NOTES

DATE 01/03/2025

LC - Biweekly Contest - 151 :-

A] 3467. Transform Array By Parity :- ✓

B] 3468. Find number of Copy Arrays :-

original = [1, 2, 3, 4]

bounds = [1,10], [2,9], [3,8], [4,7]

copy = x, y, z, q (no. of ele. satisfy the condⁿ given)

ans = x × y × z × q

copy = [x, _, _, _] (here 1 ele.)

arr[i] - arr[i-1] = copy[i] - copy[0]

d_1

$x + d_1$, x

$(x + d_1) + d_2$, $(x + d_1 + d_2) + d_3$

$d_1 = d_2 = d_3 = d_4$ => (here)

copy = [x, x+1, x+2, x+3]

bounds: [1,10] [2,9] [3,8] [4,7]

$1 \le x \le 10$, $1 \le x \le 9$, $1 \le x \le 8$, $1 \le x \le 7$, $1 \le x \le 4$

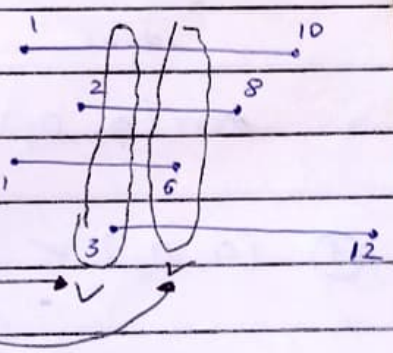
=> $1 \le x \le 4$ final

max of min. side values (lower bounds)

min. of max. side values (upper bounds)

ans

e.g. :-



C] 3469. Find min. cost to remove array elements :-

[C-1]

nums[] = [a_1, a_2, a_3, a_4] no. of op. = 2

(3) (2)

For 2nd Op. :-

nums[] = [a_1, a_2, a_3, a_4, a_5] ——— = 3

(3) (3) (1)

PAGE

⇒ Last op. can have either 2 or 1 element.

I op. $[a_1, a_2, a_3 \ldots]$

3 possibilities → a_1, a_2 / a_2, a_3 / a_3, a_1

wherever we have to try for all the possibilities ⇒ DP

→ So try for recu. for all possibilities.
→ Then memoize for our answer.

$nums[] = [\underbrace{a_1, a_2, a_3}_{\text{possibilities}}, a_4, a_5]$ (idx-1, idx, idx+1)

- We have not used idx_1, idx_2 & idx_3
- ∴ $dp[i_1][i_2][i_3]$
  - $O(n^3)$
  - We have to memoize it also.
  - We can only go up to $O(n^2 \log n)$ max.

① (a_1, a_2), $cost = max(a_1, a_2)$

Next stage :-

$nums = [a_1, a_2, a_3, a_4, a_5]$ (previdx, idx+2, idx+3; idx-1, idx, idx+1)

$cost = max(a_1, a_2) + solve(previdx, idx+2, idx+3)$

$previdx = idx+1$

② Do we need previdx ?

(a_2, a_3) ✓, $cost = max(a_2, a_3)$

$nums [\ a_1, a_2, a_3, a_4, a_5]$ (idx-1, idx, idx+1; idx+2, idx+3)

previdx (idx-1) — idx, idx+1

$max(a_2, a_3) + solve(previdx, idx+2, idx+3)$

→ $previdx = idx-1$

③ (a_1, a_3) ✓

$[a_1, a_2, a_3, a_4, a_5]$ (idx-1, idx, idx+1; idx+2, idx+3)

previdx, idx+2

$min.(a_1, a_3) + solve(previdx, idx+2, idx+3)$ → $O(n^3)$

→ $previdx = idx$

→ only previdx is variable

- So, we only need (prevIdx, idx) to derive minCost.
- code:- Base case
  memoization case
  ↳ dp[idx][previdx] != -1 => return dp[idx][previdx]

- From 3 cases:-
  ① opt-1 :- max(nums[prevIdx], nums[idx]) + solve(idx+1, idx+2)
  ② opt-2 :- max(nums[idx], nums[idx+1]) + solve(prevIdx, idx+2)
  ③ opt-3 :- max(nums[prevIdx], nums[idx+1]) + solve(idx, idx+2)

So minCost(opt1, opt2, opt3)
return dp[idx][prevIdx] minCost => memoize

- Base case:-
  ↳ 1 ele. Rem. or 2 => discussed initially
  if(idx == n) return num(prevIdx)
  if(idx == n-1) return max(num(prevIdx), num(idx