# Today's Content

* First Missing Integer.
* Search in Matrix.
* Merge Intervals.

Qn 1: Given a[N]. Find the first missing natural number.

↳ 1, 2, 3, 4 ... ∞

Eg 1: 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, 1, 2, 3} : 5

a[5] = { 1 2 3 4 5 }
ans = 1 2 3 4 5 6

Generic case:
min_ans = 1
max_ans = n+1

idea 1: Iterate & check for missing no.

```
int missingNo1(int a[]){
    n = a.length
    for(i=1; i<=n; i++){
        if(search(a,i)==false){
            return i
        }
    }
    return n+1
}
```

↳ linear search : O(n)

TC: $O(n^2)$
SC: O(1)

idea 2: Check if numbers are present from 1-n using a hashset

```
int missingNo2(int a[]){
    n = a.length
    // Insert all elements into Hashset / Set = hs
    for(i=1; i<=n; i++){
        if(hs.search(i)==false){
            return i
        }
    }
    return n+1
}
```

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

## idea 3: Sort & Search

a[7] = { -9, 2, 6, 4, -8, 1, 3 }

↳ Sort: -9 -8 1 2 3 4 6

× × ✓ ✓ ✓ ✓ ⇒ ans = 5

a[8] = { -3, 1, 5, 8, 14, 2, 7, 3 }

↳ Sort: -3, 1, 2, 3, 5, 7, 8, 14

× ✓ ✓ ✓ ⇒ ans = 4

```
int missingNo3(int a[]) {
    n = a.length
    // Sort the array.
    ans = 1
    for(i=0; i< n; i++) {
        if (a[i] == ans) {
            ans += 1
        }
    }
    return ans
}
```

TC: O(n log n + n) = O(n log n)

SC: O(1)

## idea 4: Keep element at correct posn.

N = 5

| val | idx |
|---|---|
| 1 | 0 |
| 2 | 1 |
| 3 | 2 |
| 4 | 3 |
| 5 | 4 |

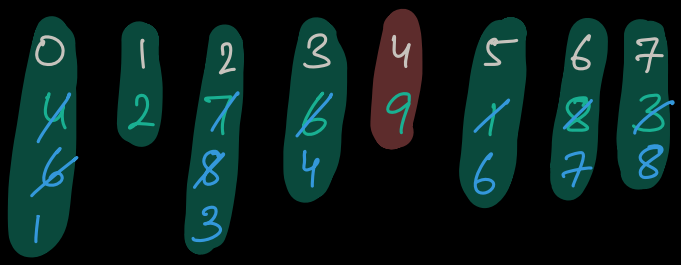
Generalize

| val | idx |
|---|---|
| 1 | 0 |
| 2 | 1 |
| ⋮ | ⋮ |
| x | x-1 |
| ⋮ | ⋮ |
| n | n-1 |

val > n } Skip it

val < 1 }

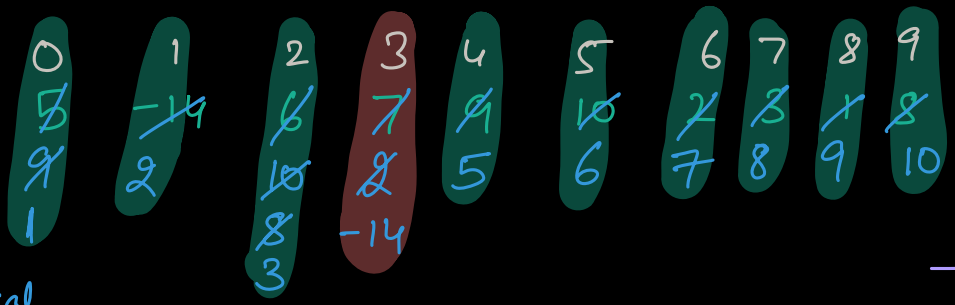
Eg:- a[8] =




ans = 5

| i | a[i] |
|---|---|
| 0 | 4 ⇒ swap(a[0], a[3]) ⇒ swap(a[0], a[5]) |
| 1 | 2 x |
| 2 | 7 ⇒ swap(a[2], a[6]) ⇒ swap(a[2], a[7]) |
| 3 | 4 x |
| 4 | 9 skip |
| 5 | 6 x |
| 6 | 7 x |
| 7 | 8 x |

a[10] =




ans = 4

| i | val |
|---|---|
| 0 | 5 ⇒ swap(a[0], a[4]) ⇒ swap(a[0], a[8]) |
| 1 | -14 skip |
| 2 | 6 ⇒ swap(a[2], a[5]) ⇒ swap(a[2], a[9]) ⇒ swap(a[2], a[7]) |
| 3 | 7 ⇒ swap(a[3], a[6]) ⇒ swap(a[3], a[1]) ⇒ skip |
| 4 | 5 x |
| 5 | 6 x |
| 6 | 7 x |
| 7 | 8 x |
| 8 | 9 x |
| 9 | 10 x |

```
int missingNo4(int a[]){
    n = a.length
    for(i=0; i<n; i++){
        while(a[i] > 0 && a[i] <= n && a[i] != i+1){   // a[i] within boundaries
            val = a[i]
            correct_idx = val - 1
            if(a[i] == a[correct_idx]) { break }   <- Edge case
            swap(a[i], a[correct_idx])
        }
    } // TC: O(n)
    // iterate & get missing No.
    for(i=0; i<n; i++){   <- O(n)
        if(a[i] != i+1){
            return i+1
        }
    }
    return n+1
}
```

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

## Edge Case:

a[5] = { 4, 1, 3, 3, 1 } { 0, 1, 2, 3, 4 } { 4, 1, 3, 3, 2 } { 3, 3, 3, 4, 2 } { 3, 2, 3, 2, 3 }

ans = 5

| i | val | |
|---|---|---|
| 0 | 4 | swap(a[0], a[3]) ⇒ swap(a[0], a[2]) ⇒ swap(a[0], a[2]) ∞-loop |
| 1 | 1 | swap(a[1], a[0]) |
| 2 | 3 | x |
| 3 | 4 | x |
| 4 | 2 | swap(a[4], a[1]) |

Break : 8:40am

TC part:

| i | # of swaps |
|---|---|
| 0 | a |
| 1 | b |
| ⋮ | ⋮ |
| n-1 | z |
| | = n swaps |

Qn: Given a 2d matrix (mat[n][m]). Every row is sorted. Every col is sorted.

Check if given element K is present or not.

| | 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 | 9 | 13 | 16 | 19 | 22 | 24 |
| 5 | 11 | 15 | 19 | 21 | 24 | 27 |
| 6 | 14 | 20 | 25 | 29 | 31 | 39 |
| 7 | 18 | 24 | 29 | 32 | 34 | 42 |

idea: iterate & search

TC: O(n*m)

SC: O(1)

idea 2: Binary search

(a) Apply BS on every row

TC: O(n log m)

(b) Apply BS on every col

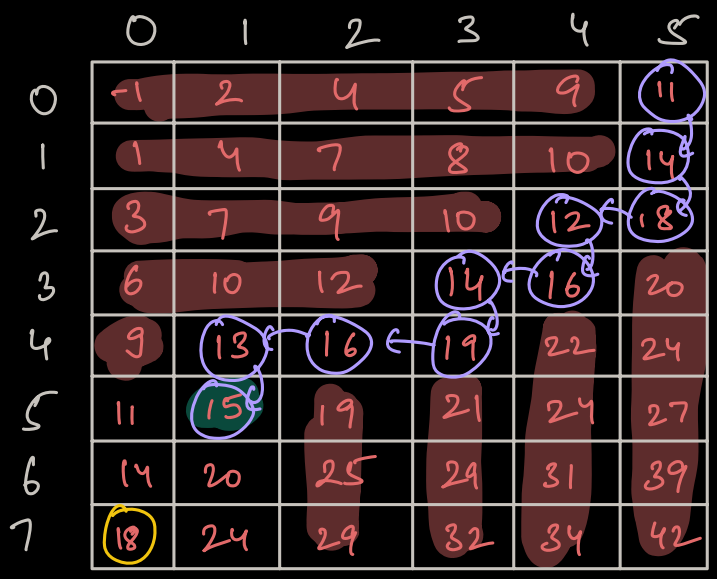
TC: O(m log n)

idea 3:

| | 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 | 9 | 13 | 16 | 19 | 22 | 24 |
| 5 | 11 | 15 | 19 | 21 | 24 | 27 |
| 6 | 14 | 20 | 25 | 29 | 31 | 39 |
| 7 | 18 | 24 | 29 | 32 | 34 | 42 |

K = 26

↳ false




K = 15

found ⇒ true

Pseudo code:

```
bool findK(int mat[][], int K) {
    n = mat.length
    m = mat[0].length
    i = 0, j = m-1
    while(i < n && j >= 0) {
        if (a[i][j] < K) { // skip row, go down
            i += 1
        }
        else if (a[i][j] > K) { // skip col, go left
            j -= 1
        }
        else { // found
            return true
        }
    }
    return false
}
```

TC: O(n+m)

SC: O(1)

# Merge Intervals

$s_1$ $I_1$ $e_1$ (2, 6)

$s_2$ $I_2$ $e_2$ (3, 7)

(2, 8)

(4, 6)

(3, 7)

(4, 10)

(3, 6)

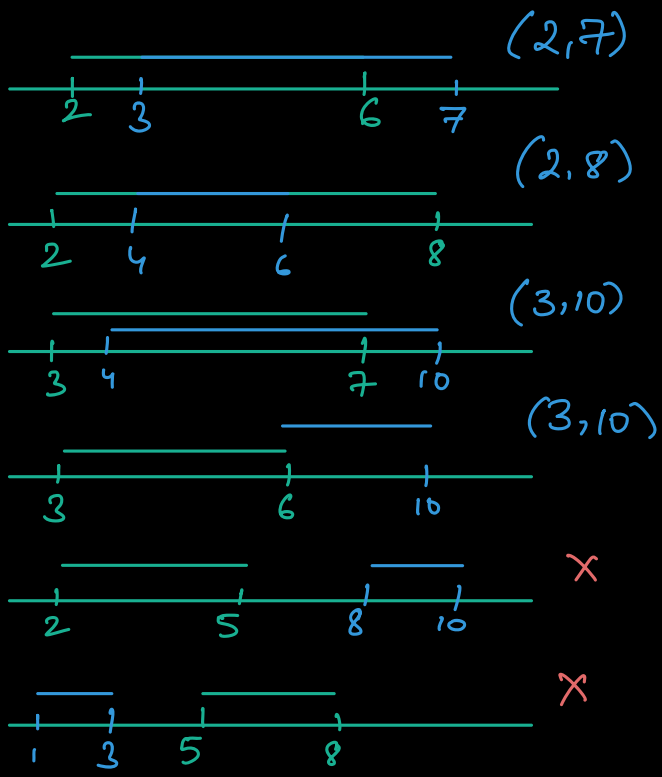
(6, 10)

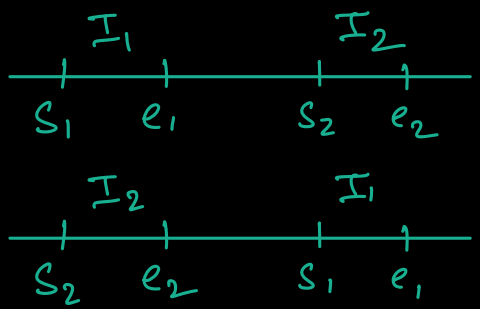
(2, 5)

(8, 10)

(5, 8)

(1, 3)




$e_1 < s_2$ ⇒ No overlap

$e_2 < s_1$ ⇒ No overlap

Overlap

$I_1$ $s_1$ $e_1$ $I_2$ $s_2$ $e_2$

$I$ $\{\min(s_1, s_2), \max(e_1, e_2)\}$

Qn: Given N non-overlapping intervals. They are sorted based on start. A new interval comes. Merge it with existing intervals. Return final set of non overlapping intervals.

N = 8

[1 3]
[4 7]
[10 14]
[16 19]
[21 24]
[27 30]
[32 35]
[38 41]

new interval [10 22]

[10 22] ⇒ [10 22]
[10 22] ⇒ [10 22]
[10 22] ⇒ [10 24]

ans

[1 3]
[4 7]
[10 24]
[27 30]
[32 35]
[38 41]

N = 5

new interval [12 22]

[1 5]
[8 10]
[11 14]
[15 20]
[21 24]

[12 22] ⇒ [11 22]
[11 22] ⇒ [11 22]
[11 22] ⇒ [11 24]

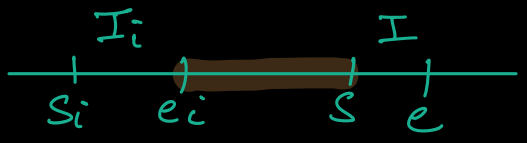
ans
[1 5]
[8 10]
[11 24]

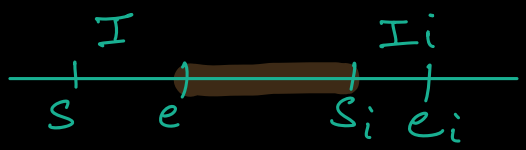
Pseudo code:

```
void mergeIntervals(int Intervals[N][2], int s, int e) {
    // Traverse the whole array.
    for(i=0; i<n; i++) {
        // ith interval = intervals[i][0], intervals[i][1]
        si = intervals[i][0], ei = intervals[i][1]

        if(ei < s) {
            print({si, ei})
        }
        else if(e < si) {
            print({s, e})

            for(j=i; j<n; j++) {
                print({intervals[j][0], intervals[j][1]})
            }
            return
        }
        else { // overlapping.
            s = min(s, si)
            e = max(e, ei)
        }
    }
    print({s, e})
}
```




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

```
0 [1   3]  [10  22]
1 [4   7]
2 [10  14]
3 [16  19]
4 [21  24]
5 [27  30]   <- i
6 [32  35]
7 [38  41]
```

S = ~~10~~ ~~16~~ 10 10

e = ~~22~~ ~~22~~ ~~22~~ 24

Si = 27

ei = 30

Output:

```
[1  3]
[4  7]
[10 24]
[27 30]
[32 35]
[38 41]
```