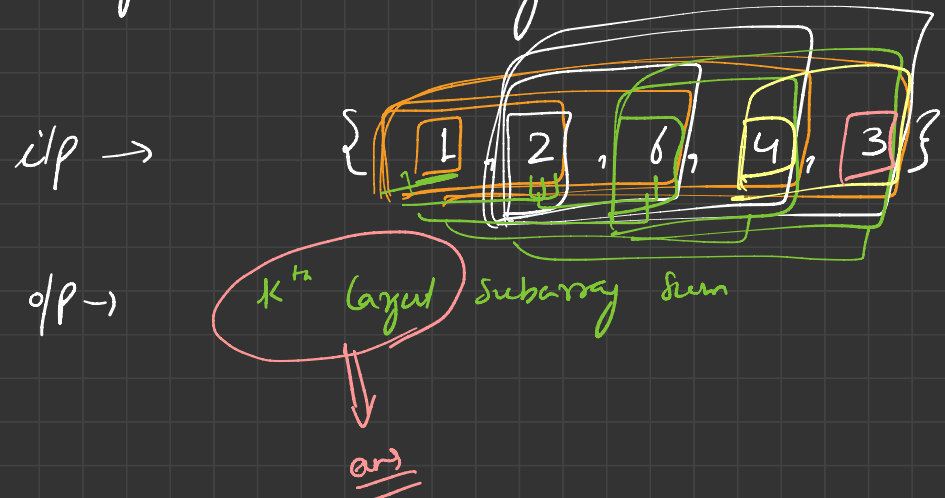
Heaps

→ K^{th} Largest Sum Subarray



{1} → 1

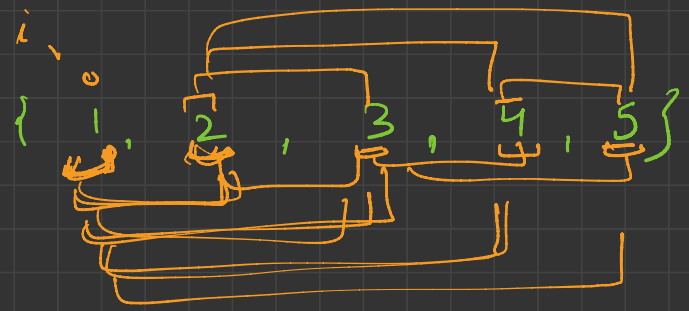
{1, 2} → 3

{1, 2, 3} → 6

{1, 2, 3, 4} → 10

{1, 2, 3, 4, 5} → 15

approach :




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

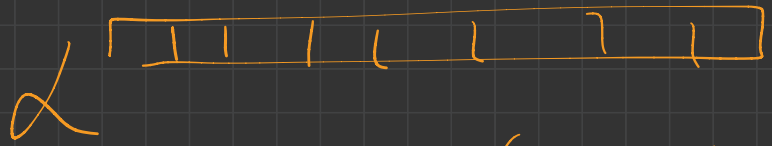
① all subarrays

sum

$O(n^2)$

vector <int> sum

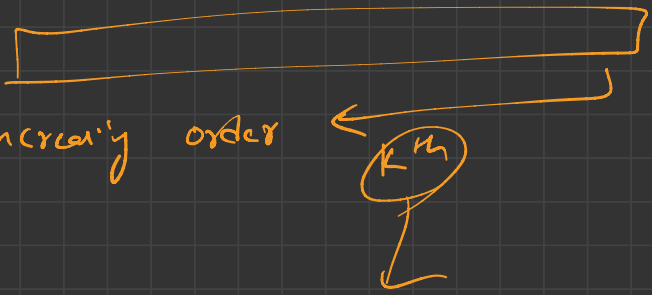
Sum



$\rightarrow L \rightarrow n^2$

sort $\rightarrow$ T.C $\rightarrow$ $(L \log(L))$ (II) $\rightarrow$ $n^2 \log n$

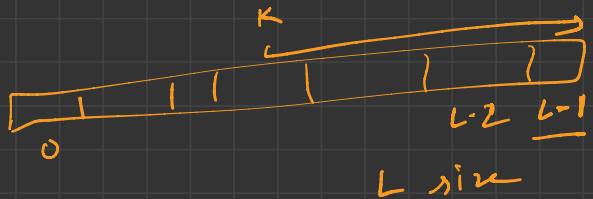
K^{th} largest



T.C $\rightarrow$ $(n^2 \log n)$

S.C $\rightarrow$ $O(n^2)$

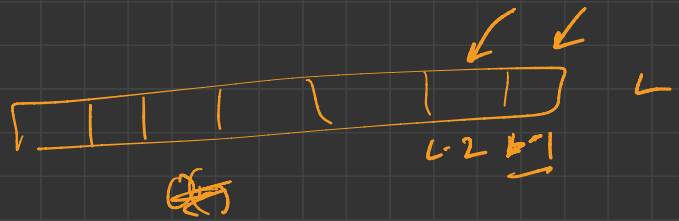
easy (III)



$I^{st} \rightarrow L-1 \rightarrow$ sirde

$2^{nd} \rightarrow L-2$

$K^{th} \rightarrow L-K$



K^th Largest → O(K) O(n²) / O(n log n)

Approach #2 (Min-Heap)

K^th Last

1st → L-1

2nd → L-2

K^th → L-K

min_heap

→ For (i = 0 < n)

{

For (j = i , j < n)

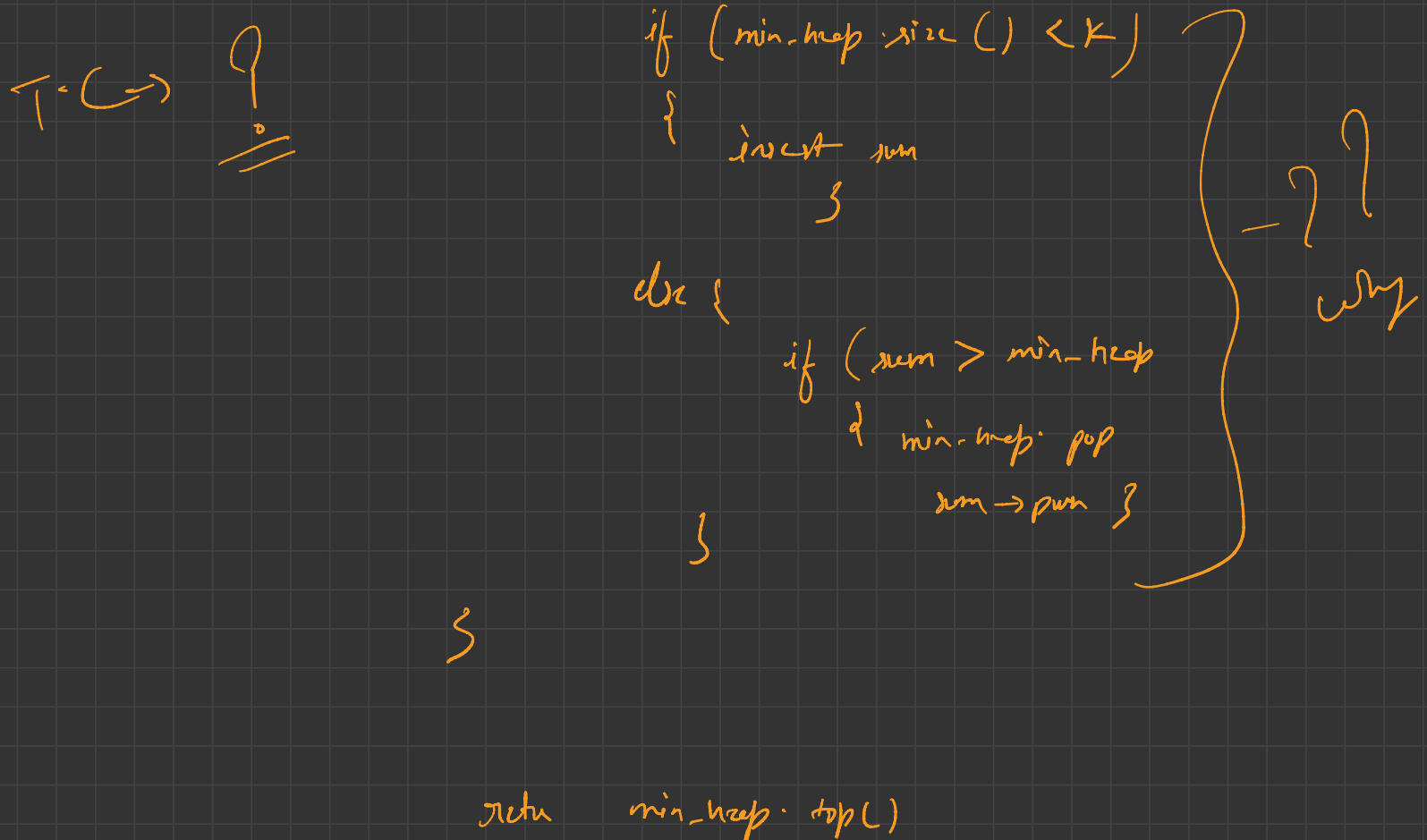
{

sum = ___

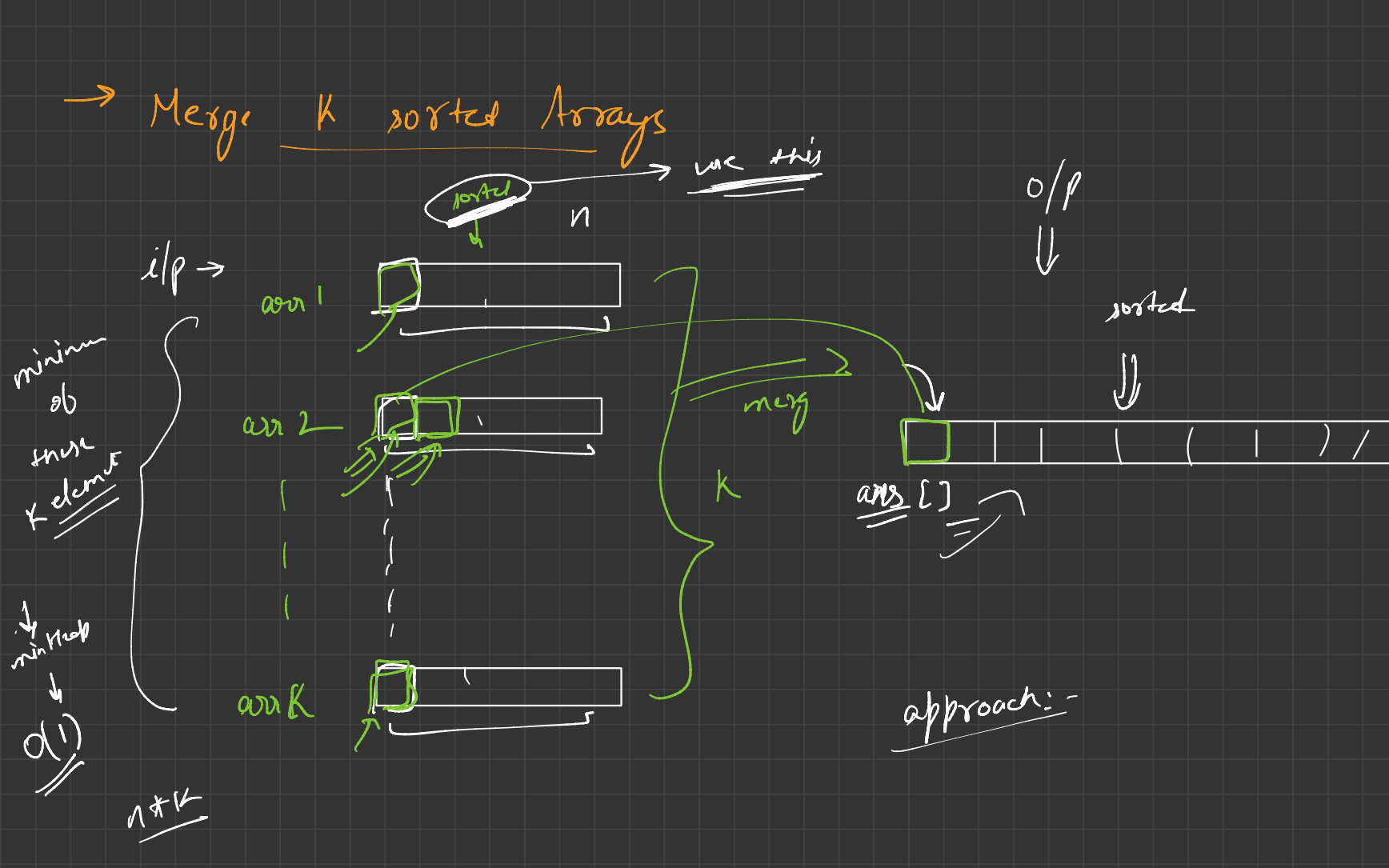
O(K)

O(n²)

→ O(K)


T<(=>)
!
!!
if (min_heap.size() < K)
{
insert sum
}
else {
if (sum > min_heap
{ min_heap.pop
sum → pum }
}
}
?
?
why
return min_heap.top()

→ Merge K sorted Arrays



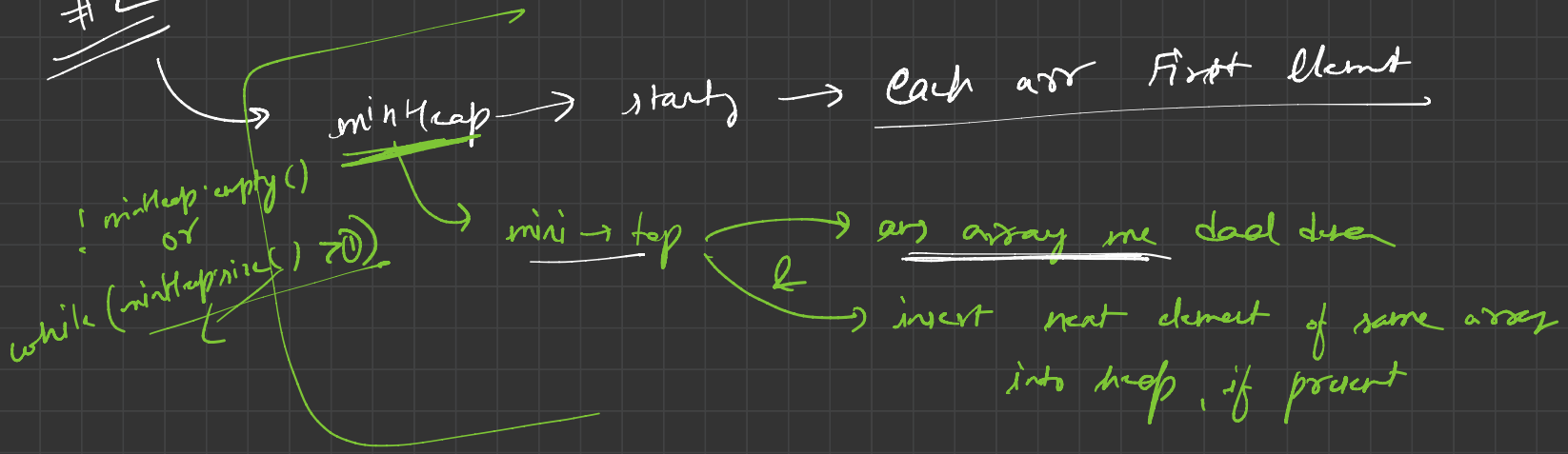
#1

① Create an array → O(1)

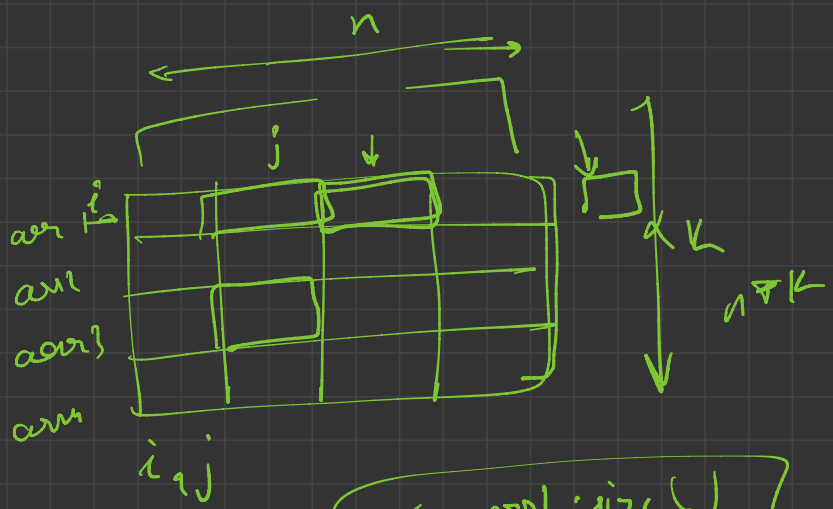
② insert all element [n*k] into an array → O(n*k)

③ sort → an array → T.C([n*k log (n*k)])

#2



minHeap < int



i < arr.size()

node
{
int data
int i;
int j;
}

K

$O(K \log K)$

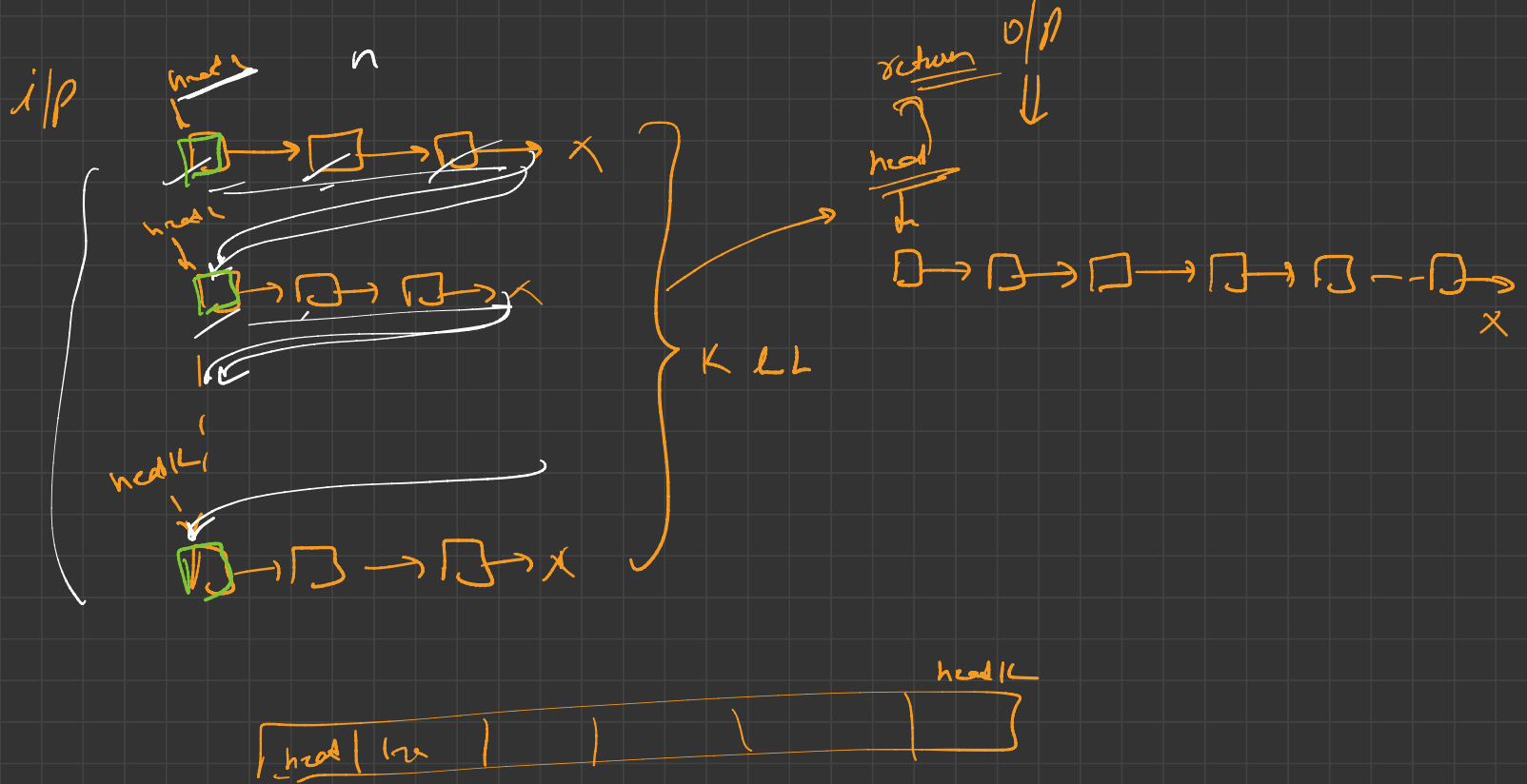
$n * K$ - K

$\sim n * K$

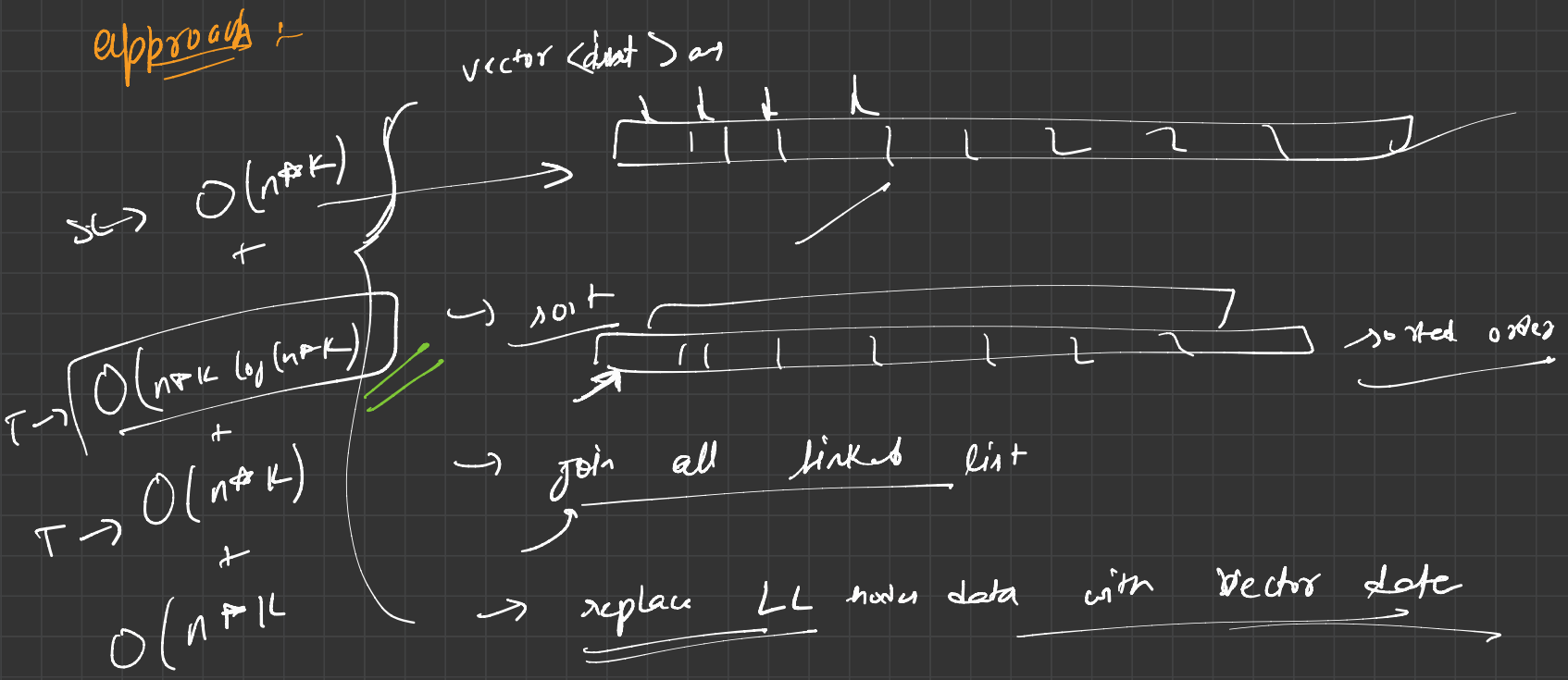
$O(n * K \log(K))$

space → $O(K) + O(n * K) \rightarrow O(n * K)$

→ merge K sorted Linked List



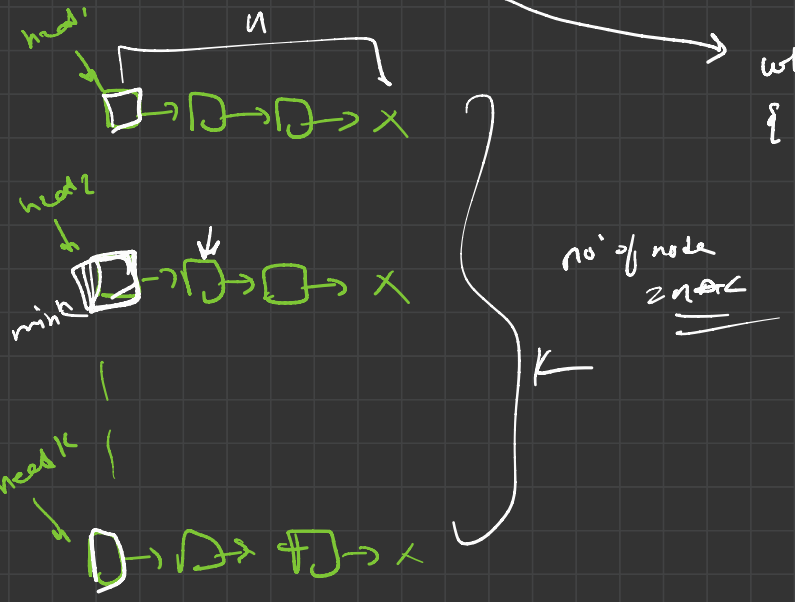
approach :-

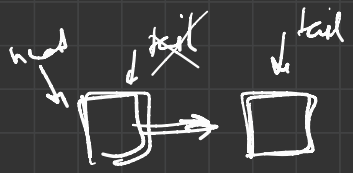

vector<int> arr
SC → O(n*k)
+
T → O(n*k log(n*k))
+
T → O(n*k)
+
O(n*k
sort
sorted order
join all linked list
replace LL node data with vector data

#2

ans → LL → [head / tail]
delete → insert

minHeap → first element of K linked list






```
while (!minHeap.empty())
{
    if (head == NULL)
    {
        head = tail = minHeap.top()   // → insert LL
        minHeap.pop()
        if (head → next != NULL)
        {
            minHeap insert }
    }
else
{
    tail → next = minHeap.top()
    minHeap.pop()
    tail = tail → next
}                                     // insert LL
```

```
if (tail -> next != NULL)
    {
        minHeap.insert(next)
    }
}

return head;
```

$T.C \rightarrow O(K \log K) + O(n * K \log K)$

$n * K - K$

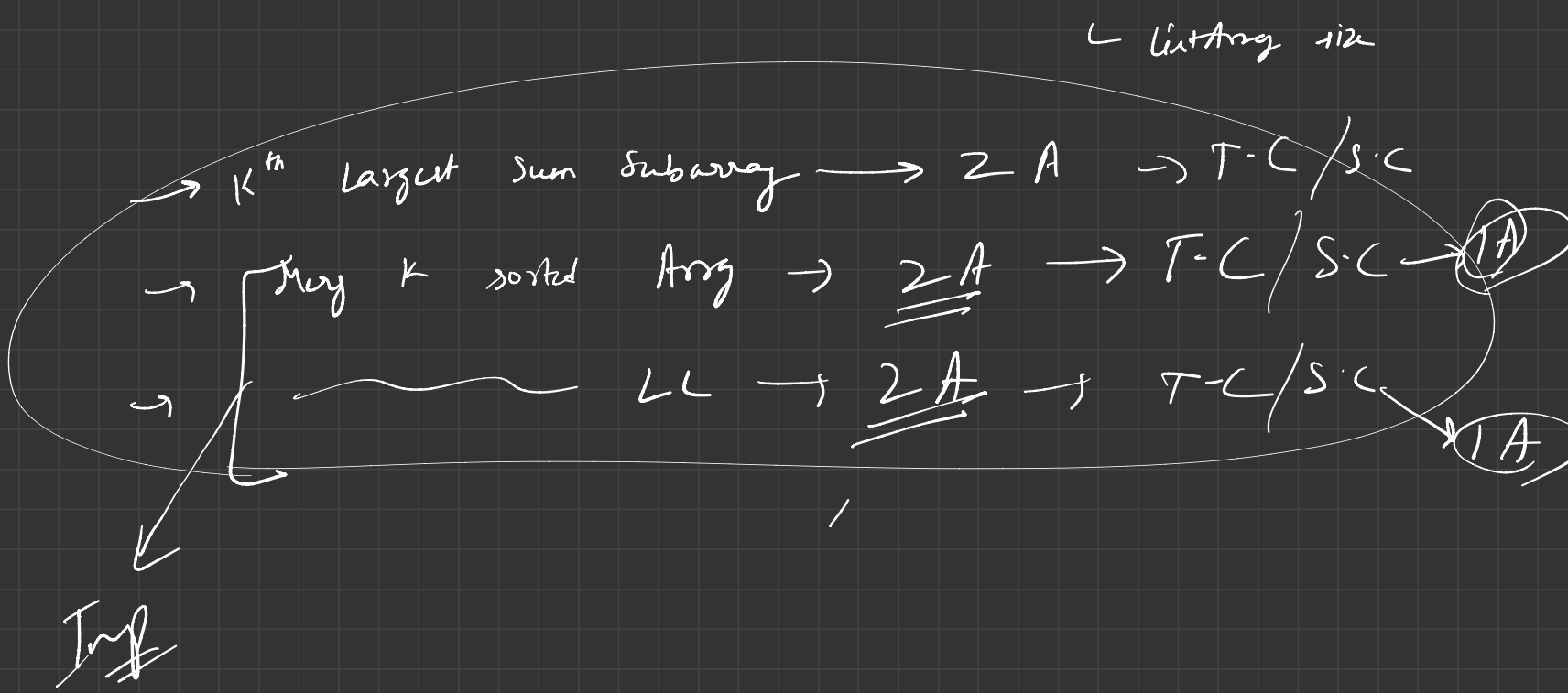
$\approx n * K$

$\rightarrow O(n * K \log K)$

no of node in a LL

$O(N \log K)$ → N = Total no of nodes

$S.C \rightarrow O(K)$


K^th Largest Sum Subarray → 2 A → T·C / S·C
Merg K sorted Arrg → 2A → T·C / S·C → 1A
LL → 2A → T·C / S·C → 1A
Imp