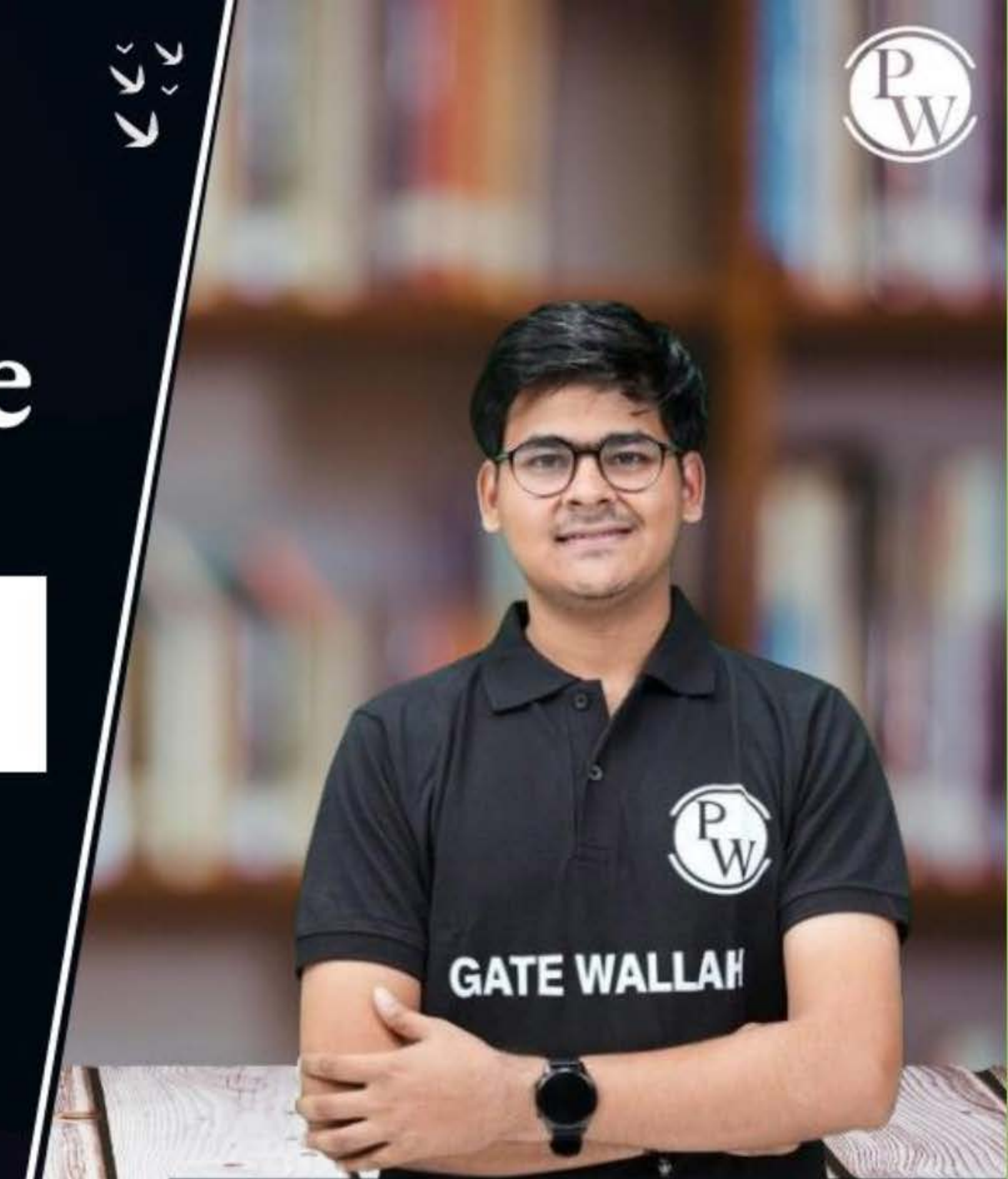
# Data Science & Artificial Intelligence

# Algorithms

Test Series 1500+




Lecture - 08

By- Aditya sir

# Recap of Previous Lecture

**Topic** Misc Concepts → Questions

**Topic**

# Topics to be Covered

Topic

Topic

Questions (Misc)

- Greedy
- DP (Graphs)
- DnC
- TC
- Sorting

Topic : Divide and Conquer

V. Imp (Apti)

#Q17. Consider the following array with 98 as the first element, all other elements can be in any order.

98, 66, 77, 105, 100, 96, 136, 64

Quick sort partition algorithm is used by choosing $1^{st}$ elements as pivot, then what is the total number of arrangements of integer is possible to preserve the effect of first pass of partition algorithm?

Soln:- Given: {98 66 77 105 100 96 136 64}

98 → Pivot

After Partition Algo:

1) Pivot → correct position

2) left of Pivot < Pivot

3) Right of Pivot > Pivot

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Sorted order: | 64 | 66 | 77 | 96 | $\boxed{98}$ | 100 | 105 | 136 |

After Partition:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 64 | 66 | 77 | 96 | 98 | 100 | 105 | 136 |

$$\underbrace{\{64 \quad 66 \quad 77 \quad 96\}}_{\text{any Combination}} \quad 98 \quad \underbrace{\{100 \quad 105 \quad 136\}}_{\text{any Combination.}}$$

$4!$ $\qquad$ $1$ $\qquad$ $3!$

$4! + 3!$

$4! * 3!$

Total Combinations

$= 4! \times 3! \times 1$

$= 24 \times 6 = \boxed{144}$

## Topic : Analysis of algorithm

$\underline{\Theta\ \%}$

#Q. Suppose that there are 3 programs $X_1$, $X_2$ and $X_3$ having time complexities $f_1(n)$, $f_2(n)$ and $f_3(n)$ respectively. Such that $f_1(n)$ is $O(f_2(n))$, $f_2(n)$ is $O(f_1(n))$, $f_1(n)$ is $O(f_3(n))$ and $f_3(n)$ is not $O(f_1(n))$. Then which one of the statements is true from the following statements?

A. $X_3$ is always faster than $X_1$ and $X_2$ for very large size inputs

less time

$\hookrightarrow (f_3 < f_1 \,\&\, f_2)$

B. $X_1$ is faster than $X_2$ and $X_3$ for very large inputs

$f_1 < f_2 \,\&\, f_3$

C. $X_3$ is slower than $X_1$ and $X_2$ for very large input

$f_3 > (f_1 \,\&\, f_2)$

D. $X_2$ is faster than $X_1$ and $X_3$ for very large size inputs

$f_2 < (f_1 \,\&\, f_3)$

Soln :- Given :-

$f_1(n) = O(f_2(n)) \rightarrow f_1 \le f_2$

$f_2(n) = O(f_1(n)) \rightarrow f_2 \le f_1$

$\boxed{f_1 = f_2}$

$f_1(n) = O(f_3(n)) \rightarrow f_1 \le f_3$

$f_3(n) \ne O(f_1(n)) \rightarrow f_3 \not\le f_1$

$\Rightarrow f_3 > f_1$

$\boxed{\text{Ans : C}}$

$f_1 \le f_3$

and $f_1 < f_3$

$\rightarrow \boxed{f_3 > f_1}$

Conclusion:

$\boxed{f_2 = f_1 < f_3}$

[MCQ]

Bellman Ford → SSSP
→ DP

#Q15. Consider the following statements

S1: for every weighted graph and any two vertices p and q, Bellman ford algorithm starting at p will always return a shortest path to q.

False

S2: At the termination of Bellman ford algorithm even if graph has negative weight cycle, correct shortest path is found for vertex for which shortest path is well-defined.

→ True

Which of the statement is correct?

Ans: B

A. only S1

B. only S2

C. Both S1 and S2 are true

D. neither S1 nor S2 is true

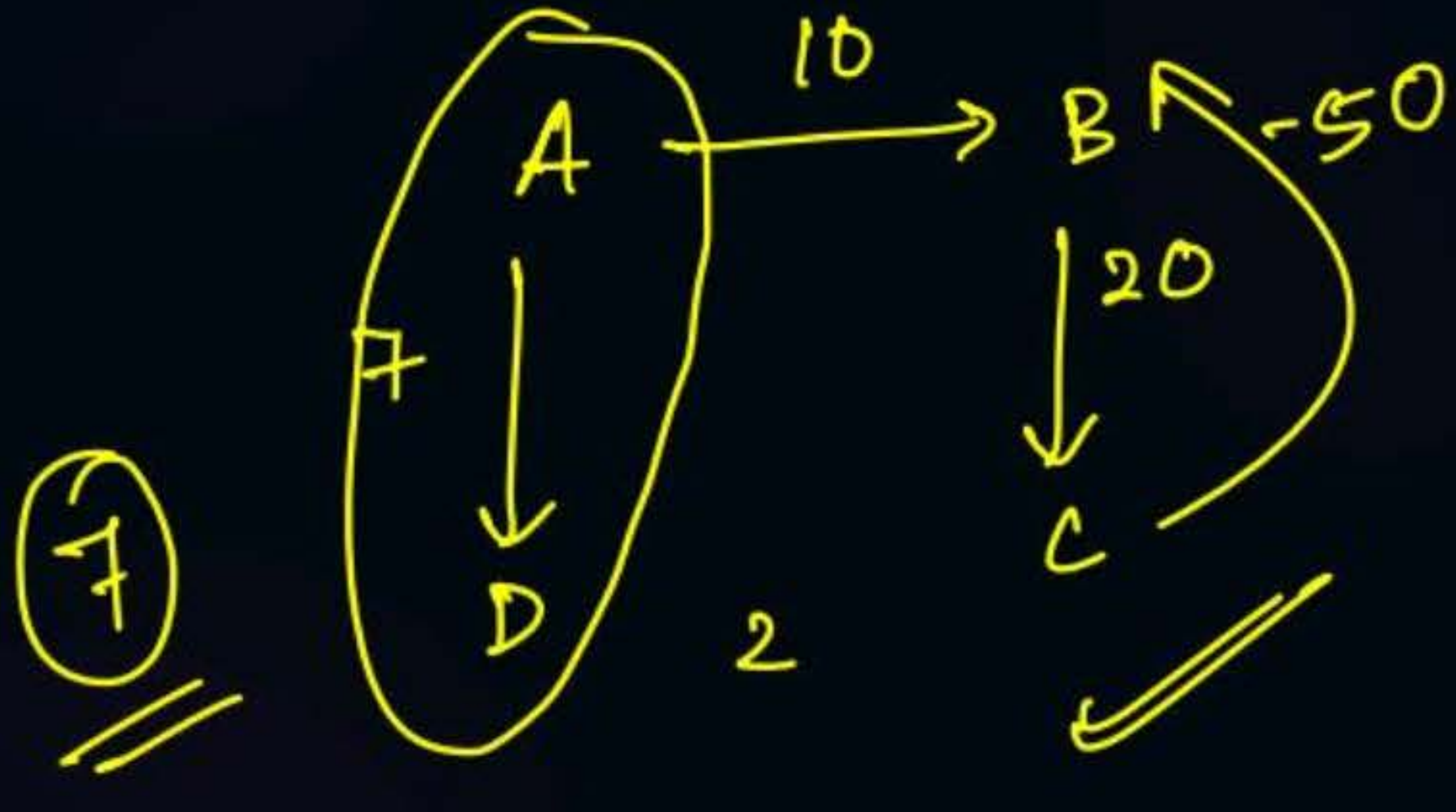
Soln :

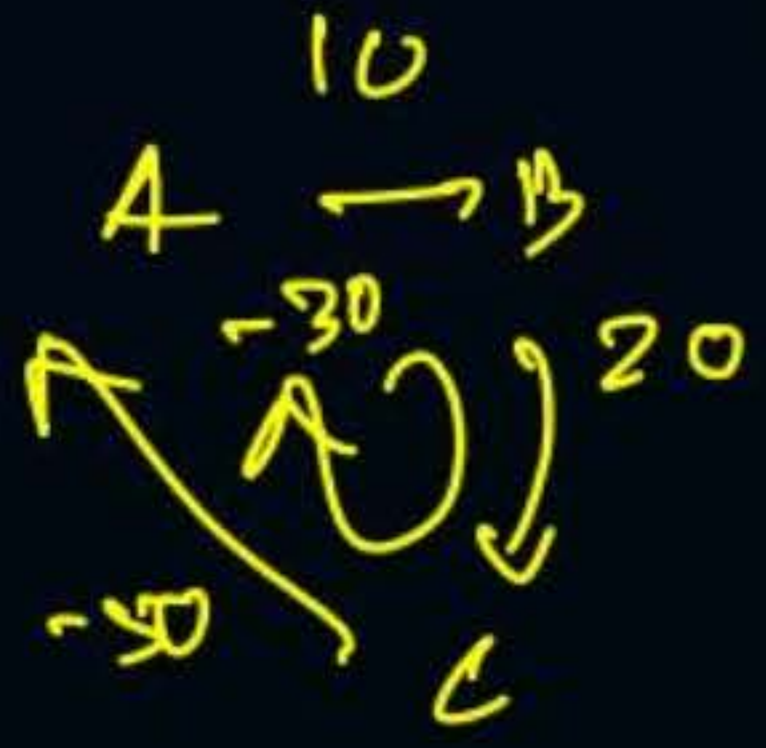
$$\hat{p} \rightsquigarrow \hat{q}$$

always?

(may or may not)

$$\left\{ \begin{array}{l} -ve \text{ wt Cycle} \longrightarrow \text{No} \\ \text{no } -ve \text{ wt Cycle} \longrightarrow \text{Yes} \end{array} \right\}$$


7
A
10
B
-50
20
7
C
D
2


10
A
B
-30
A
20
-50
C

## Topic : Divide and Conquer

#Q10. Consider a variation of merge sort in which we divide the list into 4 sub lists of equal size, recursively sorting each list, and then merging the four lists to get the final sorted list.

What is the recurrence relation that is required for the number of comparisons used by this algorithm in worst case?

4

(NOTE: Assume that the number of elements to be sorted is a power of 3 so that all of the divisions are into three sub lists workout evenly)

**A** $T(n) = 4T(n/3) + n$ ✗

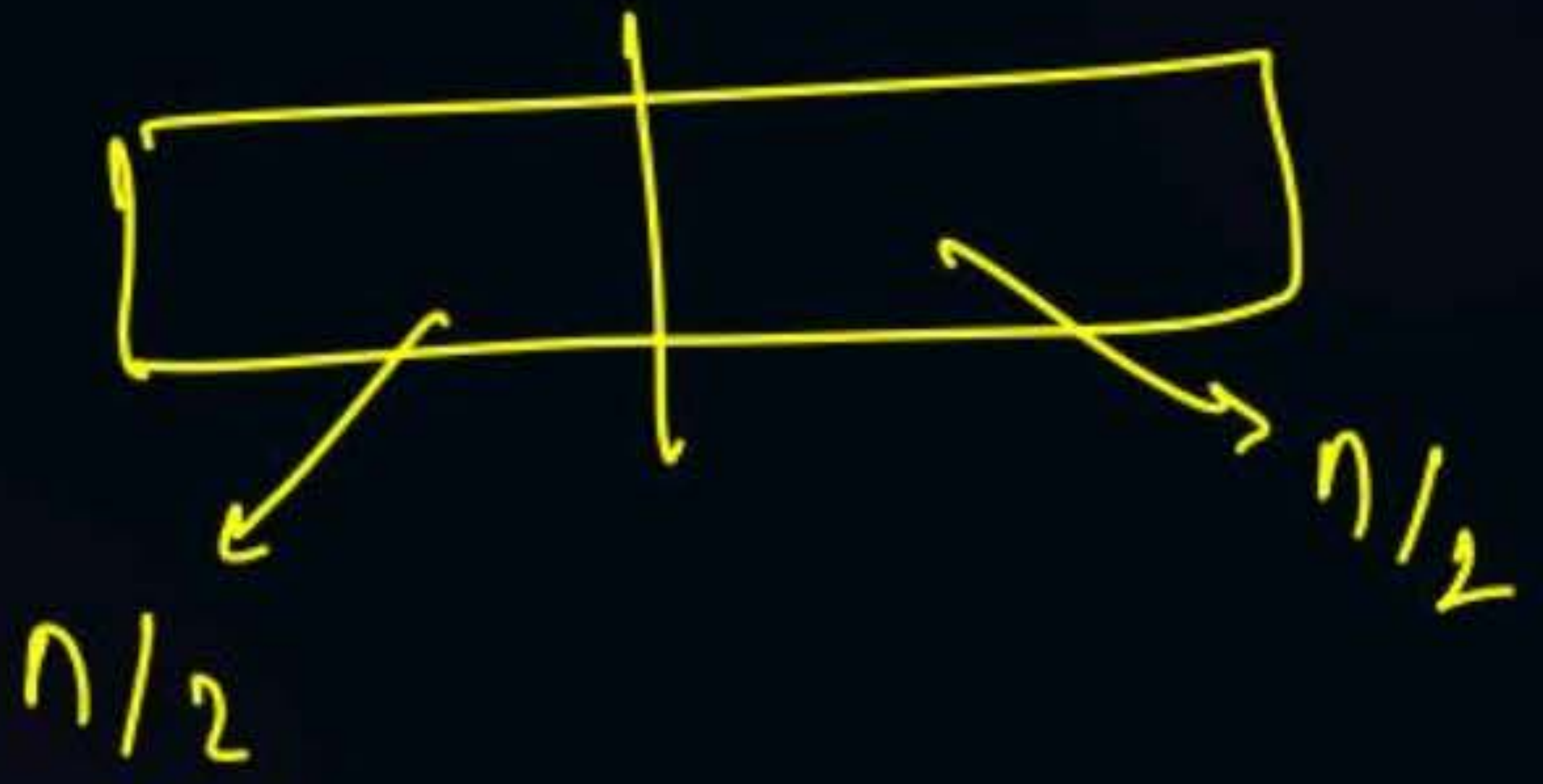
**B** $T(n) = 4T(n/4) + n - 1$ ✓

**C** $T(n) = 2T(n/4) + n - 1$ ✗

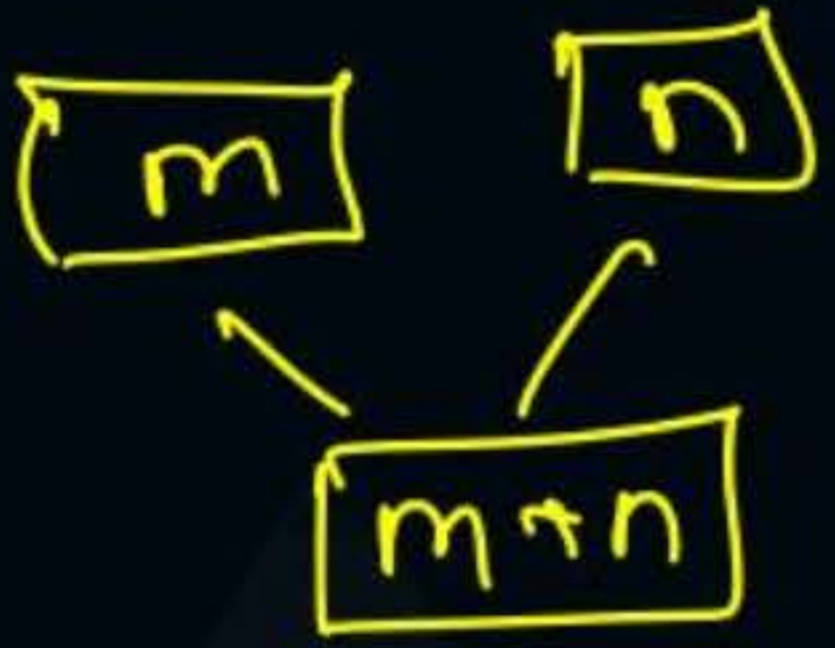
**D** None of these ✗

Ans : B

## Soln :- Standard Merge Sort :




WC : Merge Algo




m+n-1 Comp

WC: $\frac{n}{2} + \frac{n}{2} - 1$

$= (n-1)$ Comparisons

$$T(n) = T(n/2) + T(n/2) + (n-1)$$

$$\boxed{T(n) = 2T(n/2) + (n-1)}$$

New variant

$T(n)$

| $n/4$ | $n/4$ | $n/4$ | $n/4$ |
| --- | --- | --- | --- |

$$\left[\, T(n/4) \quad T(n/4) \quad T(n/4) \quad T(n/4) \,\right] \rightarrow (n-1)$$

comp

$2 \rightarrow n/2$

$4 \rightarrow n/4$

Overall TC: $4T(n/4) + (n-1)$

#Q11. Consider ~~a list which contains~~ np sorted arrays each of size n/p and is merged using merge sort, then what is the tightest upper bound worst case complexity?

A. $O(np^2 \log np)$

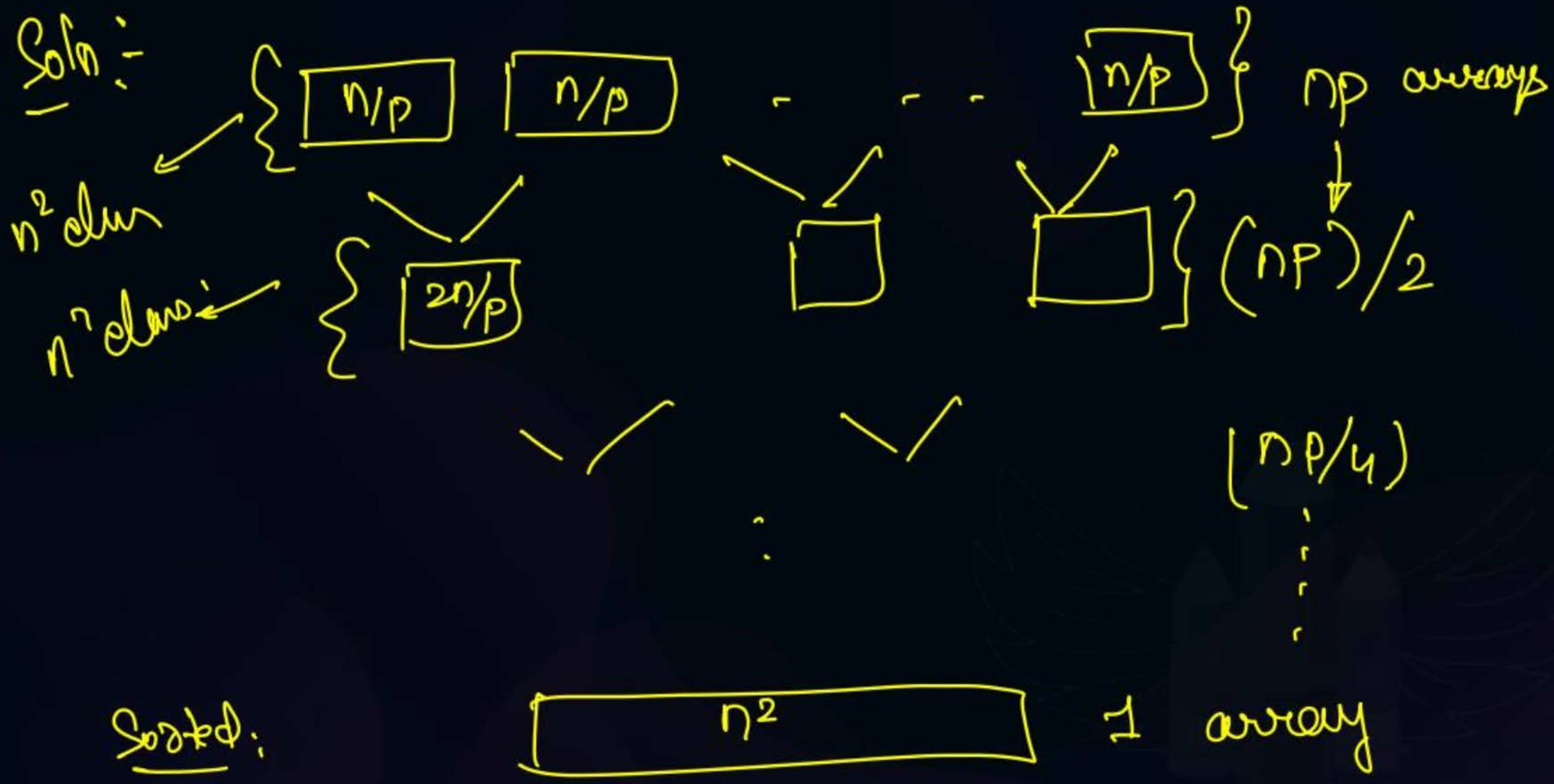
B. $O(n^2 \log n)$

C. $O(n^2 \log np)$

D. None of these

Ans : C

Soln :-



Total no. of elems = $(n\cancel{p} * n/\cancel{p})$

$= \boxed{n^2}$ elems. (at each level)

TC for merging at each level $\longrightarrow$ $O(n^2)$

No. of levels = $\log(np)$

$\left[ np \rightarrow np/2 \rightarrow np/2^2 \; . \; . \; . \rightarrow 1 \right]$

Total (overall) TC = $\log(np) * n^2$

$= \underline{\underline{n^2 \log(np)}}$

6

[NAT]

#Q17. How many binary trees are possible for 4 elements?

( diff structure )

$2^n - n \Rightarrow 2^4 - 4 = 16 - 4 = \boxed{12}$

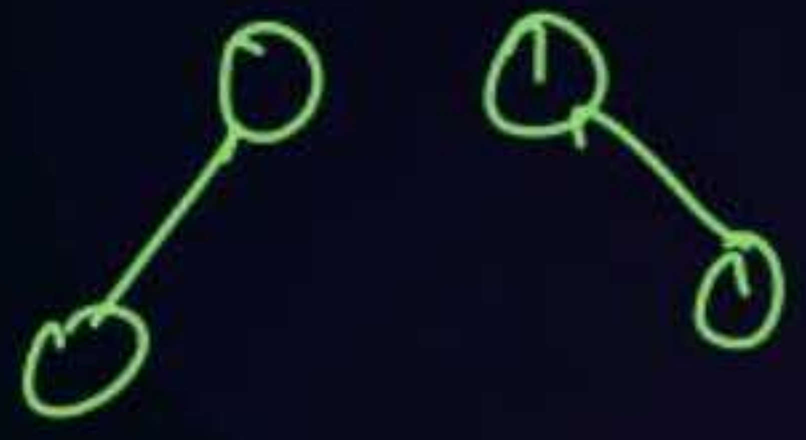
BSTs

Soln:- Catalan no: $^{2n}C_n * \frac{1}{(n+1)}$

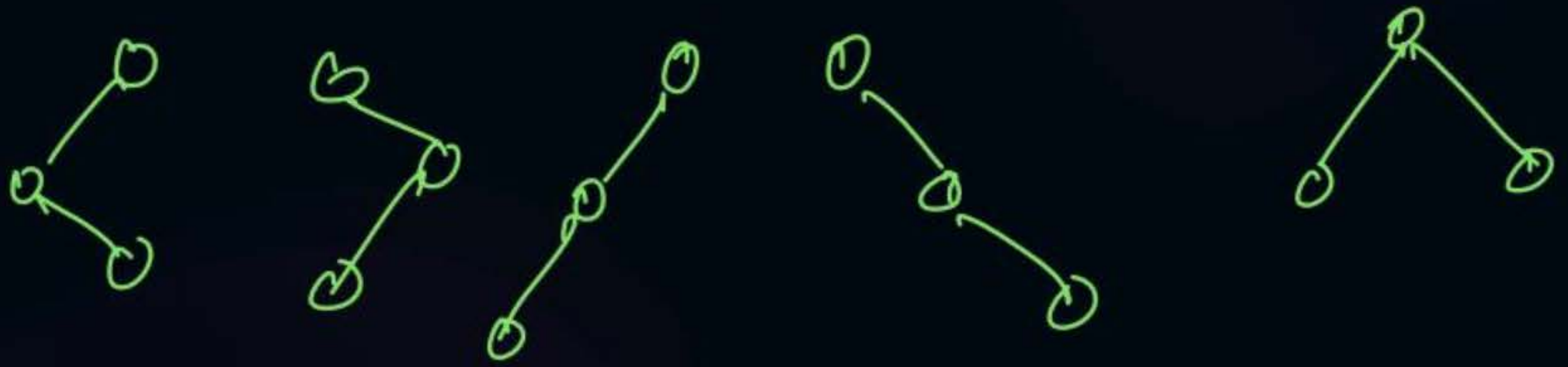
BSTs

for $n=1$ $\quad {}^2C_1 \times \frac{1}{2} = 2 \times \frac{1}{2} = 1$

$n=2$ $\quad {}^4C_2 \times \frac{1}{3} \Rightarrow \frac{4!}{2!\,2!} \times \frac{1}{3} = \frac{\cancel{4}\times\cancel{3}}{\cancel{2}\times\cancel{3}} = 2$



$n = 3$ $\qquad {}^6C_3 \times \frac{1}{4} = 5$



$\Rightarrow$ for $\underline{n = 4}$ : $\qquad {}^{2n}C_n \times \frac{1}{(n+1)}$

$$\Rightarrow {}^8C_4 \times \frac{1}{5} \Rightarrow \frac{8!}{4! \times 4!} \times \frac{1}{5}$$

$$= \frac{8 \times 7 \times 6 \times \cancel{5} \times \cancel{4!}}{4! \times \cancel{4!}} \times \frac{1}{\cancel{5}}$$

$$= \frac{\cancel{8} \times 7 \times \cancel{6}}{\cancel{4} \times \cancel{6}} = \boxed{14}$$

$2^n - n$

No. of diff Binary Trees $\longrightarrow (2^n - n)$

No. of BSTs $\longrightarrow {}^{2n}C_n * \frac{1}{(n+1)}$

# Topic : Analysis of algorithm

#Q. $f(n) = \sum_{i=1}^{n} i^3$ then choices for f(n)

$\theta(n^4)$

I. $\theta(n^3)$ ✗

II. $\theta(n^5)$ ✗

III. $O(n^5)$ ✓

IV. $\Omega(n^3)$ ✓

A I ✗

B II ✗

C III ✓

D IV ✓

Ans: C, D

Soln:-

$$F(n) = \sum_{i=1}^{n} i^3$$

$$= 1^3 + 2^3 + 3^3 \cdots\cdots n^3$$

---

$$\left(\sum n\right)^2 \Rightarrow \left[\frac{n(n+1)}{2}\right]^2 \qquad \cancel{\frac{(n(n+1))^2}{2}}$$

(1) $\sum_{i=1}^{n} i = 1+2+3 \dots n \Rightarrow \boxed{\frac{n(n+1)}{2}}$

(2) $\sum_{i=1}^{n} i^2 = 1^2+2^2+\dots n^2 \Rightarrow \boxed{\frac{n(n+1)\times(2n+1)}{6}}$

(3) $\sum_{i=1}^{n} i^3 = 1^3+2^3+ \dots n^3 = \left[\frac{n(n+1)}{2}\right]^2$

$$f(n) = \sum_{i=1}^{n} i^3 = 1^3 + 2^3 \cdot \cdot \; n^3$$

$$= \left[\frac{n(n+1)}{2}\right]^2 \longrightarrow$$

$$(ab)^2 = \underline{a^2 b^2}$$

$$= \frac{n^2 (n+1)^2}{4} = \frac{n^2 (n^2 + 2n + 1)}{4}$$

$$= \left[\frac{n^4 + 2n^3 + n^2}{4}\right] = \underline{\underline{\theta(n^4)}}$$

i) $\Theta(n^3)$ ✗

ii) $\Theta(n^5)$ ✗

iii) $O(n^5)$ ✓

iv) $\Omega(n^3)$ ✓

$\Theta(n^4)$

| $O(n^4)$ | $\Omega(n^4)$ |
|---|---|
| $O(n^5)$ | $\Omega(n^3)$ |
| $O(n^6)$ | $\Omega(n^2)$ |
| $\vdots$ | $\Omega(n)$ |
| | $\vdots$ |

[NAT]

#Q16. A

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 10 | 20 | 50 | 60 | 70 | 65 | 55 | 25 | 15 |

How many swaps are needed to sort the array by using Bubble sort______?

Ans: 16

Soln:

A: 10 20 50 60 70 65 55 25 15

i/p

pass 1: 10 20 50 60 ~~70~~ ~~65~~ ~~55~~ ~~25~~ 15 → 4

65 ~~70~~ ~~70~~ ~~70~~ → 70

55 25 15

pass 2: [10 20 50 60 ~~65~~ ~~55~~ ~~25~~ ~~15~~ 70] → 3

55 ~~65~~ ~~65~~ 65

25 15

pass 3: [10 20 50 ~~60~~ ~~55~~ ~~25~~ ~~15~~ 65 70] → 3

55 ~~60~~ ~~60~~ 15 60

25

pass 4: [10 20 50 ~~55~~ ~~25~~ ~~15~~ 60 65 70] ⟶ 2

25 ~~55~~ 55

15

pass 5: 10 20 ~~50~~ ~~25~~ ~~15~~ 55 60 65 70 ⟶ 2

25 ~~50~~ 50

15

pass 6: 10 20 ~~25~~ ~~15~~ 50 55 60 65 70 ⟶ 1

15 25

pass 7: [10 ~~20~~ ~~15~~ 25 50 55 60 65 70] ⟶ 1

15 20

pass 8: 10 15 20 25 50 55 60 65 70 ⟶ 0

Total Swaps

$= \underline{4} + \underline{3+3} + \underline{2+2} + 1 + 1 + 0$

$= 10 + 6$

$= \boxed{16}$

Topic : Test Series 1500+

#Q. Consider two array $A_1$ and $A_2$ size of array A1 and A2 is P and Q respectively if both are sorted, then how much ~~time~~ it will take to merging both array into single sorted list ?

no. of Comparisons

$\Omega$

0%

A. O(max(P, Q)) 0%

B. O(min (P, Q))

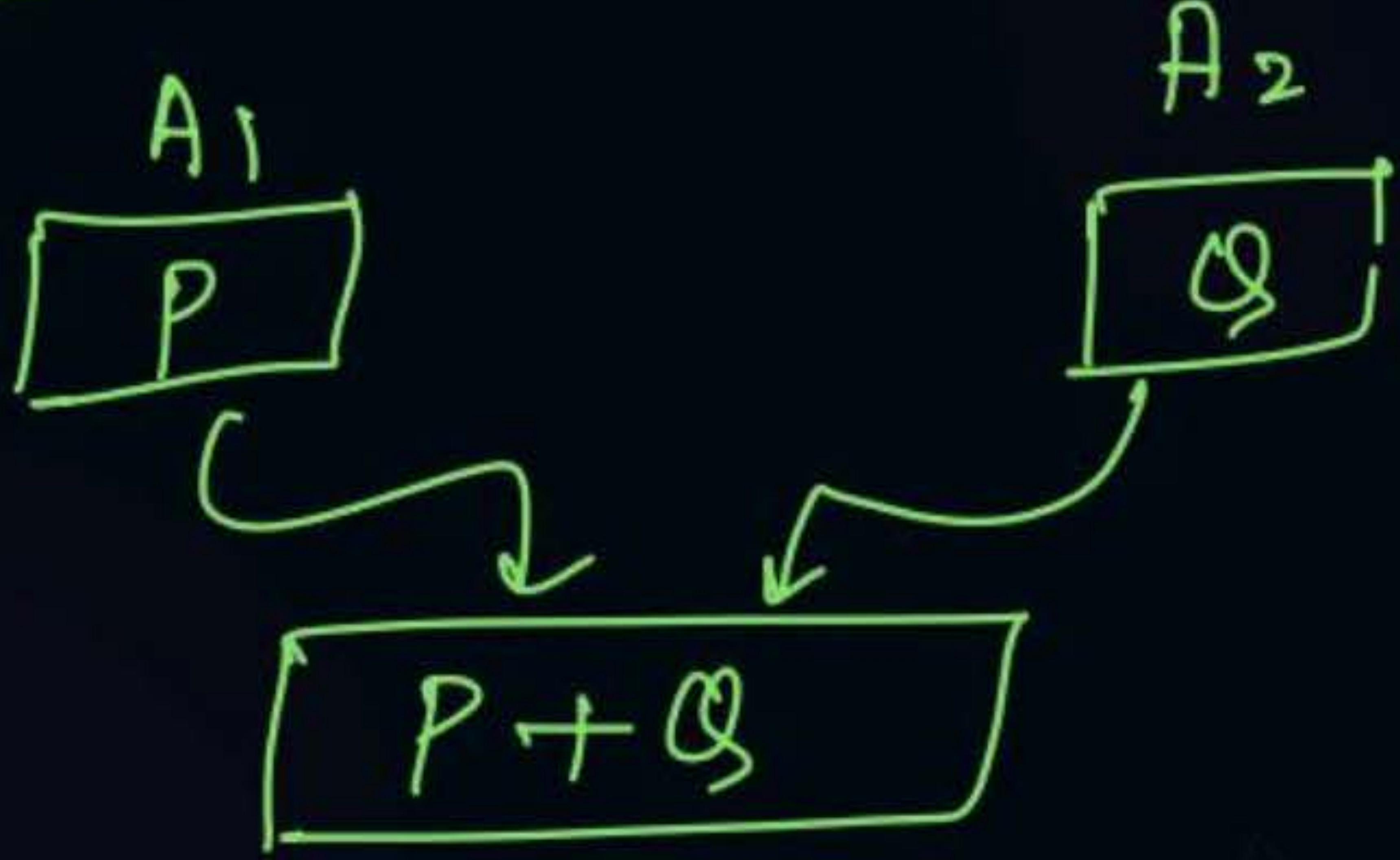
C. O(P + Q)

D. O(log P + log Q)

Ans: A, C

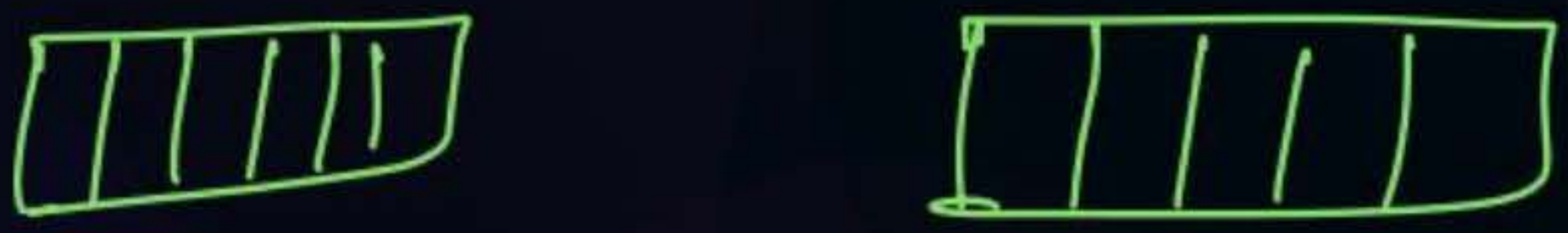
## Soln:- Merging Algo

$A_1$ & $A_2$ sorted




**No. of Comparisons**

BC :



$\Rightarrow O(\min(P, Q))$

WC :

| | | | | |
|---|---|---|---|---|
| 2 | 5 | 10 | 15 | 50 |

| | | | | |
|---|---|---|---|---|
| 16 | 17 | 18 | 19 | 20 |

$\Longrightarrow (P+Q-1)$

↳ worst case ⟶ max comparisons.

$$(P+Q-1)$$

$$\hookrightarrow O(P+Q-1) = O(P+Q)$$

$$O(P+Q) = \underline{O(\max(P, Q))}$$

$$\underline{O(n^2+n^3) = O(n^3) = O(\max(n^2, n^3))}$$

## Topic : Test Series 1500+

#Q. Consider the following elements:

80, 47, 56, 54, 13, 28, 89,16, 48, 18

If straight two way merge sort algorithm is used to sort the above elements in decreasing order. Then what is the order of these elements after $2^{nd}$ pass of the algorithm is :

A 47,54,56,80,13,16,28,89,18,48 X

B 80,56,54,47,89,28,16,13,48,18 ✓

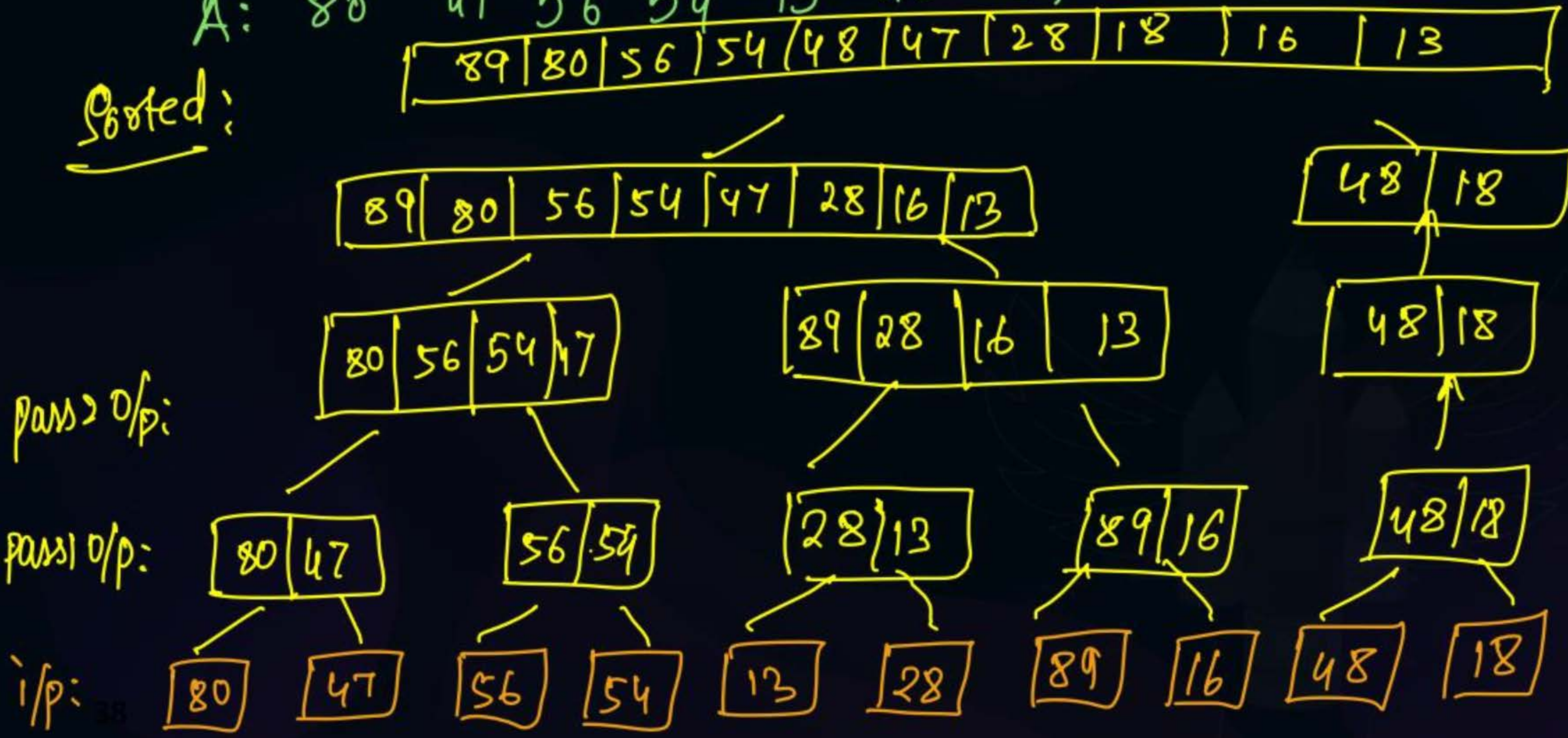
C 13,18,16, 28, 80, 47, 56, 54, 89, 48 X

D None of these X

Ans: B

Soln: Straight 2-way merge sort

A: 80 47 56 54 13 28 89 16 48 18



O/p : 80 56 54 47 89 28 16 13 48 18

pass 2

# 2 mins Summary

Topic

Topic

Topic

Misc Questions

Topic

# THANK - YOU

**Telegram Link for Aditya Jain sir:**
**https://t.me/AdityaSir_PW**