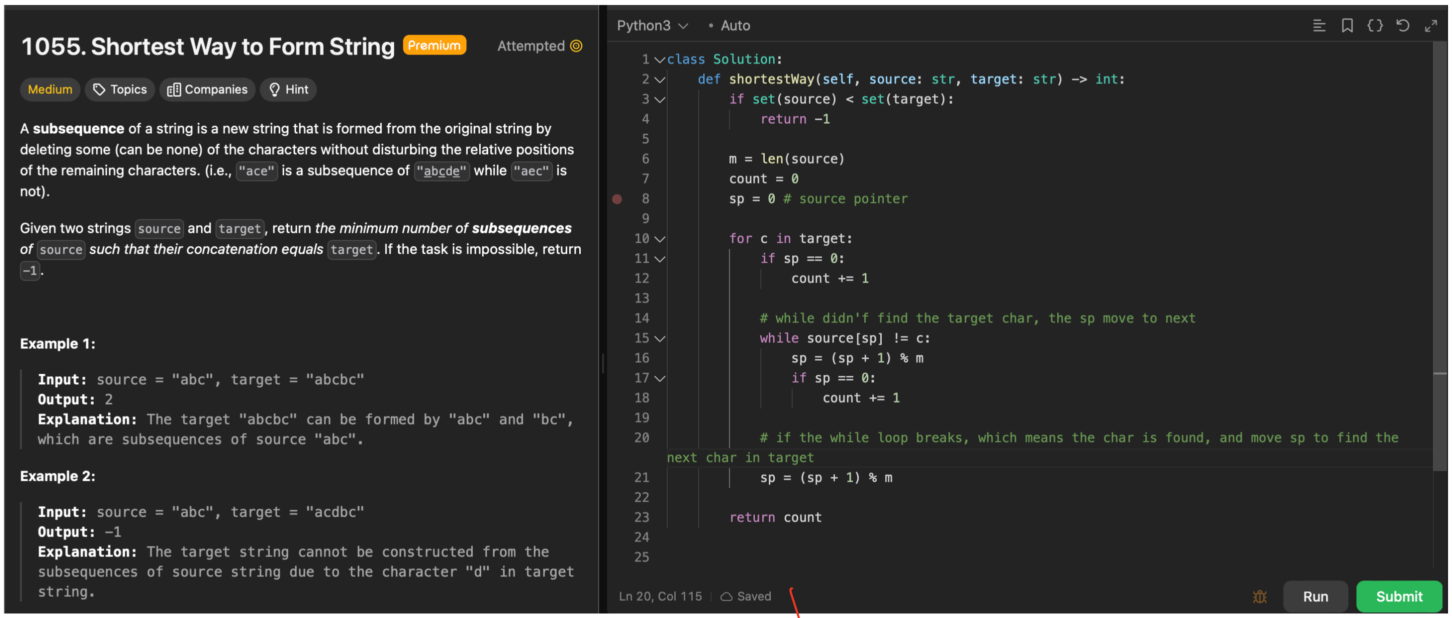


1055. Shortest Way to Form String

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:

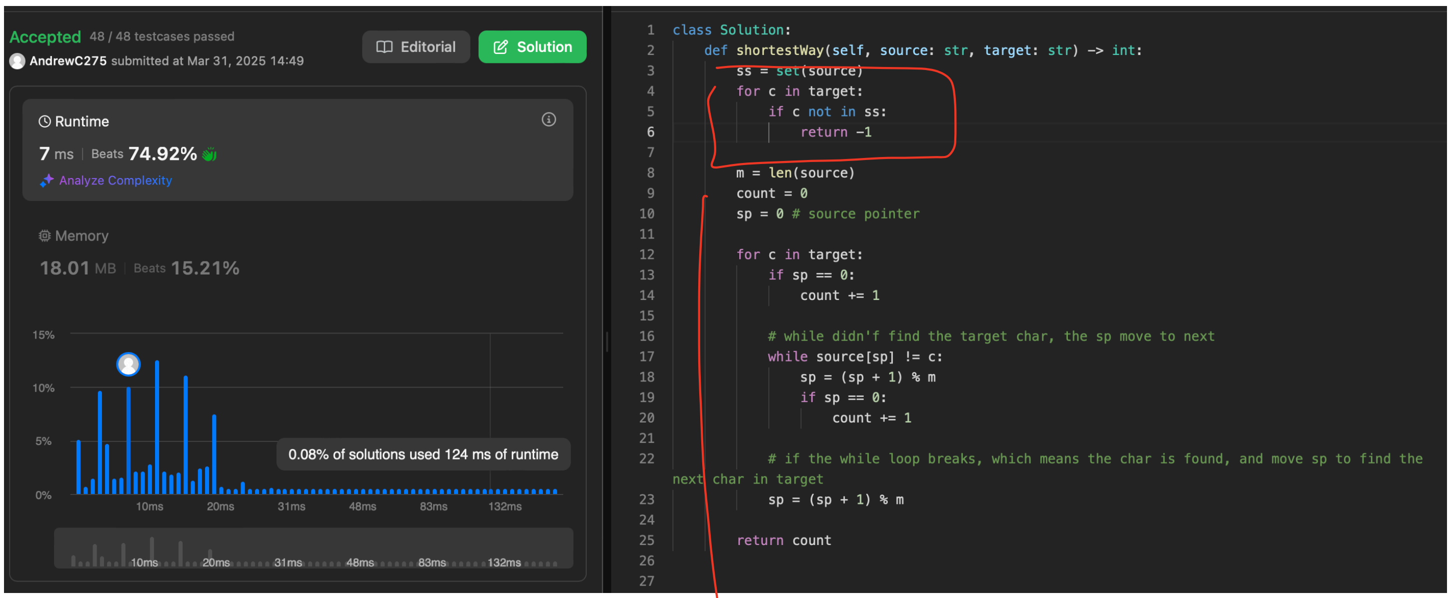
```
Input: source = "abc", target = "abcbc"
Output: 2
Explanation: The target "abcbc" can be formed by "abc" and "bc",
which are subsequences of source "abc".
```

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.
```

```python
class Solution:
    def shortestWay(self, source: str, target: str) -> int:
        if set(source) < set(target):
            return -1

        m = len(source)
        count = 0
        sp = 0 # source pointer

        for c in target:
            if sp == 0:
                count += 1

            # while didn'f find the target char, the sp move to next
            while source[sp] != c:
                sp = (sp + 1) % m
                if sp == 0:
                    count += 1

            # if the while loop breaks, which means the char is found, and move sp to find the next char in target
            sp = (sp + 1) % m

        return count
```

Time Limit Exceded.



Accepted 48 / 48 testcases passed

AndrewC275 submitted at Mar 31, 2025 14:49

```python
class Solution:
    def shortestWay(self, source: str, target: str) -> int:
        ss = set(source)
        for c in target:
            if c not in ss:
                return -1

        m = len(source)
        count = 0
        sp = 0 # source pointer

        for c in target:
            if sp == 0:
                count += 1

            # while didn'f find the target char, the sp move to next
            while source[sp] != c:
                sp = (sp + 1) % m
                if sp == 0:
                    count += 1

            # if the while loop breaks, which means the char is found, and move sp to find the next char in target
            sp = (sp + 1) % m

        return count
```

change from set comparison to this for loop.

- Creating a set : $O(n)$ $n = \text{len(source)}$
- 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 :

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