TWO POINTERS *(OUTSIDE IN)*

No.	Problem Statement	Solution	Time complexity	Space complexity
1	**Valid Pelindrome**			
	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.	- **Left and Right pointers** : `l=0 and r=n-1` - `tolower()`: to convert it to small case and -isalnum(): to check if an alphabet or a number - Skip over non alphanumberic characters Else compare `s[l] == s[r]`	O(N)	O(1)
2	**Two Sum II (Input Array Is Sorted)**			

	Given an array of integers nums and an integer target, return indices (index1 < index2) of the two numbers such that they add up to target. (You may not use the same element twice)	- **Left and Right pointers** : `l=0 and r=n-1` --> Only works if the array is sorted
		- `while(l<r)`
		`curr_sum = numbers[l] + numbers[r];`
		`if(curr_sum == target) return {l+1, r+1};`
		`if(curr_sum > target) r--;` // To reduce the sum
		`else l++;`
			O(N)	O(1)

3	**3Sum**			
	Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0.	- **Sort** the array - We need to Find `{a,b,c}` such that `a+b+c=0` - Traverse 'nums' array `i: 0 --> n-1` - `a = nums[i] & target = -1 * nums[i]` - now, treat the problem as finding a pair {b,c} such that `b + c = target`	O(N^2)	O(1)
4	**Container With Most Water**			
	Given an integer array 'height' of length n, where height[i] represents the height of verticle line at i. Find two lines that together with the x-axis form a container, such that the container contains the most water.	- **Left and Right pointers:** `l=0 and r=n-1` - `while(l<r)` - `temp = min(height[l], height[r]) * (r-l)` - `if(height[l]<height[r) l++` - // Greedily move the pointer that points to the shorter line, which increases the possibility of finding a greater area in the next iteration. - `else r--`	O(N)	O(1)
5	**Trapping Rain Water**			

	Given an integer array 'height' of length n, where height[i] represents the height at 'i'. The width of each bar is 1, compute how much water it can trap after raining.	- **Left and Right pointer** s: `l=0 and r=n-1`
		- `max_l = height[l]` // Maximum height encountered from the left
		- `max_r = height[r]` // Maximum height encountered from the right
		`while(l<r)`
		// max_l < max_r --> Water trapped depends on max_l
		`if(max_l<max_r)`
		`ans += max_l - height[l]`
		`l++` // Imp that we do 'l++' first and then update max_l
		`max_l = max(max_l, height[l])`
		`else`
		`ans += max_r - height[r]`
		`r--`
		`max_r = max(max_r, height[r])`
			O(N)	O(1)