remove_duplicated_from_sorted_array

May 8, 2025

1 Find the Index of the First Occurrence in a String

1.1 Problem Definition

Given two strings needle and haystack, return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

1.1.1 Example Cases

Example 1:

Input: haystack = “sadbutsad”, needle = “sad”
Output: 0
Explanation: “sad” occurs at index 0 and 6.
The first occurrence is at index 0, so we return 0.

Example 2:

Input: haystack = “leetcode”, needle = “leeto”
Output: -1
Explanation: “leeto” did not occur in “leetcode”, so we return -1.

1.1.2 Constraints

Constraints:

1 <= haystack.length, needle.length <= 10^4
haystack and needle consist of only lowercase English characters.

1.2 Example test cases

[2]:
```python
def test_cases():
    assert strStr(haystack="sadbutsad", needle="sad") == 0
    assert strStr(haystack="leetcode", needle="leeto") == -1
    print("All test cases passed!")
```

1.3 Solutions

$O(nm)$, n = len(haystack), m = len(needle)

[47]:
```
def strStr(haystack, needle):
    if len(haystack) < len(needle):
        return -1
    haystack_pointer = 0
    needle_pointer = 0
    n_correct = 0
    while haystack_pointer <= len(haystack) - 1:
        if len(haystack) - haystack_pointer < len(needle):
            return -1
        if haystack[haystack_pointer] == needle[0]:
            needle_pointer += 1
            n_correct += 1
            while needle_pointer <= len(needle) - 1:
                if haystack[haystack_pointer + needle_pointer] ==␣
 ↪needle[needle_pointer]:
                    n_correct += 1
                needle_pointer += 1
            if n_correct == len(needle):
                return haystack_pointer
            n_correct = 0
            needle_pointer = 0
        haystack_pointer += 1
    return -1
```

[48]:
```
test_cases()
```

```
All test cases passed!
```

The code is super messy and complex.

1.3.1 Improvements

$O(nm)$

We can use the fact that strings are sequences and slice part of it to check if it's equal to the needle or not. Also, the for loop can be stopped when the pointer is greater than len(haystack) - len(needle). This will handle the case of len(needle) > len(haystack) as well

[62]:
```
def strStr(haystack, needle):
    needle_len = len(needle)
    pointer = 0
    while pointer <= (len(haystack) - needle_len):
        if haystack[pointer: pointer+needle_len] == needle:
            return pointer
        pointer+=1
    return -1
```

[64]:
```
test_cases()
```

```
All test cases passed!
```