class Solution:

def strStr(self, haystack: str, needle: str) -> int:

if len(needle) == 0:

return 0

for i in range(len(haystack)):

if haystack[i:i+len(needle)] == needle:

return i

return -1

class Solution:

def canConstruct(self, ransomNote: str, magazine: str) -> bool:

# instead of using a dict let's just use an array that's indexed by the ascii value from a

# example: a-a = index 0, f-a = index 5

d = [0]\*26

for x in magazine:

d[ord(x)-ord('a')] += 1

for x in ransomNote:

d[ord(x)-ord('a')] -= 1

if d[ord(x)-ord('a')] < 0:

return False

return True

class Solution:

def groupAnagrams(self, strs: List[str]) -> List[List[str]]:

if len(strs) == 0:

return

# bigO of n?

wordDict = {}

for word in strs:

d = [0]\*26

for ltr in word:

# get id of word via letter counting

d[ord(ltr) - ord('a')] += 1

# repr(d) is the key for our outer dictionary

if repr(d) in wordDict:

wordDict[repr(d)].append(word)

else:

wordDict[repr(d)] = [word]

home = []

for k in wordDict:

home.append(wordDict[k])

print(home)

return home