Q1. Find the maxm subarray sum of an array.

ex ⇒ $[-1 \;\; 2 \;\; 3 \;\; -4 \;\; 6 \;\; 9 \;\; 2 \;\; -1 \;\; 8 \;\; 3]$

$\boxed{1}$ → 27

$-1 \;\; 2 \;\; 3 \Rightarrow 4$

$2 \;\; 3 \;\; -4 \;\; 6 \;\; 9 \;\; 2 \Rightarrow 18$

$6 \;\; 9 \;\; 2 \;\; -1 \;\; 8 \;\; 3 \Rightarrow 27$

$2 \;\; 3 \;\; -4 \;\; 6 \;\; 9 \;\; 2 \;\; -1 \;\; 8 \;\; 3 \Rightarrow 28.$ ←

maxm ⇒ 28 ans

total no. of subarrays ⇒ $\frac{N(N+1)}{2}$

bruteforce

i) Consider all possible subarrays

↓

i) all subarrays → N^2

ii) sum → N

TC ⇒ $O(N^3)$

ii) use prefix sum / carry forward technique

```
for ( i=0; i<N; i++){
        sum=0;
        for (j=i; j<N; j++) {
            sum = sum + arr[j]
            ans = max ( ans, sum)
        }
}
```

TC => O(N2)
SC => O(1)

all elements are positive :-

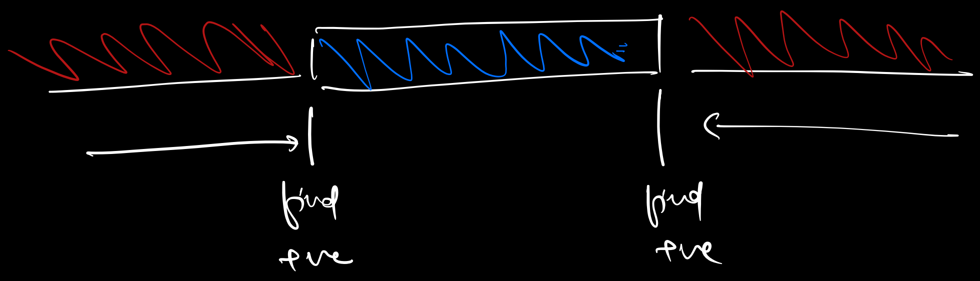
ans => sum of all elements [complete array]

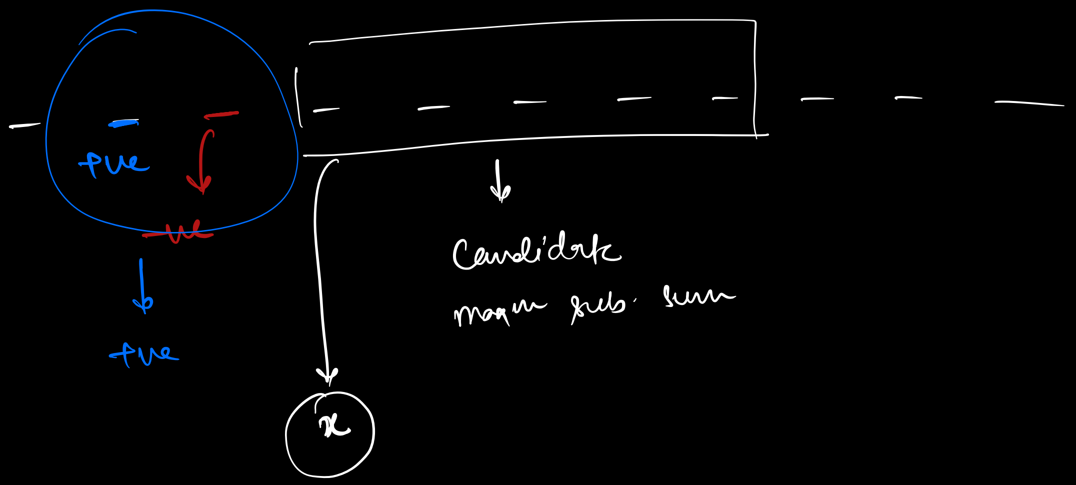
all elements are negative :-

-3 -7 -11 -1 -6

maxm => o/p => max of all in array

negative elements are on boundaries :-





	5	6	7	-3	2	-10	-12	8	12	21	-4	7
sum = 0	5	11	18	15	17	7	~~-5~~ 0	8	20	41	37	44
ans = INT MIN	5	11	18	18	18	18	18	18	20	41	41	44

ans = 44 → O/P

Kadane's Algorithm

arr →	-20	10	-12	6	5	-3	8	9
sum = 0	~~-20~~ 0	10	~~-2~~ 0	6	11	8	16	25
ans = INT MIN	-20	10	10	10	11	11	16	25

ans = 25

arr =>	-6	-10	-4	-5	-2
sum = 0	~~-6~~ → 0	~~-10~~ → 0	~~-4~~ → 0	~~-5~~ → 0	-2
ans = INT MIN	-6	-6	-4	-4	-2

O/p = -2

pseudo

```
sum = 0 , ans = INT MIN
for(i = 0; i < N; i++){
    sum = sum + arr[i]
    ans = max( ans, sum)
    if( sum < 0)
        sum = 0
}
```

TC => O(N)

SC => O(1)

if subarray is required

left , right

↓ update when sum < 0

↓ update when sum > ans

or update when sum < 0

Q2. Beggars outside temple :-

Given arr[N], all elements are zero, and given Q queries each query contains 'idx' & 'value'. Add the value to all indexes from 'idx' to end. Return the final arr after all queries are processed.

idx	value
1	3
4	2
2	1
1	-1

	0	1	2	3	4	5	6	
arr =>	0	0	0	0	0	0	0	
	0	3	3	3	3	3	3	
	0	3	3	3	5	5	5	
	0	3	4	4	6	6	6	
	0	2	3	3	5	5	5	← final arr

o/p

Brute force

for every query update the array :-

TC => O(N×Q)

array => 0 4 5 1 2

pfsum => 0 4 9 10 12

+3 +3

↘ array ⇒ 3 4 5 1 2

pfsum ⇒ 3 $\boxed{7}$ $\boxed{12}$ 13 15

array : 0 4 5 1 2

Q pfs 3 7 12 13 15.

idx	val
0	3

$pf[i] = \underline{pf[i-1]} + arr[i]$

eg ⇒ arr ⇒ 3 4 5 0 2

pfsum ⇒ 3 7 12 $\boxed{12}$ $\boxed{14}$

+6 +6

arr ⇒ 3 4 5 6 2

pfsum ⇒ 3 7 12 18 20

i) add the value to the idx

ii) propagate the value to end }

pfsum

3 0 0 0 0

prfsum => 3 3 3 3 3

0 1 2 3 4

0 0 ~~0~~ ~~0~~ 0

0 0 5 5 5

Q

ind val

2 5

0 0 ~~0~~ 0 0

0 0 5 0 0

prfsum => 0 0 5 5 5

0 1 2 3 4 5

0 0 0 0 0 0

0 5 5 5 5 5

Q

idx	val
1	5

0 5 0 0 0 0

change val at idx

0 5 5 5 5 5

prefsum =>

Q

idx	value		0	1	2	3	4
			0	0	0	0	0
2	3	→	0	0	3	3	3
1	4	→	0	4	7	7	7
1	2	→	0	6	9	9	9

Q

idx	value		0	1	2	3	4
			0	0	0	0	0
2	3	→	0	0	3	0	0
1	4	→	0	4	3	0	0
1	2	→	0	6	3	0	0

pfsum ⇒ 0 6 9 9 9

pseudo

```
for every query {                    → Q
    arr[i] = arr[i] + value
}
prefix sum                           → N
```

$\boxed{TC \Rightarrow O(N+Q)}$

⇒ Q queries ⇒ { start, end, val }

Q

start	end	value
2	4	2
1	3	1
0	2	3
3	5	4

0	1	2	3	4	5
0	0	0	0	0	0
0	0	2	2	2	0
0	1	3	3	2	0
3	4	6	3	2	0
3	4	6	7	6	4

Q

1	3	5

✓

	0	1	2	3	4
arr ⇒	0	0	0	0	0
	0	5	0	0	-5
pf	0	5	5	5	0

Q

s	e	v
1	8	3

	0	1	2	3	4	5	6
arr ⇒	0	0	0	0	0	0	0
	0	3	0	0	0	-3	0
pf ⇒	0	3	3	3	3	0	0

i) update start = start + value

ii) update (end+1) → + (-value)

iii) pfSum

pseudo

```
for( i=0; i<Q; i++){
    arr[start] = arr[start] + value;
    if(end < N-1){
        arr[end+1] = arr[end+1] - value;
    }
}
for( i=1 -> N-1){
    arr[i]= arr[i-1] + arr[i] ;
}
}
```

level-3 => array is not all zeroes, it contains some value initially

ea =>

0 1 2 3 4 5

2 4 3 1 0 6

+2 −2

Q

s	e	v
2	4	2

2 4 5 3 2 6

chang value => 2 4 5 1 0 4

p sum => 2 6 11 12 12 16

ex ⇒

q

s	e	v
2 ✓	4 ✓	2

$$\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
2 & 4 & 3 & 1 & 0 & 6 \\
 & & +2\downarrow & +2\downarrow & +2\downarrow & \\
2 & 4 & 5 & 3 & 2 & 6
\end{array}$$

$$\begin{array}{lcccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
 & 2 & 4 & 3 & 1 & 0 & 6 \\
 & & & \downarrow +2 & & & \downarrow -2 \\
\text{values} \Rightarrow & 2 & 4 & \boxed{5} & 1 & 0 & 4 \\
 & & & \hookrightarrow & 6 & 6 & 10
\end{array}$$

updating values + prefix

i) Create an arr[N] with all zeros

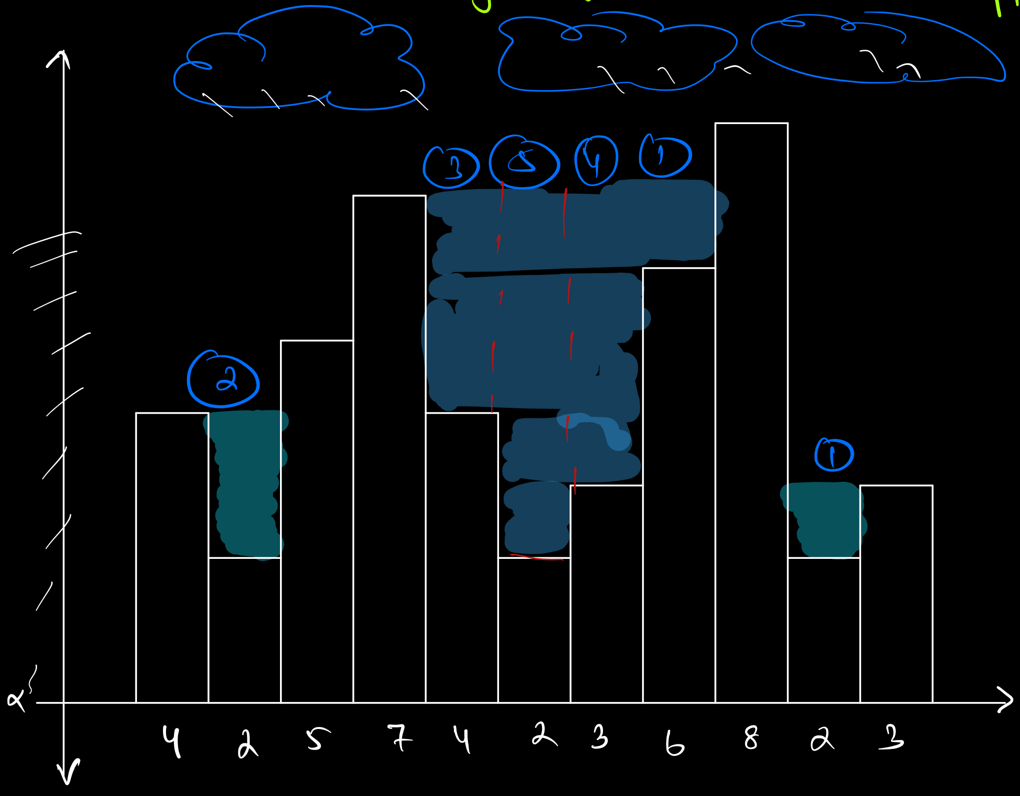
ii) solve for queries on the array with 0's

iii) add the computed arr with given arr.

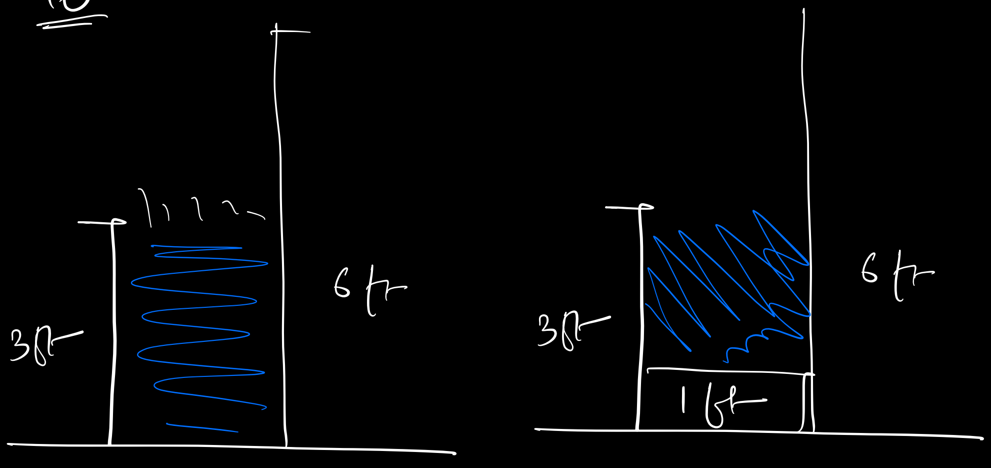
$$\boxed{\begin{array}{l} TC \Rightarrow O(N+Q) \\ SC \Rightarrow O(N) \end{array}}$$

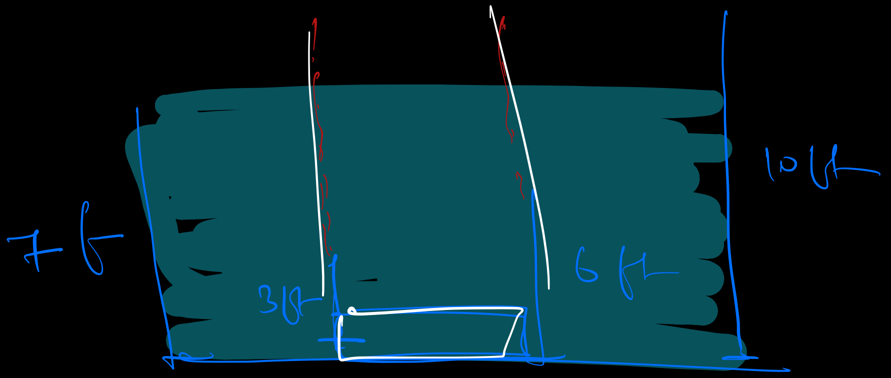
Q3. Rain Water Trapping —

Given an array[N], which contains heights of building with width = 1 (each building), find rain water trapped.



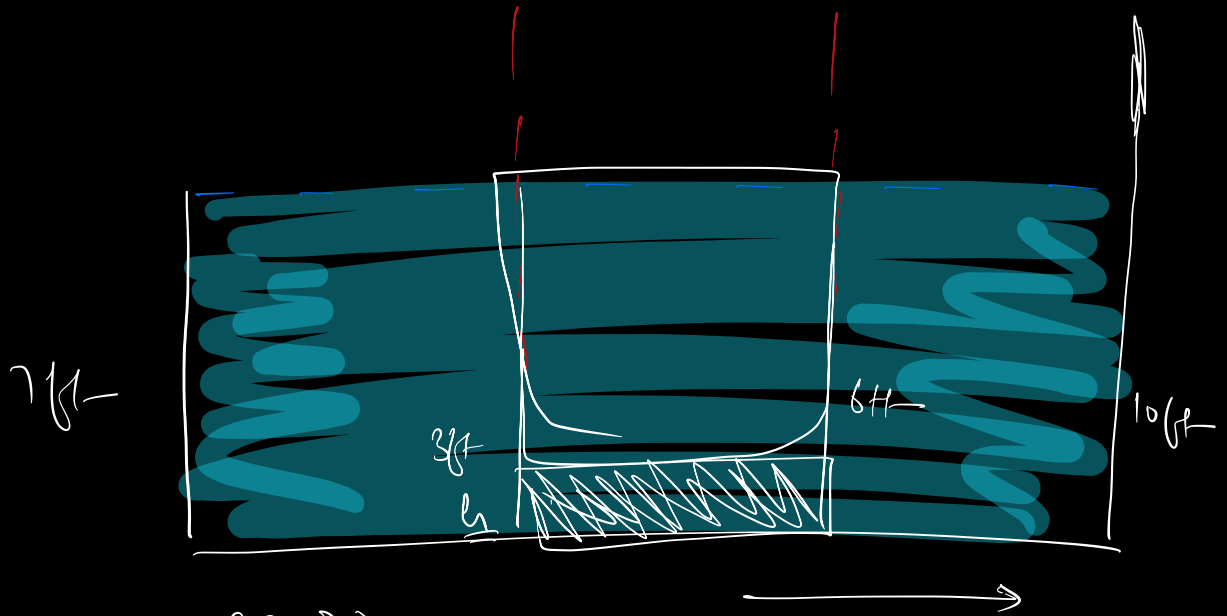
total = 16





7 ft $\longleftarrow$ $\longrightarrow$ 10 ft

$\boxed{\min(\text{leftMax}, \text{rightMax}) - h}$



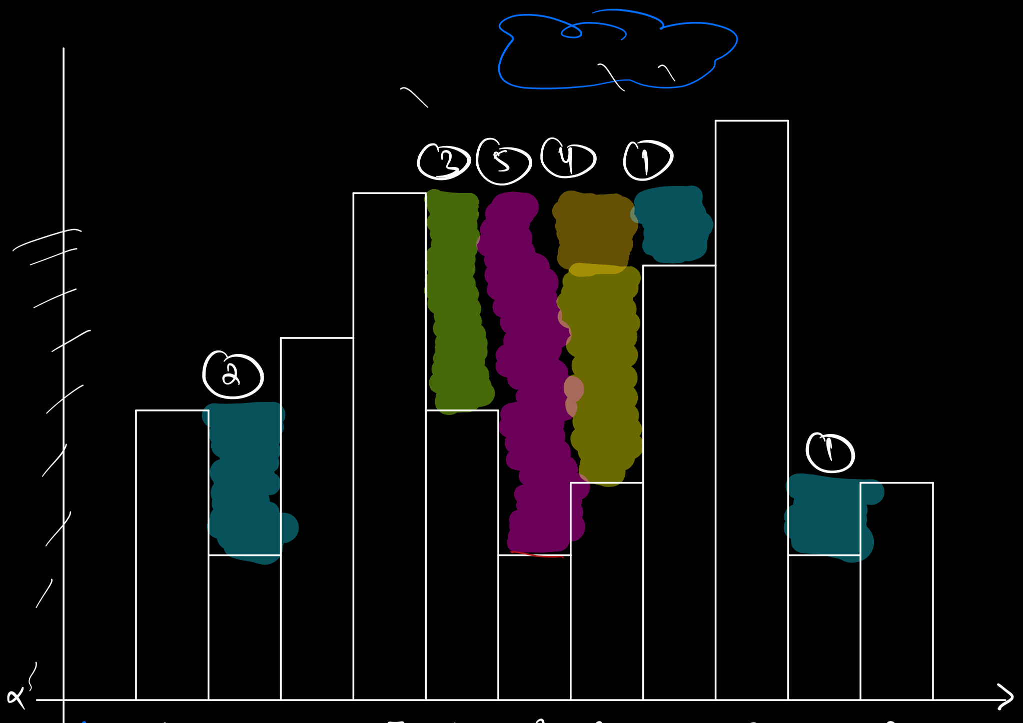
(3, 7) → 7

(6, 10) → 10

$\min(\text{leftMax}, \text{rightMax}) - h$

water for any building = min(max h. left, max h. right)

— height of building.



i/p →	4	2	5	7	4	2	3	6	8	2	3
left max ⇒	0	4	4	5	7	7	7	7	7	8	8
right max ⇒	8	8	8	8	8	8	8	8	3	3	0
min (l, r) ⇒	0	4	4	5	7	7	7	7	3	3	0
	x	2	x	x	3	5	4	1	0	1	0

min-h ⇒ ⇒ 16

i) find leftMax

ii) find rightMax

iii) arr[i]

$\underline{\text{water}}$ = min (leftMax[i], rightMax[i] − h

$\downarrow$

$\underline{\geq 0}$

TC ⇒ O(3N) ≈ O(N)

SC ⇒ O(2N) ≈ O(N) → O(1)

Two ptrs