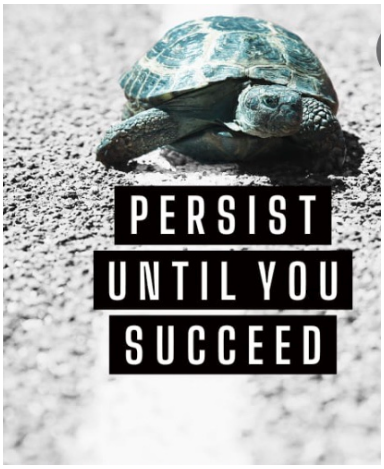


Today's content

1. Given N elements. find first missing +ve no.
2. Merge Overlapping intervals
3. Merge intervals.

Q) Find the first missing natural number. N → length of array

1, 2, 3, 4, ---

Eg1: arr[5]: [3, -1, 1, 2, 7] : 4

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

arr[6]: [1 2 5 6 4 3] : 7

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

idea-1. Check for all the numbers from 1 to N, whether they are present in array or not.

[T.C → O(N²) S.C → O(1)]

idea-2. using Hashset.

// insert all elements in hashset

```
for( i = 1; i <= N; i++){
    if ( i is not present in hashset){
        return i;
    }
}
return N+1
```

[T.C → O(N)
S.C → O(N)]

Condition → No extra space is allowed.

idea-3. Sorting

arr[] → [9 2 6 4 -8 1 3]

↓ sorting

[-8 1 2 3 4 6 9]
0 1 2 3 4 5 6

≠5

[T.C → O(NlogN)
S.C → O(1)]

① Sum of first N natural no's.

arr[] → [1 2 3 4 5 6]
0 1 2 3 4 5

$\frac{6 \times 7}{2} = 21$

arr[] → [-2 1 2 3 4 5 8]

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

sum = 21

ans = 7

ans → 7.

idea-4. Keep the element at its correct position.

N = 5.

i = 0 → 1
i = 1 → 2
i = 2 → 3
i = 3 → 4
i = 4 → 5

i → i+1.

val > N
val ≤ 0
→ ignore

arr[] = [4 2 7 6 9 1 5 3]
indices: 0 1 2 3 4 5 6 7
4 → 6 → 1
7 → 8 → 9 → 3
6 → 4
9 → 5
1 → 6
5 → 7
3 → 9

N = 8.

ans = 8

i = 0 swap(arr[0], arr[3]) swap(arr[0], arr[5])

i = 1 do nothing

i = 2. swap(arr[2], arr[6]), swap(arr[2], arr[4])

i = 3. do nothing

i = 4 do nothing

i = 5 do nothing

i = 6. do nothing

i = 7. swap(arr[7], arr[2])

arr[] → [5 | -14 | 6 | 7 | 9 | 10 | 2 | 3 | 8 | 12 | 1]

indices: 0 1 2 3 4 5 6 7 8 9 10

N = 11

2 3 -14 5 6 7 8 9 10 12

[ans = 4]

i = 0 swap(arr[0], arr[4]), swap(arr[0], arr[8]), swap(arr[0], arr[7])

swap(arr[0], arr[2]), swap(arr[0], arr[5]), swap(arr[0], arr[9])

i = 1. -

i = 2. -

i = 3. swap(arr[3], arr[6]), swap(arr[3], arr[1])

i = 4 -

i = 5 -

i = 6 -

i = 7 -

i = 8 -

i = 9 -

i = 10 swap(arr[10], arr[0])

Observation: In a single swap, at least one element will go to its correct position.

At max, how many swaps are possible ≈ N swaps

pseudo-code

```
for( i = 0; i < N; i++){
    while (arr[i] > 0 && arr[i] ≤ N && arr[i] != i+1){
        val = arr[i];
        if (arr[i] == arr[val-1]) { break }
        swap( arr[i] with arr[val-1]);
    }
}
```

// iterate again to find first missing natural no.

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

i	no. of swaps
0	S_0
1	+ S_1
2	+ S_2
⋮	+ ⋮
N-1	+ S_{N-1}
	=

total no. of swaps.

T.C → O(N)
S.C → O(1)

Duplicates ?

arr[5]: [1 3 3 3 1 | 1 3 2 | 3 3 3 | 3 4 | 2 3]

[ans = 5]

i = 0. swap(arr[0], arr[3]), swap(arr[0], arr[2]), swap(arr[0], arr[2])

— ——— infinite loop.

```
if (arr[i] == arr[val-1]){
    break;
}
```

i = 1 swap(arr[1], arr[0]) ∵ arr[1] == arr[3-1] (break)

i = 2 -

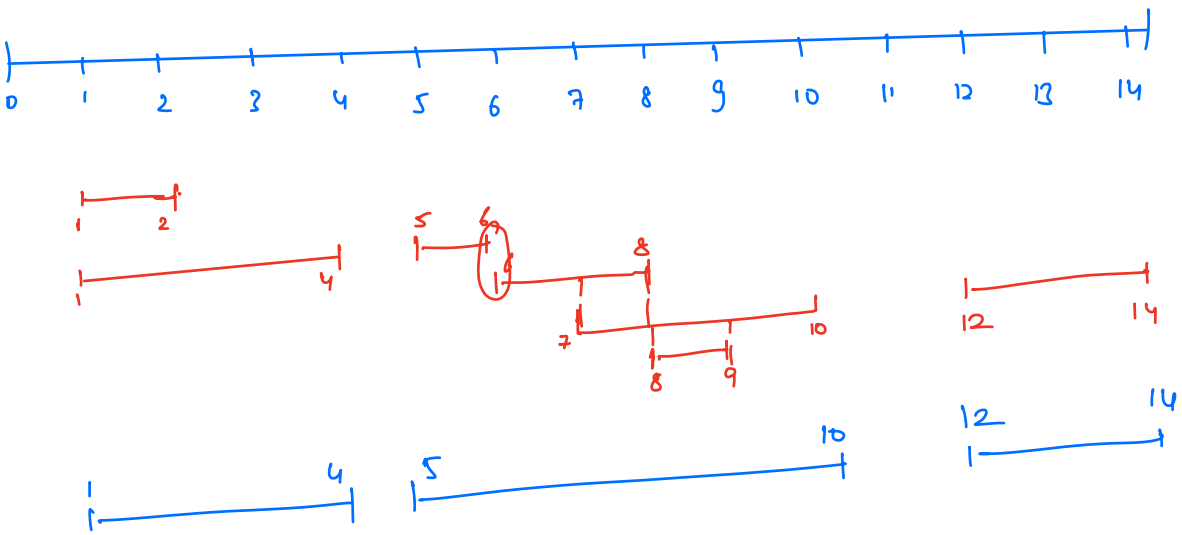
i = 3 -

i = 4 swap(arr[4], arr[1]) ∵ arr[4] == arr[3-1] (break)

Q: Given a sorted list of overlapping intervals, sorted based on start-time. Merge all overlapping intervals & return the sorted list of non-overlapping intervals.

Google
Paypal

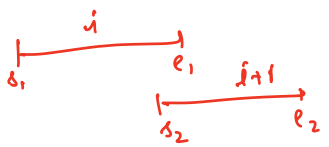
N=7.



idea → iterate from left to right →

① try to merge the adjacent intervals if they are overlapping.

② condition for overlap ?



$s[i+1] \leq e[i]$

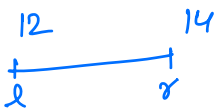
③ What is s and e of merged interval ?

$\text{Min}(s[i], s[i+1])$, $\text{Max}(e[i], e[i+1])$

$s[i]$.

s[] → [0 1 5 6 7 8 12]

e[] → [2 4 6 8 10 9 14]



ans → (0,4), (5,10), (12,14)

code :→

```
l = s[0],   r = e[0]
for( i = 1;  i < N;  i++){
    if( s[i] ≤ r) {  //overlap
        r = Max( r, e[i])
    }
    else{
        print( l + "-" + r);
        l = s[i],   r = e[i]
    }
}
print( l + "-" + r);
```

T.C → O(N)
S.C → O(1)

s[] → [0 1 5 6 7 8 12]

e[] → [2 4 6 8 10 9 14]

l = 0 5 12

r = 2 4 6 8 10 14

o/p - [0 4
5 10
12 14]

Given N non-overlapping intervals sorted based on start time. Given a new interval. Merge this with existing intervals, if possible & return final non-overlapping intervals.

N = 8.

interval

s[] → [1 4 10 16 21 27 32 38]

e[] → [3 7 14 19 24 30 35 45]

(L = 10, R = 22)

[10, 24]

ans → [[1 4 10 27 32 38], [3 7 24 30 35 45]]

N = 5.

s[] → [1 8 11 15 21]

e[] → [5 10 14 20 24]

(L → 12, R → 22)

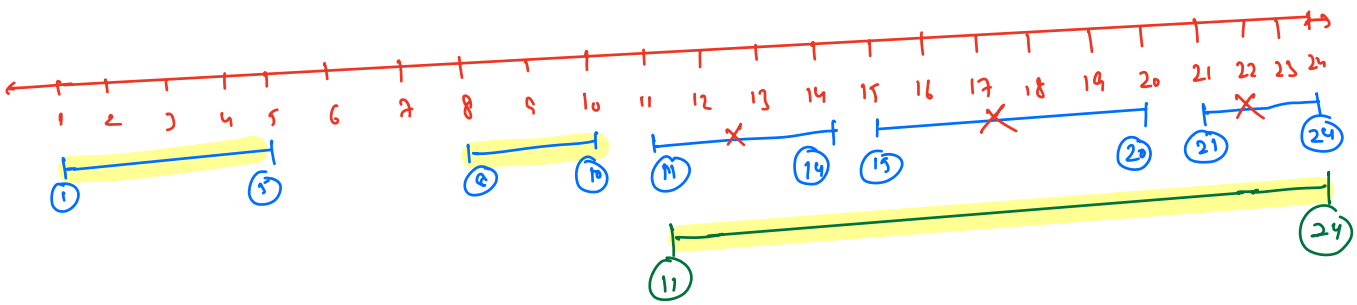
[11, 24]

ans → [[1 8 11], [5 10 24]]

$S[] \rightarrow [\ 1 \quad 8 \quad 11 \quad 15 \quad 21\ ]$

$E[] \rightarrow [\ 5 \quad 10 \quad 14 \quad 20 \quad 24\ ]$

$\begin{bmatrix} L \rightarrow 12 \\ R \rightarrow 22 \end{bmatrix}$



#code.

```
void mergeinterval( s[], e[], L, R) {
    for( i = 0; i < N ; i++) {
        if( e[i] < l) { //non-overlap
            print( s[i] + "-" + e[i]);
        }
        else if( s[i] > R) { //non-overlap
            print( l + "-" + r);
            for( j = i ; j < N; j++) {
                print( s[j] + "-" + e[j]);
            }
            return
        }
        else{      //overlapping intervals
            l = min( l, s[i]);
            r = max( r, e[i]);
        }
    }
    print( l + "-" + r);
}
```

T.C → O(N)
S.C → O(1)