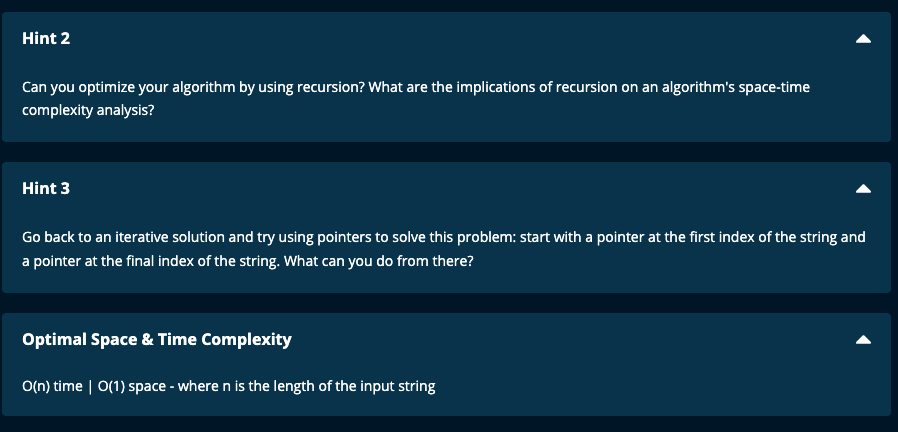
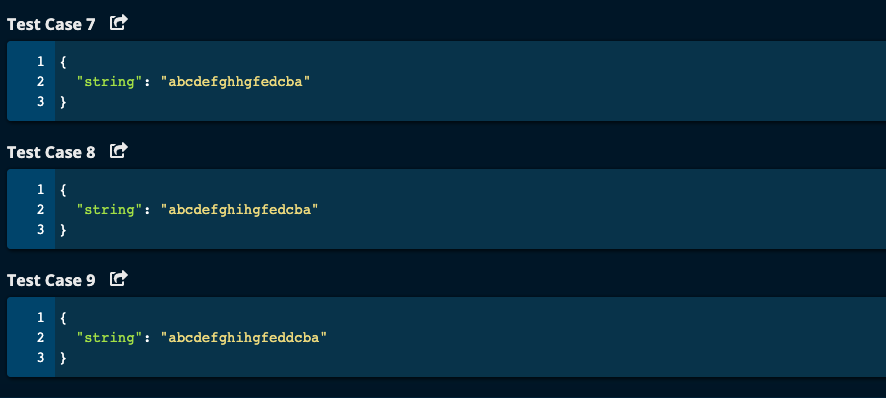
Palindrome Check









My Solutions:

Solution 1:

Reverse the string and compare it with the original string.

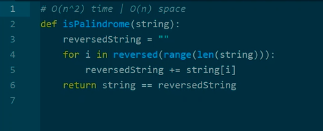
O(n^2) time | O(n) space

def isPalindrome(string):

return string == string[::-1]

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

Algoexpert Solution using reversed string:



Solution 2: Using 2 pointers

Time O(n) | Space O(1)

def isPalindrome(string):

left = 0

right = len(string) - 1

while left < right:

if string[left] == string[right]:

left += 1

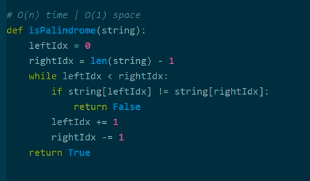
right -= 1

else:

return False

return True

Algoexpert Solution with 2 pointers



Solution 3: Same idea as two to compare the character from both end and come towards the middle

Time O(n) | Space O(1)

def isPalindrome(string):

n = len(string)

for i in range(((n - 1)//2 ) + 1 ):

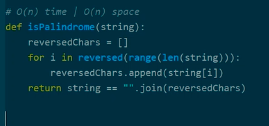
if string[i] != string[n - i - 1]:

return False

return True

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

Algoexpert Solution



Recursive Solution:

Algoexpert Solutions:

# Recursion

O(n) Time | O(n) space due to call stack

def isPalindrome(string, i = 0):

j = len(string) - i - 1

if i >= j:

return True

if string[i] != string[j]:

return False

return isPalindrome(string, i + 1)

# Another solution with recursion

def isPalindrome(string, i = 0):

j = len(string) - i - 1

return True if i >= j else string[i] == string[j] and isPalindrome(string, i + 1)

