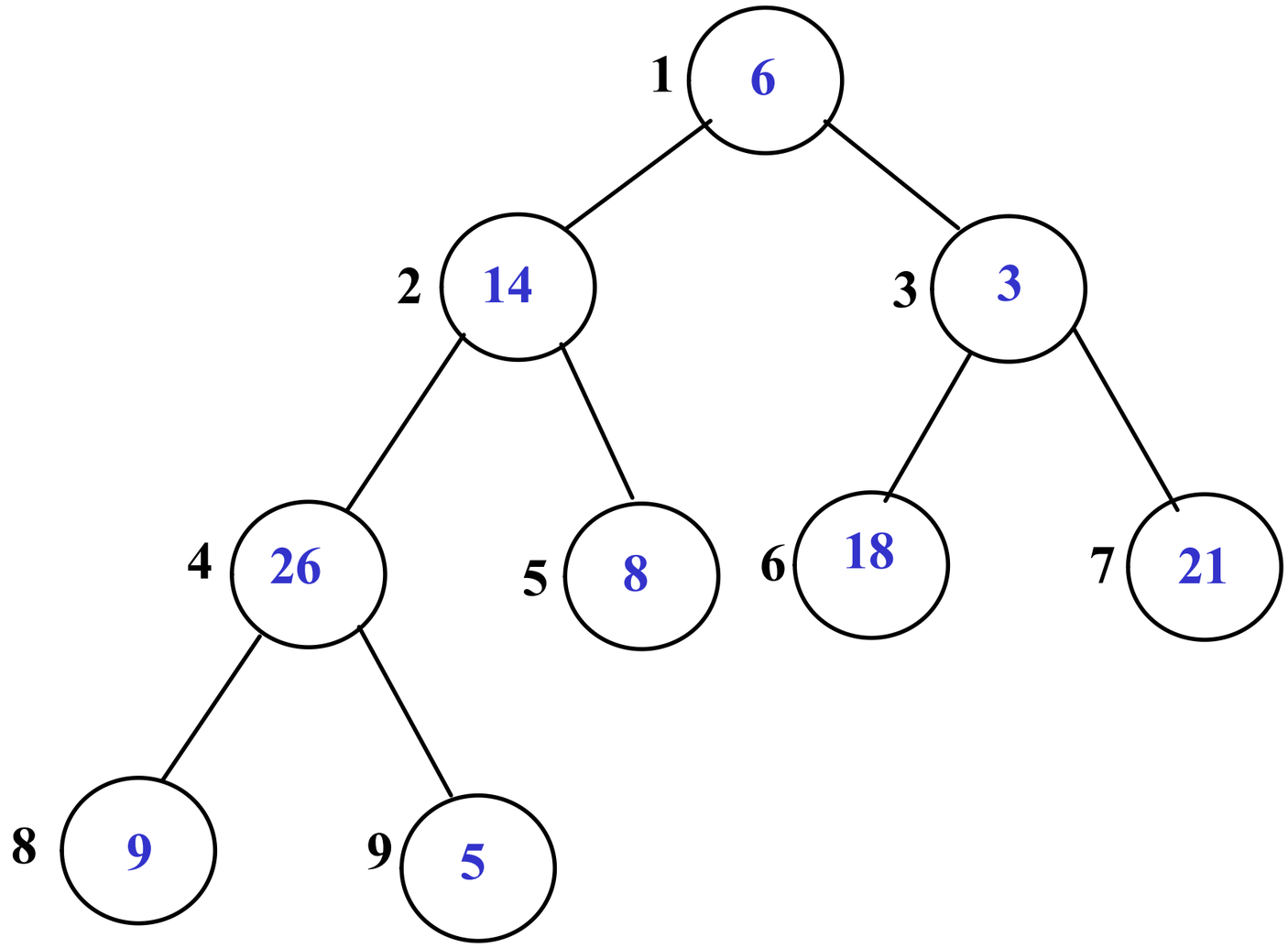
Almost Complete Binary Tree

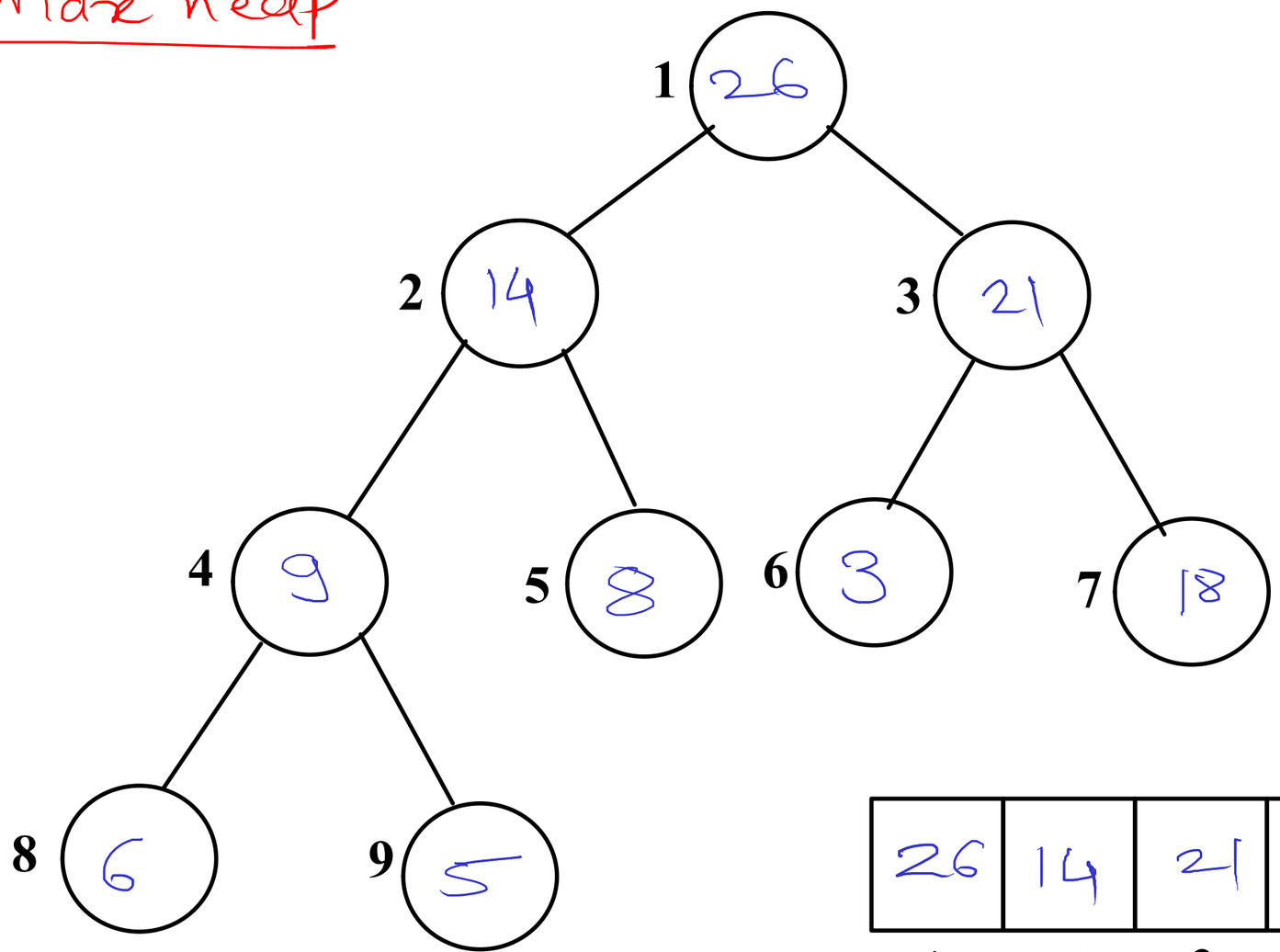


6	14	3	26	8	18	21	9	5
1	2	3	4	5	6	7	8	9

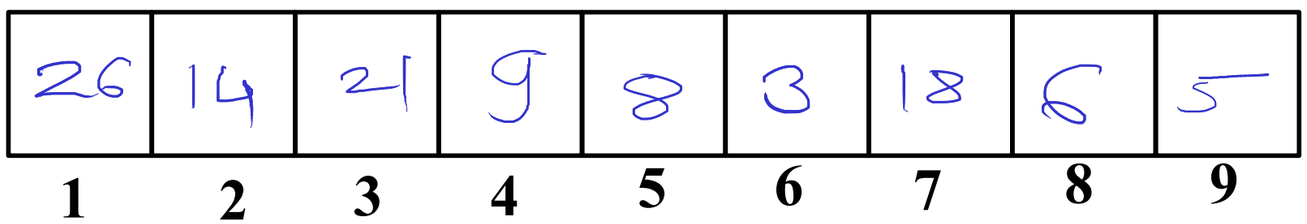
Create Max Heap

Max heap

6 14 3 26 8 18 21 9 5

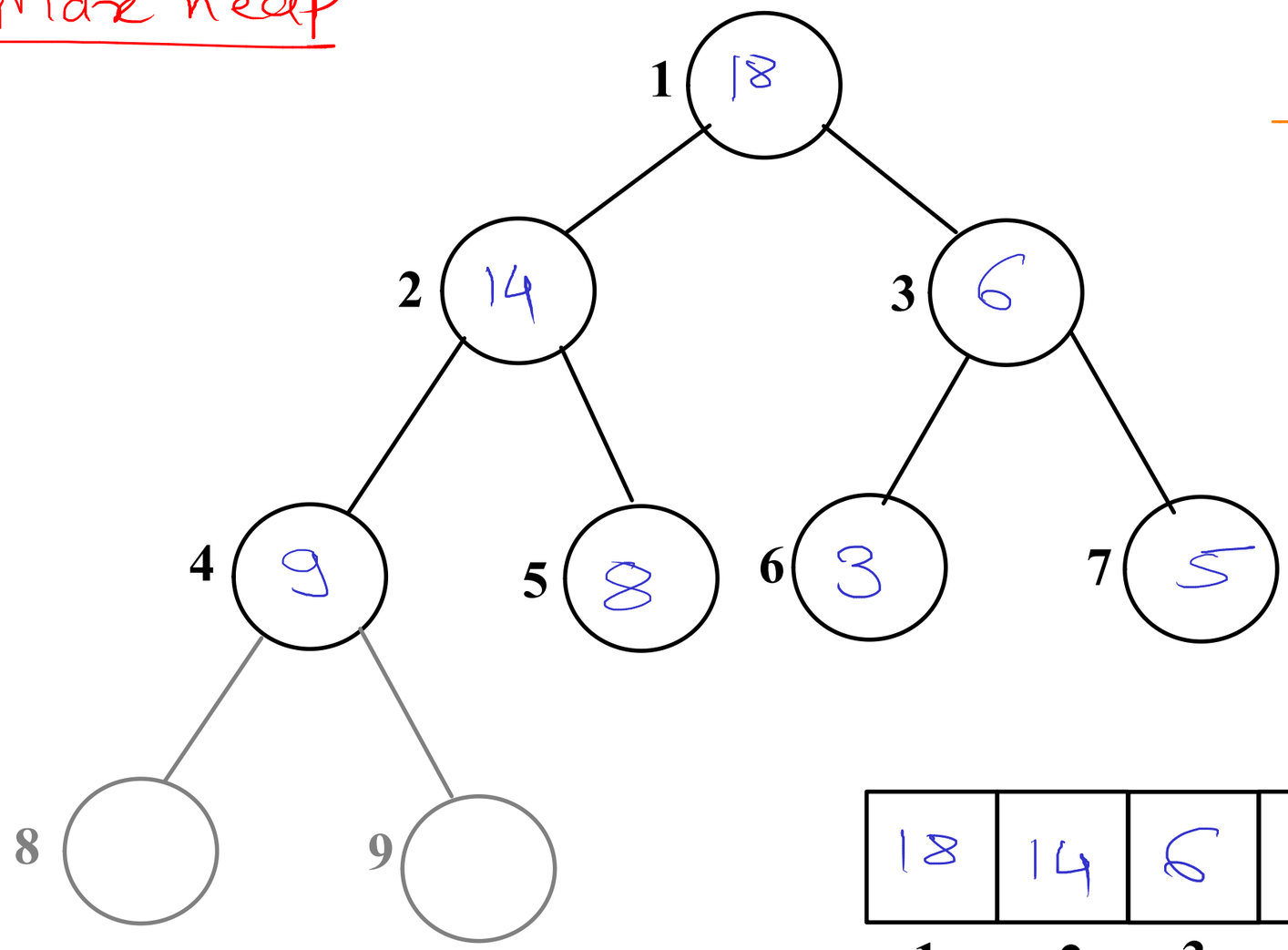


Time Complexity = O(h)
= O(log n)



delete Max Heap

Max heap



Max = 26

Max = 21

Time Complexity = O(h)

= O(log n)

18	14	6	9	8	3	5	21	26
1	2	3	4	5	6	7	8	9

Merge Sort

Divide and Conquor

//1. divide collection(array) into two partitions (parts)
//2. sort individual partition seperately
//3. merge sorted partitions in single array to get sorted array

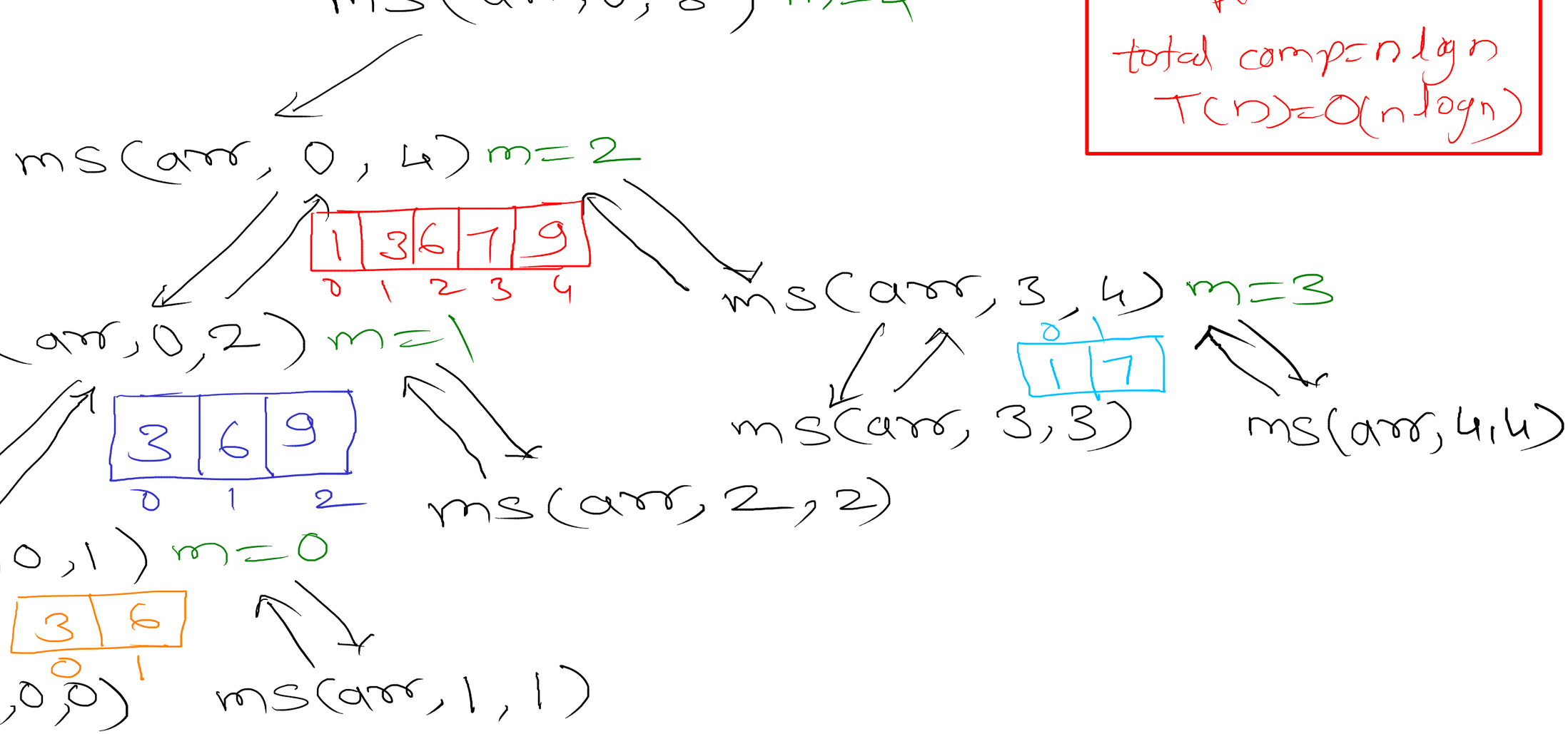
6	3	9	1	7	2	8	4	5
0	1	2	3	4	5	6	7	8

levels of division = log n

comparisions = n per level

total comp = n log n

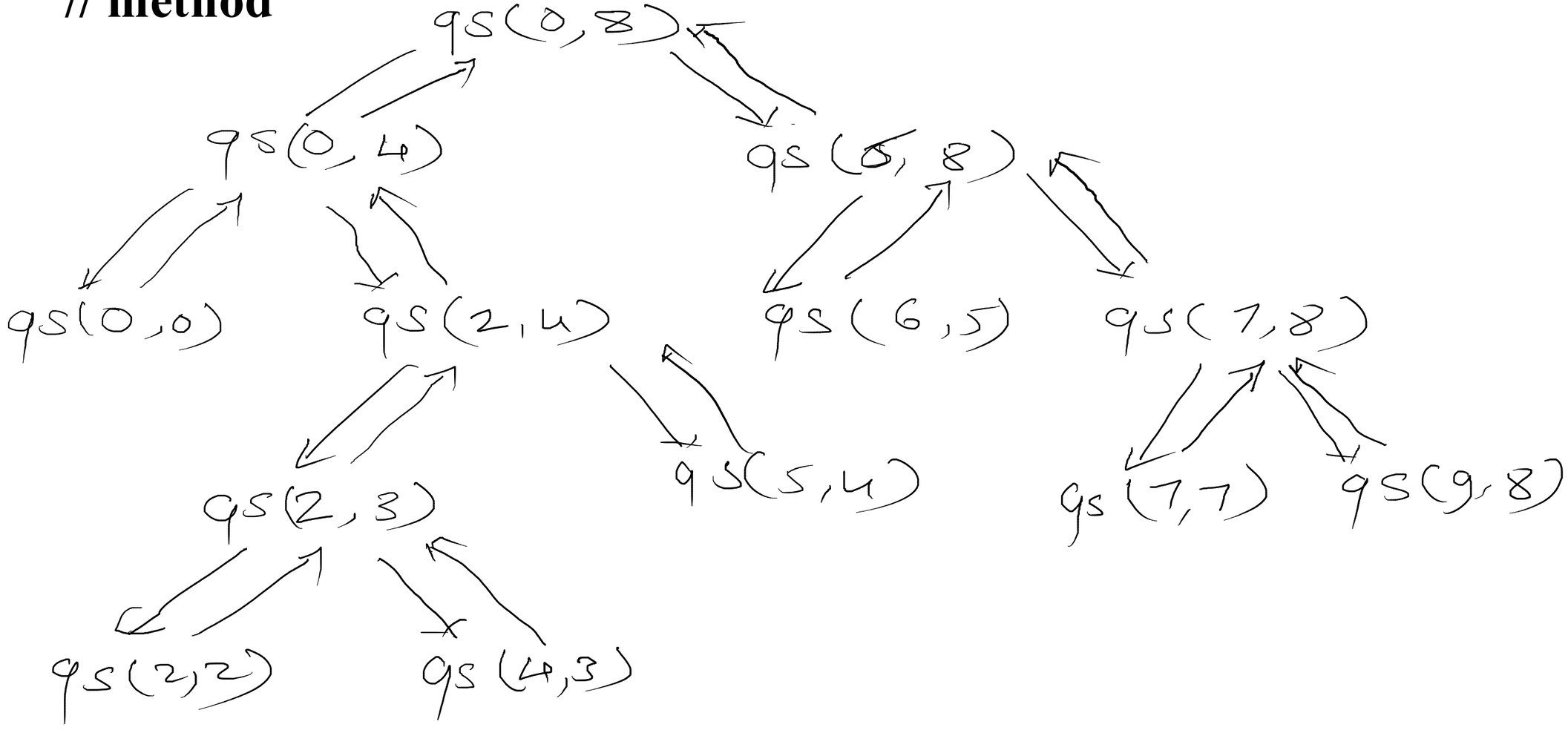
T(n) = O(n log n)



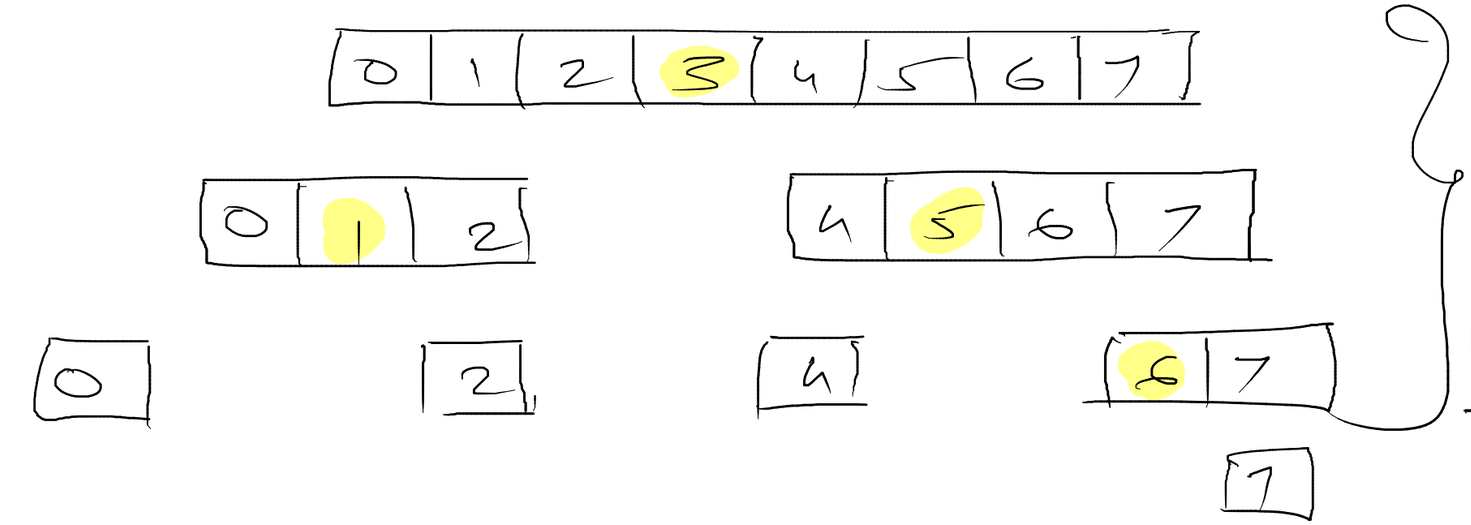
Quick Sort

Divide and Conquor

//1. select one referance/axis/pivot element from array
    //i. pivot = leftmost element
    //ii. pivot = rightmost element
    //iii. pivot = middle element
//2. arrange all lesser elements than pivot on left side of pivot
//3. arrange all greater elements than pivot on right side of pivot
//4. sort left and right partition of pivot individually by applying same
    // method



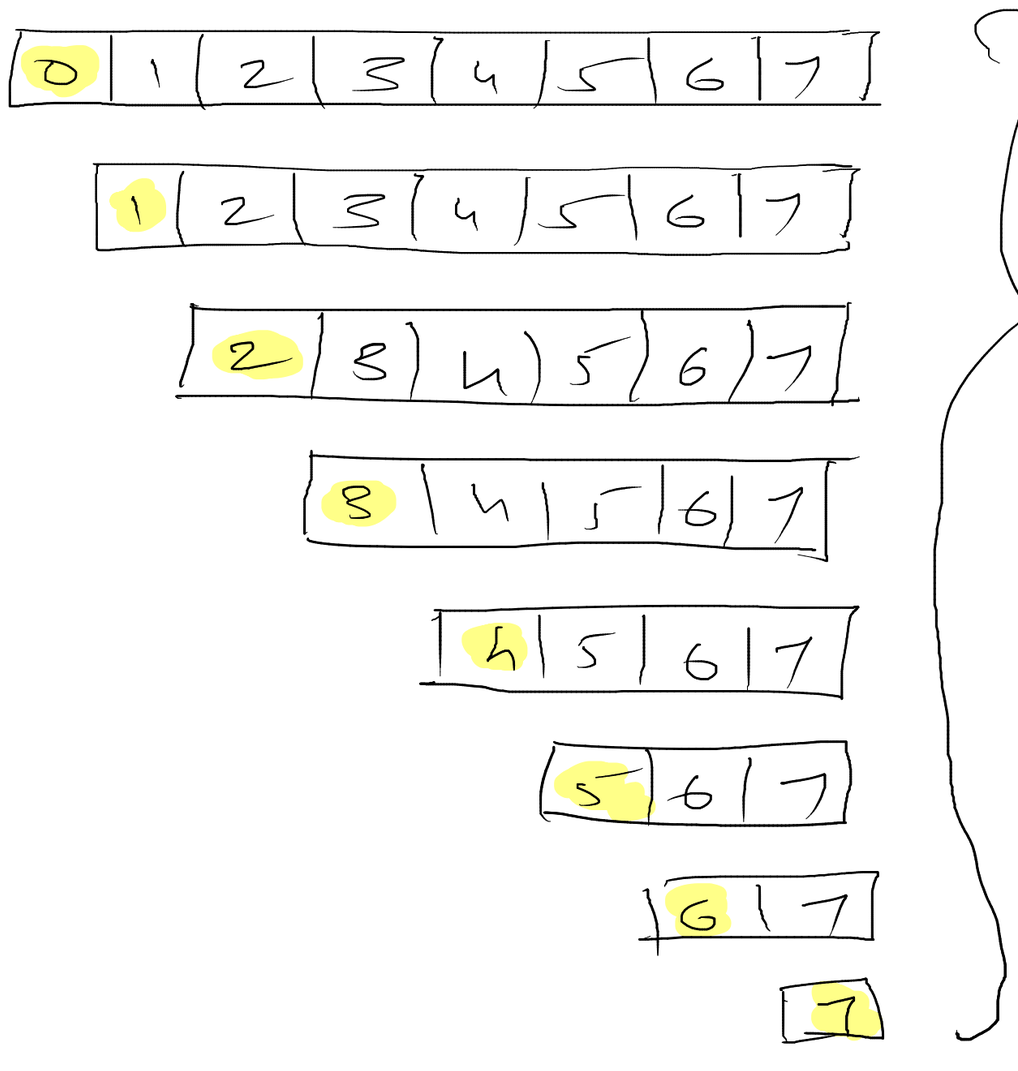
Quick Sort



levels = log n

comps ∝ n per level

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



levels = n

comp per level ∝ n

$T(n) = O(n^2)$

Time complexity of quick sort also depends on selection of pivot.

Pivot selection —
to improve performance
- dual pivot quick sort
- median of 3