Problem 1096: Brace Expansion II

Problem Information

Difficulty: Hard
Acceptance Rate: 63.34%
Paid Only: No
Tags: Hash Table, String, Backtracking, Stack, Breadth-First Search, Sorting

Problem Description

Under the grammar given below, strings can represent a set of lowercase words. Let `R(expr)` denote the set of words the expression represents.

The grammar can best be understood through simple examples:

* Single letters represent a singleton set containing that word. * `R("a") = {"a"}` * `R("w") = {"w"}` * When we take a comma-delimited list of two or more expressions, we take the union of possibilities. * `R("{a,b,c}") = {"a","b","c"}` * `R("{{a,b},{b,c}}") = {"a","b","c"}` (notice the final set only contains each word at most once) * When we concatenate two expressions, we take the set of possible concatenations between two words where the first word comes from the first expression and the second word comes from the second expression. * `R("{a,b}{c,d}") = {"ac","ad","bc","bd"}` * `R("a{b,c}{d,e}f{g,h}") = {"abdfg", "abdfh", "abefg", "abefh", "acdfg", "acdfh", "acefg", "acefh"}`

Formally, the three rules for our grammar:

* For every lowercase letter `x`, we have `R(x) = {x}`. * For expressions `e1, e2, ... , ek` with `k >= 2`, we have `R({e1, e2, ...}) = R(e1) ∪ R(e2) ∪ ...` * For expressions `e1` and `e2`, we have `R(e1 + e2) = {a + b for (a, b) in R(e1) × R(e2)}`, where `+` denotes concatenation, and `×` denotes the cartesian product.

Given an expression representing a set of words under the given grammar, return _the sorted list of words that the expression represents_.

Example 1:

Input: expression = "{a,b}{c,{d,e}}" **Output:** ["ac","ad","ae","bc","bd","be"]

Example 2:

Input: expression = "{{a,z},a{b,c},{ab,z}}" **Output:** ["a","ab","ac","z"] **Explanation:** Each distinct word is written only once in the final answer.

Constraints:

* `1 <= expression.length <= 60` * `expression[i]` consists of `'{'`, `'}'`, `','`or lowercase English letters. * The given `expression` represents a set of words based on the grammar given in the description.

Code Snippets

C++:

```
class Solution {
public:
vector<string> braceExpansionII(string expression) {

}
};
```

Java:

```
class Solution {
public List<String> braceExpansionII(String expression) {

}
}
```

Python3:

```
class Solution:
def braceExpansionII(self, expression: str) -> List[str]:
```