Intermediate → DSA 1 → DSA 2 ✓ DSA 3 DSA 4

Q → Given an integer array, find the max subarray sum out of all the subarrays.

$A = [\underset{0}{-2} \quad \underset{1}{3} \quad \underset{2}{4} \quad \underset{3}{-1} \quad \underset{4}{5} \quad \underset{5}{-10} \quad \underset{6}{7}]$ Ans = 11

$$\#\text{subarrays} = \frac{N*(N+1)}{2}$$

$\frac{7*8}{2} = 28$

$A = [\underset{0}{4} \quad \underset{1}{5} \quad \underset{2}{2} \quad \underset{3}{1} \quad \underset{4}{6}]$ Ans = 18

if $\forall i\ (A[i] >= 0) \Rightarrow Ans = \sum_i A[i]$

$A = [\underset{0}{-4} \quad \underset{1}{-3} \quad \underset{2}{-6} \quad \underset{3}{-9} \quad \underset{4}{-2}]$ Ans = -2

if $\forall i\ (A[i] <= 0) \Rightarrow Ans = \max_{\forall i}(A[i])$

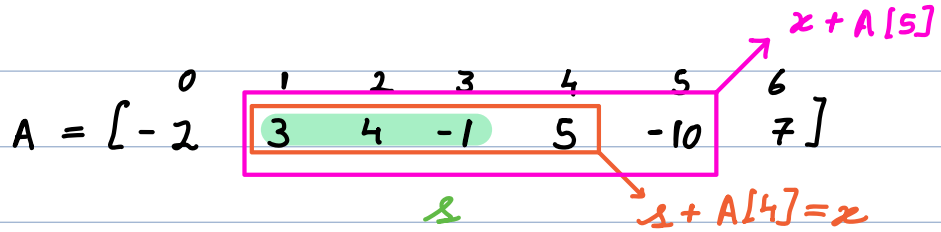
Bruteforce → ∀ subarrays, check the sum & take max.

```
ans = INT_MIN      // ans = A[0]
for l → 0 to (N-1) {
    for r → l to (N-1) {      l — r
        sum = 0
        for i → l to r {      → prefix sum / carry forward
```

```
        sum += A[i]
    }
    ans = max(ans, sum)
}
} return ans
```

TC = $O(N^3)$

SC = $O(1)$

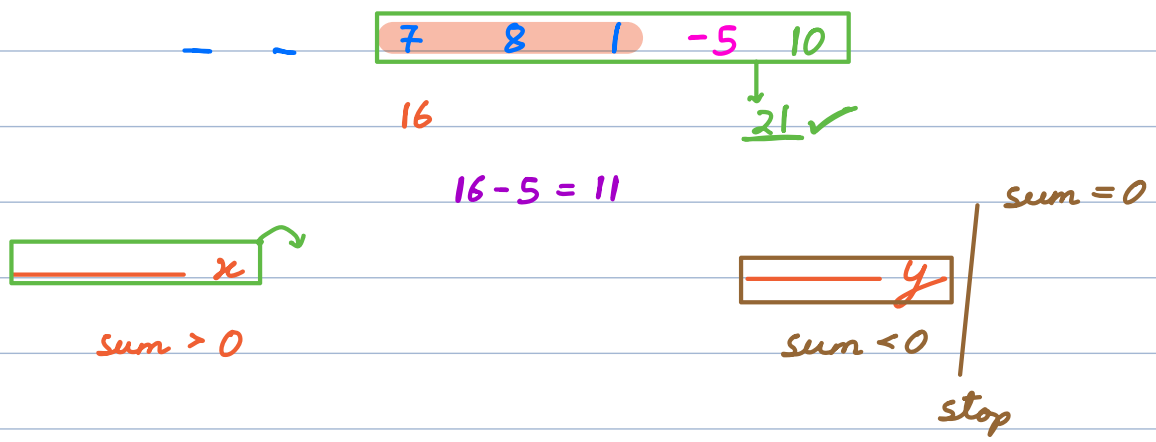


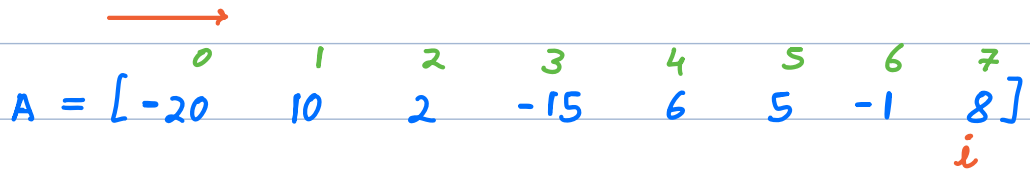

```
ans = INT_MIN      // ans = A[0]
for l → 0 to (N-1) {
    sum = 0
    for r → l to (N-1) {        l — r
        sum += A[r]             // calculate
        ans = max(ans, sum)     // use
    }
} return ans
```

SC = $O(1)$

TC = $O(N^2)$

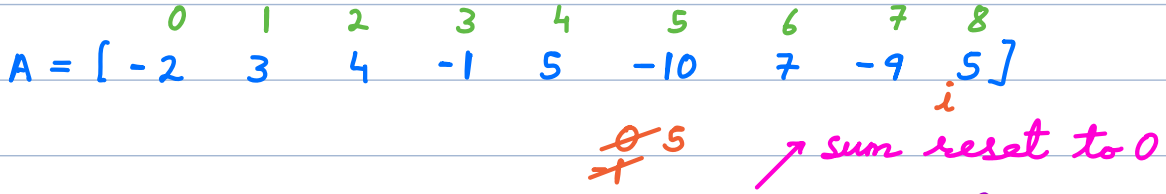
Kadane's Algo





sum = ~~-20~~ ~~0~~ ~~10~~ ~~12~~ ~~-3~~ ~~0~~ ~~6~~ ~~11~~ ~~10~~ 18

ans = ~~-∞~~ ~~-20~~ ~~10~~ ~~12~~ 18 ✓



sum = ~~-2~~ ~~0~~ ~~3~~ ~~7~~ ~~6~~ ~~11~~ ~~18~~ ~~0~~ 5

ans = ~~-∞~~ ~~-2~~ ~~3~~ ~~7~~ 11

start = ~~0~~ → 8 ↗ sum reset to 0

end = ~~0~~ ~~1~~ ~~2~~ 4

↘ ans update

↳ R (end) = i

L = start

```
ans = A[0]
sum = 0
for i → 0 to (N-1) {
    sum += A[i]
    ans = max(ans, sum)
    if (sum < 0)  sum = 0
}
return ans
```

Sum = ~~0~~ ~~-2~~ ~~0~~ ~~-5~~ ~~0~~ ~~-1~~ ~~0~~ ~~-3~~ 0

ans = ~~-2~~ -1

TC = O(N) SC = O(1)

sum = ~~2~~ ~~-3~~ ~~0~~ ~~3~~ ~~11~~ ~~-1~~ ~~0~~ 5

ans = ~~-∞~~ ~~2~~ ~~3~~ 11

start = ~~0~~ ~~2~~ 5

L = start = ~~0~~ ~~2~~ 2

R = i = ~~0~~ ~~2~~ 3

Q → Given an integer array A where $\forall i, A[i] = 0$.
Return the array after performing multiple queries i.e.]

Query (i, x) → Add x to all elements from index i to (N-1).

```
          0   1   2   3   4   5   6
A = [ 0   0   0   0   0   0   0 ]      Queries
     +3  +3  +3  +3  +3  +3             (1, 3)
         +2  +2  +2                     (4, 2)
             -1  -1  -1  -1             (3, -1)
 ---------------------------
  0  3   3   2   4   4   4
 ---------------------------
```

```
          0   1   2   3   4
A = [ 0   0   0   0   0 ]
     (+3) +3  +3  +3                    (1, 3) → TC = O(N)
 +2  +2  +2  +2  +2                     (0, 2)
             +1                         (4, 1)
 ---------------------
  2  5   5   5   6
 ---------------------
```

Total TC = O(Q * N)

SC = O(1)

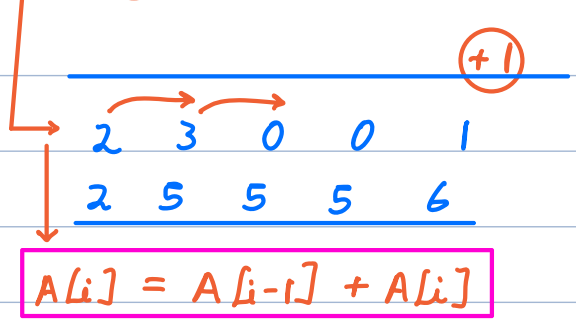
0 . . . . A[i] - - - - (N-1)

Prefix Sum

P[i] = P[i-1] + A[i]

```
          0   1   2   3
A = [ (2)  8  -5   1 ]
P = [ 2   10   5   6 ]
```

```
          0   1   2   3   4
A = [ 0   0   0   0   0 ]
     (+3)                           (1, 3)
 (+2)                               (0, 2)
```

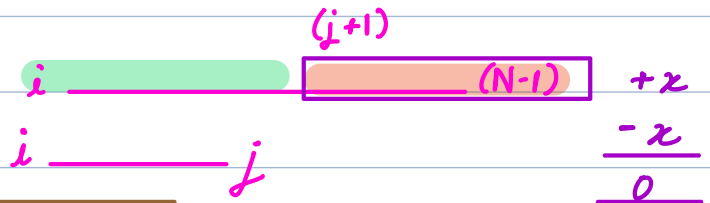



```
for i → 0 to (Q-1) {
    | ind, x
    | A[ind] += x
}
for i → 1 to (N-1) {
    | A[i] = A[i-1] + A[i]
}
```

TC = O(Q + N)

SC = O(1)

Query (i, j, x) → Add x to all elements from index i to j. [i <= j]



Query (i, x)
& Query (j+1, -x) → check if (j+1) < N

i	j	x
1	4	3
0	5	-1
2	2	4
4	6	3

A = [0 0 0 0 0 0 0 0]
0 1 2 3 4 5 6 7

+3 -3 -1 +1 4 -4 +3 -3

-1 3 4 -4 3 -3 1 -3

-1 2 6 2 5 2 3 0 ✓

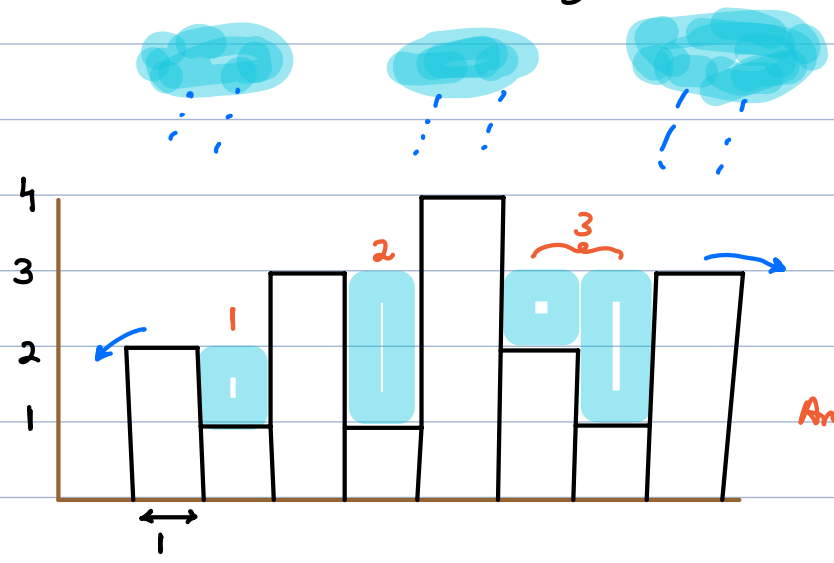
TC = O(Q + N)

SC = O(1)

10:50 PM

Q → Rain Water Trapping

∀ buildings w = 1



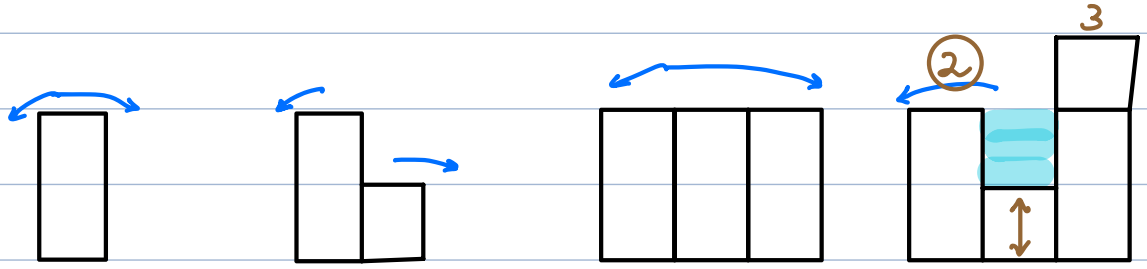
Find total water trapped.

Area → h * w

Ans = 6

A[i] → height

A = [2 1 3 1 4 2 1 3]



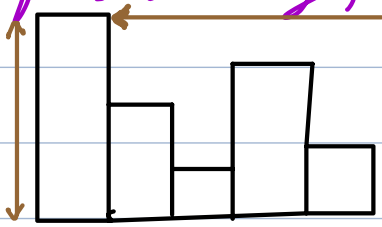
Ans → 0 0 0 1*1 = 1

h = min (left boundary, right boundary)

Area above 1 building → 1 * h

Ans = Σ water (area) above all buildings

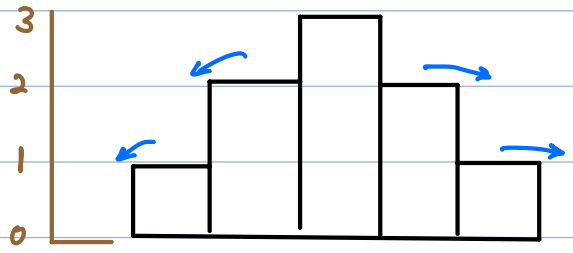
left boundary, right boundary



tallest on left

tallest building on right

A = [1 2 3 2 1]



Ans = 0

∀i find max element from index 0 to i
& from index i to (N-1)

```
ans = 0
for i → 0 to (N-1) {
    l = max on left
    r = max on right
    water = min (l, r) - A[i]
    ans += water
}
return ans
```

TC = O(N)

TC = O(N²) SC = O(1)

```
left_max[0] = A[0]
for i → 1 to (N-1) {
    left_max[i] = max(left_max[i-1], A[i])
}
right_max[N-1] = A[N-1]
for i → (N-2) to 0 {
    right_max[i] = max(right_max[i+1], A[i])
}
```

```
ans = 0
for i → 0 to (N-1) {
    water = min (left_max[i], right_max[i]) - A[i]
    ans += water
}
return ans
```

TC = O(N) SC = O(N)

	0	1	2	3	4	5	
A = [	3	1	1	2	1	3	]
L =	3	3	3	3	3	3	
R =	3	3	3	3	3	3	
w =	0	2	2	1	2	0	→ 7 (Ans)
