Problem 1647: Minimum Deletions to Make Character Frequencies Unique

Problem Information

Difficulty: Medium
Acceptance Rate: 61.41%
Paid Only: No
Tags: Hash Table, String, Greedy, Sorting

Problem Description

A string `s` is called **good** if there are no two different characters in `s` that have the same **frequency**.

Given a string `s`, return _the**minimum** number of characters you need to delete to make _`s` _**good**._

The **frequency** of a character in a string is the number of times it appears in the string. For example, in the string `"aab"`, the **frequency** of `'a'` is `2`, while the **frequency** of `'b'` is `1`.

Example 1:

Input: s = "aab" **Output:** 0 **Explanation:** s is already good.

Example 2:

Input: s = "aaabbbcc" **Output:** 2 **Explanation:** You can delete two 'b's resulting in the good string "aaabcc". Another way it to delete one 'b' and one 'c' resulting in the good string "aaabbc".

Example 3:

Input: s = "ceabaacb" **Output:** 2 **Explanation:** You can delete both 'c's resulting in the good string "eabaab". Note that we only care about characters that are still in the string at the end (i.e. frequency of 0 is ignored).

Constraints:

* `1 <= s.length <= 105` * `s` contains only lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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