6021. Maximize Number of Subsequences in a String

My Submissions (/contest/biweekly-contest-74/problems/maximize-number-of-subsequences-in-a-string/submissions/)

Back to Contest (/contest/biweekly-contest-74/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

You are given a **0-indexed** string `text` and another **0-indexed** string `pattern` of length `2`, both of which consist of only lowercase English letters.

You can add **either** `pattern[0]` **or** `pattern[1]` anywhere in `text` **exactly once**. Note that the character can be added even at the beginning or at the end of `text`.

Return *the* ***maximum*** *number of times* `pattern` *can occur as a* ***subsequence*** *of the modified* `text`.

A **subsequence** is a string that can be derived from another string by deleting some or no characters without changing the order of the remaining characters.

Example 1:

```
Input: text = "abdcdbc", pattern = "ac"
Output: 4
Explanation:
If we add pattern[0] = 'a' in between text[1] and text[2], we get "abadcdbc". Now, the number of times "ac" occurs as a subs
Some other strings which have 4 subsequences "ac" after adding a character to text are "aabdcdbc" and "abdacdbc".
However, strings such as "abdcadbc", "abdccdbc", and "abdcdbcc", although obtainable, have only 3 subsequences "ac" and are
It can be shown that it is not possible to get more than 4 subsequences "ac" by adding only one character.
```

Example 2:

```
Input: text = "aabb", pattern = "ab"
Output: 6
Explanation:
Some of the strings which can be obtained from text and have 6 subsequences "ab" are "aaabb", "aaabb", and "aabbb".
```

Constraints:

- `1 <= text.length <= 10`^{5}
- `pattern.length == 2`
- `text` and `pattern` consist only of lowercase English letters.

JavaScript

```
const maximumSubsequenceCount = (s, p) => {
    let s0 = p[0] + s, s1 = s + p[1];
    let res1 = subSelection(s0, p), res2 = subSelection(s1, p);
    // pr(s0, res1, s1, res2);
    return Math.max(res1, res2);
};

function Bisect() {
    return { insort_right, insort_left, bisect_left, bisect_right }
    function insort_right(a, x, lo = 0, hi = null) {
        lo = bisect_right(a, x, lo, hi);
        a.splice(lo, 0, x);
    }
    function bisect_right(a, x, lo = 0, hi = null) { // > upper_bound
        if (lo < 0) throw new Error('lo must be non-negative');
        if (hi == null) hi = a.length;
        while (lo < hi) {
            let mid = parseInt((lo + hi) / 2);
            a[mid] > x ? hi = mid : lo = mid + 1;
        }
        return lo;
    }
    function insort_left(a, x, lo = 0, hi = null) {
        lo = bisect_left(a, x, lo, hi);
        a.splice(lo, 0, x);
    }
    function bisect_left(a, x, lo = 0, hi = null) { // >= lower_bound
        if (lo < 0) throw new Error('lo must be non-negative');
        if (hi == null) hi = a.length;
        while (lo < hi) {
            let mid = parseInt((lo + hi) / 2);
            a[mid] < x ? lo = mid + 1 : hi = mid;
        }
        return lo;
    }
}

const counter_value_in_indexA_in = (a_or_s) => { let m = new Map(); let n = a_or_s.length; for (let i = 0; i < n;
i++) { if (!m.has(a_or_s[i])) m.set(a_or_s[i], []); m.get(a_or_s[i]).push(i); } return m; };

const subSelection = (s, p) => {
    let m = counter_value_in_indexA_in(s), a = m.get(p[0]) || [], b = m.get(p[1]) || [], bi = new Bisect(), res = 0;
    // pr(s, m, a, b)
    for (let ia of a) {
       let ib = bi.bisect_right(b, ia);
       res += b.length - ib;
    }
    return res;
};
```

Custom Testcase   Use Example Testcases

Run   Submit

Submission Result: Accepted (/submissions/detail/663056251/)   More Details (/submissions/detail/663056251/)

Share your acceptance!