Topic: Arrays And Strings

- Kadane's Algo (Max Subarray Sum)
- Sliding Window
- Two Pointers
- Prefix Sum
- Merge Intervals
- String search (KMP or Z algo)

- Questions: Tick After Completion
  Leetcode: 55, 918, 1191
- Count number of subarray with max sum
- Find longest subarray with max sum.

Core idea of Kadane's Algo

Let's say arr = [4, -1, 2, 1, -5, 4]

- We ask two questions as we walk:
  → Should I extend the current subarray (current_sum + arr[i]) ?
  → Or start a new subarray from there (arr[i]) ?
- We pick whichever gives more sum:
  current_sum = max(arr[i], current_sum + arr[i])
- Then update our answer:
  max_sum = max(max_sum, current_sum)

- Code (with Subarray Index Tracking)

```
def kadane(arr):
    max_sum = float('-inf')
    current_sum = 0
    start = end = s = 0

    for i in range(len(arr)):
        current_sum += arr[i]

        if current_sum > max_sum:
            max_sum = current_sum
            start = s
            end = i
        if current_sum < 0
            current_sum = 0
            s = i+1

    return max_sum, arr[start: end+1]
```

Sliding window

↳ fixed-size window
- Size of window is constant (e.g find max sum of subarray of size k)

↳ Variable-size window
- window grows or shrunk dynamically based on condition (e.g longest substring with k distinct characters)

- Max sum of subarray of size k

```
def max_sum_subarray-k( nums, k):
    max_sum = current_sum = sum( nums[:k] )
    for i in range (k, len (nums)):
        current_sum += nums[i] - nums[i-k]   # slide window
        max_sum = max( max_sum ,current_sum)
    return max_sum
```

k=3
ex: nums: [1,2,3,4,5]
window: [1,2,3] → sum=6
[2,3,4] sum=9
max-9

- Variable-size window · (Longest substring with at most 2 distinct characters)

```
def longest_substring-k_distincts ( s, k):
    left = 0
    max_len = 0
    char_map = defaultdict(int)   # store the count of character in current window
    for right in range( len(s)): # Expand window
        char_map[s[right] += 1
        while len( char_map ) > k :  # shrunk window (if needed)
            char-map[s[left] -= 1   } decrease the count of characters.
            if char_map[s[left] == 0   } if count = 0 remove it from window
                del char_map[s[left]]
            left += 1   } move left, right to shrunk
        max_len = max( max_len , right - left + 1)
    return max_len
```

Questions:
→ max sum of subarray of size k
→ Longest substring without repeating Le=3
→ leetcode 76
→ leetcode 239

Two Pointers

Types:
→ Opposite Direction : Sorted array : find two numbers with sum = Target
→ Same Direction (fast/ slow) : detecting cycles in a linked list or array
→ Partitioning/ Swapping : Dutch National flag problem, sorting colors

1: Two Sum in a Sorted Array

```
def two_sum_sorted(arr, target):
    left = 0, right = len(arr) - 1

    while left < right:
        curr_sum = arr[left] + arr[right]

        if curr_sum == target:
            return [arr[left], arr[right]]

        elif curr_sum < target:
            left += 1

        else:
            right -= 1

    return []
```

2: Same Direction : Detect cycle in a linked list
- slow : move 1 step
- fast : move 2 step

If there's a cycle → they will meet inside the loop

3. Partitioning/ Three pointers : low, mid, high
- low → boundary for 0s
- mid → current index
- high → boundary for 2s

Rules:
- arr[mid] == 0 →
  swap with arr[low], then increment both low and mid
- arr[mid] == 1 →
  It is the right place → just mid += 1
- arr[mid] == 2
  swap with arr[high], and decrement high

Questions: Tick after completion
- Two sum
- Three Sum
- Container with Most water
- Move Zeros
- Remove duplicate from Sorted Array
- Reverse words in String
- Detect cycle in LL