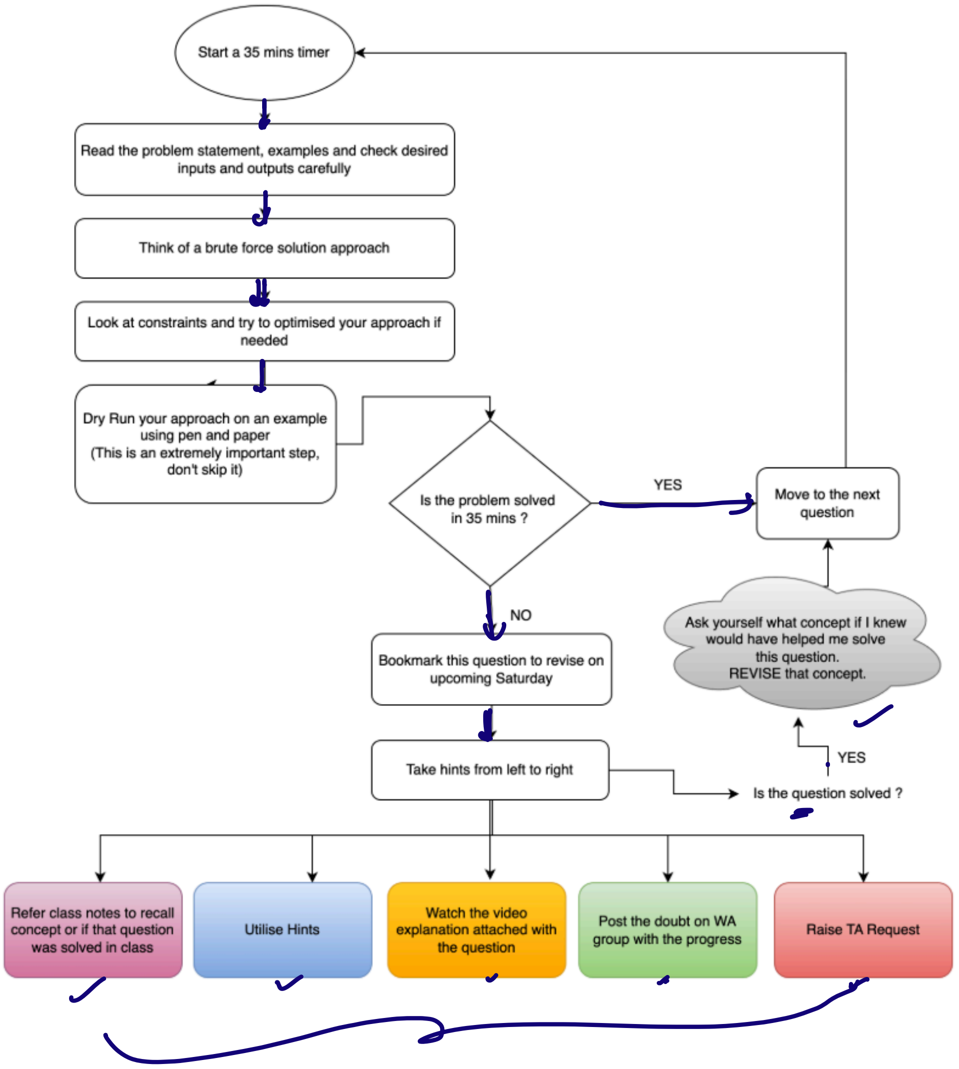

Start a 35 mins timer
Read the problem statement, examples and check desired inputs and outputs carefully
Think of a brute force solution approach
Look at constraints and try to optimised your approach if needed
Dry Run your approach on an example using pen and paper (This is an extremely important step, don't skip it)
Is the problem solved in 35 mins ?
YES
Move to the next question
NO
Bookmark this question to revise on upcoming Saturday
Ask yourself what concept if I knew would have helped me solve this question. REVISE that concept.
YES
Take hints from left to right
Is the question solved ?
Refer class notes to recall concept or if that question was solved in class
Utilise Hints
Watch the video explanation attached with the question
Post the doubt on WA group with the progress
Raise TA Request

Question: Maximum sum subarray

sum = 11

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

0 1 2 3 4 5 6

[3] [3, 4] [3, 4, -1] [3, 4, -1, 5]

3 7 6 11

N = 7

#subarrays = $\frac{7 \times 8}{2} = 28$

sum = 18

A: -3 4 6 8 -10 2 7

0 1 2 3 4 5 6

A : 4 5 2 1 6

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

Brute Force

→ Iterate over all subarrys, find their sum

→ Find max across all of them

T.C: $\frac{N(N+1)}{2} \times O(N) = O(N^3)$

S.C: $O(1)$

Prefix Sum

→ sum of each subarray : $O(1)$

T.C: $\frac{N(N+1)}{2} \times O(1) = O(N^2)$

S.C: $O(N)$

Carry forward

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

S.C: $O(1)$

Kadane's Algo

Prefixes of the array [1, 2, 3, 4, 5]

[1], [1, 2], [1,2,3] [1,2,3,4] [1,2,3,4,5]

Let's assume A[i....j] has maximum sum

[i i+1 i+2. . . . . .j-2 j-1 j]

⇓

Prefixes: [i], [i, i+1] [i, i+1, i+2]- - - - - -[i....j]

Ex2:

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

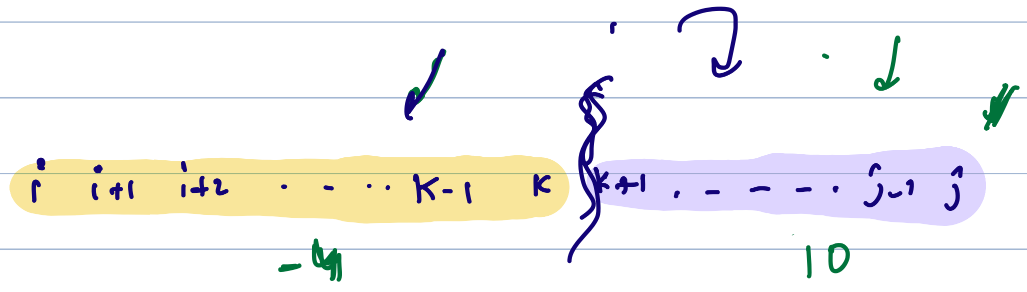
[4]	[4, -1]	[4, -1, 2]	[4, -1, 2, 1]
4	3	5	6

In the maximum sum subarray, all the prefixes will have positive sum.

Proof by contradiction:

Let's assume this claim is wrong.

then, some prefix of max sum subarray should have negative sume.



sum = $-4+10 = 6$

	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	✓
A:	1	−2	1	−3	4	−1	2	1	−5	4	
	0	1	2	3	4	5	6	7	8	9	

max_sum = −~~INF~~ ~~1~~ ~~4~~ ~~5~~ 6

curr_sum = ~~0~~ ~~1~~ ~~−1~~ ~~0~~ ~~1~~ ~~−2~~ ~~0~~ ~~4~~ 3 ~~5~~ ~~6~~
~~1~~ 5

[~~1 −2~~] = −1

~~[1 −3~~

[4 −1 2 1 −5 4

	↓	↓	↓	↓	↓	↓	↓
A:	−2	3	4	−1	5	−10	7
	0	1	2	3	4	5	6

maxSum = −~~INF~~ ~~−2~~ ~~7~~ 11

currSum = ~~0~~ ~~−2~~ ~~0~~ ~~3~~ ~~7~~ ~~6~~ ~~11~~ ~~1~~ 8

[−2] ✗

[3 4 −1 5 −10 7

	↓	↓	↓	↓	↓	↓
A =	−4	−3	−6	−9	−2	

marSum = −~~INF~~ ~~−4~~ ~~−3~~ −2

CurrSum = ~~0~~ ~~−4~~ ~~0~~ ~~−3~~ ~~0~~ ~~−6~~ ~~0~~ ~~−9~~ −2

[~~−4~~]

[~~−3~~]

[~~−6~~]

[~~−9~~]

[~~−2~~]

```
max_sum = INT_MIN
curr_sum = 0
  int maxSubArray(const vector<int> &A) {
          int n = A.size();
          int curSum = 0, maxSum = INT_MIN;
          for (int i = 0; i < n; i++) {
              curSum += A[i];
              maxSum = max(maxSum, curSum);
              if (curSum < 0) curSum = 0;
          }
     return maxSum;
    }
```

T.C: O(N)

S.C: O(1)

Question :

Problem Statement

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 numbers from index i to N-1

	Q1	Q2	Q3
Querry's:	(1, 3)	(4, -2)	(3, 1)

A :	0	0	0	0	0	0	0
	0	1	2	3	4	5	6
Q1:	0	3	3	3	3	3	3
Q2:	0	3	3	3	1	1	1
Q3:	0	3	3	4	2	2	2

Eg :

		0	1	2	3	4
A :		0	0	0	0	0
Q1 :	(1, 3)	0	3	3	3	3
Q2 :	(0, 2)	2	5	5	5	5
Q3 :	(4, 1)	2	5	5	5	6

Brute Force

For every query, iterate and update the array from [i, N-1]

T.C: O(N·Q)   N = 10^5

S.C: O(1)   Q = 10^5

Optimized Approach

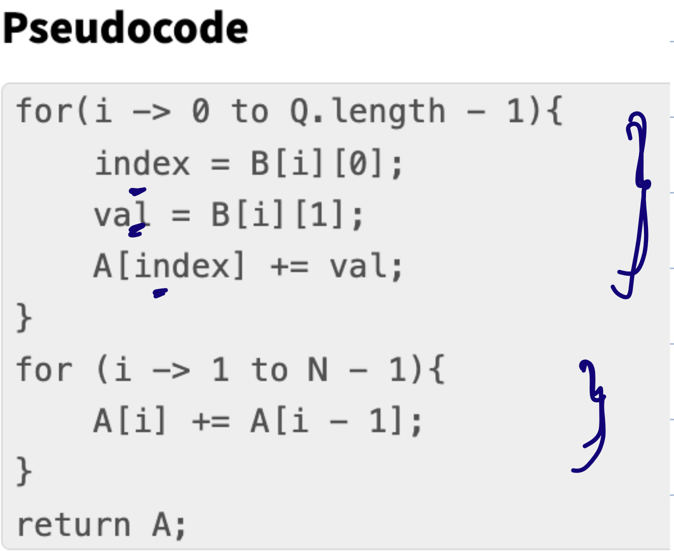
		0	1	2	3	4	
A:		0	0	0	0	0	
Q1 :	(1, 3)	0	+3	0	0	0	
✓ Q2 :	(0, 2)	2	3	0	0	0	
✓ Q3 :	(4, 1)	2	3	0	0	1	?

⇓

Compute Prefix Sum

↓

A :	2	3	0	0	1
PS :	2	5	5	5	6
	0	1	2	3	4



Pseudocode

```
for(i -> 0 to Q.length - 1){
    index = B[i][0];
    val = B[i][1];
    A[index] += val;
}
for (i -> 1 to N - 1){
    A[i] += A[i - 1];
}
return A;
```

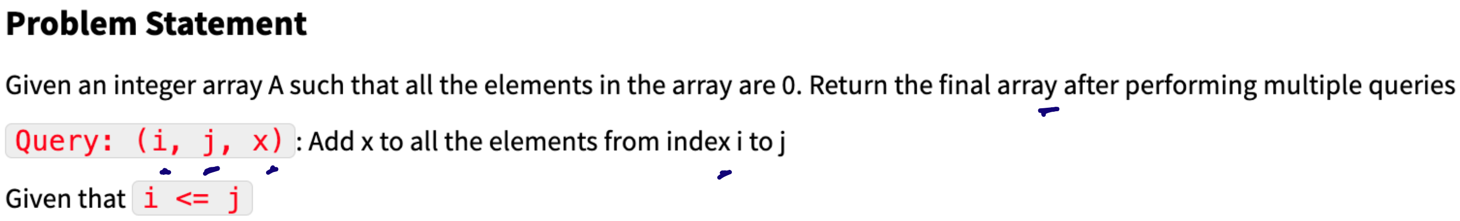
Iterate over querry O(Q)

Find prefix sum O(N)

T.C: O(N+Q)

S.C: O(1)

Question:



Problem Statement

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

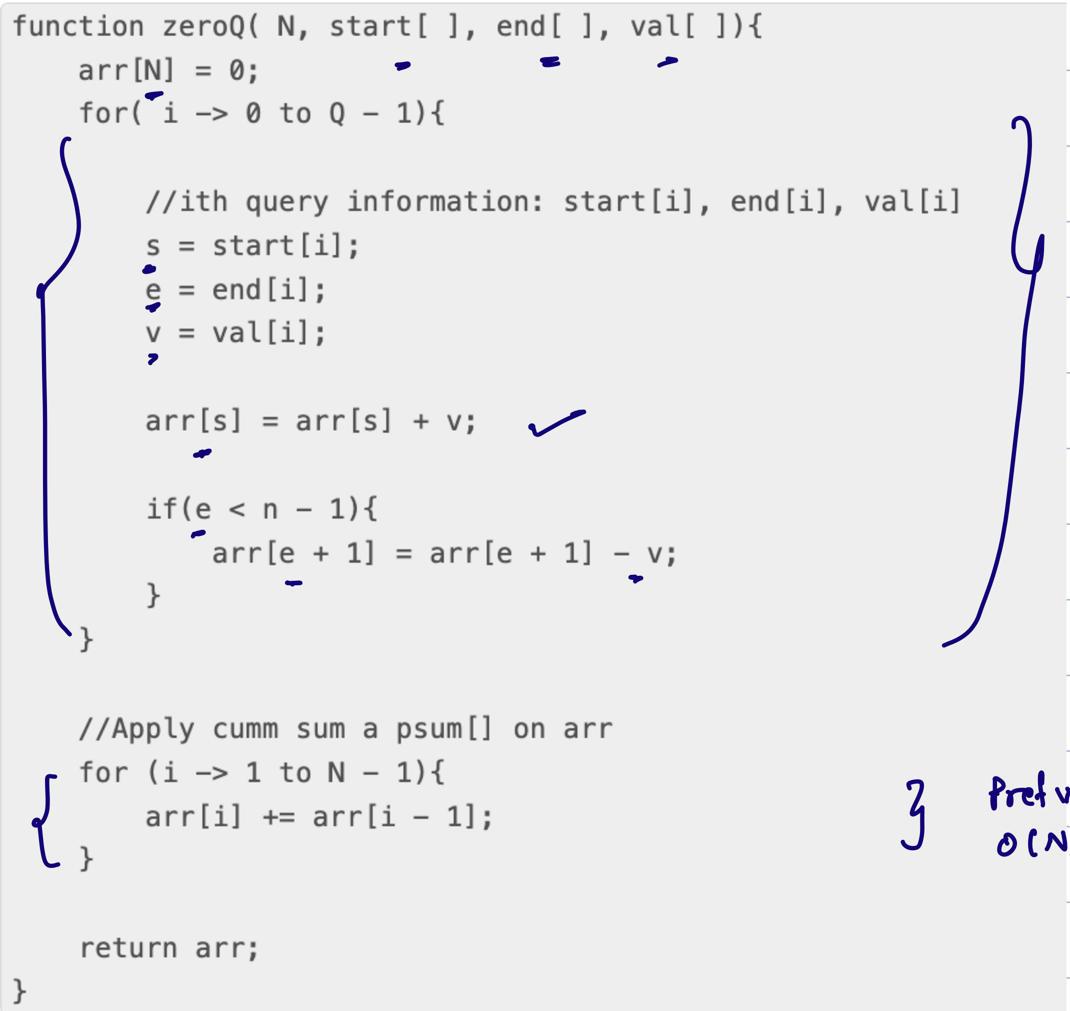
Given that `i <= j`

	0	1	2	3	4	5	6	7
A =	0	0	0	0	0	0	0	0
Q1: (1,4,3)	0	3	3	3	3	0	0	0
Q2: (0,5,-1)	-1	2	2	2	2	-1	0	0
Q3: (2,2,4)	-1	2	6	2	2	-1	0	0
Q4: (4,6,3)	-1	2	6	2	5	2	3	0

	i j x	0	1	2	3	4	5	6	7
A =		0	0	0	0	0	0	0	0
✓Q1:	(1,4,3)	0	3	0	0	0	-3	0	0
✓Q2:	(0,5,-1)	-1	3	0	0	0	-3	1	0
✓Q3:	(2,2,4)	-1	3	4	-4	0	-3	1	0
✓Q4:	(4,6,3)	-1	3	4	-4	3	-3	1	-3
PS:		-1	2	6	2	5	2	3	0

$A[i] = A[i] + x$ ✓

$A[j+1] = A[j+1] - x$



```
function zeroQ( N, start[ ], end[ ], val[ ]){
    arr[N] = 0;
    for( i -> 0 to Q - 1){

        //ith query information: start[i], end[i], val[i]
        s = start[i];
        e = end[i];
        v = val[i];

        arr[s] = arr[s] + v;

        if(e < n - 1){
            arr[e + 1] = arr[e + 1] - v;
        }
    }

    //Apply cumm sum a psum[] on arr
    for (i -> 1 to N - 1){
        arr[i] += arr[i - 1];
    }

    return arr;
}
```

Iterate over queries

O(Q)

Prefix Sum

O(N)

$\boxed{8:40}$

$e = N-1$ $A[e+1]$

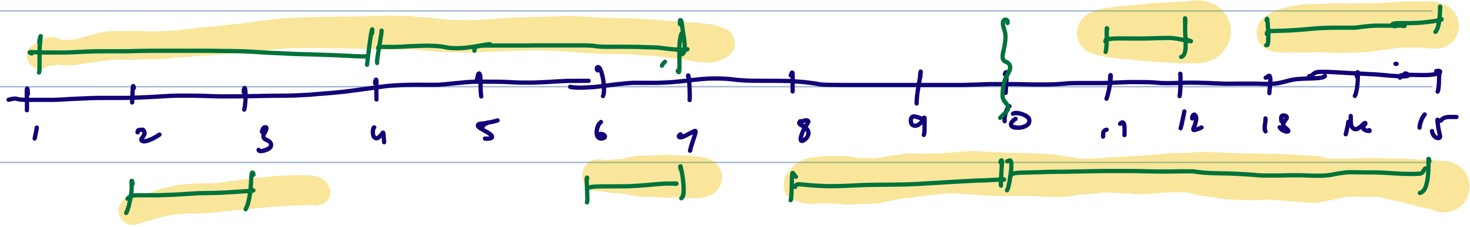
T.C: $O(N+\varphi)$

S.C: $O(1)$

Merge Interval

Interval : [start, end]   Start ≤ end
↓

Ex: [1 4] [2, 3] [4, 7] [6, 7]
[8, 10] [10, 15] [11, 12] {13, 15]

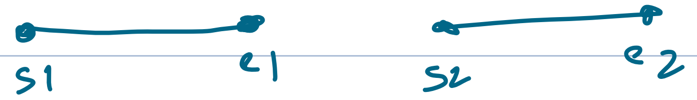


Ans = [[1, 7] [8, 15]]

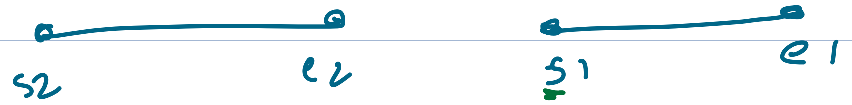
Non - overlapping condition

(S1, e1)   (S2, e2)

Case1:

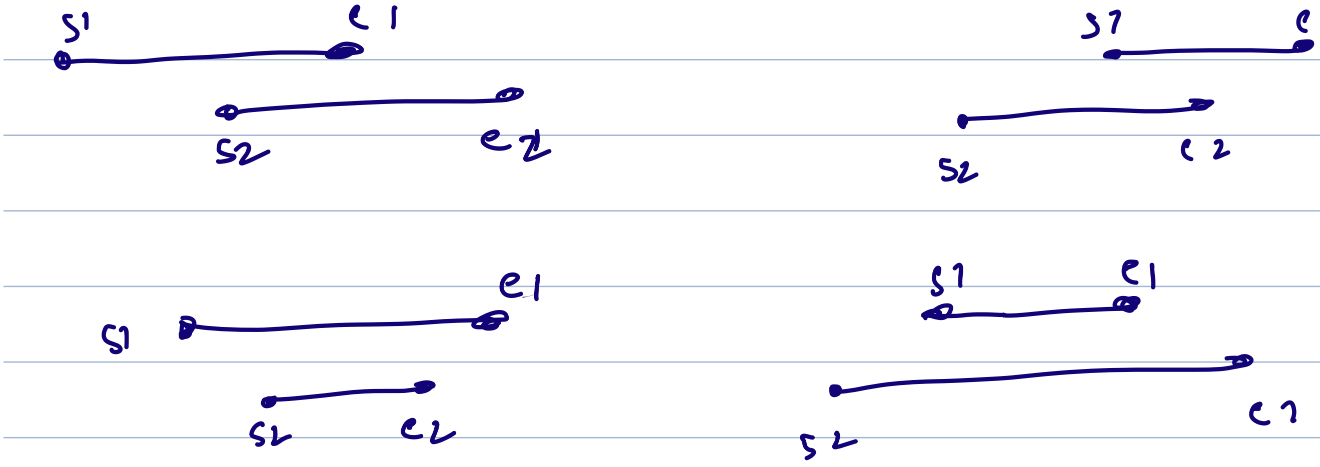


Case2:



if (S2 > e1 || S1 > e2)

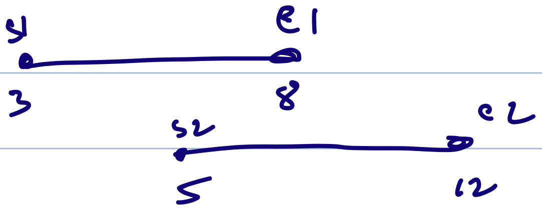
Merged Interval



Start : min (s1, s2)

End : max (e1, e2)

Quiz : [3, 8] [5, 12]



```
if( s2 ≤ e1 ) {        =>   [ 3, 12 ]
    "Overlap"
}
```

Quiz. [6, 10] [8, 15]

[6, 15]

Problem Statement

You are given a collection of intervals A in a 2-D array format, where each interval is represented by a pair of integers `[start, end]`. The intervals are sorted based on their start values.

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

A : [[0,2] [1,4] [5,6] [6,8] [7,10] [8,9], [12, 14], [14,16]]

Ans : [[0,4], [5,10], [12,16]]

CurrS = ~~0~~ ~~5~~ 12

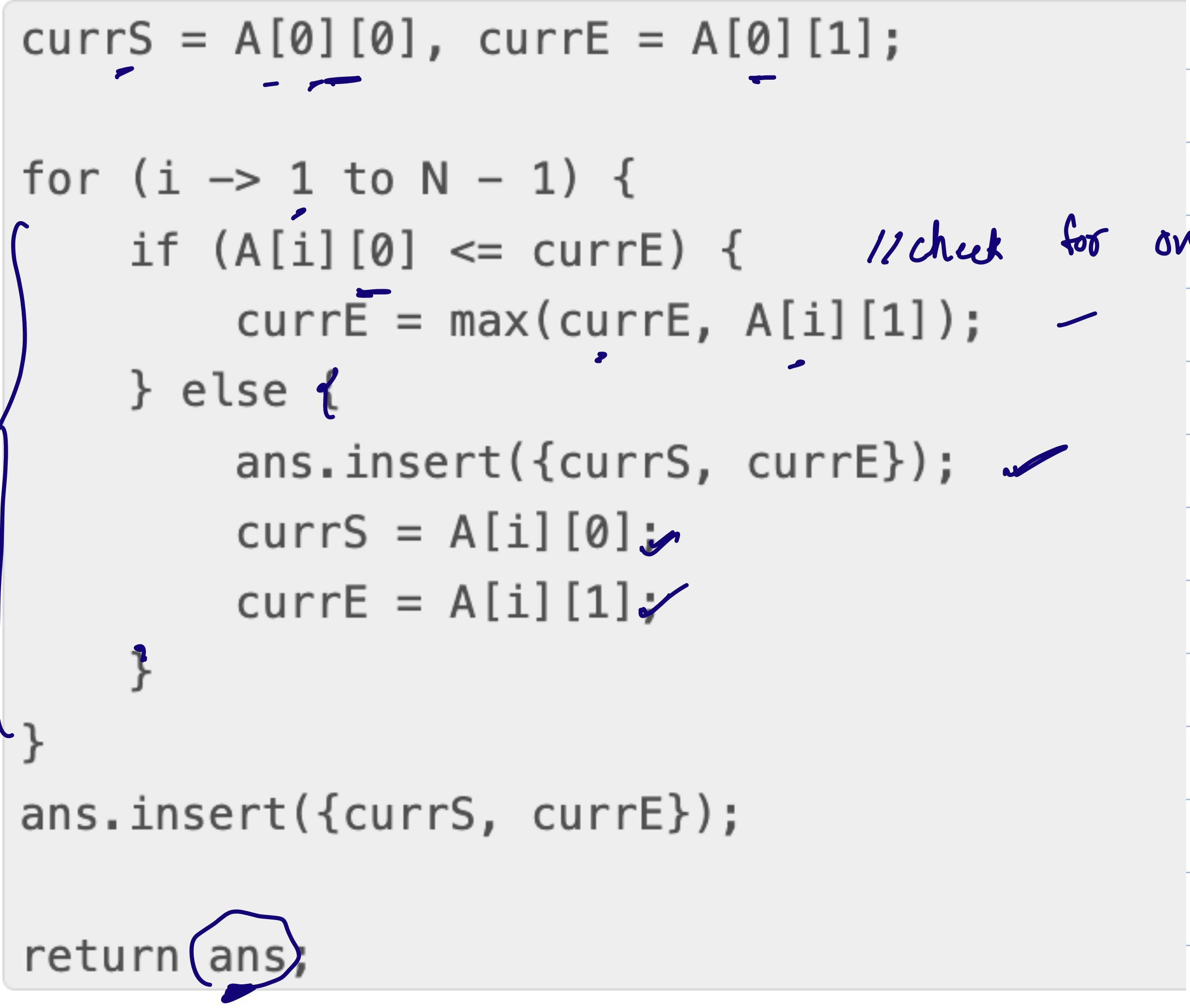
CurrE : ~~2~~ ~~4~~ ~~6~~ ~~8~~ ~~10~~ ~~14~~ 16

Quiz : [1,10] [2,3] [4,5] [9,12]]

Ans = [[1,12]]

CurrS = 1

CurrE = ~~10~~ 12



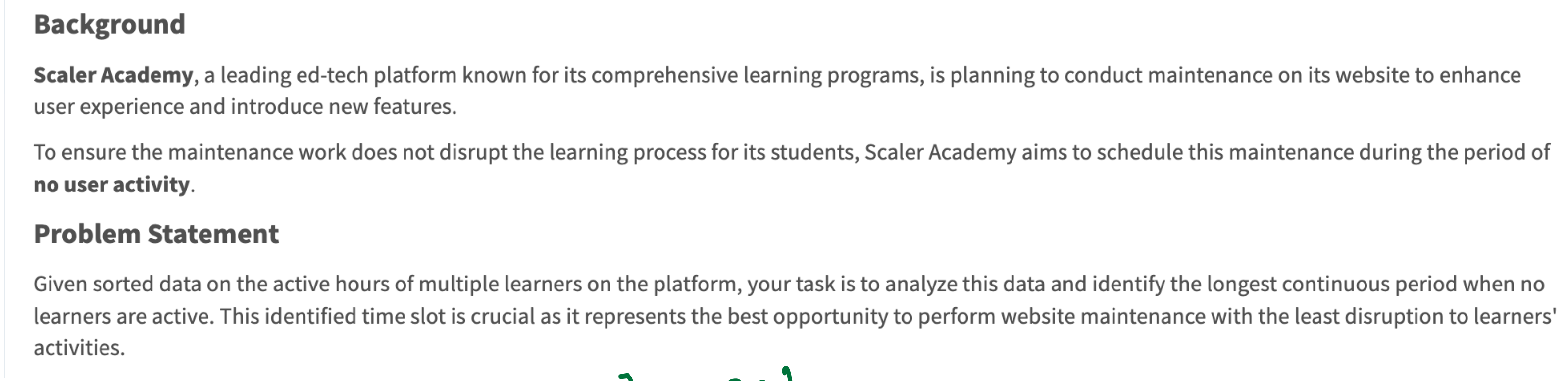
```
currS = A[0][0], currE = A[0][1];

for (i -> 1 to N - 1) {
    if (A[i][0] <= currE) {     // check for overlap
        currE = max(currE, A[i][1]);
    } else {
        ans.insert({currS, currE});
        currS = A[i][0];
        currE = A[i][1];
    }
}
ans.insert({currS, currE});

return ans;
```

T.C: O(N)

S.C: O(1)



Background

Scaler Academy, a leading ed-tech platform known for its comprehensive learning programs, is planning to conduct maintenance on its website to enhance user experience and introduce new features.

To ensure the maintenance work does not disrupt the learning process for its students, Scaler Academy aims to schedule this maintenance during the period of **no user activity**.

Problem Statement

Given sorted data on the active hours of multiple learners on the platform, your task is to analyze this data and identify the longest continuous period when no learners are active. This identified time slot is crucial as it represents the best opportunity to perform website maintenance with the least disruption to learners' activities.

] 0-23)

Input: [[9, 11] , [14, 16] , [15, 20]]

[[9, 11], [14, 20]]

[0, 8]   [12, 13]   [21, 23]

Steps

1) First, merge the overlapping Intervals

2) Find the free slots/ Intervals