Suryansh Gupta, Research - SDE II at Microsoft Research

2021 grad, IIEST Shibpur, former SWE - III (L4) at Google, former competitive coder.

Former intern at Microsoft & Directi

How to solve a problem ?

Ques 1:- You are a cryptocurrency trader and you have a ML model which predicts change in price of crypto for next N days.

You want to hold this crypto once for a consecutive sequence of days to get max profit.

0	1	2	3	4	5	6
-2	3	4	-1	5	-10	7

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

Say hold for 0 to 2 index, profit = -2 + 3 + 4 = 5

" " " 1 to 4 " " = 3 + 4 - 1 + 5 = 11

ans = 11

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

0	1	2	3	4	5	6
-3	4	6	8	-10	2	7

A = [-3 4 6 8 -10 2 7]

ans = 4 + 6 + 8 = 18

Quiz 1:- A = [4 5 2 1 6]

0	1	2	3	4
4	5	2	1	6

ans = 4 + 5 + 2 + 1 + 6 = 18

Quiz 2:- A = [-4, -3, -6, -9, -2]

0	1	2	3	4
-4	-3	-6	-9	-2

ans = -2

Approach 1:- Brute force

Check sum of all subarrays and pick the max.

total subarrays = $\frac{N * (N+1)}{2}$

Iterate in each subarray will take $O(N)$ time.

TC: $O(N^2 * N) \simeq O(N^3)$ SC: $O(1)$

```
ans = INT_MIN | A[0]
for(int i=0; i<N; i++){
    for(int j=i; j<N; j++){
        int sm = 0
        for(int k=i; k<=j; k++){
            sm += A[k]
        }
        ans = max(ans, sm);
    }
}
return ans
```

Approach 2:- Use Prefix Sum for calculating Subarray sum of each array.

TC for each subarray sum: $O(1)$

TC: $O(\underbrace{N}_{\text{Creating Prefix Array}} + N^2 \times 1) \simeq O(N^2)$ SC: $O(N)$

Approach 3 :- Use Carry forward for calculating subarray sum of each query.

TC for each subarray sum : O(1)

T C : $O(N^2)$ S C : O(1)

```
ans = INT_MIN or A[0]
for(int i=0; i<N; i++){
    int sm = 0
    for(int j=i; j<N; j++){
        sm += A[j]
        ans = max(ans, sm)
    }
}
return ans
```

Approach 4 :- Kadane's Algo

Case 1 :- If all elements are positive, ex 1 :- [1 4 5 8]

Sum of whole array

Case 2 :- If all elements are negative ex :- [-6 -2 -3]

Max element

Case 3 :- If positive is in between

A = [-ve -ve -ve +ve +ve +ve +ve -ve -ve -ve]

ans = take the positive chunk

Case 4 :- Positives at start [+ve +ve -ve -ve -ve -ve]

Case 5 :- Positives at end [-ve -ve -ve +ve +ve +ve]

→ ans = take the positive chunk

Case 6 :- Multiple positive chunks, in between -ve chunks

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

Let's take a darling example.

-2, started with -ve, so drop

+3, got happy, so continue

+4, got happy, so continue

-1, got unhappy but overall 3+4-1 = 6

+5, got happy, so continue 6+5 = 11

-10, got unhappy but overall 11-10 = 1

+7, got happy, so continue 1+7 = 8

	0	1	2	3	4	5	6	7	8
A = [	−20	10	−20	−12	6	5	−3	8	−2]

i	curSum	maxSum
		−20
0	−20	−20
reset	0	
1	10	10
2	−10	10
reset	0	
3	−12	10
reset	0	
4	6	10
5	6+5 = 11	11
6	11−3 = 8	11
7	8+8 = 16	16
8	16−2 = 14	16

Quiz 3:- A = [−2 3 4 −1 5 −10 7]

cur_sum = 0 + (−2) = ~~−2~~ 0 + 3 = 3 + 4 = 7 − 1 = 6 + 5 = 11 − 10 = 1 + 7 = 8

max_sum = ~~−2~~ 5 ~~7~~ 11

All negative: arr = [−8 −2 −6 −1 −3]

cur_sum = 0 −8 = ~~−8~~ 0 −2 = ~~−2~~ 0 −6 = ~~−6~~ 0 −1 = ~~−1~~ 0 −3 = −3

max_sum = ~~−8~~ ~~−2~~ −1

```
int maximumSubarraySum(arr[], N) {
    maxSum = arr[0]
    curSum = 0
    for(int i=0; i<N; i++) {
        curSum += arr[i]
        maxSum = max(maxSum, curSum)
        if(curSum < 0){
            curSum = 0
        }
    }
    return maxSum
}
```

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

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

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

Ex:- N = 7, A = [0 0 0 0 0 0 0]

	0	1	2	3	4	5	6
	0	0	0	0	0	0	0
		+3	+3	+3	+3	+3	+3
Query (1 3)					-2	-2	-2
Query (4 -2)				+1	+1	+1	+1
Query (3 1)	0	3	3	4	2	2	2

Quiz :- A = [0 0 0 0 0]
(0 1 2 3 4)

Query (1, 3) +3 +3 +3 +3

Query (0, 2) +2 +2 +2 +2 +2

Query (4, 1) +1

[2 5 5 5 6]

Approach 1: Bruteforce

Loop for each query and perform query operation.

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

Approach 2: Lazy

Try to avoid traversing till the end for each query.

→ Add x at arr[i] for each query

→ Take prefix once at the end

N = 7, A = [0 0 0 0 0 0 0 0]
(0 1 2 3 4 5 6)

+3 +1 −2

Query (1 3)

Query (4 −2) [0 3 0 1 −2 0 0]

Query (3 1) prefix [0 3 3 4 2 2 2]

$$A = [\ \underset{0}{0}\ \ \underset{1}{0}\ \ \underset{2}{0}\ \ \underset{3}{0}\ \ \underset{4}{0}\ ]$$

$+2$ at index 0, $+3$ at index 1, $+1$ at index 4

Query(1, 3) → -1 at index 3

Query(0, 2)

Query(4, 1)

Query(1, -1)

$[\ 2\ \ 2\ \ 0\ \ 0\ \ 1\ ]$

prefix $[\ 2\ \ 4\ \ 4\ \ 4\ \ 5\ ]$

TC: $O(Q + N)$ SC: $O(1)$

```
void performQueries(A[], B[][]) {
    for(i = 0 to B.length) {
        idx = B[i][0]
        val = B[i][1]
        A[idx] += val
    }
    for(int i = 1; i < A.length; i++) {
        A[i] += A[i-1]
    }
}
```

Query 3: Given an integer array A where every element is 0, return a final array after performing multiple queries.

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

Given $i <= j$

$N = 7$ $A = [$ 0 0 0 0 0 0 0 $]$ (indices 0 1 2 3 4 5 6)

+2 +2 +2

Q1 = (1, 3, 2) +3 +3 +2 +3

Q2 = (2, 5, 3) -1 -1

Q3 = (5, -1) $[$ 0 2 5 5 3 2 -1 $]$

Quiz 5: $N = 8$ $A = [$ 0 0 0 0 0 0 0 0 0 $]$ (indices 0 1 2 3 4 5 6 7)

+3 +3 +3 +3

-1 -1 -1 -1 -1

+4

+3 +3 +3

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

$[$ -1 2 6 2 5 2 3 0 $]$

Approach 1: Brute force

Go and apply each query.

TC: $O(Q * N)$ SC: $O(1)$

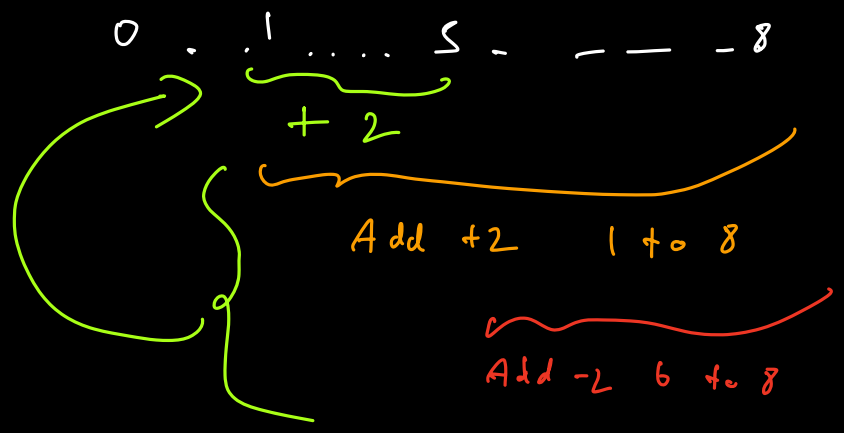
Approach 2: Explore Lazy Sum

Query(i, j, x) = Query(i, x) + Query(j+1, -x)

Add x to i to j

Add x to i to N-1

Add -x to j+1 to N-1



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

+3 −3 −1 +1 +4 −4 +3 −3

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

[-1 3 4 -4 +3 -3 +1 -3]

prefix [-1 2 6 2 5 2 3 0]

```
void performQueries(A[], B[][]){
    for(i = 0 to B.length){
        l = B[i][0]
        r = B[i][1]
        val = B[i][2]
        A[l] += val
        if(r+1 < n)
            A[r+1] -= val
    }
    for(int i = 1; i < A.length; i++){
        A[i] += A[i-1]
    }
```

§

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

N = 7 A = [0 0 0 0 0 0 0] (indices 0 1 2 3 4 5 6)

+2 at 1, −2 at 4

Q1 = (1, 3, 2) → +3 at 2

Q2 = (2, 5, 3) → −1 at 5

Q3 = (5, 6, −1) → −3 at 6

[0 2 3 0 −2 −1 −3]

(0 2 5 5 3 2 −1]

H.W: Say Everything same as previous problem except this is some initial values in array A. (A is not zero)

Merge Intervals

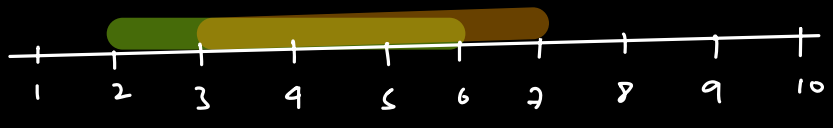
→ Defined by start and end time

→ Start ≤ end

Given some intervals, merge them if they intersect

I1 I2

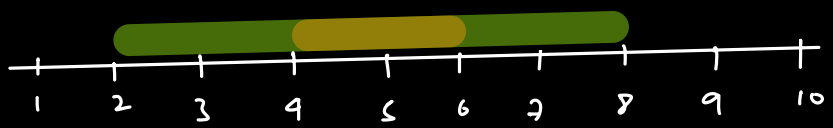
(2 6) (3 7)



Δ e

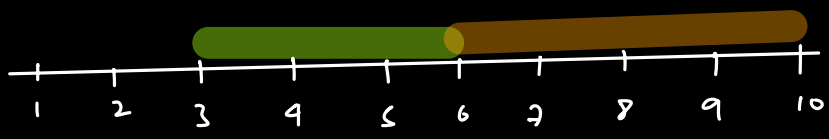
(2 7)

(2 8) (4 6)



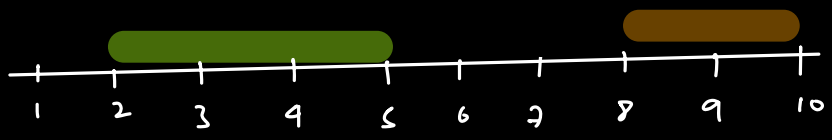
(2 8)

(3 6) (6 10)



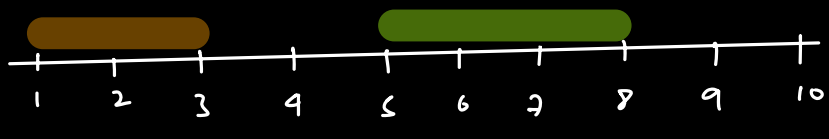
(3 10)

(2 5) (8 10)



no overlap

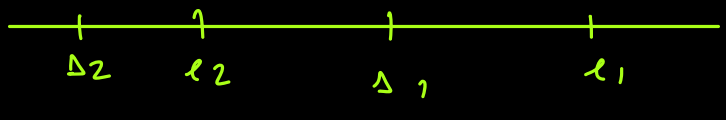
(5 8) (1 3)



no overlap

Non overlapping Condition: $I_1 = \{ \Delta 1 \quad \ell 1 \}$

$I_2 = \{ \Delta 2 \quad \ell 2 \}$




```
if (Δ2 > ℓ1  ||  Δ1 > ℓ2) {
    Non overlapping
}
else {
    Overlap

    Merged Start : min(Δ1, Δ2)
    Merged End : max(ℓ1, ℓ2)
}
```

Quiz $I_1 = \{ 3 \quad 8 \}$ $I_2 = \{ 5 \quad 12 \}$

Overlapping

Quiz $I_1 = \{6 \quad 10\}$ $I_2 = \{8 \quad 15\}$

Merged : $\{ \min(6, 8), \max(10, 15) \}$

: $\{ 6, 15 \}$

Ques 4 :- Given a collection of intervals in a 2D array format, where each interval is represented by [start, end].

The intervals are sorted by their start time.

Your task is to merge all overlapping intervals and return the resulting set of non-overlapping intervals.

Intervals = [(0 2), (1 4), (5 6), (6, 8), (7,10), (8,9), (12, 14)]
(0) (1) (2) (3) (4) (5) (6)

Output : [(0 4) (5 10) (12 14)]

Approach :- Iterate and keep merging

Interval In Hand	i	i^{th} Interval	Intersecting	Output
(0 2)	1	(1 4)	✓	
(0 4)	2	(5 6)	× ⇒	(0, 4)
(5 6)	3	(6 8)	✓	(0,4)

(5 8)	4	(7, 10)	✓	(0,4)
(5 10)	5	(8, 9)	✓	(0,4)
(5 10)	6	(12, 14)	× =>	(0,4) (5,10)
(12, 14)	=>			(0,4) (5,10) (12, 14)

Quiz :- Intervals: [(1, 10) (2 3) (4 5) (9 12)]
(0) (1) (2) (3)

Interval In Hand	i	i^{th} Interval	Intersecting	Output
(1 10)	1	(2 3)	✓	
(1 10)	2	(4 5)	✓	
(1 10)	3	(9 12)	✓	
(1 12)	=>			(1 12)

-> Create an array to store the merged interval
-> Start with 0^{th} interval in your hand
-> from i => 1 to n-1

if i^{th} interval intersect:
merge it with inhand interval
and update inhand interval

else
add interval In hand to output
take i^{th} interval in hand

```
currS = A[0][0] , currE = A[0][1]

for(i = 1 to N-1){

    if(A[i][0] <= currE){
        currE = max(currE, A[i][1])
    }
    else{
        ans.insert({currS, currE})
        currS = A[i][0]
        currE = A[i][1]
    }
}

ans.insert({currS, currE})
```

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

Doubts

"a a g a g" (0 1 2 3 4) Count no. of possible 'ag' you can make by choosing two distinct index of this String

ans: 5

(0 2) (0 4) (1 2) (1 4) (3 4)

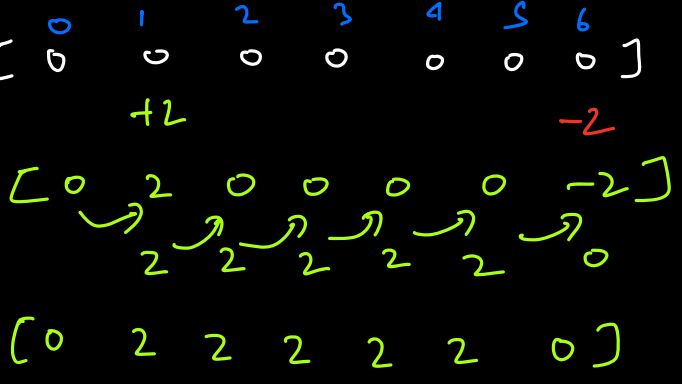
"a a g a g" (0 1 2 3 4)

aCount = 0 + 1 + 1 + 1

ans = 2 + 3 = 5

N = 7, A = [0 0 0 0 0 0 0]
(indices 0 1 2 3 4 5 6)

Query (1, 5, 2)




```
void performQueries(A[], B[][]){
    for(i = 0 to B.length){
        l = B[i][0]      // 1
        r = B[i][1]      // 5
        val = B[i][2]    // 2
        A[l] += val
        if(r+1 < N)
            A[r+1] -= val
    }
    for(int i = 1; i < A.length; i++){
        A[i] += A[i-1]
    }
}
```