Arrays - 1 : One dimensional

→ 2018 Bits Pilani CSE
→ 2021 April
→ Intuit SDE 2
→ Kotlin Springboot DynamoDb.

→ Max Subarray Sum
→ Range Queries
→ Trapping Rain water.

Q1> Given A[N]. Find the max subarray sum for subarrays starting from index 0

	0	1	2	3	4	5
A :	10	-5	7	8	-1	2

10

10 -5

10 -5 7

10 -5 7 8

10 -5 7 8 -1

10 -5 7 8 -1 2 — 21

Brute Force : starting with 0

→ Generate all subarrays ↑ and take max sum

TC : $O(N^2)$

SC : $O(1)$

Optimized

Prefix sum

	0	1	2	3	4	5
A :	10	-5	7	8	-1	2
Prefix sum	10	5	12	20	19	21

↓ ↑

	10	-5	7	8	-1	2
sum 0	10	5	12	20	19	21
any ($-\infty$)	10	10	12	20	20	21

SC : $O(1)$

TC : $O(N)$

Q2> Given A[N], find max subarray sum

	0	1	2	3	4	5	6	7	8	9
A :	-1	2	3	-4	6	9	2	-1	8	3

A : [-7 4 3 -2 -8 -4 6 -2]

9 2 -1 8

2 -4 6

→ start with 0

-1 , -1 2 , -1 2 -4 , -1 2 -4 6

start with 1

2 , 2 -4 2 -4 6

start with 2

-4 -4 6

start with 3

6

TC : $O(N^2)$

SC : $O(1)$

TC : $O(N)$

SC : $O(1)$

Observations

a) All the vals are +ve

3 100 2 6

3 + 100 + 2 + 6

b) All the vals are -ve

-1 -2 -3 -4 -1

c) some are +ve and some are neg.

If val is positive include to ans

100 -1 -2 → Don't include -1

100 -1 20 → Include -1.

2	-3	4
→ 2	-1	3
	-1	
	0	

2	-1	4
2	1	5

	-5	10
OS	-5	5

-5		10
-5	0	10

A	5	6	7	-3	2	-10	-12	8	12	21
psum	5	11	18	15	17	7	~~-5~~ 0	8	20	41
ans	5	11	18	18	18	18	18	18	20	41

A		-20	10	-12	6	5	-3	8	9
total	0	~~-20~~ 0	10	~~-2~~ 0	6	11	8	16	25
ans	(-∞)	-20	10	10	10	11	11	16	25

KADANE's Algo.

A[N]

TC : N

SC : 1

```
maxSum = -∞
total = 0

for (i -> 0 to n-1) {
    total += A[i]
    maxSum = max(maxSum, total)
    if (total < 0) {
        total = 0
    }
}
return maxSum
```

Q3> Given A[N]. All elements of array are 0
Given Q queries {idx, val}
Add this from idx till end.

Eg:

idx	val
1	3
4	2
2	1
1	-1

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

Brute force :

Execute the queries one by one and update the array.

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

	idx	val
0	1	3
1	4	2
2	2	1
3	1	-1

0	1	2	3	4	5
0	0	0	0	0	0
	3				
				2	
		1			
0	2	1	0	2	0
0	2	3	3	5	5

→ psum

TC: (N+Q)
SC: 1

BREAK 5 miny

22 : 12

Extension queries {start, end, val}

start	end	val
2	4	2
1	3	1
0	2	3
3	5	4

0	1	2	3	4	5
0	0	0	0	0	0
0	0	2	2	2	0
0	1	3	3	2	0
3	4	6	3	2	0
3	4	6	7	6	4

start	end	val
2	4	2
1	3	1
0	2	3
3	5	4

0	1	2	3	4	5
0	0	0	0	0	0
		+2			-2
	+1			-1	
+3			-3		
			+4		
3	1	2	1	-1	-2
3	4	6	7	6	4

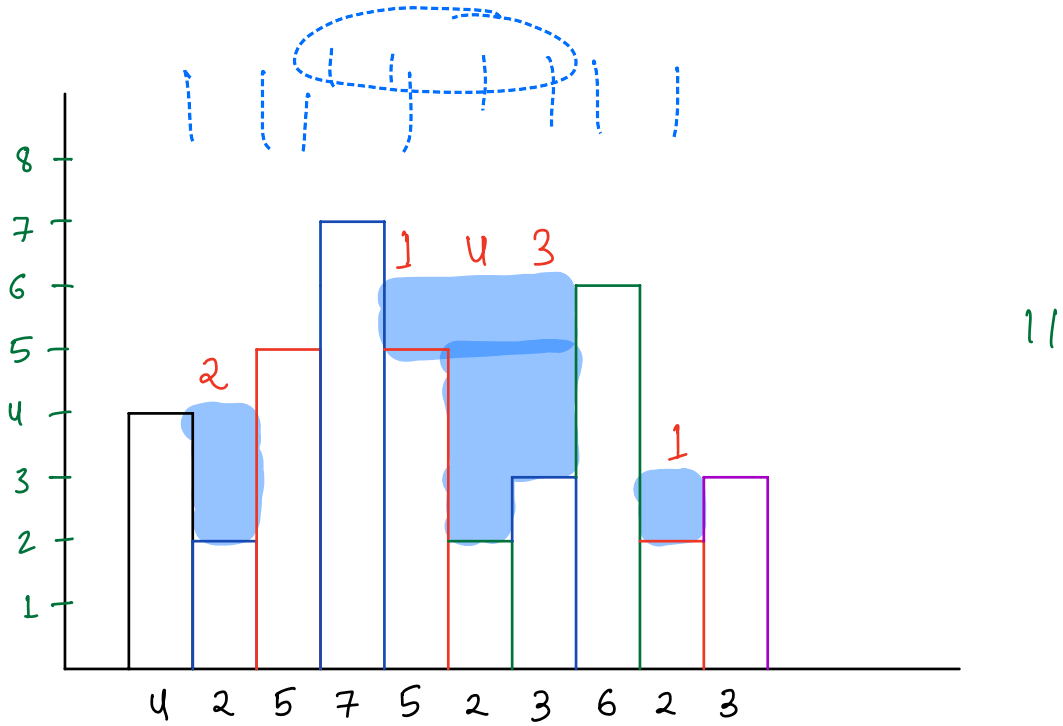
Pseudo

```
A[N]
starts [Q]
ends [Q]
vals [Q]

for ( i → 0 → Q-1 ) {

    start = stars [i]
    end   = ends [i]
    val   = vals [i]
```

TC : Q+N
SC : 1

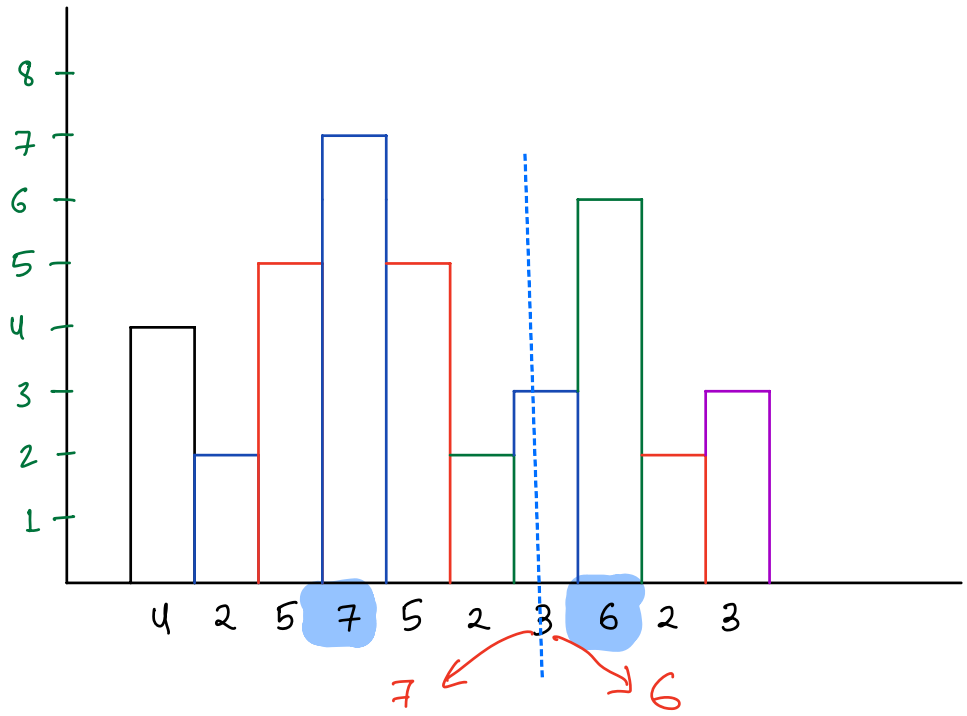
```
        A[start] += val

        if (end + 1 < A.length)
            A[end+1] -= val

    }
    // Prefix sum A

for (i → 1 to N-1) {

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

    }

return A.
```

Q> Rain Water Trapping



11

Given A[N] each value represents building height calculate the total water accumulated coz of rain



height

max Left Height , max Right Height

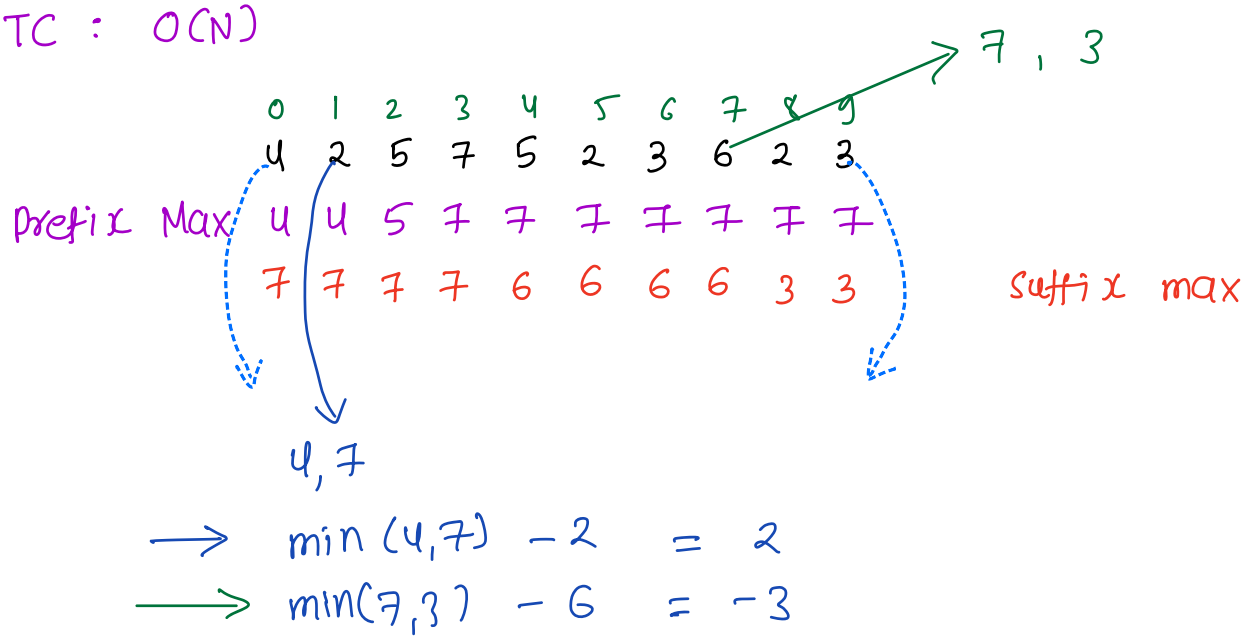
mLH mRH

water = min(mLH, mRH) − height

Bruteforce : $O(N^2)$

At each index calculate max on left and max on right and use

TC : $O(N)$



Pseudo

TC : N
SC : N

```
prefix[N] // Init
suffix[N] // Init
maxRight = maxLeft = -∞
ans = 0
for( i → 0 → N-1 ) {
    maxLeft = max( maxLeft, A[i] )
    prefix[i] = maxLeft
}

for( i → N-1 → 0 ) {
    maxRight = max( maxRight, A[i] )
    suffix[i] = maxRight
}

for ( i → 1 to N-2 ) {
    mLH = prefix[i-1]
    mRH = suffix[i+1]
    water = min(mLH, mRH) - height
    if (water > 0) {
        ans += water
    }
}

return ans
```