$$\begin{array}{l|cccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \text{Q1)} & 3 & 2 & -1 & 5 & 6 & 8 & 2 & 3 & 2 & 6 \end{array}$$

Queries: 3
[1, 4]
[3, 6]
[1, 7]
s, e

Idea: prefix sum

```
pf[i]: pf[i-1] + a[i]
pf[0] = a[0]
for (i=1 ; i<n ; i++) {
    pf[i] = pf[i-1] + a[i]
}
```

Answer for each query

```
for (i = 0; i < Q ; i++) {
        read (s, e)   // start & end
// sum [i:j] = pf[e] - pf[s-1]
  if (s == 0)
      ans = pf[e]
else
      ans = pf[e] - pf[s-1]
```

9:05

TC: O(N+Q)   SC: O(N)

Q2 Given N array elements = 0
For every query of the form index, val
add val to all indexes [index : n-1]

Q = 4

idx	val	0	1	2	3	4	5	6
		0	0	0	0	0	0	0
2	4	0	0	4	4	4	4	4
3	-1	0	0	4	3	3	3	3
0	2	2	2	6	5	5	5	5
4	1	2	2	6	5	6	6	6

$Q \Rightarrow$

Brute : Use nested loops to add for each query. TC : $O(nq)$

Idea ar[5]

0	1	2	3	4
a_0	a_1	a_2	a_3	a_4
a_0	a_0	a_0	a_0	a_0
	a_1	a_1	a_1	a_1
		a_2	a_2	a_2
			a_3	a_3
				a_4

$Q = 4$

idx	val	0	1	2	3	4	5	6
		0	0	0	0	0	0	0
2	4	0	0	4	0	0	0	0
3	-1	0	0	4	-1	0	0	0
0	2	2	0	4	-1	0	0	0
4	1	2	0	4	-1	1	0	0
		2	2	6	5	6	6	6

- For every query directly update array
- Now take prefix sum of array.

Code

```
for (i=0; i<Q; i++) {
        read (idx, val)
        ar[idx] += val
}
// Now take pref sum
for (i=1 ; i<n; i++) {
        ar[i] += ar[i-1]
}
```

TC: $O(N+Q)$ SC: $O(1)$

Q3 Given N array elements = 0
For every query of the form s, e, val
add val to all indexes [s : e]

Eg:	0	1	2	3	4	5	6	7	8
	0	0	0	0	0	0	0	0	0

3,6,1	0	0	0	1	1	1	1	0	0
$\rightarrow$ 3,1				1	1	1	1	1	1
7,-1								-1	-1

s e val	0	1	2	3	4	5	6	7	8
1,5,6		6	6	6	6	6			
$\rightarrow$		6	6	6	6	6	6	6	6
1, 6							-6	-6	-6
6, -6									

Idea [s,e,val] is same as
1) [s : n-1] add val
2) [e+1, n-1] add -val

```
for ( i=0 ; i<Q ; i++ ) {
        read (s, e, val)
        ar[s] += val
        if ( e != n-1 )
            ar[e+1] += -val
}
```

Step 2 : Take prefix sum,

TC: $O(N+Q)$ SC: $O(1)$

	0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0
	0	0	0	-1	0	0	0	1
pf:	0	0	0	-1	-1	-1	-1	0

3, 6, -1

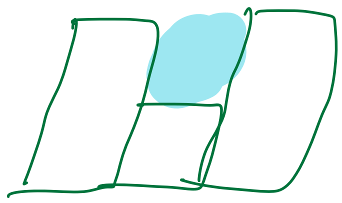
lmax[i] = max (a[i], lmax[i-1])

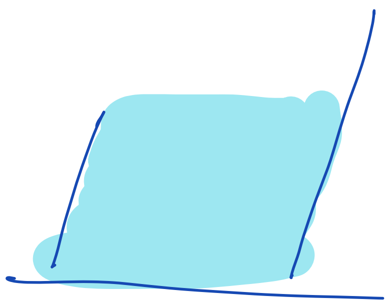
7 10 2 5 20
7 10 10 10 20

Previously studied
Leftmax & Rightmin
(from carry-fwd)

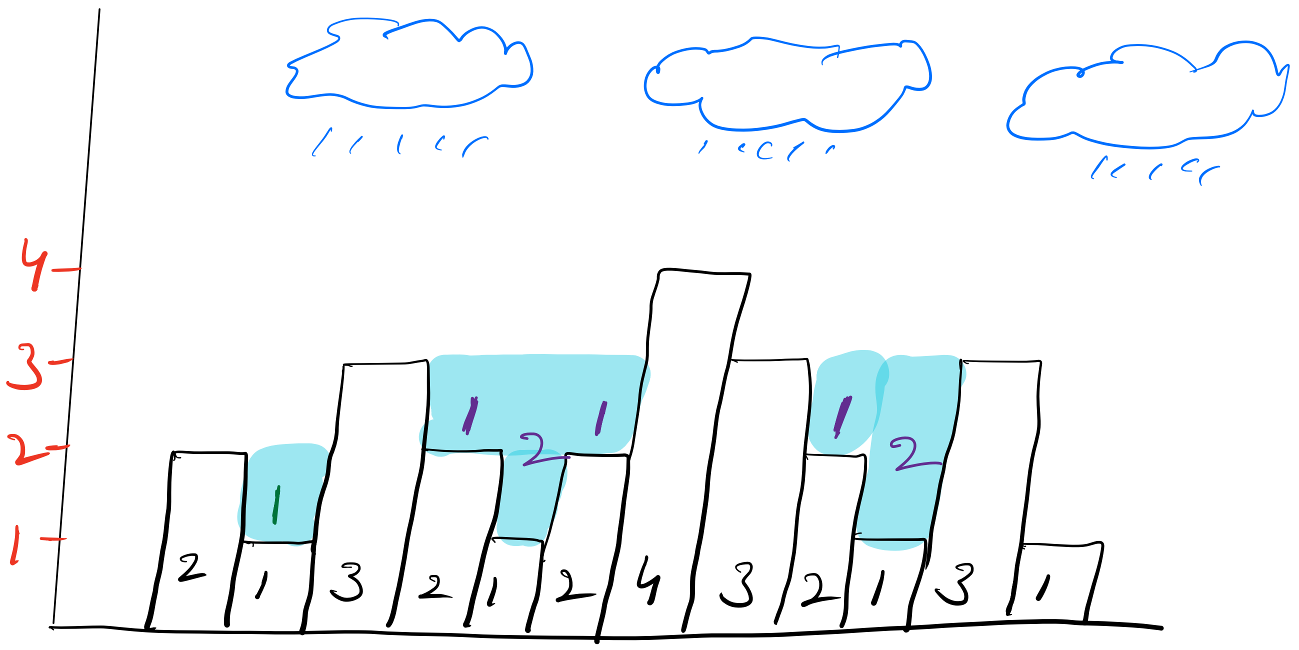
Requirements:
Leftmax & Rightmax TODO

Q4 Rain water trapped
Given array of size N, ar[i]
represents height of i^{th} building
Assume that it rains (A LOT)
Return amount of water trapped.





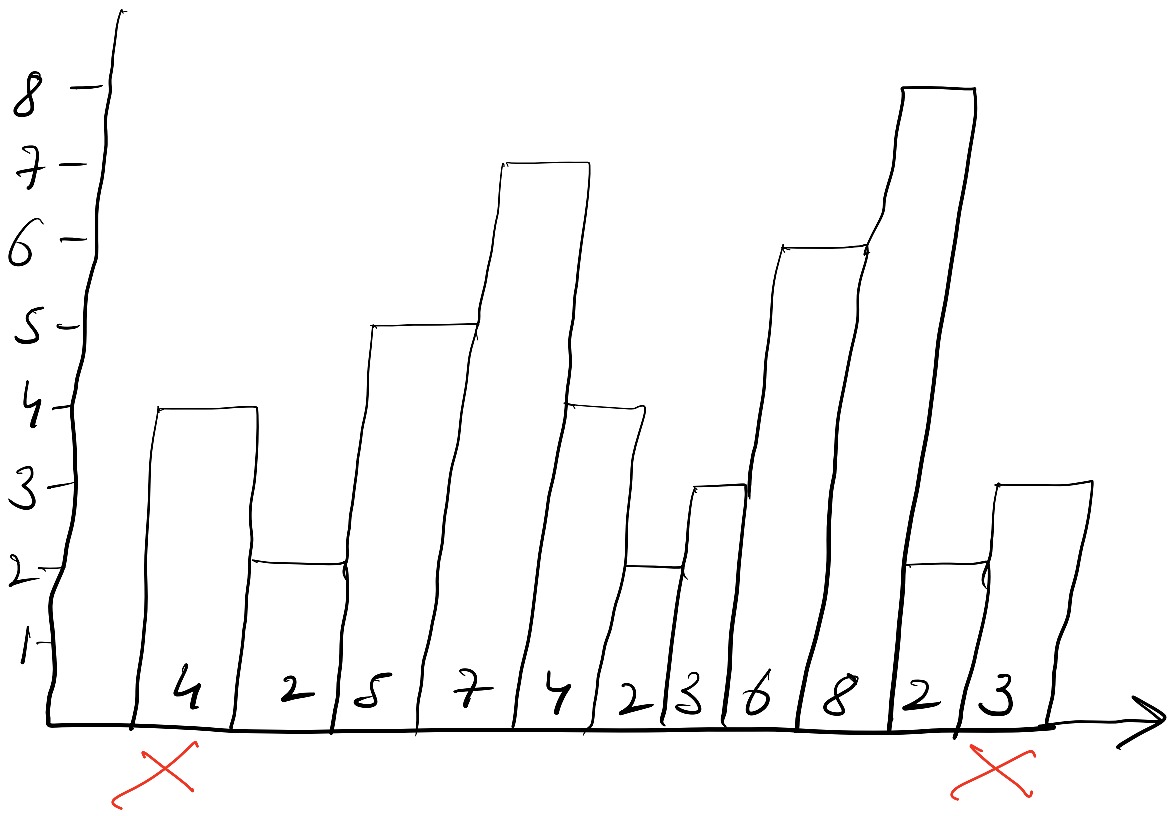
Eg: { 2, 1, 3, 2, 1, 2, 4, 3, 2, 1, 3, 1 }



total amt = 8

idea Calc the amount of water trapped on top of each building

net_sup = min(left_sup , right_sup)
left_support = leftmax [i-1]
right_support = rightmax [i+1]



L	4	4	5	7	7	7	7	7	8
R	8	8	8	8	8	8	8	3	3
Net_sup	4	4	5	7	7	7	7	3	3
W	2	0	0	3	5	4	1	0	1

= 16

Code

because endpoints =0 water

```
ans = 0
for (i=1 ; i<n-1 ; i++) {
        S_L = leftmax[i-1]
        S_R = rightmax[i+1]
        net_sup = min(S_L, S_R)
        W = max(net_sup - a(i), 0)
        ans += w
}
```

TC: $O(N)$ SC: $O(N)$

VV Imp

Q5 Max subarray sum

0	1	2	3	4	5	6
-3	2	4	-1	3	-4	3

Eg: -3, 2, 4, -1, 3, -4, 3 ans = 8

Brute: check for all subarrays
TC: $O(n^2)$

Kadane's Algorithm.

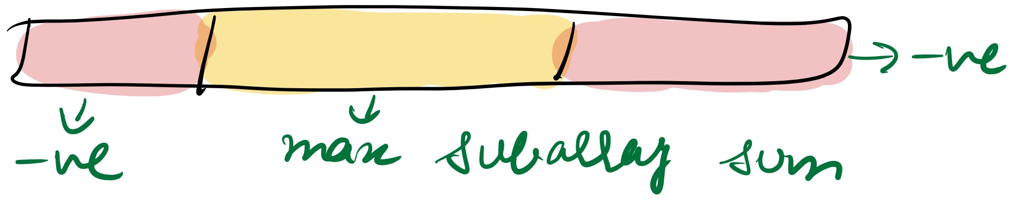
Case 1 All elem ≥ 0 whole array

3	2	1	6

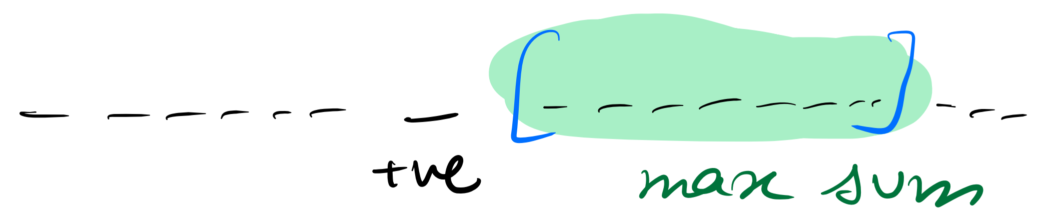
Case 2 All elem < 0 max of the array

-8	-4	-2	-10

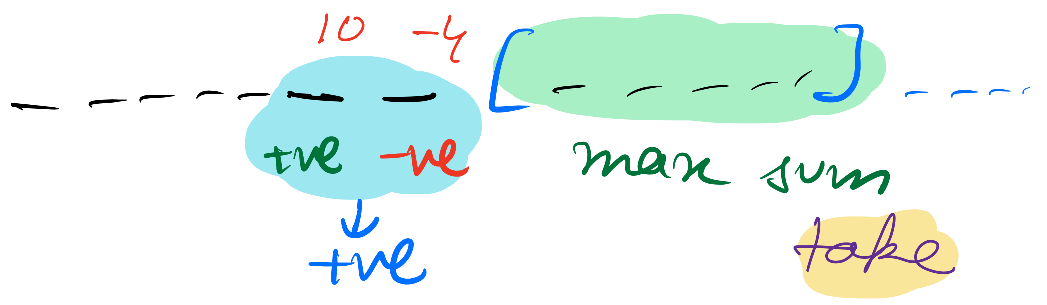
Case 3



Case 4



Case 5



If sum > 0, ⇒ we will take this sum

ar	5	6	7	-3	2	-10	-12	8
sum=0	5	11	18	15	17	7	~~-5~~ 0	8
ans = INT_MIN	5	11	18	18	18	18	18	18

Code

```
sum = 0
ans = INT_MIN
for ( i=0 ; i<n ; i++ ){
    sum = sum + a[i]
    ans = max (ans, sum)
    if (sum < 0)
        sum = 0
}
return ans
```

TC: O(n)   SC: O(1)

{done}

	0	1	2
	−2	−3	−1
s	~~−2~~ 0	~~−3~~ 0	~~−1~~ 0
a	−2	−2	−1