① Selection sort time complexity:

```
for j ← 1 to 1                          c1 → n
do min ← j                              c2 → n-1
for i ← j+1 to n                        c3 → n
    do if A[i] < A[min]                 c4 → n
    then min ← i                        c5 → n
if j ≠ min                              c6 → n-1
  then A[j] ↔ A[min]                    c7 → n-1
```

∴ Time complexity = $cn^2 + dn + c$

∴ Time complexity for best case - $O(n^2)$

for worst case - $O(n^2)$

for worst case - $O(n^2)$

Merge sort time complexity:

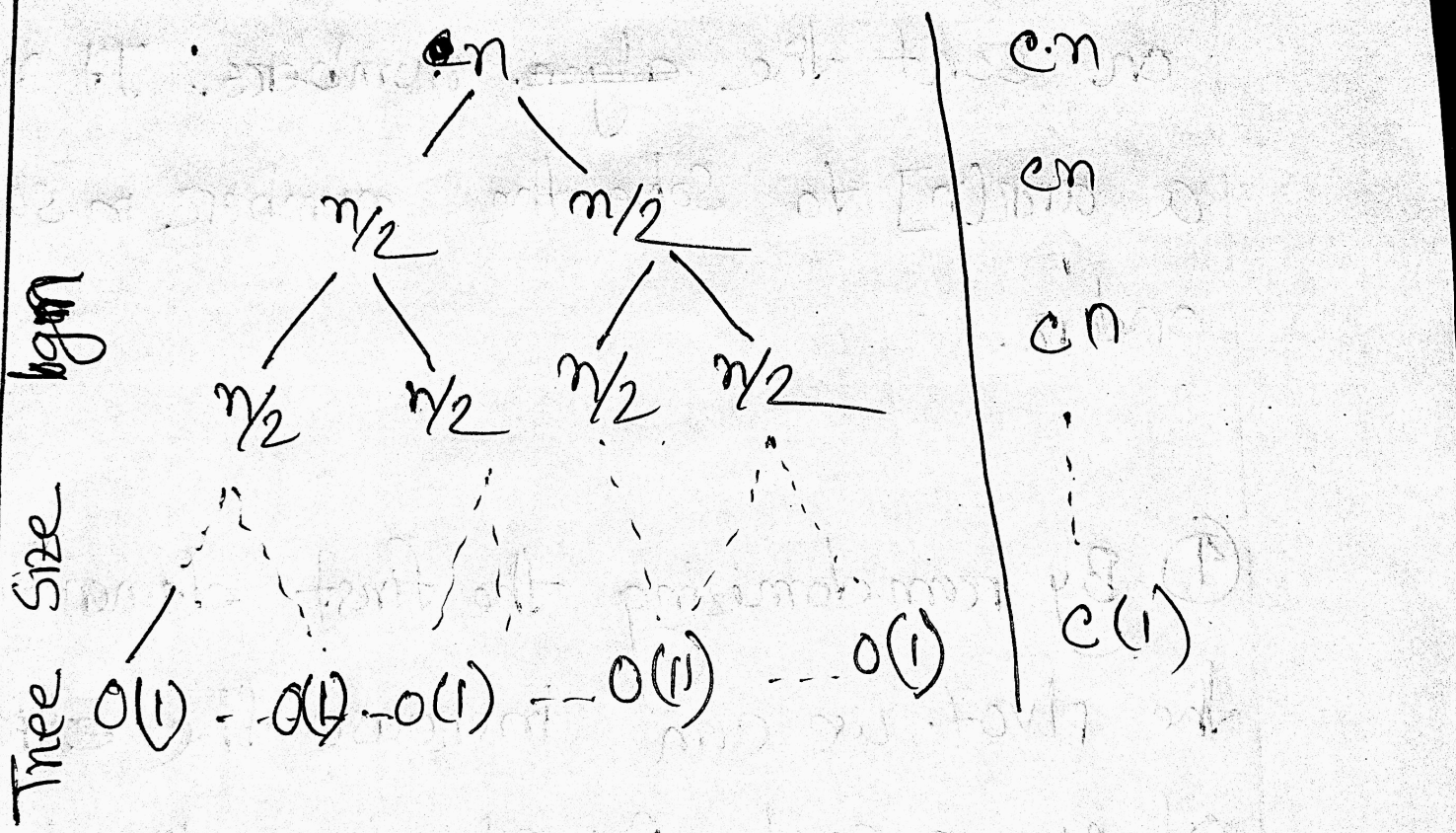
If the numeber of element is n then for best sort its complexity will be O(nlogn)

:for the worst case it will be same and also for the average case it remain same.

Quick sort time complexity:

The time complexity for Quick sort is o(n(logn) like merge sort. It also remain same for worst case and average case.

2) Recursion tree for Merge sort Algorithm.



$$\therefore \text{Total} = c(n) + c(n) + c(n) + \cdots + c(1)$$

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

$\therefore$ Size of the tree is $\log n$.

③ The optimal space for the merge sort is O(n) that required to do the task on sort the ~~algor~~ numbers. It need a arr[n] to solve the number in sorting order.

④ By randomizing the first element or the pivot we can increase the effectivity of merge sort. By choosing the pivot randomly we can ignore the worst case for merge sort.

⑤ In question 4 we already said the procedure to increase the effecitivity of merge sort algorithm as well as for quick sort.