Problem 30: Substring with Concatenation of All Words

Problem Information

Difficulty: Hard
Acceptance Rate: 33.57%
Paid Only: No
Tags: Hash Table, String, Sliding Window

Problem Description

You are given a string `s` and an array of strings `words`. All the strings of `words` are of **the same length**.

A **concatenated string** is a string that exactly contains all the strings of any permutation of `words` concatenated.

* For example, if `words = ["ab","cd","ef"]`, then `"abcdef"`, `"abefcd"`, `"cdabef"`, `"cdefab"`, `"efabcd"`, and `"efcdab"` are all concatenated strings. `"acdbef"` is not a concatenated string because it is not the concatenation of any permutation of `words`.

Return an array of _the starting indices_ of all the concatenated substrings in `s`. You can return the answer in **any order**.

Example 1:

Input: s = "barfoothefoobarman", words = ["foo","bar"]

Output: [0,9]

Explanation:

The substring starting at 0 is `"barfoo"`. It is the concatenation of `["bar","foo"]` which is a permutation of `words`. The substring starting at 9 is `"foobar"`. It is the concatenation of `["foo","bar"]` which is a permutation of `words`.

Example 2:

Input: s = "wordgoodgoodgoodbestword", words = ["word","good","best","word"]

Output: []

Explanation:

There is no concatenated substring.

Example 3:

Input: s = "barfoofoobarthefoobarman", words = ["bar","foo","the"]

Output: [6,9,12]

Explanation:

The substring starting at 6 is `"foobarthe"`. It is the concatenation of `["foo","bar","the"]`. The substring starting at 9 is `"barthefoo"`. It is the concatenation of `["bar","the","foo"]`. The substring starting at 12 is `"thefoobar"`. It is the concatenation of `["the","foo","bar"]`.

Constraints:

* `1 <= s.length <= 104` * `1 <= words.length <= 5000` * `1 <= words[i].length <= 30` * `s` and `words[i]` consist of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
vector<int> findSubstring(string s, vector<string>& words) {

}
};
```

Java:

```
class Solution {
public List<Integer> findSubstring(String s, String[] words) {

}
}
```

Python3:

```
class Solution:
def findSubstring(self, s: str, words: List[str]) -> List[int]:
```