1-D Arrays

⇒ Points to Remember

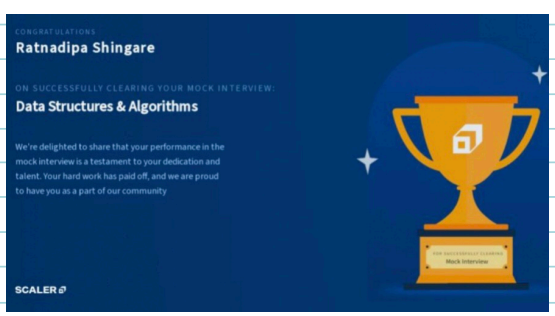
① Revision [atleast 3 time → after class, → At Weekend, → After month]

② Never have Backlog

- → Improper Revision leads to Backlog as you will end up taking more time to solve problems.
- → If Done, then Delight Outcome is awaiting.

⇒ MOCK INTERVIES [Replica of actual Interview]

- → Actual Industry Experts will Conduct.
- → 1 hour
- → Post DSA Curriculum
- → If Cleared, then Scaler will open up placement opportunities for you.



Question 1 :- Given an integer array A, find the maximum sum subarray out of all the subarrays.

Ex 1 arr[] = { -2 3 4 -1 5 -10 7 }

↳ 11 Ans.

Ex 2 arr[] = { -3, 4, 6, 8, -10, 2, 7 }

↳ 18 Ans

Quiz 1 :- For the given array A, what is the maximum subarray sum ?

A[] = { 4, 5, 2, 1, 6 }

* If all elements are +ve, take all

Ans = 18;

Quiz 2 :- For the given array A, what is the maximum subarray sum ?

A[] = { -4, -3, -6, -9, -2 }

↳ Ans

* If all elements are -ve, then take the largest No.

Brute force

↳ Iterate over all the possible Subarrays :- $\frac{N \times (N+1)}{2}$

↳ Calculate the Max Sum

PSEUDO CODE

```
ans = A[0];
for ( i=0; i < N; i++){ // Start to N
    for ( j=i; j < N; j++) { // end
        for ( k=i; k <= j; k++) {
            Sum += A[k];
        }
        ans = Math.max ( ans, Sum);
        Sum = 0; // Reset Sum for the next iteration.
    }
}
return ans;
```

TC - $O(N^3)$
SC - $O(1)$

↳ OPTIMIZATION

IDEA1 :- Use prefix Sum to find sum of sub-array from i to j.

TC - $O(N^2)$
SC - $O(N)$ ← compromised.

IDEA2 :- Use Carry forward for optimizing space.

PSEUDO CODE

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

$TC - O(N^2)$

$SC - O(1)$

IDEA-3 :- KADANE'S ALGORITHM

[Lets try to understand using Egs]

Case I :- All +ves

(+ + + + + + + + + +)

↳ Ans = Sum of all elements

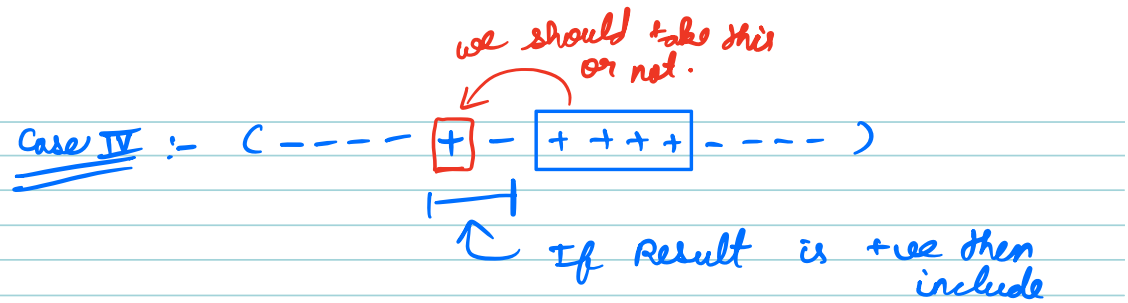
Case II :- All -ves

(- - - - - - - - - -)

↳ Ans = Largest element of Array.

Case III :- (- - - - [+ + + + +] - - - -)

↳ Ans



EXAMPLES

Ex :- [-2 3 4 -1 5 -10 7]

	↓	↓	↓	↓	↓	↓	↓
	-2	3	4	-1	5	-10	7
cSum	~~-2~~ 0	3	7	6	11	1	8
maxSum	-2	3	7	7	11	11	11

Ex 2 :- [-20 10 -20 12 6 5 -3 8 -2]

	↓	↓	↓	↓	↓				
	-20	10	-20	12	6	5	-3	8	-2
cSum	~~-20~~ 0	10	~~-10~~ 0	12	18	23	20	28	26
maxSum	-20	10	10	12	18	23	23	28	28

Quiz 3 :- Tell the output using Kadane's Algorithm.

A[] = { -2 3 4 -1 5 -10 7 }

	↓	↓	↓	↓	↓	↓	↓
	-2	3	4	-1	5	-10	7
cSum →	~~-2~~ 0	3	7	6	11	1	8
maxSum →	-2	3	7	7	11	11	11

Ans = 11

Ex :-

	↓	↓	↓	↓
	-10	-3	-1	-2
cSum	~~-10~~ 0	~~-3~~ 0	~~-1~~ 0	-2
maxSum	-10	-3	-1	-1

PSEUDO CODE

```
cSum = 0
mSum = -∞
for (i = 0; i < n; i++) {
    cSum += A[i];
    if (cSum > mSum) {
        mSum = cSum;
    }
    if (cSum < 0) {
        cSum = 0;
    }
}
return mSum;
```

TC - O(N)

SC - O(1)

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

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

Ex arr = [0 0 0 0 0 0 0]

0	1	2	3	4	5	6
	+3	+3	+3	+3	+3	+3
				+2	+2	+2
			+1	+1	+1	+1
0	3	3	4	6	6	6

Queries

idx	Val
1	3
4	2
3	1

⇐ Ans

Brute force

↳ For each query, iterate from i to N-1 & add Val in array.

TC - O(QxN)

SC - O(1)

↳ **OPTIMIZATION**

① Let observe what happens in Prefix Sum

arr = [a_0 a_1 a_2 a_3 a_4]

a_0	a_0	a_0	a_0	a_0
	a_1	a_1	a_1	a_1
		a_2	a_2	a_2
			a_3	a_3
				a_4

Learning:- when we create prefix sum the that value (A[i]) propages from i to N-1

Ex arr = [0 0 0 0 0 0 0]
0 1 2 3 4 5 6
+3 +1 +2

queries

idx	val
1	3
4	2
3	1

↳ compute Prefix sum of arr.

= [0 3 3 4 6 6 6]

PSEUDO CODE

```
arr[N];
for( i=0; i<Q; i++){
    idx;
    val;
    arr[idx] += val;
}
// Creating prefix sum.
for( i=1; i<N; i++){
    arr[i] = arr[i-1] + arr[i];
}
```

TC → O(Q+N)
SC → O(1) → Using given array.

Question3:- Given an integer array A such that all the elements in the array are 0. Return the final array after performing multiple queries.

Query (i, j, x) :- Add x to all the elements from index i to j

Ex arr[] = [0 0 0 0 0 0 0]

Queries

i	j	Val
1	3	2
2	5	3
5	6	-1

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

0 2 5 5 3 2 -1

Quiz 4 Find the final array after performing 4 given queries on array of size 8.

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

arr[8] = [0 0 0 0 0 0 0 0]

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

-1 2 6 2 5 2 3 0

Brute Force => similar to previous question.

=> OPTIMIZED

arr[8] = [0 1 2 3 4 5 6 7]
-1 +3 +4 -4 +3 -3 +1 -3

-1 2 6 2 5 2 3 0

queries

	i	j	val
→	1	4	3
→	0	5	-1
→	2	2	4
→	4	6	3

Logic :-

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

PSEUDO CODE

```
arr[N];
for( K=0; K<Q; K++){
    i -> Starting
    j -> Ending , Val
    arr[i] += val
    if( j != N-1){
        arr[j+1] -= val;
    }
}
// Creating prefix sum.
for( i=1; i<N; i++){
    arr[i] = arr[i-1] + arr[i];
}
```

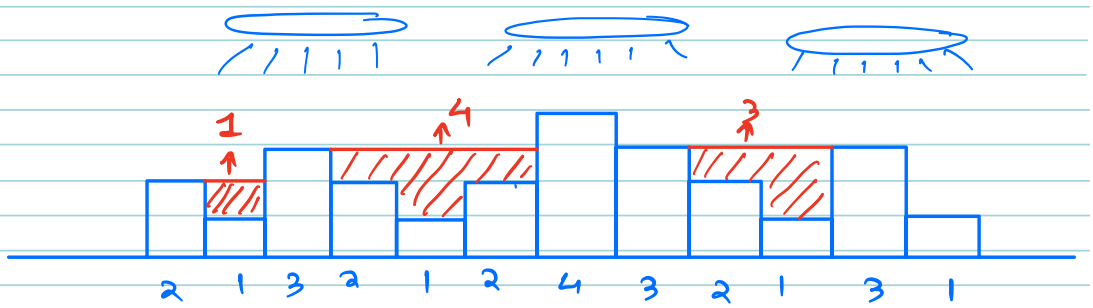
TC -> O(N+Q)

SC -> O(1)

Question 4 :- Given N Buildings with height of each building. Find the rain water trapped b/w the buildings.

Ex: arr[] =

0	1	2	3	4	5	6	7	8	9	10	11
2	1	3	2	1	2	4	3	2	1	3	1



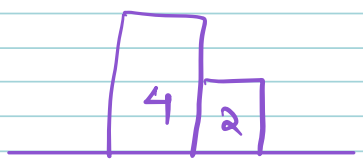
Ans = 8

OBSERVATIONS → [Need to focus on building of height 2]

IDEA

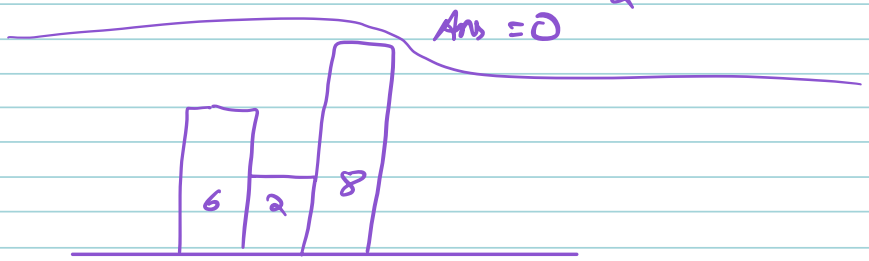
↳ How Much water will be collect on 2

Ans = 0

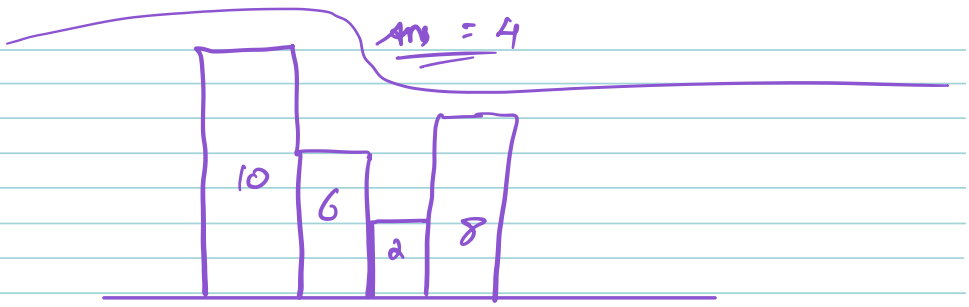


↳ How Much water will be collect on 2

Ans = 0

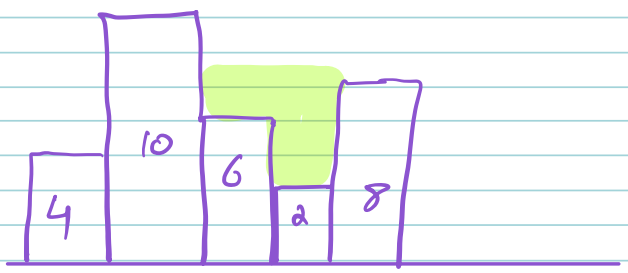


↳ How Much water will be collect on 2



↳ How Much water will be collect on 2

Ans = 6



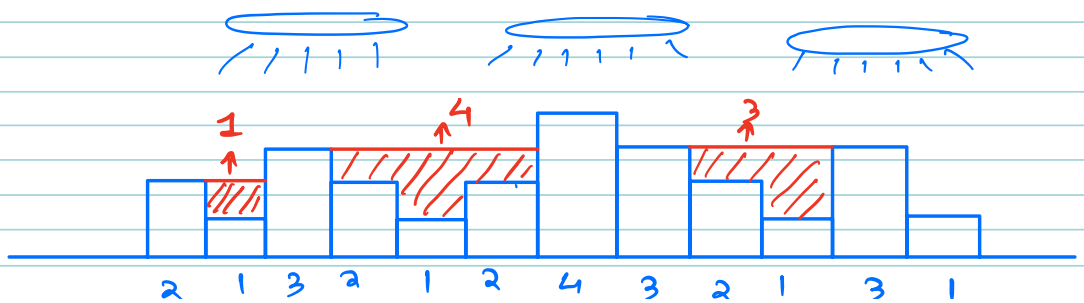
↳ How Much water will be collect on 2

Ans = 6

CONCLUSION

water on top of building = Min (LeftMax, RightMax) − H[i];

Ex: arr[] = 2 1 3 2 1 2 4 3 2 1 3 1 (indices 0 1 2 3 4 5 6 7 8 9 10 11)



LMax → 0 2 2 3 — — — — — — — —

RMax → 4 4 4 4 — — — — — — — —

Brute Force

↳ For i^{th} building find Left Max & Right Max. And use the above formula

PSEUDO CODE

```
ans = 0;                                   // TODO
for (i = 0; i < N; i++) {
    Left Max  = Max height (arr, 0, i-1);
    Right Max = Max height (arr, i+1, N-1);

    water = Min(Left Max, Right Max) - arr[i]

    if (water > 0) {
        ans += water
    }
}
```

TC → O(N²)
SC → O(1)

↳ OPTIMIZATION

Observation

↳ ① We need to optimize Left Max & Right Max computation.

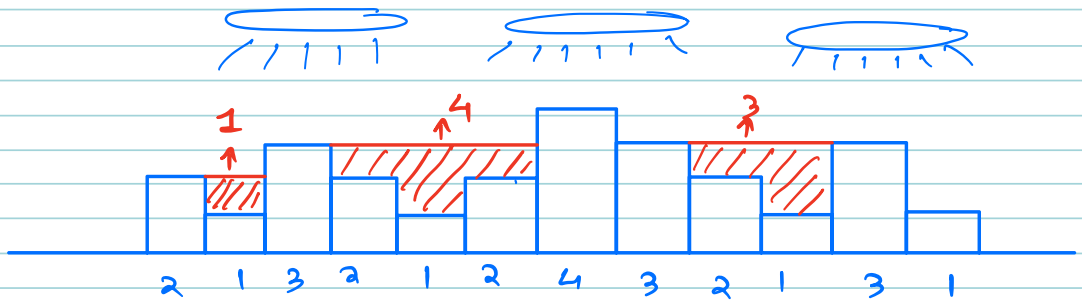
⑤ From 0 to curr, we want Max. The concept work from 0 to curr is Prefix.

So, this time we can use prefix Max.

But we will be needing Reverse also i.e. Suffix Max.

Ex. arr[] =

0	1	2	3	4	5	6	7	8	9	10	11
2	1	3	2	1	2	4	3	2	1	3	1



PfMax →	2	2	3	3	3	3	4	4	4	4	4	4
SfMax →	4	4	4	4	4	4	4	3	3	3	3	1

PfMax = Max from (0 to i)

SfMax = Max from (i to N-1)

PSEUDO CODE

PfMax Creation

```
PfMax[0] = arr[0]
for (i=1 to N-1) {
    PfMax[i] = Max(PfMax[i-1], arr[i]);
}
```

SfMax creation

```
sfMax[N-1] = arr[N-1];
for( i = N-2 to 0 ) {
    sfMax[i] = Max( sfMax[i+1], arr[i]);
}
```

```
ans = 0;
for(i = 1; i < N-1; i++) {
    LeftMax = pfMax[i-1];
    RightMax = sfMax[i+1];

    water = Min( LeftMax, RightMax) - arr[i]

    if (water > 0) {
        ans += water
    }
}
```

TC - O(N)

SC - O(N) ← compromised