	0	1	2	3	4	5	6	7	8	9
Q1)	3	2	-1	5	6	8	2	3	2	6

Queries: 3

[1, 4]

[3, 6]

[1, 7]

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] = ?
  if (s==0)
      ans =   pf[e]
  else
      ans =   pf[e] - pf[s-1]
```

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

⇒

Brute : Use nested loops to add for each query. TC : O(N*Q)

Idea arr[5]

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		0	1	2	3	4	5	6
idx	val	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	2 ~~0~~	~~4~~	~~-1~~	~~1~~	0	0
				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)
        a[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]

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

	0	1	2	3	4	5	6	7	8
1, 5, 6		6	6	6	6	6	+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+1  < n )  {
              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
	3				-3		
0	3	3	3	3	0	0	0

1:4 3

[1, 4, 3] ⇒ [1, 7, 3] + [5, 7, -3]

Previously studied

Leftmax & Rightmin

(from carry-fwd)

Requirements:

Leftmax & Rightmax

7	3	2	8	10
7	7	7	8	10
10	10	10	10	10

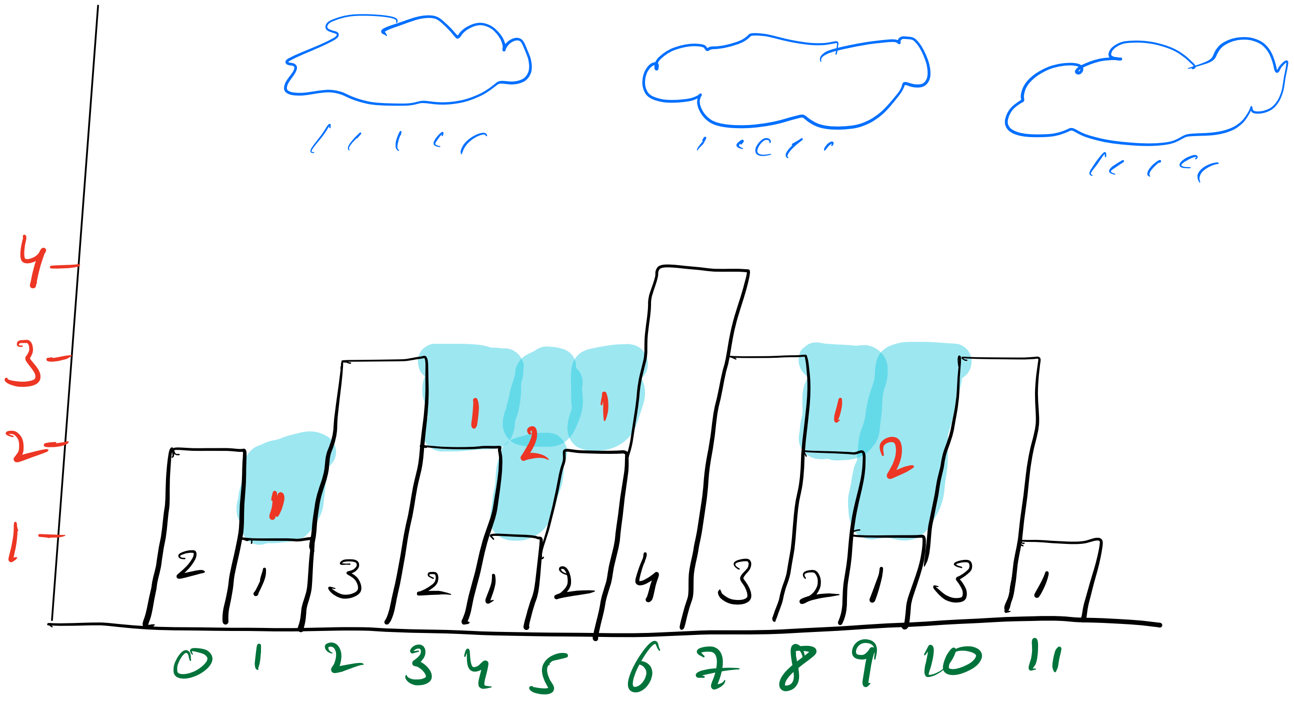
Q4 Rain water trapped

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 }

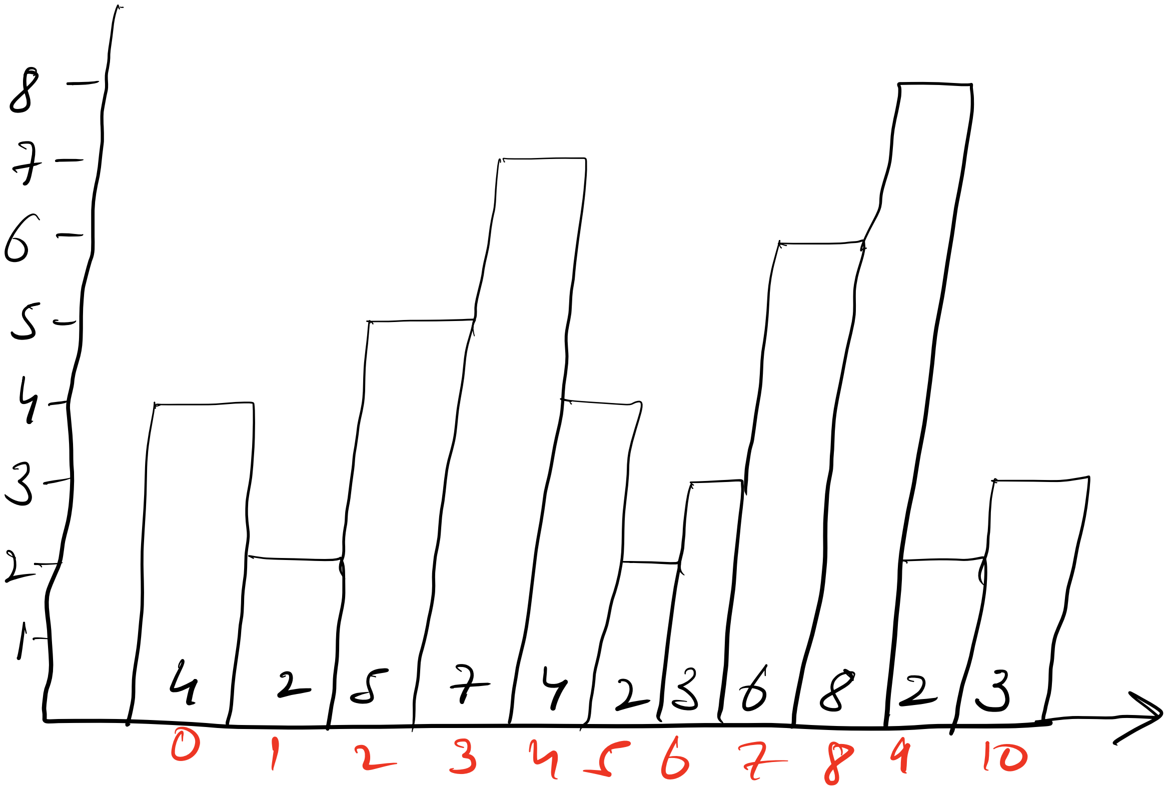


total amt = 8

idea Calc the amount of water trapped on top of each building net_sup = min(left_sup , right_sup)

left_sup = leftmax [i-1]

right_sup = rightmax [i+1]



L	4	4	5	7	7	7	7	7	8	8	8
R	8	8	8	8	8	8	8	8	8	3	3
NS	✗	4	4	5	7	7	7	7	3	3	✗
W		2	0	0	3	5	4	1	0	1	

tot = 16

Code

```
lmax[0] = ar[0]
for ( i=1 ; i<n ; i++){
    lmax[i]=max(ar[i],
                lmax[i-1])
}
```

because ?

```
ans = 0
for (i=1 ; i<n-1 ; i++){
    L sup  =  leftmax [i-1]
    R sup  =  rightmax [i+1]
    NS =  min ( L sup, R sup )
    W = max (NS - ar[i] , 0 )
    ans += W
}
```

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

4 months → 3-3.5 month

removed ✗

moved ⇒ CP elective

Q5 Max subarray sum

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

3	2	1	6

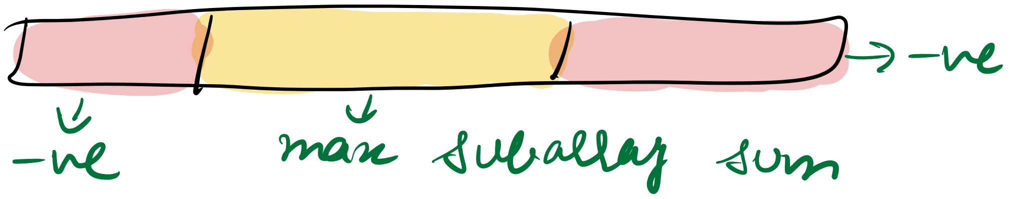
entire array

Case 2   All elem < 0

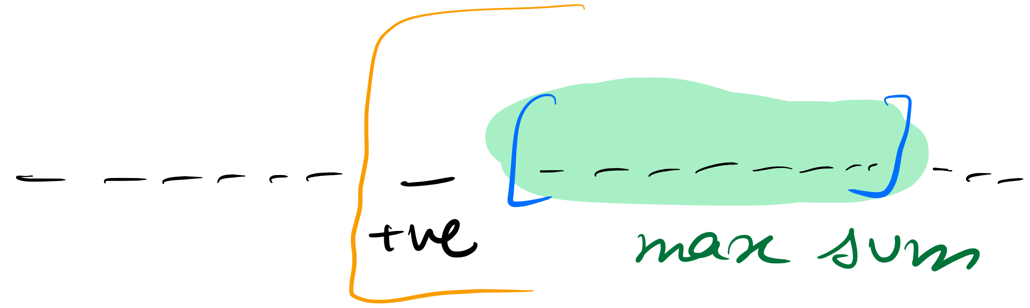
-8	-4	-2	-10

max of the array

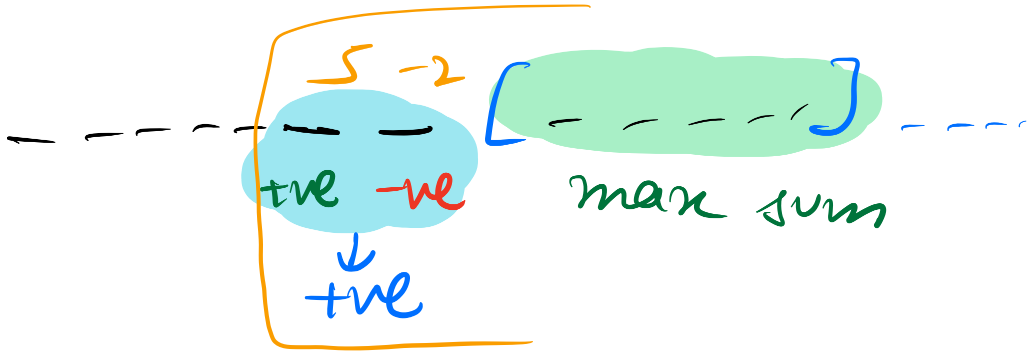
Case 3



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)

	-3	-1	-4
S	~~-3~~ 0	~~-1~~ 0	~~-4~~ 0
ans	-3	-1	-1

{done}