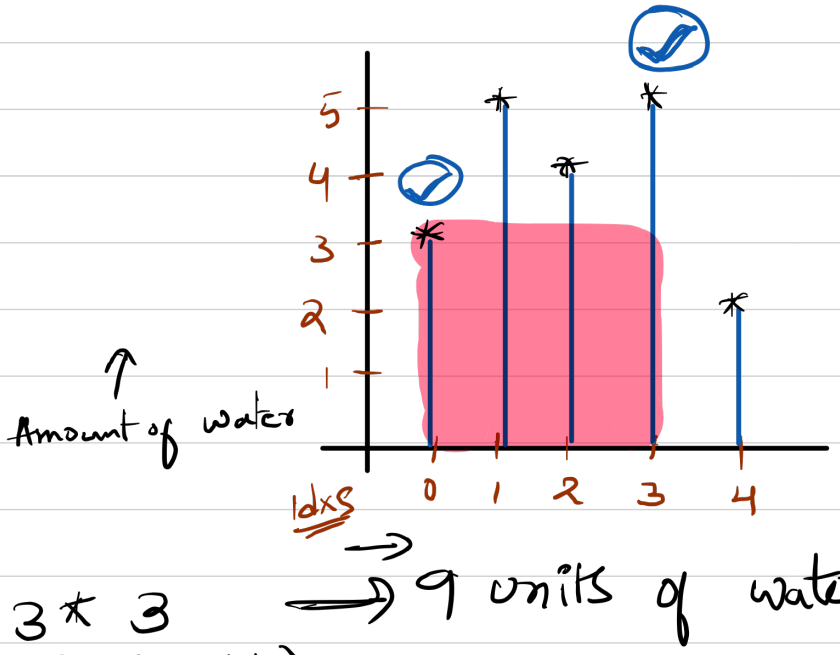
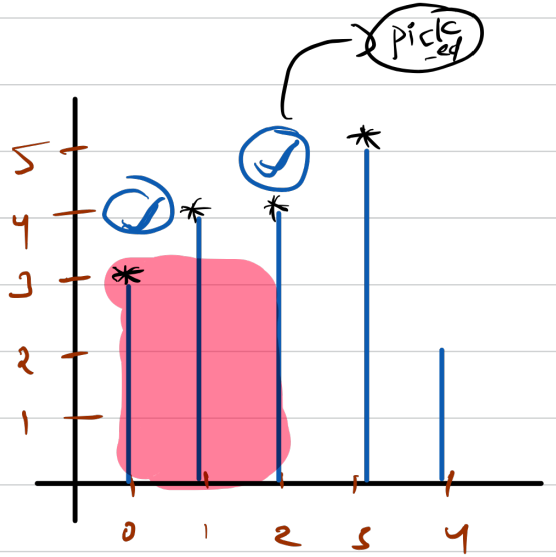
Container with most Water

↳ Uses 2 pointer technique

Given an arr[N], where A[i] represents height of each wall. Pick any 2 walls such that max water is accumulated between them.



3 * 3 → 9 units of water
(breadth) (height)



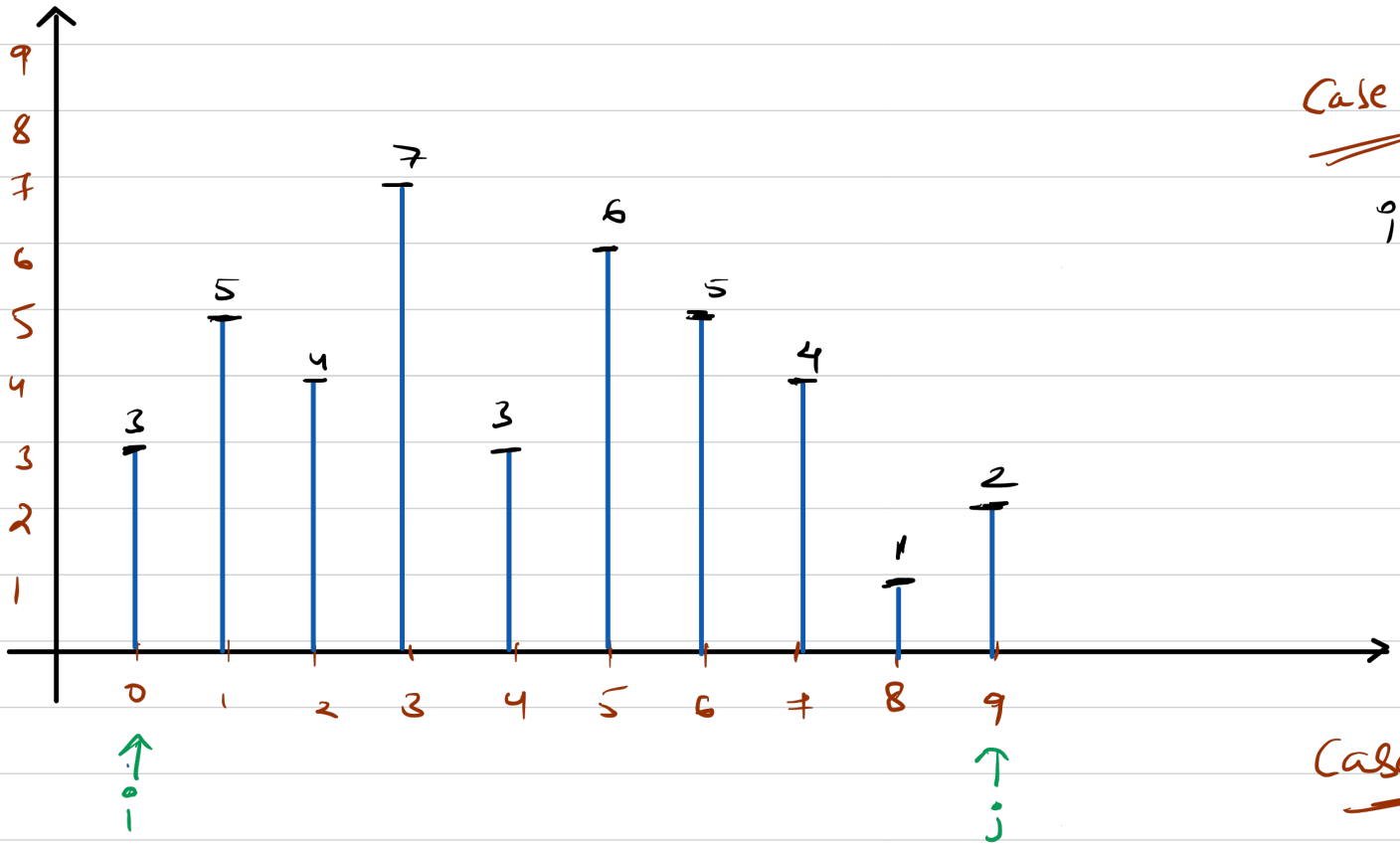
→ 2 * 3 = 6 units

Amount of water(i, j) = (j - i) * min(arr[i], arr[j])

(j - i): Breadth

min(arr[i], arr[j]): height

	0	1	2	3	4	5	6	7	8	9
a[] = {	3	5	4	7	3	6	5	4	1	2 }



Case - I

if arr[i] > arr[j]

j is not useful in making max area

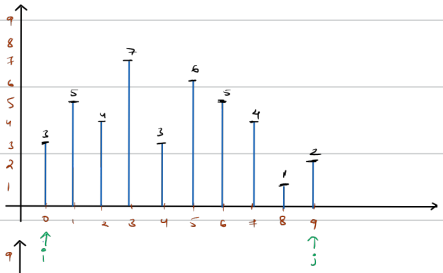
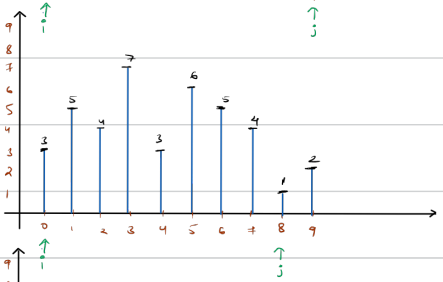
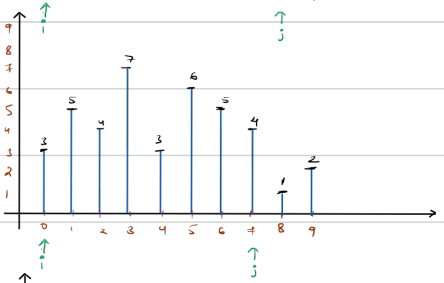
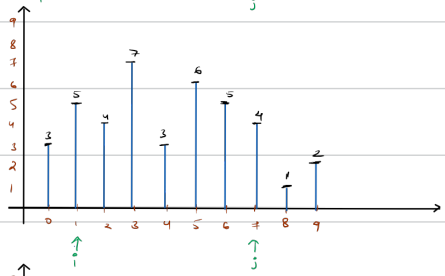
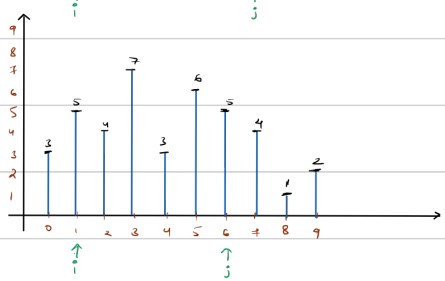
So → j--

Case - II

if arr[i] < arr[j]

i is not useful in making max area

So → i++

	a[i]	a[j]	min(a[i],a[j]) height	j-i width	amount	Max ans	operation
	3	2	2	9	18	~~0~~ → 18	a[i] ≥ a[j] ⇓ (j--)
	3	1	1	8	8	18	a[i] ≥ a[j] ⇓ (j--)
	3	4	3	7	21	~~18~~ → 21	a[i] < a[j] ⇓ i++
	5	4	4	6	24	~~21~~ → 24	a[i] ≥ a[j] ⇓ j--
	5	5	5	5	25	~~24~~ → 25	a[i] ≥ a[j] ⇓ j--

Likewise, update (i, j) and store max potential area of ans

* After i,j crosses, which area is lastly updated, that will be possible max area of water that can contain

Time Complexity :

$O(n)$

Space Complexity :

$O(1)$