前后缀分离

1563

aababbab

aababbab

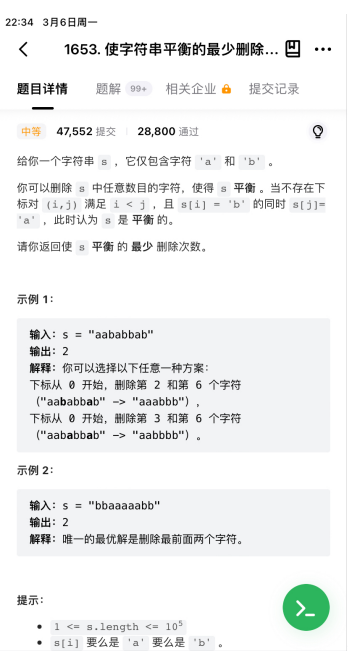
由于平衡字符串的 a 都在 b 的左边，
所以一定存在一条分割线，把 s 分成**前缀**和**后缀**，
这些 a 都在前缀中，b 都在后缀中。

换句话说，
要删除的 b 都在前缀中，
要删除的 a 都在后缀中。

枚举**所有**分割线，计算**删除次数**的最小值，即为答案。

aababbab	4	删除所有后缀中的 a， 统计 s 中 a 的个数。
aababbab	3	右移分割线， 如果是 a 就减小删除次数。
aababbab	2	
aababbab	3	如果是 b 就增加删除次数。
aababbab	2	
aababbab	3	
aababbab	4	
aababbab	3	
aababbab	4	遍历结束，最小值为 2。

```cpp
class Solution {
public:
    int minimumDeletions(string s) {
        int del = 0;
        for (char c : s)
            del += 'b' - c; // 统计 'a' 的个数
        int ans = del;
        for (char c : s) {
            // 'a' -> -1    'b' -> 1
            del += (c - 'a') * 2 - 1;
            ans = min(ans, del);
        }
        return ans;
    }
};
```



22:34 3月6日周一

1653. 使字符串平衡的最少删除...

题目详情　题解 99+　相关企业　提交记录

中等　47,552 提交　28,800 通过

给你一个字符串 s ，它仅包含字符 'a' 和 'b' 。

你可以删除 s 中任意数目的字符，使得 s 平衡 。当不存在下标对 (i,j) 满足 i < j ，且 s[i] = 'b' 的同时 s[j]= 'a' ，此时认为 s 是 平衡 的。

请你返回使 s 平衡 的 最少 删除次数。

示例 1：

输入：s = "aababbab"
输出：2
解释：你可以选择以下任意一种方案：
下标从 0 开始，删除第 2 和第 6 个字符（"aababbab" -> "aaabbb"），
下标从 0 开始，删除第 3 和第 6 个字符（"aababbab" -> "aabbbb"）。

示例 2：

输入：s = "bbaaaaabb"
输出：2
解释：唯一的最优解是删除最前面两个字符。

提示：

- 1 <= s.length <= 10^5
- s[i] 要么是 'a' 要么是 'b' 。

方法二：动态规划（一次遍历）

如果你还不熟悉动态规划（包括空间优化），可以先看看 动态规划入门。

考虑 s 的最后一个字母：

- 如果它是 'b'，则无需删除，问题规模缩小，变成「使 s 的前 $n-1$ 个字母平衡的最少删除次数」。
- 如果它是 'a'：
  - 删除它，则答案为「使 s 的前 $n-1$ 个字母平衡的最少删除次数」加上 1。
  - 保留它，那么前面的所有 'b' 都要删除；

设 $cntB$ 为前 i 个字母中 'b' 的个数。定义 $f[i]$ 表示使 s 的前 i 个字母平衡的最少删除次数：

- 如果第 i 个字母是 'b'，则 $f[i] = f[i-1]$；
- 如果第 i 个字母是 'a'，则 $f[i] = \min(f[i-1]+1, cntB)$。

代码实现时，可以只用一个变量表示 f。

答疑

问：这一次遍历怎么没两次遍历快啊?

答：方法一解释了。通过这两种方法的对比，相信你能感受到随机数据对分支预测的影响。

22:41 3月6日周一 67%
前后缀分解，一张图秒懂！（附动态规划）Python/Java/C++/Go
保留它，那么前面的所有 'b' 都要删除；
设 $cntB$ 为前 i 个字母中 'b' 的个数。定义 $f[i]$ 表示使 s 的前 i 个字母平衡的最少删除次数：
- 如果第 i 个字母是 'b'，则 $f[i] = f[i-1]$；
- 如果第 i 个字母是 'a'，则 $f[i] = \min(f[i-1]+1, cntB)$。
代码实现时，可以只用一个变量表示 f。
答疑
问：这一次遍历怎么没两次遍历快啊?
答：方法一解释了。通过这两种方法的对比，相信你能感受到随机数据对分支预测的影响。

Python3 | Java | C++ | Go

```cpp
class Solution {
public:
    int minimumDeletions(string s) {
        int f = 0, cnt_b = 0;
        for (char c : s)
            if (c == 'b') ++cnt_b; // f 值不变
            else f = min(f + 1, cnt_b);
        return f;
    }
};
```

复杂度分析

- 时间复杂度：$O(n)$，其中 n 为 s 的长度。
- 空间复杂度：$O(1)$，仅用到若干额外变量。

相似题目（前后缀分解，部分题目要结合 DP）
点赞 收藏 46 说点什么吧...