Sliding Window Max

Sunday, 3 October 2021 2:47 PM

total = n , length

ans length = $\underline{n-k+1}$ →

0	1	2	3	4	5	6	7

nums = [1, 3, -1, -3, 5, 3, 6, 7], k = 3

3 3 5 5 6 7

k=3

① Brute force

Check in Every possible window with a Loop of size k

→ kn

implementation based → O(n)   i+k < ngr(j)   ngr(j) < i+k-1

0	1	2	3	4	5	6	7	8
9	7	2	4	6	$\boxed{8}$	2	1	5

i=1

j=1

find → next Right greater Elements index ngr ↓

9*	5	3	4	5	9*	8	8	9*

9 7 6 8 8 8 5

→ loop break

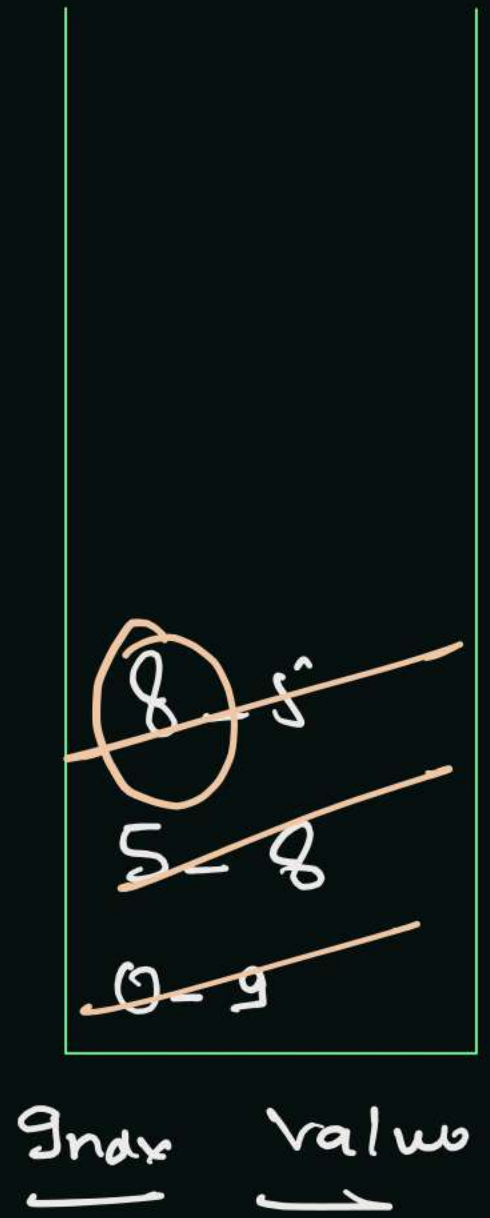
next greater in Right

length

ⓘ → starting index of window

ⓙ → try to find max in that max

	0	1	2	3	4	5	6	7	8
	9	7	2	4	6	8	2	1	5
ng→ r	9^{*}	8_5	4_3	6_4	8_5	9^{α}	5_8	5_8	9^{*}



```java
private int[] ngri(int[] arr) {
    // ngri -> next greater on right (index)
    int n = arr.length;
    int[] ngr = new int[n];
    Stack<Integer> st = new Stack<>(); // add index in stack
    st.push(0);
    for(int i = 1; i < n; i++) {
        while(st.size() > 0 && arr[i] > arr[st.peek()]) {
            ngr[st.pop()] = i;
        }
        st.push(i);
    }
    while(st.size() > 0) {
        ngr[st.pop()] = n;
    }
    return ngr;
}
```

Meeting Rooms 1

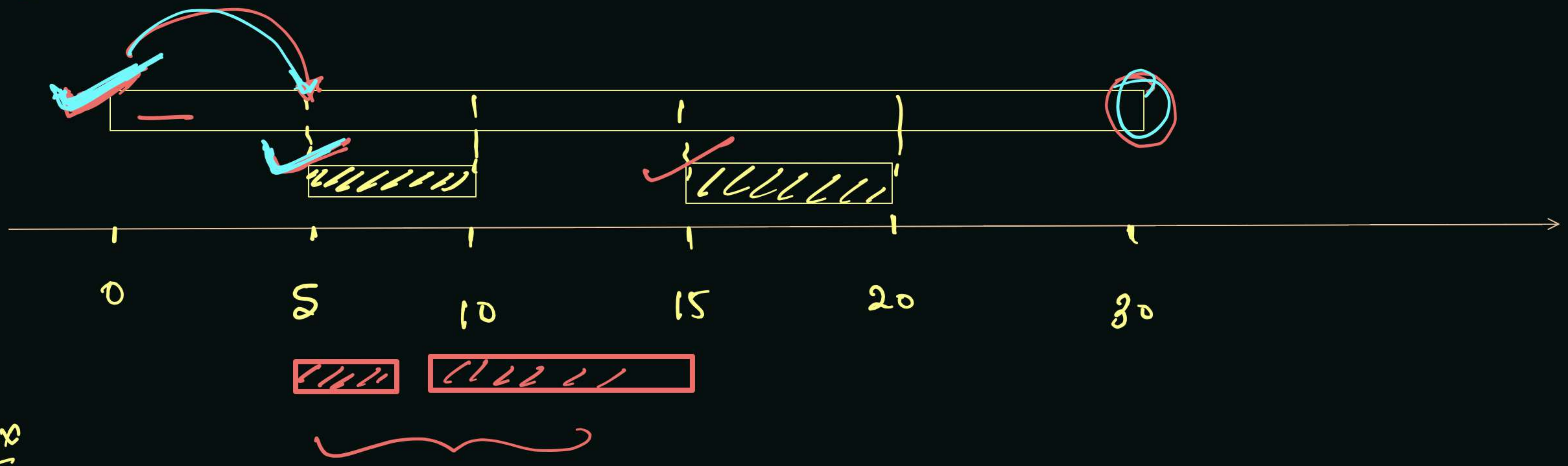
Sunday, 3 October 2021 3:40 PM

Is it possible to attend all the meeting?

```
Input: intervals = [(0,30),(5,10),(15,20)]
```

Result False.

time frame.



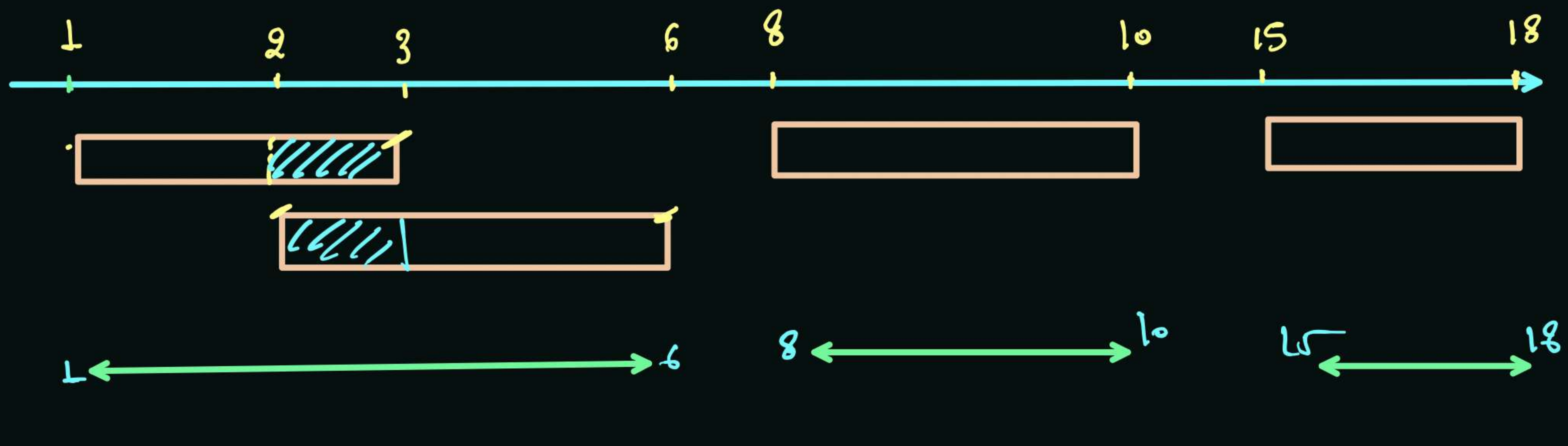
ex

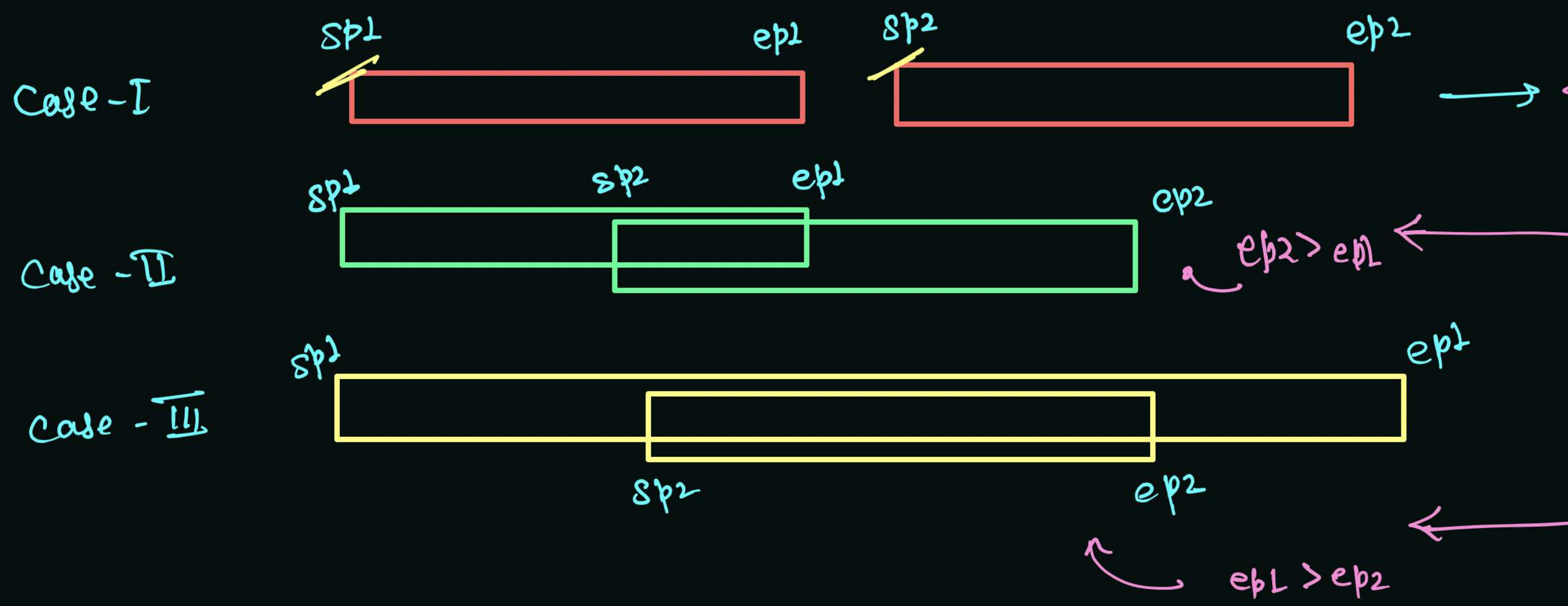
[5,8) (9,15) ← True.

Merge Intervals

Sunday, 3 October 2021 4:25 PM

[1,3] , [2,6] , [8,10] , [15,18] → intervals are sorted on the basis of starting time

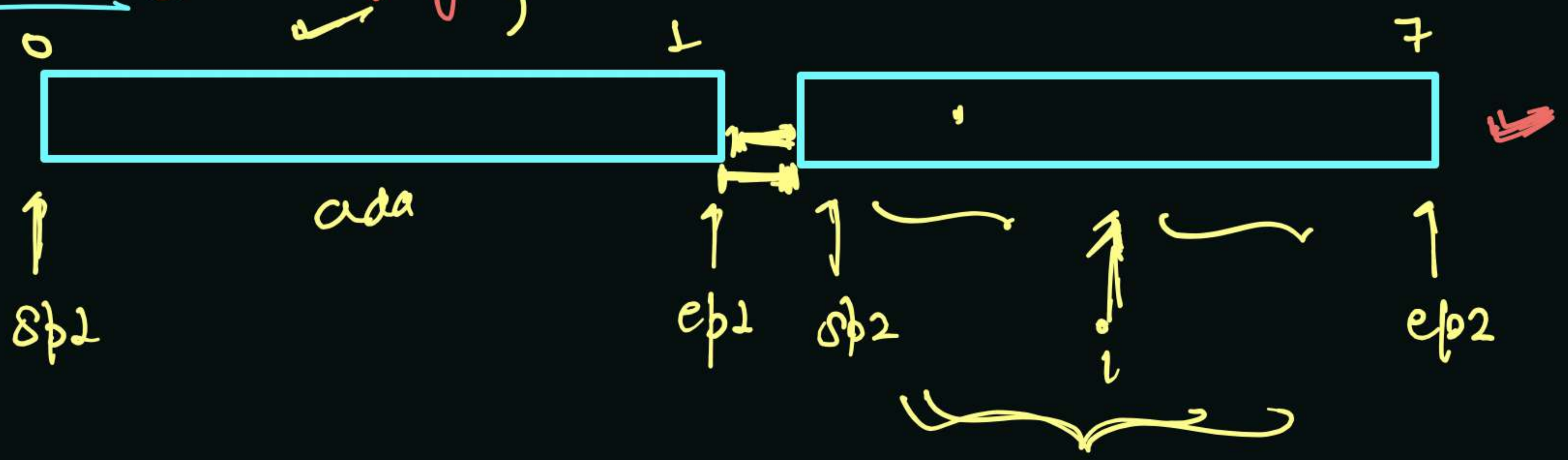




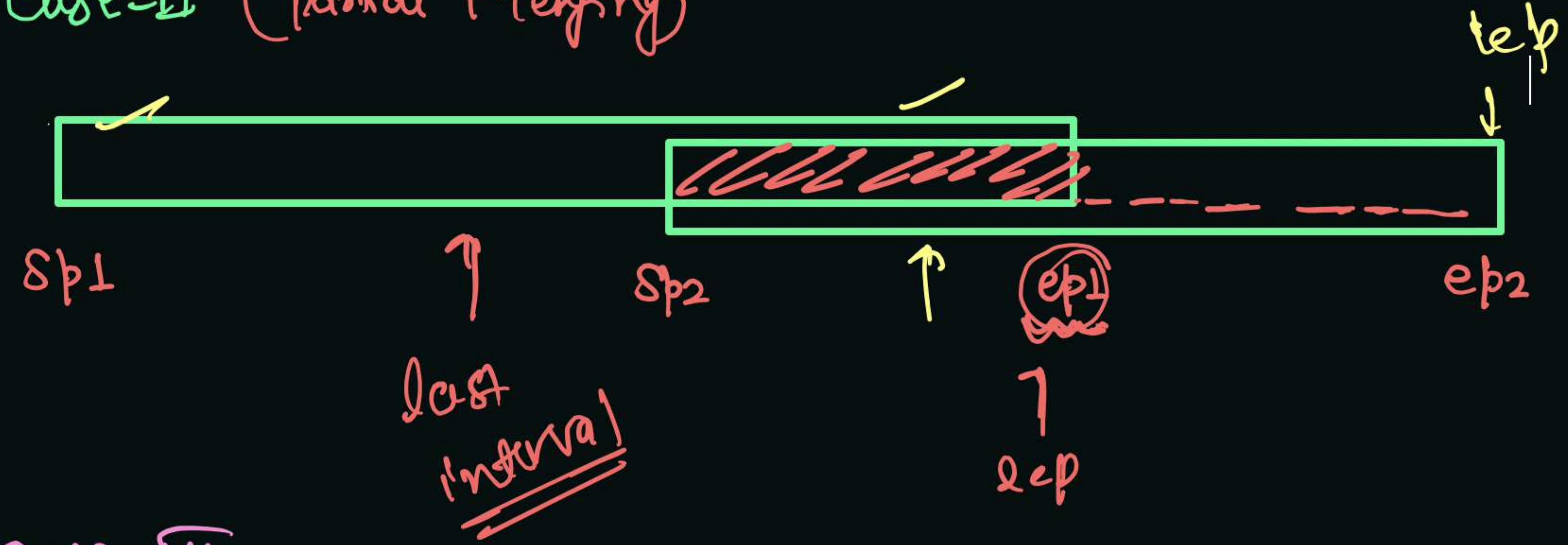
← sp2 > ep1 only case I is possible.

← sp2 < ep1

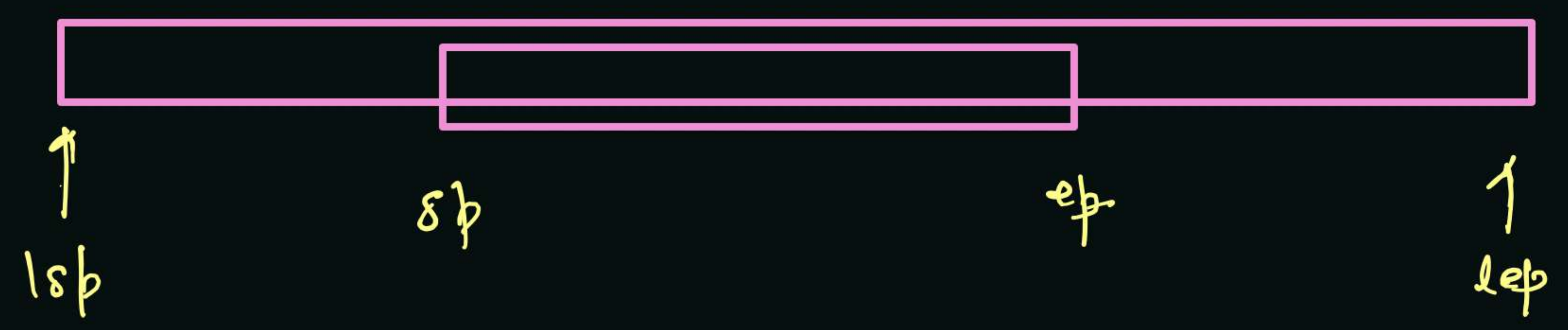
case-I (No merging point)



case-II (partial Merging)



Case-III



lsp = 0
lep = 1

loop

```
if( sp2 > lep) {
    int[] narr = { lsp, lep };
    res.add(narr);
    lsp = sp2;
    lep = ep2;
} else if( lep < ep2) {
    lep = ep;
} else { // fully merging
    already covered
}
```

lsp. lep.

Interval List Intersection

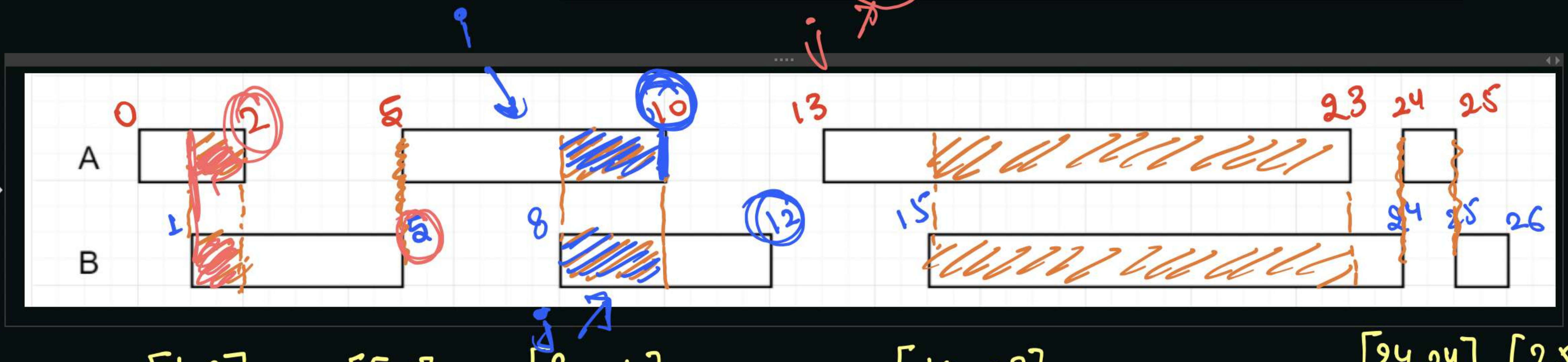
Sunday, 3 October 2021 5:35 PM

firstList = [[0,2] , [5,10] , [13,23] , [24,25]],
secondList = [[1,5] , [8,12] , [15,24] , [25,26]]

$a < b \rightarrow$ swap

a.compareTo(b) $\rightarrow$ swap a,b

$-1 \rightarrow$ a is equal



[1,2] [5,5] [8 , 10] [15-23] [24,24] [25,25]

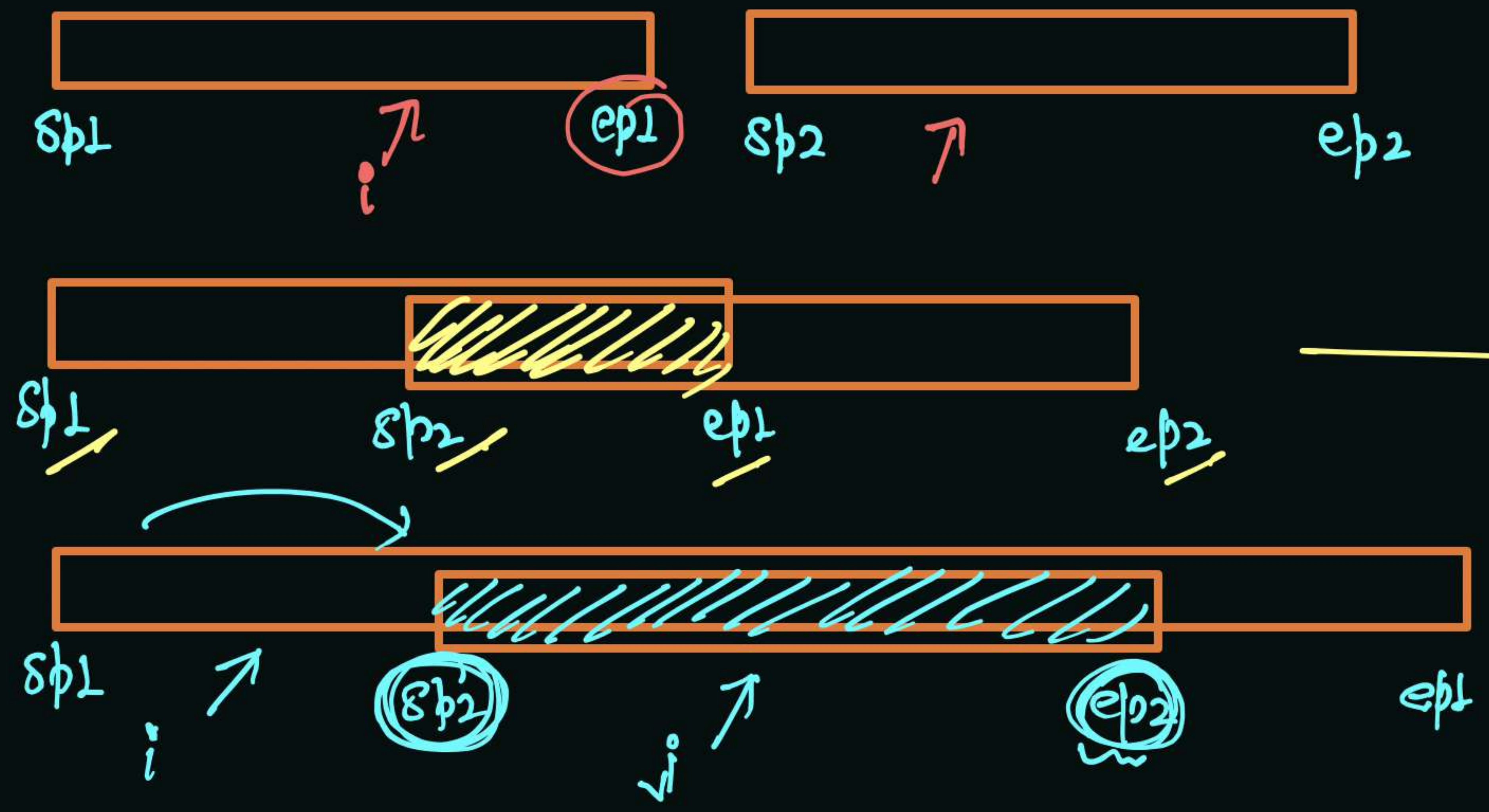
$sp = \max(sp1, sp2)$; $\equiv sp2$

$ep = \min(ep1, ep2)$; $\equiv ep1$

intersection of interval happened

if(st <= end){

}



$sp = \max(sp1, sp2) \equiv sp2$

$ep = \min(ep1, ep2) = ep1$

for intersection $\rightarrow$

$sp = \max(sp1, sp2)$; $\equiv sp2$

$ep = \min(ep1, ep2)$; $\equiv ep2$