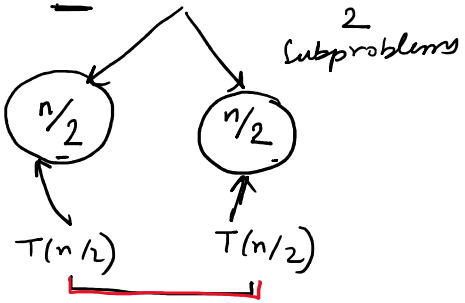
18 February 2025 08:37

Merge Sort Analysis -

$\rightarrow T(n) = 2T(n/2) + n.$ (2T(n/2): Divide; n: Merge / Conquer)

$n \rightarrow$ Size of problem. $\leftarrow T(n)$



$O(n) \rightarrow$ Merge 2 sorted arrays into a single sorted array.

$T(n) = 2T(n/2) + n.$ $\qquad$ $T(n) = aT(n/b) + n^k \log^p n$

$\Downarrow$ By Master Th^m -

$a = 2, b = 2, k = 1, p = 0$

$a \; b^k$

$2 = 2^1$ $\qquad$ Case 2a $\rightarrow$ $T(n) = O\left(n^{\log_b a} \times \log^{p+1} n\right)$

Property $\log_a a = 1$

$= O\left(n^{\log_2 2} \cdot \log^{0+1} n\right)$

$= O(n^1 \cdot \log n) = n \log n.$

Space complexity - $O(n) \rightarrow$ Merge 2 sorted arrays into 1 sorted array then it uses

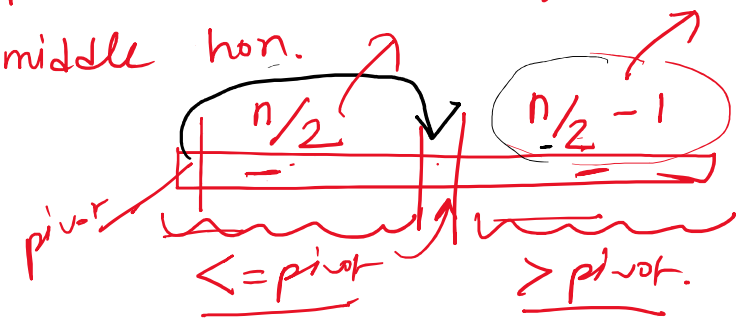
into 1 sorted array then it uses
a temporary array of size =
left_arr size + right_arr_size.

Temp arr ka max size = n

Quick Sort Analysis → TC, SC.

SC → O(1) → No XTRA space used.

TC → Best Case = ① Pivot element ka original position Hamesha middle hoga.



n

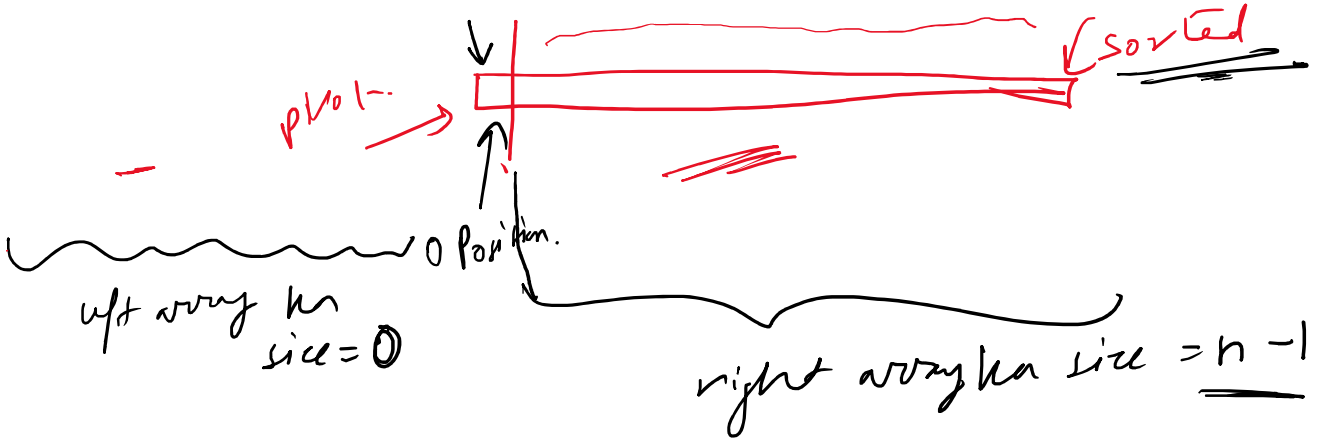
Recurrence Relation = $T(n) = 2T(n/2) + n$ → Swapping perform whole array traverse.

Master Thm

$T(n) = O(n \log n)$.

$T(n) = O(n \log n)$.

Worst Case - ① Input array is already sorted and you r considering 1st element as pivot.



RR ⇒ $T(n) = T(n-1) + n$ ← Swapping $T(n) = aT(n/b) + \left(n^k \log^p n\right)$

⇓ Substitution method.

$T(n) = T(n-1) + n$.

$T(n-1) = T(n-2) + (n-1)$

$T(n-2) = T(n-3) + (n-2)$

⋮

$$T(2) = T(1) + 2.$$

$$T(1) = 1.$$

$$T(n) + T(n-1) + T(n-2) + \cdots + T(2) + T(1) = T(n-1) + n + T(n-2) + (n-1) + T(n-3) + (n-2) + \cdots + T(1) + 2 + 1.$$

$$\Rightarrow T(n) = n + (n-1) + (n-2) + \cdots + 2 + 1$$

$$= \frac{n \times (n+1)}{2} = \frac{n^2 + n}{2} \simeq O(n^2).$$

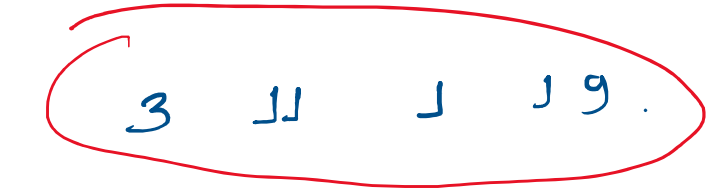
Avg Case — TC → $O(n \log n)$.

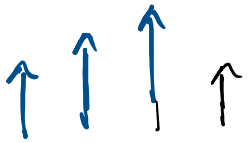
3, 3, 3, 3, 3

SC → $O(1)$ — ∵ No x TRA space

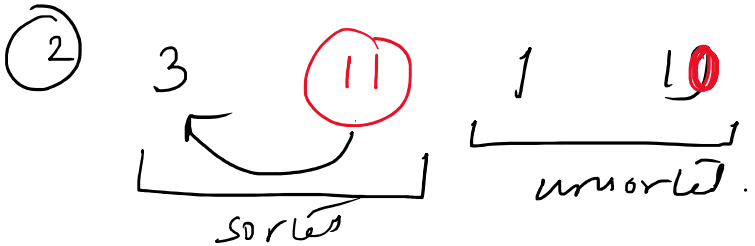
5

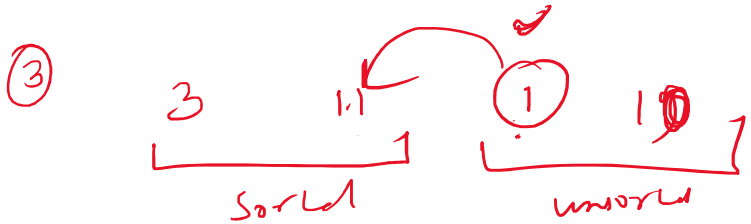
Insertion sort





Eg





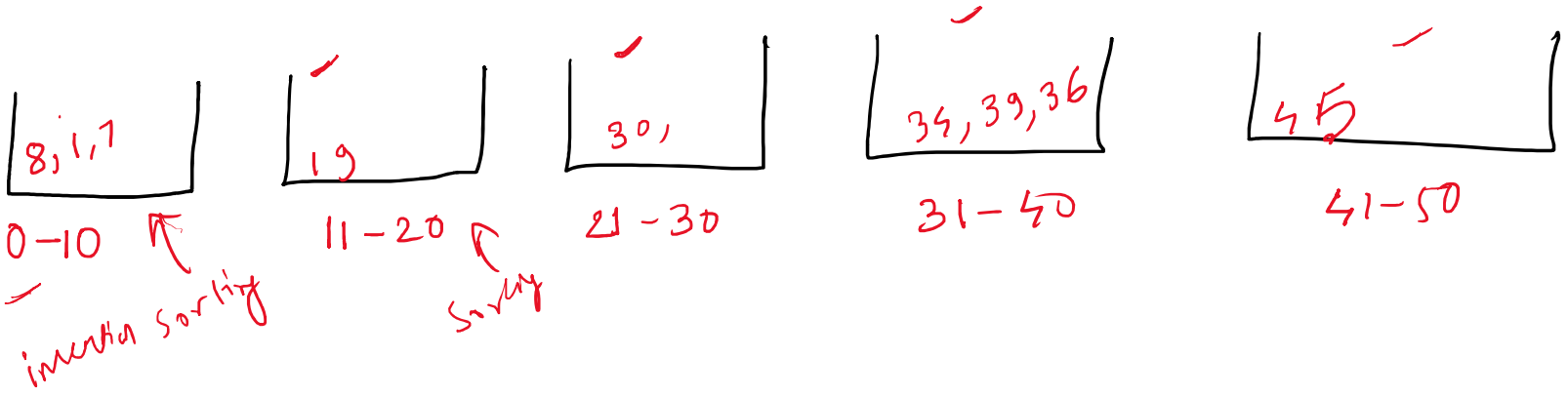
(4) 1 3 10 11

Sorted.

Bucket sort:

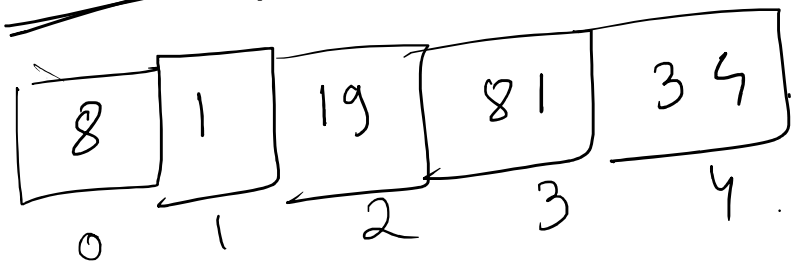
8 1 19 89 34 7 30 82 39 45 36

Q) No. of buckets = 10.



1, 7, 8, 19, 30, 34, 36, 39, 45, 82, 89 ← sorted

Searching - Array → Element to Search -



X = 81

return index.

```
ind = -1 ;
for (i=0 ; i<=n-1 ; i++) {
    if (arr[i] == X) {
        ind = i ;
        print(ind) ;
        break ;
    }
}
```

Eg

8 1 5 1 20.

X = 1

return index where X occurs for 1st time.

return index where X

```
}
if(ind == -1) {
        print(Element not found)
}
```

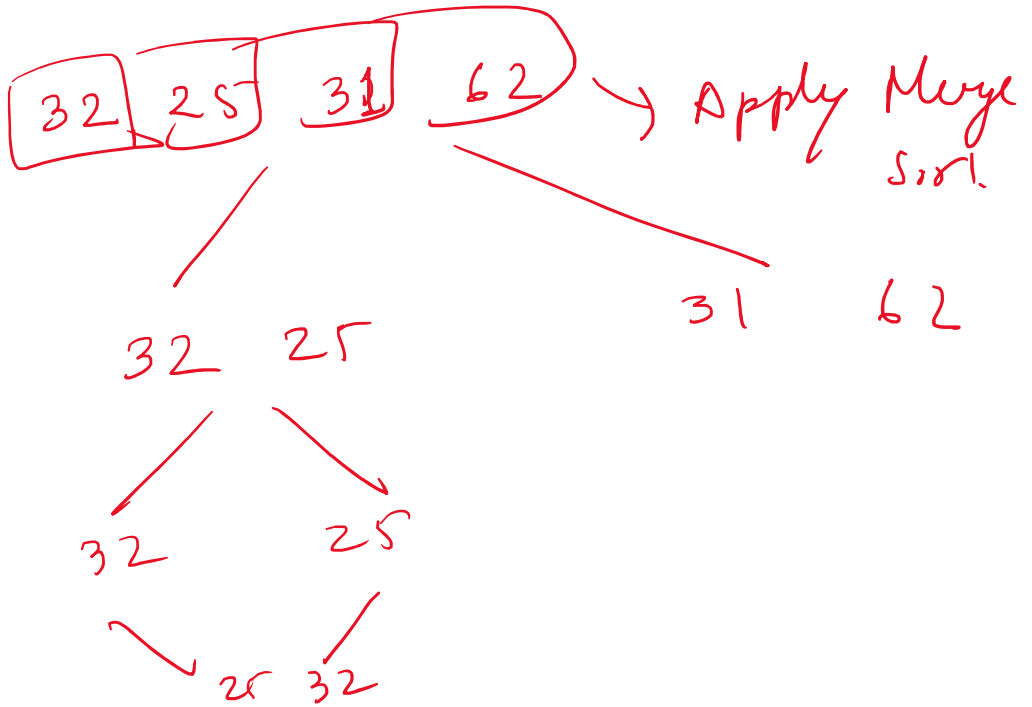
return index where X occurs for last time.

TC →

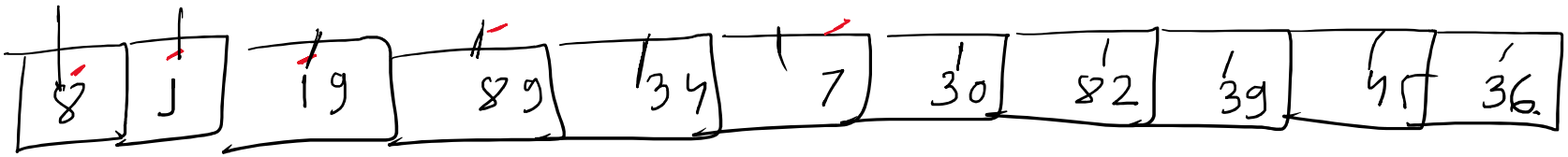
bcdbchsdbcsdbkcs

Algorithm

example →



Inp Arr =



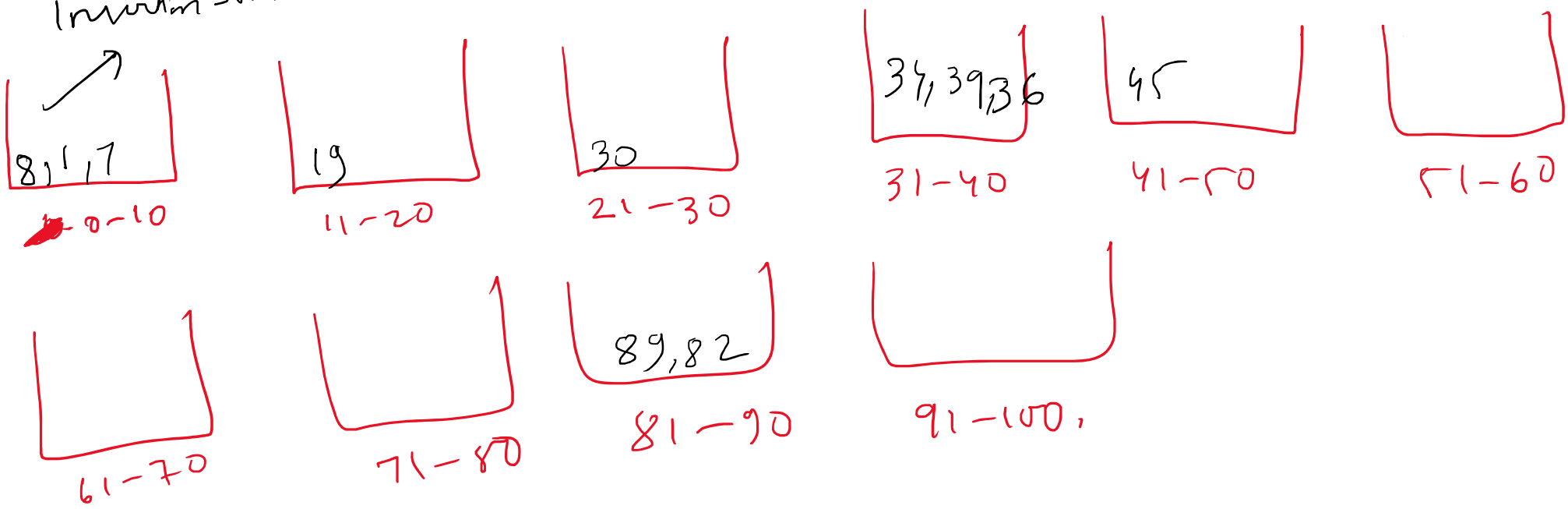
Arr =

No. of buckets = 10.

Min = 1

Max = 89.

Insertion Sort



1, 7, 8, 19, 30, 34, 36, 39, 45, 82, 89.