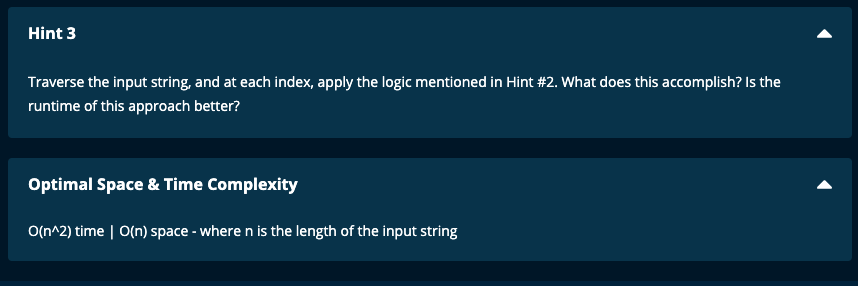
Longest Palindromic Substring. (Medium)





My Solutions:

Solution 1: Time Complexity: O(n^2) | Space Complexity: O(n)

def longestPalindromicSubstring(string):

def isPalindrome(substring):

if len(substring) == 1:

return True

for i in range(len(substring)//2):

if substring[i] != substring[len(substring) - 1 - i]:

return False

return True

print("string = ", string)

n = len(string)

result = ''

for i in range(n):

for j in range(n, i, - 1):

substring = string[i : j]

print("substring = ", substring, "result = ", result)

if isPalindrome(substring):

print("isPalindrome(substring) = ", isPalindrome(substring))

if len(substring) > len(result):

result = substring

print("updated result = ", result)

break

print("final result = ", result)

return result

JJ Notes:

1. Initialize result to be an empty string to hold the result substring.
2. Iterate through the string and consider substrings starting at index i and ending at index j where j is from the end of the string.
3. Using the subfunction isPalindrome(), then check if the substring is a palindrome. For this match the characters in the substring from both ends until the index of half substring length. If a match is not found return false. Otherwise if there all the characters match from both ends, then return True.
4. If the substring is a palindrome, check if its length is greater than the length of result. If so update result with the current substring.
5. Finally return the result.

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

Solution 2:

def longestPalindromicSubstring(string):

def isPalindrome(substring):

if len(substring) == 1:

return True

return substring == substring[::-1]

print("string = ", string)

n = len(string)

result = ''

for i in range(n):

for j in range(n, i, - 1):

substring = string[i : j]

print("substring = ", substring, "result = ", result)

if isPalindrome(substring):

print("isPalindrome(substring) = ", isPalindrome(substring))

if len(substring) > len(result):

result = substring

print("updated result = ", result)

break

print("final result = ", result)

return result

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

JJ Notes:

Same as Solution 1 except for checking Palindrome we check the substring against the substring reversed.

Solution 3:

def longestPalindromicSubstring(string):

if len(string) == 1:

return string

longestSubstring = ""

longestSubstringLength = 0

n = len(string)

for i in range(1, n - 1):

curChar = string[i]

# odd substring

left = i - 1

right = i + 1

curLen = 1

while left >= 0 and right <= n - 1:

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

left -= 1

right += 1

curLen += 2

else:

break

left += 1

right -= 1

if curLen > longestSubstringLength:

longestSubstring = string[left: right + 1]

longestSubstringLength = curLen

# even substring

left = i

right = i + 1

curLen = 0

while left >= 0 and right <= n - 1:

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

left -= 1

right += 1

curLen += 2

else:

break

left += 1

right -= 1

if curLen > longestSubstringLength:

longestSubstring = string[left: right + 1]

longestSubstringLength = curLen

return longestSubstring

JJ Notes:

1. If the length of the string is 1, then return the string.
2. Initialize longestSubstring to an empty string, and longestSubstringLength to 0.
3. Let n be the length of the string.
4. Iterate the string from index 1 to the last but one index.

The current character called curChar is the character of the string at index i.

1. For substrings of odd length, let left be current index – 1, and right be current index + 1.

If the characters at index left and index right are equal then, the left pointer will proceed to the beginning of the string and the right pointer will proceed to the end of the string. The current length of the substring curLen is updated at every step. If curLen is greater than the longestSubstringLength, then update the longestSubstringLength with curLen.

1. For substrings of even length, let left be current index, and right be current index + 1.

If the characters at index left and index right are equal then, the left pointer will proceed to the beginning of the string and the right pointer will proceed to the end of the string. The current length of the substring curLen is updated at every step. If curLen is greater than the longestSubstringLength, then update the longestSubstringLength with curLen.

1. Return the longestSubstring.

Algoexpert Solutions:

Solution 1: Brute Force



Solution 2:

