# Arrays - Interview Problems

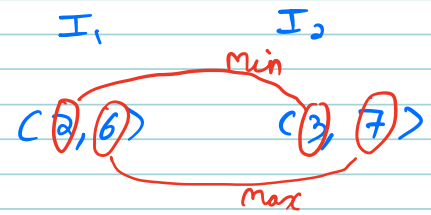
## # MERGE INTERVALS

↳ An Interval is defined by Start & End time where start <= End.

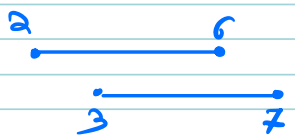
Ex:- [5 6] → [2 3]


↳ [1 5] →

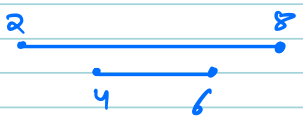

↳ Let's say we have been given a list of intervals. we have to merge if they overlap.




↳ [2, 7]

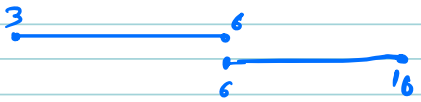



=> (2,8) (4,6) => [2,8]




=> (3,7) (4,10)
↳ [3, 10]


=> (3,6) (6,10) => [3,10]



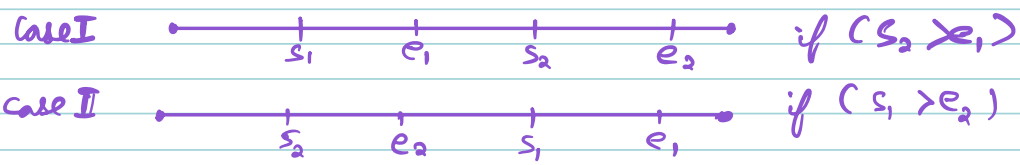

=> (2,5) (8,10) => X


=> (5,8) (1,3) => X

## Observation

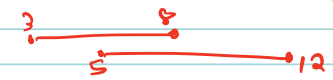
① what will be the conditions for not overlap?

$I_1 (s_1, e_1)$ $I_2 (s_2, e_2)$




② what well be the condition if intervals overlap.

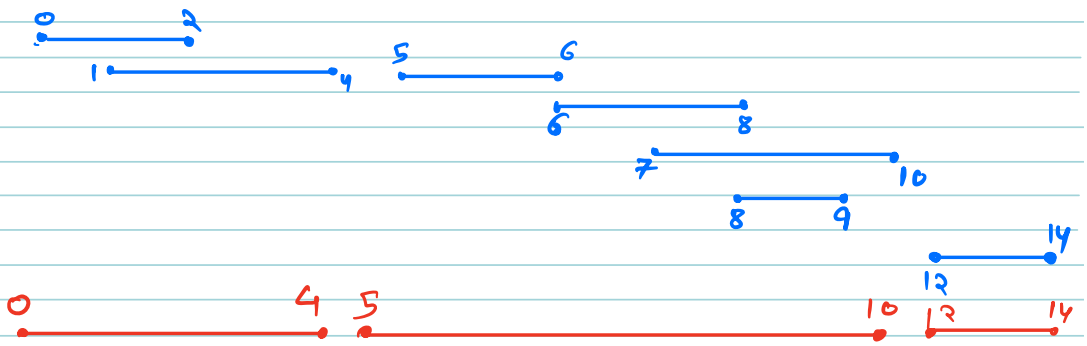
↳ If overlap ⇒ the ans will be $[\min(s_1, s_2), \max(e_1, e_2)]$

Quiz 1 :- $[3,8]$ $[5,12)$  ⇒ Yes

Quiz 2 :- $[6,10]$ $[8,15]$  ⇒ $[6,15]$

**Question** :- Given a sorted list of overlapping interval, sorted based on start time, merge all overlapping intervals & return sorted list.

Ex:- $\{(0,2)(1,4)(5,6)(6,8)(7,10)(8,9)(12,14)\}$

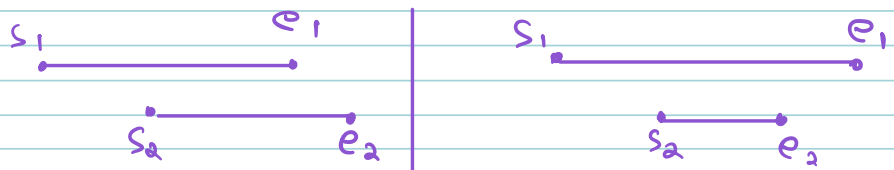


Ans => {(0,4) (5,10) (12,14)}

## Observation

① If two intervals overlap, then condition will be.

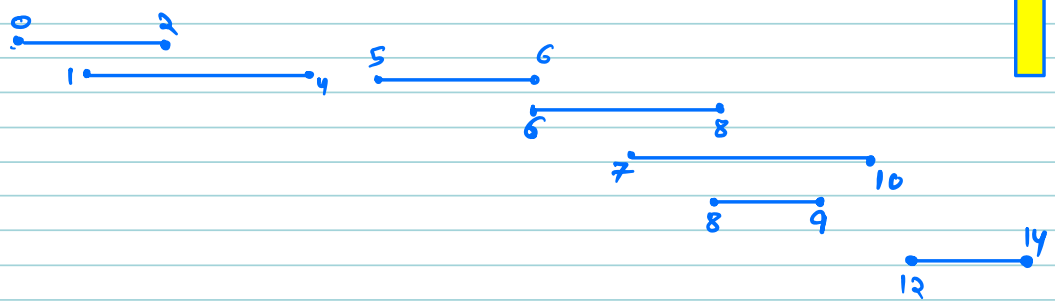
$I_1$ $(s_1, e_1)$ $I_2$ $(s_2, e_2)$ $[s_1 <= s_2]$




condition => if ( $s_2 <= e_1$ )

## DRY RUN

Ex:- {(0,2) (1,4) (5,6) (6,8) (7,10) (8,9) (12,14)}




| Curr | Next Interval | After Merging | Answer List |
|---|---|---|---|
| (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) | ——— |
| (5, 10) | (12, 14) | ——— | (0, 4) (5, 10) |
| (12, 14) | ——— | ——— | (0, 4) (5, 10) (12, 14) |

* we can say there is No overlap b/w current & next Available then curr will move to Ans.

* when we are able to merge then nothing goes to Final Ans because we are not sure whether anything from future will merge or not.

## PSEUDO CODE

if, int[] arr → int.

over here, Interval[] arr → arr[0].s, arr[0].e

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

curr_s = arr[0].s ;
curr_e = arr[0].e ;

for (i=1 ; i < arr.length ; i++){
    if ( arr[i].s <= curr.e ){
        // overlapping
        curr_e = Max( curr_e, arr[i].e);
    } else {
        // Non-overlapping.
        Interal temp( curr_s, curr e);
        ans.add( temp );
        curr_s = arr[i].s ;
        curr_e = arr[i].e ;
    }
```

```
}
Interval temp(curr_S, curr_E);
ans.add(temp);
return ans;
```

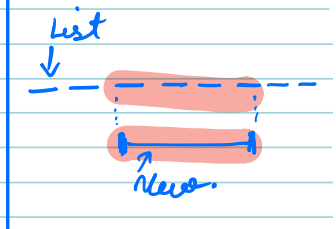
TC — O(N)

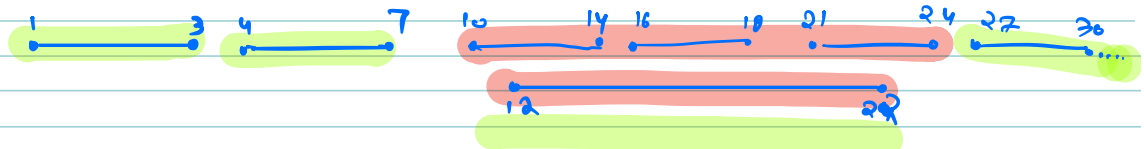
SC — O(1) ← Given we can ignore the ans space.

**Question :-** Given a sorted list of non overlapping Intervals based on start time, insert a new interval such that the final list of intervals is also sorted & non-overlapping. Print the Intervals.

Ex:- N=9

| | Given New Interval | Ans |
|---|---|---|
| (1, 3) | (12, 22) | (1, 3) 1 |
| (4, 7) | (12, 22) | (4, 7) 2 |
| (10, 14) | (12, 22) : (10, 22) | — |
| (16, 19) | (10, 22) : (10, 22) | — |
| (21, 24) | (10, 22) : (10, 24) | — |
| 4 (27, 30) | (10, 24) | (10, 24) 3 |
| 5 (32, 35) | | |
| 6 (38, 41) | | |
| 7 (43, 50) | | |



## Observation

① We need to move interval in Ans if its ending before given interval.

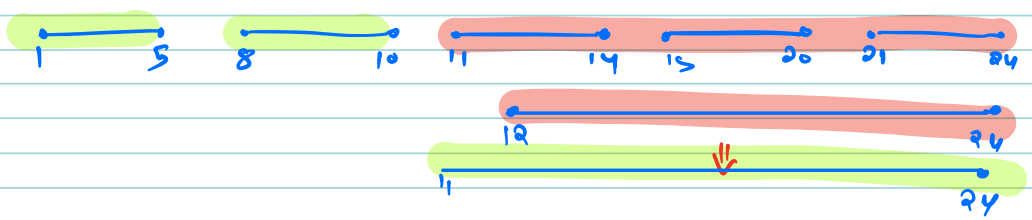
② If the removed Interval overlap with given interval then we will move ahead without pushing anything into Ans.

③ Continuous Step ② till you get interval which is larger than given interval Now you can push given & all other into Ans.

## Ex 2

N = 5

| | Given New Interval | Ans |
|---|---|---|
| (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) | —— |
| —— | (11, 24) | (11, 24) |




## PSEUDO CODE

(ns, ne) ⟶ new Interval

```
for( i = 0 ; i < arr. length ; i++){
    cInterval = arr[i];
```

```
// Non overlapping
if ( ns > cInterval.e) {
        Print (cInterval)
}
else if ( ne < cInterval.s) {
    print (ns, ne);
    for (j=i; j<arr.length; j++){
        Print (arr[j].s, arr[j].e);
    }
    return;
} else {
    ns = Min ( cInterval.s, ns);
    ne = Max ( cInterval.e, ne);
}
Print (ns, ne);
```

when new lies before current

when overlapping

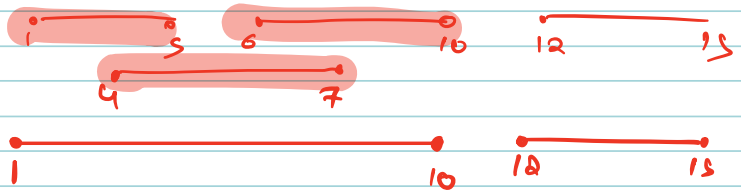
Ex:- N=9

| | Given New Interval | Ans |
|---|---|---|
| (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) |
| (30, 35) | | |
| (38, 41) | | |
| (43, 50) | | |

TC - O(N)

SC - O(1)

Quiz 4 :- (1, 5) (6, 10) (12, 15)




Question :- Given an unsorted array of integers, Find first missing Natural Number. → {1, 2, 3, ....}

Ex 1 arr[5] = {3, -2, 1, 2, 7}
↳ Ans = 4

Ex 2 arr[7] = {-9, 2, 6, 4, -8, 1, 3} ⇒ Ans = 5

Ex 3 {-2, 4, -1, -6, 3, 7, 8, 4, -3} ⇒ Ans = 1

Ex 4 {1, 0, -5, -6, 4, 2} ⇒ Ans = 3

*** Ex 5 {1, 2, 5, 6, 4, 3} ⇒ Ans = 7

Ex 6 {-4, 8, 3, -1, 0} ⇒ Ans = 1

Ex 7 {4, 2, 1, 3} ⇒ Ans = 5

Quiz 5 :- [5, 3, 1, -1, -2, -4, 7, 2] Ans ⇒ 4

Observations
↳ ① If length is N, then ans will be definately in range of [1, N+1]

## Brute force

**Solution 1** :- Since ans is in range [1 to N+1] so, I can search missing no. by traversing repeatedly over array.

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

**Solution 2** :- Sort the array → traverse

TC - $O(n \log n)$ SC - $O(1)$

**Solution 3** :- Use Set, insert all element in Set & then search from 1 to N+1.

TC - $O(N)$ SC - $O(N)$

## ↳ OPTIMIZATION

Hint: We need to use concept that ans will be in range of [1, N+1]

Ex :- {4, 2, 8, 9, 1, 3}

indexes: 0 1 2 3 4 5

0

index → A[i]-1

↳

| index | element | Presence marked at index | Actual Array |
|---|---|---|---|
| 0 | 4 | 3 | {4, 2, 8, -9, 1, 3} |
| 1 | 2 | 1 | {4, -2, 8, -9, 1, 3} |
| 2 | 8 | 7 | X |
| 3 | 9 | 8 | X |
| 4 | 1 | 0 | {-4, -2, 8, -9, 1, 3} |
| 5 | 3 | 2 | {-4, -2, -8, -9, 1, 3} |

↳ indexes: 0 1 2 3 4 5

{-4, -2, -8, -9, 1, 3} → Ans = 5

1 to 7

=> With negatives

Ex 2 :- { 1, 0, -5, -3, 4, 2 }  (8 8 8)

| index | element | Presence marked at index | Actual Array |
|---|---|---|---|
| 0 | 1 | 0 | {-1, 8, 8, 8, 4, 2} |
| 1 | 0 8 | -1 | X |
| 2 | -5 8 | — | X |
| 3 | -3 8 | — | X |
| 4 | 4 | 3 | {-1, 8, 8, 8, 4, 2} |
| 5 | 2 | 1 | {-1, -8, 8, 8, 4, 2} |

=> { -1, 0, -5, -3, 4, 2 }  (indices 0 1 2 3 4 5)

IDEA :-

```
If ( A[i] <= 0) {
    A[i] = N+2;  // Can't be Ans.
}
```

Eg :- { -1, -8, 8, -8, 4, 2 }  (indices 0 1 2 3 4 5)

↳ Ans = 3

## PSEUDO CODE

```
for ( i:=0;  i< N;  i++){
    if(A[i] <=0){
        A[i] = N+2;
    }
}
```

```
for (i = 0; i < N; i++){
    ele = abs(A[i]);
    if (ele >= 1 && ele <= N){
        A[ele-1] = -1 * abs(A[ele-1]);
    }
}
for (i = 0; i < N; i++){
    if (A[i] > 0) return i+1;
}
return N+1;
```

TC - O(N)
SC - O(1)

Ex:- {4, 2, 8, 1, 5, 3}
0 1 2 3 4 5