Striver SDE Sheet : Day-2 :: Arrays :-

A] Rotate matrix :- (or Rotate Image)

- Rotate a matrix 90° clockwise.
-
- Approach-1 :-

→ $\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 & 8 \\ 2 & 9 & 10 & 11 & 12 \\ 3 & 13 & 14 & 15 & 16 \end{array} \xrightarrow{90^\circ} \begin{bmatrix} 13 & 9 & 5 & 1 \\ 14 & 10 & 6 & 2 \\ 15 & 11 & 7 & 3 \\ 16 & 12 & 8 & 4 \end{bmatrix}$

i j → j (n-i-1)

→ [0][0] → [0][3]
[0][1] → [1][3]
[0][2] → [2][3]
[0][3] → [3][3]

[1][0] → [0][2]
[1][1] → [1][2]
[1][2] → [2][2]
[1][3] → [3][2]
i → (n-1-i)

- Approach-2 :-

→ in-place solⁿ.

→ Row becomes column ⇒ Transpose of matrix
(Row-wise) Reverse transpose matrix = ans

Tranpose :- $\begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix} \xrightarrow{\text{Reverse}}$

[0][1] → [1][0]
[0][2] → [2][0]
[0][3] → [3][0]
[1][2] → [2][1]
[1][3] → [3][1]
[2][3] → [3][2]

→ Transpose → Reverse

↪ All diagonal ele. stay where they are

Reverse
↪ only need to traverse for right half
↪ 0th row j = 1 to 3
1st row j = 2 to 3
2nd row j = 3
3rd row no triversel ⇒ (n-2) loop

$\underbrace{i}_{\text{outer}} \rightarrow \underbrace{j+1}_{\text{inner loop}}$

B] merge overlapping subintervals :-

- Given an array of intervals, merge all the overlapping intervals & return an array of non-overlapping intervals.

e.g:- [1,3], [2,6], [8,9], [9,11], [8,10], [2,4], [15,18], [16,17]]

⇒ ans → [[1,6], [8,11], [15,18]]

- Approach-2:-

① Group closer intervals by sorting &

② Transverse arr & insert 1st ele. in ans list.

③ Now, for next elements :-

case-1: If curr interval can be merged with last inserted interval of answer list

⇒ update end (last interval) = max. (end (curr int), end (last int))

case-2: If curr interval cannot be merged with last inserted interval in answer list.

⇒ Insert curr interval in answer array as it is

$$\text{interval}[\,][\,] = [\overset{a}{\downarrow}[1,3],\ \overset{b}{\downarrow}[2,6],\ \overset{c}{\downarrow}[8,10],\ \overset{d}{\downarrow}[15,18]]$$

→ mergedIntervals [ ][ ] = [[1,3]] ⇐ a

[[1,6]] ⇐ b [1,3] + [2,6]

[[1,6], [8,10]] ⇐ c

[[1,6], [8,10], [15,18]] ⇐ d

- Approach-3:-

→ Space optimization by in-place merging.

→ Pointer based merging.

C] merge two sorted arrays without extra space :-

- Given two sorted arrays (arr1 [ ], arr2 [ ]) of size n & m in non-decreasing order. merge them in sorted order. modify arr1 so that it contains first N elements & modify arr2 so that it contains last M elements.

- Approach -3:-

→ Gap method (intuition from shell sort technique)

① Assume the two arrays as a single array & calculate the gap value ~~&~~ i.e. ceil (size of arr1[] + size of arr2[]/2)

② Perform following operations for each gap until the value of gap becomes 0.

a) Place 2 pts in their correct pos. like left ptr = 0 & right ptr = left + gap

b) Again run loop till right < n+m, with 3 conditions:-

→ If left ptr is inside arr1[] & right pts inside arr2

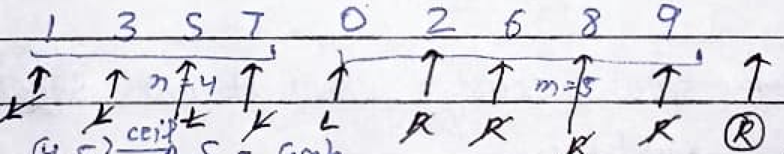
→ If both ptrs are in arr2[]

→ If both ptrs are in arr1[]

c) After right ptr reaches end, we will decrease value of gap & it will become ceil (curr gap/2).

~~S~~

→ arr1 = [1, 3, 5, 7] arr2 = [0, 2, 6, 8, 9]



1) $\frac{n+m}{2}$ = (4.5) $\xrightarrow{ceil}$ 5 = Gap

when ® moves out of bound stop & restart.

2) $\frac{Gap}{2}$ = $\frac{5}{2}$ = (gap = 3) = right-1 => (right=4), Left=0

Again same

3) 3/2 = [1.5] = 2 = gap , Left=0, right = 2+1=3

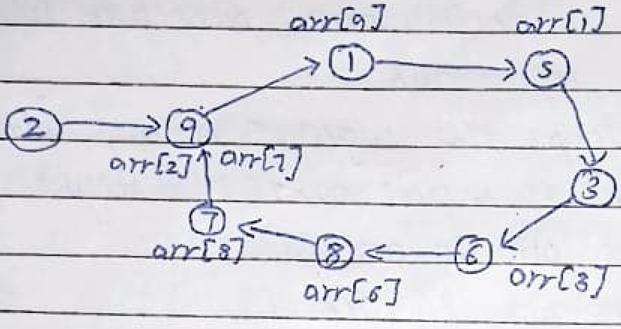
4) 2/1 = [1] = 1

5) 1/2 = [0.5] = 1 → stop here, so whenever you get 1 for first ~~then~~ time then it is your final iteration.

Q] Find the duplicate in an array of N+1 integers:-

- Given an array of size N+1, each ele. is b/w 1 to N. Assuming there is only 1 duplicate number, your task is to find the duplicate number.
- Approach-3:- Linked List cycle method

→ arr = [2, 5, 9, 6, 9, 3, 8, 9, 7, 1] (indices 0 1 2 3 4 5 6 7 8 9) → form cycle (if it does) → surely there is a duplicate number



→ Now apply tortosise-hare method (slow-fast pointer)

```
slow = slow+1;
fast = fast+2;
```

→ starting both at idx=0
→ Till they both collide

```
slow = slow+1;
fast = fast+1;
```

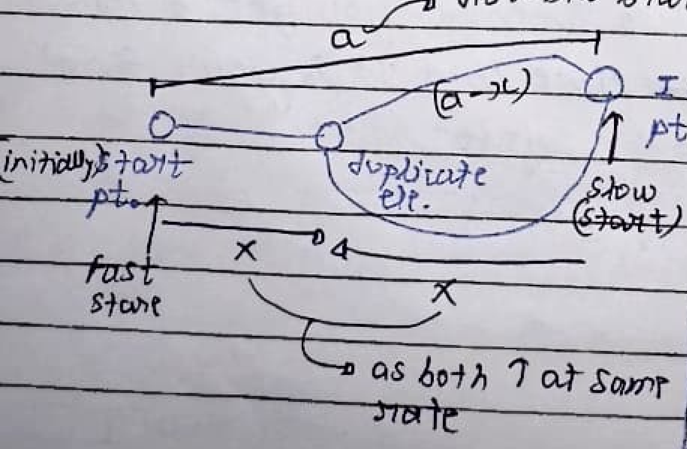
→ slow (exactly where it was last)
Fast = 0
→ Now this

The point where they collide again → Duplicate number

→ Proof:-

I collision:- cycle exist, f(+=2) & s(+=1) ⇒ they will collide

II collision:-



slow → a
fast → 2a

$\boxed{\Delta = a}$

This (a) will always be multiple of len. of this cycle.

slow & fast are bound to meet at duplicate ele.

→ TC = O(n) , SC = O(1)

E] Find the Repeating & missing numbers:-

- You are given a read-only array of N integers with values also in range [1, N] both inclusive. Each integer appears exactly once except A which appears twice & B which is missing. The task is to find the repeating & missing numbers A & B where A repeat twice & B is missing.

- Approach-3:- Math.

→ x : Repeating no. , y : missing no.

→ $S_n = \frac{n(n+1)}{2}$ ✓, Sarr → 2x, 0y included ✓

$\boxed{S_{arr} - S_n = x - y}$ ✓ —①

$$\boxed{\begin{aligned} S^2_{arr} - S^2_n &= x^2 - y^2 \\ &= (x-y)(x+y) \\ &\Rightarrow x+y \checkmark \quad —② \end{aligned}}$$

$S^2_n = \frac{n(n+1)(2n+1)}{6}$

From ① & ② x = ✓, y = ✓

- * Approach-4:- XOR ★

→ X : Repeating no. , Y : missing no.

→ $0 \wedge 0 = 1 \wedge 1 = 0$, $\underbrace{1 \wedge 0 = 0 \wedge 1 = 1}_{\text{one difft bit}}$

→ arr[] = [4 3 6 2 1 1] N = 6 } → int [1 to N]

[4 3 6 2 1 1] ⟷ [1 2 3 4 5 6]

(e.g.) These are 3 bit integers (here). Lets look at bit No. 2

→ Bit No. 2 : LSB

0 | 1

odd one out ← XOR = 1 → [3, 2, 1, 1, 1, 2, 3] | [4, 6, 4, 5, 6] → XOR = 5 → odd one out

2nd bit

0 0 1 → 1

^ 1 0 1 → 5

1 0 0

all No.'s → even (2x)

X → odd (3x) , Y → odd (1x)

→ ans = {1, 5} → which is x & y ?

→ Summary:-

1. $\underset{(XOR)}{(\overset{(0 \text{ to } n-1)}{arr[i]})}$ ^ (1^2^....^n) = num

2. Find a differentiating bit in num ; first one from right

3. Put them into 2 parts $\begin{cases} \to 0 \\ \to 1 \end{cases}$

E] Inversion of Array:-

- Given:- int[] arr, N integers, N size

  count inversion of array (using merge sort). Inversion of arr: definition (∀ i & j (< N) of array, if $\boxed{i<j}$, find pair (A[i], A[j]) s.t. $\boxed{A[j] < A[i]}$. Return count.

- Approach - 2 :- Using merge sort Algorithm.

→ Intuition: modified ver. of Q. (Given two sorted array)

a1[] = $[\overset{\downarrow i}{2}, 3, 5, 6]$ & a2[] = $[\overset{\downarrow j}{2}, 2, 4, 8]$

count pairs (a1[i], a2[j]) s.t. a1[i] > a2[j]

Cases:- (a1[i] <= a2[j]) → $\boxed{i++}$ //can't pair

(a1[i] > a2[j]) //can pair up

→ All ele. after a1[i] > a2[j] //can form pair

→ No. of pairs = $\boxed{n1}$ - i → len. of arr1

→ count = count + no. of pairs ;

$\boxed{j++}$; ~~count = 0+3+3+2+2 = 10~~

~~① a1[i] = 2 , a2[j] = 2 → i++ , count = 0~~

~~② a1[i] = 3 , a2[j] = 2 → count = 0 + (4-1) = 3 , i++, j++~~

~~③ a1[i] = 5 , a2[j] = 2 → count = 3 + (4-2) = 5 , i++, j++~~

~~④ a1[i] = 6 , a2[j] = 4 →~~

- Applying above obs. using merge sort Algo. (2 sorted array at each step → left & right)