Text Justification

**Approach**

Unlike most problems on LeetCode, this is one that can be solved by just doing exactly what the problem statement is telling us to do.

You don't need any data structures or algorithmic tricks to solve this problem. The point of this problem is to test your ability to quickly write clean code while navigating edge cases.

Divide it into 2 subproblems:

1. Find all words that set in each line
2. Justify words in each line

***1) Finding all words for each line***

getWords(i): here i denotes the start of the word for each line

Line = []

# keep adding new words to line starting from i such that length of line does not exceed maxWidth.

# condition to check if we exceeded maxWidth: count + len(word[i]) <= maxWidth.

***2) Justify words in each lin***e

# 3 cases

Case 1: last line or 1 word in line, no justification

Case 2: equally spaced : spacePeWord = remainingSpaces // (noOfWords-1)

Case 3 unequally spaced: starting from first word(from left) Keep adding extra spaces until remainingSpaces % (noOfWords-1) reaches 0

**CODE**

class Solution:

def getWords(self,i):

count = 0

currentLine = []

while i < len(self.words) and count + len(self.words[i]) <= self.maxWidth:

currentLine.append(self.words[i])

count += len(self.words[i]) + 1

i+=1

return currentLine

def justifyLine(self, currentLine, i):

# find length of preserved line(len\_of\_each\_word + single space after each word except last)

lineLength = -1

for word in currentLine:

lineLength += len(word) + 1

remSpaces = self.maxWidth - lineLength

justifiedLine = ""

# CASE 1: last line or 1 word no justification to be done

if i == self.n or len(currentLine) == 1:

spaces = " " \* (remSpaces)

return " ".join(currentLine) + spaces

noOfWords = len(currentLine) - 1

spacePerWord = remSpaces // noOfWords

extraSpace = remSpaces % noOfWords

# CASE 2 : number of spaces on a line does not divide evenly between words

for k in range(extraSpace):

currentLine[k] += " "

# CASE 3 : equally spaced

for k in range(noOfWords):

currentLine[k] += " " \* spacePerWord

return " ".join(currentLine)

def fullJustify(self, words: List[str], maxWidth: int) -> List[str]:

i = 0

self.words = words

self.maxWidth = maxWidth

self.res = []

self.n = len(words)

while i < len(words):

# find all words in current line using getWords(start\_index)

currentLine = self.getWords(i)

# increment i to point to the first word of next line

i += len(currentLine)

# arrange/ justify words in currentLine

self.res.append(self.justifyLine(currentLine, i))

return self.res