* **Merge Sort** :- ↳ Divide And Conquer Algorithm
↳ Stable Algorithm
↳ $O(n \log n)$ time and $O(n)$ Aux Space
↳ Best for Linked Lists. Works with $O(1)$ Auxiliary Space.
↳ Used in External Sorting.
↳ For general arrays, Quick Sort outperform Quick sort.

Java 8 → uses Merge + Quick Sort
Python → uses Merge + Insertion Sort = Tim Sort

Date
DELTA Pg No

Time complexity = $\theta(\log_2 n) * \theta(n)$
$= \theta(n \log_2 n)$

↳ Generally, $\lceil \log_2 n \rceil$ * 1 total no. of steps.
At each step merging takes n iterations
there for $O(\log_2 n * n)$ time complexity.

- # QUICK SORT (Only Naive Partition is Stable partition)
  ↳ O(n) extraspace.
  ↳ Not Inplace

→ Lamuto Partition → Unstable.
↳ Inplace

We work on the approach that elements till 0 → i are less than pivot and elements from i+1 to j-1 are greater than pivot

pivot is always last element then :-

Code :-

```
int pivot = a[h]          // h → last index
int i = l-1;              // l → lowest index
for(int j=l; j<=h-1; j++){
    if(a[j] < pivot){
        i++;
        swap(a[i], a[j]);
    }
}
swap(a[i+1], a[h]);
return i+1;
```

↳ position of pivot.

Corner Cases :-

(1) When pivot is smallest

{70, 60, 80, 40, 30}

i=-1

↳ No swapping will be done so i will remain -1 and at the end.

i++ ⇒ i=0 {30, 60, 80, 40, 70}

(2) When pivot is largest

{30, 40, 20, 50, 80}

↳ here at each iteration ~~pivot~~ each element will be swapped by itself until it reaches end of array.

Time Complexity : O(N)
Space " : O(1)

What if we want pivot to be any element then just add the code given below to the begaining of source code :-

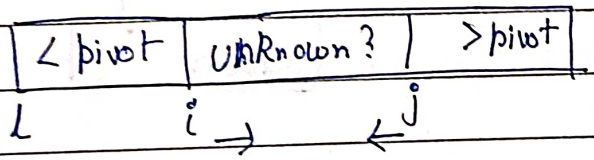
```
swap(a[p], a[h]);
p = h;
```

Date
DELTA Pg No

→ Hoare's Partition → Better than lamu to partition.
↳ Inplace but Not Stable.

→ It takes O(1) extra space and O(n) time. Also 1 traversal of Input array.

→ We consider low always as the pivot element.
→ We move i from starting of array till ~~when~~. when we get the element that is greater than pivot. Similarly we move j from ~~then~~ end of array to the begaining until we get to the starting of array or get an element less than pivot.
→ Then we swap j & i.
→ If i >= j we break from the loop and return j^th index that will be the index (j+1) from which all elements left of it are less than that and right of that are greater.



* Corner Case

* We do not fix pivot to its correct position rather, we return an index from which all to the left of it are lesser and right of it are greater.

Corner Cases

① Pivot is smallest → Pivot

$\{5, 10, 9, 12\}$ $i=-1$, $j=4$

initially $i=-1$, $j=4$

Now i will increment only once i.e $i=0$ after that j will decrement till it reach to ith element.

finally $i=0, j=0$ return j;

② Pivot is largest

$\{12, 10, 5, 9\}$

↑ pivot

$i=0, j=3$ $\{9, 10, 5, 12\}$

$i=3, j=2$ $\{9, 10, 5, 12\}$

↑ j

return j

③ All elements are same → this proves however is not stable partition due to which Quicksort will also be unstable.

$\{5, 5, 5, 5\}$

↑ $i=0$ ↑ $j=3$

Swap 1st 5 and last 5

hence prove unstable.

$\{5, 5, 5, 5\}$

↑ $i=1$ ↑ $j=2$

Swap 2nd 5 and 3rd 5.

Unstable.

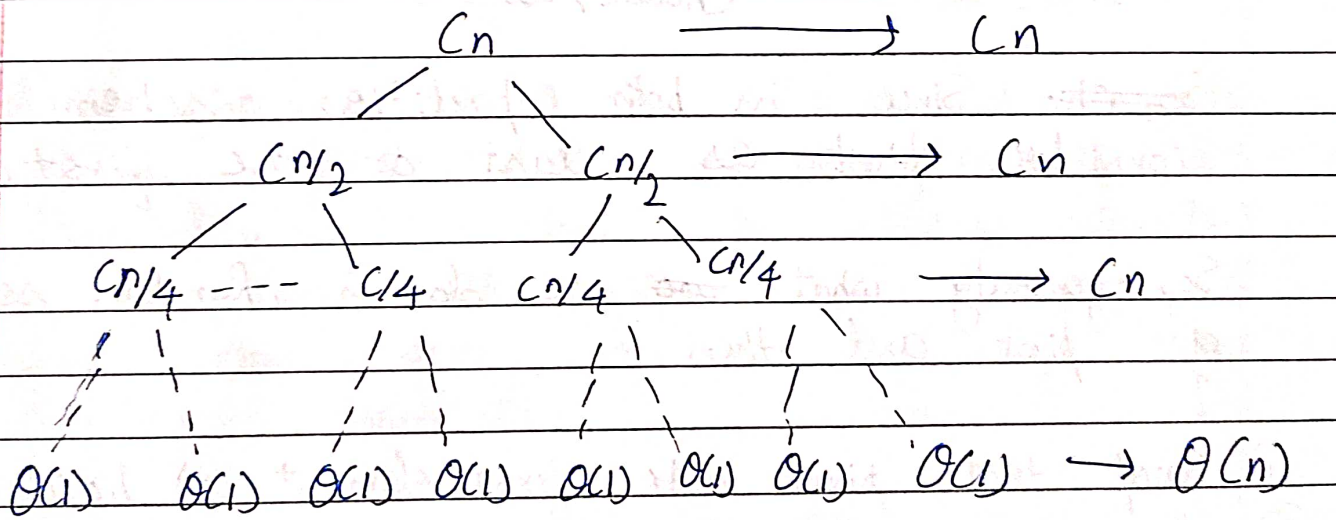
Date
DELTA Pg No.

• Quick Sort Algo

(1) Divide ~~Ang~~ And Conquer Algorithm.
(2) The key part is partitioning (Hoare, Lomuto, Naive)
(3) Worst case time complexity $O(n^2)$
(4) Despite having higher worst case time, it is preferred over other algorithm ~~ans~~ many time due to :-

(1) Tail Recursive
(2) In-Place
(3) Cache friendly
(4) Average case (nlogn)

Best Case $O(n \log n)$

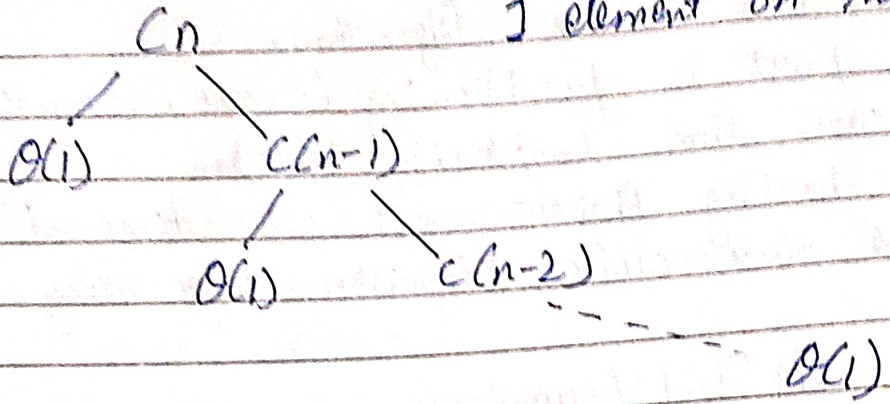


$\underbrace{Cn + Cn \cdots}_{\log_2 n \text{ times}} + \theta(n)$

$\therefore O(n \log n)$

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

- **WORST CASE** $O(n^2)$ → whenever partition divides $(n-1)$ elements on one side and 1 element on the other side.



$$T(n) = T(n-1) + \Theta(n)$$

Imp: If array is sorted ascending or decending order, both partition scheme goes into worst case. ie $O(n^2)$ (hoares, lamuto)

So, etc Since in both partition scheme we consider pivot as last or the first element.

So, generally what ~~we~~ we do is, Random selection of pivot. and then

① Swap that pivot with first element → hoares
② " " " " last " → lamuto

★ Best way to choose pivot is choosing middle element.