Today's Content

- prefix Sum
- problems on prefix Sum
- LeftMax , RightMax
- Water accumulated
- Max Subarray Problem

Pf[i] = x

[0 ---- i

Prefix Sum ?

(Q1) Given N array elements = 0

$1 \le n \le 10^5$

$1 \le q \le 10^5$

	0	1	2	3	4	5	6
arr[7] =	0	0	0	0	0	0	0
			4	4	4	4	4
				-1	-1	-1	-1
			2	2	2	2	2
						1	1
	2	2	6	5	6	6	6

Q = 4

a	val
2	4
3	-1
0	2
4	1

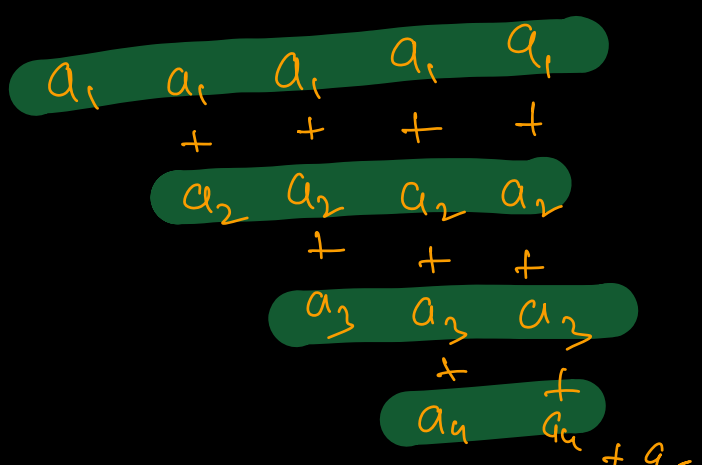
1) for each query iterate array & update

Tc: O(nq)

Prefix Sum

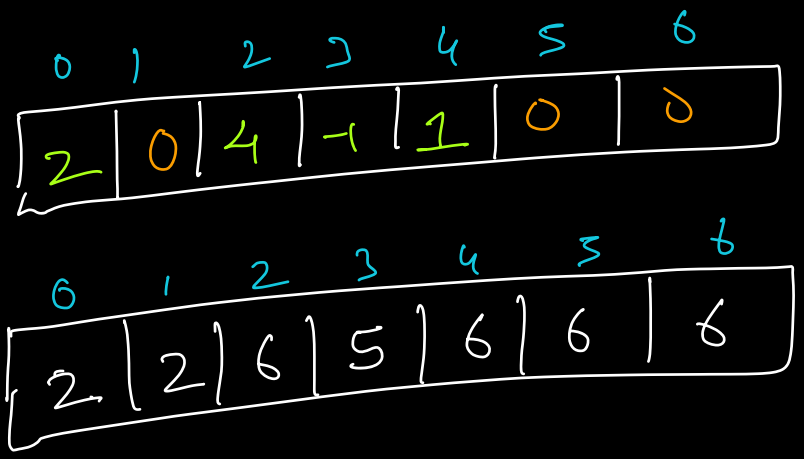
a_1	a_2	a_3	a_4	a_5

pf[7]: $a_1 + a_2 + a_3 + a_4 + a_5$



Q = 4

a	val
2	4
3	-1
0	2
4	1



Tc: O(Q + N)

Sc: O(1)

Pseudo Code

```
while (Q--) {
// a, val
    arr[a] += val
}

for (int i = 1 ; i < n; i++) {
    arr[i] = arr[i] + arr[i-1];
}
```

Input + Extra

Sc

Q Given N array elements = 0

	0	1	2	3	4	5	6	7	8	9
arr[10] =	0	0	0	0	0	0	0	0	0	0
				1	1	1	1			
				3	3	3	3	3	3	
							-3	-3	-3	-3
			2	2	2	2	2	2	2	2
	0	2	5	6	6	3	3	2	-1	2

Q = 4

s	e	val
3	6	1
2	7	3
5	8	-3
1	9	2

Observation

Q

s	e	val		
3	6	1	[3, 6]	[7, 9] → [e+1, n-1]
2	7	3	[2, 7]	[8, 9]

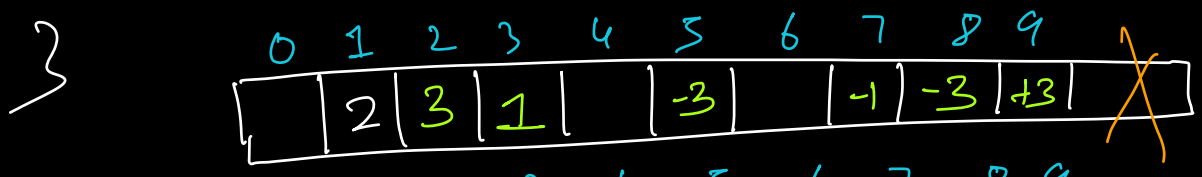
Step 1: arr[s] += val

arr[e+1] -= val

e = n-1

Pseudo Code

```
while (Q>0) {
  // s, e, val
  arr[s] += val;
  if (e != (n-1)) {
    arr[e+1] -= val;
  }
  Q--;
```



Q = 4

s	e	val	
3	6	1	⇒ 3 | 7
2	7	3	⇒ 2 | 8
5	8	-3	⇒ 5 | 9
1	9	2	⇒ 1

[0,1]

Q3 Find Plm[i] = max of all Elements from [0, i]

	0	1	2	3	4
arr =	-5	1	2	3	2

Left max

	0	1	2	3	4
Plm =	-5	1	2	3	3

TC: O(n)

TODO

Q4 Find slm[i] = max of all Elements from [i, n-1]

	0	1	2	3	4
arr =	-5	1	2	3	2

[3, 4]

slm =	3	3	3	3	2

Right Max

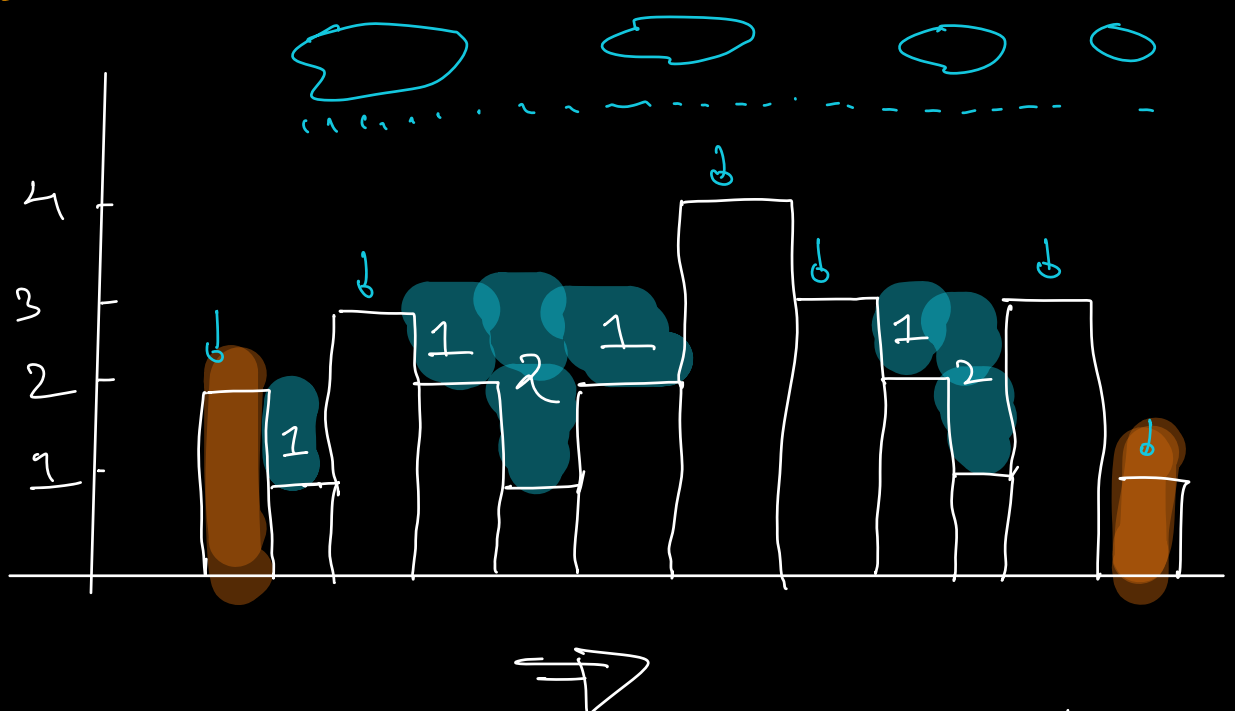
TC: O(n)

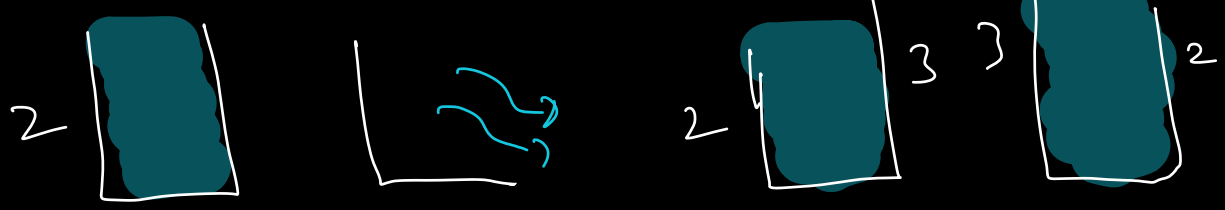
Q 5) Rain water trapped

Given N array elements, where arr[i] represents height of the building.

Return the amount of water trapped between building.

Ex arr[] = {2, 1, 3, 2, 1, 2, 4, 3, 2, 1, 3, 1}

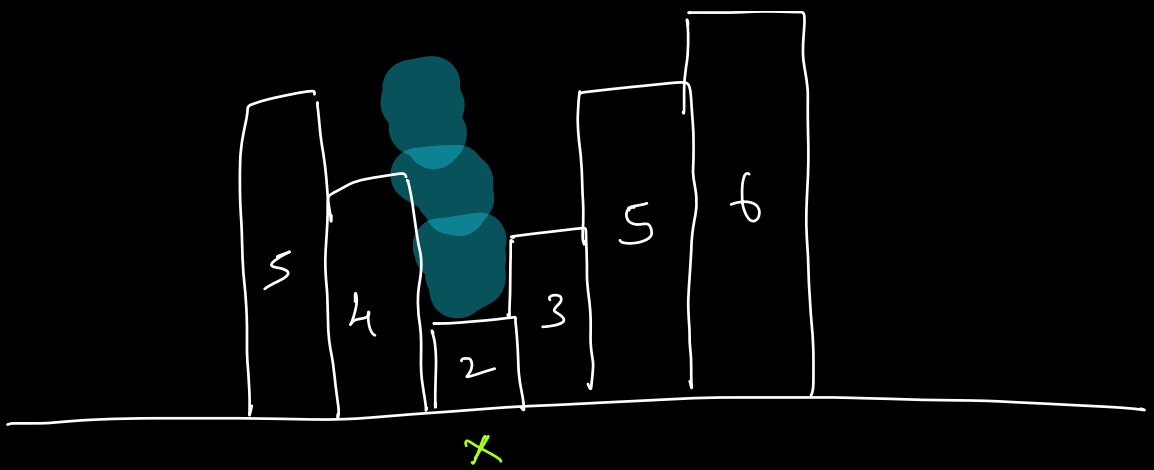




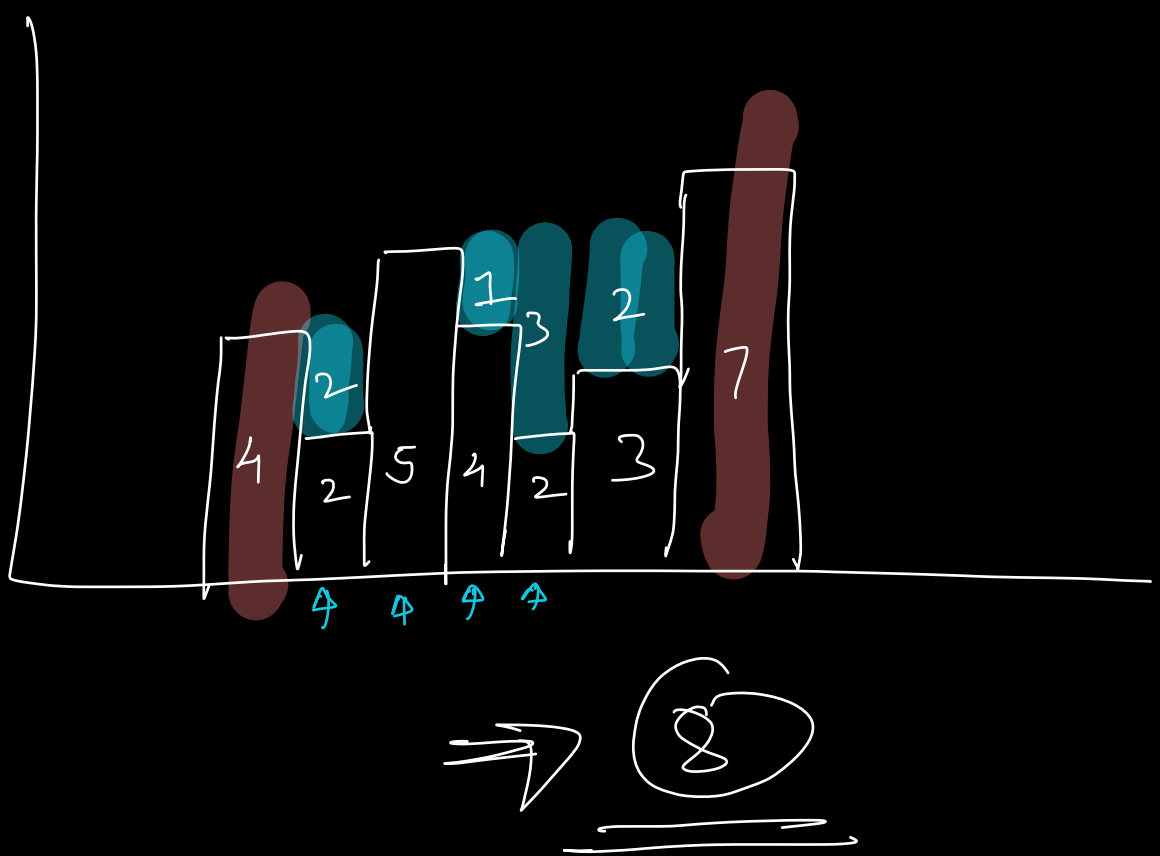
Both ends need support

ans = 2

ans = 2



$$\min\{leftmax, rightmax\} - height$$



$\Rightarrow 8$

Pseudo Code

```
// left_max O(n)
// right_max O(n)
int ans = 0
for (int i = 1 ; i < (n-1); i++){
    int support => min( left_max[i-1] ,
                        right_max[i+1] )
    int water => support - arr[i];
    if (water > 0)
        ans += water;
}
return ans;
```

TC : O(n)

SC : O(n)

10:47

Q6) Given N array Elements, Calculate Maximum Subarray Sum.

Ex arr[] = {-1, 4, 2, 8, -2, 3}

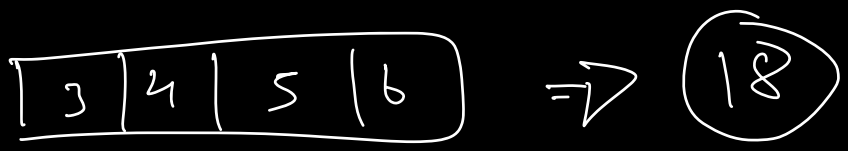
ans = 15

Approach 1 : TC : $O(n^3)$

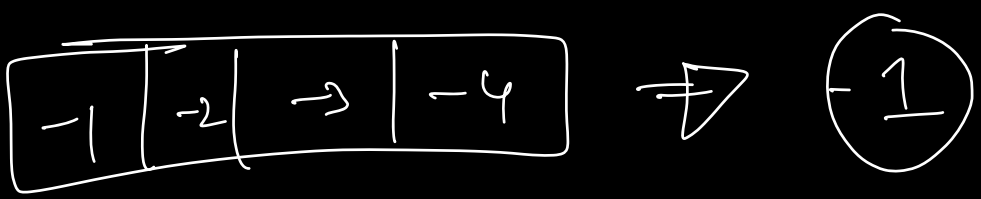
Approach 2 : TC : $O(n^2)$

using prefix sum

Scenario 1 : All element > 0



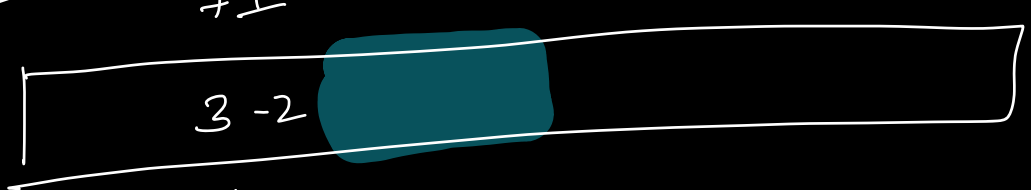
Scenario 2 : All element < 0



Scenario 3

↳ max subarray sum

Scenario 4



If sum > 0, then carry sum

arr[] :	5	6	7	-3	2	-10	-12	8	12	21	-4	7
Sum=0	5	11	18	15	17	7	0	8	20	41	37	44
ans = INT.MIN	5	11	18	18	18	18	18	18	20	41	41	44

```
S = 0
ans = INT.MIN

// find maximum                    O(n)
    if (max < 0)
        return max;

for ( int i = 0 ; i < n ; i++) {    Tc: O(n)
                                    Sc: O(1)
    S = S + arr[i];
    ans = max(ans, S);
    if (S < 0)
        S = 0;
}
```

return ans;

TODO

1) Can we find the indices of max Subarray.

2) Can you handle edge case in a different way.