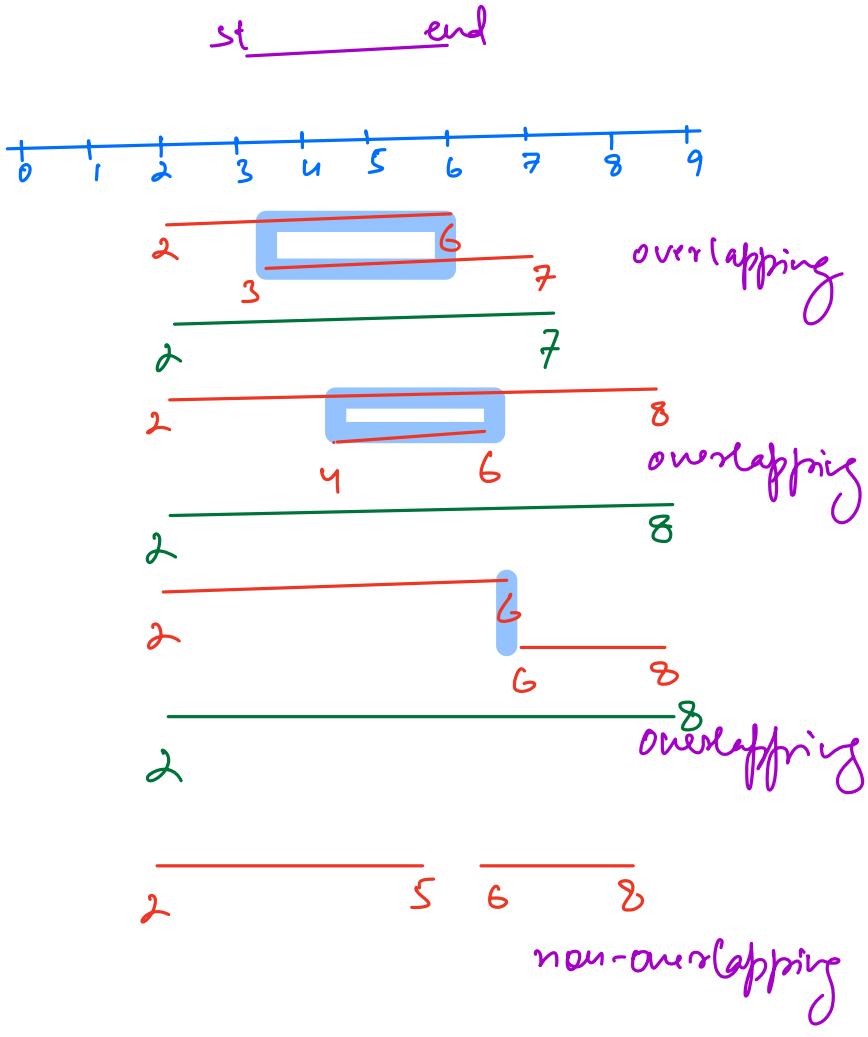
Array 3 - Interview Problems

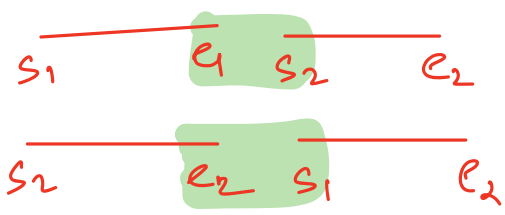
Merge Intervals



Non-overlapping condition

$s_1 \text{ —— } e_1$

$s_2 \text{ —— } e_2$



$$e_1 < s_2 \;||\; e_2 < s_1$$

Merge 2 overlapping intervals

$s_1 — e_1$
$s_2 — e_2$
} → $st = \min(s_1, s_2)$
$end = \max(e_1, e_2)$

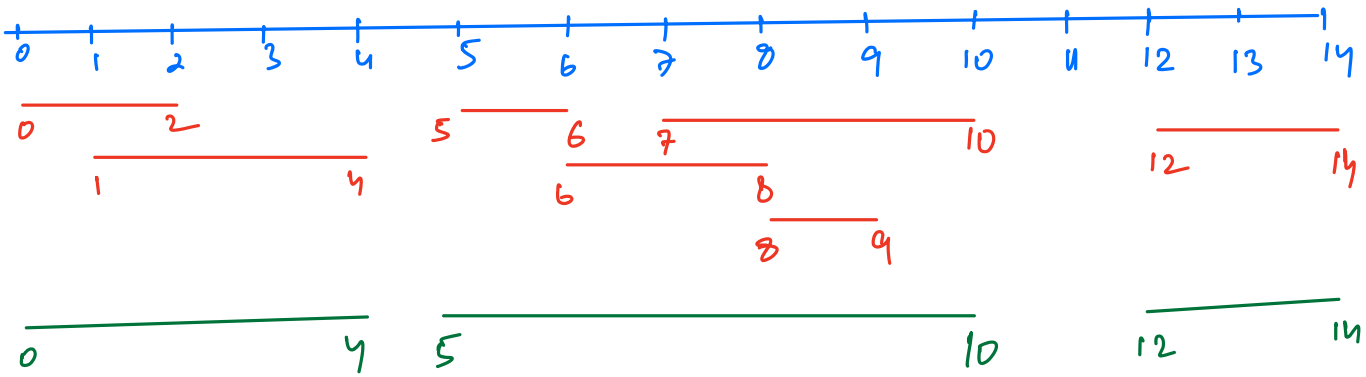
Question

Given a list of intervals, sorted w.r.t start time.

Merge all overlapping intervals & return the sorted list.

st = [0 1] [5 6 7 8] [12]
end = [2 4] [6 8 10 9] [14]

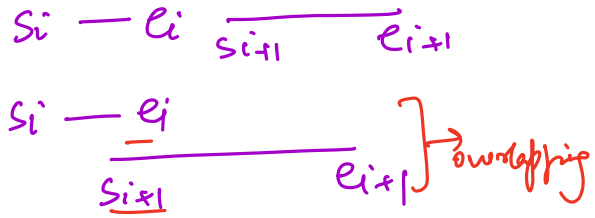
S = ∅ 5 12
E = 2 4 6 8 10 14



output: 0 4 ✓
5 10 ✓
12 14 ✓

sorted w.r.t start time !?

$s_i <= s_{i+1}$



overlap → $s_{i+1} <= e_i$

Code

```
S = st[0]   , E = end[0]

for(i = 1 to n-1 ) {
    if (st[i] <= E ) {  // overlapping
        E = max( E, end[i])
    }
    else {   // not overlapping
        print (S, E)
        S = st[i] , E = end[i]
    }
}
print(S, E)
```

TC : O(N)

SC : O(1)

eg

✓	✓	✓
1	3	5
2	4	6

~~3~~ 5 4 6

S = ~~1~~, E = ~~2~~

(1,2)

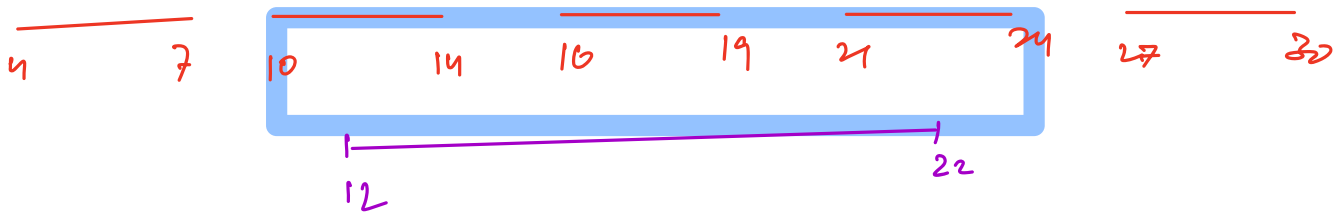
(3,4) (5,6)

Question 2

Given a list of non-overlapping intervals, sorted w.r.t start time. Insert a new interval in the sorted list s.t. final list is non-overlapping.

st = [4 10 16 24 27]
end = [7 14 19 24 30]

L = 12
R = 22



output :

4 7

10 24

27 30

st : [4 10 16 24 27]

end : [7 14 19 24 30]

L = 25

R = 26

L — R

1. st(i) — end(i) L — R ⇒ print (st(i), end(i))

2. L — R st(i) — end(i) ⇒ print (L, R)
print all future intervals

3. overlapping ⇒ merge

Code

```
// L, R
for (i=0 to n-1) {
    if (end(i) < L) {
        print (st(i), end(i))
    }
    else if ( R < st(i)) {
        print (L, R)
        for(j= i to n-1) { print (st(j), end(j)) }
        return;
    }
}
```

TC : O(N)

SC : O(1)

~~nested loop ⇒ O(N^2)~~

```
    else {
        L = min(L, st[i])
        R = max(R, end[i])
    }
}
print(L, R)
```

st = [4 10 16 24 27]

end = [7 14 19 24 30]

L = 25

R = 26

output:
4 7
10 14
16 19
21 24
25 26
27 30

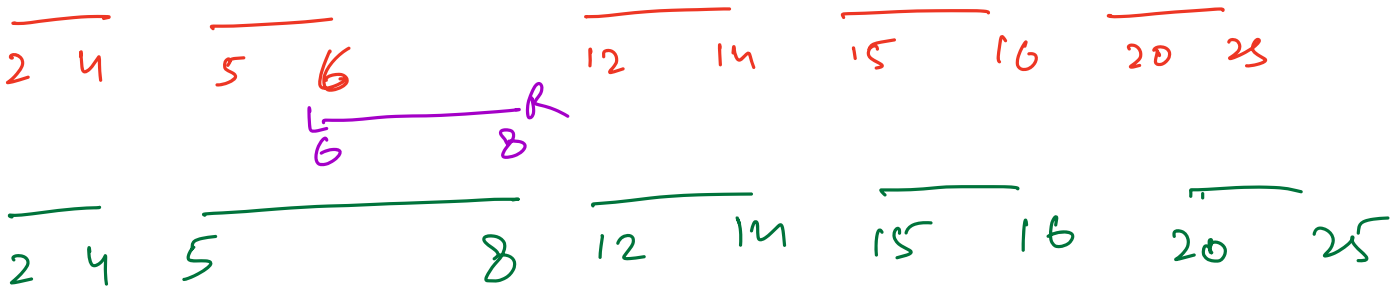
st = [4 10 16 24 27]

end = [7 14 19 24 30]

L = ~~12~~ 10

R = ~~22~~ 24

output:
4 7
10 24
27 30



Question 3

Given an unsorted array of integers, find first missing natural no. $>=1$

A = [3 -2 1 7 2] ans = 4

A = [-8 7 2 5 3] ans = 1

A = [4 1 3 2] ans = 5

minAns = 1 $\Rightarrow$ $1 <= ans <= N+1$

maxAns = N+1

Bruteforce → check no. from 1 to N+1 if it is present in array

TC: $O(N^2)$

SC: $O(1)$

Use hashset → insert all array in hashset & then check from 1 to N+1

TC: O(N)

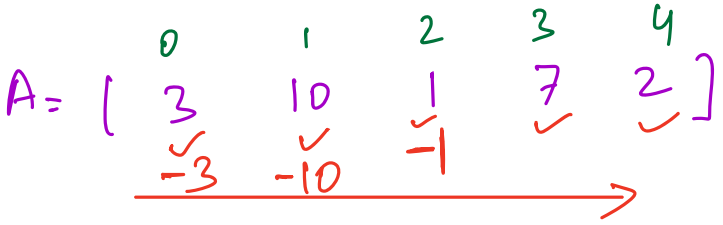
SC: O(N)

Sorting → TC: (NlogN) ✗

check if an element (X) is present in array?

1 <= X <= N+1

let (i, a[i]) = -ive



am = 4

am index

1 → 0

2 → 1

.

4 → 3

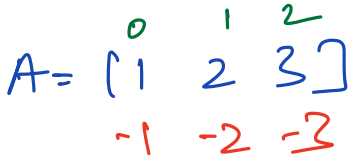
.

N → N-1

N+1 → not needed

if a[i] : -ive

⇒ (i+1) is present in array



A = [3, 10, 1] (indices 0, 1, 2)

-3, -1

ans = 2

A = [11, 12, 13, 17] (indices 0, 1, 2, 3)

ans = 1

Code

```
for(i = 0 to n-1) {
    if(a[i] <= 0)  a[i] = N+2
}
for (i = 0 to n-1){
    x = abs(a[i])
    if( 1<=x && x<=n) {
        idx = x-1
        if(a[idx] > 0)
            a[idx] *= -1
    }
}
for( i = 0 to n-1 ) {
    if(a[i] > 0)
        return i+1
}
return n+1
```

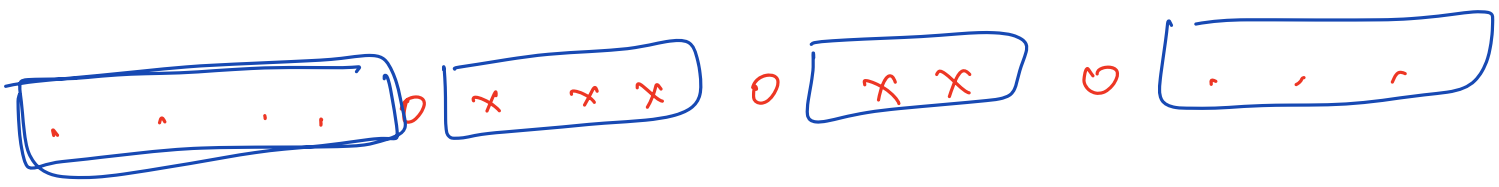
TC: O(N)

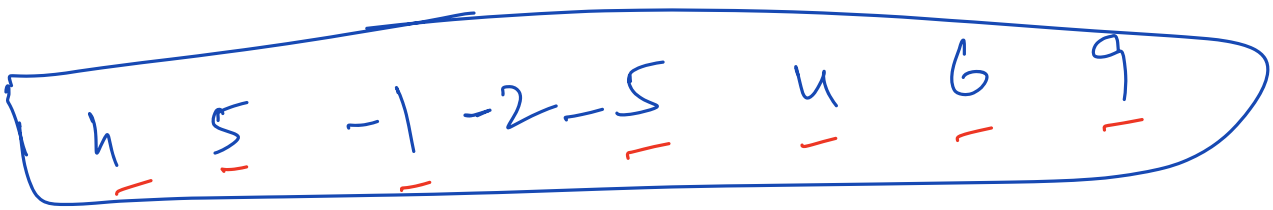
SC: O(1)

if a[i] <= 0 => a[i] = N+2

a = [1 ~~-100~~ 6 3 4] N=4

Doubt





9 7 -2 5 6 -2 7 -5 10^5