344. Reverse String:

I/P: s = ["h", "e", "l", "l", "o"]

O/P: ["o", "l", "l", "e", "h"].

we need to:

* Reverse the array in-place ↔ modify the original array
* use O(1) extra memory → can't create a new array.
* Return nothing (just modify s).

Brute force:

1. copy the array into a new array
2. Reverse the new array
3. copy back to original

✗ problem: use extra memory, not allowed.

Efficient approach (two-pointer)

use two pointers:

* left → start of array
* right → end of array

* swap s[left] and s[right]
* Move left rightward and right leftward
* stop when left >= right.

```
Class Solution {
    Public void reverseString (char[] s){
        int left = 0;
        int right = s.length - 1;

        while (left< right) {
            char temp = s[left];
            s[left] = s[right];
```

```
            s[right] = temp;
            left++;
            right--;
        }
    }
}
```

Dry Run:

s = ["h","e","l","l","o"]

left = 0 right = 4

step 1: swap s[0] and s[4] → ['o','e','l','l','h'],

left = 1, right = 3

step 2: swap s[1] and s[3] → ['o','l','l','e','h'],

left = 2, right = 2

stop (left >= right)

Result = ['o','l','l','e','h'].

Time → O(n) space → O(1)