🖥️Lecture 01 — Two-Pointer Pattern

🕑Objective

Aaj hum seekhenge: **Two Pointer Pattern kya hota hai**, **kab use hota hai**, aur **kyun important hai**. Yeh DSA ke sabse powerful patterns me se ek hai jo aapko **$O(n^2)$** se **O(n)** tak optimize karne me madad karta hai ⚡

💡Intuition (Simple Story)

Socho ek **line** hai — ek taraf ek **ladka**, doosri taraf ek **ladki**.

- Dono ek-ek step aage badhte hain, kabhi dono ek saath bhi badhte hain.
- Jab dono **mil jaate hain**, hum kehte hain — condition meet ho gayi 🦜
- Yehi logic hum **array**, **string** ya **linked list** par apply karte hain.

📘Definition (Formal)

Two Pointer Pattern ek **technique** hai jisme **do variables/pointers** ek hi linear data structure ko **alag-alag positions se simultaneously traverse** karte hain.

Common Data Structures:

- Array
- String
- Linked List

⚙️Characteristics

1. **Two variables/pointers:** i aur j
2. **Linear Data Structure:** Array, String, ya Linked List
3. **Different Start Points:** Dono alag jagah se move karte hain
4. **Termination Condition:** Depends on problem (e.g. $i \geq j$, pointer null ho gaya, etc.)

🔦Direction Types

- **Towards Each Other:** e.g. Palindrome Check, Pair Sum Problem

- **Away From Each Other:** e.g. Expand From Middle
- **Same Direction (Fast-Slow):** e.g. Find Middle in Linked List, Detect Cycle

---

Fast–Slow Pointer Concept (Important)

Slow pointer (i): ek step move karta hai
Fast pointer (j): do steps move karta hai

- Jab **fast pointer end tak pahuchta hai**, slow pointer **middle element** par hota hai.
  Yehi logic use hota hai in:
- Find Middle of Linked List
- Detect Loop in Linked List (Floyd's Algorithm)

⏳ Why Use Two Pointer Pattern?

- **Efficient:** Time complexity kam karta hai ($O(n^2) \rightarrow O(n)$)
- **Early Stopping:** Jaise hi condition fail ho, loop break kar do
- **In-place:** Extra memory nahi lagti (Space O(1))

Mnemonic: PCS → Partitioning, Comparison, Searching
Agar problem me ye words dikhein to Two Pointer try karo.

🔍 Example 1 — Palindrome Check (String)

Definition: String ko agar left se aur right se padho to same dikhna chahiye.

Example: "level" → Palindrome 🦜

Brute Force Approach

Reverse string banao aur compare karo → Time: O(n), Space: O(n)

Optimized (Two Pointer)

```
i = 0
j = s.length - 1
while i < j:
  if s[i] != s[j]:
    return false
  i += 1
```

```
  j -= 1
return true
```

Time: O(n)
Space: O(1) — In-place, no extra string banayi

💐**Tip:** Reverse string banane se bacho, wahi kaam Two Pointer se kar lo.

🔍Example 2 — Pair Sum in Sorted Array

Problem: Sorted array diya hai aur ek target sum. Find karo 2 elements jinka sum target ke barabar ho.

Approach:

```
left = 0
right = n - 1
while left < right:
  if arr[left] + arr[right] == target:
    return [left, right]
  elif arr[left] + arr[right] < target:
    left += 1
  else:
    right -= 1
```

Time: O(n)
Brute Force: $O(n^2)$

🔍Example 3 — Remove Duplicates from Sorted Array

Idea:

- 1 Pointer (slow) → unique values store karega
- 2nd Pointer (fast) → array traverse karega
- Jab naye element mile, slow++ aur copy karo

Result: In-place unique array without extra space

🔍Example 4 — Merge Two Sorted Arrays

Concept: Two inputs ko merge karte hue dono ko exhaust karna.

```
i = 0, j = 0, k = 0
while i < n1 and j < n2:
  if a[i] <= b[j]:
    res[k] = a[i]
    i += 1
  else:
    res[k] = b[j]
    j += 1
  k += 1
// append remaining elements
```

Use Case: Merge Sort me sorted arrays merge karte waqt.

🧹 Templates Summary

Template A — Single Array (Both Ends)

```
left = 0
right = n - 1
while left < right:
  if arr[left] == arr[right]:
    left += 1
    right -= 1
  else:
    break
```

Used In: Palindrome, Pair Sum

Template B — Two Arrays (Exhaust Both)

```
i = 0; j = 0
while i < n1 and j < n2:
  if a[i] < b[j]:
    i++
  else:
    j++
```

Used In: Merge Sort, Intersection/Union Problems

💡 Extra Tips & Notes

Tips to Remember:

- Early stopping: Palindrome me jaise hi mismatch ho, return false.
- Space optimization: Reverse copy avoid karo — same array me kaam karo.
- Fast-slow pointer trick: Linked List me use karna must hai.
- Sorted array: Milte hi socho — left/right pointer se O(n) me solve kar sakte ho.

📘 When to Think Two Pointer?

Think Two Pointer when:

- Problem me words ho → pair, sum, compare, partition, merge, duplicate, middle, cycle
- Array sorted ho → Try Two Pointer
- Linked list me indexing possible nahi → Fast-Slow pointer approach

---

📋 One-Shot Summary (Revision)

Concept	Explanation
Technique	Two pointers traverse same data structure
Data Structures	Array, String, Linked List
Time Complexity	O(n) (Mostly Linear)
Space Complexity	O(1) (In-Place)
Major Uses	Palindrome, Pair Sum, Duplicate Removal, Merge, Middle/Cycle Detection
Keywords	Partition, Comparison, Searching

💴 Final Note

Two Pointer ek **powerful DSA pattern** hai jo time aur space dono optimize karta hai.
Har ek example likh-likh ke practice karo aur diagram ke saath samjho.

Keep Learning! DSA me mastery tabhi milegi jab patterns samjhoge — aur Two Pointer uska pehla kadam hai.