139: Work Break

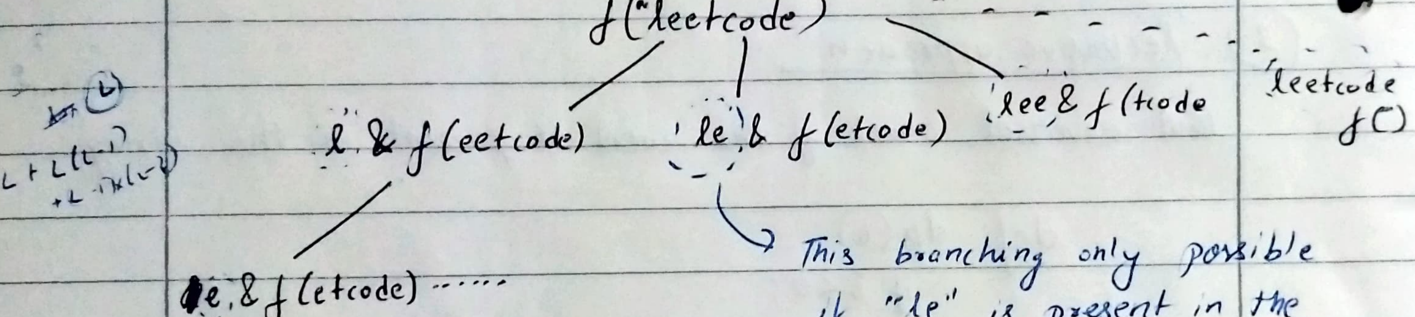
Breaking the given word in such a way that all broken words are present in the dictionary

Eg: "leetcode" dict = ['leet', 'code'] ⇒ True

"catsanddogs" dict = ['cat', 'cats', 'and', 'sand', 'dogs'] = False

↳ no break leads to all words in dictionary.

① Brute force approach



This branching only possible if "le" is present in the dictionary

→ Worst time complexity = n^n, if each character is in dictionary.

→ Can be improved through "lru_cache (max_size = None)" i.e memorization.

→ ~~Can be improved by~~

Can be improved by not going above the length of maximum word present in the dictionary.

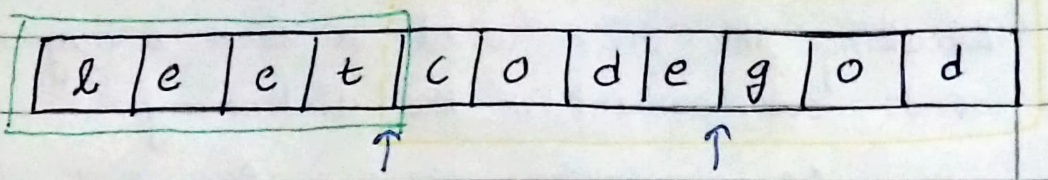
If the dictionary contains all single characters.

{leetcode, dict = {l, e, t, c, o, d}

then no need to check anything above length 1

i.e, no recursion for le & f(etcode)

(2) Recursive approach (My approach)



→ Find the first word in the string "leet code go d"
(present in dictionary)
→ Do the same for the left-over string.
→ Keep doing it until the end when string is empty
→ If recursion leads to empty string then output is true, otherwise output comes out to be false.

(3) Recursive approach

But reversal, check if the word is present in the string

```
def dp(i):
    if i<0:
        return True

    for word in wordDict:
        if s[i-len(word)+1 : i+1] == word
            and dp(i-len(word)):
            return True

    return False.

dp(len(s)-1)
```

This ~~problem~~ solution can be converted to a "tabulation".