K-Catenation

Sunday, 12 September 2021 9:32 PM

Kadane's Algorithm.

A

arr = {1, 2} , K = 3

(k times)

find max. sum subarray with k- concatinated array.

Virtual → B = {1, 2, 1, 2, 1, 2} → Max. sum subarray??

(k times arr)

k - concatenated array.] ✗

K- catenated - virtual

→ cyclic loop

Constraints

(1) A size of array = 10^5] length

(2) range of K = 10^5.

⇒ we can't make an array with concatenation.

(3) Time limit = 1 sec.

⇒ In 1 sec. we can do 10^9 operation.

→ ✗ cyclic loop of k- catcatenated array will not work.

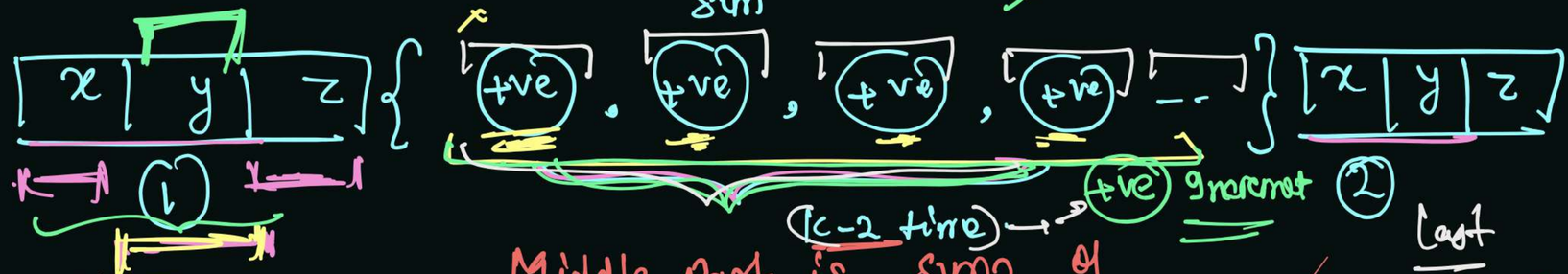
Find Total sum of array ?? significance ?

Tsum → sum of array → (single array) given A

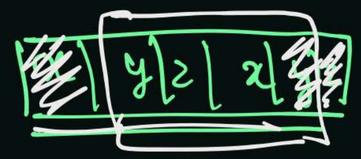
Tsum → sum ≥ +ve

Case-I if(Tsum >= 0)

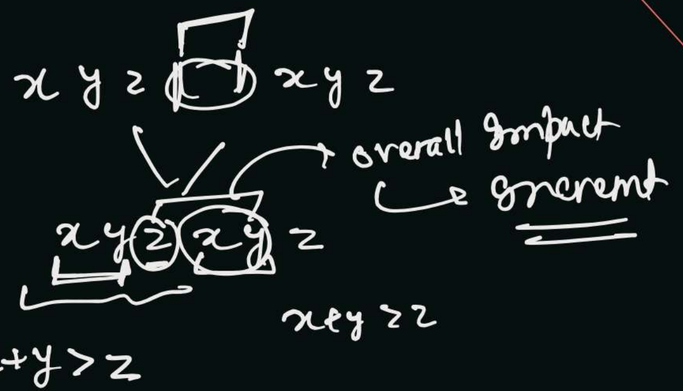
Tsum = +ve → include in Subarray - positive increment



Middle part is sum of +ve, so always contribute in sum of subarray



overlapping of array in finding maximal kadane will encounter



concatenate array

x	y	z	x	y	z

a)-

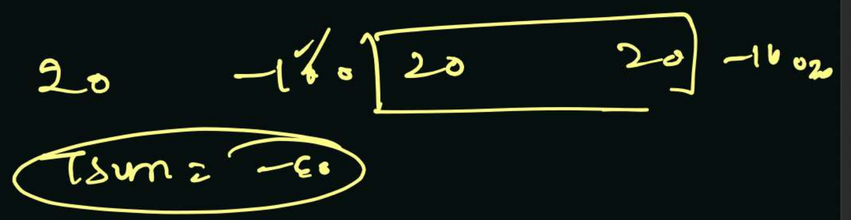
b)-

c)-

d)-

Case-II if(Tsum < 0)

Kadane's →

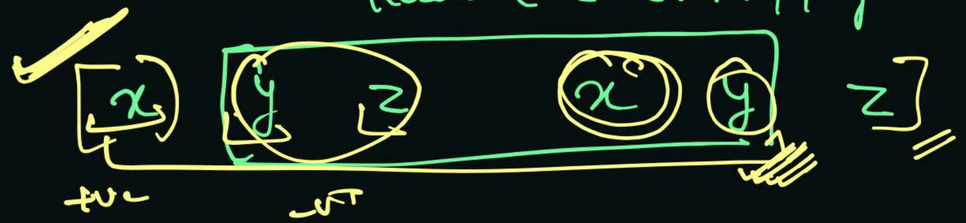


Why overlapping of array in mutual Kadane's will encounter →

array → x , y , z.

$x + y + z > 0$

Kadane - overlapping -



Kadane is in single array

x, [y, z]

$x > (y + z)$ $\quad y + z > x$

$-x$

$y, z \rightarrow$ +ve.

x

$x + y > z$

middle

$x < y + z$ $\qquad y + z > x$

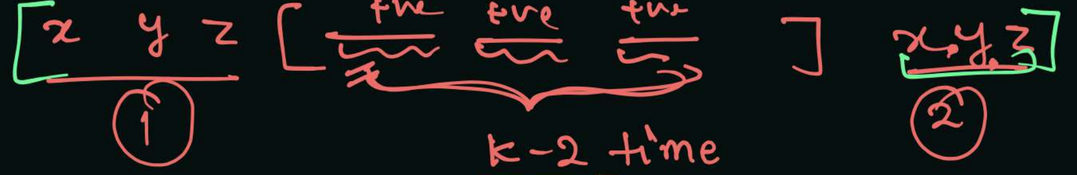
$y + z$ + (+ve)

$x + y + z \Rightarrow$ +ve

So, If total sum is +ve or 0

Sum = Tsum

$$\text{Max Sum} = (\text{sum} * k-2) + \text{kadanes}(1+2)$$

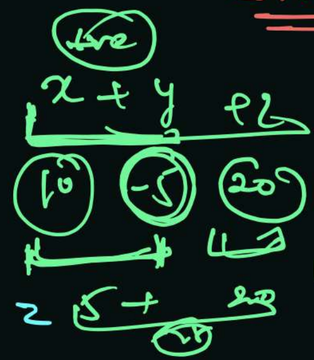


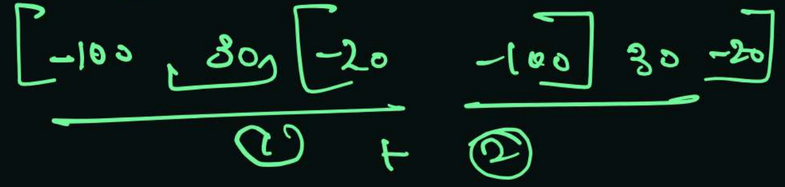
concatenate

case-II

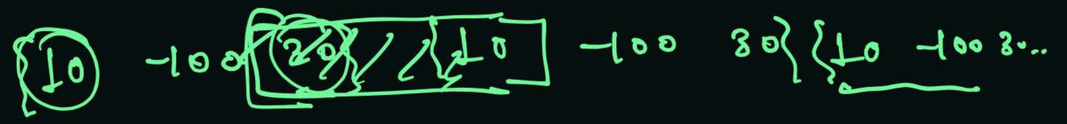
Total sum < 0

x+z > y





kadane's of (1)+(2) max possible sum. K time cont



(1)+(2) Kadanes.

Example. →

$A = [10 \quad -100 \quad 30]$

$$\text{max sum} = \text{Kadanes}(1+2)$$

Maximum Sum of Smallest and Second Smallest in all possible subarray

Saturday, 18 September 2021 10:31 AM

arr → 4 3 1 5 6

maximum of min & second min

Ce adjacent two elements

max

All possible Subarrays with atleast two values →

4 3 → 4+3=7
3 1 → 3+1=4
1 5 → 1+5=6
5 6 → 5+6=11 max
4 3 1 → 3+1=4
3 1 5 → 3+1=4
1 5 6 → 1+5=6
4 3 1 5 → 3+1=4
3 1 5 6 → 3+1=4
4 3 1 5 6 → 3+1=4

use less Subarray

Result = 11

Observational logic

logic → find max sum of two adjacent Elemt.

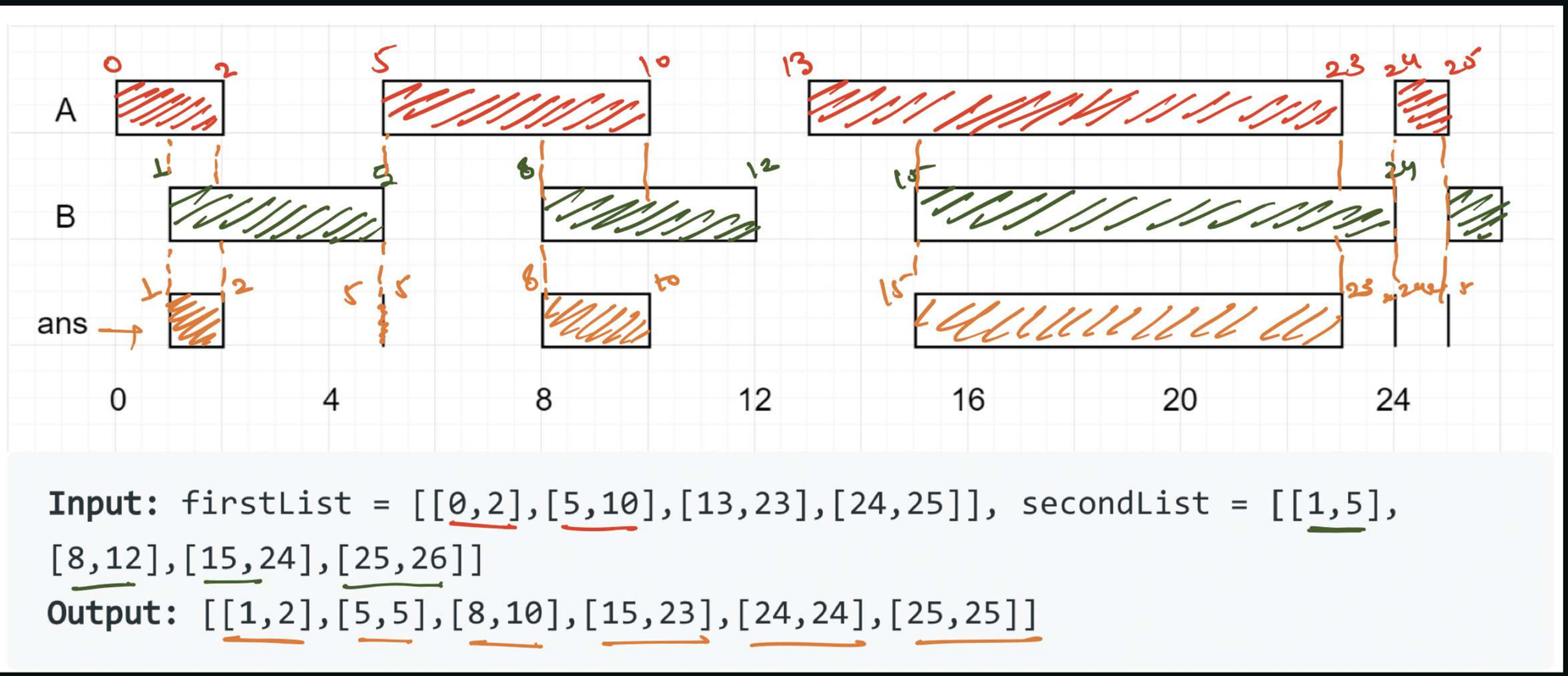
4 3 1 5 6

43 31 15 56

7 4 6 11 ⇒ 11 Result

Interval List Intersection

Saturday, 18 September 2021 10:31 AM



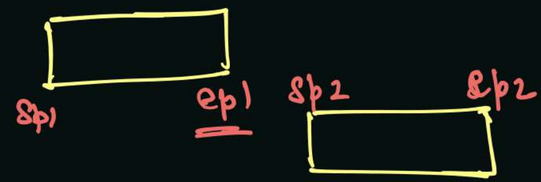
Input: firstList = [[0,2],[5,10],[13,23],[24,25]], secondList = [[1,5],[8,12],[15,24],[25,26]]

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

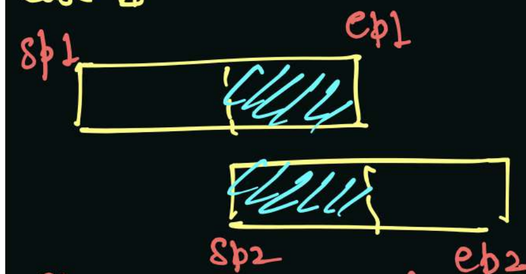
Possible cases with two list

Case-I

ep1 < sp2 → No Intersection.



Case-II

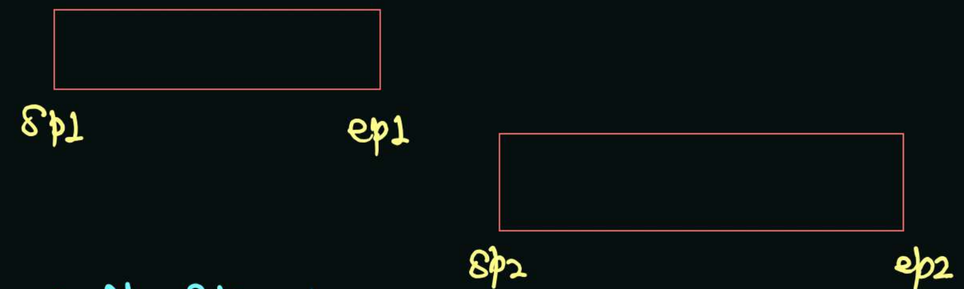


Intersection st → Max(sp1, sp2)

End → min(ep1, ep2)

Case-III

Case-I



→ No intersection

Starting = max(sp1, sp2) = sp2

Ending = min(ep1, ep2) = ep1

Here ep1 < sp2
So it is not possible to end before interval start.

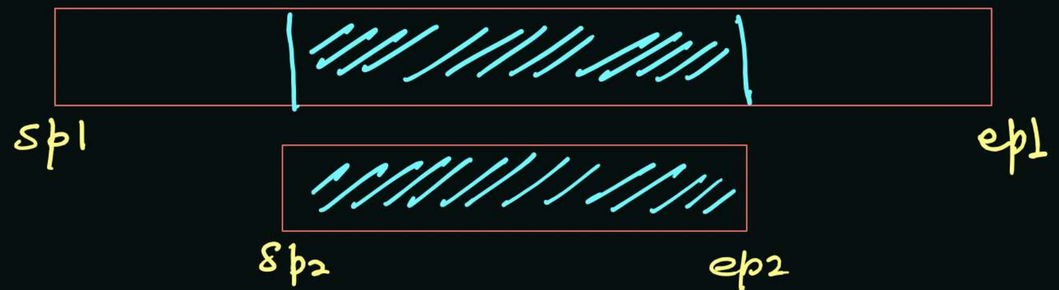
i.e. ⇒ No intersection

Case-II

Starting = Max(sp1, sp2) = sp2

Ending = min(ep1, ep2) = ep1
(sp2 - ep1)

Case-III

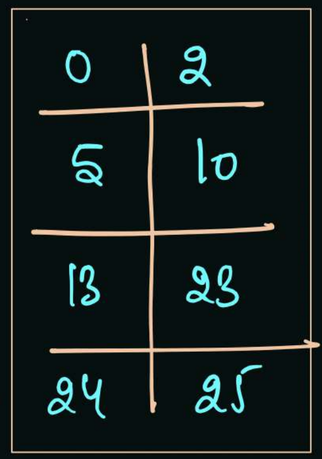


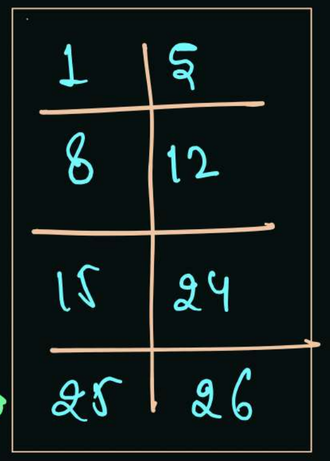
Starting = max(sp1, sp2) = sp2

Endin = min(ep1, ep2) = ep2

Interval = (sp2, ep2)

First List = [[0,2],[5,10],[13,23],[24,25]],
second List = [[1,5],[8,12],[15,24],[25,26]]





[i →] ✓

Result

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

How to increase pointer -

→ on the basis of ending point.

Insert Intervals

Saturday, 18 September 2021 10:31 AM

Merge Intervals → time interval

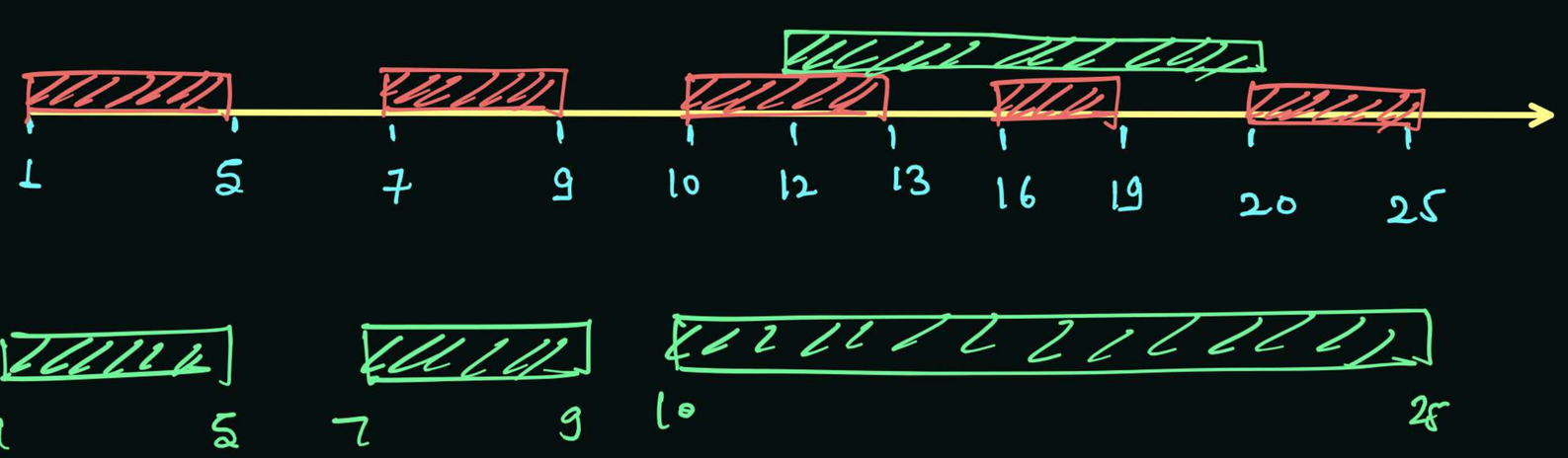
$O(n \log n)$

Intervals → 5 → no. of intervals

1 5
7 9
10 13
16 19
20 25

n intervals

new interval → 12 20

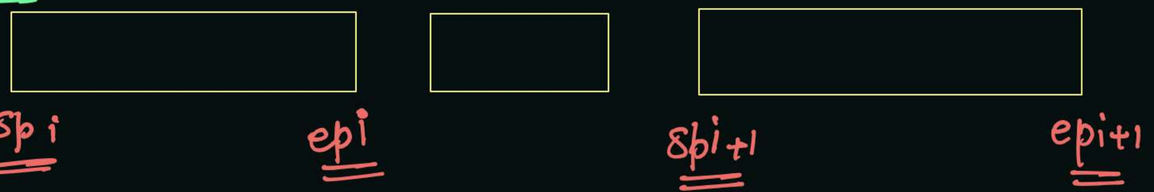


After insertion

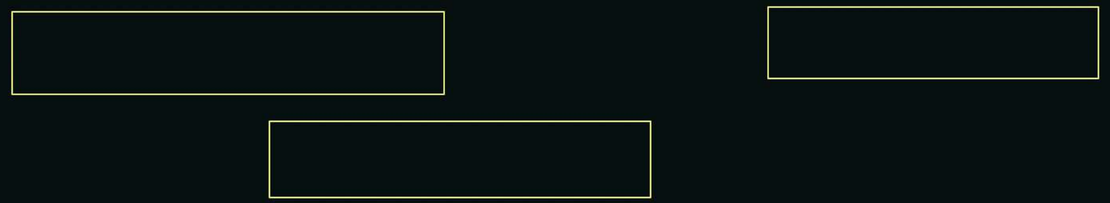
1-5 7-9 10-25

Allowed → $O(n)$

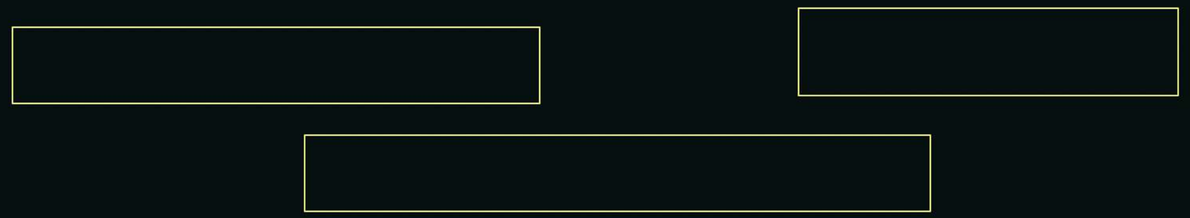
Case -I

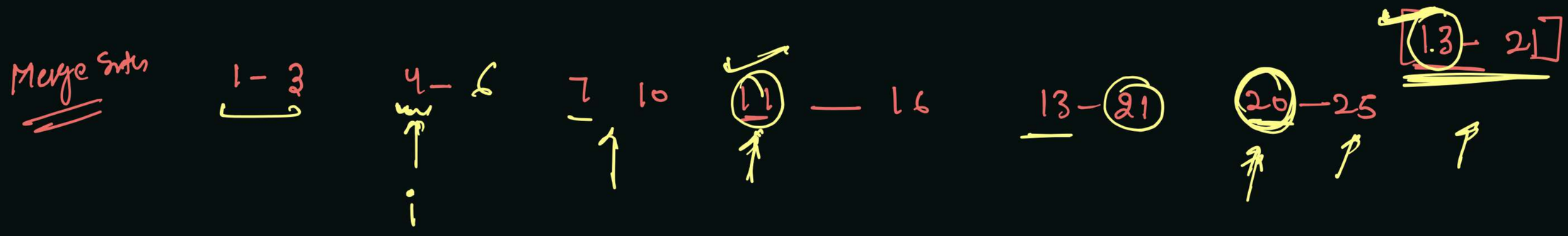


Case -II



Case -III





lsp = ~~1~~ ~~4~~ ~~7~~ 11

lep = ~~3~~ ~~6~~ ~~10~~ ~~16~~ ~~21~~ 25

[1-8] [4-6] [7-10] [11-20]