

1055. Shortest Way to Form String Premium

Attempted

Medium Topics Companies Hint

A **subsequence** of a string is a new string that is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (i.e., "ace" is a subsequence of "abcde" while "aec" is not).

Given two strings `source` and `target`, return the *minimum number of subsequences* of `source` such that their concatenation equals `target`. If the task is impossible, return `-1`.

Example 1:

Input: `source = "abc", target = "abcbc"`

Output: 2

Explanation: The target "abcbc" can be formed by "abc" and "bc", which are subsequences of source "abc".

Example 2:

Input: `source = "abc", target = "acdbc"`

Output: -1

Explanation: The target string cannot be constructed from the subsequences of source string due to the character "d" in target string.

Python3 Auto

```
1 class Solution:
2     def shortestWay(self, source: str, target: str) -> int:
3         if set(source) < set(target):
4             return -1
5
6         m = len(source)
7         count = 0
8         sp = 0 # source pointer
9
10        for c in target:
11            if sp == 0:
12                count += 1
13
14            # while didn't find the target char, the sp move to next
15            while source[sp] != c:
16                sp = (sp + 1) % m
17                if sp == 0:
18                    count += 1
19
20            # if the while loop breaks, which means the char is found, and move sp to find the
21            next char in target
22            sp = (sp + 1) % m
23
24        return count
25
```

Ln 20, Col 115

Save

Run

Submit



Time Limit Exceeded.

Accepted 48 / 48 testcases passed

AndrewC275 submitted at Mar 31, 2025 14:49

Editorial

Solution

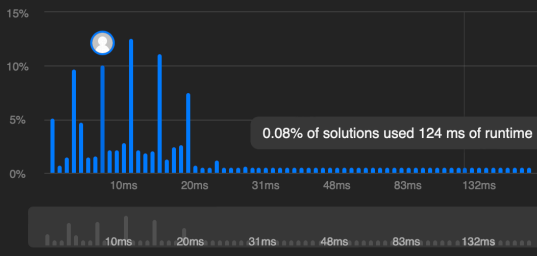
Runtime

7 ms | Beats 74.92%

Analyze Complexity

Memory

18.01 MB | Beats 15.21%



```
1 class Solution:
2     def shortestWay(self, source: str, target: str) -> int:
3         ss = set(source)
4         for c in target:
5             if c not in ss:
6                 return -1
7
8         m = len(source)
9         count = 0
10        sp = 0 # source pointer
11
12        for c in target:
13            if sp == 0:
14                count += 1
15
16            # while didn't find the target char, the sp move to next
17            while source[sp] != c:
18                sp = (sp + 1) % m
19                if sp == 0:
20                    count += 1
21
22            # if the while loop breaks, which means the char is found, and move sp to find the
23            next char in target
24            sp = (sp + 1) % m
25
26        return count
27
```

change from set comparison
to this for loop.

- Creating a set : $O(n)$

$n = \text{len}(\text{source})$

- Set Comparison : $O(n_1)$

where $n_1 < n_2$

↳ ① using Set Comparison :

$$O(\text{source}) + O(\text{target}) + O(\text{source})$$

② for loop :

$$O(\text{source}) + O(\text{target})$$