Avg PSP → 81% → 85% next week target

100% PSP → 68 → 100 93 learners with PSP > 90%.

Next monday → $2^{nd}$ oct no class

Next Week → Tue & Thur. ✓

---

Q → Given an integer array, find the max subarray sum.

$$A = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ [-2 & 3 & 4 & -1 & 5 & -10 & 7] \end{array}$$

Ans = 11

$$A = \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ [4 & 5 & 2 & 1 & 6] \end{array}$$

Ans = 18

$$A = \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ [-4 & -3 & -6 & -9 & -2] \end{array}$$

Ans = -2

Bruteforce → ∀ subarrays, calculate sum & take max.

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

A = [1 2 3]

```
1
1 2     } 3
1 2 3
  2
  2 3   } 2
    3   } 1
    ----
      6
```

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

→ subarray sum

1) prefix sum → P[j] − P[i-1]

2) carry forward.

TC = O($N^3$)

SC = O(1)

```
    }
}  return ans
```

---

```
ans = INT_MIN
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
```

$$\begin{array}{cccc} & 0 & 1 & \overset{i}{2} \\ A = [ & 4 & -2 & \underset{j}{3} ] \end{array}$$

sum = ~~0~~ 3

ans = ~~-∞~~ ~~4~~ 5 ✓

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

---

Observations →

1) $\forall i,\ A[i] >= 0 \Rightarrow Ans = \sum_{i=0}^{N-1} A[i]$

$$\begin{array}{ccccccc} & 0 & 1 & 2 & 3 & 4 & \\ A = [ & 4 & 5 & 2 & 1 & 6 & ] \end{array}$$

Ans = 18

2) $\forall i,\ A[i] < 0 \Rightarrow Ans = \underset{i}{\forall}\ max(A[i])$

$$\begin{array}{ccccccc} & 0 & 1 & 2 & 3 & 4 & \\ A = [ & -4 & -3 & -6 & -9 & -2 & ] \end{array}$$

Ans = -2

3) A = [ -ve  -ve  +ve  +ve  +ve  -ve ]

decrease the sum

4) 7  -3  5  8  2

increase the sum.

sum = 15

Kadane's Algo

$$\begin{array}{cccccccccccc} & \overset{st}{✓} & & & & \overset{end}{✓} & & & & & & \\ & ✓ & ✓ & ✓ & ✓ & ✓ & ✓ & ✓ & ✓ & ✓ & ✓ & \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \\ A = [ & -2 & 3 & 4 & -1 & 5 & -10 & -7 & 2 & -5 & 12 & ] \\ sum = & \cancel{-2} \to \cancel{0} & \cancel{3} & \cancel{7} & \cancel{6} & \cancel{11} & \cancel{1} & \cancel{-6} \to \cancel{0} & 2 & \cancel{-3} \to 0 & 12 & \\ ans = & \cancel{-2} & \cancel{3} & \cancel{7} & & \cancel{11} & & & & & \underline{12} & \end{array}$$

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

TC = O(N)

SC = O(1)

---

Q → Given an integer array where ∀i, A[i] = 0. Return the final array after performing multiple queries.

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 |
|   |   | +3 | +3 | +3 | +3 | +3 | +3 |
|   |   |   |   |   | +2 | +2 | +2 |
|   |   |   |   | -1 | -1 | -1 | -1 |
|   | 0 | 3 | 3 | 2 | 4 | 4 | 4 ✓ |

Queries

(1, 3) → TC = O(N)

(4, 2)

(3, -1)

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

Prefix Sum → $P[i] = P[i-1] + A[i]$

✓ $A[i] = A[i-1] + A[i]$ // L → R

$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A = [ & 0 & \cancel{0} & 0 & \cancel{0} & \cancel{0} & 0 & 0] \\ & & 3 & & -1 & 2 & & \end{array}$$

→ 0 3 3 2 4 4 4

Queries

(1, 3)

(4, 2)

(3, -1)

```
for i → 0 to (Q-1) {   // B[Q][2] → { B[i][0], B[i][1] }
                                      i         x
                       0<=i<N
    A[B[i][0]] += B[i][1]
}
for i → 1 to (N-1) {
    A[i] = A[i-1] + A[i]
}
return A
```

$TC = O(N+Q)$

$SC = O(1)$

---

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

Query $(i, j, x)$ → Add x to all elements from index i to j.

$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A = [ & 0 & 0 & 0 & 0 & 0 & 0 & 0] \\ & & +3 & +3 & +3 & +3 & & \\ & & & & & +2 & +2 & \\ & & & & -1 & -1 & -1 & -1 \end{array}$$

→ 0 3 3 2 4 1 -1

Queries

(1, 4, 3)

(4, 5, 2)

(3, 6, -1)

$$\begin{array}{lcccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
A = [ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 ] \\
 & & 3 & 3 & 3 & 3 & & & \\
 & -1 & -1 & -1 & -1 & -1 & -1 & & \\
 & & & 4 & & & & & \\
 & & & & & 3 & 3 & 3 & \\
\rightarrow & -1 & 2 & 6 & 2 & 5 & 2 & 3 & 0
\end{array}$$

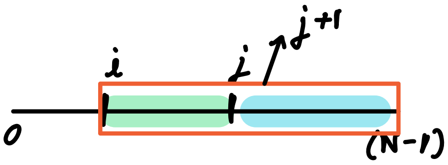
**Queries**

(1, 4, 3)

(0, 5, -1)

(2, 2, 4)

(4, 6, 3)



$(i, j, x)$ $\quad i — j \quad +x$

$(i, x) \rightarrow i$ ____ $(N-1) \quad +x$

$(j+1, x) \rightarrow (j+1)$ ____ $(N-1) \quad -x$

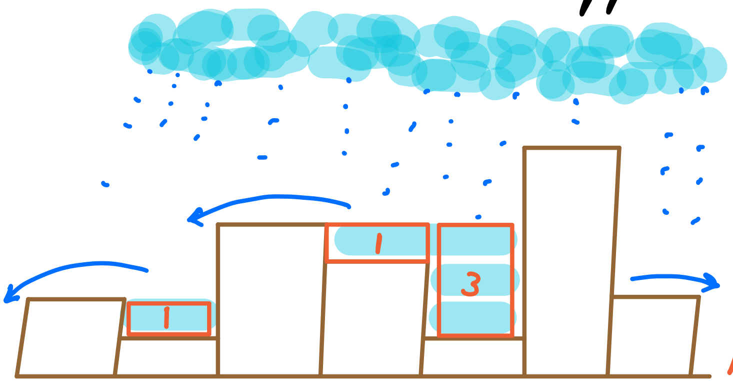
```
for i → 0 to (Q-1) {                  // (B[i][0], B[i][1], B[i][2])
                                            i        j        x
    A [B[i][0]] += B[i][2]
    if ( B[i][1] < (N-1))   A[B[i][1] + 1] -= B[i][2]
}
for i → 1 to (N-1) {
    A[i] = A[i-1] + A[i]
}
return A
```

TC = O(N+Q)

SC = O(1)

10:58 PM

Q → Given N buildings & height of each building, find the rain water trapped b/w the buildings.



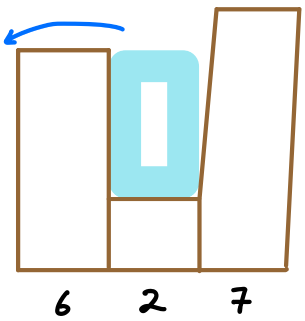
Area

- Height → A[i]
- Base → 1, ∀i

Ans = 5

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| A = [ | 2 | 1 | 4 | 3 | 1 | 6 | 2 ] |

area = (6-2) = 4



Area of water above $i^{th}$ building

= min ( maxleft , maxRight ) - A[i]

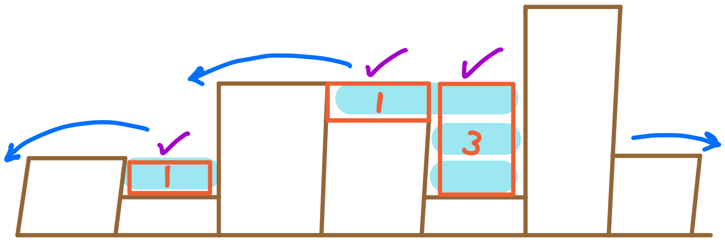
Ans = $\sum_{i=0}^{N-1}$ water above every building

O(N) * O(N) → TC = $O(N^2)$

Prefix Max

prefix sum L ← R

⇒ suffix sum



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|---|
| A = [ | 2 | 1 | 4 | 3 | 1 | 6 | 2 ] | |
| maxL = | 2 | 2 | 4 | 4 | 4 | 6 | 6 | |
| maxR = | 6 | 6 | 6 | 6 | 6 | 6 | 2 | |
| | 0 | 1 | 0 | 1 | 3 | 0 | 0 | → Ans = 5 |

maxL[i] =

max (maxL[i-1], A[i])

maxR[i] =

max (maxR[i+1], A[i])

min ( maxL[i] , maxR[i]) - A[i]

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

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

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

```
ans = 0
for i → 1 to (N-2) {
        ans += min(maxL[i], maxR[i]) - A[i]
}
return ans
```

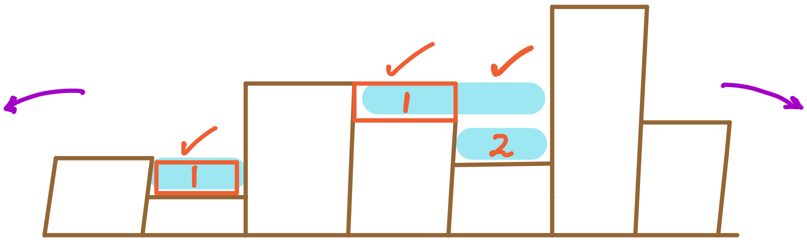
A = [4  1  1  4]

maxL = 4  4  4  4

maxR = 4  4  4  4

0  3  3  0 → Ans = 6

---



$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A = [ & 2 & 1 & 4 & 3 & 2 & 6 & 3\ ] \\ & & & & & r & & \\ & & & & & l & & \end{array}$$

Ans = 4

maxL = $\cancel{2}$ 4

maxR = $\cancel{3}$ 6

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

$$\begin{array}{ccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A = [ & 1 & 5 & \boxed{3} & \boxed{2} & 7 & 4 & 3 & 1\ ] \\ & \cancel{l} & \cancel{l} & \cancel{l} & r & \cancel{r} & \cancel{r} & \cancel{r} & r \\ & & & & l & & & & \end{array}$$

5 - 3 = 2    5 - 2 = 3    Ans = 5

maxL = ~~1~~ 5

maxR = ~~1~~ ~~3~~ ~~4~~ 7

```
maxL = A[0]
for l → 1 to (N-1) {
    if (A[l] > maxL)   maxL = A[l]
    else   break
}

maxR = A[N-1]
for r → (N-2) to 0 {
    if (A[r] > maxR)   maxR = A[r]
    else   break
}

ans = 0
while (l <= r) {
    if (maxL < maxR) {
            if (A[l] > maxL)   maxL = A[l]
            else   ans += maxL - A[i]
            l++
    } else {
            if (A[r] > maxR)   maxR = A[r]
            else   ans += maxR - A[i]
            r--
    }
}   return ans
```