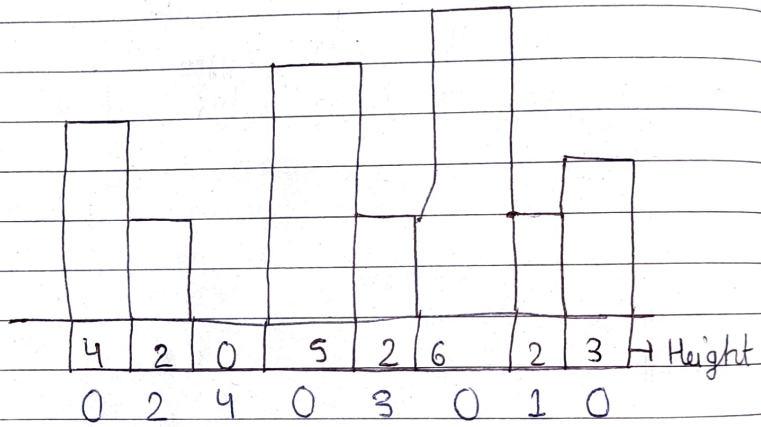
Date _____ Page _____

Day - 40

~~More Array~~

Important Array Questions

* Trapping Rain Water:



⇒ For trapping rain water, ~~we~~ the water should have support from the left & right.

⇒ So for calculating the trapped rain water at that building, we should do this—

⇒ we will check the left side bigger building & right side bigger building & select the min. from them.

⇒ And the minus with the building height.

⇒ At 4, there is no building at left. So, here 0.

⇒ At 2, left side bigger building is — 4 & at right side — 6,

Date _____ Page _____

⇒ min. is → 4

⇒ So, 4 - 2 = 2

⇒ We will create maxleft & maxright —

maxleft	0	4	4	4	5	5	6	6

maxright	6	6	6	6	6	3	3	0

⇒ At 4, min. of 0 & 6 → 0

⇒ So, 0 - 4 = -1

⇒ whenever you ~~get~~ will get -ve value that means 0 water ~~s~~ trapped ~~in~~ at that building.

Code

```
vector <int> height;
leftmax[n], rightmax[n];
leftmax[0] = 0
for (i = 1; i<n; i++)
    leftmax[i] = max(leftmax[i-1], height[i-1]);
rightmax = (n-1)
for(i = n-2; i >= 0; i--)
    rightmax[i] = max(rightmax[i+1], height[i+1];
int water = 0;
for(i = 0; i<n; i++){
    minheight = ~~max~~ min(leftmax[i],
                rightmax[i]);
```

Date _____ Page _____

```
if(minheight - height[i] > 0)
    water += minheight - height[i];
}
return water;
```

⇒ So, O(N) → T.C
O(N) → S.C

⇒ Can we solve this question ~~in~~ with O(1) → S.C. ?

⇒ To solve this problem, we have to ~~find~~ do something as we will ~~auto~~ know what is the max in left & right.

⇒ First we calculate ~~maxRight~~ max building and their index. So, ~~we~~ now we will know the maxRight to that max building that means we only have to check maxleft & check the water.

⇒ And we will use the same conditions as we use in the above approach.

⇒ ~~An~~ After that max building, we will traverse from ending and here we will calculate max Right & because maxleft is the max building.

Date _____ Page _____

Code

```
maxleft = 0 , maxRight = 0, water;
maxHeight = height[0], index = 0;
for (i=1; i<n; i++){
    if (height[i] > maxheight){
        maxheight = height[i];
        index = i;
    }
for(i=0; i<index; i++){
    if (leftmax > height[i])
        water += leftmaxleft - height[i];
    else{
        maxleft = height[i];
    }
for( i= n-1; i > index; i++){
    if ( maxright > height[i])
        water += maxRight - height[i];
    else
        maxRight = height[i];
}
return water;
```

⇒ If we want to do this in one traversal then with the help of two pointers approach, we can do this.

Date _____ Page _____

* 3 Sum

1	4	45	6	10	8
			3		

X = 13

⇉ Here, we have to find no. that sum is equal to the desired sum.

⇉ In Brute force approach, we will run three loop and find that three no.

⇉ If we find the three no. then we will return 1 otherwise 0.

Code

```
for( i=0; i<n-2; i++){
    for (j= i+1; j<n-1 ; j++){
        for( k= j+1; k<n ; k++){
            if (arr[i] + arr[j] + arr[k] > X)
                return 1;
        }
    }
}
return 0;
```

N^3

T.C → $O(N^3)$

⇉ 2nd Approach

⇉ In this approach, we will use binary search to find the no..

Date _____ Page _____

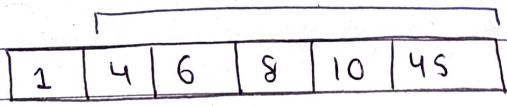
Code

```
for(i=0; i<n-2; i++){
    for(j=j+1; j<n-1; j++){
        find = X - arr[i] - arr[j];
        start = j+1 ; end = n-1;
                Binary search → return1
    }
}
return 0;          T.C → O(N^2 log N)
```

⇉ 3rd Approach

⇉ Here, we will convert our 3 sum problem into 2 sum problem.

⇉ ~~Exa~~ First, we select a no. then minus with X & the remaining no. ⁄ is our desired 2 sum that we will find by 2 sum approach.



→ Here we will apply two sum to find remaining ~~sum~~ two no.

Code

```
for(i=0; i<n-2 ;i++){
    ans = X-arr[i];
    start = i+1 ; end = n-1;
```

Date Page

```
while( start < end){                                    ans
    if (arr[start ] + arr[end ] == target)
        return 1;                                       ans
    else if (arr[start] + arr[end] > target)
        end--;
    else
        start++;
}
return 0;
```

⇒ T.C → $O(N^2)$
S.C → $O(1)$

* 4 sum

1	5	1	0	6	0	X = 7

⇒ First, we convert this to three sum.

⇒ Then use three sum approach in this.