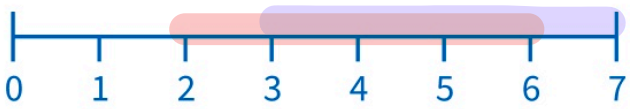
SCALER

Arrays 3: Interview

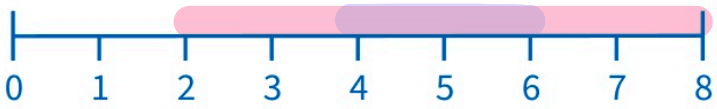
TABLE OF CONTENTS

Merge Overlapping Intervals

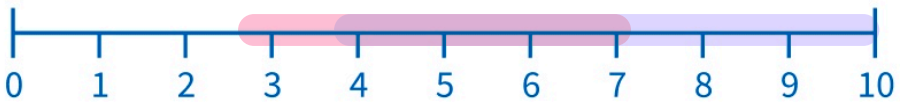
I_1 I_2

(2, 6) (3, 7) → 

ans : (2, 7)

(2, 8) (4, 6) → 

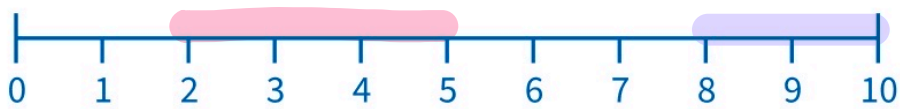
ans : (2, 8)

(3, 7) (4, 10) → 

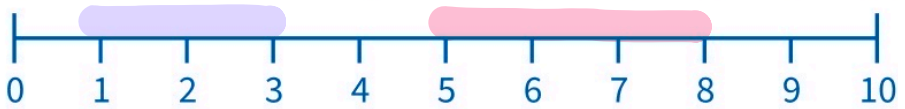
ans : (3, 10)

(3, 6) (6, 10) → 

ans : (3, 10)

(2, 5) (8, 10) → 

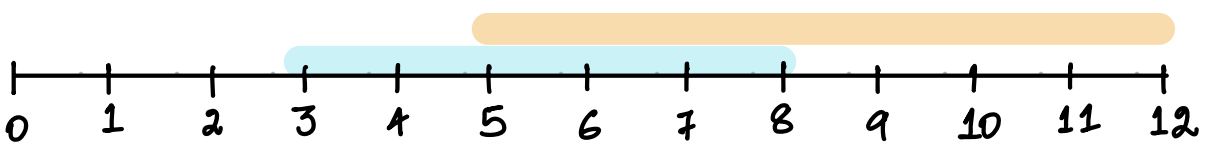
No overlap.

(5, 8) (1, 3) → 

No overlap.

Quiz 1 :

(3,8) (5,12)

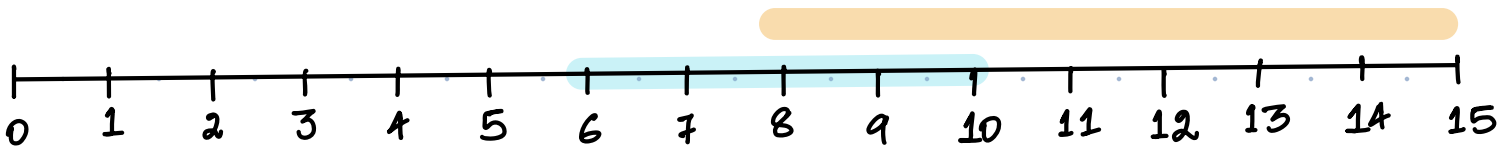


ans : (3, 12)

Merge Overlapping Intervals

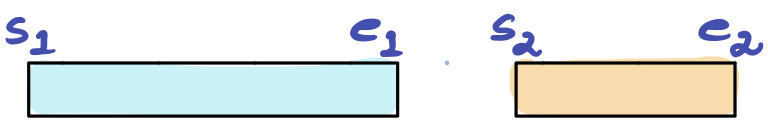
Quiz 2 :

(6,10) (8,15)



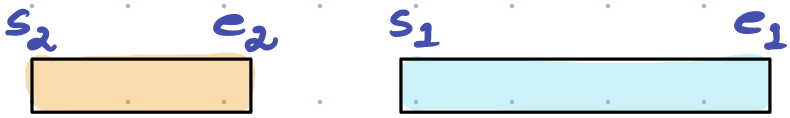
ans : (6, 15)

NON-OVERLAPPING INTERVALS



$e_1 < s_2$

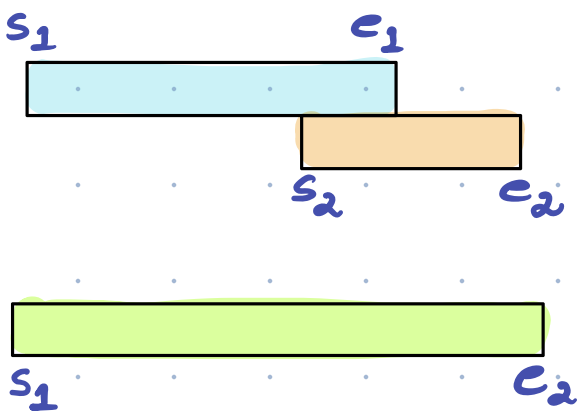
OR

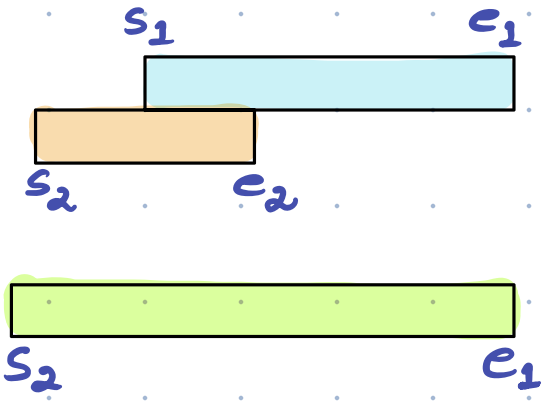


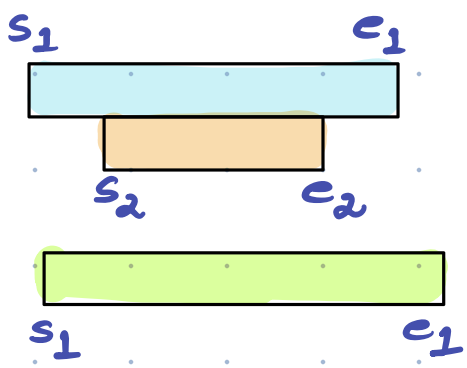
$e_2 < s_1$

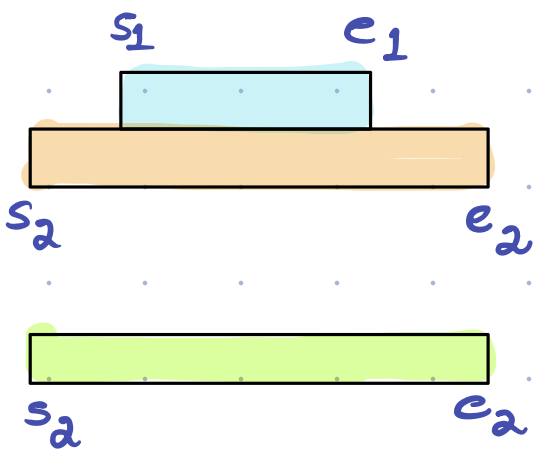
Merge Overlapping Intervals

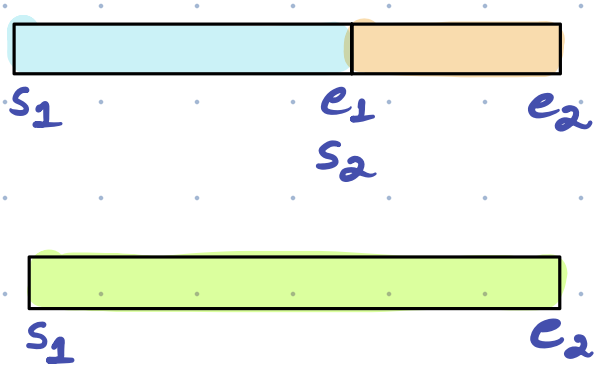
OVERLAPPING INTERVALS

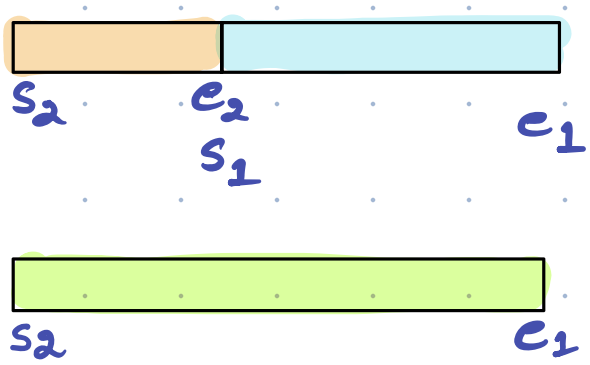












Starting Pt of Merged Intervals : $\min(s_1, s_2)$

Ending Pt of Merged Intervals : $\max(e_1, e_2)$

Overlapping Interval : $(\min(s_1, s_2), \max(e_1, e_2))$

Merge Sorted Overlapping Intervals

Question

Google

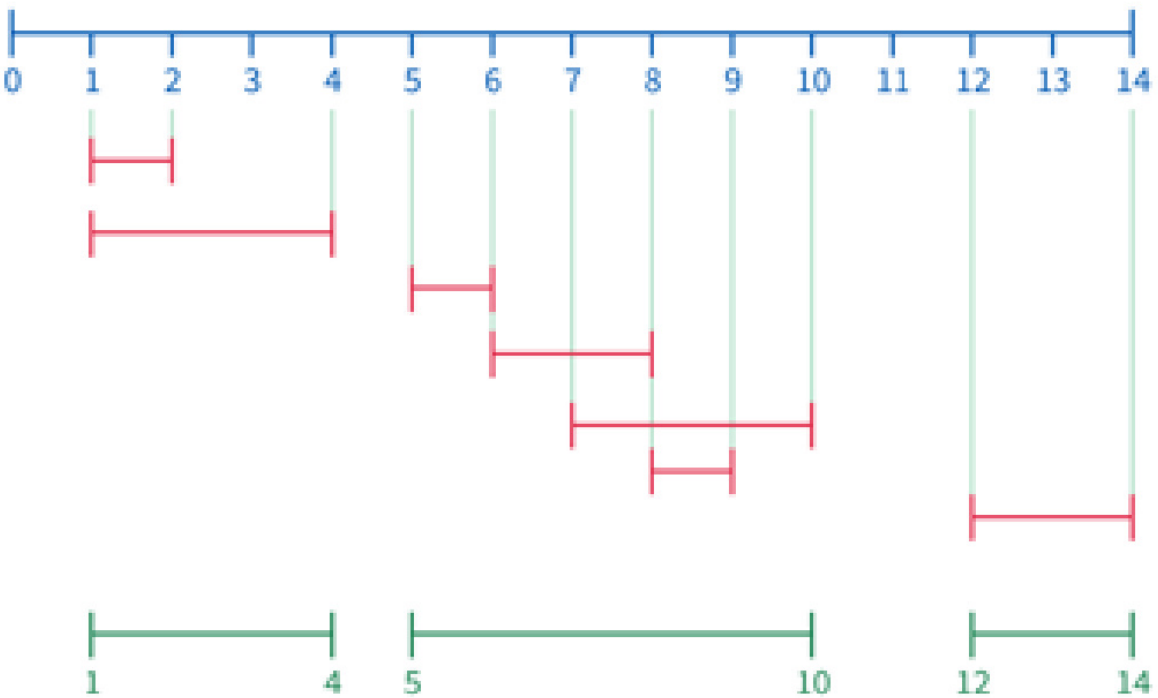
PayPal

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

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

Intervals:

start time	end time
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)

ans : ((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)

ans : ((1,12))

Merge Sorted Overlapping Intervals

Code :

```
int[][] MergeOverlappingIntervals(int[][] intervals){
    start_1 = intervals[0][0];
    end_1 = intervals[0][1];
    int[][] ans;
    for(i=1; i<intervals.size(); i++){
        start_2 = intervals[i][0];
        end_2 = intervals[i][1];
        if( start_2 ≤ end_1){          // 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;
        }
    }
    ans.insert({start_1, end_1});
    return ans;
}
```

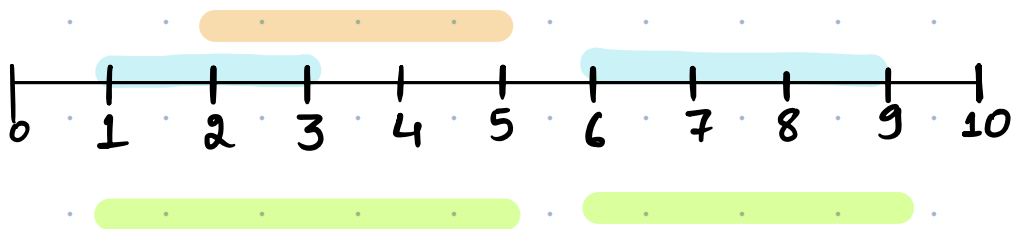
T.C : O(N)
S.C : O(1)

Merge Intervals II

Question

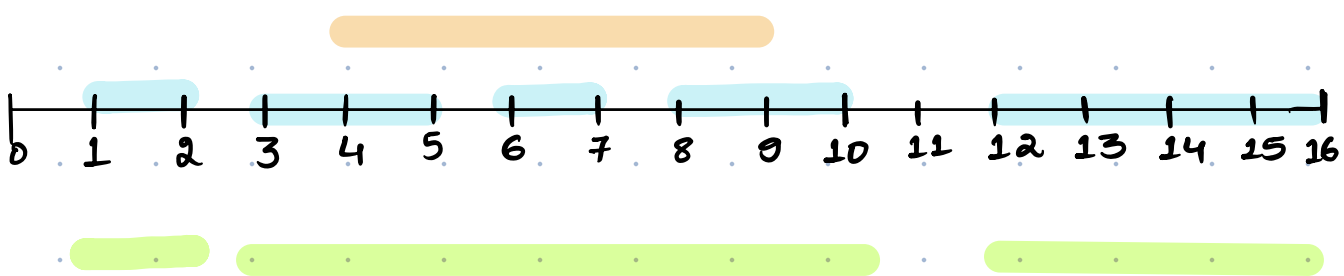
Given N non-overlapping intervals sorted based on start time. Given a new Interval. Merge this with existing intervals if possible & return final non-overlapping interval.

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



ans : ([1,5] [6,9])

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



ans: ([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)

	New Interval	Overlapping ?	ans
(1,5)	(4,11) → (1,11)	✓	
(6,10)	(1,11) → (1,11)	✓	
(11,12)	(1,11) → (1,12)	✓	(1,12)

ans : ((1,12))

Merge Intervals II

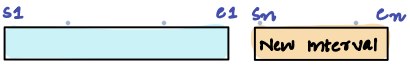
Approach :

	New Interval	Overlapping?	ans
(1,3)		X	(1,3)
(4,7)		X	(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)

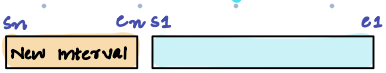
Non-overlapping

Overlapping.

Non-overlapping

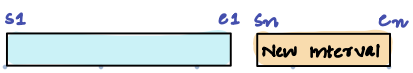


$e_1 < s_n$



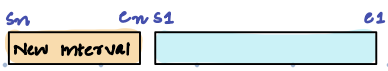
$e_n < s_1$

Case 1 : $e_1 < s_n$



Adding current interval into ans.

Case 2 : $e_n < s_1$



- First insert New interval into ans
- Insert all remaining intervals into ans
- Return ans

Non-Overlapping

Case 3 : Overlapping Intervals

Update New interval as Merged Interval.

Merge Intervals II

Code :

Start & end of New Interval

```
int[][] MergeOverlappingIntervals(int[][] intervals, int sn, int en) {
    int[][] ans;
    for (i = 0; i < intervals.size(); i++) {
        s1 = intervals[i][0];
        e1 = intervals[i][1];
        if (e1 < sn) {    // Non-Overlapping case 1
            ans.insert({s1, e1});
        }
        else if (en < s1) {    // Non-Overlapping case 2
            ans.insert({sn, en});
            for (j = i; j < intervals.size(); j++) {
                ans.insert({intervals[j][0], intervals[j][1]});
            }
            return ans;
        }
        else {    // Overlapping case.
            sn = Math.Min(s1, sn);
            en = Math.Max(e1, en);
        }
    }
    ans.insert({sn, en});
    return ans;
}
```

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

Given N elements, find first missing +ve number

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

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

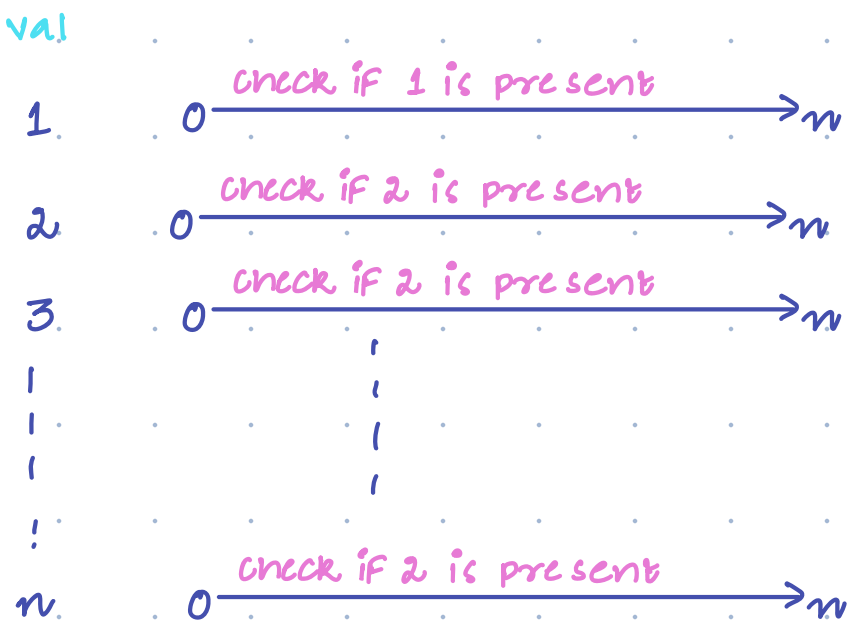
[8, 9, 11, 56]

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

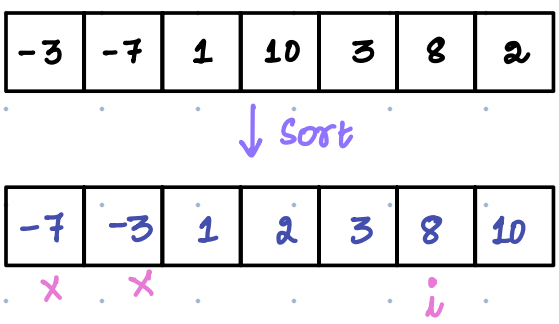
Brute force approach Starting from 1 check for all no in array.



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

S.C : $O(1)$

Idea -2 : Sort the array



val : x x 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

arr[4] → [1, 2, 3, 4]

- If array contains all no from [1 N] : ans : (N+1)

arr[5] → [3, -1, 1, 2, 7]

- If arrays contains no other than [1 N] → (-ve No) / (Numbers > N)

ans : [1 ... N]

Range of Ans : [1 , N+1]

Mark presence of no from 1 to N.

(-) or no > N → Ignore

Iterate again & check for missing no.

Given N elements, find first missing +ve number

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

No's : [1 , N]

array index : [0 , N-1]

1 ⟶ 0th index

2 ⟶ 1st index

3 ⟶ 2nd index

⋮

N ⟶ (N-1)th index

---

i ⟶ (i-1)th index

A[i] != (i+1) ans : (i+1)

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

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

ans : 4

Given N elements, find first missing +ve number

Example : 2

N = 7

[1, 7]

i

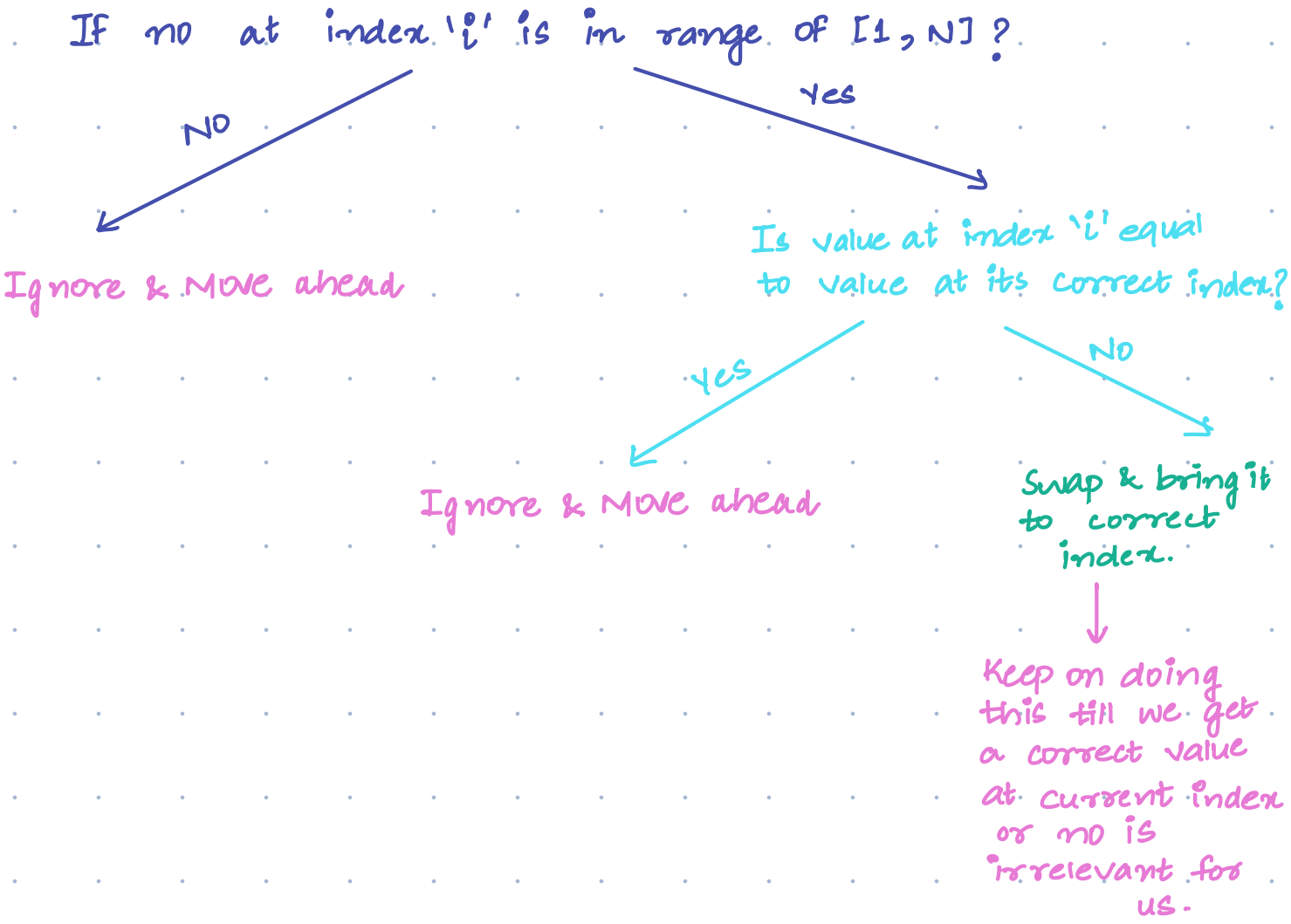
	0	1	2	3	4	5	6
	~~-3~~	~~-7~~	~~1~~	10	~~3~~	8	~~2~~
	1	2	~~-8~~ 3		-3		-7
Is no in range of [1....7]	X	X	✓	X	✓	X	✓
Correct index			0		2		1
Is no equal to value at correct index.							

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 :

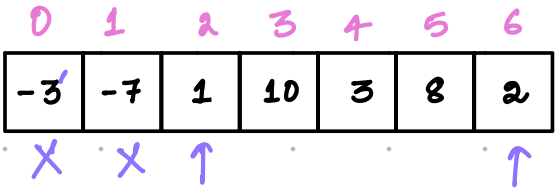
```
int i=0;
while (i<n) {
    if ( A[i] >=1 && A[i] <=n && A[i] != A[A[i]-1]) {
        Swap( A[i], A[A[i]-1]);
    }
    else {
        i++;
    }
}

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

return (n+1);
```

In Range

Not equal to value at its correct index.

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

DOUBTS :



i = 1

```
int i=0;
while (i<n) {
    if ( A[i] >= 1 && A[i] <= n)
        correct_idx = A[i] - 1;
        if ( A[i] != A[correct_idx]) {
            Swap( A[i], A[correct_idx]);
        }
    }
    else {
        i++;
    }
}
```

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