SCALER

One Dimensional Array

TABLE OF CONTENTS

Stock

1	2	3	4	5	6	7
A = [-20	30	40	-10	50	-100	70]

Max profit in one trade → 30 + 40 - 10 + 50 = 110

< Question > : Given arr[N]. Find the maximum subarray sum out of all subarrays.

$1 \le N \le 10^5$

Example 1 : arr[ ] → [-3 , 2 , 4 , -1 , 3 , -4 , 3]

0 1 2 3 4 5 6

Ans = 8

Example 2 : arr[ ] → [4 , 5 , 2 , 1 , 6]

0 1 2 3 4

Ans = 18

Example 3 : arr[ ] → [-4 , -3 , -6 , -9 , -2]

Ans = -2

$x > y$

$-x < -y$

Bruteforce → ∀ subarray, calculate sum & take max.

$\frac{N*(N+1)}{2}$

travel → O(N)

Total TC = $O(N^3)$ SC = $O(1)$

O(1) → Prefix Sum ✓

Carry Forward ✓

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

TC = $O(N^2)$ SC = $O(1)$

Observations

Case 1 — When all elements are positive

2	4	5	1	3

Ans = $\sum A[i]$ $\forall i$

Case 2 — When all elements are negative

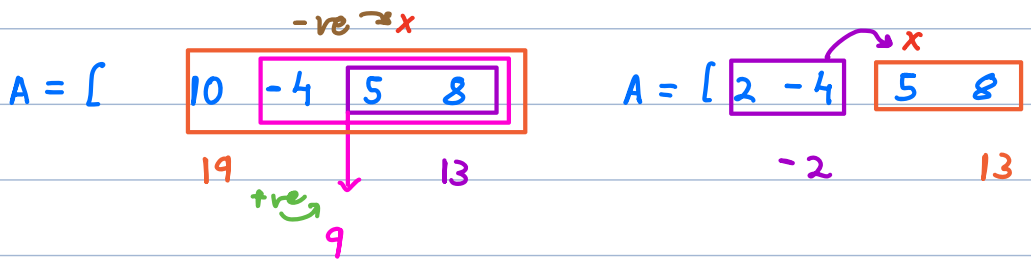
-7	-3	-11	-8	-15

Ans = $\max(A[i])$ $\forall i$

Case 3

	+ve	

Kadane's Algo



*Dry-Run

arr[ ] →	5	6	7	-3	2	-10	-12	8	12	-4	7	-2
	0	1	2	3	4	5	6	7	8	9	10	11

k = 0, l = 7, i = 11

sum = 5 11 18 15 17 7 -5 0 8 20 16 23 21

ans = 5 11 18 20 23

</> Code

```
ans = INT_MIN
sum = 0        l = 0
for i → 0 to (N-1) {
    sum += A[i]
    if (ans < sum) {
        ans = sum
        L = l      R = i
    }
    if (sum < 0) {
        sum = 0    l = i+1
    }
} return ans
```

Kadane's Algo

Subarray → L — R

	0	1	2 (i)
A =	-2	-1	-5

sum = 0 -2 0 -1 0 -5 0

ans = -2 -1

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

Continuous Sum Query

< *Question* > : There are A beggars sitting in a row outside a temple. Each beggar has an empty pot initially. There are N devotees. Each devotee gives some fixed amount to beggars from indices l to r.

Given a 2-D Array B[N][3], where B[i][0] represents l [left index] l
B[i][1] represents r [right index] r
B[i][2] represents amount

Given an integer array, where every element is 0 initially. Update the array with multiple queries.
Query (l, r, x) → ∀ i → l to r A[i] += x

Example : A = 5 , B[N][3] →

l	r	Amount(x)
1	2	10
2	3	20
2	5	25

0	0	0	0	0
1	2	3	4	5

+10 +10
+20 +20
+25 +25 +25 +25

10 55 45 25 25 (Ans)

Query (i, x) →
Add x to all the numbers from i to (N-1).

A = [0 0 0 0 0]
0 1 2 3 4

+3 +3 +3 +3
+2 +2 +2 +2 +2
+1

2 5 5 5 6 (Ans)

(N-1)

I/P → 1 4 3
0 4 2
4 4 1

BF Idea

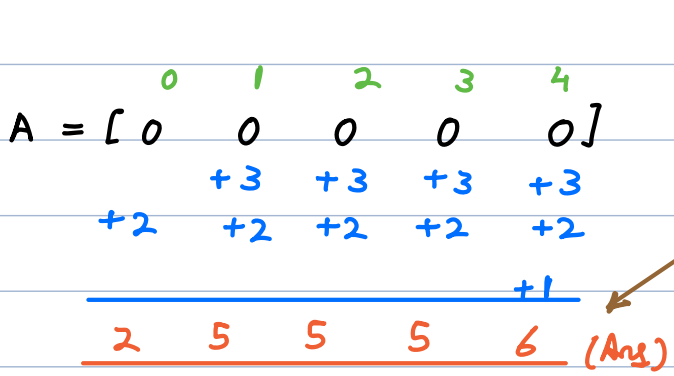
∀ query, travel from l to r & add x.

length of array → A

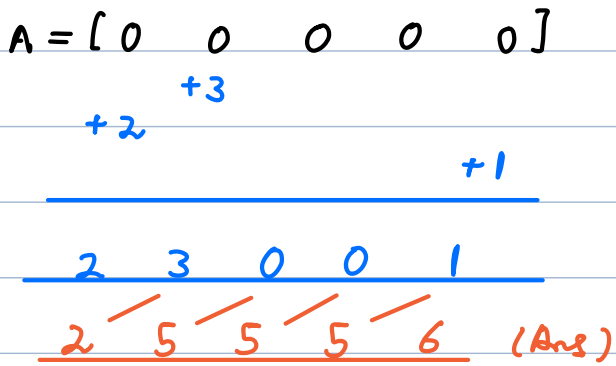
queries → N

```
for i → 0 to (N-1) {   // query    // For 1-based do -1
    l = B[i][0]    r = B[i][1]    x = B[i][2]
    for j → l to r {
        ans[j] += x
    }
}
return ans
```

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



Add x from i ⟶ (N-1)



Prefix Sum

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

```
for i → 0 to (N-1) {  // query
    l = B[i][0]      x = B[i][1]
    ans[l] += x
}

for i → 1 to (A-1) {
    ans[i] = ans[i-1] + ans[i]
}      P[i]   P[i-1]  + a[i]

return ans
```

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

A = [0 0 0 0 0 0 0] (indices 0 1 2 3 4 5 6)

+2 at 1, -2 at 4, -5 at 4, +3 at 2, -3 at 6

0 2 3 0 -7 0 -3

0 2 5 5 -2 -2 -5

Queries

l	r	x
1	3	2
4	6	-5
2	5	3

```
for i → 0 to (N-1) {  // query
    l = B[i][0]      r = B[i][1]      x = B[i][2]
    ans[l] += x
    if (r != (N-1))   ans[r+1] -= x
}

for i → 1 to (A-1) {
    ans[i] = ans[i-1] + ans[i]
}      P[i]   P[i-1]  + a[i]

return ans
```

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

< Question > : Given arr[N], where arr[i] → height of building.

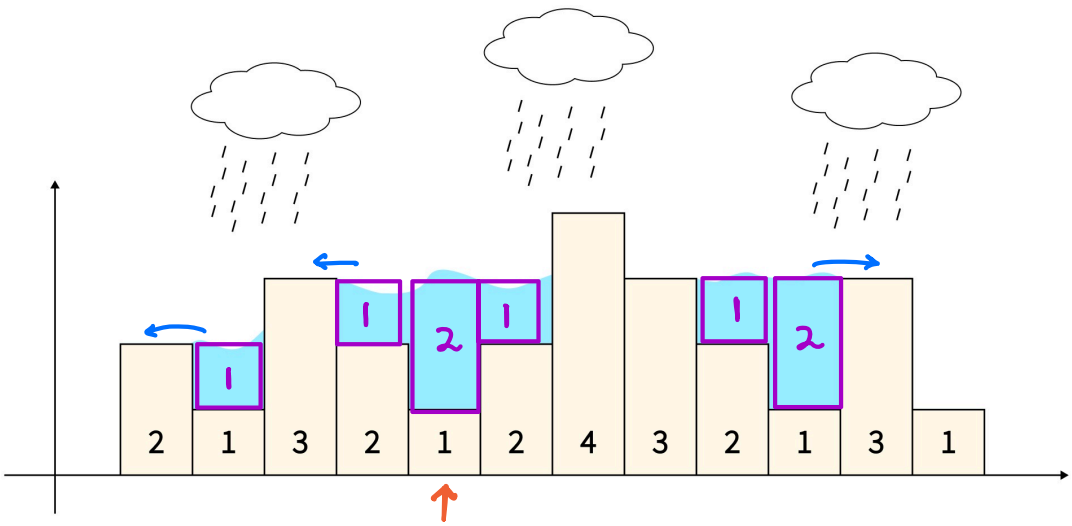
Base = 1 ∀ buildings

Return amount of water trapped on all the buildings.

Area = B * H

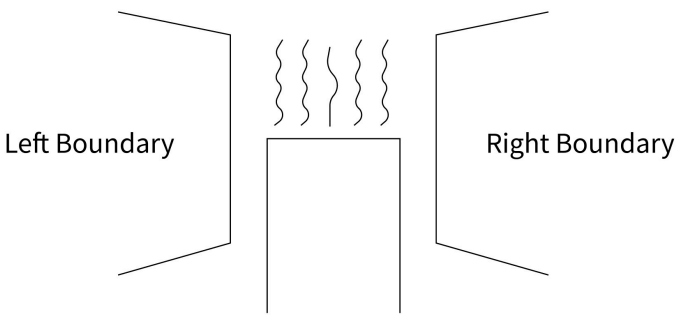
arr [2 1 3 2 1 2 4 3 2 1 3 1]

$1 \le N \le 10^5$

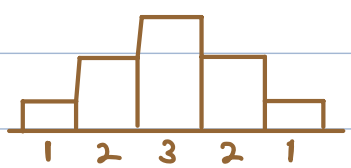


Ans = 8

Idea -1 Find the amount of water trapped on every building.



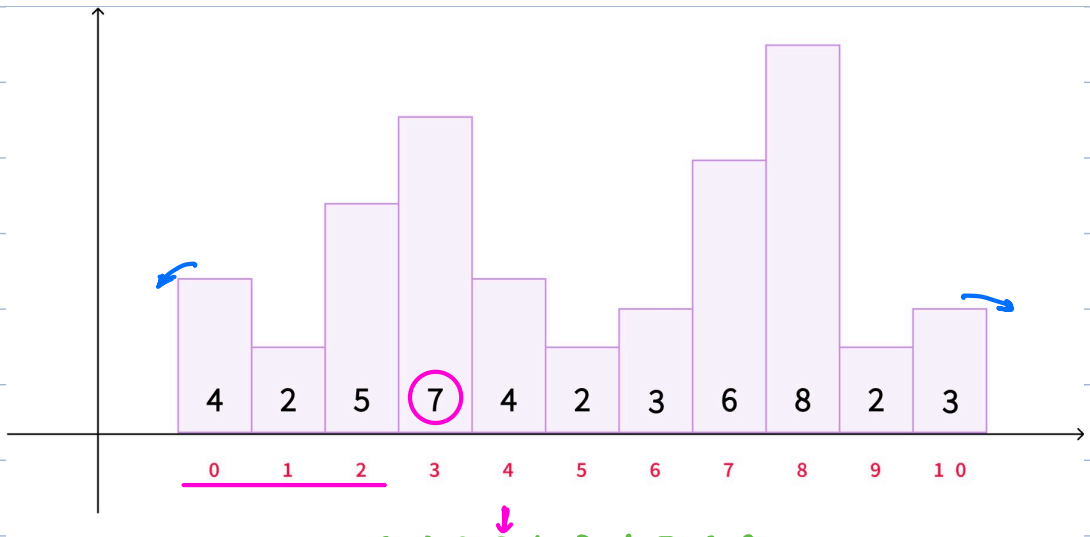
min (max A[i] on left , max A[i] on right) - A[i]



Ans = 0

Bruteforce → TC = O(N²)

SC = O(1)



0 1 2 3 4 5 6 7 8 9 10

→ lmax[] → [4 4 5 7 7 7 7 7 8 8 8]

→ rmax[] → [8 8 8 8 8 8 8 8 3 3]

```
lmax[0] = A[0]
for i → 1 to (N-1) {        // →
    lmax[i] = max(lmax[i-1], A[i])
}

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

Prefix / Suffix → sum, max, min etc

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

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