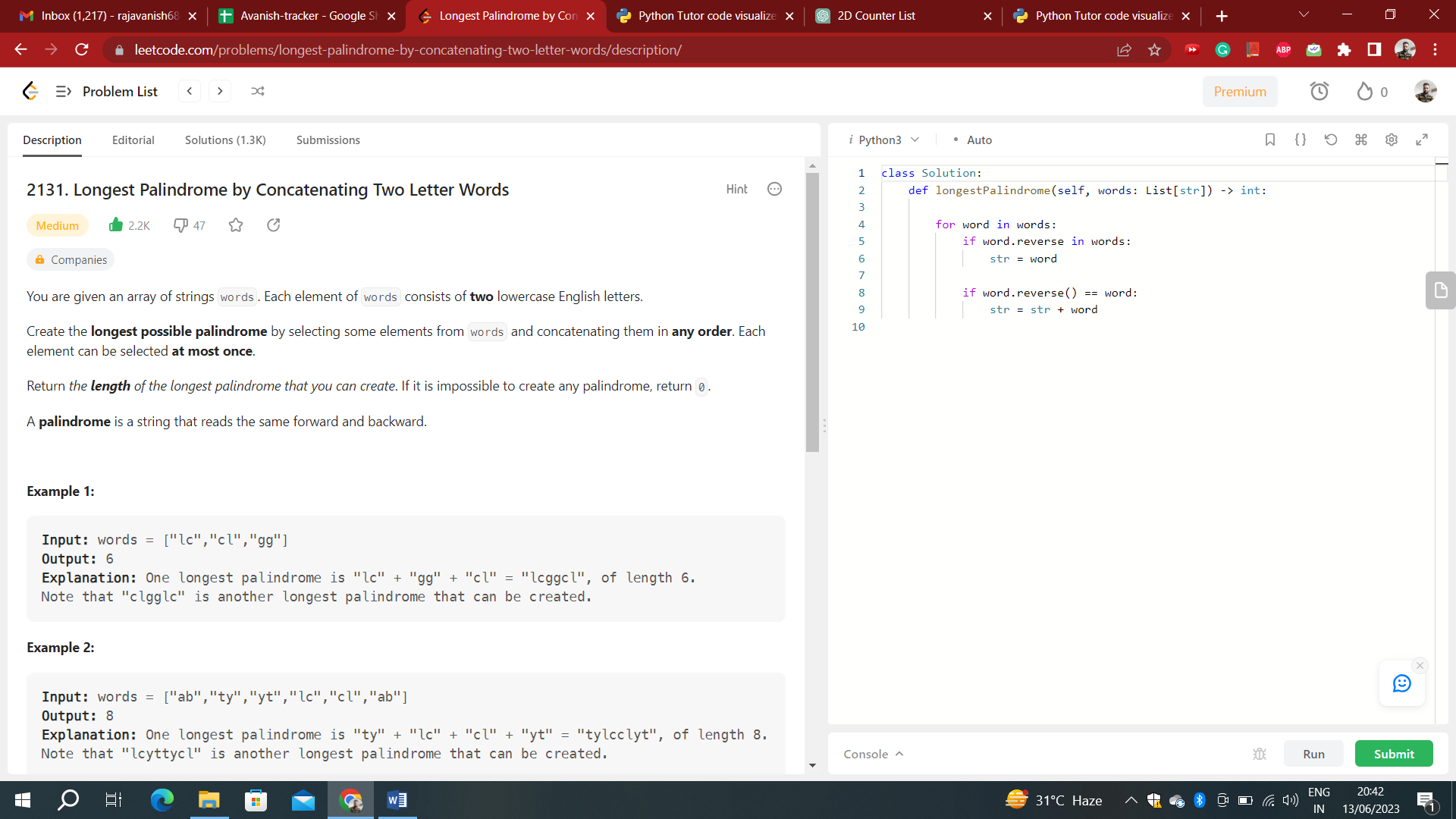
**Longest Palindrome by Concatenating Two Letter Words**

<https://leetcode.com/problems/longest-palindrome-by-concatenating-two-letter-words/description/>



**Time complexity: O(n)**

***Explanation:***

2 letter words can be of 2 types:

1. Where both letters are same
2. Where both letters are different

Based on the above information:

* If we are able to find the mirror of a word, ans += 4
* The variable unpaired is used to store the number of unpaired words with both letters same.
* Unpaired here means a word that has not found its mirror word.
* At the end if unpaired same letter words are > 0, we can use one of them as the center of the palindromic string.

from collections import defaultdict

class Solution:

def longestPalindrome(self, words):

m = defaultdict(int)

unpaired = ans = 0

for w in words:

if w[0] == w[1]:

if m[w] > 0:

unpaired -= 1

m[w] -= 1

ans += 4

else:

m[w] += 1

unpaired += 1

else:

if m[w[::-1]] > 0: # w[::-1] -> reversed w

ans += 4

m[w[::-1]] -= 1

else:

m[w] += 1

if unpaired > 0: ans += 2

return ans

s = Solution()

s.longestPalindrome(words = ["ab","ty","yt","gg","gg","lc","cl","ab","ba"])

**Time complexity: O(n)**

class Solution:

def longestPalindrome(self, words):

counter, ans = [[0] \* 26 for \_ in range(26)], 0

for w in words:

a, b = ord(w[0]) - ord('a'), ord(w[1]) - ord('a')

if counter[b][a]:

ans += 4

counter[b][a] -= 1

else:

counter[a][b] += 1

for i in range(26):

if counter[i][i]:

ans += 2

break

return ans

s = Solution()

s.longestPalindrome(words = ["lc","cl","gg"])



