Problem 3389: Minimum Operations to Make Character Frequencies Equal

Problem Information

Difficulty: Hard
Acceptance Rate: 25.92%
Paid Only: No
Tags: Hash Table, String, Dynamic Programming, Counting, Enumeration

Problem Description

You are given a string `s`.

A string `t` is called **good** if all characters of `t` occur the same number of times.

You can perform the following operations **any number of times** :

* Delete a character from `s`. * Insert a character in `s`. * Change a character in `s` to its next letter in the alphabet.

Note that you cannot change `'z'` to `'a'` using the third operation.

Return __ the **minimum** number of operations required to make `s` **good**.

Example 1:

Input: s = "acab"

Output: 1

Explanation:

We can make `s` good by deleting one occurrence of character `'a'`.

Example 2:

Input: s = "wddw"

Output: 0

Explanation:

We do not need to perform any operations since `s` is initially good.

Example 3:

Input: s = "aaabc"

Output: 2

Explanation:

We can make `s` good by applying these operations:

* Change one occurrence of `'a'` to `'b'` * Insert one occurrence of `'c'` into `s`

Constraints:

* `3 <= s.length <= 2 * 104` * `s` contains only lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

```
}
}
```

Python3:

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