Problem 3144: Minimum Substring Partition of Equal Character Frequency

Problem Information

Difficulty: Medium
Acceptance Rate: 39.57%
Paid Only: No
Tags: Hash Table, String, Dynamic Programming, Counting

Problem Description

Given a string `s`, you need to partition it into one or more **balanced** substrings. For example, if `s == "ababcc"` then `("abab", "c", "c")`, `("ab", "abc", "c")`, and `("ababcc")` are all valid partitions, but `("a", **" bab"**, "cc")`, `(**" aba"**, "bc", "c")`, and `("ab", **" abcc"**)` are not. The unbalanced substrings are bolded.

Return the **minimum** number of substrings that you can partition `s` into.

Note: A **balanced** string is a string where each character in the string occurs the same number of times.

Example 1:

Input: s = "fabccddg"

Output: 3

Explanation:

We can partition the string `s` into 3 substrings in one of the following ways: `("fab, "ccdd", "g")`, or `("fabc", "cd", "dg")`.

Example 2:

Input: s = "abababaccddb"

Output: 2

Explanation:

We can partition the string `s` into 2 substrings like so: `("abab", "abaccddb")`.

Constraints:

* `1 <= s.length <= 1000` * `s` consists only of English lowercase letters.

Code Snippets

C++:

```
class Solution {
public:
int minimumSubstringsInPartition(string s) {

}
};
```

Java:

```
class Solution {
public int minimumSubstringsInPartition(String s) {

}
}
```

Python3:

```
class Solution:
def minimumSubstringsInPartition(self, s: str) -> int:
```