Problem 320: Generalized Abbreviation

Problem Information

Difficulty: Medium
Acceptance Rate: 60.24%
Paid Only: Yes
Tags: String, Backtracking, Bit Manipulation

Problem Description

A word's **generalized abbreviation** can be constructed by taking any number of **non-overlapping** and **non-adjacent** substrings and replacing them with their respective lengths.

* For example, `"abcde"` can be abbreviated into: * `"a3e"` (`"bcd"` turned into `"3"`) * `"1bcd1"` (`"a"` and `"e"` both turned into `"1"`) * `"5"` (`"abcde"` turned into `"5"`) * `"abcde"` (no substrings replaced) * However, these abbreviations are **invalid** : * `"23"` (`"ab"` turned into `"2"` and `"cde"` turned into `"3"`) is invalid as the substrings chosen are adjacent. * `"22de"` (`"ab"` turned into `"2"` and `"bc"` turned into `"2"`) is invalid as the substring chosen overlap.

Given a string `word`, return _a list of all the possible**generalized abbreviations** of_ `word`. Return the answer in **any order**.

Example 1:

Input: word = "word" **Output:** ["4","3d","2r1","2rd","1o2","1o1d","1or1","1ord","w3","w2d","w1r1","w1rd","wo2","wo1d","wor1","word"]

Example 2:

Input: word = "a" **Output:** ["1","a"]

Constraints:

* `1 <= word.length <= 15` * `word` consists of only lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
vector<string> generateAbbreviations(string word) {

}
};
```

Java:

```
class Solution {
public List<String> generateAbbreviations(String word) {

}
}
```

Python3:

```
class Solution:
def generateAbbreviations(self, word: str) -> List[str]:
```