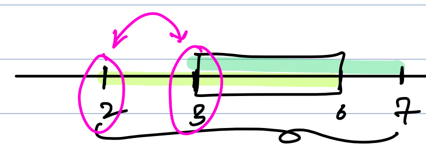
# Merge Intervals

range




| $I_1$ | $I_2$ |
|---|---|
| (2, 6) | (3, 7) |

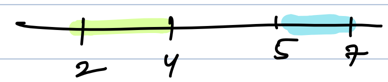
overlapping ✓

(2, 7) Merge

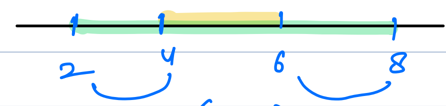
min(2, 3) max(6, 7)

(2, 4) (5, 7)

non-overl

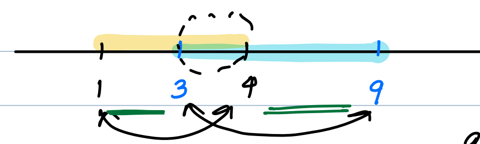



(2, 8) (4, 6)



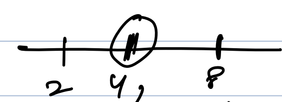

(2, 8)

(3, 9) (1, 4)




$\boxed{1, 9}$

question




non-overlapping

N no of intervals → sorted form

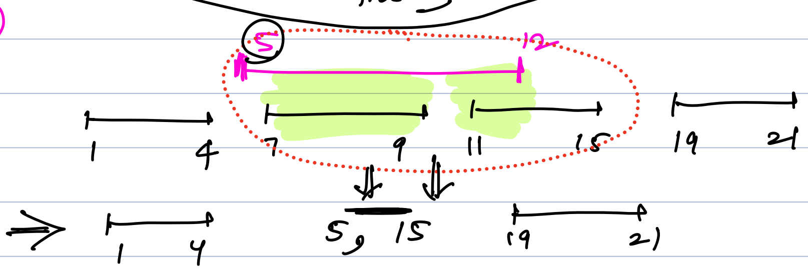
non-overlapping

One extra interval

insert

(1, 4)
(7, 9)
(11, 15)
(19, 21)

(5, 12)



$I_2$ $\quad$ $I_1$ $\quad$ $I_2$

$s_2$ $\quad$ $E_2$ $\quad$ $s_1$ $\quad$ $E_1$ $\quad$ $s_2$ $\quad$ $E_2$

Non-overlapping

$\boxed{S_2 => E_1}$

$E_2 <= S_1$

| | | new list |
|---|---|---|
| [1, 3] | [12, 22] | [1, 3] |
| [4, 7] | [12, 22] | [4, 7] |
| [10, 14] + [12, 22] ⇒ [10, 22] | | |
| [16, 19] [10, 22] → [10, 22] | | |
| [21, 24] + [10, 22] → [10, 24] | | [10, 24] |
| [27, 30] → [10, 24] | | [27, 30] |
| [32, 35] | | [32, 35] |

Intervals[n][2] { Intervals[i][0] → start
Intervals[i][1] → end

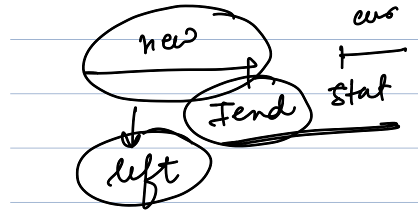
array, vector, ArrayList

[Istart, Iend] $\quad$ list< > ans;

```
for( i=0; i<n; i++)
{
  if( Intervals[i][1] <= Istart)
      ans.insert(Intervals[i][0], Int[i][1]).
```

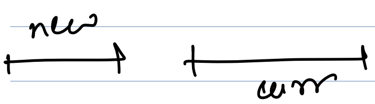
new

Is

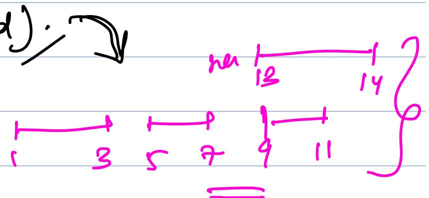



27 30 10, 24

```
else if (intervals[i][0] >= Iend)
{
    ans.insert(Istart, Iend);
    while (i<n)
    {   ans.insert(interval[i]);
        i++;
    }
    return ans;
}
```




```
else
{
    Istart = min(intervals[i][0], Istart);
    Iend = max(interval[i][1], Iend);
}
}
ans.insert(Istart, Iend);
return ans;
```




{Merge overlapping intervals}

Q Find first missing positive no / natural no.

Given an array of N integers. natural no starts from 1

{ 3, -2, 1, 2, 7 } ans = 4

{ 1, 2, 5, 6, 4, 3 } ans = 7

{ 1, 0, -5, -6, 4, 2 } ans = 3

N+1 ← array size N, 1 → N

1 → N, ans = N+1;

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

```
for (i = 1; i <= N; i++)
{
    flag = false;
    for (j = 0; j < N; j++)
    {
        if (arr[j] == i)
            flag = true;
    }
    if (flag == false)
        return i;
}
```

Hashmap/Hashset → if an element is present — O(1)

T.C: O(N)
S.C: O(N)

③ sorting approach

{ 1, 0, -5, -6, 4, 2 }

sort → { -6, -5, 0, 1, 2, 4 }

i = ~~1~~ ~~2~~ 3

T.C: $O(n \log_2 n)$ → sorting algo

S.C:

{ 1, 0, -5, -6, 4, 2 } = 6

$1 \rightarrow N$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| F | T | T | F | T | F | F |

T.C: O(N)
S.C: O(N)

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

$1 \rightarrow N$

④

1–6

size = 6

indexes → 0 — N−1

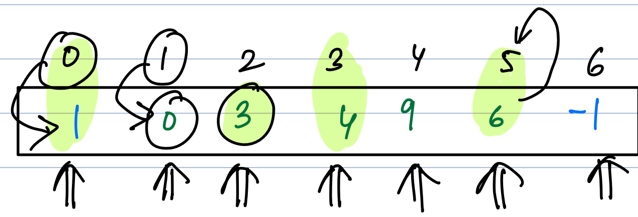
③

0 1
1 2
2 3
3
4
5

$arr[i] \rightarrow 1 \leq arr[i] \leq N$

⇓

$arr[i]-1$ index




2

i+1 == arr[i]

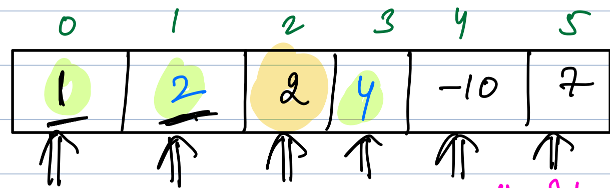
1 → 7

arr[i]    arr[i]-1

3 → 2

6 → 5

arr[i]    arr[i]-1 index




1 — 6

\# if a particular index is already occupied by correct | Don't swap

```
int i=0;
while( i<n)
{
    // check if you can order
    if( arr[i] >=1 && arr[i] <=n)
    {
        int corr_idx = arr[i]-1;
        if( arr[corr_idx] != arr[i])
        {
            swap( arr[corr_idx], arr[i])
        }
        else
            i++;
    }
    else i++;
}
for( i=0; i<n; i++)( if( i+1 != arr[i]) return i+1; }
return n+1;
```

10:45

T.C: O(N)

S.C: O(1)

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 4 | -10 | 7 |

i=0  i++

Takeaway

1 → n

indexes

0  n(n+1)/2

Q  matrix → only consists 0's & 1's

Every row is sorted. Find max no of 1's of in any row.

Takeaway
↓
matrix / 1D arr
↓
sorted

Binary search

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 |

$O(N^2)$

ans=4

i
↓
1 . . . . 1

T.C: $O(n+m)$

1
0

# sorting makes searching easier

Q Maximum absolute Difference → { Google, Microsoft, Amazon }

abs

array, max |arr[i] − arr[j]|

max min

|arr[i] − arr[j]| O(N)

max min

• max( |arr[i] − arr[j]| + |i − j| )

indexes

n−1 0

sort ? ✗

B.F: consider every possible pair $O(N^2)$

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 2 | 5 | 4 | 1 | 6 |

$|x|$

|arr[i] − arr[j]| + |i − j|

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 4 | 4 | 4 | 8 |
| 1 | | 0 | 2 | 6 | 4 |
| 2 | | | 0 | 4 | 4 |
| 3 | | | | 0 | 6 |
| 4 | | | | | 0 |

$|-1| = 1$

$|-7| = 7$

$|7| = 7$

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

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

$$f(i, j) = |\underline{arr[i] - arr[j]}| + |i - j|$$

i, j
3, 4
4, 3

**Case I** $\quad arr[i] \geq arr[j]$ & $i > j$

$$arr[i] - arr[j] + i - j$$
$$= (arr[i] + i) - (arr[j] + j) \quad (1)$$

| i | j |
|---|---|
| 4 | 3 |
| 3 | 4 |

**Case II** $\quad arr[i] < arr[j]$ & $i > j$

$$arr[j] - arr[i] + i - j$$
$$= (arr[j] - j) - (arr[i] - i) \quad (2)$$

**Case III** $\quad arr[i] \geq arr[j]$ & $i < j$ $\qquad |i - j|$

$$arr[i] - arr[j] + j - i$$
$$= (arr[i] - i) - (arr[j] - j) \quad (3)$$

Case II $\quad arr[i] \leq arr[j]$ for $i < j$

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

$i > j$

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

mx1, mx2, ans
min1, min2

```
for( int i=0; i<n ; i++)
{
    mx1 = max(mx1, arr[i]+i);
    mx2 = max(mx2, arr[i]-i);

    min1 = min( mn1, arr[i]+i);
    mn2 = min (mn2, arr[i]-i);

}
    ans = max( mx1-mn1, mx2-mn2);
```