Max Subarray Sum
Queries
Rainwater Trapping

Reattempt 1 → 23 - 24 Sept

① 2 → 25 Sept → 3 Oct

② Oct 2 Monday → Holiday

1. Given an integer array A, find max subarray sum out of all subarrays.

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

$$A[7]: -2 \quad \underbrace{3 \quad 4 \quad -1 \quad 5}_{11} \quad -10 \quad 7 \qquad \boxed{ans = 11}$$

$$\begin{array}{lccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A[7]: & -3 & 4 & 6 & 8 & -10 & 2 & 7 \end{array}$$

$$A[7]: -3 \quad \underbrace{4 \quad 6 \quad 8}_{18} \quad -10 \quad 2 \quad 7 \qquad \boxed{ans = 18}$$

ar → N subarrays

$$\downarrow$$

$$cnt = \frac{7 * 8}{2} = \boxed{28} \qquad \frac{N * (N+1)}{2}$$

$$\begin{array}{lccccc} & 0 & 1 & 2 & 3 & 4 \\ A \rightarrow & 4 & 5 & 2 & 1 & 6 \end{array}$$

$$A \rightarrow \underline{4 \quad 5 \quad 2 \quad 1 \quad 6} \qquad sum = 18$$

$$\begin{array}{lccccc} & 0 & 1 & 2 & 3 & 4 \\ A \rightarrow & -4 & -3 & -6 & -9 & -2 \\ & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ & -4 & -3 & -6 & -9 & -2 \end{array} \qquad sum = -2$$

BF → go to every subarray, calculate sum, pick max sum

```
int ans
for (s=0 ; s<n ; s++) {
    for (e=s ; e<n ; e++) {
        sum=0
        for (i=s ; i≤e ; i++) {
            sum += arr[i]
        }
        ans = max (ans, sum)
    }
}
```

TC : O(n³)

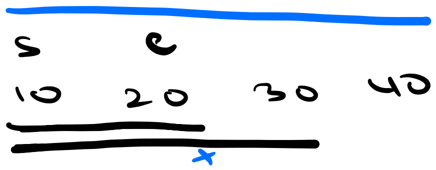
SC : O (1)

N² subarray

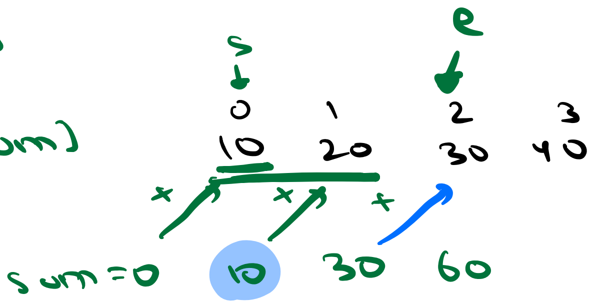
```
// create pf [ ]
int ans
for (s=0 ; s<n ; s++) {                                  s—e
    for (e=s ; e<n ; e++) {
        if (s==0)   sum=pf [e]
        else {
            sum = pf[e] - pf[s-1]
        }
        ans = max (ans, sum)
    }
}
```

TC : O(N²)

SC : O (N)



```
ans = -∞ / INT_MIN

for (s = 0 ; s < N ; s++) {
    sum = 0
    for (e = s ; e < N ; e++) {
        sum += arr[e]
        print (sum)
        ans = max (ans, sum)
    }
}
```



TC : O(N²)

SC : O(1)

s e sum

0 – 0 → arr [0] → valid subarray sum ?

Case 1 : A : 4 2 1 6 7

complete arr

Case 2 : A : -4 -8 -9 -3 -5

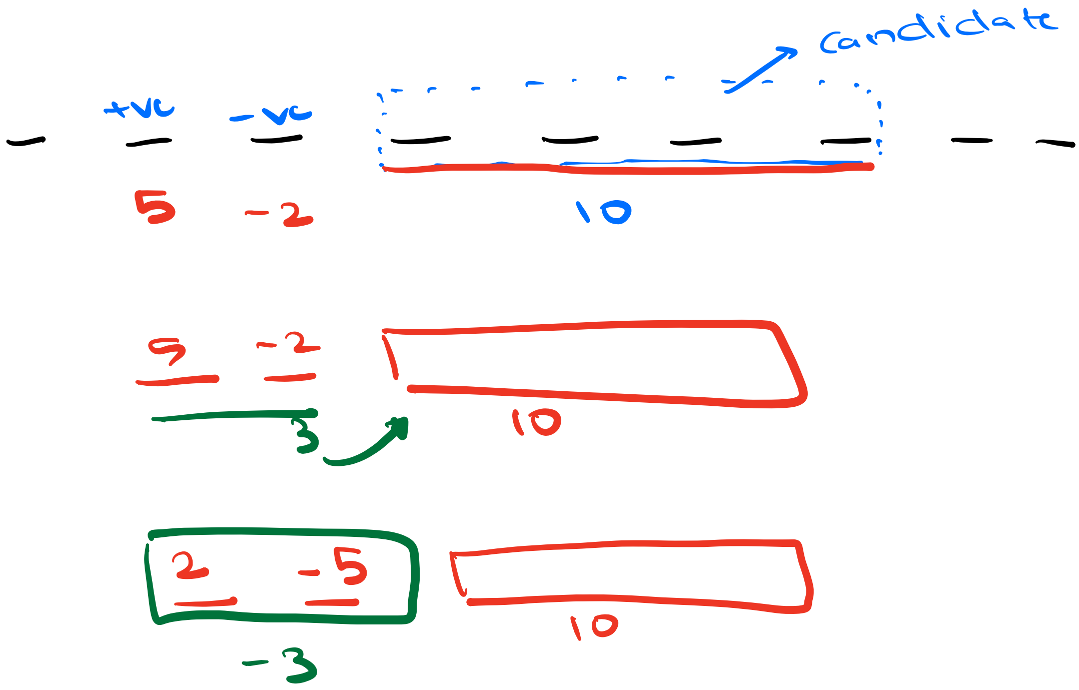
max of arr

Case 3 :

[-ve -ve -ve +ve +ve +ve -ve -ve]

Case 4 :

[-ve -ve -ve +ve +ve +ve +ve]



	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 = -∞	5	11	18	18	18	18	18	18	20	41	41	44

ans → 44

	-2	3	4	-1	5	-10	7
sum = 0	-~~2~~0	3	7	6	11	1	8
ans = -∞	-2	3	7	7	11	11	11

	-4	-2	-8
sum=0	-~~4~~0	-~~2~~0	-8
ans = -∞	-4	-2	-2

ans = -2

Kadane's algorithm

```
int sum = 0 , ans = INT_MIN

for (i=0 ; i <n ; i ++) {
    sum += ar[i];
     if (sum > ans)
                    ans = sum

     if (sum <0)
             sum=0
}
```

TC: O(N)
SC: O(1)

10 : 28

2. Given an integer array A where every element is 0, return final array after performing multiple queries.

Query (i, x) : Add x to all nos. from idx i → N-1

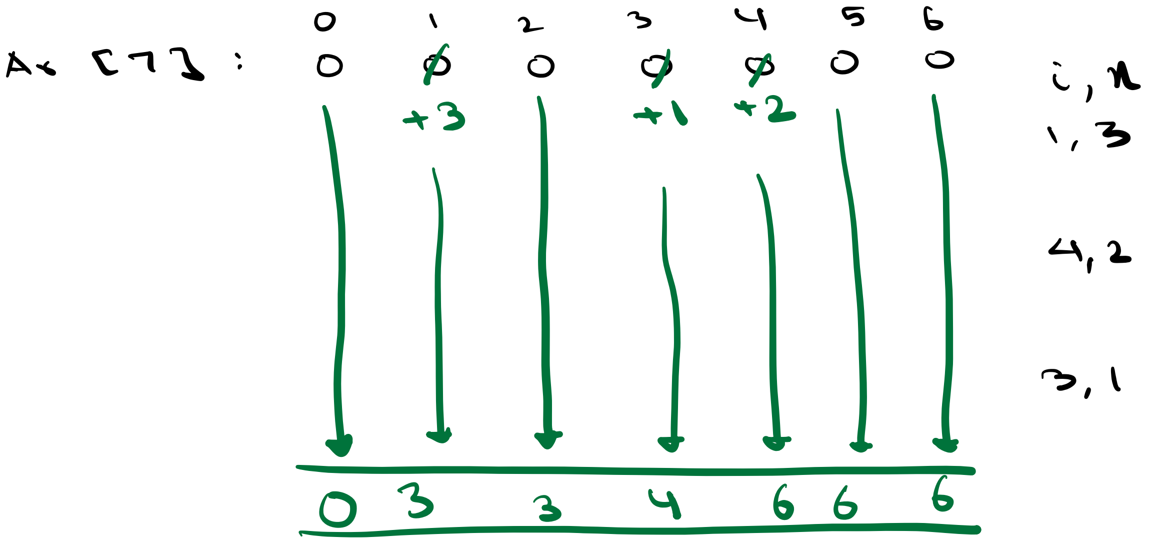
	0	1	2	3	4	5	6	i, x
Ar [7] :	0	0	0	0	0	0	0	
		+3	+3	+3	+3	+3	+3	1, 3
	0	3	3	3	3	3	3	
					+2	+2	+2	4, 2
	0	3	3	3	5	5	5	
				+1	+1	+1	+1	3, 1
	0	3	3	4	6	6	6	

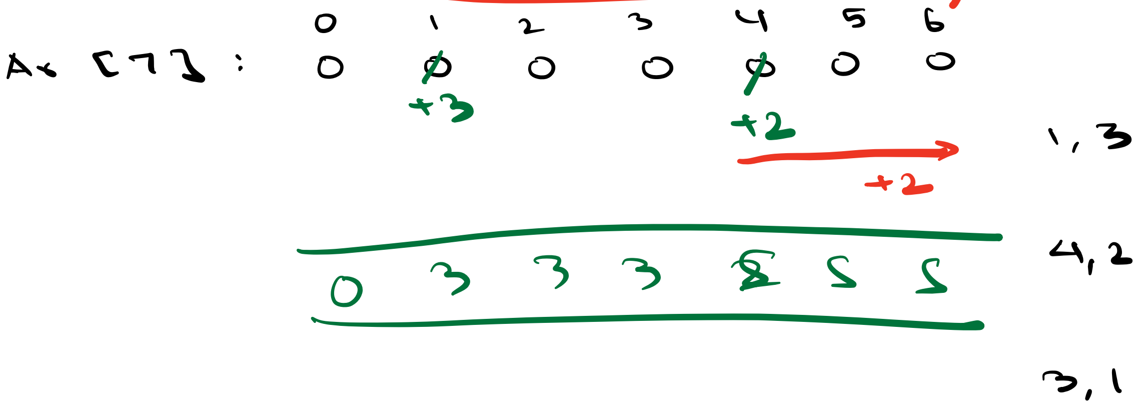
Query (i, x) → Iterating from i → N-1 adding value x to arr[i]

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

Reach final array in less time

	0	1	2	3	4
A :	a_0	a_1	a_2	a_3	a_4
Pf[]	a_0	a_0+ a_1	a_0+ a_1+ a_2	a_0+ a_1+ a_2+ a_3	a_0+ a_1+ a_2+ a_3+ a_4





```
for (i=0 ; i<Q ; i++) {
    // i, x
    ar[i] += x
}
```

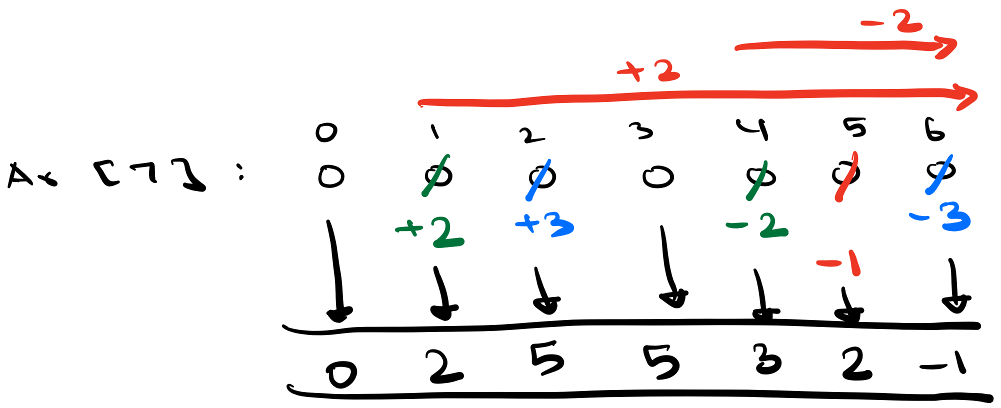
TC: O(Q+N)
SC: O(1)

```
for (i=1 ; i<N ; i++) {
    ar[i] = ar[i-1] + ar[i]
}
    pf[i] = pf[i-1] + ar[i]
```

3. Given an integer array A where every element is 0, return final array after performing multiple queries.

Query (i, j, x): Add x to all nos. from idx i → j (i <= j)

	0	1	2	3	4	5	6	i, j, x
Ar [7] :	0	0	0	0	0	0	0	
		+2	+2	+2				
	0	2	2	2	0	0	0	→ (1, 3, 2)
			+3	+3	+3	+3		
	0	2	5	5	3	3	0	→ (2, 5, 3)
						-1	-1	
	0	2	5	5	3	2	-1	(5, 6, -1)



i, j, x

(1, 3, 2)

(2, 5, 3)

(5, 6, -1)

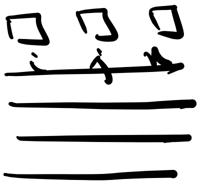
TC: O(Q+N)

SC: O(1)

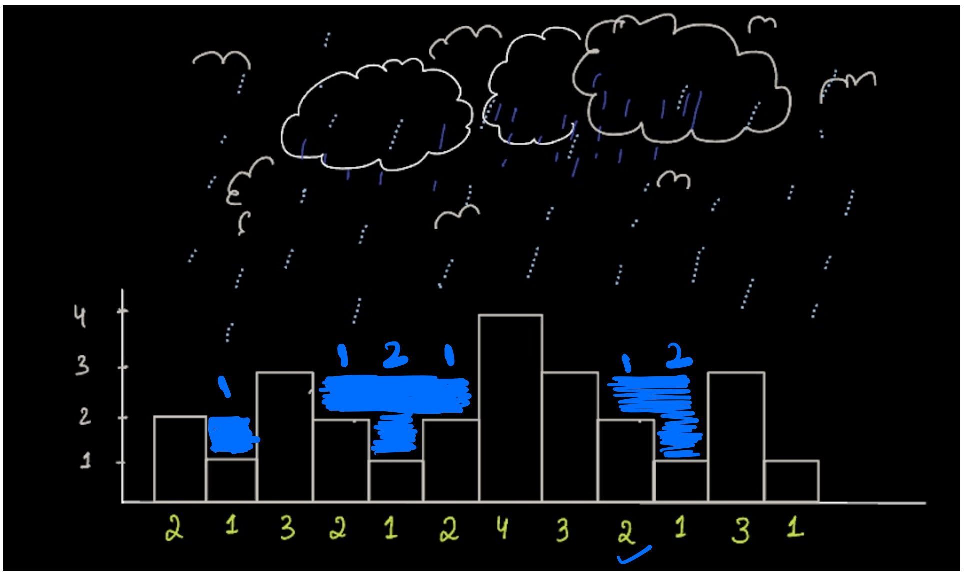
```
for (q=0 ; q<Q ; q++) {
        // i, j, x

        ar[i] += x
        if (j+1 < n)
        ar[j+1] -= x
}
for (i=1 ; i<N ; i++) {
        ar[i] = ar[i-1] + ar[i]
}
        pf[i] = pf[i-1] + ar[i]
```

Queries → 2D array

q x 3

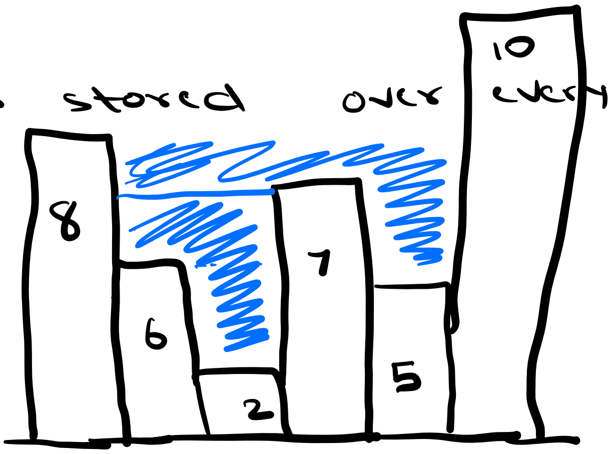


4. Given N buildings with height, find rain water trapped between buildings.



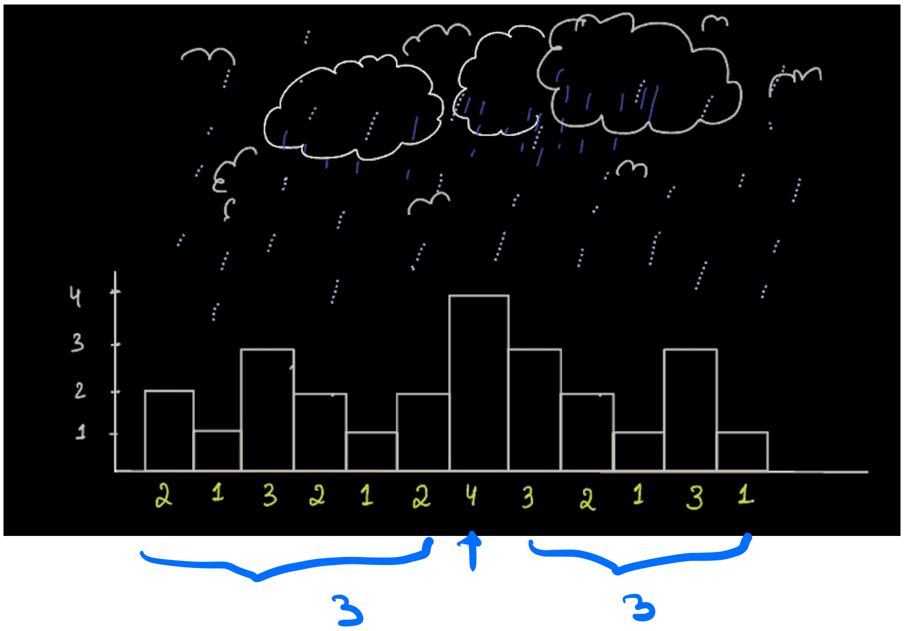
ans = 8

Water stored over every building ?



min (left, right) - ht of building
↓ left max
↓ right max

Water = min (lmax, rmax) - ht of building



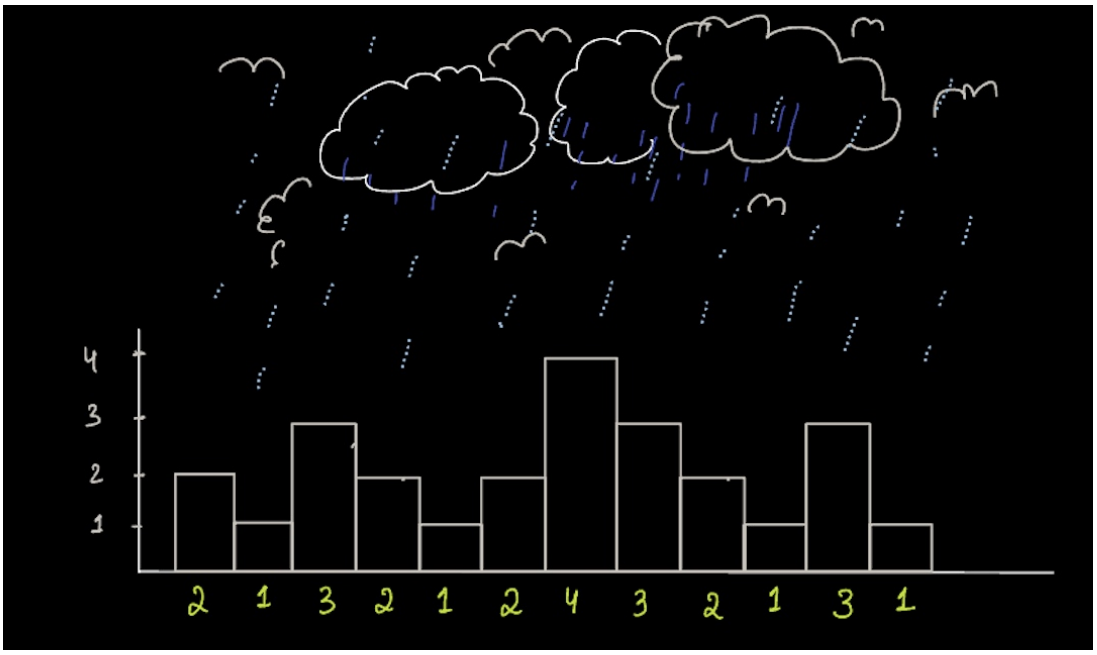
min(3,3)
↓
3 - 4
= -1

BF :

```
        int    ans=0
    for (i=1; i ≤ N-2 ; i++) {

        // ith building → lmax and rmax
        int  lmax = 0
        for ( j=0 ; j < i ; j++) {
              lmax = max (lmax, h[j])
        }

        int  rmax = 0
        for ( j=i+1 ; j < N; j++) {
              rmax = max (rmax, h[j])
        }

     int water = min (lmax, rmax)- h[i]
        if (water > 0)
              ans += water
    }
```

TC: $O(N^2)$

SC: $O(1)$

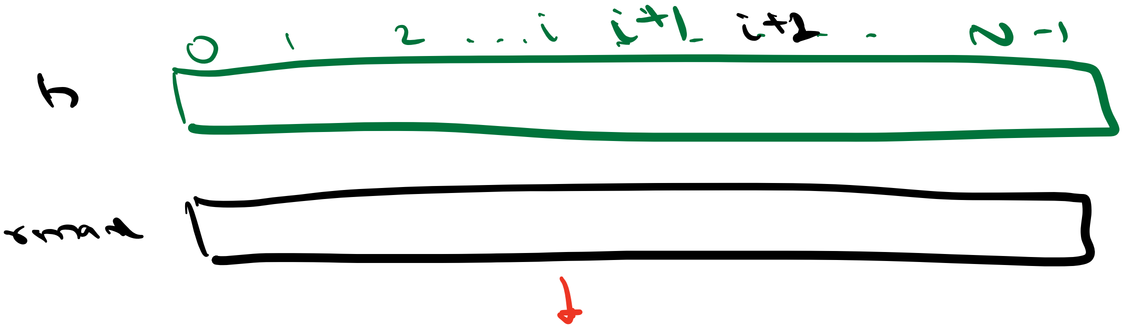


lmax	0	2	2	3	3	3	3	4	4	4	4	4
rmax	4	4	4	4	4	4	3	3	3	3	1	0
water	~~2~~	1										

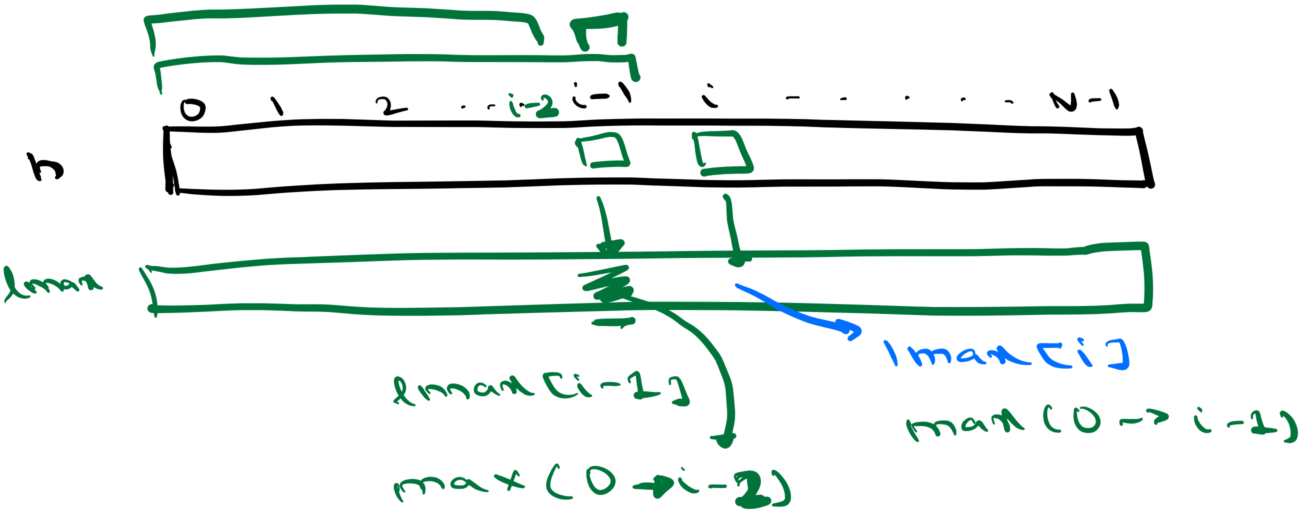
```
            int  ans = 0
            int  lmax[N] , rmax[N]
   lmax[0] = 0
  for (i=1  ; i < N  ; i++) {
        // lmax[i]          0 —i-1
      lmax[i]  = max (lmax[i-1],    h[i-1])
           ↓                ↓                ↓
       0 -> i-1        0 -> i-2            i-1
  }
```

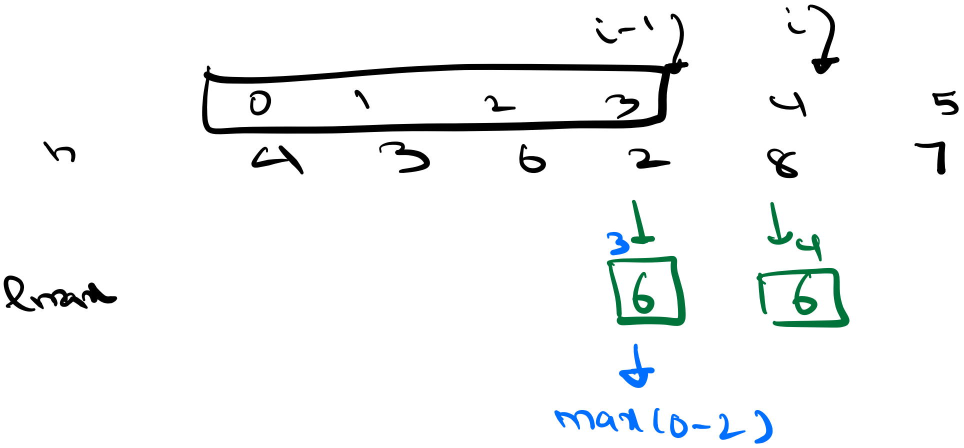
```
rmax [N-1] =0

for (i= N-2 ; i ≥ 0 ; i--) {

    rmax[i]  = max(rmax[i+1] , h[i+1])
        ↓                  ↓
    i+1 → N-1          i+2 → N-1
}

int  ans=0
  for  (i=1 ;  i ≤ N-2  ; i++) {

    water = min ( lmax[i], rmax[i]) - h[i]
      if  (water > 0)
                 ans+ =water
  }
```

TC : O(N)   SC : O(N)



max (i+1, N-1)





i=3

	0	1	2		3	4
ar →	2	~~4~~ 6	~~6~~ 12	+	~~8~~ 20	4

pf[3] = pf[2] + a[3]

$\underline{a[3]} = \underline{a[2]} + a[3]$

pf[2] = pf[1] + ar[2]

$\underline{a[2]} = a[1] + a[2]$

pf = pf + cur el.

pf → $\boxed{2 \quad 6 \quad 12 \quad 20 \quad 29}$

$\overset{\times}{1} \quad 6 \quad 8 \quad \overset{\times}{5} \quad \underbrace{10 \quad 20 \quad 26 \quad 30}_{4} \quad \overset{\times}{35}$

12

13

① n^3

② n

			X		X				X
	6	30	1	12	5	24	36	30	35
cnt=0	1	2	X0	1	X0	1	2	3	0
ans=0	1	2	X	2	2	2	2	3	3