Problem # 409: Longest Palindrome

<https://leetcode.com/problems/longest-palindrome/>

1. Get the length of s. If it is less than or equal to 1, return the length of the string.
2. Use collections to import Counter put the letters into a dictionary based on the frequency(i.e. number of occurrences of the letter) -- key = letter and value = number of occurrences This step is my\_dict = Counter(s) This is equivalent to the following:

my\_dict = {}

for letter in s:

if letter not in s:

my\_dict[letter] = 1

else:

my\_dict[letter] += 1

1. Now count the number of letters that can be used in the palindrome. Get the list values from the dictionary (my\_dict). values = my\_dict.values(0 Check if each value is odd or even. If even then add the value to count. If odd then add value -1 to count, since we want the biggest even number less than the odd value
2. Finally, if there are odd letters left in the string, then we can increase count by 1 more, since that odd value can be in the middle of the palindrome. If there are no odd values left, return the count.
3. My code is given below:

from collections import Counter

class Solution:

def longestPalindrome(self, s: str) -> int:

n = len(s) # length of the string

if n <= 1:

return(n)

my\_dict = Counter(s) # Put the string in a dictionary and get the count of each letter

values = my\_dict.values() # get the list of values in the dictionary

count = 0 # count of letters used to build the palindrome

for val in values:

if val % 2 == 0: # if val is even then add this to count

count += val

else: # val is odd and so take the largest even number of letters lower than val

count += val - 1

if n - count > 0: # if there are more odd numbers left

return(count + 1) # we can have one odd value in the palindrome

return(count) # else case when there are no odd values left in s

```