6246. Append Characters to String to Make Subsequence

My Submissions (/contest/weekly-contest-321/problems/append-characters-to-string-to-make-subsequence/submissions/)

Back to Contest (/contest/weekly-contest-321/)

You are given two strings `s` and `t` consisting of only lowercase English letters.

Return *the minimum number of characters that need to be appended to the end of* `s` *so that* `t` *becomes a* ***subsequence*** *of* `s`.

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.

User Accepted:	765
User Tried:	805
Total Accepted:	766
Total Submissions:	823
Difficulty:	Medium

Example 1:

```
Input: s = "coaching", t = "coding"
Output: 4
Explanation: Append the characters "ding" to the end of s so that s = "coachingding".
Now, t is a subsequence of s ("coachingding").
It can be shown that appending any 3 characters to the end of s will never make t a subsequence.
```

Example 2:

```
Input: s = "abcde", t = "a"
Output: 0
Explanation: t is already a subsequence of s ("abcde").
```

Example 3:

```
Input: s = "z", t = "abcde"
Output: 5
Explanation: Append the characters "abcde" to the end of s so that s = "zabcde".
Now, t is a subsequence of s ("zabcde").
It can be shown that appending any 4 characters to the end of s will never make t a subsequence.
```

Constraints:

- $1 <= s.length, t.length <= 10^5$
- `s` and `t` consist only of lowercase English letters.

JavaScript

```
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 appendCharacters = (s, t) => {
    let ms = counter_value_in_indexA_in(s), mt = counter_value_in_indexA_in(t), pre = -1, bi = new Bisect();
    for (let i = 0; i < t.length; i++) {
        let a = ms.get(t[i]) || [], idx = bi.bisect_right(a, pre);
        if (idx == a.length) {
            return t.length - i;
        } else {
            pre = a[idx];
        }
    }
    return 0;
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/850431674/) | More Details > (/submissions/detail/850431674/)

Share your acceptance!

1