M T W T F S S
Page No.:
Date
YOUVA

-> 26 June 2024]

- Converting a char array to String Java

String ans = new String(arr);

-> Reverse words in a String]

"The_sky_is_blue"
... 1.5 ! trim() ✓ !
↓

"blue_is_sky_the"

Worked

U
N
D
E ① ~~Two~~ One pointer approach (~~start~~, end)
R ② Space differentiates words
③ Keep track of ~~two~~ variable.
~~start length~~ and ~~endlength~~ end length
4 ④ Swap inplace ✗ New array.
O ⑤ Append ogArray [end] -> [end + end length
M ⑥ Add space after each words
I ⑦ trim, as extra space one ~~at~~
N ~~end~~. If end pointer at
Donot -> 0th idx, edge case
add space
after
(30 min approx)
⑧ Convert to String, return.

Multiple spaces -> space counter

[Problem: pointer -- even if space ≠ 0)

Continue >> Approach (2).

" the sky is blue "

↓

" blue is sky the "

① Trim array.
② ~~Reverse~~ Split
③ Reverse add each word ~~and~~ trim word again if multiple spaces need left in trim.
④ ① after each word one space.
⑤ Convert array to String and return.

-> 238. Product of Array Except Self]

Complexity should be O(n).
Cannot use [division operation]

Brute force -> $O(n^2)$

Num Range pretty small: -30 to 30

(since array)

[1, 2, 3, 4]

↓

[24, 12, 8, 6]

Expected space Complexity O(1)

-> Division Operator >> Use Multiply add, return array with elems -> multiplied value / self.

-> Solutions are easy to code but difficult to optimize.

Anybody can write code in complexity $O(n^2)$ or even $O(n)$ in some cases.

[left]

-> Sort an Array containing only 0's and 1's as elems.

① Use O(n) to count 0's and change the original array by appendy 0's the 1's in left space.

② Two pointer Approach if start == 1, stop else ++
if end == 0 else --

Swap start and end.

Repeat till start <= end.

still (O(n)) but better ~ much ~~

- Sort by Parity (even / odd) regardless of relative order of elems
(same as previous)

- For a non decending array, return the array of squares of each number sorted in non-decending array
(Array might contain negative nums)

[-10, -3, -2, 1, 4, 5]
100 9 4 1 16 25
[1, 4, 9, 16, 25, 100]

→ Compare the absolute values of elements at start and end pointer. whichever is the lowest / Highest append in array while

Reverse the array, square it and return
——————> new ->
If don't wish to reverse, then save the ~~et~~ index where array starts positive elems

Initialize both pointers here, go left and right ~~and~~ keep comparing for min and append

While performing square sized so no extra O(n)

Comparison

-> for ① iter. Non decending array achieved ① Squares	-> for ① iter. Only squares and pivot found.
-> for ② length/2 Reversed array. ↑ High Readable	-> for ② iter len/2 -> sorting. ↑ Complexity pretty much same.

Best (How)?

Directly add at (length - 1) and decrease counter ~