(/) Explore(/explore/) Problems(/problemset/all/) Interview Contest

☐ (https://leetcede.com/subscribe/? Discuss(/discuss/) Storeref=nb_npl&source=nav-tag)

(/subscribe?



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

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 ("<u>a</u>bcde").
```

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 \leq s.length, t.length \leq 10⁵
- s and t consist only of lowercase English letters.

```
JavaScript
                                                                                                                             \mathfrak{C}
1 ▼ function Bisect() {
 2
        return { insort_right, insort_left, bisect_left, bisect_right }
З ч
        function insort_right(a, x, lo = 0, hi = null) {
 4
            lo = bisect_right(a, x, lo, hi);
 5
            a.splice(lo, 0, x);
 6
 7 ,
        function bisect_right(a, x, lo = 0, hi = null) { // > upper_bound
            if (lo < 0) throw new Error('lo must be non-negative');
 8
            if (hi == null) hi = a.length;
9
10 ▼
            while (lo < hi) {
                 let mid = parseInt((lo + hi) / 2);
11
12
                 a[mid] > x ? hi = mid : lo = mid + 1;
13
14
            return lo;
15
        function insort_left(a, x, lo = 0, hi = null) {
16
17
            lo = bisect_left(a, x, lo, hi);
18
            a.splice(lo, 0, x);
19
20 •
        function bisect_left(a, x, lo = 0, hi = null) { // >= lower\_bound
            if (lo < 0) throw new Error('lo must be non-negative');
21
22
            if (hi == null) hi = a.length;
23 ،
            while (lo < hi) {
```

```
let mid = parseInt((lo + hi) / 2);
 24
 25
                                                                a[mid] < x ? lo = mid + 1 : hi = mid;
26
27
                                                 return lo;
 28
                               }
29
                }
 30
                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++)\ \{\ let\ m\ =\ new\ Map();\ let\ n\ =\ a\_or\_s.length;\ for\ (let\ i\ =\ 0;\ i\ <\ n;\ i++)\ \{\ let\ m\ =\ new\ Map();\ let\ n\ =\ a\_or\_s.length;\ for\ (let\ i\ =\ 0;\ i\ <\ n;\ i++)\ \{\ let\ m\ =\ new\ Map();\ let\ n\ =\ a\_or\_s.length;\ for\ (let\ i\ =\ 0;\ i\ <\ n;\ i++)\ \{\ let\ m\ =\ new\ Map();\ let\ n\ =\ a\_or\_s.length;\ for\ (let\ i\ =\ 0;\ i\ <\ n;\ i++)\ \{\ let\ m\ =\ new\ Map();\ let\ n\ =\ new\ Map();
 31
                 if (!m.has(a_or_s[i])) m.set(a_or_s[i], []); m.get(a_or_s[i]).push(i); } return m; };
32
                const appendCharacters = (s, t) \Rightarrow {
33 •
                                let ms = counter_value_in_indexA_in(s), mt = counter_value_in_indexA_in(t), pre = -1, bi = new Bisect();
34
                                for (let i = 0; i < t.length; i++) {
35 ▼
                                                let a = ms.get(t[i]) || [], idx = bi.bisect_right(a, pre);
36
37 ▼
                                                 if (idx == a.length) {
38
                                                                return t.length - i;
39 •
                                                } else {
40
                                                                pre = a[idx];
41
                                                }
42
43
                                return 0;
 44
                };
```

☐ Custom Testcase

Use Example Testcases

Run

Submission Result: Accepted (/submissions/detail/850431674/) ?

More Details > (/submissions/detail/850431674/)

Share your acceptance!

Copyright @ 2022 LeetCode

⊲1

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

United States (/region)