# 344. Reverse String



Write a function that reverses a string. The input string is given as an array of characters s.

You must do this by modifying the input array in-place with 0(1) extra memory.

#### Example 1:

```
Input: s = ["h","e","l","l","o"]
Output: ["o","l","l","e","h"]
```

# Example 2:

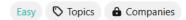
```
Input: s = ["H","a","n","n","a","h"]
Output: ["h","a","n","n","a","H"]
```

#### Constraints:

- 1 <= s.length <= 10<sup>5</sup>
- s[i] is a printable ascii character.

# class Solution:

```
def reverseString(self, s: List[str]) -> None:
    left, right = 0, len(s) - 1
    while left < right:
        s[left], s[right] = s[right], s[left]
        left += 1
        right -= 1</pre>
```



A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.

Given a string s, return true if it is a palindrome, or false otherwise.

#### Example 1:

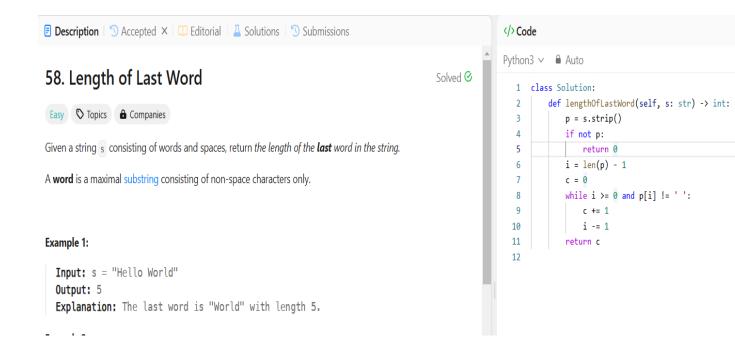
```
Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a palindrome.
```

#### Example 2:

```
Input: s = "race a car"
Output: false
Explanation: "raceacar" is not a palindrome.
```

#### class Solution:

```
def isPalindrome(self, s: str) -> bool:
    n=len(s)
    1=0
    r=n-1
    while l<r:
        if not s[l].isalnum():
            1+=1
            continue
        if not s[r].isalnum():
            r-=1
            continue
        if s[l].lower()!=s[r].lower():
           return False
        1+=1
        r-=1
    return True
```



# 28. Find the Index of the First Occurrence in a String

Solved **⊘** 



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.

### 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.
```

# class Solution:

```
def strStr(self, haystack: str, needle: str) -> int:
    if not needle:
        return 0
    len_hay, len_ndl = len(haystack), len(needle)
    for i in range(len_hay - len_ndl + 1):
        if haystack[i:i + len_ndl] == needle:
            return i
    return -1
```

# 3. Longest Substring Without Repeating Characters

Solved **⊘** 

```
Medium ♥ Topics ♠ Companies ♥ Hint
```

Given a string s, find the length of the **longest substring** without repeating characters.

# Example 1:

```
Input: s = "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.
```

# Example 2:

```
Input: s = "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.
```

#### Example 3:

```
Input: s = "pwwkew"
Output: 3
Explanation: The answer is "wke", with the length of 3.
Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.
```

# class Solution: