Problem 3458: Select K Disjoint Special Substrings

Problem Information

Difficulty: Medium
Acceptance Rate: 18.46%
Paid Only: No
Tags: Hash Table, String, Dynamic Programming, Greedy, Sorting

Problem Description

Given a string `s` of length `n` and an integer `k`, determine whether it is possible to select `k` disjoint **special substrings**.

A **special substring** is a substring where:

* Any character present inside the substring should not appear outside it in the string. * The substring is not the entire string `s`.

Note that all `k` substrings must be disjoint, meaning they cannot overlap.

Return `true` if it is possible to select `k` such disjoint special substrings; otherwise, return `false`.

Example 1:

Input: s = "abcdbaefab", k = 2

Output: true

Explanation:

* We can select two disjoint special substrings: `"cd"` and `"ef"`. * `"cd"` contains the characters `'c'` and `'d'`, which do not appear elsewhere in `s`. * `"ef"` contains the characters `'e'` and `'f'`, which do not appear elsewhere in `s`.

Example 2:

Input: s = "cdefdc", k = 3

Output: false

Explanation:

There can be at most 2 disjoint special substrings: `"e"` and `"f"`. Since `k = 3`, the output is `false`.

Example 3:

Input: s = "abeabe", k = 0

Output: true

Constraints:

* `2 <= n == s.length <= 5 * 104` * `0 <= k <= 26` * `s` consists only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
bool maxSubstringLength(string s, int k) {

}
};
```

Java:

```
class Solution {
public boolean maxSubstringLength(String s, int k) {

}
```

```
}
```

Python3:

```
class Solution:
def maxSubstringLength(self, s: str, k: int) -> bool:
```