🧠✨ Lecture 09 — ARRAY PROBLEM SOLVING (C++)

> “Problem solving is not about writing code.
> It’s about choosing the **right idea**.”

Made By Harshal Chauhan

🔁 ① CYCLIC ROTATION BY ONE (RIGHT ROTATION)

❓ Problem

Array ko **right side se 1 step rotate** karna hai.

Input
`[1, 2, 3, 4, 5]`

Output
`[5, 1, 2, 3, 4]`

🧠 First-Thought Principle

Right rotation ka matlab:

- Last element **front** me aayega
- Baaki sab **1 step right shift**

🛠 Logic (Core Thinking)

1. Last element ko save karo (warna lost ho jayega)
2. Right se left shift karo (overwrite se bachne ke liye)
3. Saved element ko index `0` pe daal do

⏱ Complexity

- **Time:** `O(n)`
- **Space:** `O(1)`

🧠 Important Insight

❌ Left-to-right shift karoge → data overwrite
✅ Right-to-left shift → safe movement

📌 **Rule yaad rakho**

- Right rotation → loop `i--`
- Left rotation → loop `i++`

🔁 ② ROTATE ARRAY BY k (CLOCKWISE)

❓ Problem

Array ko **k steps right rotate** karna hai.

❌ APPROACH 1 — Repeated Rotation (Brute Force)

🧠 Idea

1 step rotation ko **k baar repeat**

⏱ Complexity

- **Time:** `O(n × k)`
- **Space:** `O(1)`

❌ **Large k → TLE (Rejected)**

📌 **Interview note:**
"Correct but inefficient"

✅ APPROACH 2 — EXTRA ARRAY + MODULO

🧠 Observation

New index = `(i + k) % n`

🛠 Logic

- Ek helper array lo
- Har element ko uski final position pe daalo
- Wapas original array me copy

⏱ Complexity

- **Time:** `O(n)`
- **Space:** `O(n)`

✅ Accepted
❌ Extra memory use

🧠 Example

```
arr = [1,2,3,4,5], k=2

new positions:

3 → 0

4 → 1

1 → 2

2 → 3

3 → 4
```

⭐ APPROACH 3 — REVERSE METHOD (BEST)

🧠 First-Thought Insight

Rotation = **Reversal ka game**

🛠 Logic

1. Poora array reverse
2. First `k` elements reverse
3. Remaining `n-k` elements reverse

⏱ Complexity

- **Time:** `O(n)`
- **Space:** `O(1)`

⭐ **Most optimal**
⭐ **Interview favourite**

📌 **Golden Line**

> “Rotation can be achieved using three reversals.”

🔢 ③ SUM OF UNIQUE ELEMENTS

❓ Problem

Sirf wo elements jinka **frequency = 1** ho
unka sum nikaalna hai.

❌ APPROACH 1 — Brute Force

🧠 Idea

Har element ke liye poora array check karo

⏱ Complexity

- **Time:** `O(n²)`
- **Space:** `O(1)`

❌ Slow

✅ APPROACH 2 — FREQUENCY ARRAY (BEST)

🧠 Key Observation

Constraints limited hain (1–100)

🛠 Logic

1. Frequency count
2. Sirf `freq == 1` wale add karo

⏱ Complexity

- **Time:** `O(n)`
- **Space:** `O(1)` (fixed size)

📌 Interview Line

> "We use a frequency array to track unique elements efficiently."

🥈 ④ SECOND LARGEST ELEMENT

❓ Problem

Array ka **second largest distinct** element find karo.

🧠 Two-Pass Thinking

1. Largest nikaalo
2. Largest ko ignore karke second largest dhoondo

⏱ Complexity

- **Time:** `O(n)`

- **Space:** `O(1)`

📌 **Why two passes?**

- Distinct condition maintain hoti hai
- Edge cases handle hote hain

🥉 ⑤ THIRD MAXIMUM NUMBER (LeetCode 414)

❓ Problem

- 3rd **distinct** maximum
- Agar nahi mile → largest return

🧠 Key Insight

`LLONG_MIN` use karo
taaki negative extreme values safe rahein

🛠 Logic

1. First max
2. Second max (≠ first)
3. Third max (≠ first, second)

⏱ Complexity

- **Time:** `O(n)`
- **Space:** `O(1)`

📌 **Interview Tip**

> “Always think about extreme constraints.”

⚪⚫ ⑥ SEGREGATE 0s AND 1s

❓ Problem

- Saare `0` left
- Saare `1` right

✅ APPROACH 1 — COUNTING

🧠 Idea

- Count zeros
- Pehle zeros bhar do
- Baaki ones

⏱

- Time: `O(n)`
- Space: `O(1)`

⭐ APPROACH 2 — TWO POINTER (BEST)

🧠 Logic

- Left pointer → 0 dhundhe
- Right pointer → 1 dhundhe
- Galat jagah mile → swap

⏱ Complexity

- **Time:** `O(n)`
- **Space:** `O(1)`

📌 **Interview Line**

> “Since values are only 0 and 1, two pointers works optimally.”

👑 ⑦ MAJORITY ELEMENT (LeetCode 169)

❓ Problem

Element jo **n/2 se zyada baar** aaye.

❌ Brute Force

- `O(n²)`
- ❌ Slow

⭐ MOORE’S VOTING ALGORITHM (BEST)

🧠 First-Thought Principle

Majority element **cancel hone ke baad bhi bachta hai**

🛠 Logic

- Count = 0 → new candidate
- Same → count++
- Different → count--

⏱ Complexity

- **Time:** `O(n)`
- **Space:** `O(1)`

📌 **Golden Interview Line**

> “Moore’s Voting Algorithm finds majority element in linear time and constant space.”

🧠 REAL LEARNING FROM LECTURE 09

Concept	Skill Built
Rotation	Index manipulation
Reverse	Two pointer mastery
Frequency	Constraint-based optimization
Max problems	Multi-pass logic
Segregation	Pointer movement
Majority	Mathematical cancellation

🏁 FINAL SUMMARY (SAVE THIS)

- Brute force = clarity, not efficiency
- Modulo = index rotation magic
- Reverse = space-optimal rotation
- Frequency = constraint exploitation
- Two pointer = in-place optimization
- Moore’s Voting = pure algorithmic beauty

🧠INTERVIEW GOLD LINES

- “Reverse technique reduces space to O(1).”
- “Two pointer avoids extra memory.”
- “Moore’s algorithm relies on cancellation.”

✨ **Made By — Harshal Chauhan**
