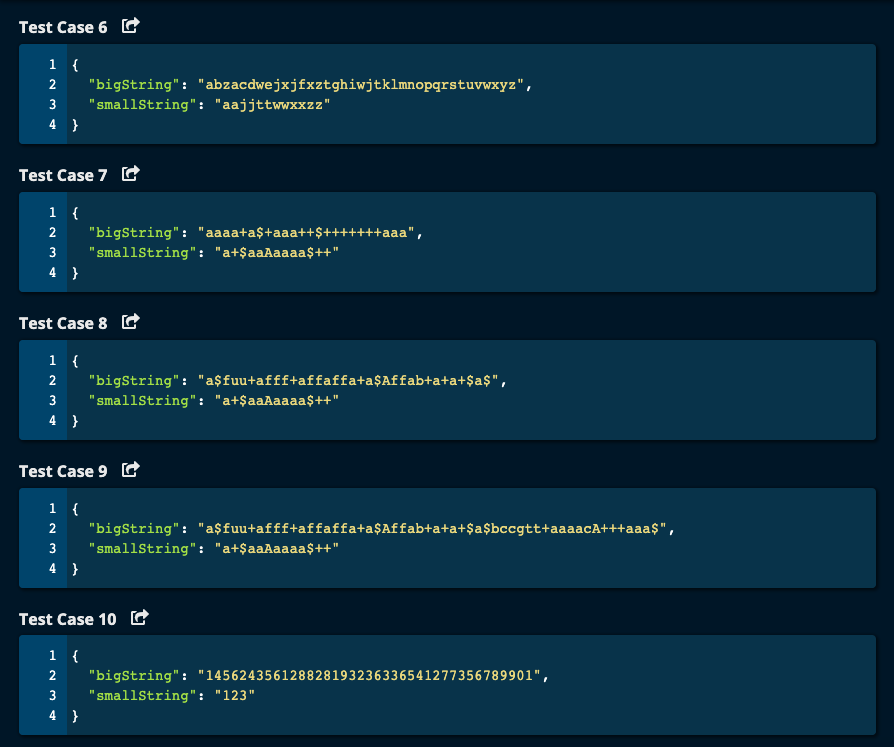
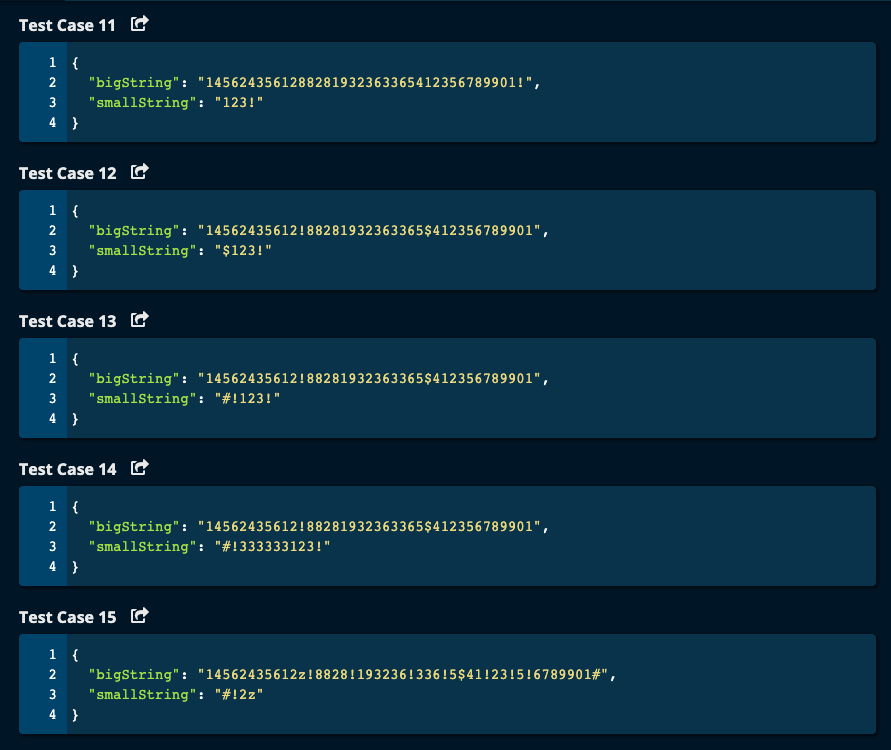
Smallest Substring Containing (Very Hard)

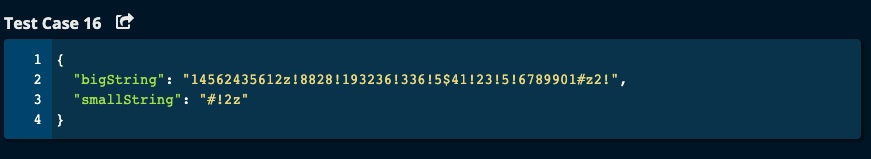












My Solution:

JJ Notes:

1. Check whether bigString has all the characters present in smallString with the required frequency using function isValidBigString(). This returns True is bigString is good and False otherwise. If False, return an empty string.
2. Traverse through bigString to find the smallest substring. Compare the lengths of the currentSubstring with the smallestSubstring and if currentSubstringLen is smaller update the smallestSubstring.
3. Return the smallest substring.
4. This is a sliding window approach where the left pointer moves by 1 character on the bigString for every iteration of the while loop and the right pointer also moves to the right but never goes back to the left.

Time Complexity: O(b + s)

Space Complexity: O(b + s)

Where b is the length of bigString and s is the length of smallString

--------------------------

import math

def smallestSubstringContaining(bigString, smallString):

'''

Given two strings bigString and smallString, find the smallest substring in bigString that contains all of

the smallString's characters.

:param bigString: given

:param smallString: given

:return: smallest substring of bigString containing all the characters in smallString

'''

# check if bigString has all the characters in smallString with required frequency

isBigStringGood = isValidBigString(bigString, smallString)

if not isBigStringGood:

return ""

# Traverse through bigString to find substring

smallestSubstring = ""

smallestSubstringLen = math.inf # initialize to a biggest value

currentSubstring = ""

currentSubstringLen = 0

left = 0

right = len(smallString) - 1 # substring has to be at least as big as smallString

isFound = False

uniqueCharsFound = 0

smallStringDict = createDict(smallString)

while left < len(bigString) and right < len(bigString) and left <= right:

currentSubstring = bigString[left: right + 1]

isGoodSubstring = checkValidSubstring(currentSubstring, smallStringDict)

while not isGoodSubstring and right < len(bigString):

right += 1

currentSubstring = bigString[left: right + 1]

isGoodSubstring = checkValidSubstring(currentSubstring, smallStringDict)

print("In while loop, left = ", left, "currentSubstring = ", currentSubstring, "isGoodSubstring = ",

isGoodSubstring)

if isGoodSubstring:

currentSubstringLen = len(currentSubstring)

if currentSubstringLen < smallestSubstringLen:

smallestSubstringLen = currentSubstringLen

smallestSubstring = currentSubstring

left += 1 # for sliding window

print("At end, left = ", left, "currentSubstring = ", currentSubstring, "isGoodSubstring = ", isGoodSubstring,

"smallestSubstring = ", smallestSubstring)

return smallestSubstring

def createDict(astring):

'''

create a dictionary with key as characters from the given string and value as its frequency

:return: a dictionary of frequency

'''

adict = {}

print("in createDict() astring ")

for ch in astring:

if ch not in adict:

adict[ch] = 1

else:

adict[ch] += 1

return adict

def isValidBigString(bigString, smallString):

'''

Check if bigString contains all the characters in smallString and with the required frequency

:return: a boolean -- True if bigString is good and False if it is not

'''

smallStringSet = set(smallString)

bigStringSet = set(bigString)

if smallStringSet.issubset(bigStringSet):

print("bigString has all the characters in smallString")

else:

print("bigString does not has all the characters in smallString")

print("Terminating program")

return False

# Check if bigString has the required frequency as that of the smallString

smallStringDict = createDict(smallString)

bigStringDict = createDict(bigString)

for ch in smallStringDict.keys():

if ch not in bigStringDict or bigStringDict[ch] < smallStringDict[ch]:

print("For ch = ", ch, "bigString frequency = ", bigStringDict[ch], "smallString frequency = ",

smallStringDict[ch])

print("Terminating program")

return False

return True

def checkValidSubstring(substring, smallStringDict):

'''

Given a substring from bigString check if it contains all the characters in smallString with the required frequency

:param substring: substring from bigString

:param smallStringDict: dictionary with characters from smallString with their respective frequencies.

:return: return a boolean whether it is a valid substring

'''

substringDict = {}

for ch in substring:

if ch in smallStringDict.keys():

if ch not in substringDict:

substringDict[ch] = 1

else:

substringDict[ch] += 1

print("In checkValidSubstring, smallStringDict = ", smallStringDict, "substring = ", substring, "substringDict = ", substringDict)

# Check if all characters and their respective frequencies in substring satisfies smallStringDict

isValid = True

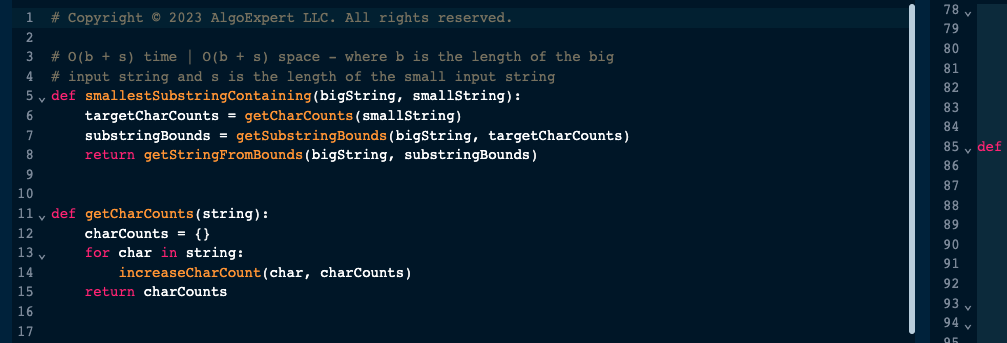
for key in smallStringDict.keys():

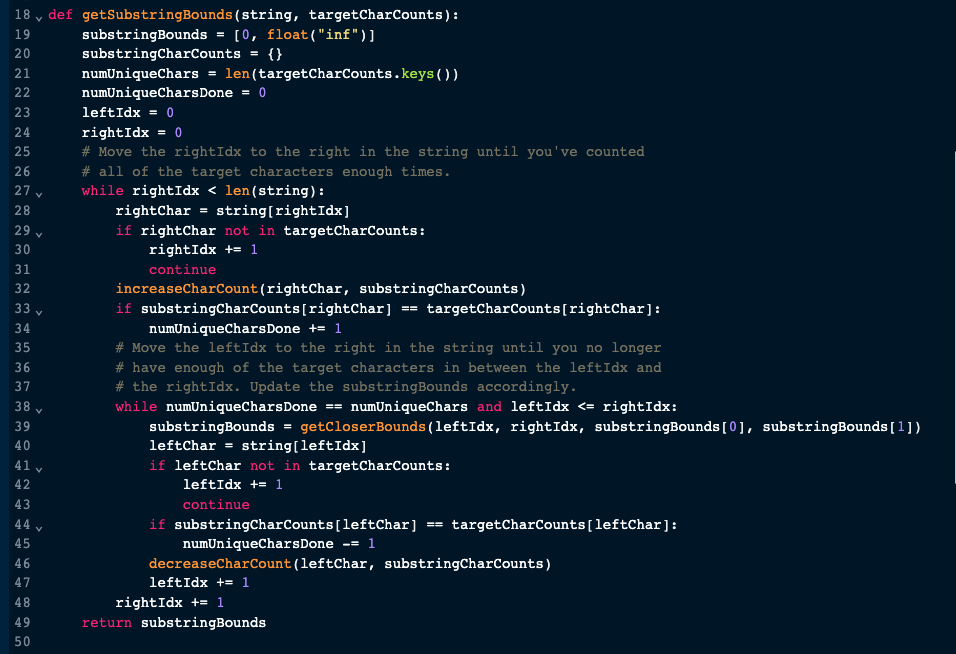
if key not in substringDict.keys() or smallStringDict[key] > substringDict[key]:

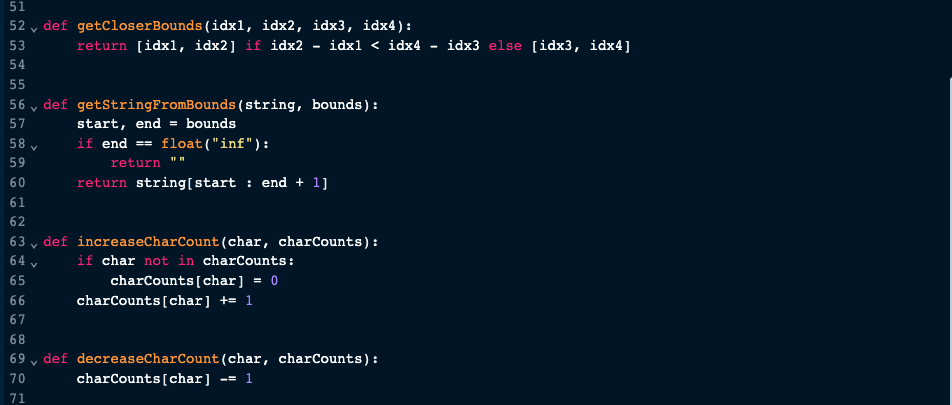
isValid = False

return isValid

Algoexpert Solution:







Most Upvoted Python Solution:

