Arrays -1

Mathy
→ Prime
→ GCD

1) Max absolute Diff
2) 1st Missing +ve Integer
3) Merge Overlapping Interval
4) Rain water Trapping

Question: Max absolute Difference

A = [1, 3, -1] (indices 0, 1, 2) N = 3

Maximum value of

$$\max \left(|arr[i] - arr[j]| + |i - j| \right)$$

Possibly arr[i] = 1, arr[j] = 3

i = 1, j = 2

$|1-3| + |0-1|$ (−2) + |−1|

$2 + 1 = 3$ ✓

$|3-(-1)| + |1-2|$

$4 + 1 = 5$

A = [−1, 1, 3] (indices 2, 0, 1)

$|3-(-1)| + |1-2| = 4 + 1 = 5$

A = 2 3 4 5 6

0 1 2 3 4 5

|6 − 1| + |5 − 4|

= 5 + 1 = 6

= |6 − 2| + |5 − 0| = 4 + 5 = 9

Brute force :

consider all pairs of nos.

$F(i,j) = |arr[i] - arr[j]| + |i-j|$

$F(i,i) = 0$

$F(i,j) = F(j,i)$

$F(i,j) \& F(j,i)$

$(i<j)$

A = 1 3 −1

0 1 2

i	j	
0	0	|1−1| + |0−0| = 0
0	1	|3−1| + |1−0| = 3
0	2	|−1−1| + |2−0| = 4
1	0	|3−1| + |1−0| = 3
1	1	|3−3| + |1−1| = 0
1	2	|−1−3| + |2−1| = 5
2	0	|−1−1| + |2−0| = 4
2	1	|−1−3| + |2−1| = 5
2	2	|−1+1| + |2−2| = 0

Ans = 5

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

S.C. $O(1)$

```
for(i = 0 to n-1) {
    for(j = 0 to n-1) {

    }
```

F(i,j) F(j,i)

$f(i,i) = |arr[i] - arr[i]| + |i-i|$
$0 + 0$

$f(i,j) = |arr[i] - arr[j]| + |i-j|$

$f(j,i) = |arr[j] - arr[i]| + |j-i|$

$(i < j)$

$f(i,j) = |arr[i] - arr[j]| + |i-j|$

```
for(i=0 ; i<n; i++){
    for(j=i+1; j<n; j++)
```

$i = n-1$
$j = n$
$(i < j)$
$i = n-1$

}

T.C : $O(\frac{n^2}{2}) \Rightarrow O(n^2)$
S.C : $O(1)$

$x = -5$ $\quad |x| = 5$ $\quad 5 = |x|$
$x = 5$ $\quad |x| = x$

Observations

$O(n \log n)$

$|x| \Rightarrow x$ if $x \geq 0$
$-x$ if $0 > x$

$x = -2$
$|x| = 2$
$= -(x)$
$= -(-2) = 2$

$i < j$

$f(i,j) = |arr[i] - arr[j]| + |i-j|$

$|arr[i] - arr[j]| + j - i$ $\quad (i < j)$

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

$\max(x[j] - x[i])$ $\quad f(i,j) = \min$

$A = 5\ 4\ 1\ 0\ 6$

$(A[i], A[j])$ $\quad \max \quad \min$

Case 1:

Get rid of this modulus ?

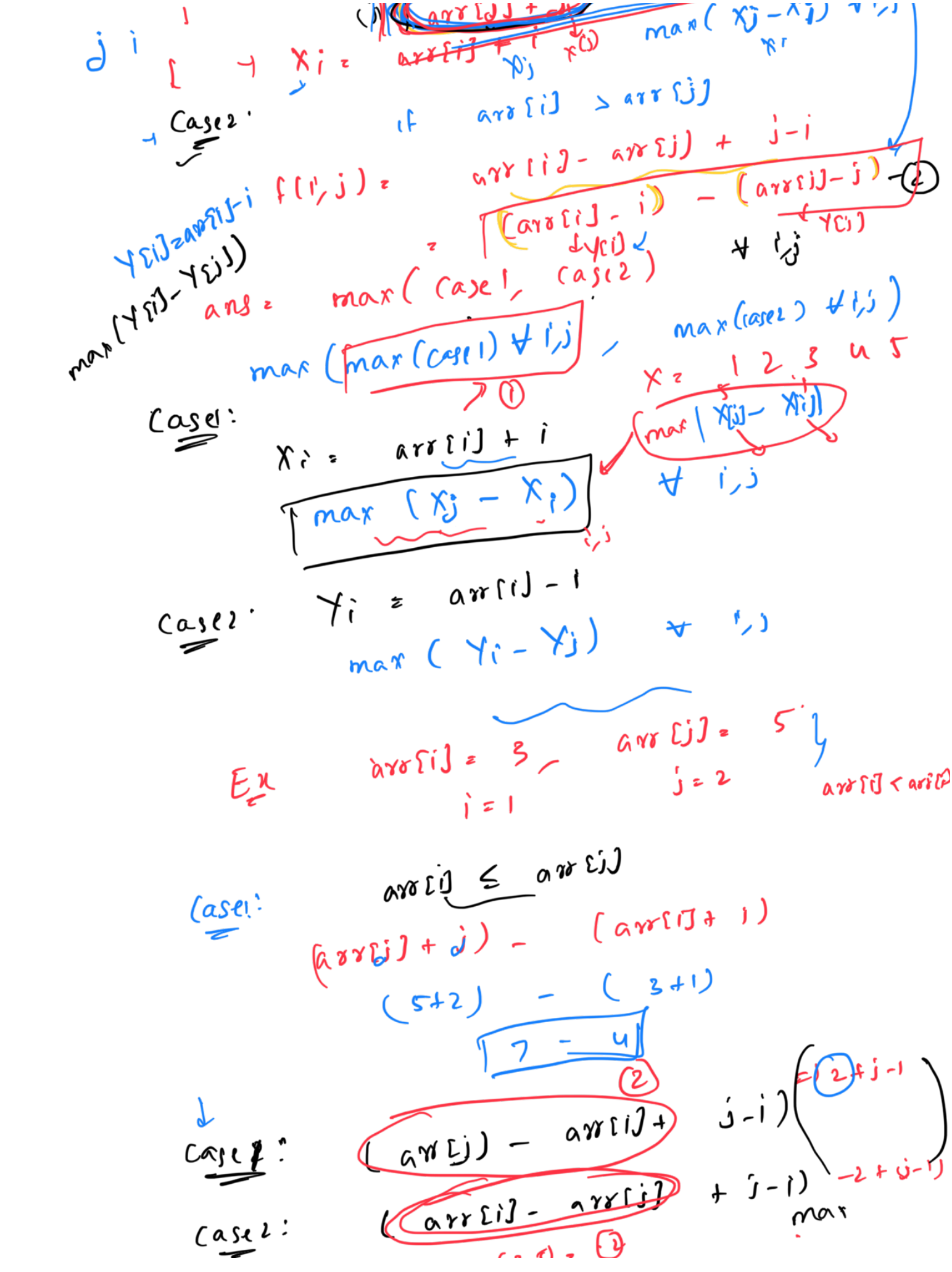
$x[j] = arr[j] + j$
$x[j]$

$arr[i] \geq arr[j]$

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

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

$= x[i] - x[j]$ ①



$(j-i) > 0$

Case1: $2 + (j-i)$

Case2: $-2 + (j-i)$

Case1:

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

maximize

$x_i = arr[i] + i$

Max(case1) $=$ $\max (x_j - x_i) \quad \forall \ (i,j)$

Case1:

$x_j = arr[j] + j$

Simple: Array of numbers

$X[i] = A[i] + i$

$\max (arr[j] - arr[i])$ (max, min)

Case1:

A = 5 2 0 1 6

$x_i = A[i] + i$

	0	1	2
A =	1	3	-1
x =	1	4	1

$\max(x_i) = 4$

$\min(x) = 1$

$4 - 1 = 3$ → Max for Case 1

Max for Case 1 : 3

Case2:

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

$y_i = arr[i] - i$

max(case2) : $Max(y_i - y_j) \quad \forall\ i, j$

	0	1	2
A =	1	3	-1
y :	1	2	-3

max(y) = 2
min(y) = -3
max - min = 2 - (-3) = 5

max (case2) = 5

Ans = (3, 5) = 5

	0	1	2	3
A =	2	3	1	6
x =	2	4	3	9
y =	2	2	-1	3

$|6-2| + |3-0|$
$4 + 3 = 7$

max(x) - min(x)
9 - 2 = 7

max(y) - min(y)
= 3 - (-1)
= 4

max(7, 4) = 7

$x\_max \rightarrow \max(arr[i] + i)$
$x\_min \rightarrow \min(arr[i] + i)$
$y\_max = \max(arr[i] - i)$
$y\_min = \min(arr[i] - i)$

ans = $\max(x\_max - x\_min, y\_max - y\_min)$

(x, y)

$x\_max = -INF, \quad y\_max = -INF$
$x\_min = INF, \quad y\_min = INF$

x_min = INF

```
for (i=0; i<n; i++) {
    x_max = max(x_max, A[i]+i)
    x_min = min(x_min, A[i]+i)
    y_max = max(y_max, A[i]-i)
    y_min = min(y_min, A[i]-i)
}
return max(x_max - x_min, y_max - y_min);
```

4 operations

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

Small steps

Question: First Missing Non-Negative Integer {0}

Given unsorted array with the +ve elements

	0	1	2	3	4	5	6
A =	8	10	1	-3	2	-5	0

Ans = 3

A = 2 3 7 6 8 -1 -10 15

Ans = 0

1) max(array)
2) length(array)

Brute Force:

0, 1, 2, 3

T.C: O(n×n)
O(n²)

→ for (i=0; i<n; i++) {
check if 'i' is there in array
//or

{ { { // check for

Min(ans) = 0 → [0, 1, 2, 3 . . . .]

Max(ans) → N → [0, n-1]

A = [0, 1, 2, 3 . . . . . n-1]

[0 . . . n-1]

N = 5 → 5 → n

A = [0, 1, 2, 3, 4]

N = 6 → > 6 !

A = [0, 1, 0, 3, 4, 5, 6]

T.C: O(n²)

S.C: O(1)

Range: [0, N]

Approach2:

Use a hash set to check if element is present in array.

→ for(i=0; i <= n; i++) { if(i is not there in set) return i } O(1)

T.C: O(n x 1) = O(n)

S.C: O(n)

A = [-100, 0, 1, 10002, . .]

A = [1, 2, 3, 4, 5]

6 ! => [0]

A = [0, 1, 2, 3, 4]

(-INF, INF)

A2 [-99, -76, 0, 1, 1002, 105) (5)

A = [-3, 0, 1, 9] ✗

Without Extra Space

O(n), O(n)

A =

temp = 0

T.C: O(nlogn)

temp = 0 1 2 3

Sort

T.C: O(nlogn + n) = O(nlogn + logn)

⇒ O(nlogn)

O(nlogn)

S.C: O(1)

A = 0, 1, 2, 2, 3

→ skip all duplicate elements when moving index

temp = 0 1 2

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

Efficient Approach:

ans: [0, n]

Let us consider the range [0, n-1] array elements only in

Case 1: If all nos in range are present

answer = n

this range

case 2 : First missing num in [0, n-1]

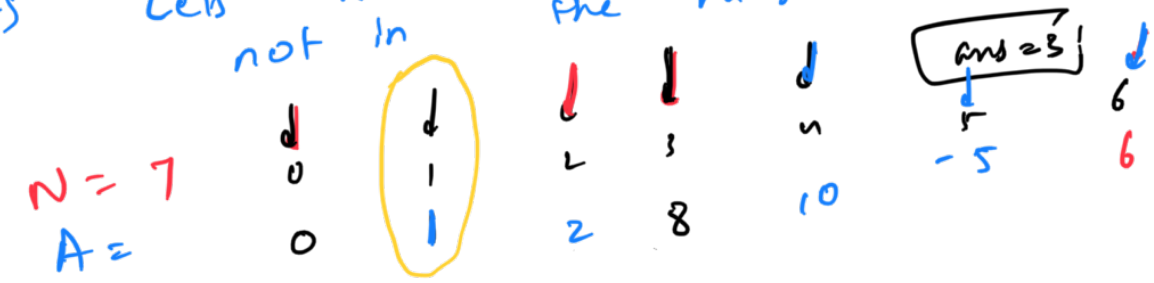
Range: [0, n-1]

0, 1, 2, 3 ...

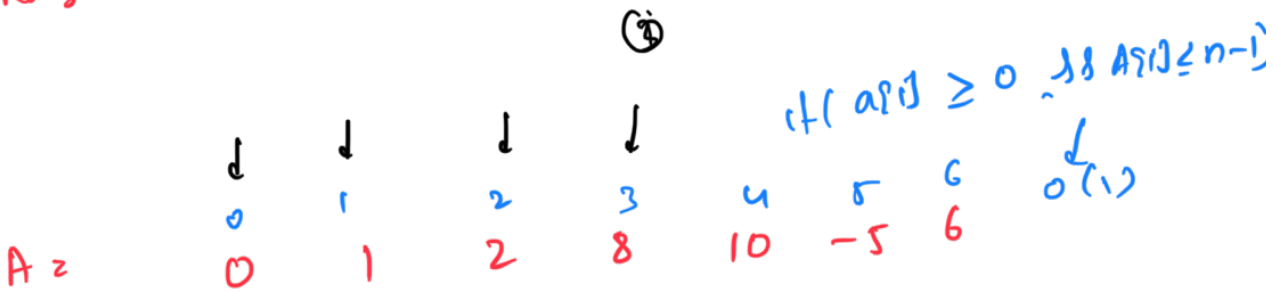
Hashset : O(n) space

Can we do it in O(1) space ?

-> Lets not consider the numbers which are not in the range (0, n-1)



Range: [0, 6]



Edge Case: Duplicate Elements

Range : [0, 3]

N = 4

Infinite loop

-> Swap elements only if they are not equal

T. C : O(n)

break

[0, N-1]

S.C: O(1)

[0, 3]

N = 4

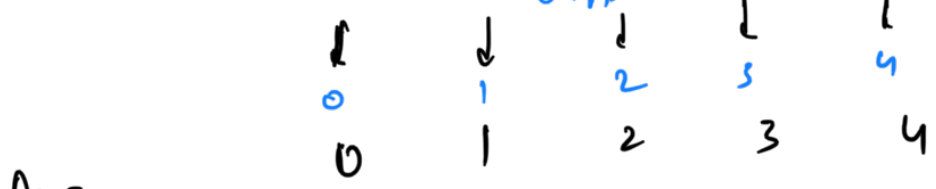
arr[i]

arr[i]

for((if (i is in arr)) i with 1

O(n)

N = 5



A =

5

Range: [0, 4]

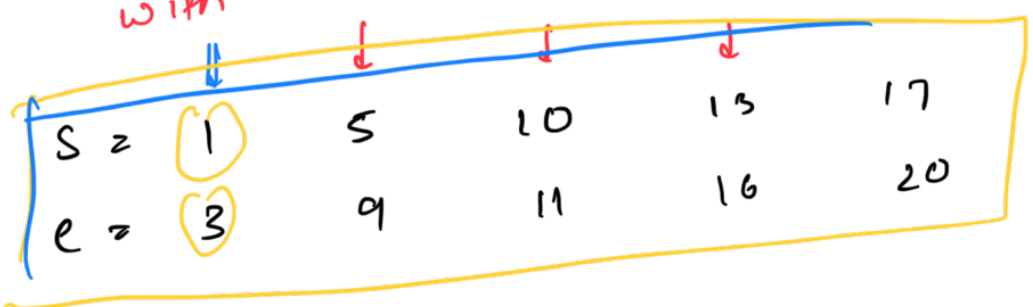
[0, 4] is

Return N.

Question: Merge overlapping intervals

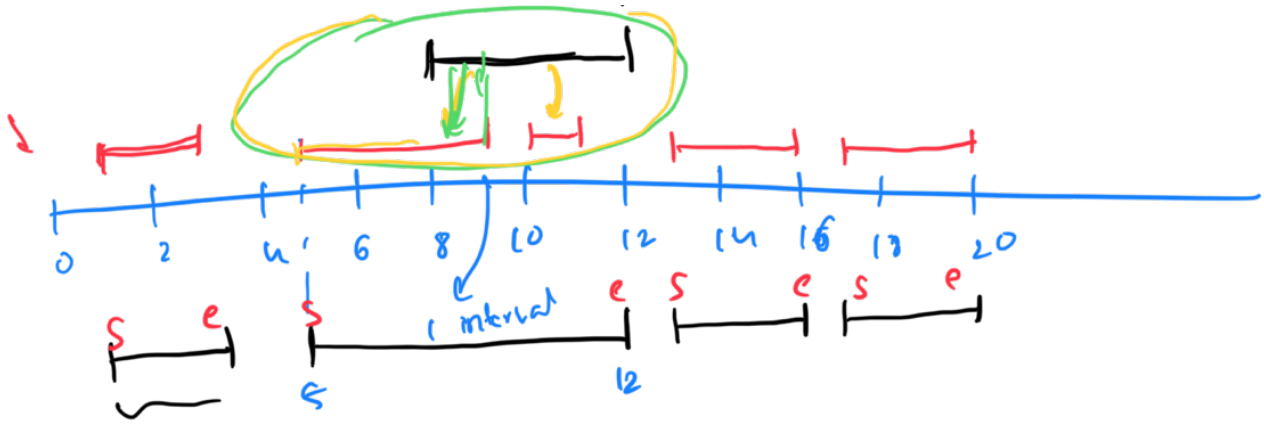
→ Given non-overlapping intervals in sorted order

→ Given a new interval, merge this with existing interval



[8, 12]

8 12

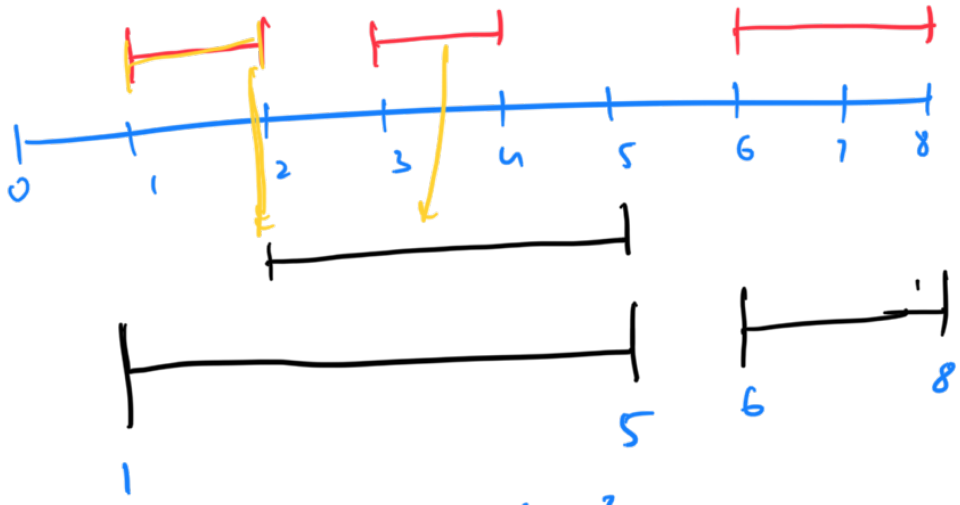


Ex2

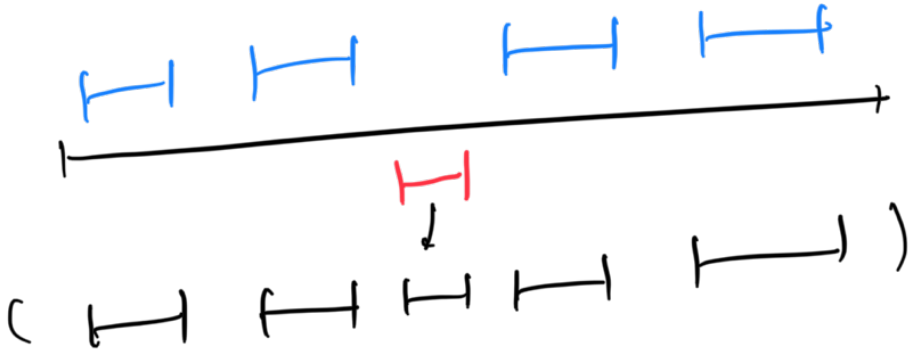
s = 1 3 6

e = 2 4 8

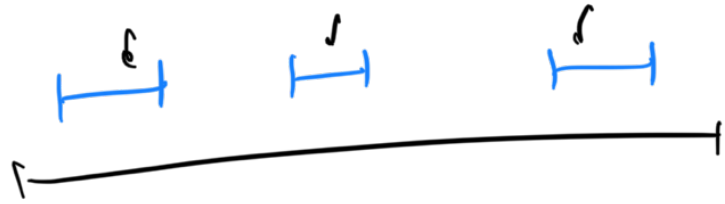
new interval: [2, 5]

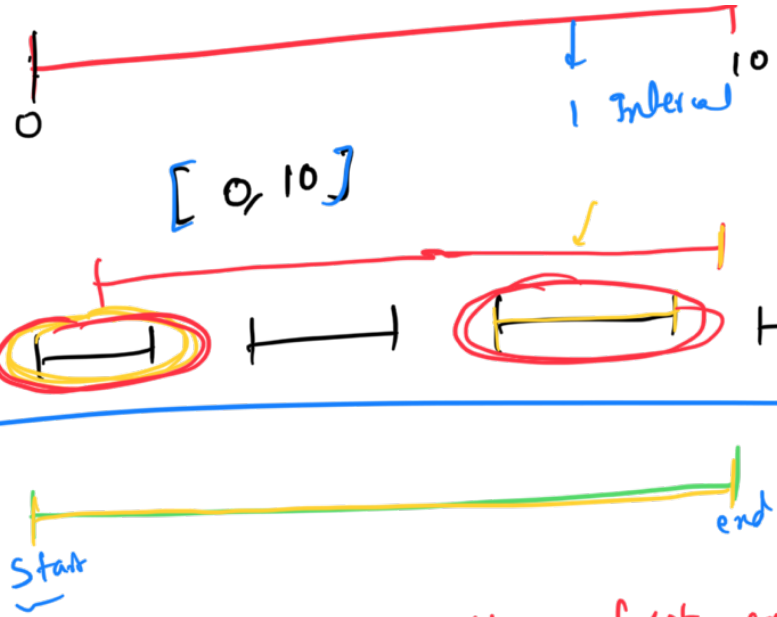


[1, 5], [6, 8]



Ex3





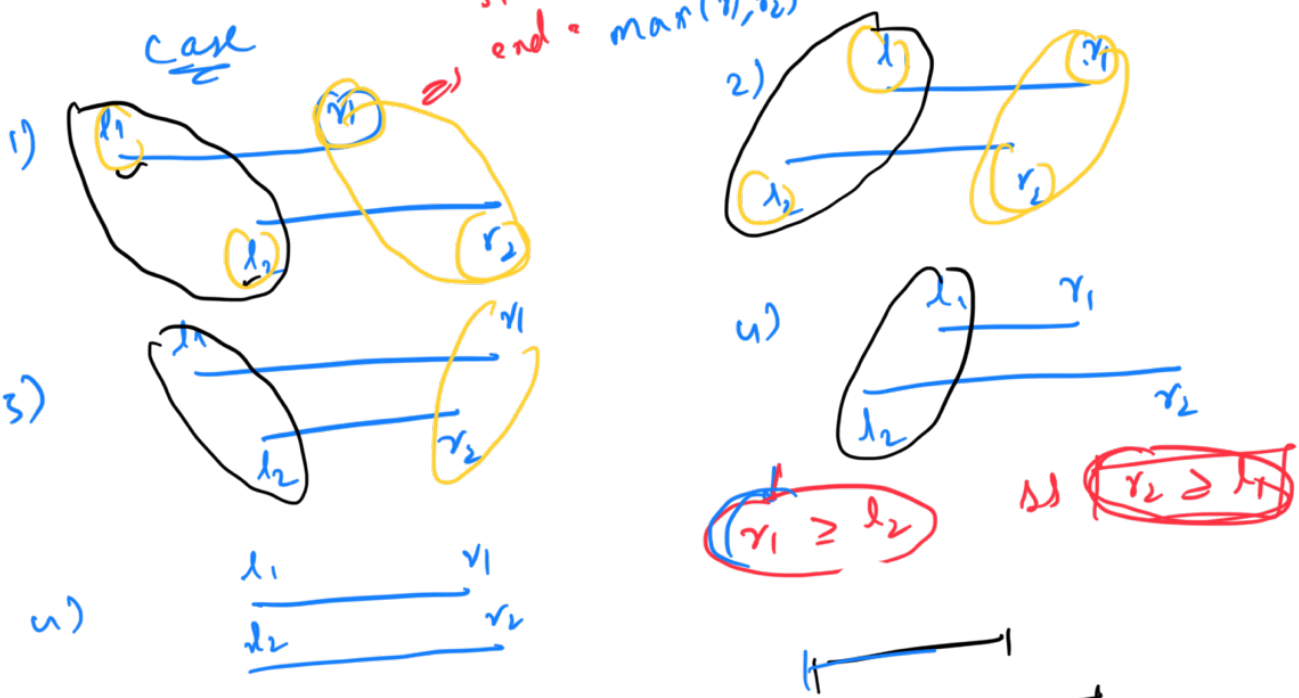
→ Concerned with the first and last overlapping interval

How do we know if 2 intervals are overlapping: (l_1, r_1) (l_2, r_2)

start = $\min(l_1, l_2)$

end = $\max(r_1, r_2)$

Case



→ One interval has to start before other ends

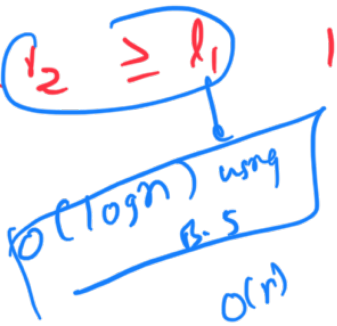
(l_1, l_2)

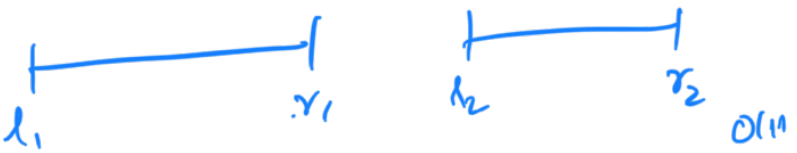
(r_1, r_2)

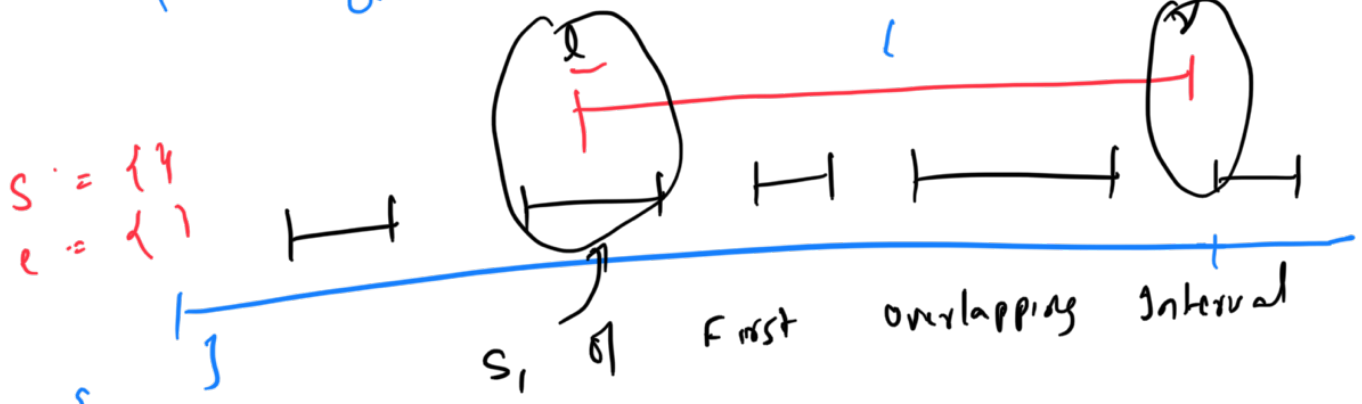
$l_1 \le r_1, r_2 \Rightarrow l_1 \le r_2$ && $l_2 \le r_1, r_2 \Rightarrow l_2 \le r_1$

1) $(r_2 \ge l_1$ && $r_1 \ge l_2) \Rightarrow$ False

2) $(l_2 \ge l_1$ || $r_1 \ge l_2) \Rightarrow$ True







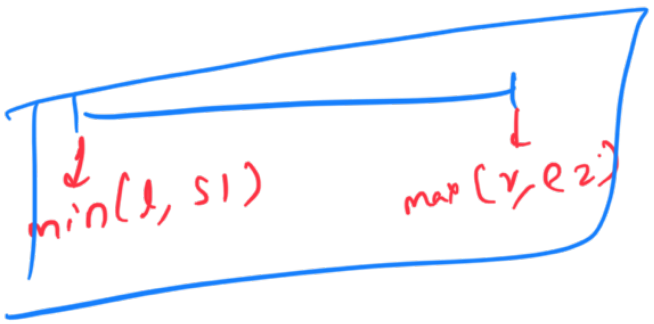
inp = [ ]
out = [ ]

```
for(i=0; i<n; i++) {
    if( end[i] ≥ l && r ≥ start[i] ){
        break;
    }
}
```

ith interval is the 1st overlapping

```
for(i=n-1; i≥0; i--) {
```

New Merged Interval!



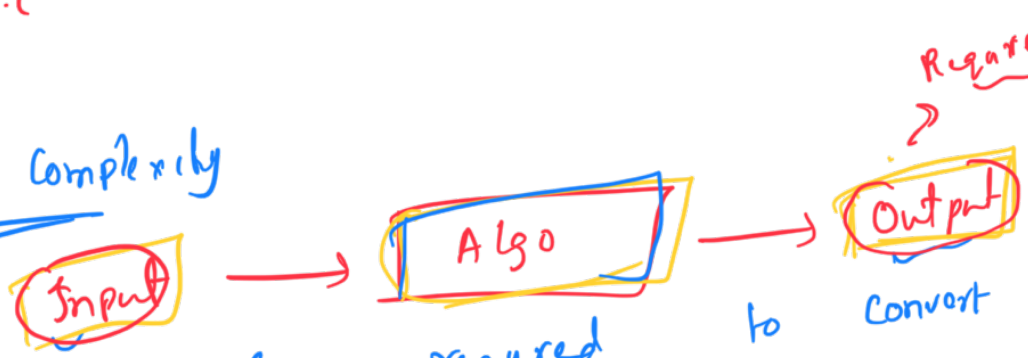
l, r

s1, e2

T.C:

S.C:

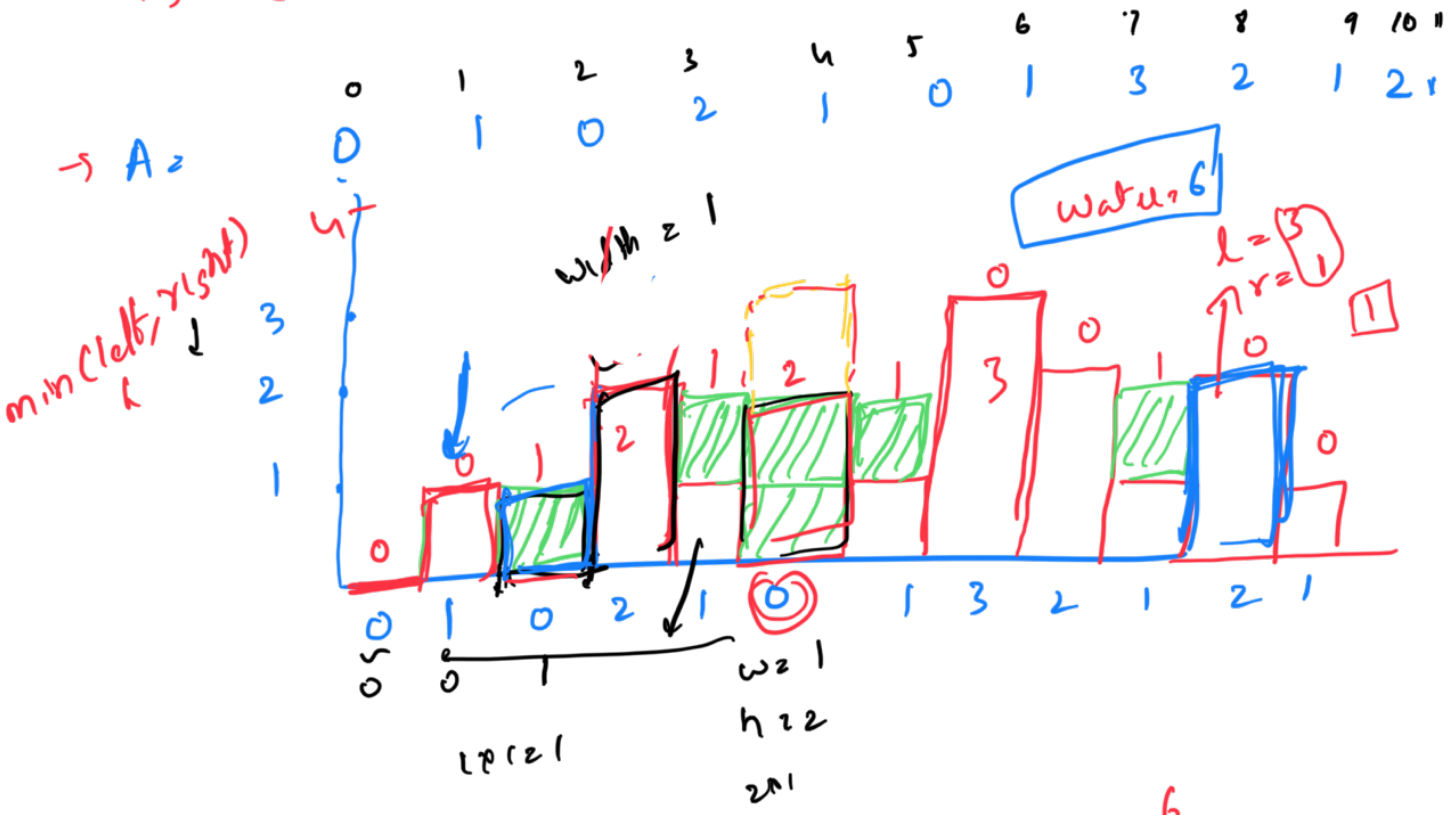
Space Complexity



Question: Trapping Rain Water

Given an array of n elements each representing height of building

→ Compute the water trapped after raining



0 + 0 + 1 + 0 + 1 + 2 + 1 + 1 = 6

Water stored above ith building

$w[i] =$

$ans = \sum w[i]$

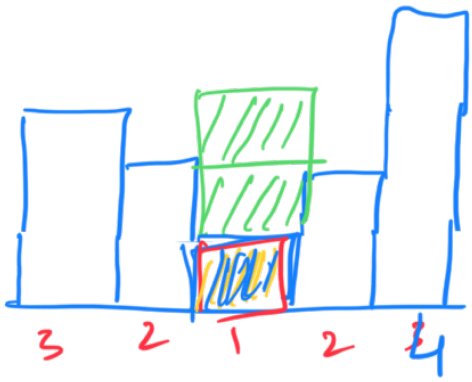
→ (Compute contribution of each element of the array and adding them up gives the answer) → Technique.

Brute force:

```
for(i=0; i<n; i++) {
    l = tallest building to left
```

}

Any building 'i': $\min(l, r)$




```
for(j=i; j<n; j++)
    r = max(r, a[j]);
```

```
for(j=i; j>=0; j--){
    L = max(L, a[j])
}
```

$l = 3$

$r = 4$

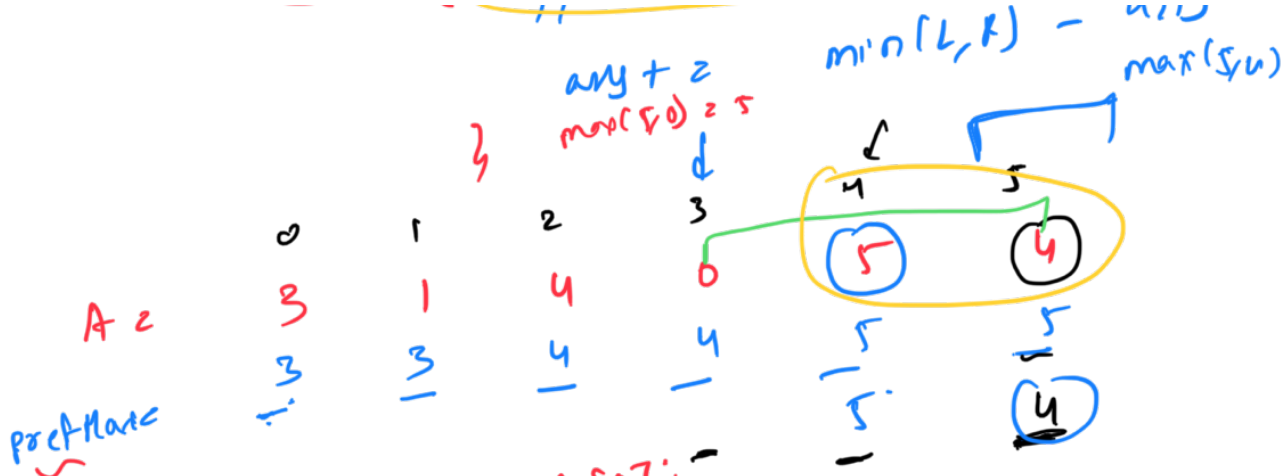
$\min(4, 3) = 3$

$\min(l, r) - a[i]$

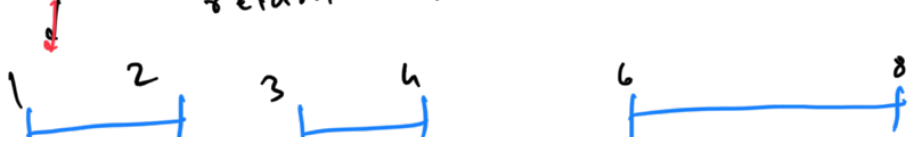
T.C: $O(n^2)$

```
for(i=0; i<n; i++) {
```

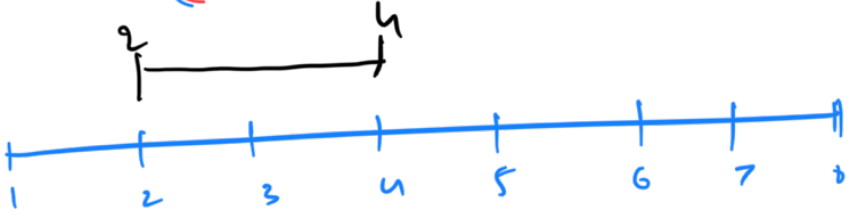
O(1) { // O(n) for finding L
// O(n) for " R
a[i]




```
prefMax[0] = A[0];
for(i = 1; i < n; i++){
    prefMax[i] = max(prefMax[i-1], A[i])
}
suffix Max

T.C: O(n)

S.C: O(n)
    { prefMax & suffMax

suffMax[n-1] = arr[n-1];
for(j = n-2; j >= 0; j--)
    suffMax[j] = max(suffMax[j+1], A[j])

ans = 0;
for(i = 0; i < n; i++){
    L = prefMax[i];
    R = suffMax[i];
    ans = ans + min(L,R) - a[i]
}
return ans;
```

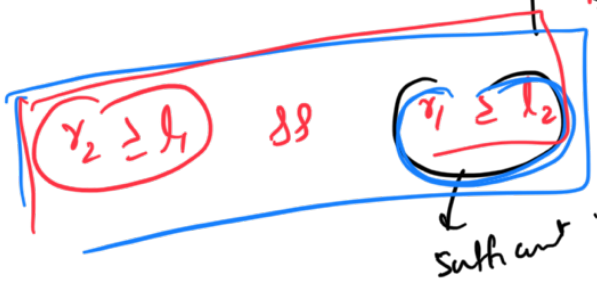


(start of array, interval, end of new interval)



(2, 3)

sz min(start)
& max

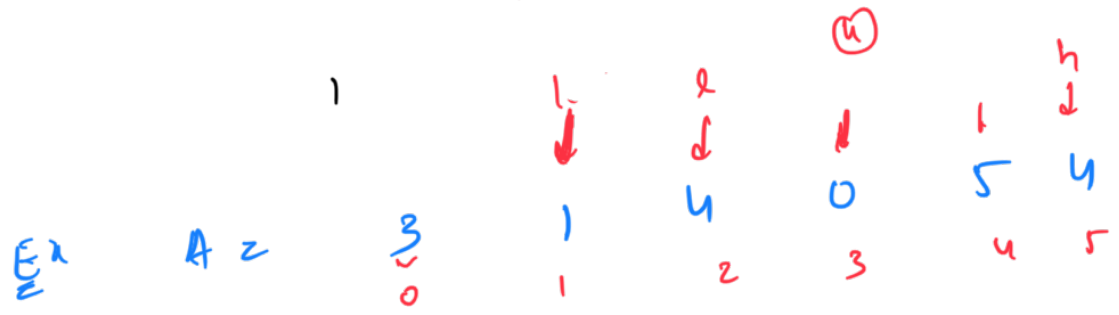


$r_1 \geq l_2$

(S[i], e[i])

(L, R)


```
for (i=0; i<n; i++) {
    if( e[i] ≥ L && R ≥ s[i] ) {
        r2   l1     r1   l2
        first_overlap = i;
        break;
    }
}

for(i=n-1; i≥0; i--) {
```



while (low ≤ high) {

low = 0

high = n-1

S.C: O(1)

Max Absolute Difference

```
X_max = INT_MIN, X_min = INT_MAX;
Y_max = INT_MIN, Y_min = INT_MAX;

for(i = 0; i < n; i++){
      X_max = max(X_max, A[i] + i);
      X_min = min(X_min, A[i] + i);
      Y_max = max(Y_max, A[i] - i);
      Y_min = min(Y_min, A[i] - i);
}
ans = max(X_max - X_min, Y_max, Y_min)
```

First Missing Positive Integer

```
i = 0;
while(i < n){
      if(arr[i] >= 0 && arr[i] <= N-1){
            if(arr[i] != arr[arr[i]){
                  temp = arr[i];
                  arr[i] = arr[arr[i]];
                  arr[arr[i]] = temp;
            }
      }
      if(arr[i] not in range [0, N-1])
            i++;
      // else we'll again swap this number to its index
}
// Return the first instance where arr[i] != i
for(int i = 0; i < n; i++)
      if(arr[i] != i)
            return i;
// Return n if we have all numbers in [0, n-1]
return n;
```

Rainwater trapping

```
// Prefix Max
int L[];
L[0] = A[0]
for(int i = 1; i < n; i++){
      L[i] = max(L[i-1], A[i])
}

// Suffix Max
int R[];
R[n-1] = A[n-1]
for(int i = n-2; i >=0 ; i--){
      R[i] = max(R[i+1], A[i])
}

ans = 0;
for(int i = 0; i < n; i++){
      ans += min(L[i], R[i]) - A[i];
}
```

Merge Intervals

```
for(int i = 0; i < n; i++){
      if(r >= s[i] && e[i] >= l){
            // First interval
            first_start = s[i];
            break;
      }
}
for(i = n-1; i >= 0; i--){
      if(r >= s[i] && e[i] >= l){
            // Last interval
            last_end = e[i];
      }
}
newMergedInterval = [ min(l, first_start)     max(r, last_end)  ];
```