Today's Agenda

merge Intervals → ②

find first +ve Integer
↑
missing

Intro

Microsoft → L60

Pepcoding → Cofounder

Scaler → 2 years.

↳ Certification.

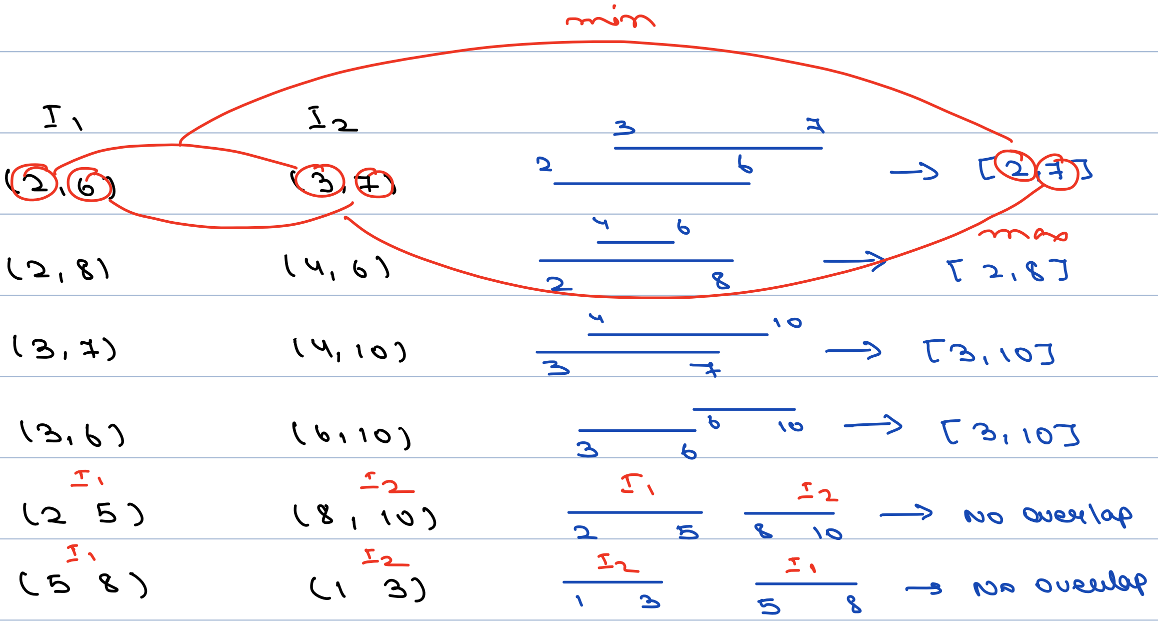
LLD, Databases, HLD

Project

Merge Interval

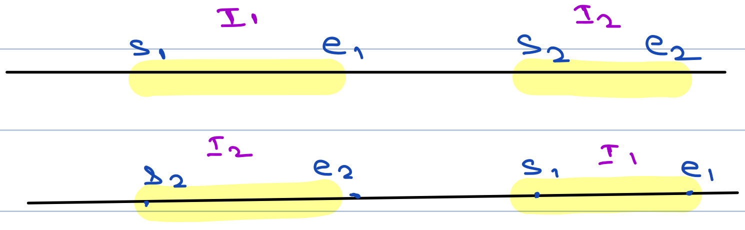
Interval [s e] → [2, 5]

[1 5]



I1	I2	
(2,6)	(3,7)	→ [2,7]
(2,8)	(4,6)	→ [2,8]
(3,7)	(4,10)	→ [3,10]
(3,6)	(6,10)	→ [3,10]
I1 (2 5)	I2 (8 , 10)	→ no overlap
I1 (5 8)	I2 (1 3)	→ No overlap

I1 (s1 e1)   I2 (s2 e2)



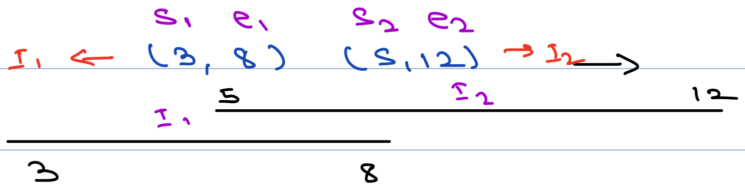
if (s2 > e1)

if (s1 > e2)

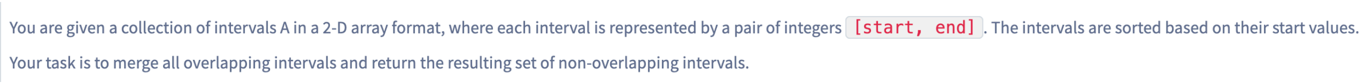
if (s2 > e1 || s1 > e2) → no Overlap

if overlap → merged interval

[min(s1 , s2) , max(e1, e2)]



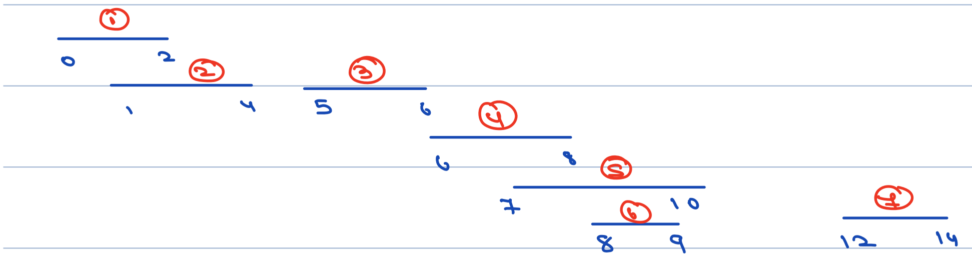
Ques

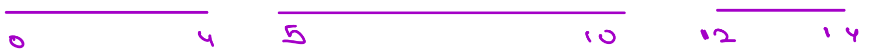


You are given a collection of intervals A in a 2-D array format, where each interval is represented by a pair of integers `[start, end]`. The intervals are sorted based on their start values.

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

Interval[] = {(0,2), (1,4), (5,6), (6,8), (7,10), (8,9), (12,14)}

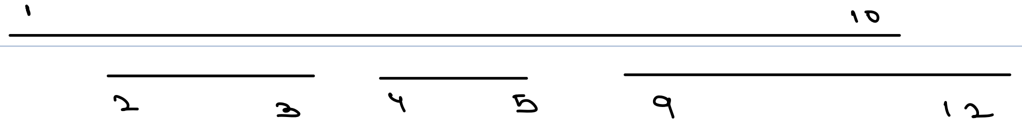


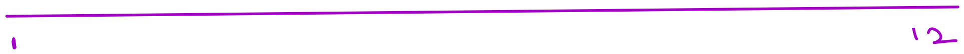


Ans = {(0,4), (5,10), (12,14)}

Quiz

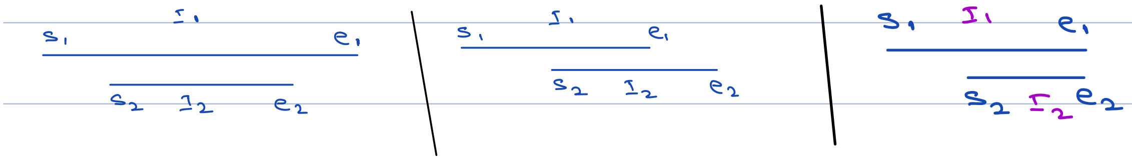
(1,10), (2,3), (4,5), (9,12)



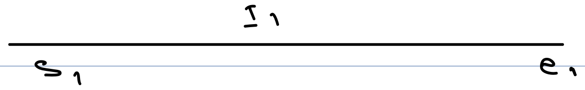


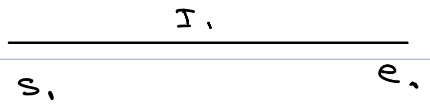
I_1 I_2

(s_1, e_1) (s_2, e_2)



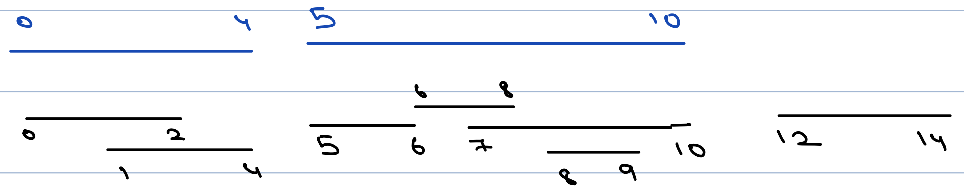
overlap $s_2 <= e_1$





Overlap → $s_2 <= e_1$

arr[0][0]
arr[0][1]



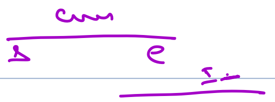
Intervals[] = {(0,2), (1,4), (5,6), (6,8), (7,10), (8,9), (12,14)}

curr	next Available	After Merged	final ans
(0-2)	(1-4)	(0-4)	
(0-4)	(5-6)	—	(0-4)
(5-6)	(6-8)	(5-8)	(0-4)
(5-8)	(7-10)	(5-10)	(0-4)
(5-10)	(8,9)	(5-10)	(0-4)
(5-10)	(12-14)		[(0-4) (5-10)]
(12-14)			[(0-4) (5-10) (12-14)]

int[] arr → int , arr[0].s

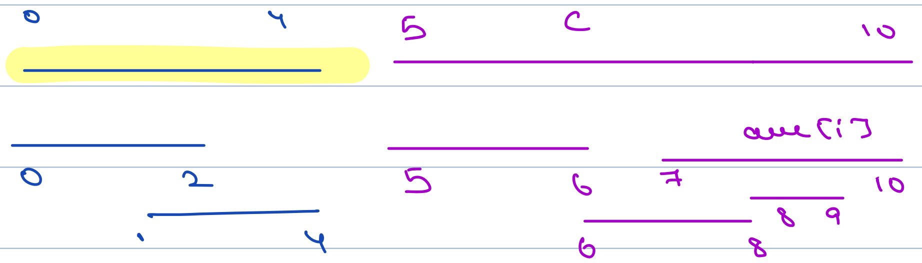
Interval [] arr → interval arr[0].e ,

```
Interval [] arr;          //Given.
List <Interval> ans;

curr_start = arr[0].s ,  curr_end = arr[0].e
 for (i=1; i < n; i++) {
     if (arr[i].s <= curr_end) {        //overlap
              // merge
          curr_end = max (curr_end, arr[i].e);
     }
     else {
          print ( curr_start, curr end); // store it to ans;
          curr_start = arr[i].s;
          curr_end = arr[i].e;
     }
 }

 print ( curr_start, curr end);
```



T.C -> O(n)

S.C -> O(1)



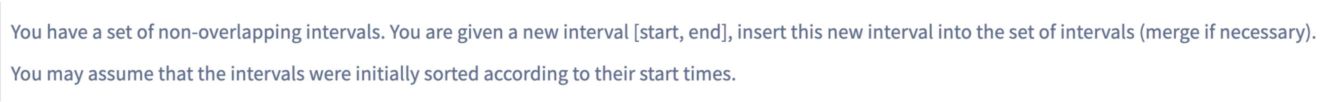
i = ~~1~~ 2

c

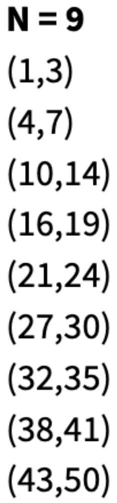
Intervals[] = { (0,2), (1,4), (5,6), (6,8), (7,10), (8,9) }

curr	next Available	After merging	final ans
(0-2)	(1,4)	(0-4)	
(0-4)	(5,6)		(0-4)
(5,6)	(6,8)	(5,8)	
(5,8)	(7,10)	(5,10)	
(5,10)	(8,9)	(5,10)	

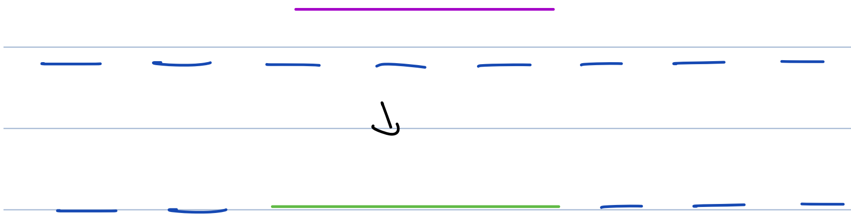
Ques



You have a set of non-overlapping intervals. You are given a new interval [start, end], insert this new interval into the set of intervals (merge if necessary). You may assume that the intervals were initially sorted according to their start times.



N = 9
(1,3)
(4,7)
(10,14)
(16,19)
(21,24)
(27,30)
(32,35)
(38,41)
(43,50)



new interval = [12, 22]

N = 9

(1,3) (12,22) → (1,3)

(4,7) (12,22) → (4,7)

(10,14) (12,22) : (10:22)

(16,19) (10,22) : (10:22)

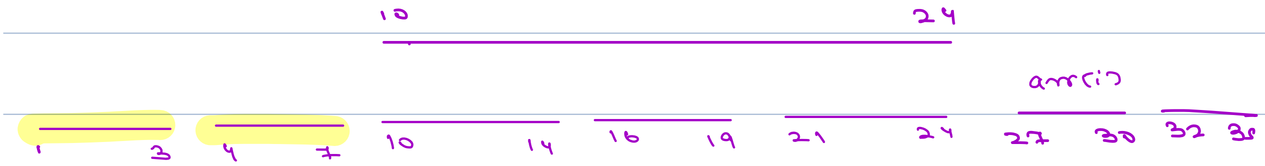
✓ (21,24) (10,22) : (10:24)

✓ (27,30) (10:24) → (10:24)

✓ (32,35)

✓ (38,41)

✓ (43,50)



Ex 2

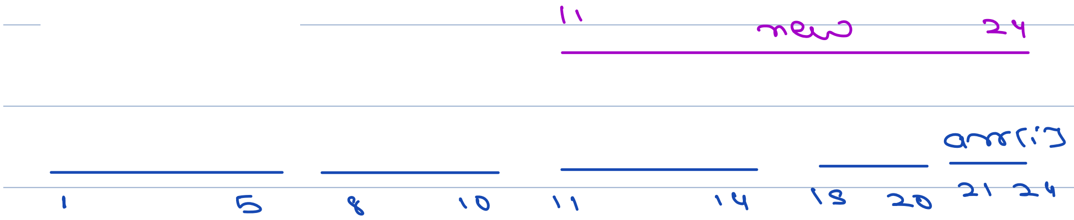
(1,5) (12,24) → (1,5)

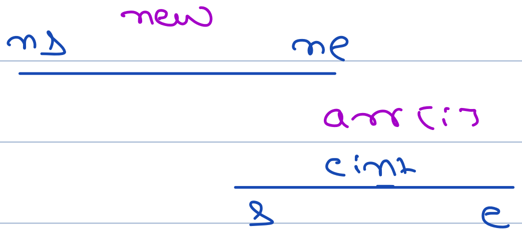
(8,10) (12,24) → (8,10)

(11,14) (12,24) : (11,24)

(15,20) (11,24) : (11,24)

(21,24) (11,24) → (11,24)

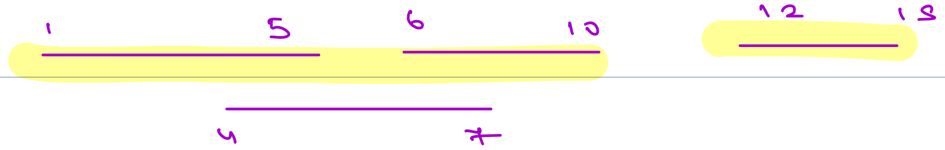




T.C -> O(n)
S.C -> O(1)

(ns, ne) —> new interval.

```
for (i=0; i<n; i++) {
        cInterval = arr[i];
        if (ns > cInterval.e) {
            print (cInterval);
        }
        else if (cInterval.s > ne) {
            print (ns, ne);
            for (j=i; j<n; j++) {
                print (arr[j]);
            }
            return;
        }
        else {
            ns = Min(cInterval.s, ns);
            ne = Max (cInterval.e, nE);
        }
}
print (ns, ne);
```



8:27 am → 8:35 am

Ques

→ +ve Integer

Given an unsorted array of integers, Find first missing Natural Number.

arr[5] = {3, -2, 1, 2, 7} → 4

arr[7] = {-9, 2, 6, 4, -8, 1, 3} → 5

{-2, 4, -1, -6, 3, 7, 8, 4, -3} → 1

{1, 0, -5, -6, 4, 2} → 3

{1, 2, 5, 6, 4, 3} → 7

{-4, 8, 3, -1, 0} → 1

→ -4, -1, 0, 3, 8

{4, 2, 1, 3} → 5

[7, 8, 3, 4, 6] → 1

Observation → n → 1 to n+1

n → 5 → 1 → [1, 2, 3, 4, 5] ↳ 6

n → 5 → 6

s.len → (1 . . . 5) [1, 6, 3, 2, 4] ↳ 5

Soln 1:- check all the numbers from 1 to n+1

```
for (i=1; i<=n; i++) {
    → check i in array —
        ↳ i is missing → return n.
}
return n+1
```

T.C → $O(n^2)$

S.C → $O(1)$

Soln 2:-

Add all elements to the set and check if element(1 to N) is present or not.

T.C → $O(n)$

S.C → $O(n)$

Soln 3:- Sort & check

T.C → $O(n\log n)$

Soln 4:-   T.C → O(n),   S.C → O(1) .

N = 5

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

n → 0 → 1
n → n-1 → n

valid .
1 to 8

5 is missing

arr[8] =

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

i	arr[i]		
0	4	s(0,3)	s(0,5)
1	2	set	
2	7	s(2,6)	s(2,7)
3	4		
4	9	ignore	
5	set		
6	set		
7	set		

Ans ⇒ 5

n → 10 → 0 → 1
10 → 9 → 10

n → 0 → 1
→ n-1 → n

10 → 0 → 1
10 → 9 → 10

1 to 10

	0	1	2	3	4	5	6	7	8	9
arr[10] =	~~5~~ ~~9~~ 1	~~-14~~ 2	~~6~~ ~~-10~~ 3	~~7~~ ~~2~~ -14	~~9~~ 5	~~-10~~ 6	~~2~~ 7	~~3~~ ~~-10~~ 8	~~1~~ 9	~~8~~ -10

i	arr[i] = val	
0	5	s(0,4)
1	ignore	
2		
3		

Ans → 4

s(i, val-1);

```
for (i=0; i<n; i++) {
                                            → valid
    while (arr[i] > 0 && arr[i] <= n &&
                          arr[i] != i+1) {
                                   → not set
        val = arr[i];
        if (arr[i] == arr[val-1]) { break }
            s(i, val-1);
    }
}
```

Iterate & find mising no.

```
for (i=0; i<n; i++) {
        if (arr[i] != i+1)
            return i+1;
        }
}
return n+1;
```

Duplicate issue,

0	1	2	3	4	5
~~4~~,	~~1~~,	~~3~~,	~~3~~,	~~2~~,	3
~~3~~	~~3~~	3	4	3	
~~3~~	2				
1					

Ans → 5.

T.C → O(n)

S.C → O(1).

i	arr[i]
0	4

n=10

n2

1∞

0 - 1, 2, 3, . . . 9

```
for (i=0; i<n; i++) {
    while (arr[i] > 0 && arr[i] <= n &&
                        arr[i] != i+1) {
                                  ↳ not set
        val = arr[i];
        if (arr[i] == arr[val-1]) { break }
            s(i, val-1);
    }
}
```

```
for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
    }
}
```

n^2

$n=10$

i	j
0	n
1	n
2	n
⋮	
	n

$n + n^2$

n ε

$n+\varepsilon$

i	while
0	-
1	-
2	-
3	..

n + n times

$n + n$