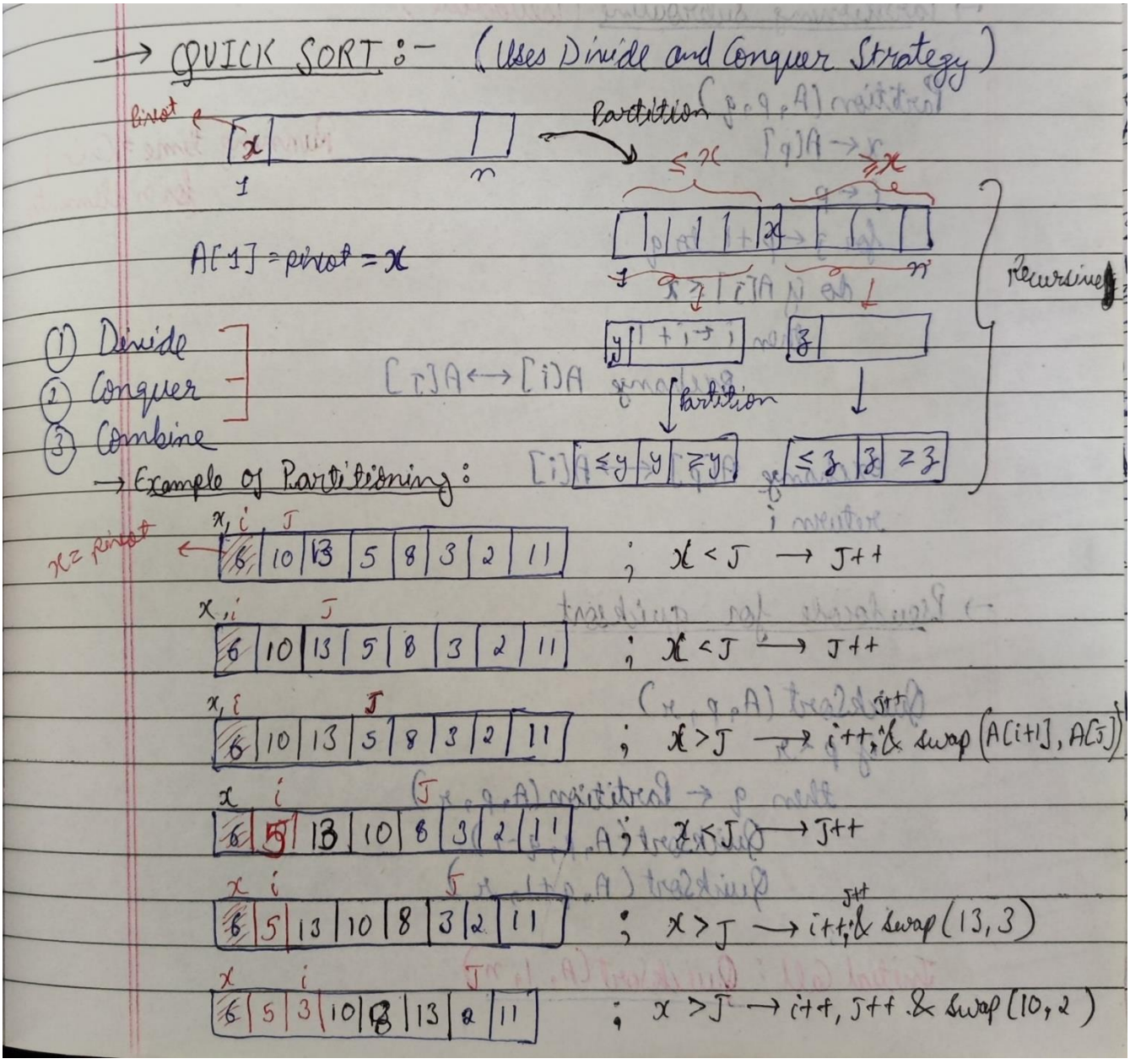
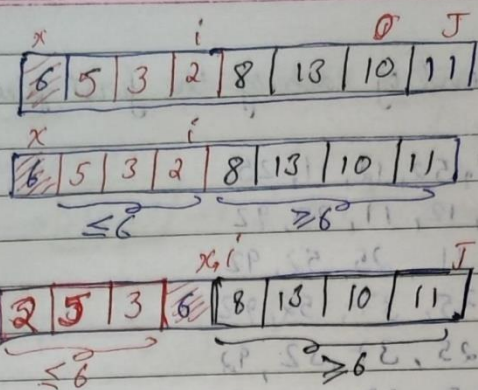
QUICK SORT


→ QUICK SORT :- (Uses Divide and Conquer Strategy)
pivot
Partition
A[1] = pivot = x
(1) Divide
(2) Conquer
(3) Combine
Recursive
→ Example of Partitioning :
x = pivot
; x < J → J++
; x > J → i++, & swap (A[i+1], A[J])
; x > J → i++, & swap (13, 3)
; x > J → i++, J++ & swap (10, 2)



→ Partitioning subroutine (Pseudocode)

```
Partition(A, p, q)
    x ← A[p]
    i ← p
    for j ← p+1 to q
      do if A[j] ≤ x
          then i ← i+1
               exchange A[i] ↔ A[j]
    exchange A[p] ↔ A[i]
    return i
```

Running time = O(n) for n elements

→ Pseudocode for quicksort

```
QuickSort(A, p, r)
    if p < r
      then q ← Partition(A, p, r)
           QuickSort(A, p, q-1)
           QuickSort(A, q+1, r)
```

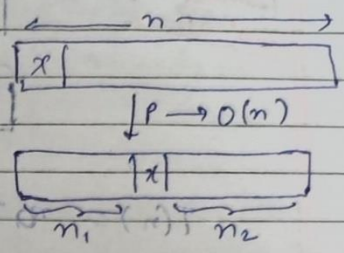
Initial Call: QuickSort(A, 1, n)

→ Time complexity of Quick sort:-

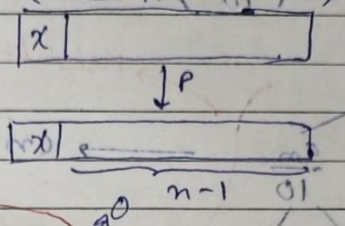
$T(n) = T(n_1) + T(n_2) + c.n$ → Partition

& $n_1 + n_2 + 1 = n$

Total elements in array



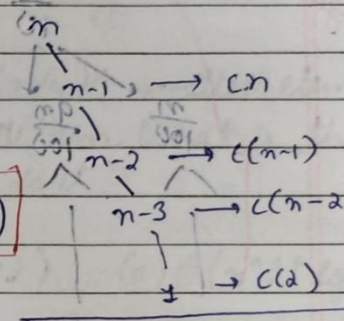
Worst Case :



where x: smallest/min element & already sorted array

b/c there is nothing to sort (pivot is the smallest element)

$T(n) = T(0) + T(n-1) + cn$

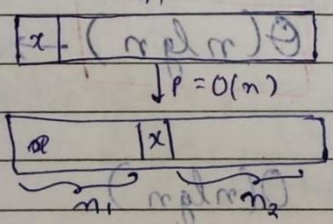


Worst Case Time Complexity = $O(n^2)$

$= c[n + (n-1) + (n-2) + \dots + 1]$

$= c\left[\frac{n(n+1)}{2}\right] = O(n^2)$

Best Case :



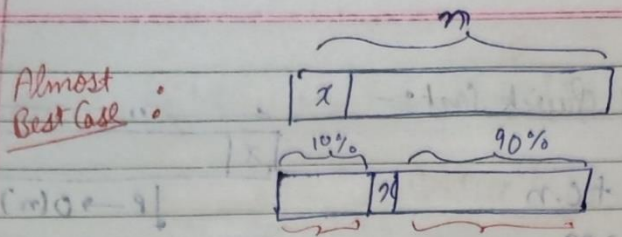
where $n_1 \approx \frac{n}{2} \approx n_2$

$T(n) = T(n/2) + T(n/2) + cn$

$= 2T(n/2) + cn$ ← Same as merge sort

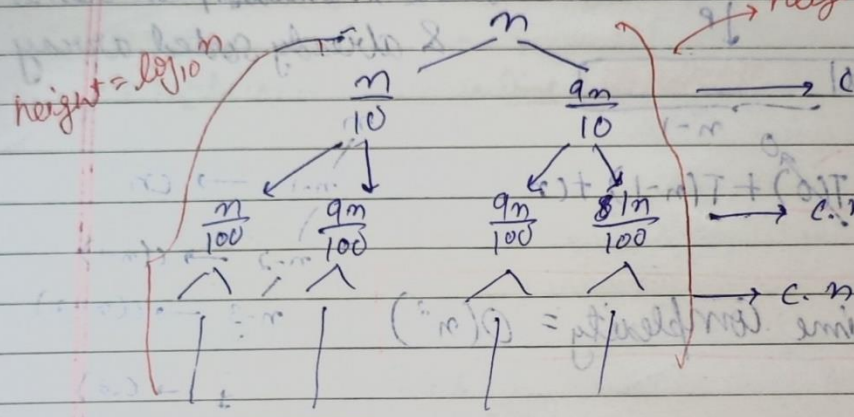
$T(n) = O(n \log n)$ → Best Case

Almost Best Case :



$n_1 \simeq \frac{n}{10}$ & $n_2 \simeq \frac{9n}{10}$

$$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9n}{10}\right) + cn$$



$$cn * \log_{\frac{10}{9}} n$$

$\Theta(n \lg n)$ → Time complexity

→ Randomized Quicksort :- $O(n \lg n)$

↳ pivot = random element from $A[p \ldots r]$

Steps : ① Pick a random number (1 to n). → Let's say m

② $A[1] \overset{\text{swap}}{\Longleftrightarrow} A[m]$

③ pivot = $A[1]$