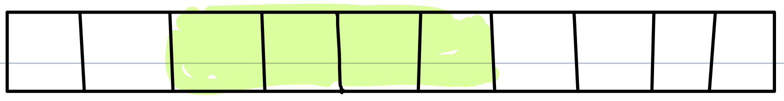
Maximum Subarray Sum



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

$\Downarrow$

28

-7 4 3 -2 -8 6 -4 2

$\Uparrow$

Basic :- consider all subarrays $\longrightarrow$ $O(n^3)$

$\downarrow$

$O(n^2)$

$\downarrow$

$\frac{n(n+1)}{2}$

$O(n)$

# positive integers

$\Downarrow$

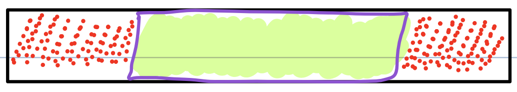
sum of all elements

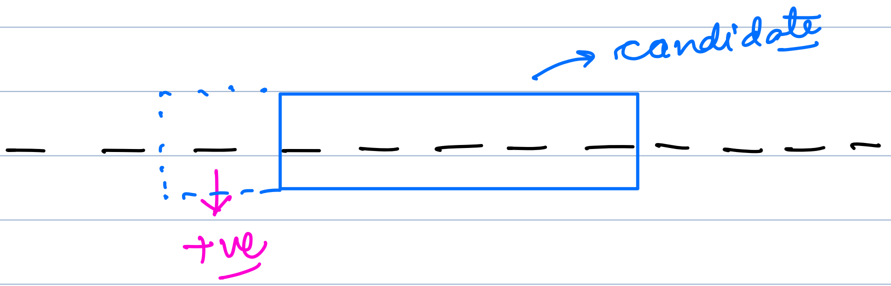
# all are -ve

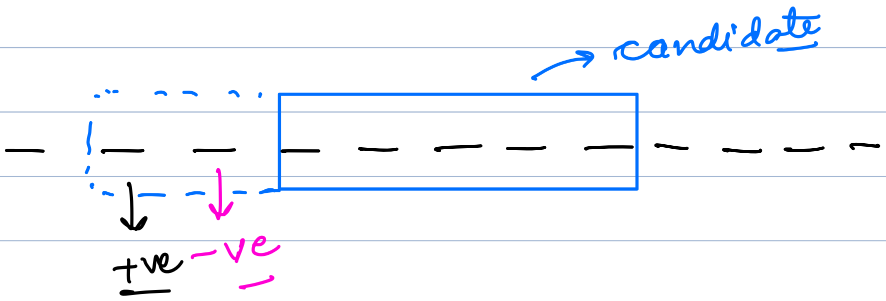
-9 -6 -3 -15 -12

— maxvalue of arr

$\Uparrow$







+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=-∞ → INT_MIN	5	11	18	18	18	18	18	18	20	41	41	44

sum subarr

	-20	10	-12	6	5	-3	8	9
sum=0	~~-20~~ 0	10	~~-2~~ 0	6	11	8	16	25
ans =-∞	-20	10	10	10	11	11	16	25

	-20	-10	-6	-15	-2	-30
sum =0	~~-20~~ 0	~~-10~~ 0	~~-6~~ 0	~~-15~~ 0	~~-2~~ 0	-30
ans =-∞	-20	-10	-6	-6	-2	-2

{
# update sum
# update ans
# reset sum to 0 if -ve

ans_s, ans_end
s, end

Kadane's → O(n)

```
sum=0 , ans = -∞
for( int i=0; i<n; i++)
{
    sum += arr[i];
    ans = max(ans, sum);
    if (sum<0)
        sum=0;
}
```

Extension :- subarray ? → HW

{Beggar's outside temple}

array of size N. All elements are zero.

q queries → { index, value }
↓
Add this 'value' starting from 'index' till end.

q=4

index	value
1	3
4	2
2	1
1	-1

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

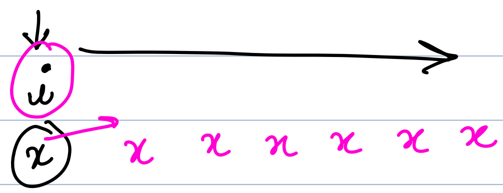
Return the final array!

Basic :- for every query just go and add!

$\boxed{T.C: O(q * n)}$

space O(1)

prefix sum



2 1 3 6 9

2 2+1 2+1+3 2+1+3+6 2+1+3+6+9

index	value
1	3
4	2
2	1
1	-1

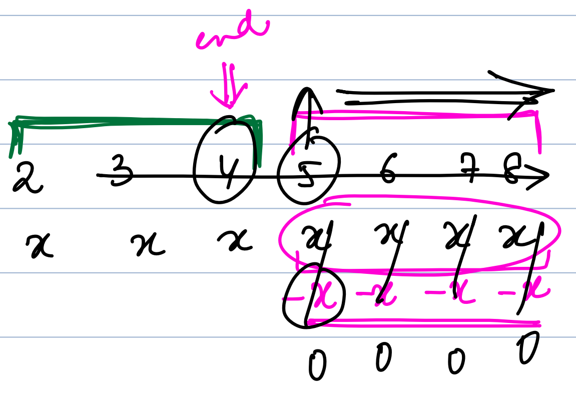
0	1	2	3	4	5	6
0	0	0	0	0	0	0
0	$\overset{+3}{3}$	0	0	0	0	0
0	3	0	0	2	0	0
0	$\underset{-1}{3}$	1	0	2	0	0
0	2	1	0	2	0	0
0	2	3	3	5	5	5

$$\begin{array}{r} 3 \\ -1 \\ \hline 2 \end{array} \ldots\ldots$$

q {start, end, value}

start	end	value
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 + 1
please add x	please add (-x)

start	end	value
2	4	2
1	3	1
0	2 +1	3
3	5	4

6

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

-3
$+4$

1

already given

```
arr[N] = {0}

for( i=0; i<q; i++)
{
        // start, end, value

        arr[start] += value;
        if( end+1 < n)
            arr[end+1] -= value;
}
// take pf sum
```

O(1)

O(q)

O(q+N)

S.C: O(1)

whenever the q^n asks you to give the ans after all queries

$\downarrow$

Think about how can delay my operation to make queries faster

what if initial values were not 0 $\rightarrow$ $\longrightarrow$

+ initial amount

0	1	2	3	4	5
2	3	1	6	9	-5
	+5	+5	+5		
2	8	6	11	9	-5
2	8	9	14	12	-5

1	3	5
2	4	3

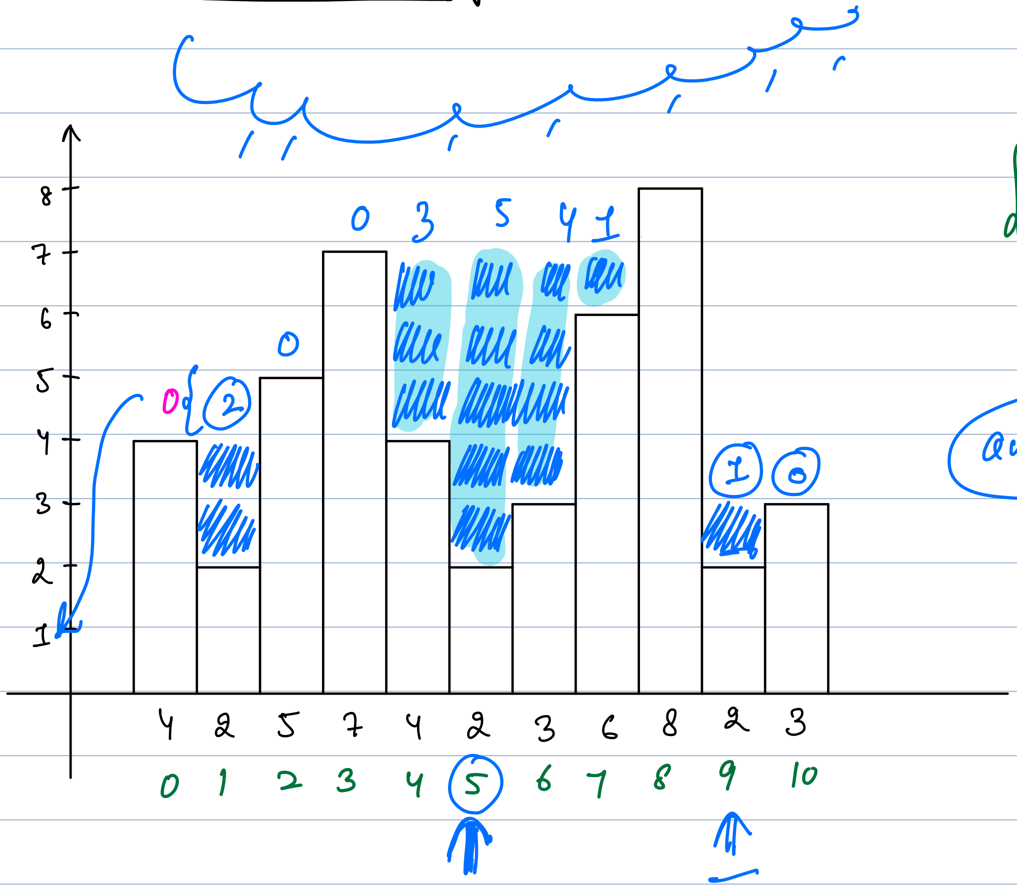
0	1	2	3	4	5
2	3	1	6	9	-5
2	8	1	6	4	-5
2	8	4	6	4	-8
2	10	14	20	24	16

4

1	3	5
2	4	3

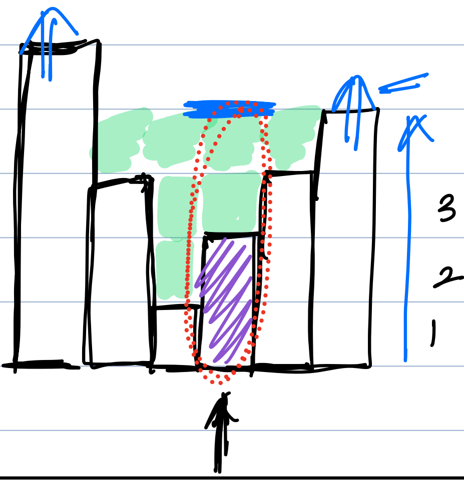
10:40

Rain water Trapping



{Total water accumulated}

ans = 16



{left bar :- largest on left
right ↳ largest on right

$$\boxed{\text{water} = \min(\text{leftmax}, \text{rightmax}) - \text{height}}$$

traverse & find Lm & rm

T.C: $O(n^2)$

$pf[i] = pf[i-1]$ for sum(0–i)

pfsum

$\boxed{4 \quad 2 \quad 5 \quad 7 \quad 4 \quad 2 \quad 3 \quad 6}$ 8 2 3

$lm(0-i) = 7$

$i+1 = 8$

$lm(0-i+1) = 8$

$$\boxed{lm[i] = max(lm[i-1], arr[i])}$$

i ⤵

lm[]	4	2	5	7	4	2	3	6	8	2	3
	4	4	5	7	7	7	7	7	8	8	8

rm

$i \longrightarrow n-1$ $\quad$ $i+1 \longrightarrow n-1$ $\quad$ reverse

$rm[i] = max(rm[i+1], arr[i])$

$$\boxed{water \mathrel{+}= min(lm[i], rm[i]) - height[i];}$$

T.C: O(N)
S.C: O(N) → O(1) → Two pointer

Try to use only 1 array

2D Matrix
pfsn / arr