Description | Editorial | Solutions | Submissions

408. Valid Word Abbreviation Premium

Solved

Easy | Topics | Companies

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 nternationalizatio 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.

Seen this question in a real interview before? 1/5

Yes No

Accepted 183.4K | Submissions 512.7K | Acceptance Rate 35.8%

Topics

Companies

Similar Questions

Discussion (45)

713 45

Code

Python3 • Auto

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

        keys = []
        numx = ""
        for i in range(0, len(abbr)):
            if abbr[i].isdigit():
                numx += abbr[i]
            else:
                if numx != "":
                    if numx[0] == "0":
                        return False
                    keys.append(numx)
                    numx = ""

                keys.append(abbr[i])
        if numx != "":
            if numx[0] == "0":
                return False
            keys.append(numx)

        i, j = 0, 0

        try:
            while i < len(keys):
                if keys[i].isdigit():
                    j += int(keys[i])
                    i += 1
                elif keys[i] == word[j]:
                    j += 1
                    i += 1
                else:
                    return False
        except:
            return False

        if j == len(word):
            return True
        else:
            return False
```

Saved Ln 8, Col 32

Testcase | Test Result