344. Reverse String

Solved

Easy | Topics | Companies | Hint

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 `O(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 <= 10^5`
- `s[i]` is a printable ascii character.

```python
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
```

125. Valid Palindrome

Solved

Easy | Topics | Companies

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.
```

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

        return True
```

Description | Accepted × | Editorial | Solutions | Submissions

58. Length of Last Word

Solved

Easy | Topics | Companies

Given a string `s` consisting of words and spaces, return *the length of the **last** word in the string.*

A **word** is a maximal substring consisting of non-space characters only.

Example 1:

```
Input: s = "Hello World"
Output: 5
Explanation: The last word is "World" with length 5.
```

Code

Python3 | Auto

```python
class Solution:
    def lengthOfLastWord(self, s: str) -> int:
        p = s.strip()
        if not p:
            return 0
        i = len(p) - 1
        c = 0
        while i >= 0 and p[i] != ' ':
            c += 1
            i -= 1
        return c
```

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

Solved

Easy | Topics | Companies

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.
```

```python
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.
```

```python
class Solution:
    def lengthOfLongestSubstring(self, s: str) -> int:
        l=0
        lng=0
        s1=set()
        n=len(s)
        for r in range(n):
            while s[r] in s1:
                s1.remove(s[l])
                l+=1
            w=(r-l)+1
            lng=max(lng,w)
            s1.add(s[r])
        return lng
```