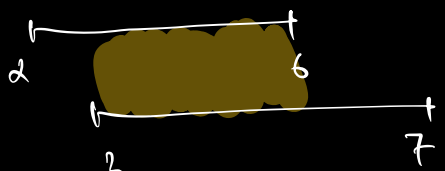
Q1. Merge Intervals



↓

overlapping

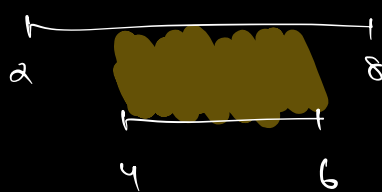
↓

merge

(2 to 7)

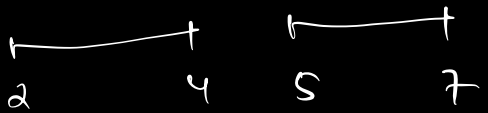
Q1

(2,8) (4,6) =>

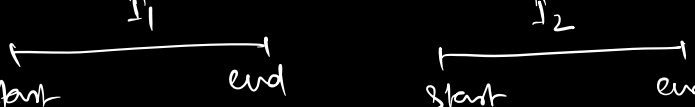


→ (2,8)

Q2. (2,4) (5,7)

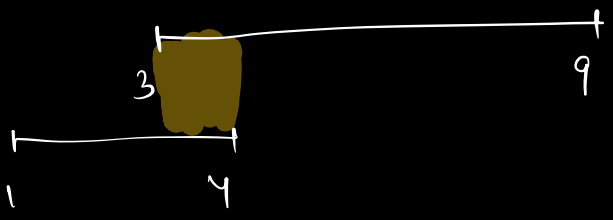


=> non-overlapping.



if I2.start > I1.end => no overlapping

Q-3 (3,9) (1,4)

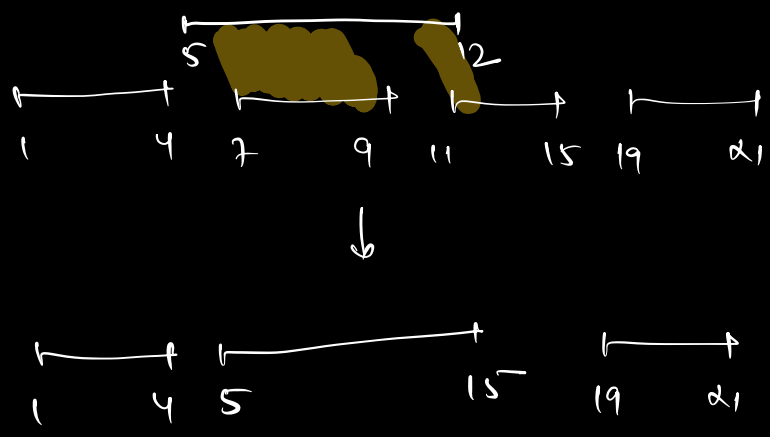


(1-9).

Q1. Merge Interval

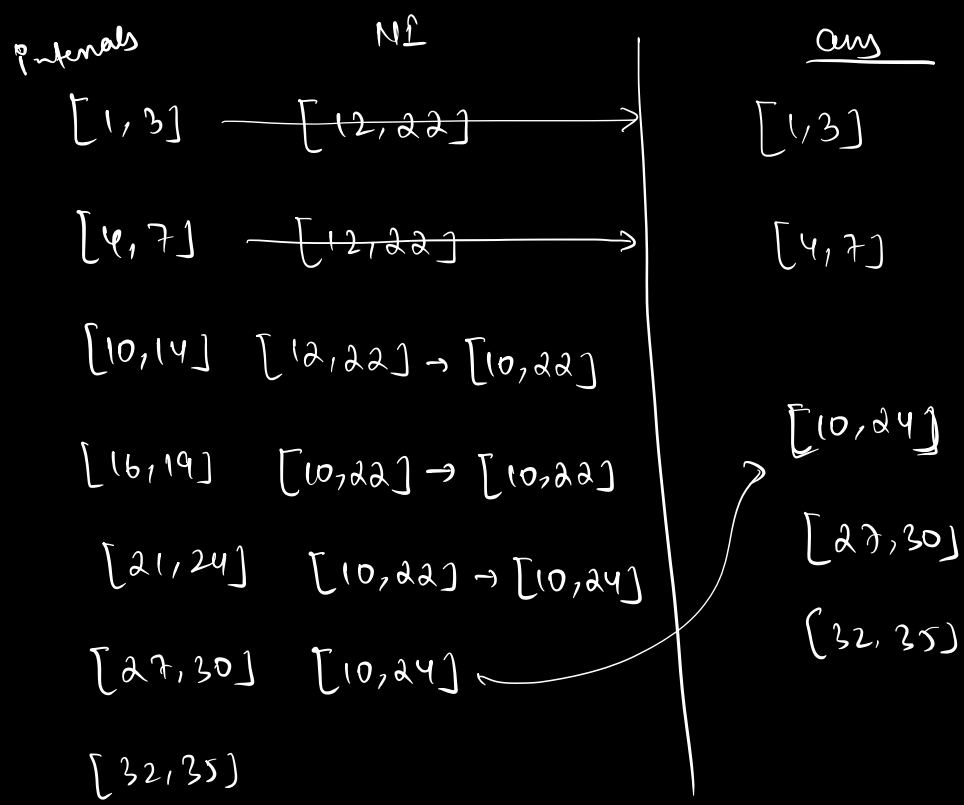
Given a sorted array of non-overlapping intervals, and given another new interval, add this interval in the given list, and update the list of intervals.

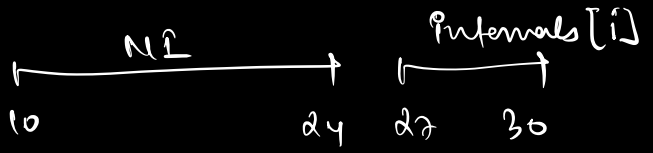
eg => [1,4]
[7,9]
[11,15]
[19,21]



NI => [5 12]

O/P => [[1,4] [5,15] [19,21]]






```
for( i=0; i<N; i++ ){
    1) if( intervals[i].end < NI.start-){
            ans.insert( intervals[i])
       }
    11) else if( NI.end < intervals[i].start ){
            ans.insert( NI)
            while( i < N){
                ans.insert( intervals[i])
                i++
            }
       }
    return ans
```

```
        } else {
            NI.start = min( intervals[i].start, NI.start )
            NI.end = max( intervals[i].end, NI.end )
        }
    }
    ans.insert( NI )
    return ans.
}
```

TC ⇒ O(N)
SC ⇒ O(N)

$[1 \le N \le 10^5]$

Q2. First missing natural no. / positive no.

Given an array of size N. find the first missing natural no. / positive no in the array. If all present (1→N) (N+1)

ex ⇒ { 3 -2 1 2 7 } ⇒ N = 5 (1→5)

1 2 3 ✗ ⇒ O/p = 4.

ex ⇒ { -9, 2, 6, 4, -8, 1, 3 } ⇒ N = 7

1 2 3 4 ✗ O/p = 5. (1→7)

Q 4. $\{1, 2, 5, 6, 4, 3\}$ $N = 6$

$1 \to 6$ $(1 \to N)$

1 2 3 4 5 6 $N+1$

$O/p = 7$

Q 5. $\{1, 0, -5, -6, 4, 2\}$ $N = 6$

1 2 ✗ $1 \to 6$ $O/p = 3$

① Brute force (N) $1 \to N$

Linear (1 to N)

TC ⇒ $O(N^2)$

SC ⇒ $O(1)$

② Sorting

array ⇒ $\{-1, 0, 3, 6, 9, 4, 1\}$

sorted ⇒ $\{-1, 0, 1, 3, 4, 6, 9\}$ $N = 7$

✗ ✗ ✓ ↑ 1 to 7

$O/p = 2$

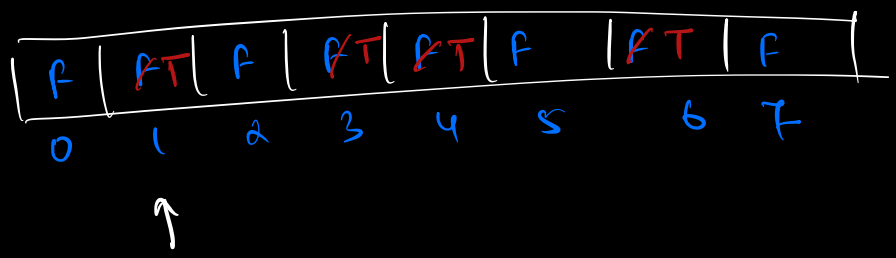
TC ⇒ $O(N \log N + N) \simeq O(N \log N)$

SC ⇒ $O(\ )$ → depends on sorting algo

(III) Hashmap | Hashset | array

ex ⇒ { -1, 0, 3, 6, 9, 4, 1 } N = 7 (1→7)

present arr ⇒ [F]

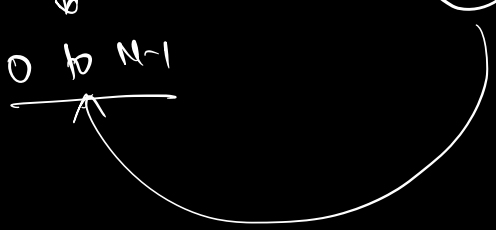



```
for(i=1; i<=N; i++){
    if( present[i] == false)
        return i
}
return N+1;
```

TC ⇒ O(N)
SC ⇒ O(N)

(IV) * update the input array :

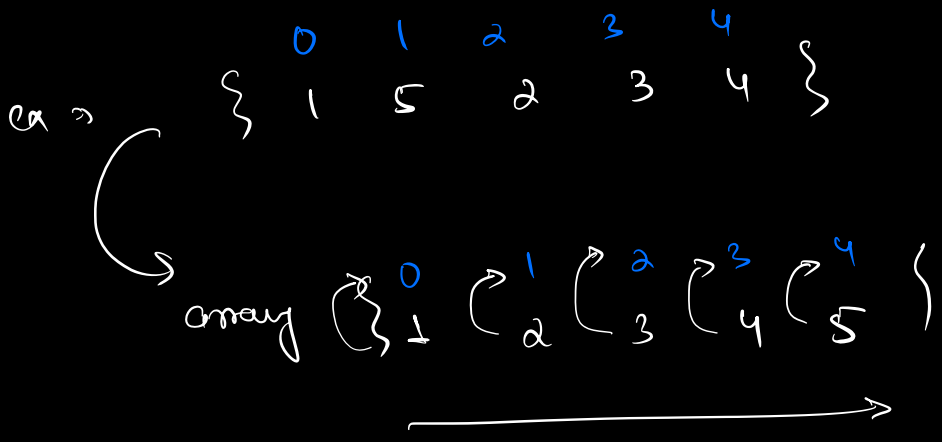
arr ⇒ N ⇒ 1 to N.
(-1)
↓
0 to N-1



element	index
3	2
x	x-1
arr[i]	arr[i]-1

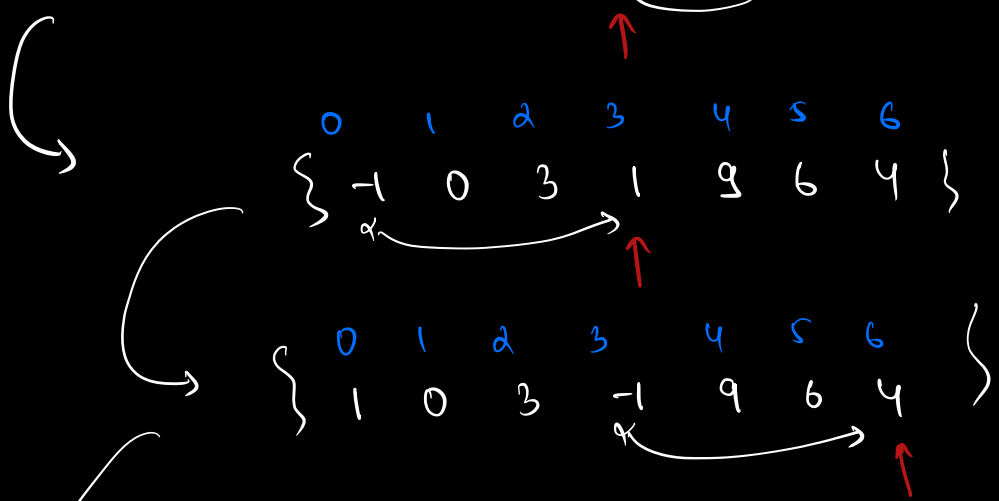
if we arrange all nos. to their respective idx ($x, x-1$) then the first idx where arrangement fails.

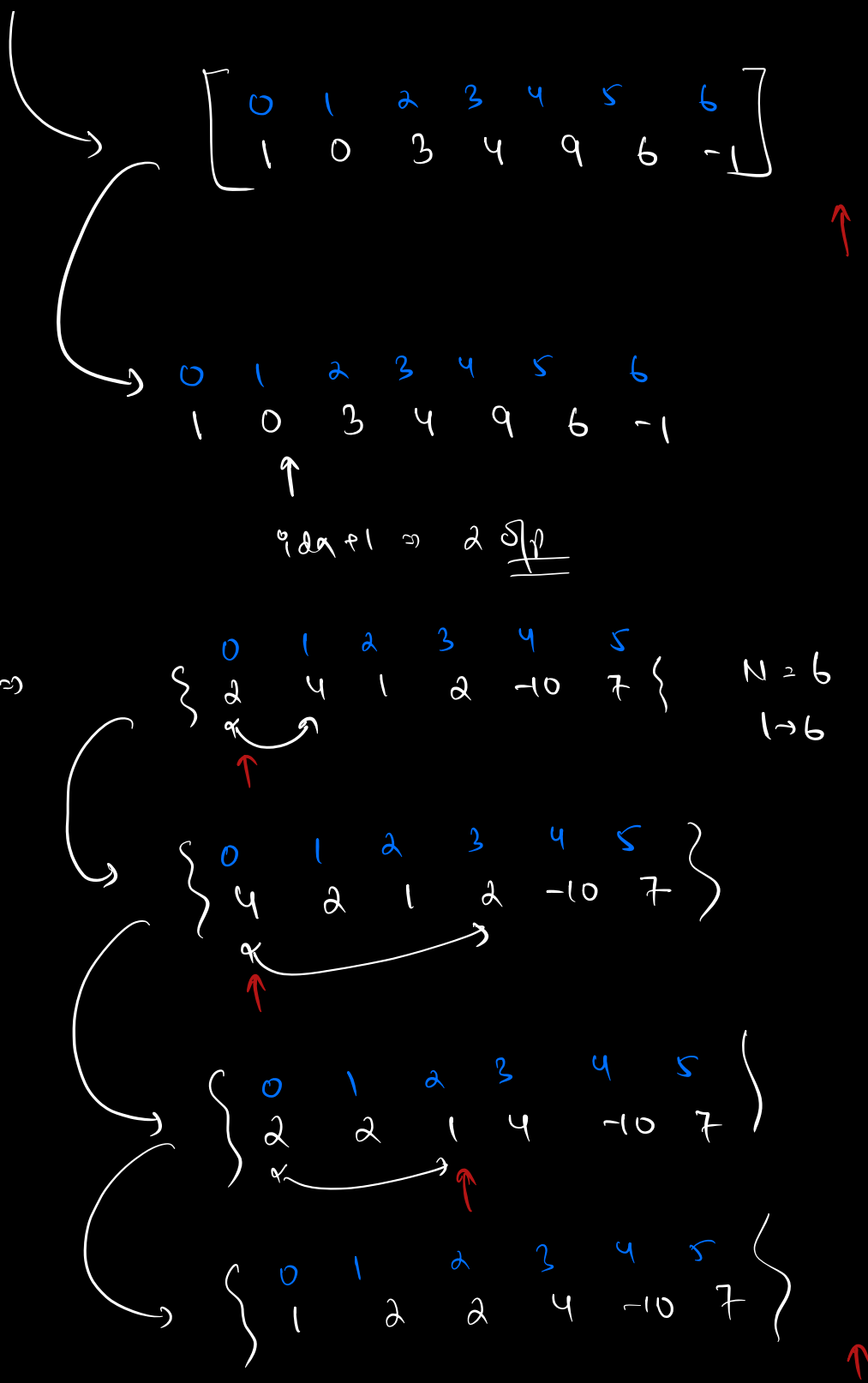
ex ⇒

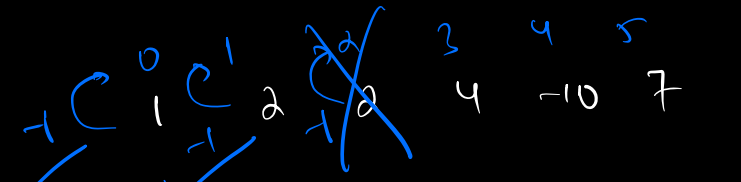


O/P = 6.

ex ⇒ arr ⇒ { -1 0 3 6 9 1 4 } n = 7, 1-7




0 1 2 3 4 5 6
1 0 3 4 9 6 -1
0 1 2 3 4 5 6
1 0 3 4 9 6 -1
2*a+1 ⇒ 2 OP
eg ⇒
0 1 2 3 4 5
2 4 1 2 -10 7
N = 6
1→6
0 1 2 3 4 5
4 2 1 2 -10 7
0 1 2 3 4 5
2 2 1 4 -10 7
0 1 2 3 4 5
1 2 2 4 -10 7



O(p = 3)

Pseudo

```
i = 0
while (i < N) {
  if( arr[i] >= 1  && arr[i] <= N) {
      correctIdx = arr[i] - 1
      if( arr[correctIdx] != arr[i]) {
          swap( arr[correctIdx], arr[i])
      } else {
          i++
      }
  } else {
      i++
  }
}

for( i = 0 → N-1) {
    verify → i+1
}
N+1
```

TC ⇒ O(N)
SC ⇒ O(1)

MAANG

Q3. Maxm absolute difference

Given an arr[N]. Find maxm value

$$|arr[i] - arr[j]| + |i-j|.$$

Ex ⇒

0	1	2	3	4
2	5	4	1	6

$|6-1| + |4-3| = 5+1 = 6X$

$|6-2| + |4-0| = 4+4 = 8$ ✓

BF

all possible combns of i & j. find maxm

TC ⇒ O(N²)

SC ⇒ O(1)

$$|x| = \begin{cases} -x & x < 0 \\ x & x \geq 0 \end{cases}$$

$$|arr[i] - arr[j]| + |i-j|$$

(I) $arr[i] > arr[j]$ && $i > j$

$$|arr[i] - arr[j]| + |i-j|$$

$$= arr[i] - arr[j] + i - j$$

$$= arr[i] + i - (arr[j] + j) \quad \text{—(I)}$$

(II) $arr[i] > arr[j]$ && $i < j$

$$|arr[i] - arr[j]| + |i-j|$$

$$= arr[i] - arr[j] - (i - j)$$

$$= arr[i] - i - arr[j] + j$$

$$= (arr[i] - i) - (arr[j] - j) \quad \text{—(II)}$$

(III) $arr[i] < arr[j]$ && $i > j$

$|arr[i] - arr[j]| + |i-j|$

$= -(arr[i] - arr[j]) + (i-j)$

$= (arr[j] - j) - (arr[i] - i)$ — (III)

(IV) $arr[i] < arr[j]$ && $i < j$

$|arr[i] - arr[j]| + |i-j|$

$= -(arr[i] - arr[j]) + (-(i-j))$

$= -arr[i] + arr[j] - i + j$

$= (arr[j] + j) - (arr[i] + i)$ — (IV)

$(arr[i] + i) - (arr[j] + j)$ — (I)

$(arr[i] - i) - (arr[j] - j)$ — (II)

$(arr[j] - j) - (arr[i] - i)$ — (III)

$(arr[j] + j) - (arr[i] + i)$ — (IV)

$$\max\left[\underbrace{(arr[i]+i)}_{\max} - \underbrace{(arr[j]+j)}_{\min} \rightarrow \max,\ (arr[i]-i) - (arr[j]-j) \rightarrow \max\right] \rightarrow \max$$

(i) for any two idxes

find $\left(sum(element, idx)\right) \rightarrow \max$ diff

(ii) for any two idxes

find $\left(diff(elem, idx)\right) \rightarrow \max$ diff

	0	1	2	3	4		
	2	5	4	1	6	maxelem ↓	minelem ↓
arr[i] + i ⇒	2	6	6	4	10	⇒ 10 − 2 = 8	
arr[i] − i ⇒	2	4	2	−2	2	⇒ 4 − (−2) = 6	
						↑ max diff	↑ min diff

8

Q2

maxSum minSum maxDiff minDiff

```
for(i=0; i<N; i++) {
    sum = arr[i] + i
    diff = arr[i] - i
    maxSum = max(maxSum, sum)
    minSum = min(minSum, sum)
    maxDiff = max(maxDiff, diff)
    minDiff = min(minDiff, diff)
}
return max(maxSum - minSum, maxDiff - minDiff)
```

TC = O(N)
SC = O(1)