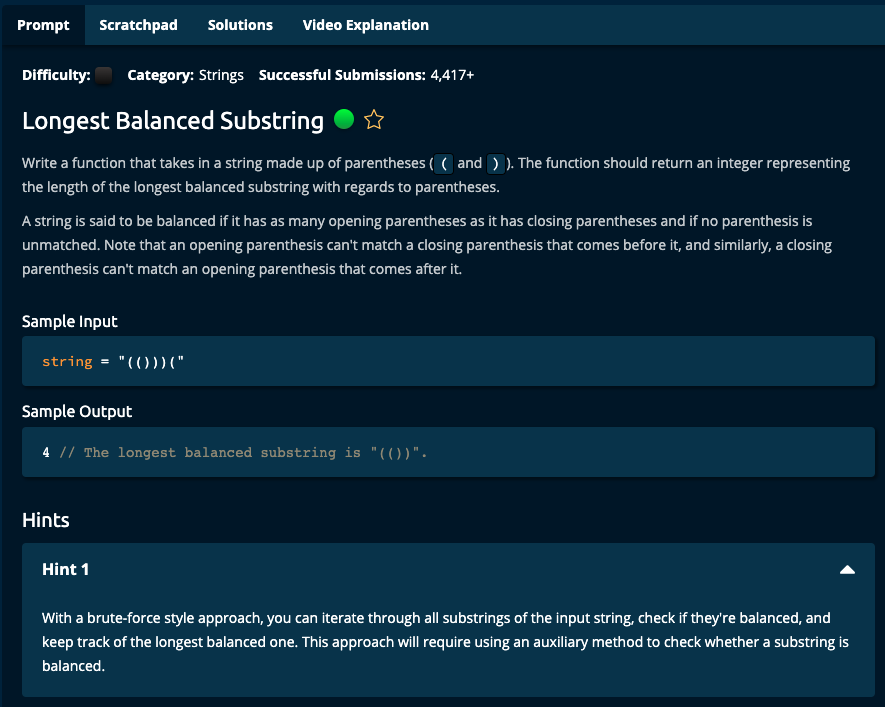
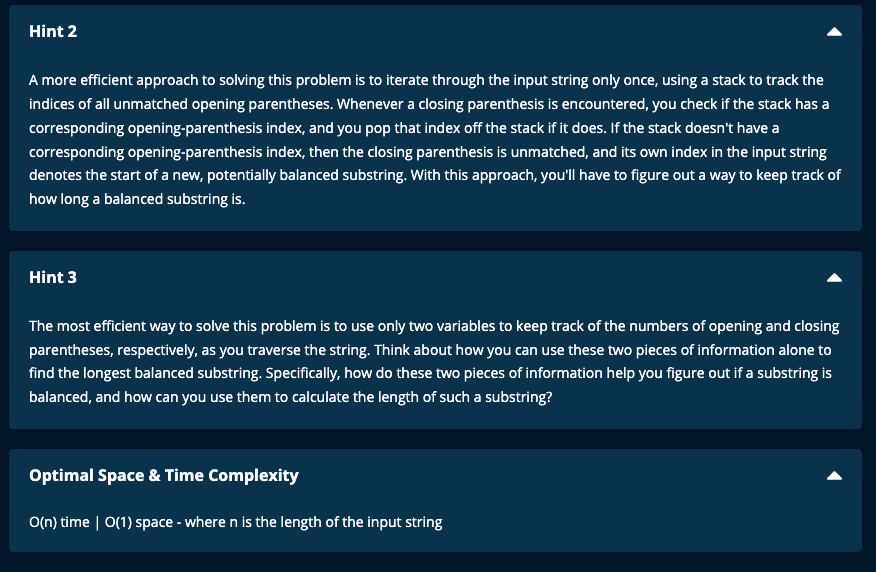
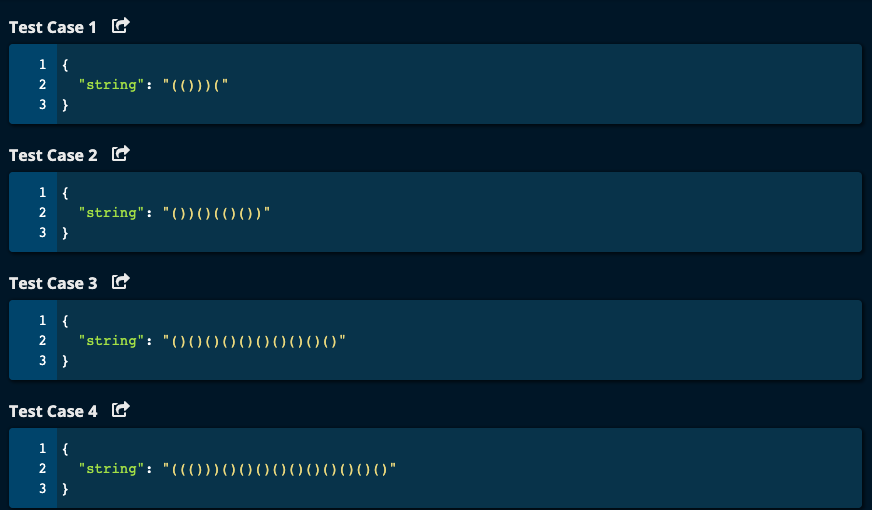
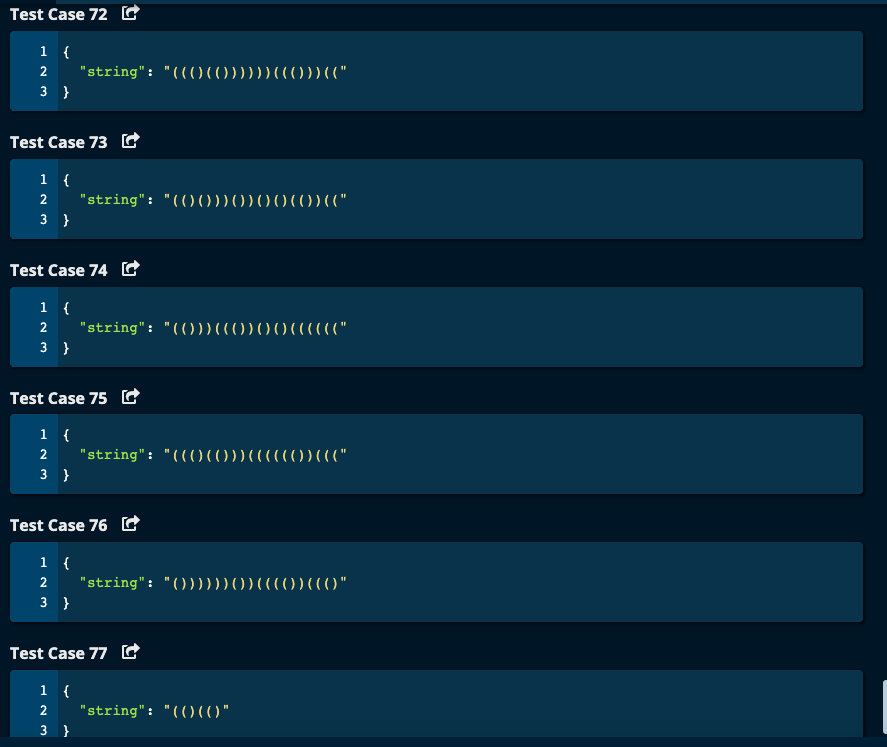
Longest Balanced Substring. (Hard)









My Solution based on Algoexpert :

JJ Notes:

1. Initialize maximum length of the substring to 0
2. Initialize count of opening and closing parentheses to 0.
3. Iterating through the string from left to right, count the opening and closing parentheses. If count of opening and closing parenthesis are equal, then we have a valid substring which is balanced. Then update maxlen.

However, if count of closing parenthesis exceeds the opening parenthesis, then we don’t have a valid substring and we have start over by resetting the counts to 0.

1. Iterating through the string from right to left, count the opening and closing parentheses. If count of opening and closing parenthesis are equal, then we have a valid substring which is balanced. Then update maxlen.

However, if count of closing parenthesis is less than the opening parenthesis, then we don’t have a valid substring and we have start over by resetting the counts to 0.

1. Finally return maxLen.

# Algoexpert Solution with opening and closing parentheses counted in both direction

# Left to Right -- Opening must come before closing to match

# Right to Left -- closing must come before opening to match

# O(n) Time | O(1) Space

def longestBalancedSubstring(string):

# left to right direction

maxLen = 0

opening = 0 # Count of opening parenthesis

closing = 0 # Count of closing parenthesis

for i in range(len(string)):

if string[i] == '(':

opening += 1

else: # string[i] == ')'

closing += 1

if opening == closing:

maxLen = max(maxLen, opening \* 2)

if opening < closing: # now reset opening and closing

opening = 0

closing = 0

# right to left direction

opening = 0

closing = 0

for i in range(len(string) - 1, -1, -1):

if string[i] == ')':

closing += 1

else:

opening += 1

if opening == closing:

maxLen = max(maxLen, opening \* 2)

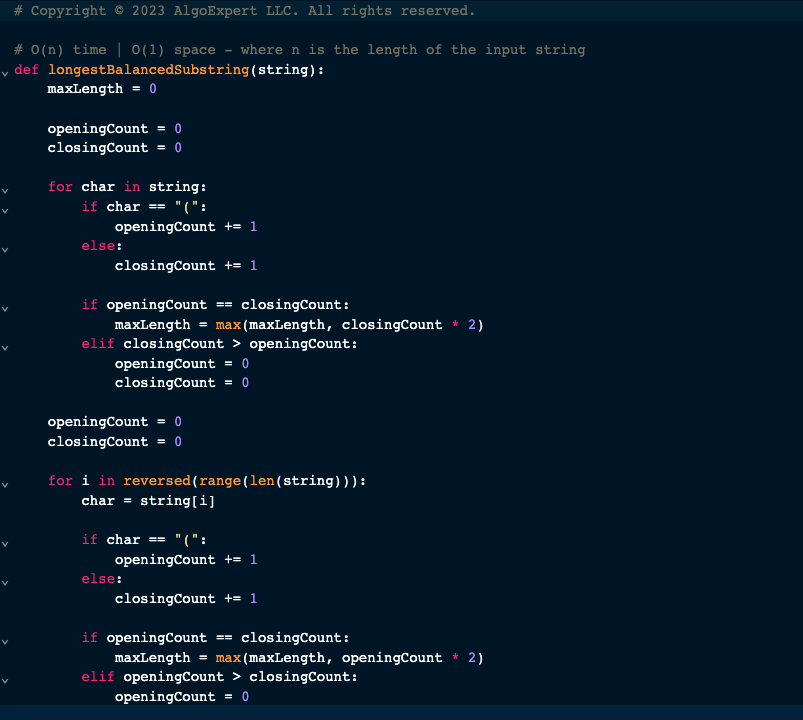
if opening > closing:

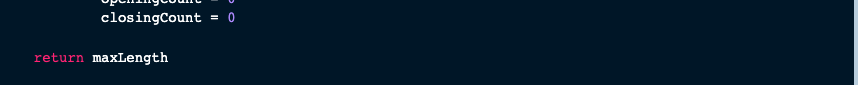
opening = 0

closing = 0

return maxLen

Algoexpert Solution:





Algoexpert Solution using Stack

1. Initialize maxLength of the substring to 0.
2. Initialize the index stack called idxStack to an empty list.
3. Append -1 to the index stack.
4. Iterate through the string. If the character is an opening parenthesis, then append its index to the stack. If it is a closing parenthesis, then pop the stack. If the length of the stack is 0, then append this index of the closing parenthesis to the stack. If not, current length of the substring = current index - the last element in the index stack.
5. Update maxLength to be the max of maxLength and currentLength
6. Return maxLength.

