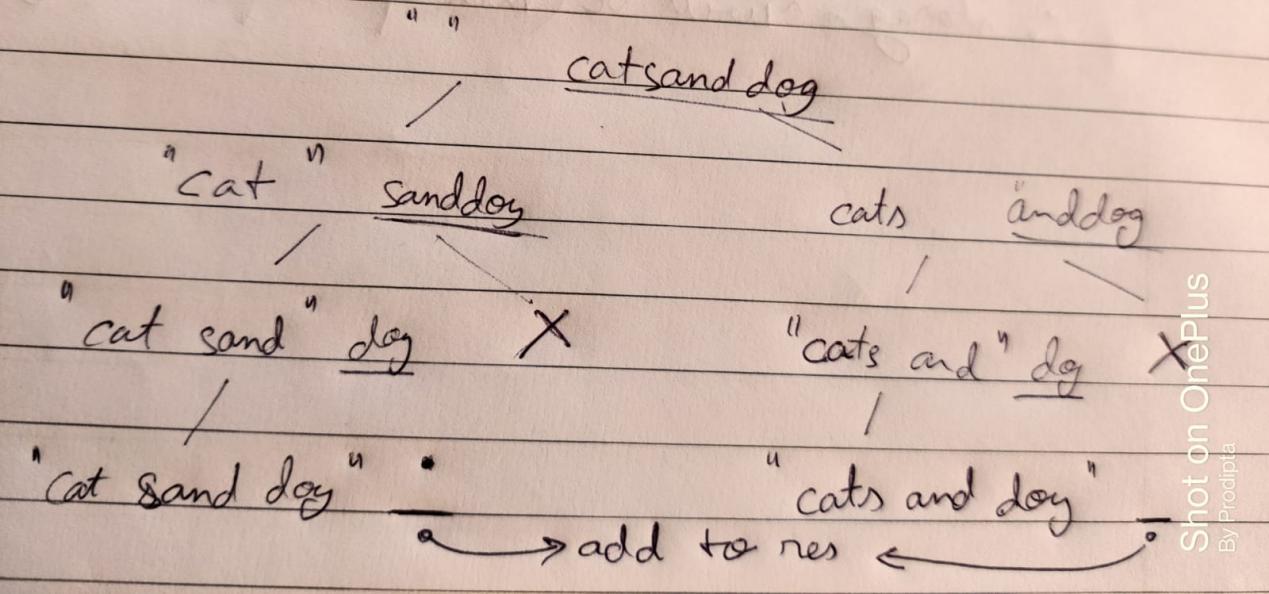
Question: https://leetcode.com/problems/word-break-ii/

So we use recursion and backtracking to solve the problem.

We use a helper function, it traverses the string and checks if any substring starting from 0th index exists in the dict or noe, if exists then we search the rest of the string for other such elements now when we have completed searching and searchable string is empty that is its length is 0; we add the string formed my dict elements to the res list.



Code:  
class Solution {

List<String> res = new ArrayList<>();

public List<String> wordBreak(String s, List<String> wordDict) {

helper("", s, wordDict);

return res;

}

public void helper(String el, String s, List<String> wordDict){

String key = "";

if(s.length()==0){

res.add(el);

}

for(int i=0; i<s.length(); i++){

char c = s.charAt(i);

key+=c;

if(wordDict.contains(key)){

String newel = el.equals("")?key:el+" "+key;

helper(newel, s.substring(i+1,s.length()), wordDict);

}

}

return;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>