SCALER

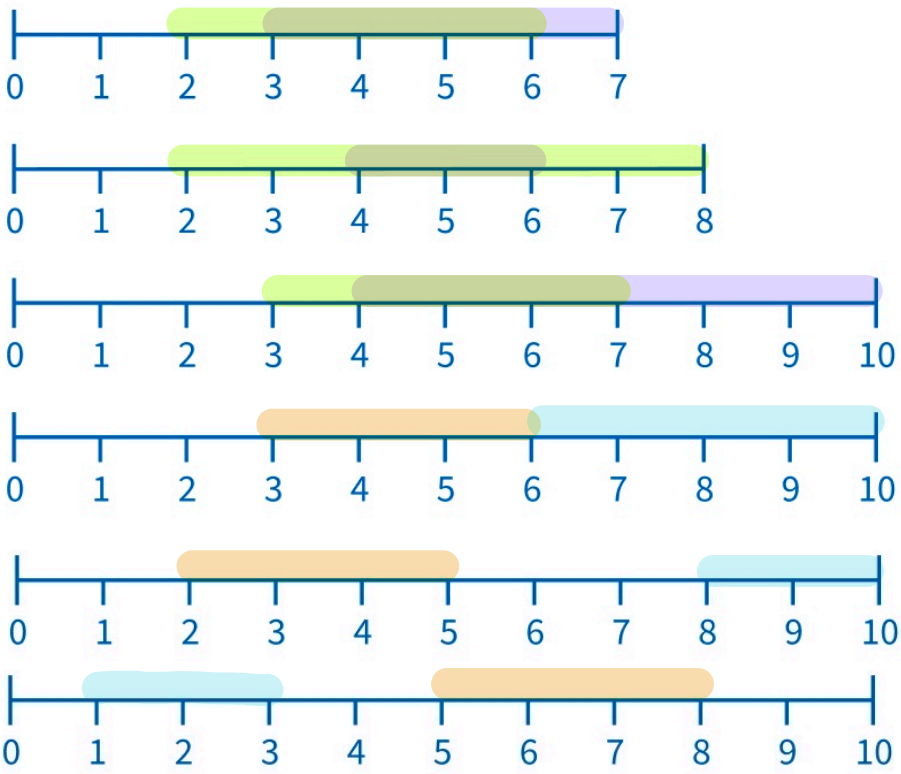
Arrays 3: Interview

TABLE OF CONTENTS

Merge Overlapping Intervals

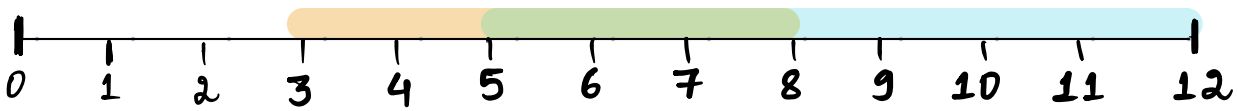
Ans

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)	→	No overlap
(5, 8)	(1, 3)	→	No overlap



Quiz 1 :

(3,8) (5,12)

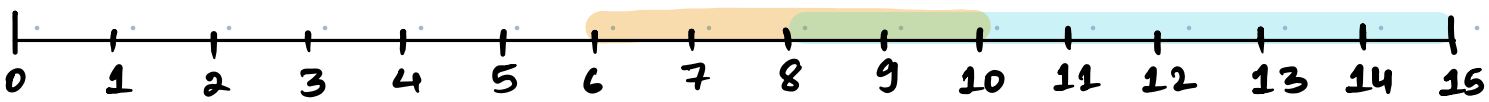


(3, 12)

Merge Overlapping Intervals

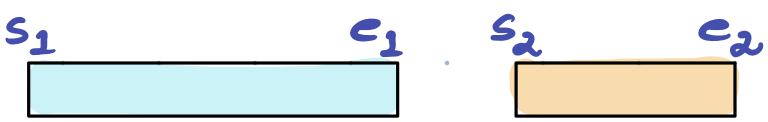
Quiz 2 :

(6,10) (8,15)



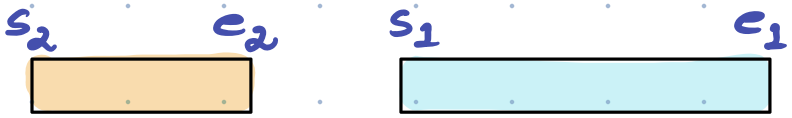
(6, 15)

NON-OVERLAPPING INTERVALS



$s_2 > e_1$

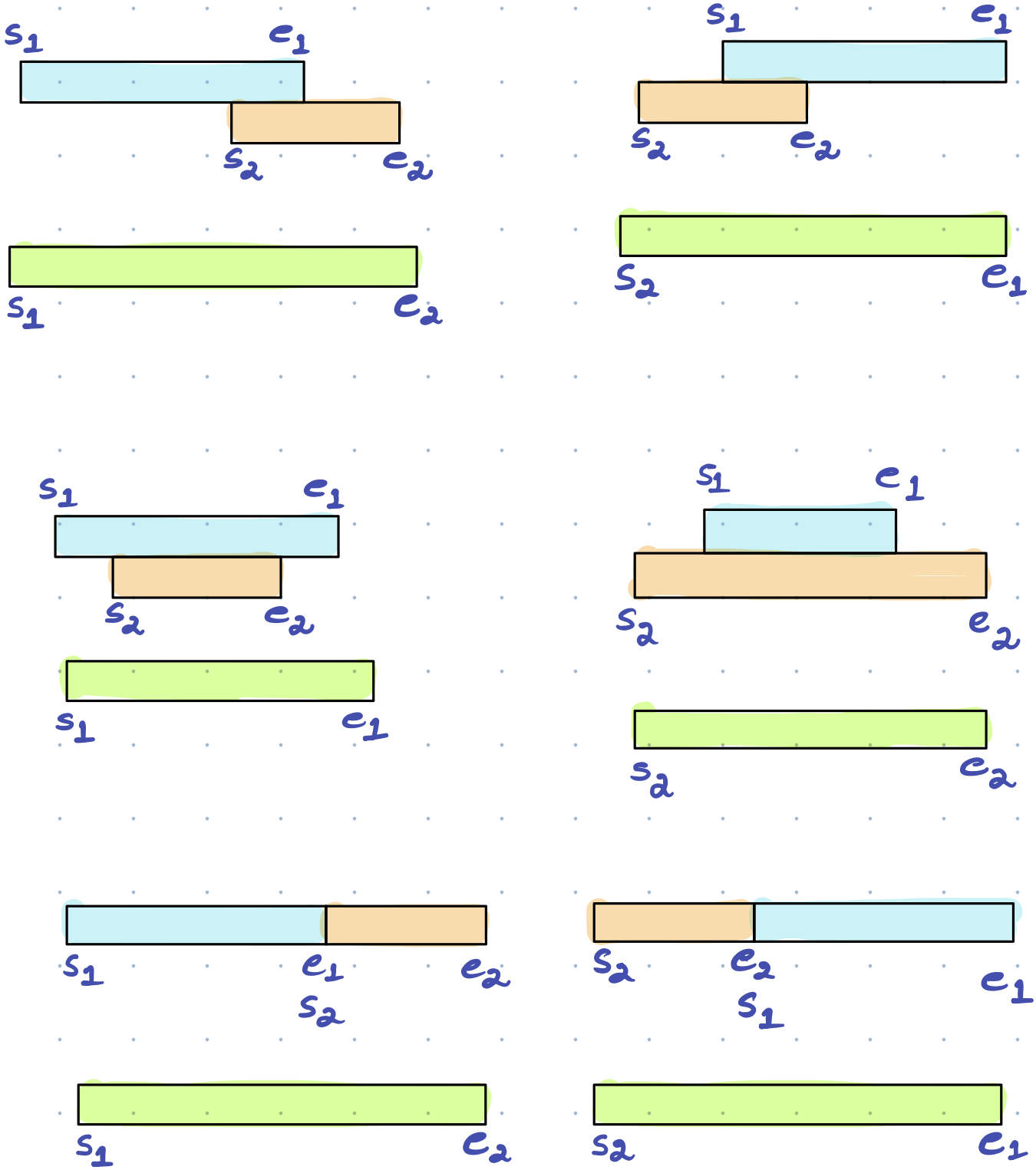
or



$s_1 > e_2$

Merge Overlapping Intervals

OVERLAPING INTERVALS



Starting pt of merged interval : $\min(s_1, s_2)$

Ending pt of merged interval : $\max(e_1, e_2)$

Overlapping interval : $[\min(s_1, s_2), \max(e_1, e_2)]$

Merge Sorted Overlapping Intervals

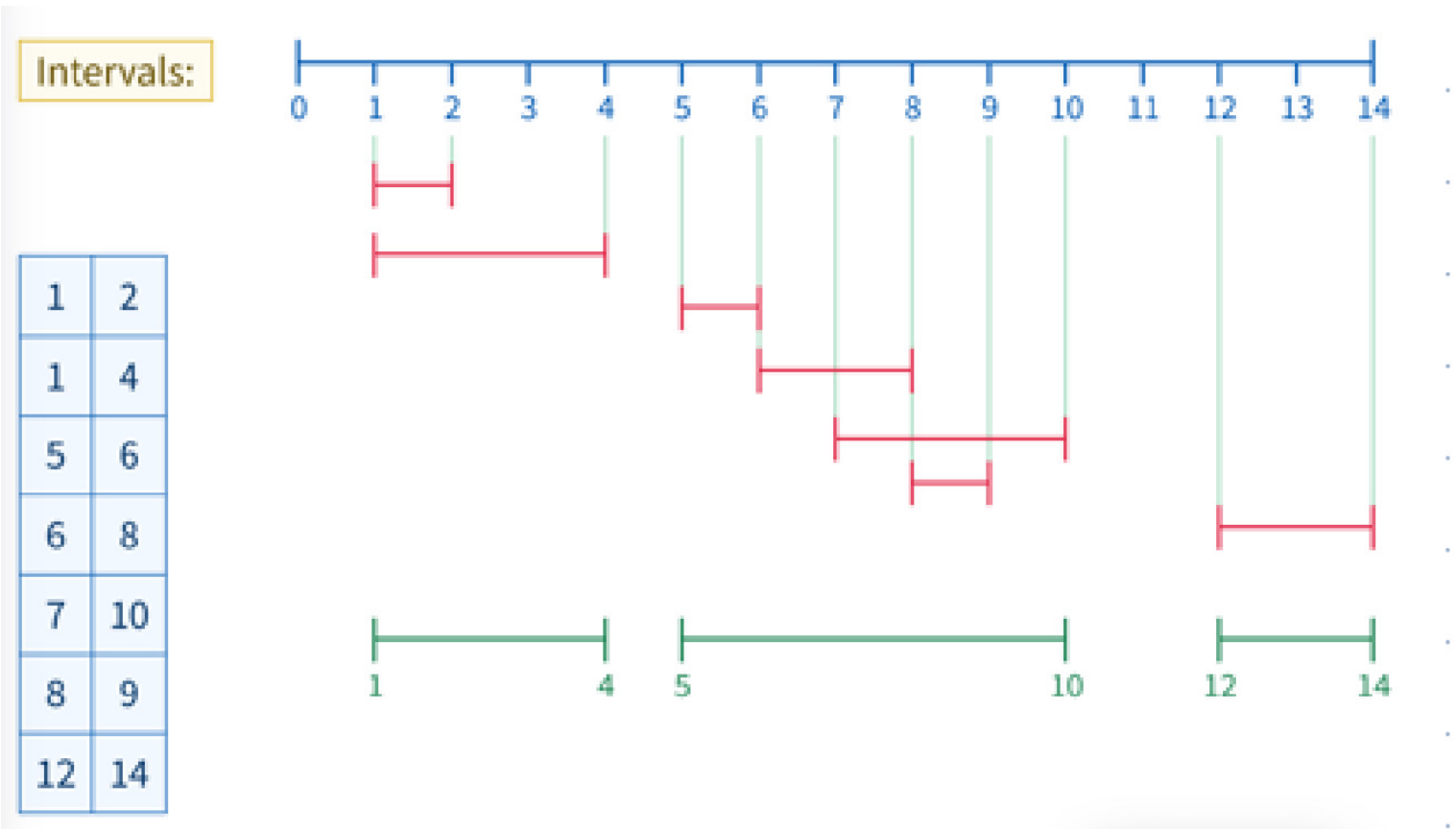
Question

Google

PayPal

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

$[\ 1 \le N \le 10^J\ ]$



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

Merge Sorted Overlapping Intervals

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

Interval 1	Interval 2	Overlapping?	Merged Interval	Ans
[1,2]	[1,4]	✓	[1,4]	
[1,4]	[5,6]	x		[1,4]
[5,6]	[6,8]	✓	[5,8]	
[5,8]	[7,10]	✓	[5,10]	
[5,10]	[8,9]	✓	[5,10]	
[5,10]	[12,14]	x		[5,10]
[12,14]	—			[12,14]

o/p : [[1,4] [5,10] [12,14]]

Quiz 3 :

Given a sorted list of overlapping intervals, sorted based on start time, merge all overlapping intervals and return sorted list.

Input:

Interval[] = { (1,10), (2, 3), (4, 5), (9, 12)}

Interval 1	Interval 2	Overlapping?	Merged Interval	Ans
[1,10]	[2,3]	✓	[1,10]	
[1,10]	[4,5]	✓	[1,10]	
[1,10]	[9,12]	✓	[1,12]	
[1,12]	—			[1,12]

o/p : [1,12]

Code :

```
int start_1 = arr[0][0];
int end_1 = arr[0][1];

for(i=1; i<N; i++){
    int start_2 = arr[i][0];
    int end_2 = arr[i][1];
    if(end_1 >= start_2){ // overlapping
        start_1 = min(start_1, start_2);
        end_1 = max(end_1, end_2);
    }
    else{ // Non-overlapping
        ans.insert({start_1, end_1});
        start_1 = start_2;
        end_1 = end_2;
    }
}
// Insert last interval
ans.insert({start_1, end_1});
return ans;
```

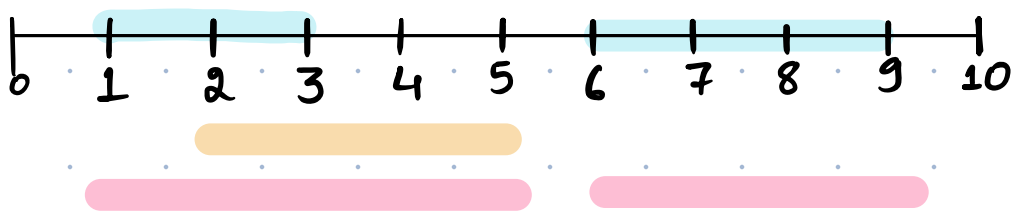
T.C : O(N)
S.C : O(1)

Merge Intervals II

Question

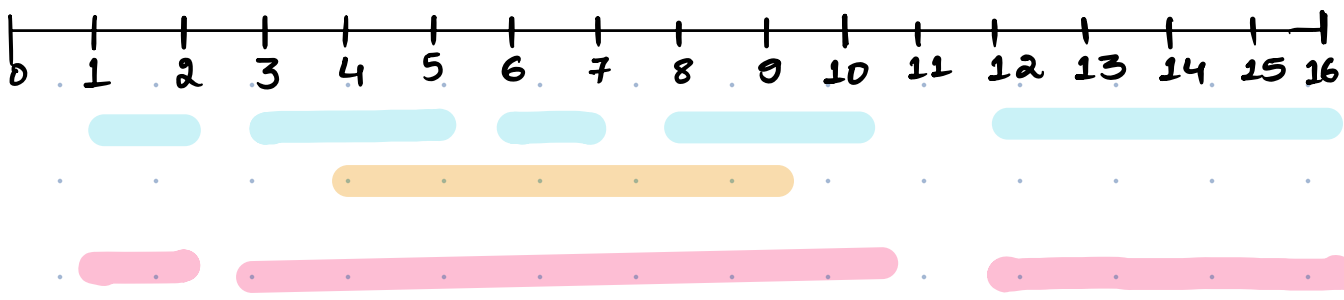
Given N non-overlapping intervals sorted based on start time. Given a new Interval. Merge this with existing intervals if possible & return final non-overlapping interval.

```
Intervals = [[1,3],[6,9]]
newInterval = [2,5]
```



o/p : [[1,5] [6,9]]

```
Intervals = [[1,2],[3,5],[6,7],[8,10],[12,16]]
newInterval = [4,9]
```



o/p : [[1,2] [3,10] [12,16]]

Merge Intervals II

Example :

	New Interval	Overlapping?	Ans
(1, 3)		✗	(1, 3)
(4, 7)		✗	(4, 7)
(10, 14)	(10, 22) → (10, 22)	✓	
(16, 19)	(10, 22) → (10, 22)	✓	
(21, 24)	(10, 22) → (10, 24)	✓	
(27, 30)	(10, 24)	✗	(10, 24)
(32, 35)			(27, 30) (32, 35)

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

Quiz :4

If the sorted set of ~~non-overlapping~~ intervals is [1, 5], [6, 10], and [9, 12], what happens if you add the interval `[4, 7]` such that the final list of intervals is also sorted and non-overlapping.?

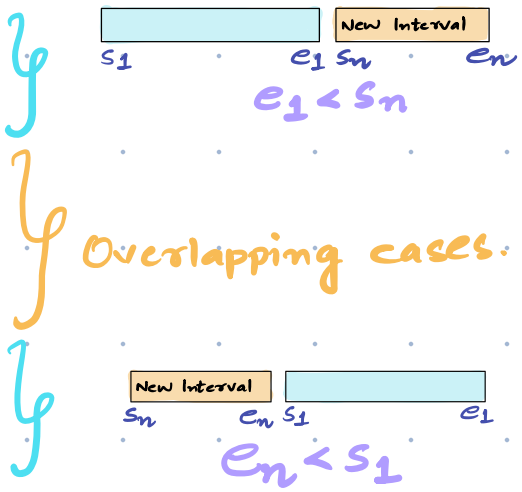
	New Interval	Overlapping?	Ans
(1, 5)	(4, 7) → (1, 7)	✓	
(6, 10)	(1, 7) → (1, 10)	✓	
(9, 12)	(1, 10) → (1, 12)	✓	
	(1, 12)		(1, 12)

o/p : [1, 12]

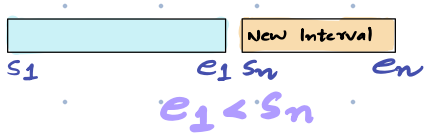
Merge Intervals II

Approach :

	New Interval	Overlapping?	Ans
(1,3)		✗	(1,3)
(4,7)		✗	(4,7)
(10,14)	(10,22) → (10,22)	✓	
(16,19)	(10,22) → (10,22)	✓	
(21,24)	(10,22) → (10,24)	✓	
(27,30)	(10,24)	✗	(10,24)
(32,35)			(27,30) (32,35)

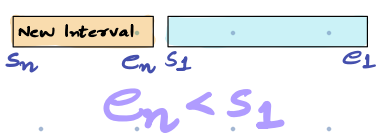


Case I :



Insert my current interval in ans

Case II :



- First insert new interval in ans
- Insert all remaining intervals in ans.
- Return ans.

Case III : Overlapping Intervals.

- Updating new interval as merged interval.

Merge Intervals II

Code :

```
// New interval
sn, en;

for(i=0; i<N; i++){
    s = arr[i][0];
    e = arr[i][1];
    if(e < sn){          // Non overlapping case I
        ans.insert({s,e});
    }
    else if(en < s){     // Non overlapping case II
        ans.insert({sn,en});
        for(j=i; j<n; j++){
            ans.insert({arr[j][0], arr[j][1]});
        }
        return ans;
    }
    else{                // overlapping case
        sn = min(sn, s);
        en = max(en, e);
    }
}
ans.insert({sn,en});
return ans;
```

T.C : O(N)

S.C : O(1)

Given N elements, find first missing +ve number

Given an unsorted array of integers, find first missing natural number.

Question

[1 . . . . N]



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

In the array [5, 3, 1, -1, -2, -4, 7, 2], what is the first missing natural number?

ans : 4

Given N elements, find first missing +ve number

Brute force approach

check for all no from 1 till we get ans.

val

1 — check if 1 is present: 0 → n

2 — check if 2 is present: 0 → n

3 — check if 3 is present: 0 → n

!

!

!

n — check if n is present: 0 → n

T.C : $O(N^2)$

S.C : $O(1)$

Idea -2 :

Sort the array.

-3	-7	1	10	3	8	2

[-7 -3 1 2 3 8 10]

val : ~~1~~ ~~2~~ ~~3~~ 4

ans : 4

T.C : $O(N \log N)$

S.C : $O(1)$

Given N elements, find first missing +ve number

Optimised approach :

Observation

Range of Ans : [1 . . . . N+1]

If array elements from [1 . . . n] are present
ans : (N+1)

If any no other than [1 . . . n] is present in array then missing no is out of 1 to N.

N : size of array

- Iterate & mark presence of no [1 . . . . N].

Area of Interest

(-) or No > N
Ignore

- Iterate again & check for missing no.

How to mark the presence of numbers from [1.....N] ?

Given N elements, find first missing +ve number

No's in array : [1 ... N]

Array Indices : [0 - ... N-1]

1 ⟶ 0^{th} index

2 ⟶ 1^{st} index

3 ⟶ 2^{nd} index

⋮

N ⟶ $(N-1)^{th}$ index

---

i ⟶ $(i-1)^{th}$ index

Use indices to mark presence.

- In 1^{st} iteration bring elements to their correct position. $i \longrightarrow (i-1)^{th}$ index
- Iterate again & check which index doesn't has correct value. $A[i] != (i+1) \rightarrow ans : (i+1)$
- If we have reached end of array that means no from [1 ... N] are present in my array.

ans : N+1

Given N elements, find first missing +ve number

Example : 1

N = 8

[1 ... 8]

i

	0	1	2	3	4	5	6	7
	5	3	1	-1	-2	-4	5	2
	-2 → 1	1 → -1 → 2	3		5			-2
Is no in range of [1....8]	✓	✓ ✓	✓	x	✓	x	✓	✓
Correct index	4	2 → 0	2		4		4	1
Is no equal to value at correct index:	x	x x	✓		✓			x

0	1	2	3	4	5	6	7
1	2	3	-1	5	-4	5	-2

i^{th} index → $(i+1)$

ans : $(i+1) = 4$

Given N elements, find first missing +ve number

Example : 2

N = 7 [1 ... 7]

i

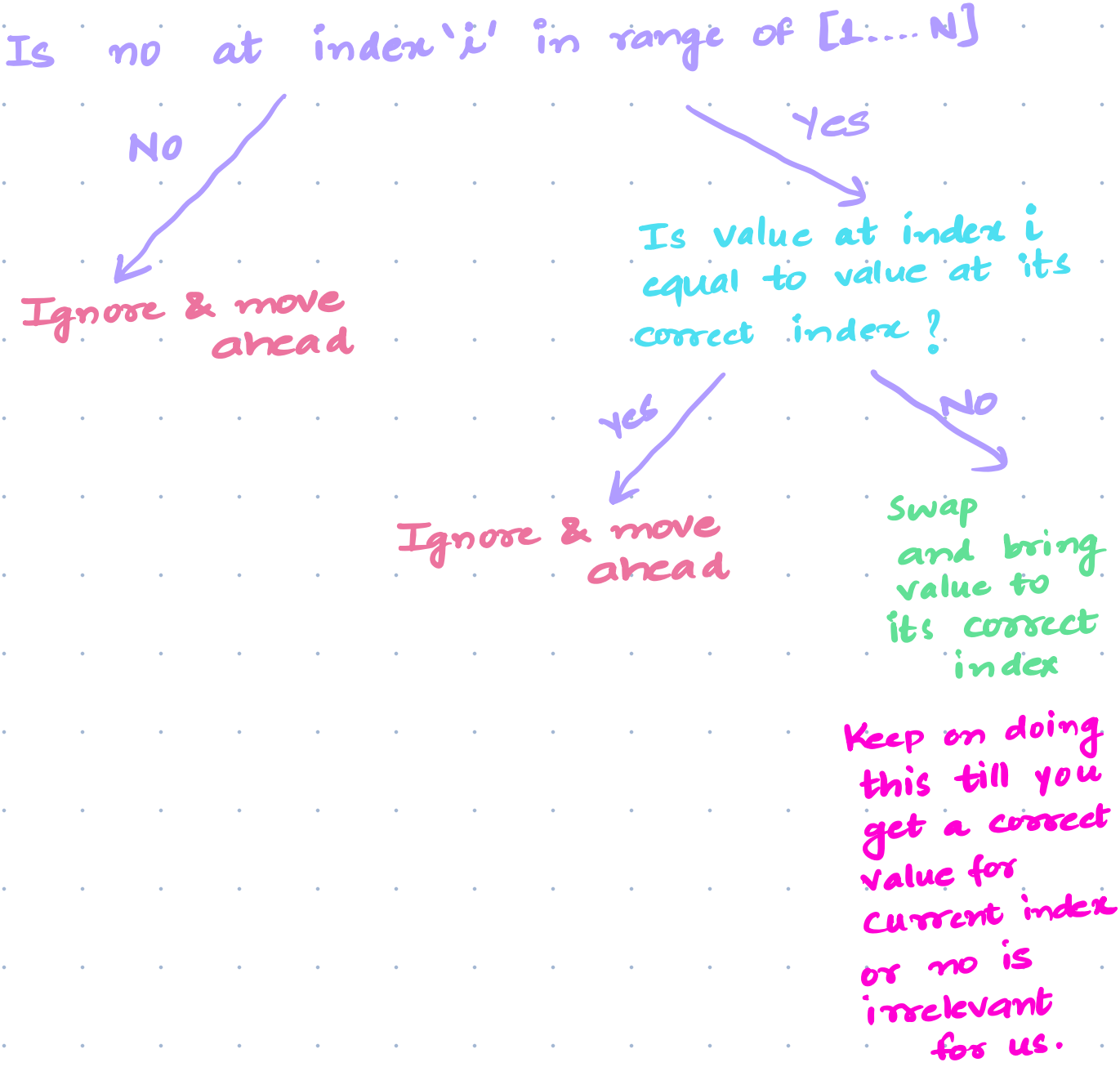
	0	1	2	3	4	5	6
	-3	-7	1	10	3	8	2
	1	2	-3 3		-3		-7
Is no in range of [1....7]	X	X	✓ X	X	✓ X	X	✓ X
Correct index			0		2		1
Is no equal to value at correct index.			X				

0	1	2	3	4	5	6
1	2	3	10	-3	8	-7

ans : 4

Given N elements, find first missing +ve number

Approach :



Given N elements, find first missing +ve number

Code :

```
int n = arr.size();
int i=0;

while(i<n){
    if(A[i]>=1 && A[i]<=N && A[i]!=A[A[i]-1]){
        swap(A[i], A[A[i]-1]);
    }
    else{
        i++;
    }
}

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

return (n+1);
```

In Range

Not Equal to value at correct index

T.C: O(N)

S.C: O(1)