Welcome to Advanced Module ☺

Today's content:

- → prefix sum
- → problems based on prefix sum
- → left max[], right max[]
- → Water accumulated
- → Max subarray sum [Google, Amazon, Directi & many more.]

Revision →

arr[10] : [3 2 -1 5 6 8 2 3 2 6]

0 1 2 3 4 5 6 7 8 9

Queries.

s	e	
1	4	: 12
3	6	: 21
1	7	: 25

idea-1.

For every query, iterate and calculate the sum from s to e.

```
// take input for Q.
while ( Q > 0) {
    // take input of s and e
    // find sum of all elements from s
       to e.
    for( i = s ; i <= e ; i++){
        sum += arr[i]
    }
    print(sum);
    Q--
}
```

$$\boxed{\begin{array}{l} T.C = O(N*Q) \\ S.C = O(1) \end{array}}$$

idea 2. [Using prefix Sum]

```
// take input for Q., //Create psum[].          ] N
while ( Q > 0) {
    // take input of s and e
    // find sum of all elements from s
       to e.
    if ( i == 0) print psum[e]
    else print ( psum[e] - psum[s-1])

    print(sum);
    Q--
}
```

Q.

[sum(s e) = psum[e] - psum[s-1]]

$\boxed{\begin{array}{l} T.C = O(N+Q) \\ S.C = O(N) \end{array}}$

⇓

[S.C to O(1) by modifying the given array.]

Q Initially all elements of an arr[] are 0. Then you are given Q queries. Every query contains idx & value. Increment elements from ith idx to last idx by value. Return final state of arr[].

arr[] :	[0	0	0	0	0	0	0]
	0	1	2	3	4	5	6
				+4	+4	+4	+4
		+3	+3	+3	+3	+3	+3
					-2	-2	-2
	0	3	3	7	5	5	5

Queries :

idx.	val
3	4
1	3
4	-2

idea: for every query, iterate and update the values.

```
// take input of Q.
while (Q > 0){
    Q--;
    // take input idx & val.
    for( i = idx ; i < N ; i++){
        arr[i] = arr[i] + val;
    }
}
return arr[];
```

T.C → O(Q·N)
S.C → O(1)

idea.

a → | a_0 | a_1 | a_2 | a_3 | a_4 |
0 1 2 3 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]

pseudo-code (idea-2)

```
// take input of Q.
while ( Q > 0) {
    Q--;
    // take input idx & val.

    arr[idx] += val;
}

//2. convert arr[] to psum[].

for ( i = 1 ; i < N ; i++){
        arr[i] = arr[i] + arr[i-1]
}

return arr[]
```

$$\boxed{\begin{array}{l} T.C \rightarrow O(N+Q) \\ S.C \rightarrow O(1) \end{array}}$$

Eg:

arr →

0	1	2	3	4
0	6	5	-1	0

idx	value
1	3
3	-1
2	5
1	3

0	1	2	3	4
0	6	11	10	10

arr →

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

psum →

0	1	2	3
3	5	4	9

psum[i] = psum[i-1] + arr[i]

arr[i] = arr[i-1] + arr[i]

Q) Initially all elements of arr[] are 0. Given Q queries, Every query contains [s, e, val]. Increment elements from s to e by val.

Google

Return the final state of arr[].

arr[10]: [0 0 0 0 0 0 0 0 0 0]

0	1	2	3	4	5	6	7	8	9

Queries.

s	e	val
3	6	+3
2	7	-3
1	9	+4

			+3	+3	+3	+3			
		-3	-3	-3	-3	-3	-3		
+4	+4	+4	+4	+4	+4	+4	+4	+4	
0	4	1	4	4	4	4	1	4	4

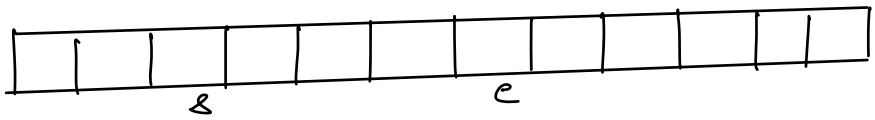
arr[10]: [0 0 0 0 0 0 0 0 0 0]

0	1	2	3	4	5	6	7	8	9

s	e	val
3	6	1
2	7	3
1	6	5
2	5	-4
6	9	2
5	5	10

idx	val.	idx	val.
(3	+1)	(7	-1)
(2	+3)	(8	-3)
(1	+5)	(7	-5)
(2	-4)	(6	+4)
(6	+2)	# do nothing.	
(5	+10)	(6	-10)

Generalize.



Q.

s, e, val. : arr[s N-1] add val

arr[e+1 N-1] add -val. [for counterbalancing].

pseudo-code.

```
// take input of Q.
while (Q > 0){
    Q--;
    // take input of s, e, val.
    arr[s] += val
    if (e+1 < N) { arr[e+1] += (-val) };
}

// 2. convert arr[] to psum[].
for ( i = 1 ; i < N ; i++){
    arr[i] = arr[i] + arr[i-1]
}

return arr[]
```

$\boxed{\begin{array}{l} T.C \rightarrow O(N+Q) \\ S.C \rightarrow O(1) \end{array}}$

→ Given an arr[] . Create 2 arrays pfm[], sfm[].

pfm[i] = max of all elements from 0 to i.

sfm[i] = max of all elements from i to N-1.

arr[] =	1	-6	3	8	4	5	2
	0	1	2	3	4	5	6
pfm[] →	1	1	3	8	8	8	8

---

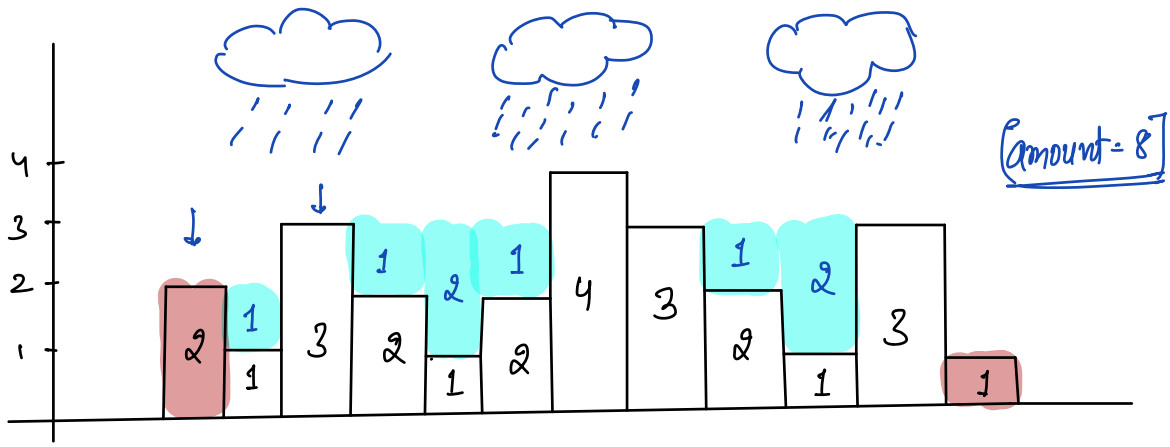
arr[] =	1	-6	3	8	4	5	2
	0	1	2	3	4	5	6
sfm[] →	8	8	8	8	5	5	2

Todo. $\boxed{T.C \rightarrow O(N)}$.

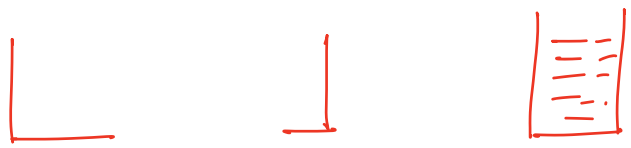
Rain Water Trapped

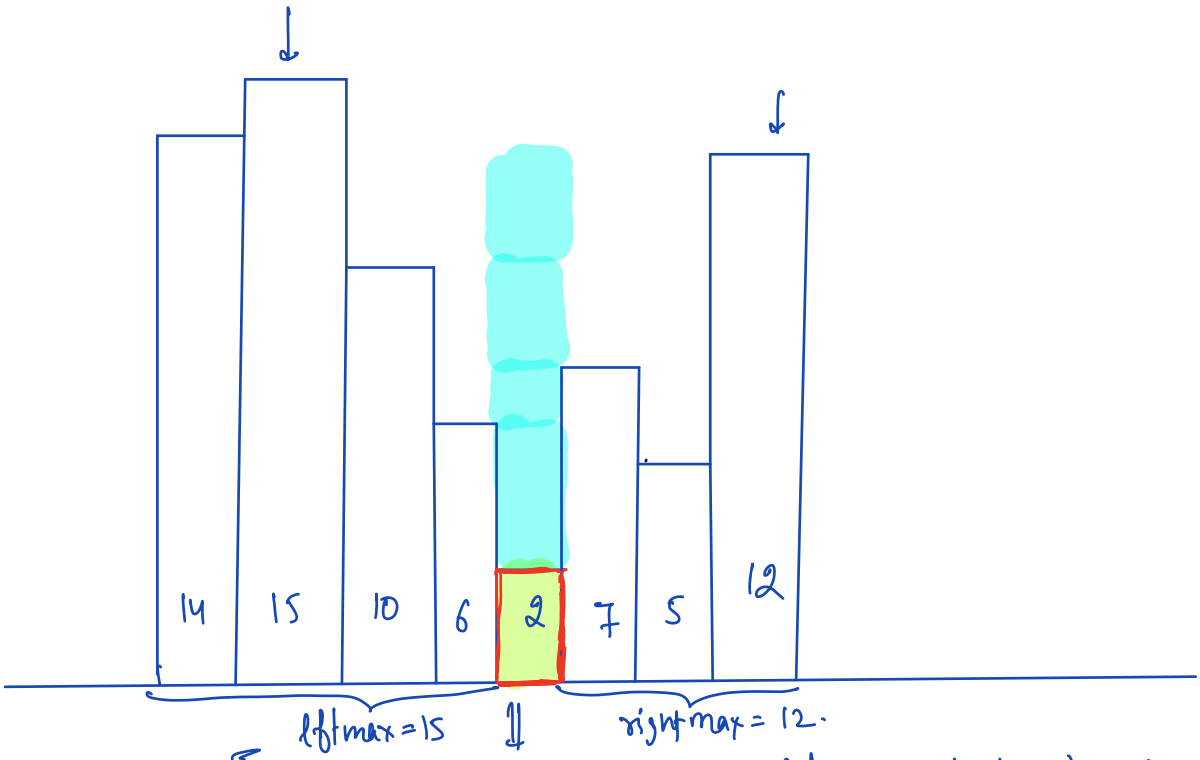
Given N array elements, where arr[i] represents height of the building. Return amount of water trapped on all buildings.

Eg: arr[] = { 2 1 3 2 1 2 4 3 2 1 3 1 }



idea → Sum of water accumulated at each building.



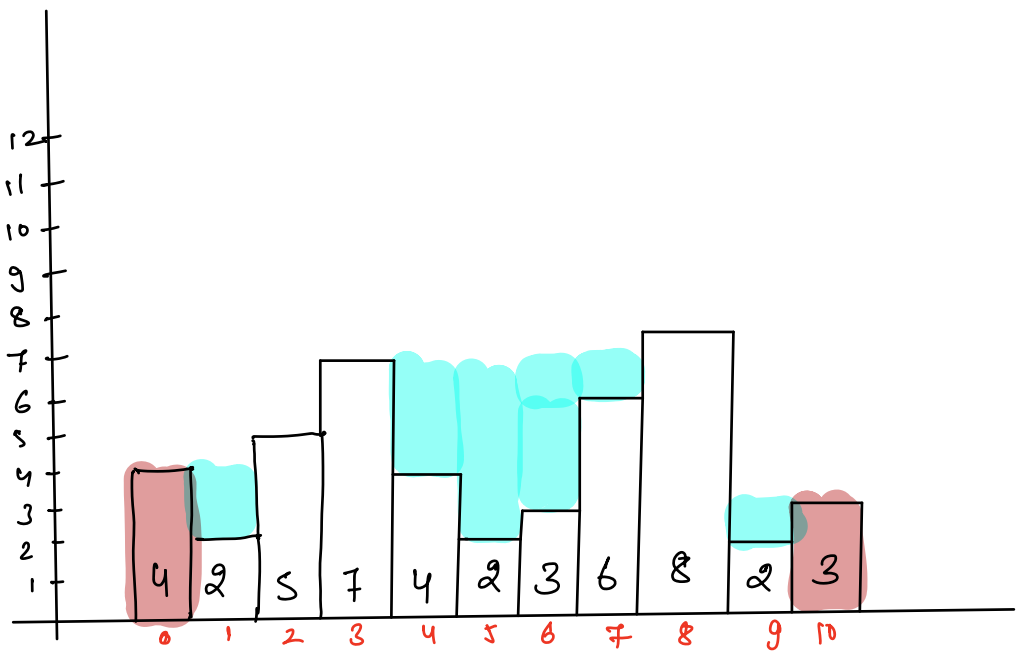


[Water accumulated = Min(leftmax, rightmax) - ht of building]

$$= \text{Min}(15, 12) - 2$$

$$= 12 - 2 = \boxed{10}$$

Eg:



l: [4 4 5 7 7 7 7 7 8 8 8]

r: [8 8 8 8 8 8 8 8 8 3 3]

min(l, r): [4 4 5 7 7 7 7 7 8 3 3]

water[i]: [0 2 0 0 3 5 4 1 0 1 0] ans = 16.

pseudo-code.

```
//Create pfm[] and sfm[]. (for optimisation)
ans = 0 ;

for( i = 1 ; i < N-1 ; i++){
        min = Min( pfm[i], sfm[i])
        water = min - arr[i];
        ans += water
}

return ans;
```

$\boxed{\begin{array}{l} T.C \rightarrow O(N) \\ SC \rightarrow O(N) \end{array}}$

Q) Given N array elements. Calculate maximum subarray sum.

↓

contiguous part of an array.

arr → { -3 2 4 -1 3 -4 3 } ⇒ ans = 8

(indices: 0, 1, 2, 3, 4, 5, 6)

A1: Consider all subarrays.
For every subarray, iterate & find sum.

T.C → $O(N^3)$

A2: pSum[]

T.C → $O(N^2)$
S.C → $O(N)$

Carry forward.

T.C → $O(N^2)$
S.C → $O(1)$

TC1. All array elements are +ve. : Entire array will be ans.

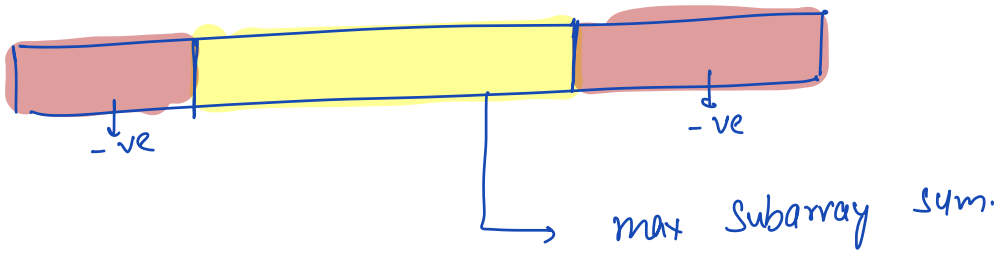
3	2	1	6

$\Rightarrow$ 12.

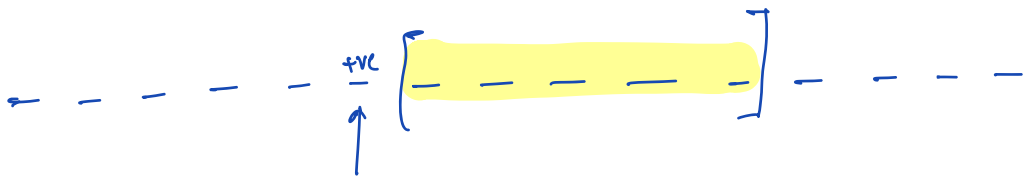
TC2: All array elements are -ve : Max of Entire array will be ans.

-7	-3	-11	-15	-6
0	1	2	3	4

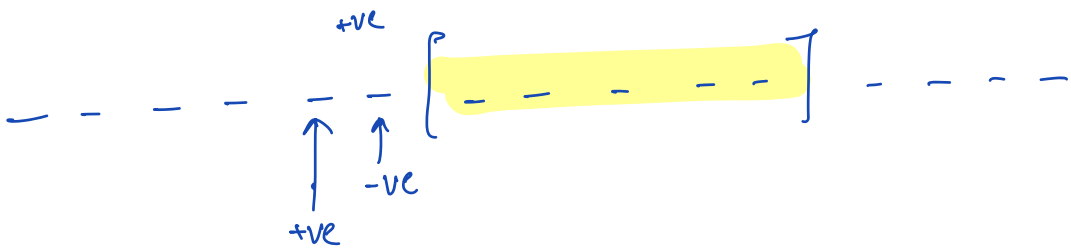
TC3:



TC4:



TC5:



[If sum > 0, only then carry forward sum.]

arr[] :	5	6	7	-3	2	-10	-12	8	12	-4	7
sum=0	5	11	18	15	17	7	~~-5~~ 0	8	20	16	23
ans = -∞	5	11	18	18	18	18	18	18	20	20	23

Kadane's Algorithm

pseudo-code.

```
sum = 0 ,   ans = Integer.MIN_VALUE

for( i=0 ; i < N ; i++){
        sum += arr[i]
        ans = Max(ans, sum) ;
        if (sum < 0){
            sum = 0      //reset.
        }
}
return ans
```

T.C → O(N)
S.C → O(1)

---

① Flip → (max subarray sum).

$f(8\ 3\ 5\ 5\ 7)$

$\Downarrow$

$f(24)$

$\Downarrow$

$f(10)$

$\Downarrow$

$f(1)$

83557.

$\downarrow 10$

8355, $r = 7$

$\downarrow 10$ $\quad +$

835, $r = 5$

$\downarrow 10$ $\quad +$

83, $r = 5$

$\downarrow 10$ $\quad +$

8, $r = 3$

$\downarrow 10$ $\quad +$

0, $r = 8$

28.

[Recursion]

[→ Break down your logic into simple (baby) steps.
→ Then try to code it step by step.]

$\Uparrow$

[200 Questions]

Brute force. [n^2]. ①

$n \log n$

→ sorting

→ heap. (pq)

n

② [T.C. + edge cases.]

③ Diagrams.

[Fun while solving problems.]