Todays Content.

- → Max Subarray Sum
- → Zero Queries.

Max Subarray Sum

Given arr[N] elements, return max subarray sum

: continous part of an array.

Constraints.

$1 <= N <= 10^5$

$-10^6 <= arr[i] <= 10^6$

Ex:

	0	1	2	3	4	5	6	
arr[7] =	3	2	-6	8	2	-9	4	ans = 10

Ex:

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

Ideas: Generate all subarray sums & get max of them.

↳ a. Using 3 loops.

TC: $\underset{\text{subarray}}{\underset{\downarrow}{O(N^2)}} * \underset{\text{Iterate \& get sum}}{\underset{\downarrow}{O(N)}} = O(N^3)$ SC: O(1)

```
for(s=0; s<=N-1; s++){
    for(e=s; e<=N-1; e++){
        currentSum = 0;
        for(i=s; i<=e; i++){
            currentSum += arr[i];
        }
    }
```

b. Optimization using Pfsum array:

TC: $O(N^2)$ SC: $\underset{\text{Pfsum array}}{\underset{\downarrow}{O(N)}}$

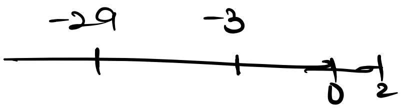
c. Optimized. (using carry forward)

TC: $O(N^2)$ SC: O(1)

Note: Please refer to Subarray class.

$N=10^5$ $N^2 = 10^{10}$ → 100% TLE.

Optimization:



Case I: If all arr[] > 0

arr[] : 4 2 1 6 7

ans = 20 // add all of the elements.

Case II: If all arr[] < 0

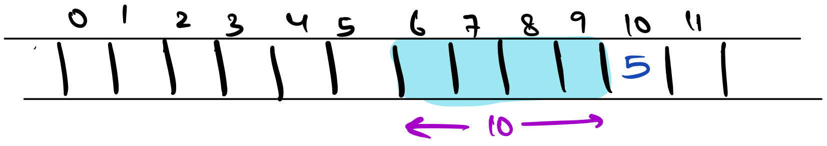
arr[] : -4 -8 -9 -3 -5

ans = -3 // max of arr[].

Case III: arr : -ve +ve -ve.

ans = sum of all +ve number

Case IV:

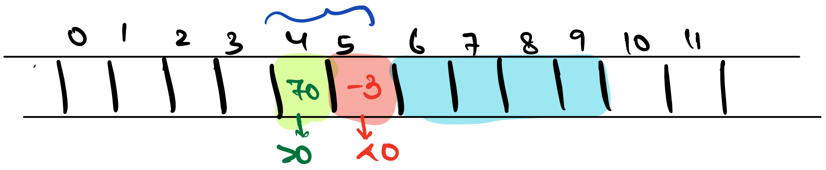


Ass1 : [6 -9] is the max subarray sum.

Ass2 : arr[10] > 0 : max subarray : [6 - 10].

Case V:

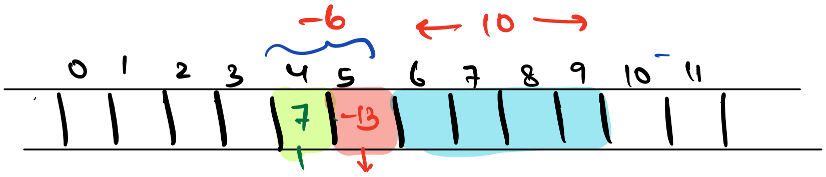
arr[4] + arr[5] > 0



Ass1 : [6 -9] is the max subarray sum.

arr[5] < 0

arr[4] > 0 || arr[4] + arr[5] > 0



Idea: If sum is +ve, carry it forward.

	0	1	2	3	4	5	6	7	8	9	10
arr[] =	5	6	7	-3	2	-10	-12	8	12	-4	7.
s = 0	5	11	18	15	17	7.	-5 0	8	20	16	23
ans = 0	5	11	18	18	18	18	18	18	20	20	23

	0	1	2	3	4	5	6	7
arr[] =	-3	6	-2	4	-5	2	4	-6
s = 0	-3 0	6	4	8	3	5	9	3
ans =	-3	6	6	8	8	8	9	9

	0	1	2	3	4
arr[] :	-3	-6	-40	-30	-2
sum = 0	-3 0	-6 0	-40 0	-30 0	-2
ans	-3	-3	-3	-3	-2

Algo: Kadane's algo

```
long  subSum( int[] arr, int N){
    long  sum = arr[0]
    long  ans = arr[0]

    for(i=1 ; i<N; i++){
        if(sum <0)  sum =0
         sum = sum + arr[i];
         if( sum > ans)  ans = sum;      || ans= Math.max (ans , sum)
    }
     return  ans;
}
```

Zero Queries.

Given arr[N] = 0 . , all zeroes & Q Queries.
For every Query : Given (s, v) Add element v to all index elements from [s ..... N-1] ,
Once all queries are done return final ans

Constraints.

$1 <= N, Q <= 10^5$

$1 <= V <= 10^6$

$0 <= S < N$

Ex : N = 7

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

final ans[].

Queries : 3

s	v
1	3
4	-2
3	1

Ex : N = 7

	0	1	2	3	4	5	6
arr[7] =	0	0	0	0	0	0	0
			+6	+6	+6	+6	+6.
	-1	-1	-1	-1	-1	-1	-1
				+2	+2	+2	+2
						+4	+4.
				+3	+3	+3	+3
	-1	-1	5	10	10	14	14.

final ans.

s[]	v[]
2	6
0	-1
3	2
5	4
3	3

idea1: for every query iterate from [s .... N-1] & add v.

TC: O(N * Q)

$N = 10^5$ $Q = 10^5$

$N * Q = 10^{10}$ → TLE.

Ex : N = 7

	0	1	2	3	4	5	6
arr[7] =	0	0	0	0	0	0	0
		3		1	-2.		
	0	3	0	1	-2	0	0
pf[] :	0	3	3	4	2	2	2.

Queries : 3

S	V	
1	3	: update
4	-2	: update.
3	1	: update.

Once all updates are done.
Apply pfsum[].

5

Ex : N = 7

	0	1	2	3	4	5	6
arr[7] =	0	0	0	0	0	0	0
	-1		6	2+3 = 5		4.	
	-1	0	6	5	0	4	0
pf[] =	-1	-1	5	10	10	14	14.

Queries : 4

S	V	
2	6	: update.
0	-1	: update.
3	2	: update.
5	4	: update
3	3	: update

```
zeroQ( int N , int s[Q] , int v[Q]){        TC: O(N+Q)
    long arr[N] = {0} →O(N)   // initialize all elements with 0.
    for(i=0 ; i< Q ; i++)   ——→ TC: O(Q)
        // i^th Query information is s[i] & v[i]
        arr[s[i]] = arr[s[i]] + v[i];
    }
                                                   SC: O(N)

    Apply pf sum on the arr[]   ——→ TC: O(N)
    return arr[N]
}
```

Zero Queries : 2

Given arr[N] =0 , all zeroes & Q Queries.

For every Query : Given (s, e, v) Add element v to all index elements from [s ..... e] .

Once all queries are done return final arr

Constraints.

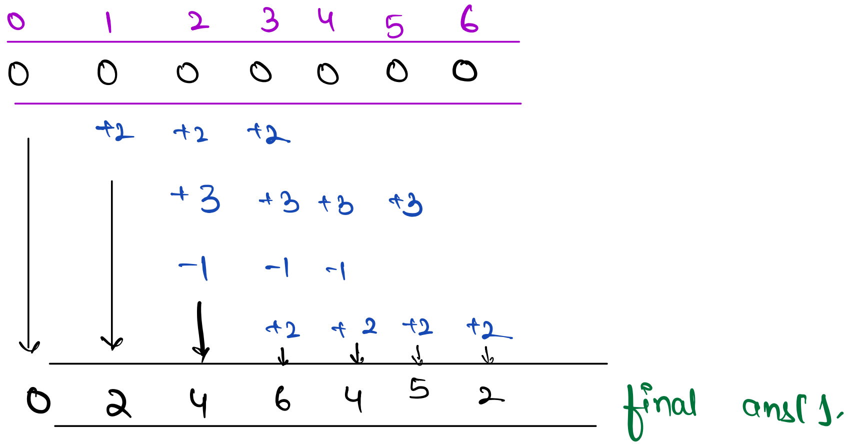
$1 <= N, Q <= 10^5$

$1 <= V <= 10^6$

$0 <= S < N$

Ex : N=7

arr[7] =



Queries

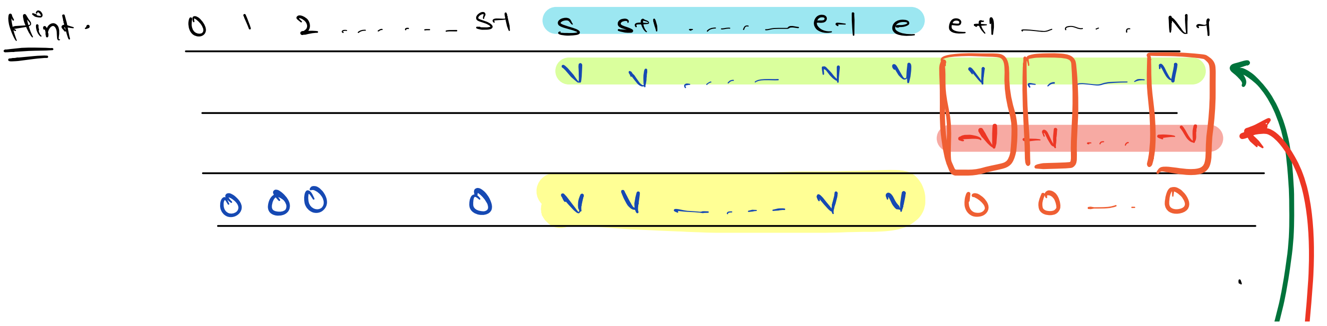
s	e	v
1	3	2
2	5	3
2	4	-1
3	6	2

idea 1: For every query iterate from [s e] and add v

TC: O(N * Q) = // N=10^5 Q=10^5 N*Q =10^{10} → TLE.

(Time limit Exceeded)

Hint.



0 1 2 .... s-1 s s+1 ...... e-1 e e+1 ..... N-1
v v ..... v v v v
-v -v ... -v
0 0 0 0 v v ..... v v 0 0 .. 0

Query :

s e v

1. Add v from [s ... N-1] → green part

Query (s , +v)

2. Add -v from [e+1 ... N-1] → red part

Query (e+1 , -v)

```
zero Q2 ( int N , int[Q] s, int [Q] e , int[Q] v){    TC:O(N+Q)

    long arr[N] = { 0 }
    for(i=0 ; i< Q ; i++)                       → O(Q)
        // ith Query inf: s[i] e[i] v[i]
        st = s[i] , end = e[i] , val= v[i]
        arr[st] = arr[st] + val;
        if( end+1 < n){
            arr[end+1] = arr[end+1] - val;
        }
    }

    Apply pfsum on arr[];                       → O(N)
    return arr;
}
```

Q3 Given arr[N]

Create pM[N] s.t. pM[i] = max of all elements from [0.....i]

	0	1	2	3	4	5
ar[6] =	1	-6	3	3	8	7
max =	1	1	3	3	8	8
pM[6] =	1	1	3	3	8	8

ar[5] =	3	-2	6	2	8
max =	3	3	6	6	8
pM[6] =	3	3	6	6	8

```
int[] constructpfmax (int[] arr){          TC: O(N)
    N= arr.length;
    int[] pfMax[N]
    pfMax[0] = arr[0]
    max = arr[0].
    for(i=1; i<N; i++){
        max = Math.max(max, arr[i]);
        pfMax[i] = max;
    }
    return pfMax[];
}
```

4Q Given arr[N]

Create sM[N] s.t. sM[i] = max of all elements from [i.....N-1]

	0	1	2	3	4	5	6
ar[7] =	3	10	6	7	0	2	-1
max	10	10	7	7	2	2	-1
sM[i] =	10	10	7.	7.	2	2.	-1

```
int[] construct sfmax (int[] arr){.        TC: O(N)
    N= arr.length;
    int[] sfMax[N]
    sfMax[N-1] = arr[N-1]
    max = arr[N-1]
    for(i= N-2 ; i>=0 ; i--) {
        max = Math.max(max, arr[i]);
        sfMax[i] = max;
    }
    return sfMax[] ;
}
```