Q1. Max^m subarray sum of a given array.

eg ⇒ arr ⇒ -1 2 3 -4 6 9 2 -1 8 3

↓
27

Subarr ⇒ -1 2 3 ⇒ 4

2 3 -4 6 9 2 ⇒ 18

6 9 2 -1 8 3 ⇒ 27

2 3 -4 6 9 2 -1 8 3 ⇒ (28) → O/P

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

Brute force

1) Consider all subarrays

↓

i) all subarrays → N^2

ii) sum → N

TC ⇒ $O(N^3)$

ii) use prefix sum / carry forward

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

TC ⇒ $O(N^2)$

all elements in array are positive :-

↓

sum of all elements (whole array)

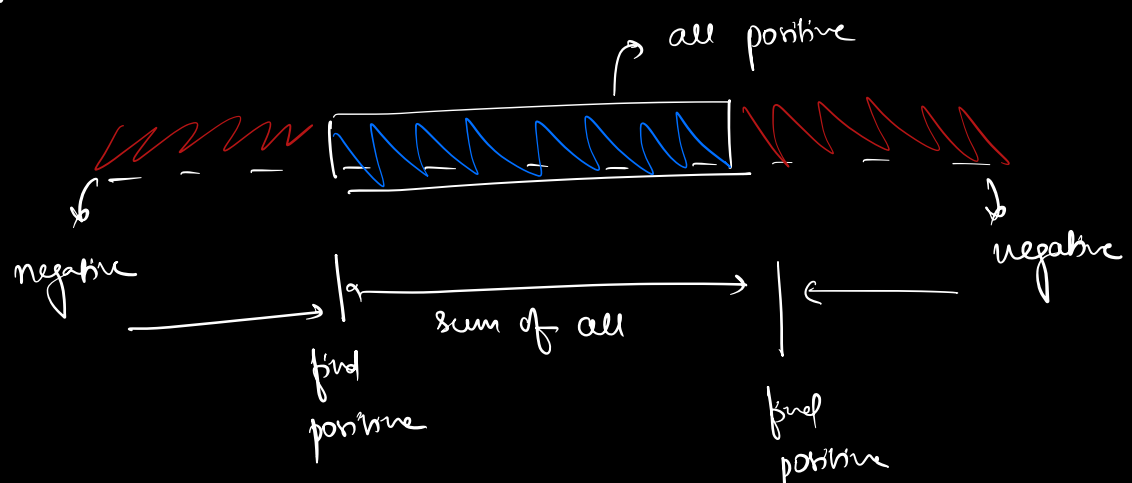
all elements are negative :-

-3 -7 -11 (-1) -6

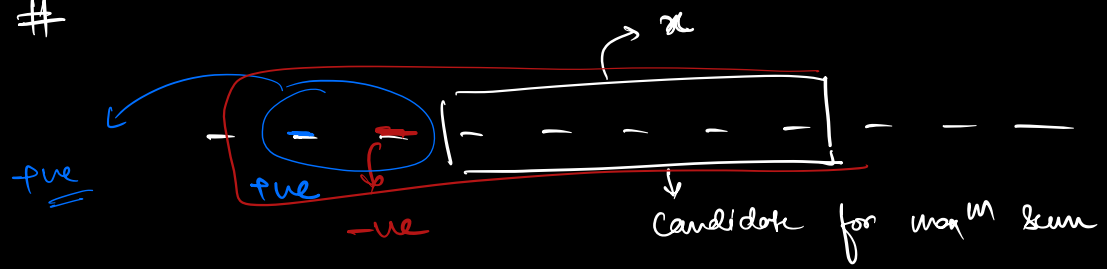
↓

Choose the maxm element

negative elements are on boundaries :-



#



$x_{e1} > x$

	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 Op

	-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

	-12	-6	-1	-5	-4
sum = 0	-12 → 0	-6 → 0	-1 → 0	-5 → 0	-4
ans = INT_MIN	-12	-6	-1	-1	-1

ans = -1

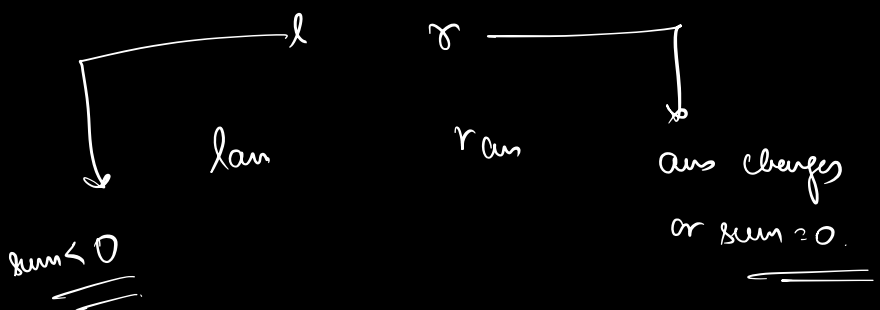
Kadane's Algo

pseudo

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

TC ⇒ O(N)
SC ⇒ O(1)

If subarray is required :- [H.W]



Q2. Beggar's outside temple :-

Given arr[N], all elements are zero. Q queries [idx, value]. Add the 'value' to all indexes from 'idx' till end. Return the final arr

Q	
idx	value
1	3
4	2
2	1
1	-1

0	1	2	3	4	5	6
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

Brute force — for each query, iterate & update the arr

TC = O(N x Q)

arr ⇒ 0 4 5 1 2

Q o/p ⇒ 3 7 8 4 5

0 | 3

arr ⇒ ~~0~~ 3 4 5 1 2

prsum ⇒ 3 7 12 13 15

5

arr ⇒ 4 6 ~~0~~ 1 2

prsum ⇒ 4 10 10 11 13

+5 +5 +5

arr ⇒ 4 6 5 1 2

prsum ⇒ 4 10 15 16 18

arr ⇒ 4 0 2 3 0 1 2

prsum ⇒ 4 4 6 9 9 10 12

+1 +1 +1 +1 +1 +1

+2 +2 +2

arr ⇒ 4 1 2 3 2 1 2

prsum ⇒ 4 5 7 10 12 13 15

Q

idx	value
1	3
4	2
2	1
1	-1

0	1	2	3	4	5	6
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

Q

idx	value
1	3
4	2
2	1
1	-1

PFsum

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

* add the values at each idx for all queries → Q

* take PFsum ⟶ N

TC ⇒ O(N+Q)

SC ⇒ O(1)

↓

Inplace

⇒ level ⇒ 2

Q queries ⇒ { start, end, value }

↓

add value to all indexes from start to end

	0	1	2	3	4	5
	0	0	0	0	0	0

Q

s	e	v
1	3	2
2	4	3
0	2	-1
4	5	6

0	2	2	2	0	0
0	2	5	5	3	0
-1	1	4	5	3	0
-1	1	4	5	9	6

	0	1	2	3	4	5
arr ⇒	0	0	0	0	0	0

s	e	v
1	3	2

0	2	2	2	0	0

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

pfsum →

0	1	2	3	4	5
0	2	2	2	0	0

Q

s	e	v
1	3	2
2	4	3
0	2	-1
4	5	6

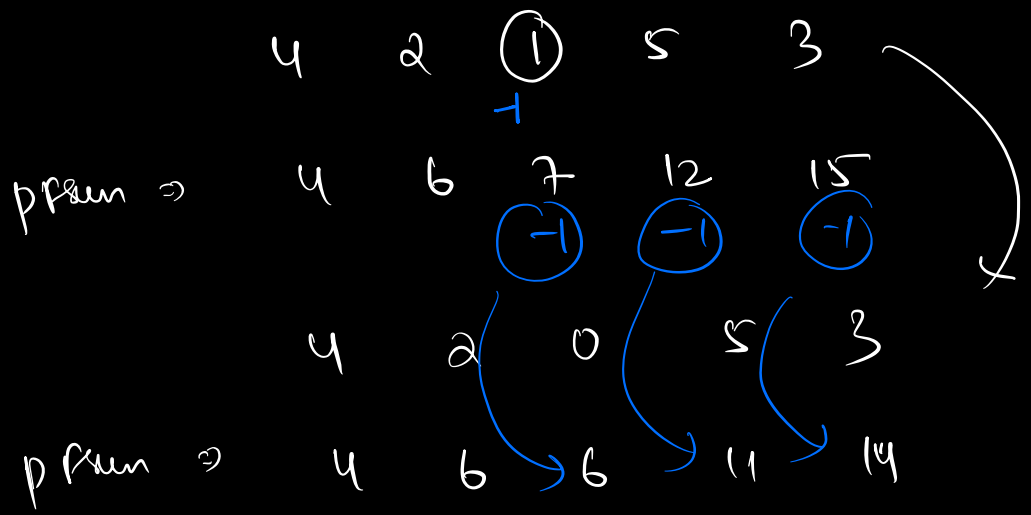
0	1	2	3	4	5
0	0	0	0	0	0
0	2	2	2	0	0
0	2	5	5	3	0
-1	1	4	5	3	0
-1	1	4	5	9	6

Q

s	e	v
1	3	2
2	4	3
0	2	-1
4	5	6

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

pfsum → [-1 1 4 5 9 6]



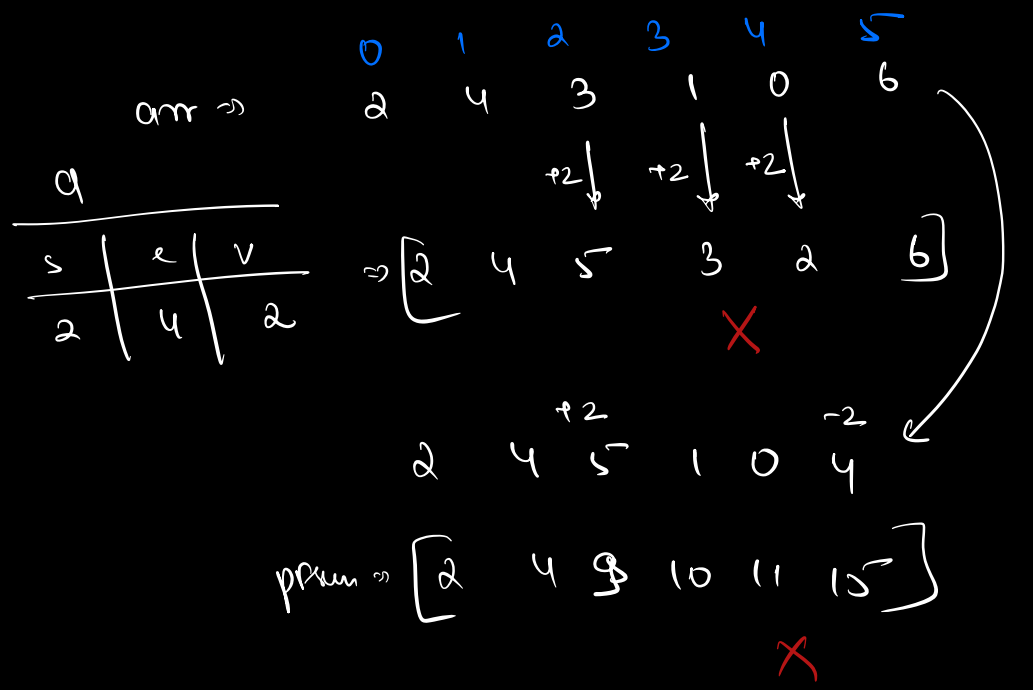
pseudo

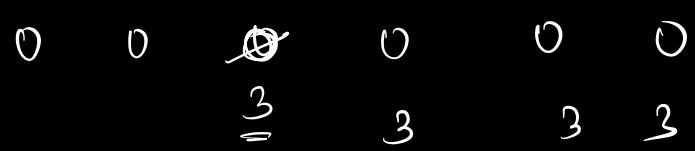
```
for(i=0; i<Q; i++){
    arr[start] = arr[start-1] + value;
    if(end < N-1){
        arr[end+1] = arr[end+1] - value;
    }
}
// prefix sum
```

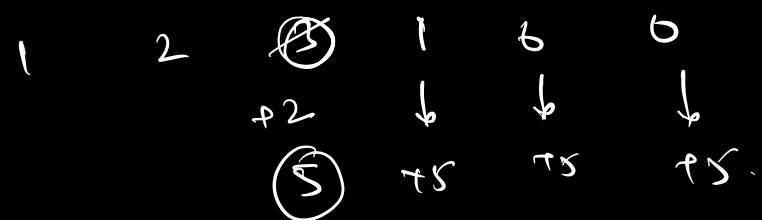
Break ⇒ 7 mins

8:50pm IST

level-3 ⇒ initially the array is not zero.







1) Create a separate arr[N], with zeroes
2) Save the 0's array for given queries → update → pfsum
3) add fresh array with i/p array

0	1	2	3	4	5
0	0	0	0	0	0
	+2			-2	
0	2	0	0	-2	0

pfsum = 0 2 2 2 0 0

s	e	v
1	3	2

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

s	e	v
1	3	2
2	4	3

ans = 1 5 9 7 9 5

0	1	2	3	4	5
0	0	0	0	0	0
	+2			-2	
			+3		-3
0	2	3	0	-2	-3

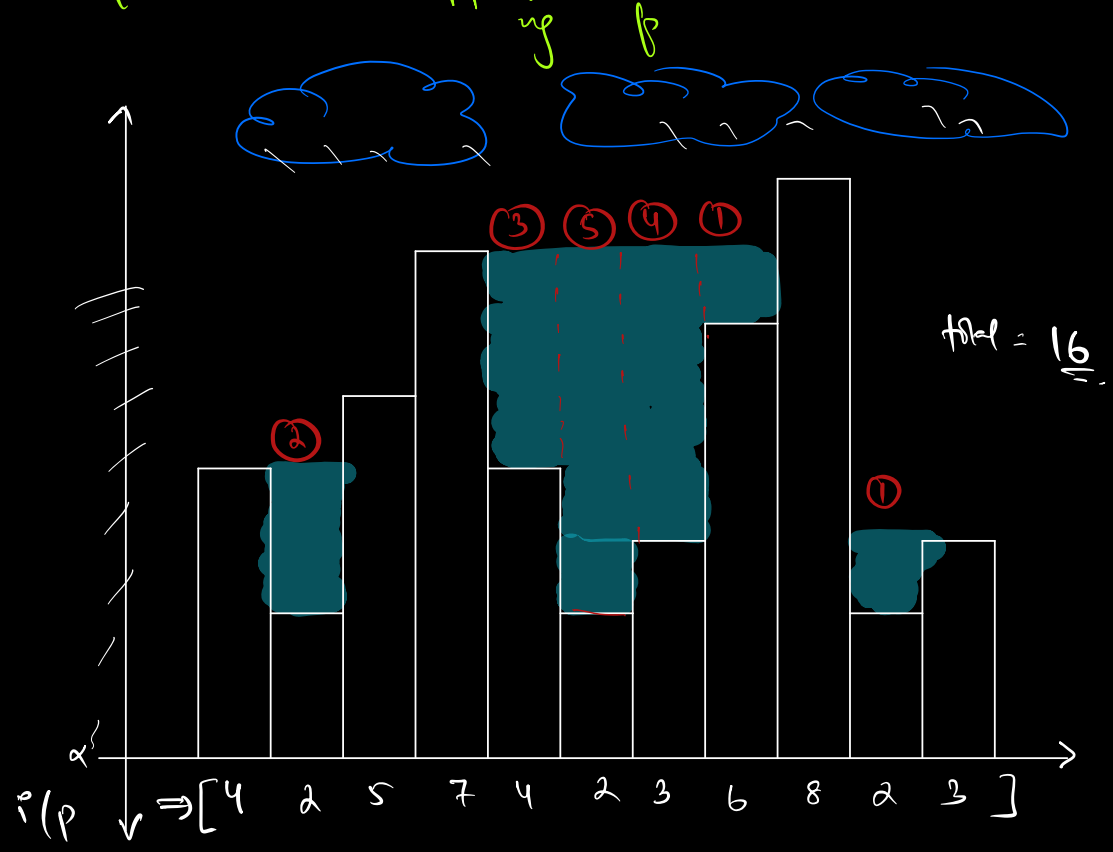
pfsum = 0 2 5 5 3 0

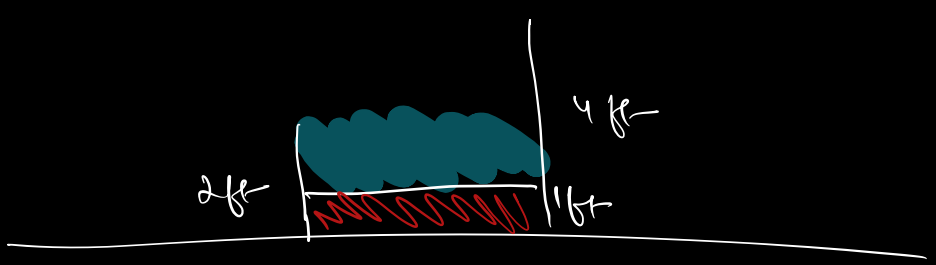
1	3	4	2	6	5
1	5	9	7	9	5

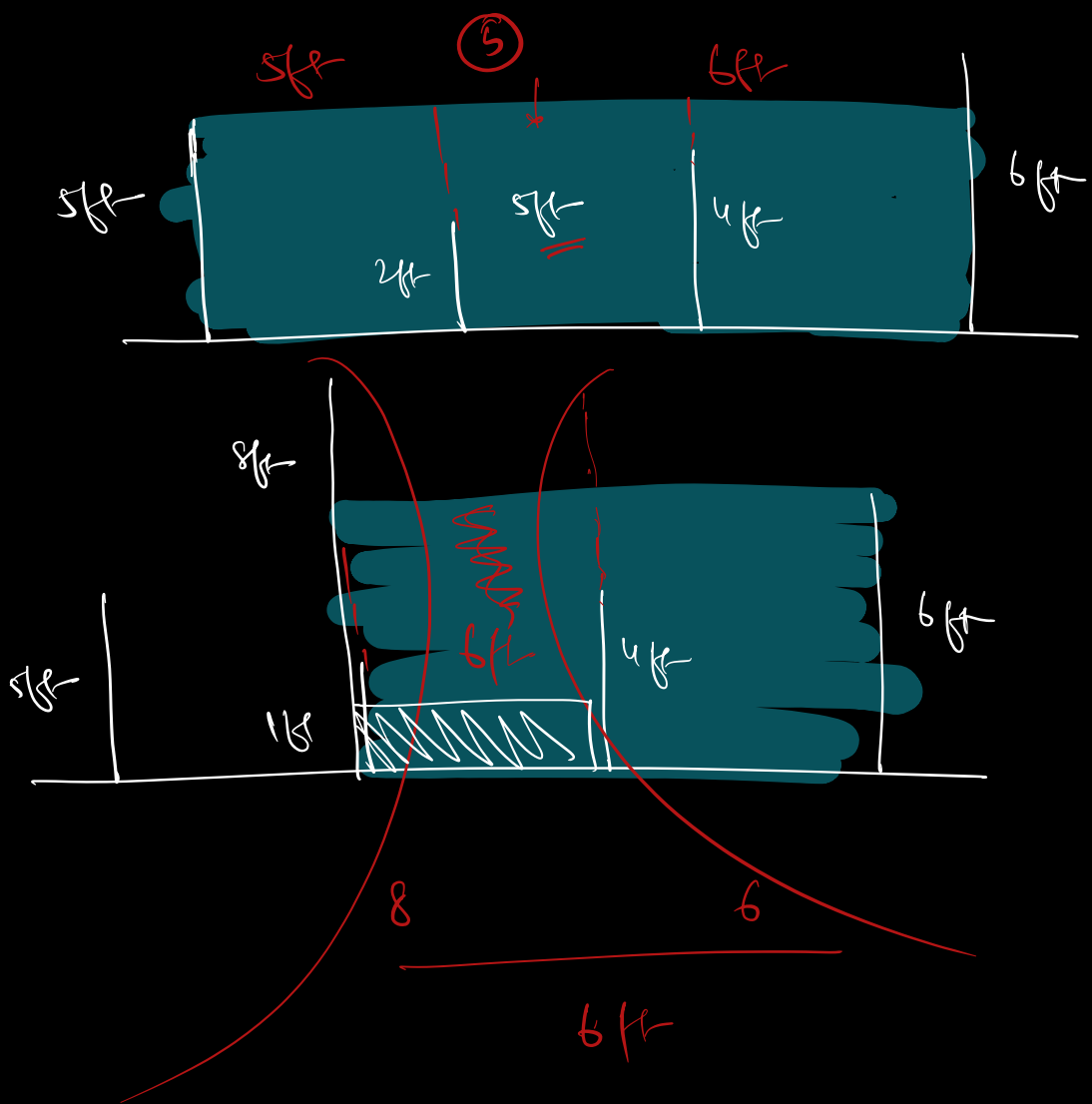
TC = O(N*Q)

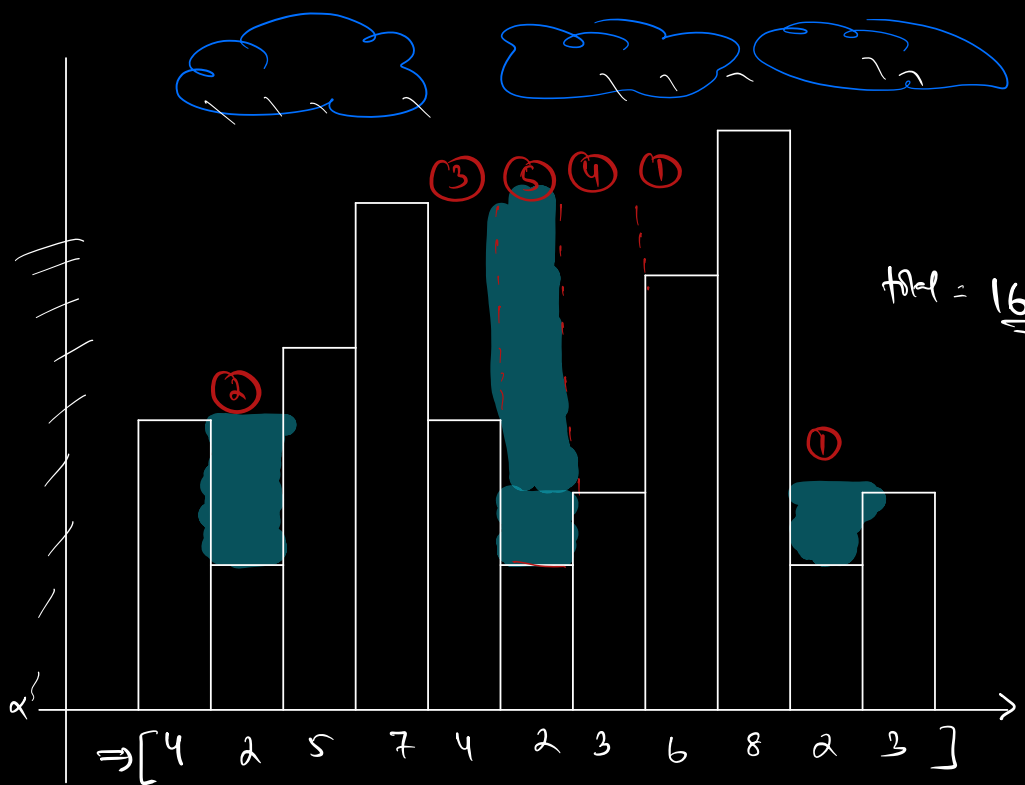
SC = O(N)

Q 3. Rain water trapping :






5ft
5
6ft
5ft
5ft
2ft
4ft
6ft
8ft
6ft
5ft
1ft
4ft
6ft
8
6
6ft



leftMax ⇒	0	4	4	5	7	7	7	7	7	8	8
rightMax ⇒	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
water ⇒		2			3	5	4	1		1	

= 16

- fake leftMax → O(N)
- fake rightMax → O(N)
- water = min(leftMax[i], rightMax[i]) → O(N)
  - h(arr[i])

TC → O(N)

SC → O(N)

O(1) → Two pointers