2 pointer

Q.39 given array, find triplets adding up to 0.

Approach 1 Brute force

→ try out all the triplets. TC: $O(n^3 \log m)$ SC = $O(m)$

→ add them to set

log m: time taken for insertion(set)

Approach 2 Hashing.

2 loops + hashing for third element.

```
for (i = 0 → n-1)  → hash[a[i]]--  } exhausted
  for (j = i+1 → n-1)  → hash[a[j]]--; }
      c = -(a+b)        → if c exist in hashmap
    (re add)                ~~ans set~~
                            add the triplet to set
```

TC → $O(N^2 \log m)$ ☹

SC → $O(m) + O(n)$ ☹

Approach 3 2 ptr approach

→ sort the array

→ keep a constant and b and c to the pointer

→ keep low ptr just after a and high just at the end of array

→ if ar[b] + ar[c] < ar[a], low++

else if ar[b] + ar[c] > ar[a], high--

else { set.add(ar[a], ar[b], ar[c])
  low++, high--; }

ignore duplicate while(num[i] = num[i+1]: i++

TC → $O(N \times N)$

SC → $O(M)$ auxiliary: $O(1)$.

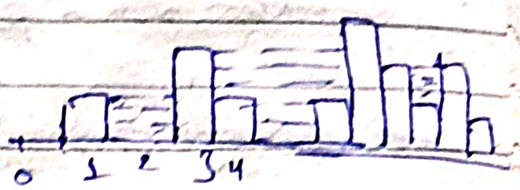
Q.40. Trapping Rainwater

Given n non -ve int, represents elevation map

→ find how much water it can trap after raining

Approach 1.

Brute force approach.



check condition

min (left(i), right(i)) - a[i]

TC → O(N^2)

SC → O(1) ☹

Approach 2 (Prefix Sum)

	0	1	0	2	1	0	1	3	2	1	2	1	
pref	0	1	ϕ	2	2	2	2	3	3	3	3	3	max(ξi-1)
suf	3	3	3	3	3	3	3	3	2	2	2	1	max(ξi+1)

TC → O(N) + O(N) + O(N)

ans = min (left(i), righ(i) - a[i]

SC → O(2n) ☹

Approach 3. (2 ptr. approach)

```
on l=0          res=0.          rightmax=0
   r=n-1        leftmax =0
if (a[l) <= a[r]) {
    if(a[i] = a[leftmax])
        leftmax = a[i]
    else
        res += (leftmax - a[i])
    l++
else
    if a[i] >= rigtmax
        rightmax = a[i]
    else
        res += (rmax - a[r]);
```

Q41. Remove duplicates from sorted array

Approach 1 Use hashset

TC → O(NLogN) + O(N)

SC ⇒ O(N)

Approach 2. 2 ptr approach

```
while (j<n)
    while (i==j)
        j++;
    ~~while~~ i++;
    a[i]=a[j];
    j++;
return i+1
```

TC → O(N)

SC → O(1)

Q42 Max consecutive ones

→ given binary array, find max. consecutive ones in array.

use 2 var.

cnt =0   max =0

→ when find 1, cnt += 1 and max = max(cnt, max)

→ when find 0, cnt =0.

Ans · max   TC → O(n)   SC → O(1)