SCALER

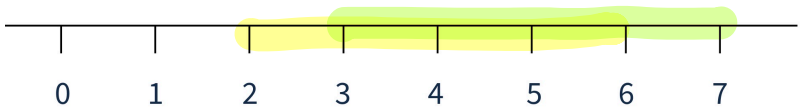
Interview Problems

TABLE OF CONTENTS

Merge Overlapping Intervals

merged-interval

I_1 I_2

(2,6) (3,7) → 

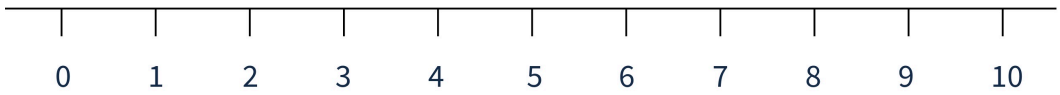
[2,7]

(2,8) (4,6) → 

[2,8]

(3,7) (4,10) → 

[3,10]

(3,6) (6,10) → 

[3,10]

(2,5) (8,10) → ✗

(5,8) (1,3) → ✗

(6,10) (3,5)
s_1 e_1 s_2 e_2

[3,8] [5,12] → 3-12

(6,10) (3,7)
s_1 e_1 s_2 e_2

[6,10] [8,15] → [6,15]

< / > Code

s_1, e_1, s_2, e_2 (given)

I_1 I_2

6, 10 3, 7

s1 e1 s2 e2

```
if ( s2 > e1 || s1 > e2 ) {
    intervals are not overlapping
}
else {  // intervals will overlap
    merged interval -> min(s1, s2) , max(e1, e2)
}
```

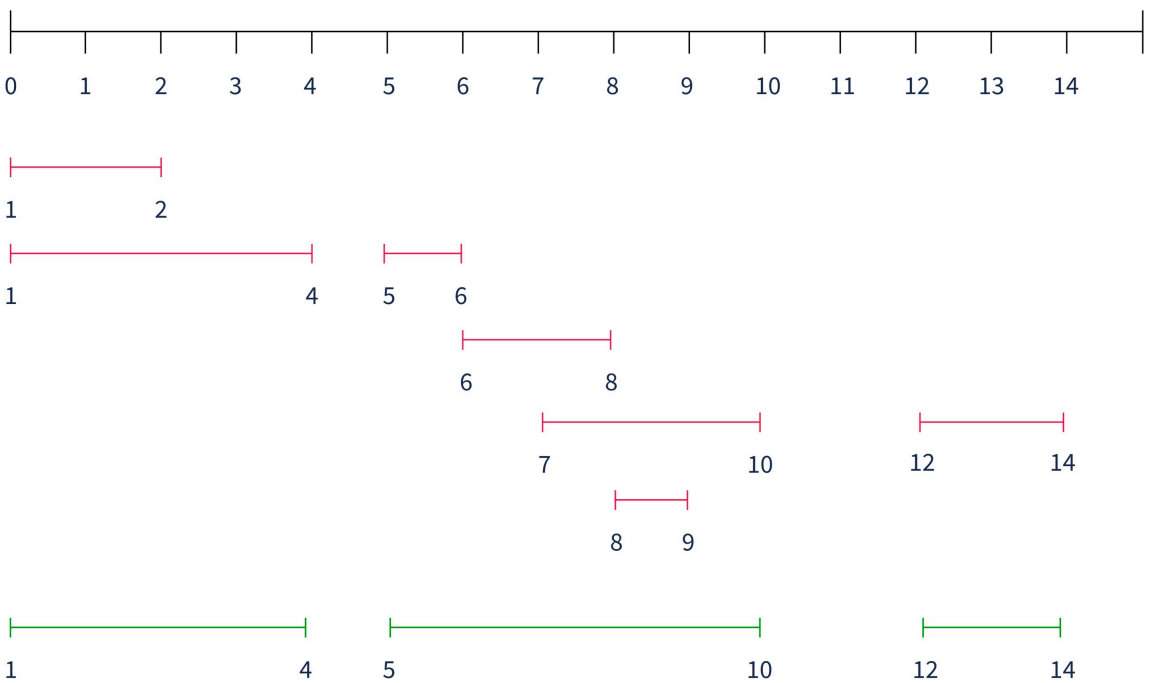
< *Question* > : Given a sorted list of overlapping intervals, sorted based on start-time, Merge all overlapping intervals and return the sorted list of non-overlapping intervals.

$1 \leq N \leq 10^5$

N = 7

Intervals

1	2
1	4
5	6
6	8
7	10
8	9
12	14



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

[1,10]

[1,10]

[1,12]

Example :

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

l = 12, r = 14

ans = [[0, 4] , [5, 10] , [12, 14]]

[[1,1] , [2,2] , [3,3]]

l = 3, r = 3

ans = [[1,1] , [2,2] , [3,3]]

< / > Code

```
l → arr[0][0],   r → arr[0][1] ,  list<pair> ans;

for( i → 1 to N) {

    if( arr[i][0] ≤ r) {   // overlapping
    {
        r = Max(r, arr[i][1]);
    }
    else{
    {
        ans.insert( l, r);
        l = arr[i][0];
        r = arr[i][1];
    }
}
ans.insert(l,r);
return ans;
```

T.C → O(N)
S.C → O(1)

Practical Scenrio (maintenance of site when no users are active)

Data → [9,11), [14,16], [15,20)

⇅

[9,11) [14,20)

↓ 9 hours

↓ 3 hours

↓ 4 hours

< *Question* > : Given N non-overlapping intervals sorted based on start-time. Given a new interval. Merge this with existing intervals, if possible and return final non-overlapping intervals.

[[1,3], [4,7], [10,14], [11,19], [21,24], [27,30], [32,35], [40,50]]

[10,24]

[

ans [1,3], [4,7], [10,24], [27,30], [32,35], [40,50]

Resultant Intervals →

N = 5 I = (12,22)

Intervals( ) → [(1,5) , (8,10) , (11,14) , (15,20) , (21,24)]

(11,24)

Resultant Intervals → [[1,5], [8,10], [11,24]]

↑

ans.

Quiz →

arr → [[1,5], [6,10], [12,15]]

I → [4,7)

[1,10]

---

resultant intervals ⇒ [[1,10], [12,15]]

Code →

```
l = I.start ,   r = I.end

list<pair> ans;

for( i → 0 to N) {
    if( arr[i][1] < l) {   //no-overlap
        ans.insert( arr[i][0], arr[i][1]);
    }
    else if( arr[i][0] > r) {   //no overlap
        ans.insert ( l, r);
        for( j → i to N) {
            ans.insert( arr[j][0], arr[j][1]);
        }
        return ans;
    }
    else{                    //overlap
        l = min( l, arr[i][0]);
        r = max( r, arr[i][1]);
    }
}

ans.insert(l,r);
return ans;
```

T.C → O(N)
S.C → O(1)

< Question > : Find the first missing natural number. N = length of array

1, 2, 3, 4, 5, - ——— ∞

$-10^9 \le arr[i] \le 10^9$

$1 \le N \le 10^6$

Example 1: arr [5] : [3 -1 1 2 7] ans = 4

Example 2: arr [7] : [9 2 6 4 -8 1 3] ans = 5

Example 3 : arr [6] : [1 0 -5 -6 4 2] ans = 3

Example 4 : arr [6] : [1 2 5 6 4 3] ans = 7

Example 5 : arr [4] : [1 2 3 4] ans = 5

Quiz → [5, 3, 1, -1, -2, -4, 7, 2] ans = 4
(indices: 0 1 2 3 4 5 6 7)

BF Approach →

For every natural no's, search in array by iterating on all the elements.

[T.C → $O(N^2)$
S.C → $O(1)$]

Idea -2 Sorting

arr[] → [9 2 6 4 1 -8 3]

↓ Apply sorting

[-8 1 2 3 4 6 9]

T.C → O(N log N)

S.C → depends on inbuilt sorting algo.

Expected T.C → O(N), S.C → O(1)

arr[N] → [_ _ _ _ _ _ _]

if all elements from 1 to N are present → N+1

otherwise → ans → any number from 1 to N

range of ans → [1, N+1]

How do you mark the presence?

arr[] → [-8, -1, -4, -2, 6, -3]
idx: 0 1 2 3 4 5

ans = 5

1	present → 0
2	→ 1
3	→ 2
4	→ 3
5	→ 4

val → val - 1

What if -ve elements are also there in array?

arr[] → [-8, -1, -4, -2, 6, 5, 1, 3]
idx: 0 1 2 3 4 5 6 7

-ve elements → replace with N+2 and 0

[-10, 10, -10, 10, -6, -5, 1, 3]
idx: 0 1 2 3 4 5 6 7

ans = 2

</> Code

```
for( i → 0 to N) {
    if ( arr[i] ≤ 0){
        arr(i) = N+2;
    }
}
```

```
for( i → 0 to N) {
    val = abs ( arr[i]);
    if( val ≥ 1 && val ≤ N){
        arr[val-1] = -1 x abs (arr[val-1]);
    }
}
```

```
for( i → 0 to N-1) {
    if( arr[i] > 0){
        return i+1;
    }
}
return N+1;
```

T.C → O(N)
S.C → O(1)

[-1, -2, -6, -5, -4, -3]
 0 1 2 3 4 5

arr → [6 2 4 2 -5 3 4]
0 1 2 3 4 5 6

(1) → [6 -2 -4 -2 9 -3 4]
0 1 2 3 4 5 6

ans = 1

O(N²)
↓
O(NlogN)
↓
O(N) with hashset/hashmap
↓
O(N) & O(1)
T.C S.C.

modify the array