Q1) Find first missing natural number

Ex 1   3, -2, 1, 2, 7 → 4

-9, 2, 6, 4, -8, 1, 3 → 5

1, 2, 5, 6, 4, 3 → 7

size of array = n

max possible answer ⇒ n+1

answer is b/w 1 & n+1

- Idea ⇒ Store elements in hashset and start checking from 1 till the time no is not found.

```
for( i=0 ; i<N ; i++)
     hs.insert(arr[i])
for (i=1 ; i ≤ (N+1) ; i++) {
     if ( i not in hs) {
          return i
     }
}
```

TC: O(N)   SC: O(N)

Constraint Expected TC: O(N) SC: O(1)

Idea: Keep element in correct position.

N = 5

val	idx
1	0
2	1
3	2
⋮	
N	n−1

Generalize

val	idx
1	0
2	1
⋮	
x	x−1
⋮	
N	n−1

$\underbrace{\text{any val} > N \quad \text{any val} < 1}_{\text{ignore}}$

	0	1	2	3	4	5	6	7
ar[8]	1	2	3	4	9	6	7	8

i

0

1

2

3

4

5

6

7

Final array ⇒

0	1	2	3	4	5	6	7
1	2	3	4	9	6	7	8

Start from 0 idx & find missing no.

Ex 2

0	1	2	3	4	5	6	7	8	9
1	2	3	-1	4	5	6	7	8	9	10

ans = 4

0	1	2	3
-3	4	-2	4

Code

```
for(i=0; i<N; i++) {
    while(ar[i]>=1 && ar[i]<=n && ar[i]!=(i+1)) {
        int val= ar[i]
        if (ar[i] == ar[val-1]) break
        else swap (ar[i], ar[val-1])
    }
}
//iterate to get missing no.
for(i=0; i<N; i++) {
    if (ar[i] != (i+1))
        return i+1

return n+1
```

TC: $O(N)$ SC: $O(1)$

How TC $O(N)$

i	swap_count
0	S_0
1	S_1
2	S_2
⋮	⋮
$n-1$	S_{n-1}

$S_1 + S_2 + \ldots + S_{n-1}$ tot no of swaps

Obs: Each swap places atleast 1 elem in correct place.

max no of swaps = N

Hence $O(N)$

difference $\Rightarrow$ $a - b$

absolute value $x \geq 0 \Rightarrow x$

$x < 0 \Rightarrow -x$

Q2 Max absolute diff.

Given array A of size, find max of $|A_i - A_j| + |i-j|$

Eg-

$$\begin{array}{ccc} 0 & 1 & 2 \\ 1, & 3, & -2 \end{array}$$

1, 2

$\Rightarrow 3-(-2) + |1-2|$
$= 6$

Brute: nested for loops   TC: $O(n^2)$

Note: If I ask max of $A_i - A_j$
ans = maxval − minval.

Idea $f(i,j) = f(j,i)$

$\Rightarrow f(i,j) = |A_i - A_j| + (i-j)$ such that $i \geq j$

$A_i \geq A_j$	$A_i < A_j$
$A_i - A_j + i - j$	$A_j - A_i + i - j$.
$A_i + i - (A_j + j)$	$A_j - j - (A_i - i)$
$X_i - X_j$.	$X_j - X_i$
$X_k = A_k + k$	$Y_k = A_k - k$

$$\begin{array}{cccc} & 0 & 1 & 2 \\ A = & 1 & 3 & -2 \\ X = & 1 & 4 & 0 \end{array}$$

$max - min = 4$

$X_i - X_j$

$\Rightarrow max(X) - min(X)$

$$\begin{array}{cccc} & 0 & 1 & 2 \\ A = & 1 & 3 & -2 \\ Y = & 1 & 2 & -4 \end{array}$$

$max - min = 6$

$Y_i - Y_j$

$max(Y) - min(Y)$

$ans = max(ans_x, ans_y)$

TC: $O(N)$

SC: $O(N)$

x, y

for $i=0$; $i<n$ $i{+}{+}$

$x[i] = a[i] + i$
$y[i] = a[i] - i$

$ans_x = \max(x) - \min(x)$

$ans_y = \max(y) - \min(y)$

return $\max(ans_x, ans_y)$

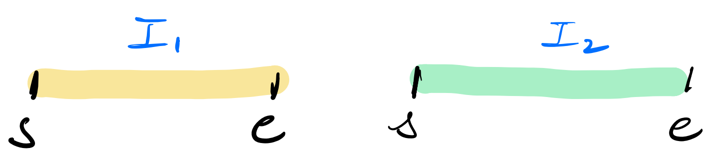
Q3 Merge intervals.

Interval is [a, b]

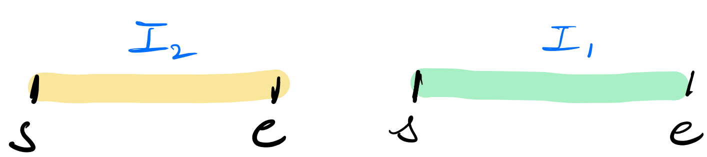
overlap ⇒ If intersecting at 1 or more elem, then they overlap.

Ex

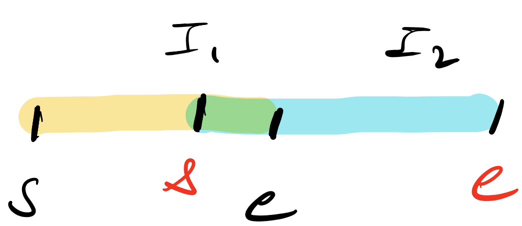
2,6	3,7	2,7
2,8	4,6	2,8
3,7	4,10	3,10
3,6	6,10	3,10
2,5	8,10	no overlap



$I_1.e < I_2.s$



$I_2.e < I_1.s$



merged interval

s: min$(I_1.s, I_2.s)$

e: max$(I_1.e, I_2.e)$

Given N non overlap intervals, sorted based on start
new interval I comes, merge all

[1,3]
[4,7]
[10,14]
[16,19]
[21,24]
[27,30]
[32,35]

I = ~~12,22~~
~~10,22~~
10 24

1,3
4,7
10,24
27,30
32,35
return.

N=5

[1,5]
[8,10]
[11,14]
[15,20]
[20,24]

I = ~~[12,22]~~
~~11,22~~
11,24

1,5
8,10
11,24

Code   Say ar[N] Intervals, Interval I comes

```
Intervals[] merge (Intervals ar[], Interval I){
                              arraylist<Interval>ans
 for(i=0; i<N; i++){
   // i^th Interval = ar[i]
   if ( ar[i].e < I.s)
        ans.insert (ar[i])
   else if ( I.e < ar[i].s ){
        ans.insert (I)
        for (j=i ; j<n ; j++)
            ans.insert (ar[j])
        return ans;
   }

   else {                 //
      // update Interval I after ?
      I.s = min (I.s, ar[i].s)
      I.e = max (I.e, ar[i].e)
 }
```

}

ans.insert(I)
return ans
}

TC : O(n)   SC: O(1)

{done}