Two Pointer

Q1

01 subarrays with distinct integer

eg [1 | 2 | 1 | 3 | 4] → atmost 3

yes yes

1 + 2 + 3 + 4 + 3

= 13.

Now we use

atmost (3) − atmost (2) = ans

= 13 − 10

= 8.

eg. 1 2 1 3

→ 1 + 2 + 3 + 4 = 10 → atmost 2

= 10

→ 1 3 4 ~~+~~ 18 total

3 = 8

02 Two sum (without map).

eg [2, 6, 5, 8, 11] target = 14.

so, sort [2 5 6 8 11]

left right

2 + 11 = 13 < 14 left ++.

5 + 11 = 16 > 14 right −−

5 + 8 = 13 < 14 left ++.

6 + 8 = 14 = 14 yes.

- yeh yes or no yaha puche uske liye ye best way hai.

→ Now use hashing (map)

arr [] = [2 | 6 | 5 | 8 | 11]

target = 14

0 1 2 3

2 6 5 8 11

2 14−2 12 X

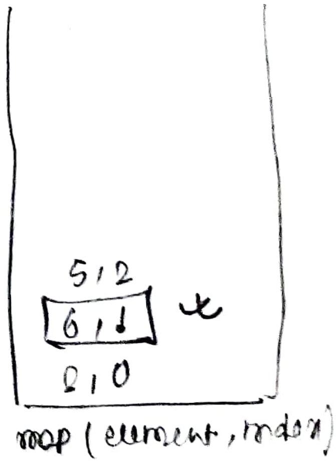
6 14−6 8 X

5 14−5 9 X

8 14−8 6 ✓ yes

TC = O(n log n)

SC = O(n).



return [1 | 3]

yes

03 3 sum

arr [] = [−1, 0, 1, 2, −1, −4]

ans = [−1, 2, −1] ans

[0, 1, −1]

return unique triplets.

arr [i] + arr [j] + arr [k] = 0

i != j != k.

que

arr[] = [-2, -2, -2, -1, -1, -1, 0, 0, 0, 2, 2, 2, 2]

i   i   j   j   j   k   j   k   k   sorted order

Fixed

-2 -2 +2 = -2 < 0   j↑

-1 -2 +2 = -1 < 0 → [-2, 0, 2]

-2 +0 +2 = 0 = 0. → [-1 -1 2].

if (k < j) loop ended. → [0 0 2].

now move i

04 4 sum

arr[] = after sorting.   target = 8

[1 1 1 2 2 2 3 3 3 4 4 4 5 5]

i j k   l

* num[i] + num[j] + num[k]. + num[l] == target.

* i ≠ j ≠ k ≠ l.

05 4 sum II

- Yaha ye logic lagana hai A[i] + B[j] = -(C[k] + D[l]).
- Here we put a part in map and find it in that map of that -ve part.

06 3 sum closest

arr = [-1 -3 2 4 5]

sort it [-3 -1 2 4 5]

sum = 2.   i   j   k   difference

-3 -1 +5 = +1 < 2 = 1

-3 +2 +5 = 4 > 2 = 2

-3 +2 +4 = 3 > 2 = 1