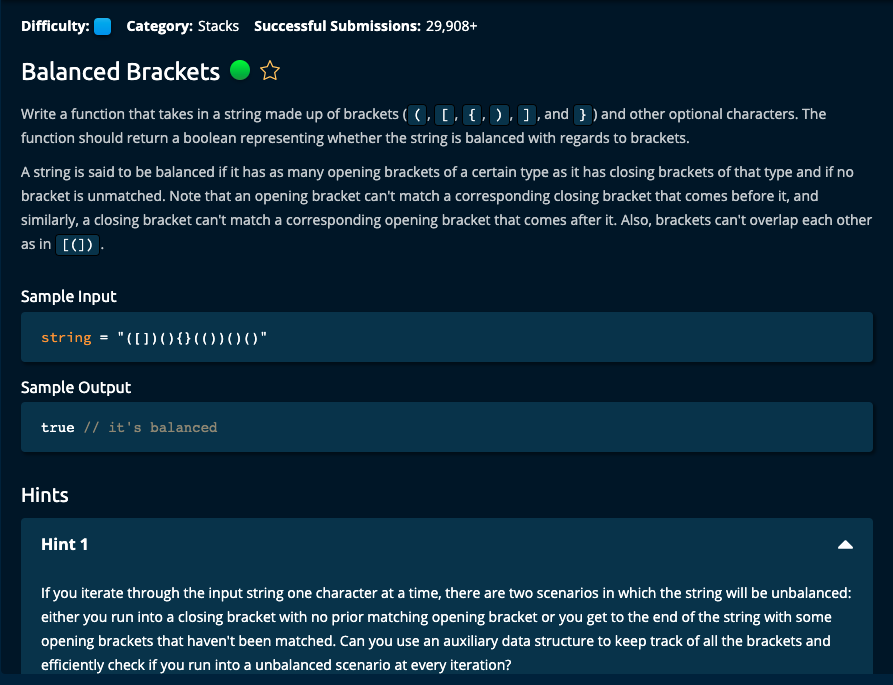
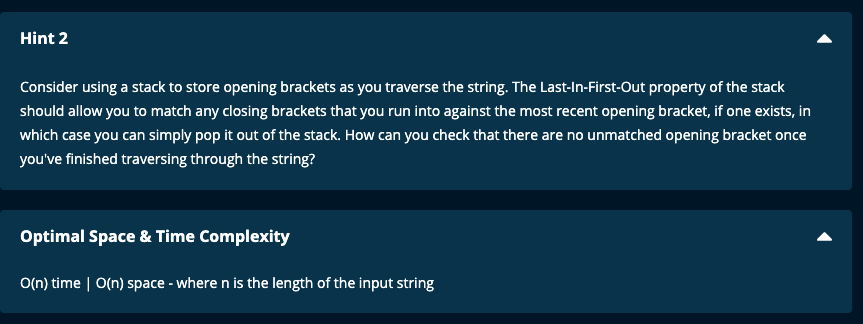
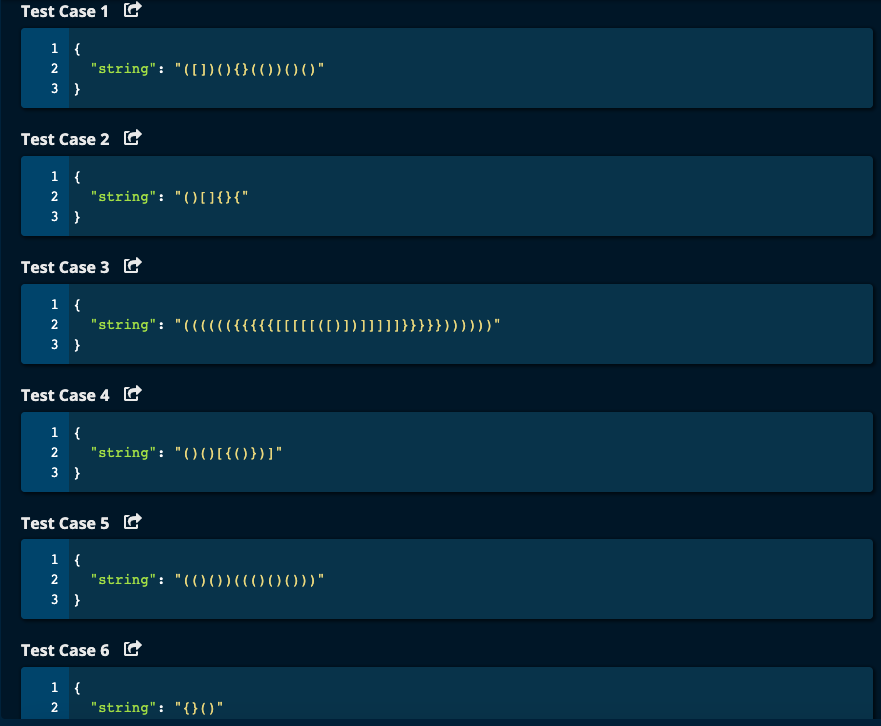
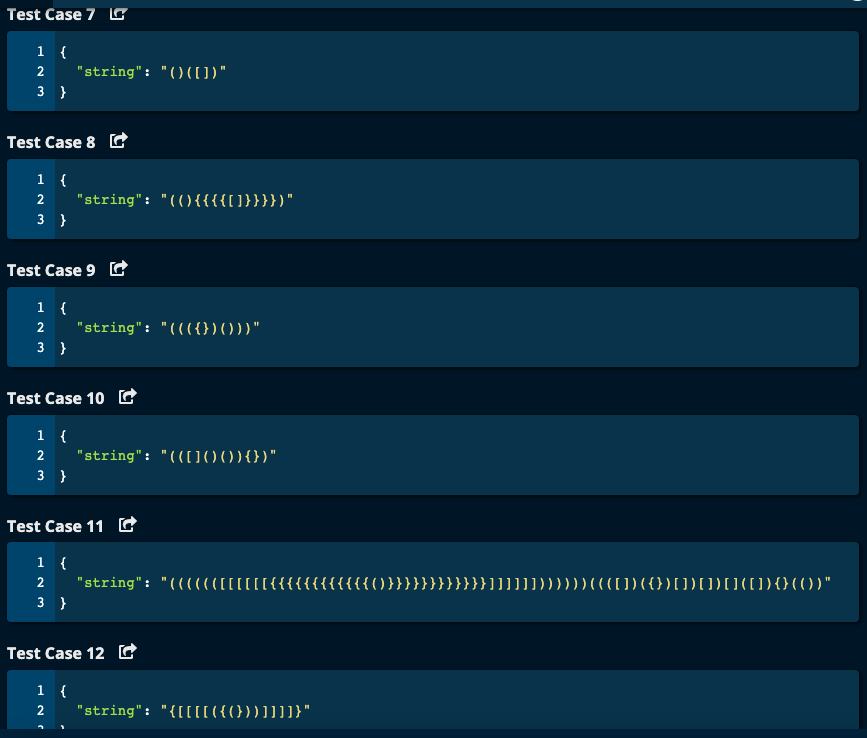
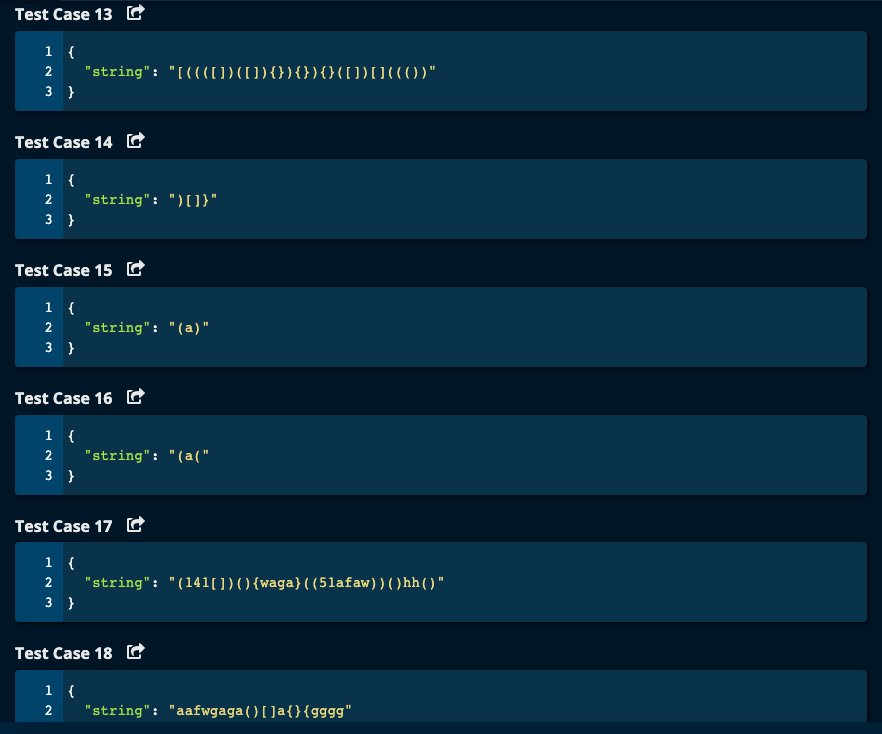
Balanced Brackets. (Medium

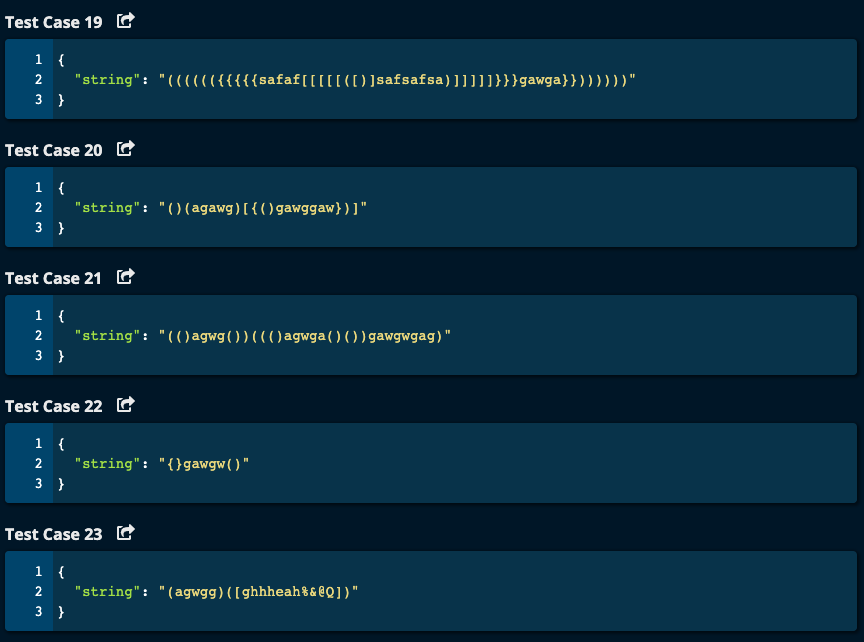












My Solutions:

Solution 1:

def balancedBrackets(string):

if not string:

return True

stack = []

leftBrackets = ['(', '[', '{']

rightBrackets = [ ')', ']', '}']

for i in range(len(string)):

if string[i] in leftBrackets:

stack.append(string[i])

elif string[i] in rightBrackets:

if not stack or (string[i] == ')' and stack.pop() != '(') or \

(string[i] == ']' and stack.pop() != '[') or \

(string[i] == '}' and stack.pop() != '{'):

return False

if stack:

return False

return True

JJ Notes:  
1. If string is empty, return True.

2. Initialize stack to an empty list.

3. Initialize leftBrackets with a list consisting of round, square and curly brackets.

4. Initialize rightBrackets with a list consisting of round, square and curly brackets.

5. Traverse the string. If the character is in leftBrackets, then append the character to the stack.

Otherwise if the character is in rightBrackets, check if the stack is empty or if the right bracket and last character popped out of the stack, is not the corresponding left bracket – in which case return False

6. If we have not returned False so far after traversing the string, if the stack is not empty (i.e. there are left brackets remaining in the stack that are unmatched), return False.

7. Finally, if we have not returned False so far, return True.

Solution 2: # Using a dictionary

def balancedBrackets(string):

stack = []

matchingBrackets = {')' : '(', ']': '[', '}' : '{'}

leftBrackets = matchingBrackets.values()

rightBrackets = matchingBrackets.keys()

for i in range(len(string)):

if string[i] in leftBrackets:

stack.append(string[i])

elif string[i] in rightBrackets:

if not stack or matchingBrackets[string[i]] != stack.pop():

return False

return len(stack) == 0

JJ Notes:

1. Use a dictionary with keys as the right bracket and the values as the corresponding left bracket. Get a list of leftBrackets and rightBrackets from the dictionary.
2. Initialize a stack as an empty list.
3. Traverse through the string and if the character is in leftBrackets, append it to the stack.

If the character is in rightBrackets, and if the stack is empty or if the matchingBrackets of the character is not the last character popped from the stack, return False.

1. Check if the stack is empty. Return True if empty and False otherwise.

Algoexpert Solution (Very similar to my solution 2).

