Arrays Interview Problems

Today's Content

01 First Missing Integer

02 Insert Intervals

03 Search in a row wise,
col wise sorted matrix.

(Amazon /Google ....)

Q> Given A[N], find first missing natural number
1.........

Eg:

A[5] =	3	-2	1	2	7			4
A[7] =	-9	2	6	4	-8	1	3	5
A[6] =	1	2	5	6	4	3		7
A[5] =	-4	8	3	-1	0			1
A[4] =	4	2	1	30		$\underline{\boxed{31}}$		5
A[4] =	-8	-3	-1	-5				1

Idea - Brutforce

ans = [1, n+1]

```
→ i 1 → n+1 ──────────→ O(N)
  | i is not present in A ──────────→ O(N)
  |     return i
  }
                              TC : O(N²)
```

Put all element in Hashset

Idea 2>

```
  i 1 → n+1 ──────────→ O(N)
  | i is not present in HS ──────────→ O(1)
  |     return i
  }
                              TC : O(N)
                              SC : O(N)
```

Idea 3> Sort and check.

TC : NlogN

SC : 1

				✓	✓	✓	X
A	=	-1	0	1	2	3	5
		X	X	1	2	3	4 → return

A	=	-1	0	1	2	2	3	5
		X	X	1	2	X	3	4 → return

Idea → Sum of n natural no. - sum of all pos.

-1 -1 -1 $\frac{3 \times 4}{2} = 6 - 0$

X $= 6$

Optimised : Bring elements to its correct position

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

index	value
i	$i+1$
$v-1$	v

A[8] =

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

i

0    $A[0] \rightleftarrows A[3]$

$A[0] \rightleftarrows A[5]$

$i = i+1$

1    $A[1] == 2$

$i = i+1$

2

$A[2] \rightleftarrows A[6]$

$A[2] \rightleftarrows A[7]$

$i = i+1$

3    $A[3] == 4$ $i = i+1$

4    $A[4] = 9$ { since data is invalid we cont }

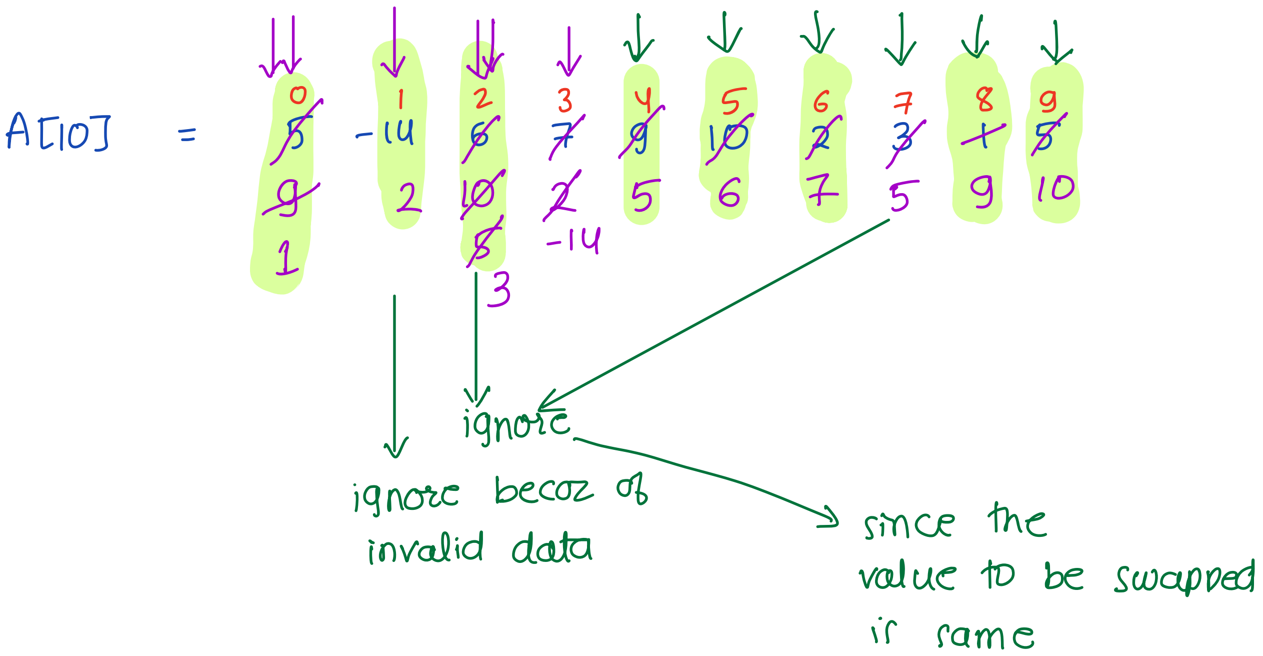
5

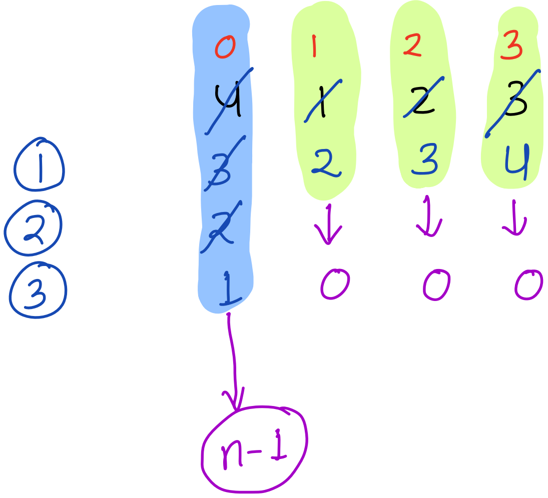
6

7

stop

→ loop through all values and check for mismatch
Return the first mismatch.



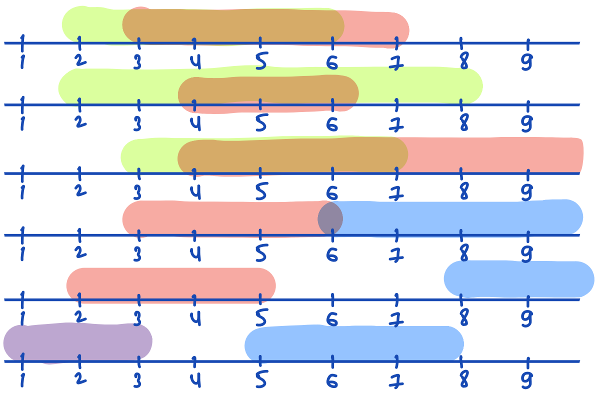


0 → O(n)
1 → O(1)
2 → O(1)
3 → O(1)

```
firstMissing ( A ) {

    for ( i → 0 to n-1 ) {
                     till val is not same                 valid value.
        while ( A[i] != i+1 && A[i] <= n && A[i] >= 1 ) {
               v = A[i]
               // Duplicate
               if( A[i] == A[v-1) { break }
               swap ( A[i] , A[v-1] )
        }

    }

    // check
  for ( i → 0 to n-1 ) {
        if ( A[i] != i+1 ) {  // mismatch
               return i+1
        }
  }

  return n+1
}
```

Merge Intervals

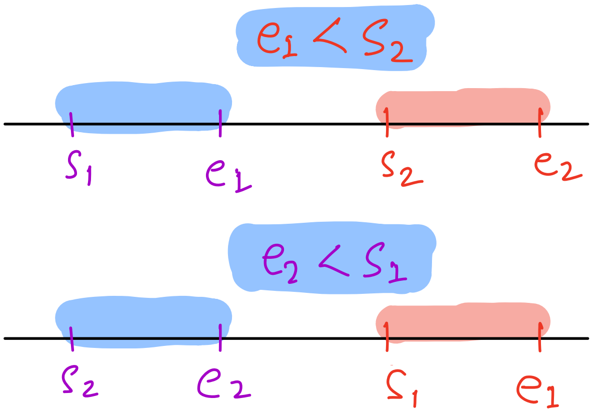
I_1	I_2	merged Interval
(2 6)	(3 7)	2 7
(2 8)	(4 6)	2 8
(3 7)	(4 10)	3 10
(3 6)	(6 10)	3 10
(2 5)	(8 10)	no overlap
(5 8)	(1 3)	



Non-overlap case

I_1 s_1 e_1

I_2 s_2 e_2



Overlapping Interval

I_1 s_1 e_1

I_2 s_2 e_2 $=$ $\min(s_1, s_2)$, $\max(e_1, e_2)$

Break : $\boxed{8:35}$

Q> Given N non-overlapping Intervals, in increasing order of their start times, insert new Interval {given}. and print non-overlapping intervals. {output}

N = 9	new Interval	Output
1 3	12 22	1 3
4 7	12 22	4 7
10 14	12 22 {10, 22}	10 24
16 19	10 22	27 30
21 24	10 22 {10, 24}	32 35
27 30	10 24	38 41
32 35		43 50
38 41		
43 50		

N = 5	new Interval	Output
1 5	12 22	1 5
8 10	12 22	8 10
11 14	12 22 {11 22}	11 24
15 20	11 22	
21 24	11 22 {11 24}	

N = 5	new Interval	Output
1 5	11 14	1 5
7 9	11 14	7 9
15 20	11 14	11 14
21 24		15 20
27 30		21 24

27   30

Pseudo

```
                                    new interval
merge ( int      A: [N][2] , int s , int e ){
    for (i ⟶ 0 to n-1 ) {
            $s_i$ = A[i][0]
            $e_i$ = A[i][1]

            if ($e_i$ < s ) { // 1   5            11   14
                print ($s_i$ , $e_i$)
            }

        else if (e < $s_i$ ) {    15   20          11   14
                print (s , e)
              for (j = i  to  n-1) {
                    print ( A[j][0], A[j][1]))
              }
              return ;  // exit from function.
        }
        else {  // overlap
              s = min($s_i$ , s)
              e = max ($e_i$ , e)
        }
    }
    print (s, e)
}
```

TC : O(N)

SC : O(1)

Q> Given a row & column sorted matrix. Return true if an element K exists.

R
C

	0	1	2	3	4	5
0	-1	2	4	5	9	11
1	1	4	7	8	10	14
2	3	7	9	10	12	18
3	6	10	12	14	16	20
4	11	15	19	21	24	27
5	18	24	29	32	34	42

K = 15

Brute force : Check each and every element if its K or not

TC : O(RC)
SC : O(1)

	0	1	2	3	4	5
0	-1	2	4	5	9	11
1	1	4	7	8	10	14
2	3	7	9	10	12	18
3	6	10	12	14	16	20
4	11	15	19	21	24	27
5	18	24	29	32	34	42

K = 15

```
searchMatrix ( A[][] , K ){
        R // No. of rows
        C // No. of cols

        r = 0 , c = C-1

        while ( c>=0 && r<R ){
                val = A[r][c]

                if(val == K){
                    return true
                }

                if(val < K){
                    r += 1
                }
                else{
                    c -= 1
                }
        }
        return false
}
```

```
1   2   3   4   5   6        K=0
7   8   9   10  11  12       K=100
```

Intervals

```
1  2        0  100  ->  0  100
4  5        0  100
6  7        0  100
```

new interval

```
merge(int  A[N][2], int s, int e){

    for(i -> 0 to n-1){
        si = A[i][0]
        ei = A[i][1]

        if(ei < s){ // 1  5          11  14
            print(si, ei)
        }

    else if(e < si){      15  20          11  14
        print(s, e)
        for(j = i to n-1){
            print(A[j][0], A[j][1])
        }
        return; // exit from function.
    }
    else{ //overlap
        s = min(si, s)
        e = max(ei, e)
    }
    }
```

TC : O(N)

SC : O(1)

```
    print (s, e)
}
```