Problem 408: Valid Word Abbreviation

Problem Information

Difficulty: Easy
Acceptance Rate: 36.98%
Paid Only: Yes
Tags: Two Pointers, String

Problem Description

A string can be **abbreviated** by replacing any number of **non-adjacent** , **non-empty** substrings with their lengths. The lengths **should not** have leading zeros.

For example, a string such as `"substitution"` could be abbreviated as (but not limited to):

* `"s10n"` (`"s _ubstitutio_ n"`) * `"sub4u4"` (`"sub _stit_ u _tion_ "`) * `"12"` (`"_substitution_ "`) * `"su3i1u2on"` (`"su _bst_ i _t_ u _ti_ on"`) * `"substitution"` (no substrings replaced)

The following are **not valid** abbreviations:

* `"s55n"` (`"s _ubsti_ _tutio_ n"`, the replaced substrings are adjacent) * `"s010n"` (has leading zeros) * `"s0ubstitution"` (replaces an empty substring)

Given a string `word` and an abbreviation `abbr`, return _whether the string**matches** the given abbreviation_.

A **substring** is a contiguous **non-empty** sequence of characters within a string.

Example 1:

Input: word = "internationalization", abbr = "i12iz4n" **Output:** true **Explanation:** The word "internationalization" can be abbreviated as "i12iz4n" ("i _nternational_ iz _atio_ n").

Example 2:

Input: word = "apple", abbr = "a2e" **Output:** false **Explanation:** The word "apple" cannot be abbreviated as "a2e".

Constraints:

* `1 <= word.length <= 20` * `word` consists of only lowercase English letters. * `1 <= abbr.length <= 10` * `abbr` consists of lowercase English letters and digits. * All the integers in `abbr` will fit in a 32-bit integer.

Code Snippets

C++:

```
class Solution {
public:
bool validWordAbbreviation(string word, string abbr) {

}
};
```

Java:

```
class Solution {
public boolean validWordAbbreviation(String word, String abbr) {

}
}
```

Python3:

```
class Solution:
def validWordAbbreviation(self, word: str, abbr: str) -> bool:
```