<https://www.cnblogs.com/cnoodle/p/16454352.html>

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.

**Attempt 1: 2024-06-10**

**Solution 1: Two Pointers (10 min)**

class Solution {

public boolean validWordAbbreviation(String word, String abbr) {

// corner case

if (word == null || abbr == null) {

return false;

}

// normal case

int i = 0;

int j = 0;

while (i < word.length() && j < abbr.length()) {

if (word.charAt(i) == abbr.charAt(j)) {

i++;

j++;

} else if (Character.isDigit(abbr.charAt(j)) && abbr.charAt(j) != '0') {

int num = 0;

while (j < abbr.length() && Character.isDigit(abbr.charAt(j))) {

num = num \* 10 + abbr.charAt(j) - '0';

j++;

}

i += num;

} else {

return false;

}

}

return i == word.length() && j == abbr.length();

}

}

Time Complexity: O(N)

Space Complexity: O(1)

**Refer to**

<https://www.cnblogs.com/cnoodle/p/16454352.html>

**思路是双指针，两个指针 i, j 分别指向 word 和 abbr。如果两边指向的字母相同，则分别往前走一步；如果 abbr 这一边指向了一个不是 0 开头的数字，则我们算一下这个数字 num 到底是多少，然后让 i 指针往前走 num 步。如果顺利，最后两个指针应该是同时到达 word 和 abbr 的尾部。**

时间O(n)

空间O(1)

Java实现

class Solution {

public boolean validWordAbbreviation(String word, String abbr) {

// corner case

if (word == null || abbr == null) {

return false;

}

// normal case

int i = 0;

int j = 0;

while (i < word.length() && j < abbr.length()) {

if (word.charAt(i) == abbr.charAt(j)) {

i++;

j++;

} else if (Character.isDigit(abbr.charAt(j)) && abbr.charAt(j) != '0') {

int num = 0;

while (j < abbr.length() && Character.isDigit(abbr.charAt(j))) {

num = num \* 10 + abbr.charAt(j) - '0';

j++;

}

i += num;

} else {

return false;

}

}

return i == word.length() && j == abbr.length();

}

}