Sunbeam Institute of Information Technology
Pune and Karad

Data structures and Algorithms

Trainer - Devendra Dhande

Email – devendra.dhande@sunbeaminfo.com

Heap sort

Algorithm :

1. create heap from given array.
   - max (Ascending order)
   - min (Descending order)
2. delete all elements from heap one by one and use vaccant locations of array from right side to keep delete elements.

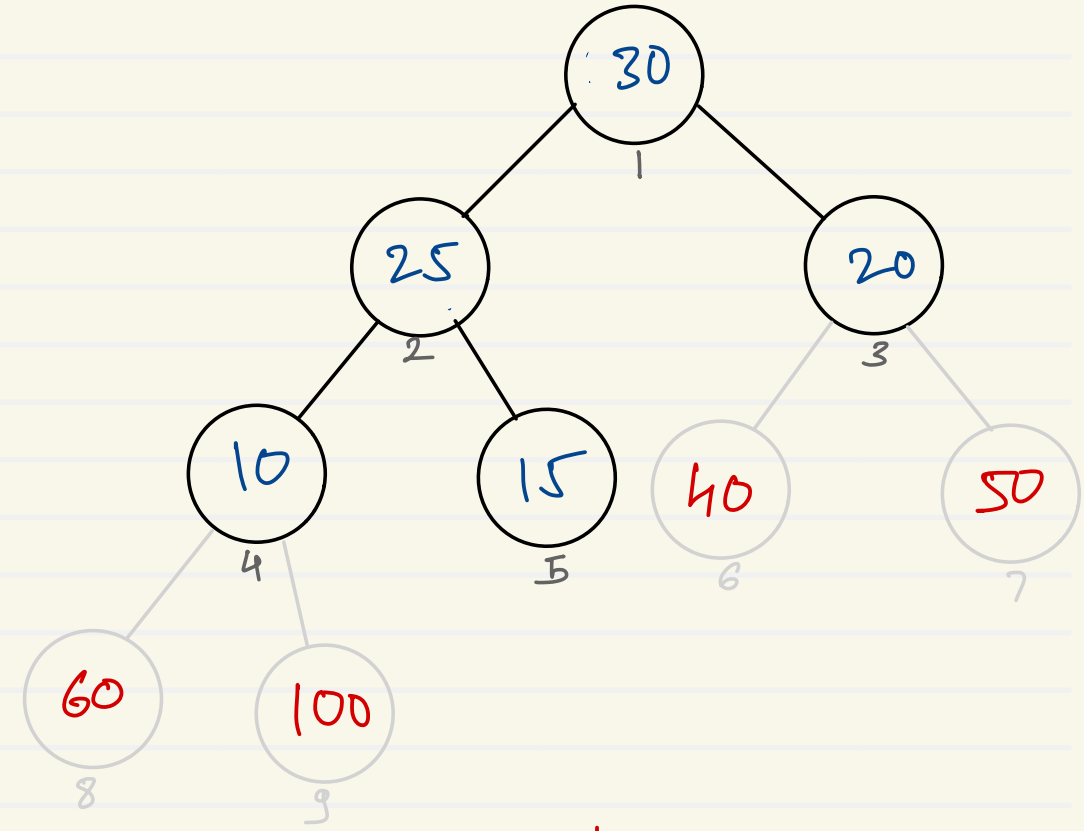
to create heap : $n \log n$

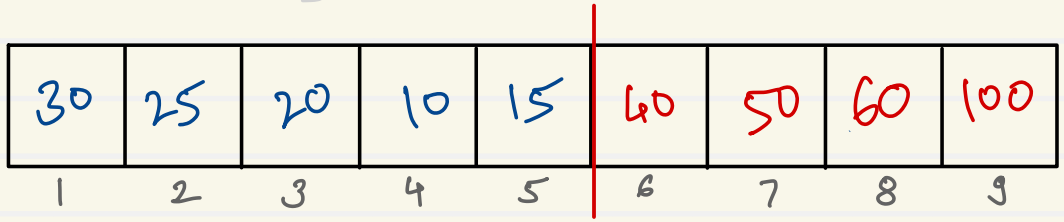
to delete heap : $n \log n$

$2 n \log n$

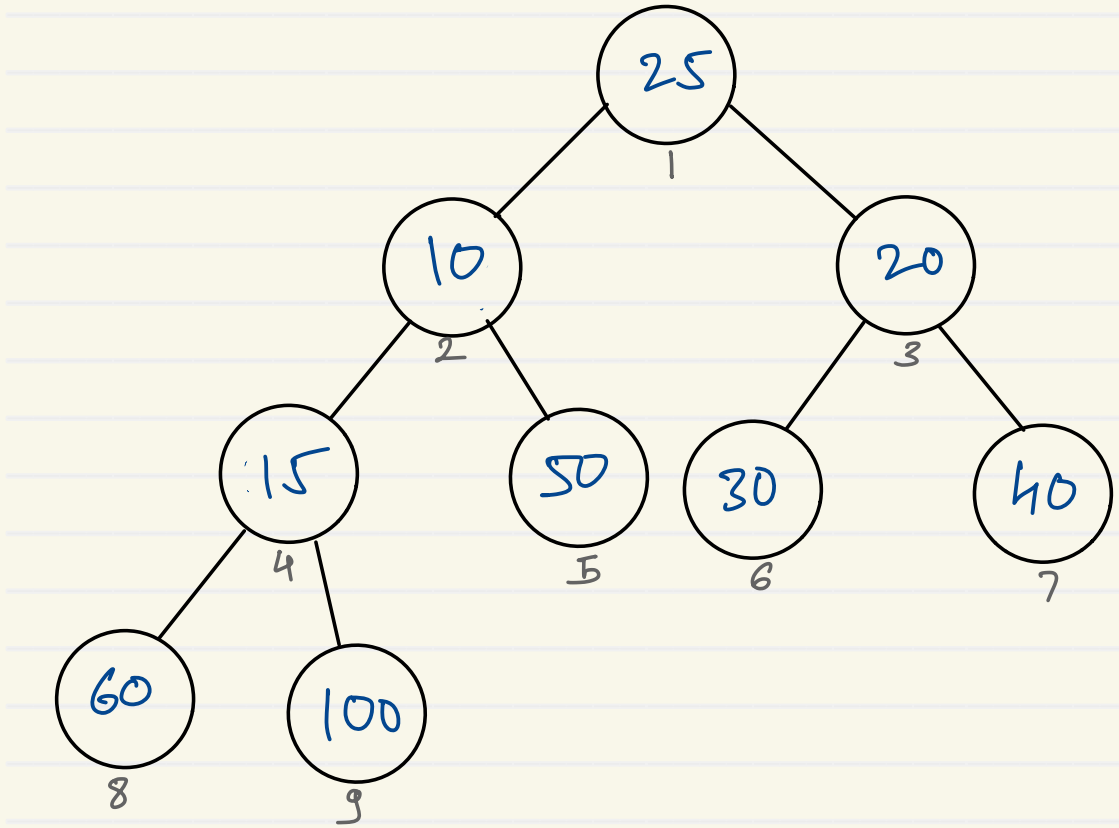
$T(n) = O(n \log n)$ Best Avg Worst

$S(n) = O(1)$



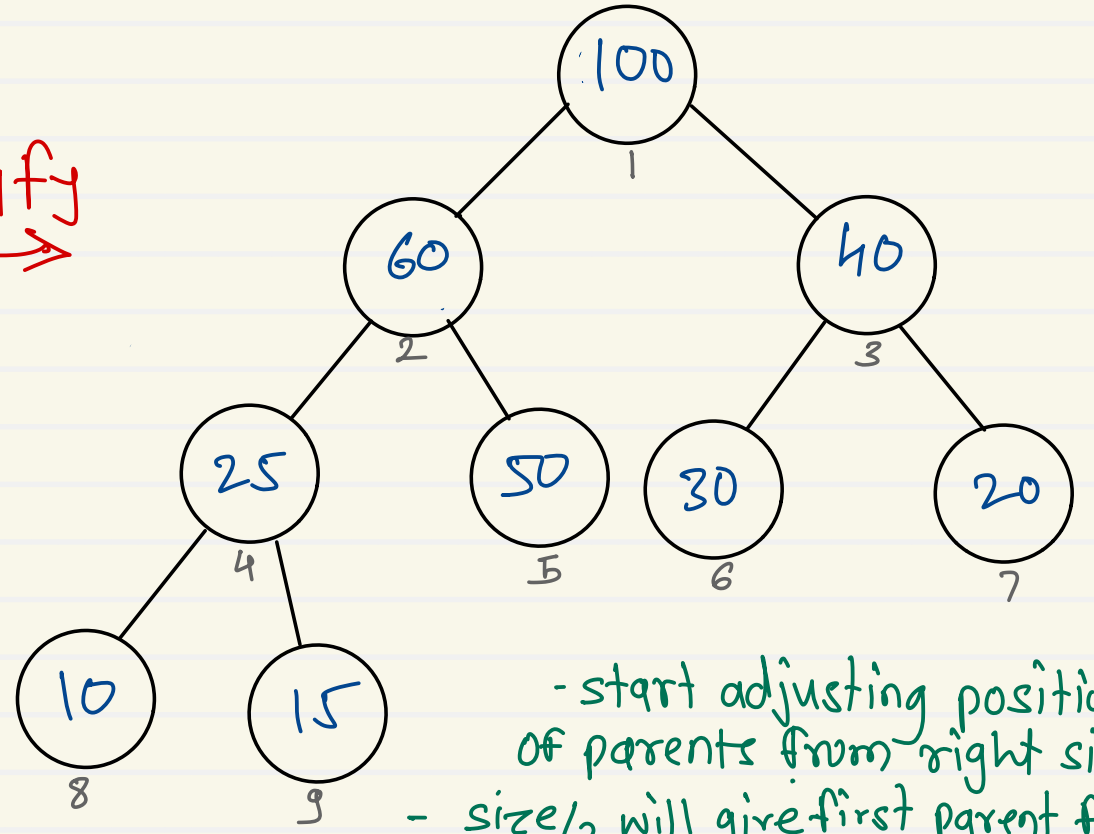


Heapify (to convert array into heap (max/min)) → T(n) = O(n)



25	10	20	15	50	30	40	60	100
1	2	3	4	5	6	7	8	9

Heapify →



100	60	40	25	50	30	20	10	15
1	2	3	4	5	6	7	8	9

- start adjusting position of parents from right side
- size/2 will give first parent from right

Merge sort

1. Divide array in two parts
2. Sort both partitions individually (by merge sort only)
3. Merge sorted partitions into temporary array
4. Overwrite temporary array into original array

$m = \frac{l + r}{2}$

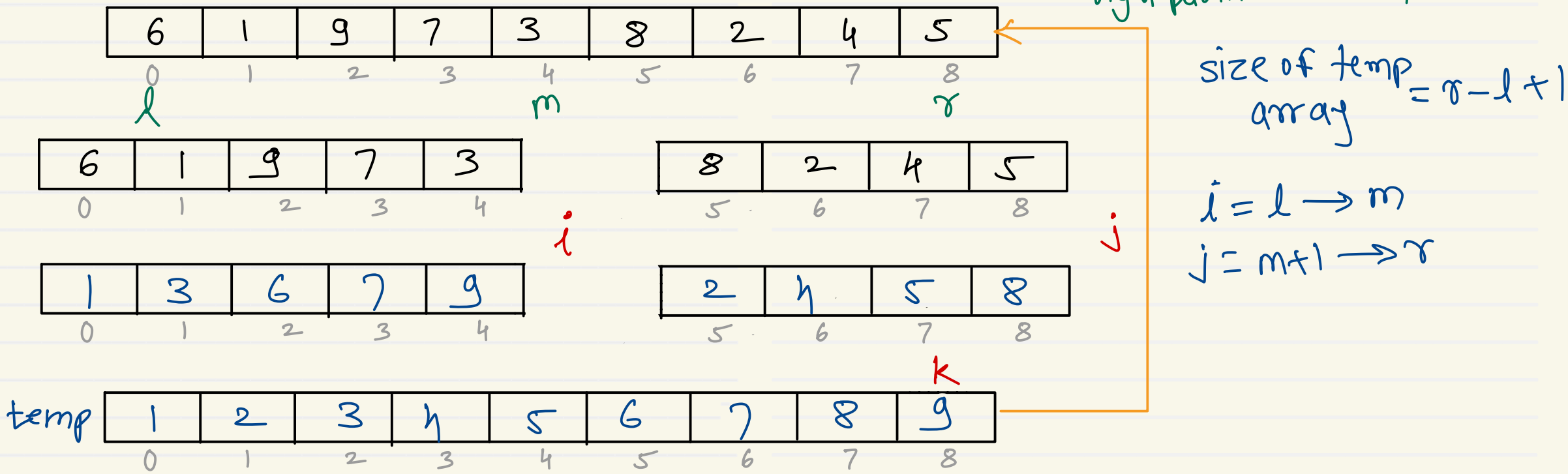
left partition : $l \rightarrow m$

right partition : $m+1 \rightarrow r$

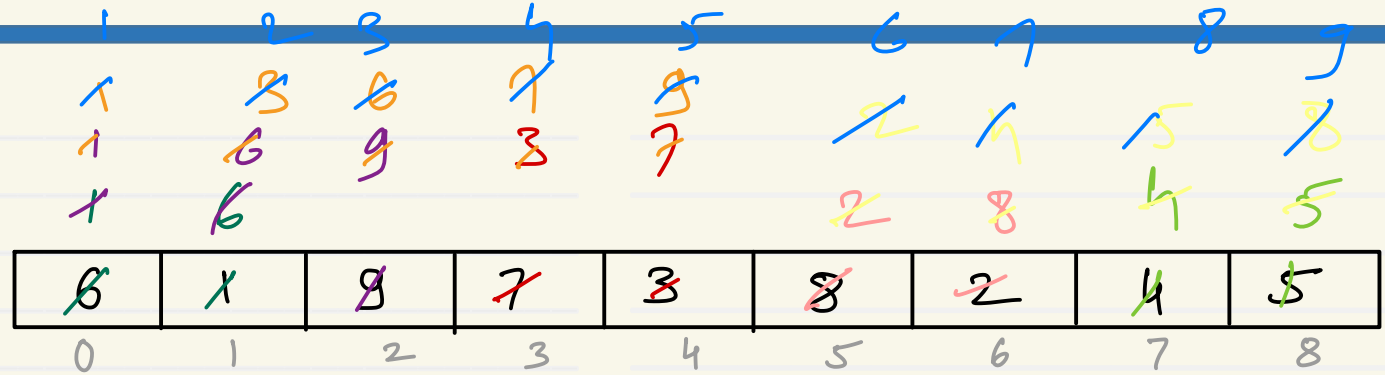
size of temp array $= r - l + 1$

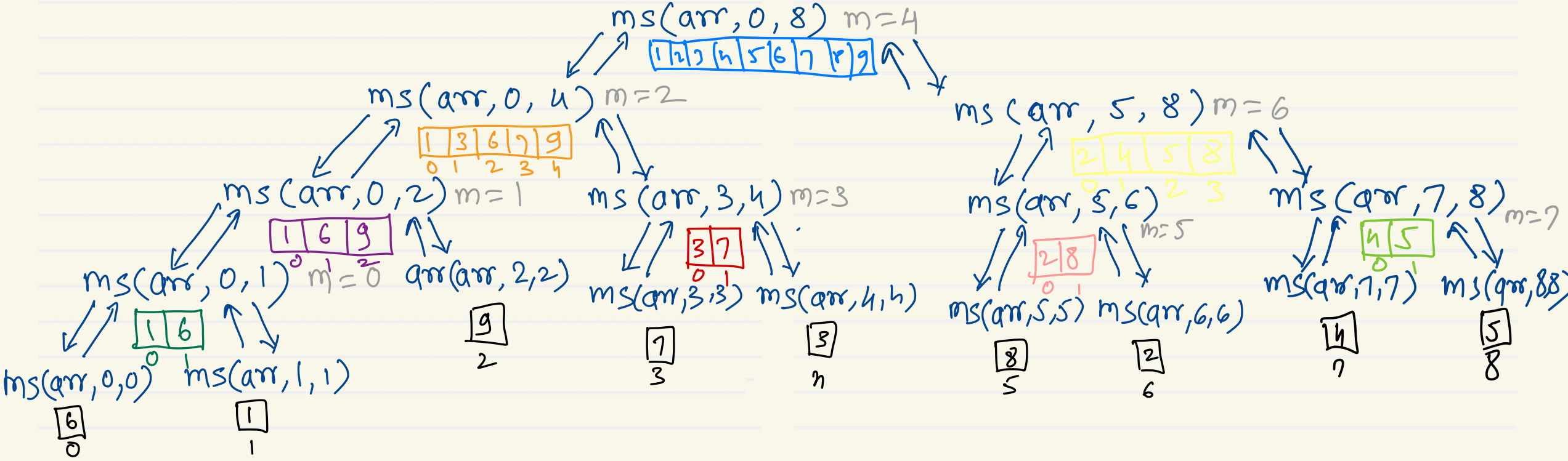
$i = l \rightarrow m$

$j = m+1 \rightarrow r$



Merge sort


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


ms(arr,0,8) m=4
ms(arr,0,4) m=2
ms(arr,0,2) m=1
ms(arr,0,1) m=0
ms(arr,0,0)
ms(arr,1,1)
arr(arr,2,2)
ms(arr,3,4) m=3
ms(arr,3,3)
ms(arr,4,4)
ms(arr,5,8) m=6
ms(arr,5,6) m=5
ms(arr,5,5)
ms(arr,6,6)
ms(arr,7,8) m=7
ms(arr,7,7)
ms(arr,8,8)

Merge sort

Number of elements = n

levels of division = log n

comps per level = n

Total comps = n log n

Time ∝ comps

Time ∝ n log n

Best
Avg
Worst

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

temp → extra space needed to merge two sorted partitions.

Auxiliary space ∝ n

$$S(n) = O(n)$$

- merge sort is out place sorting algo

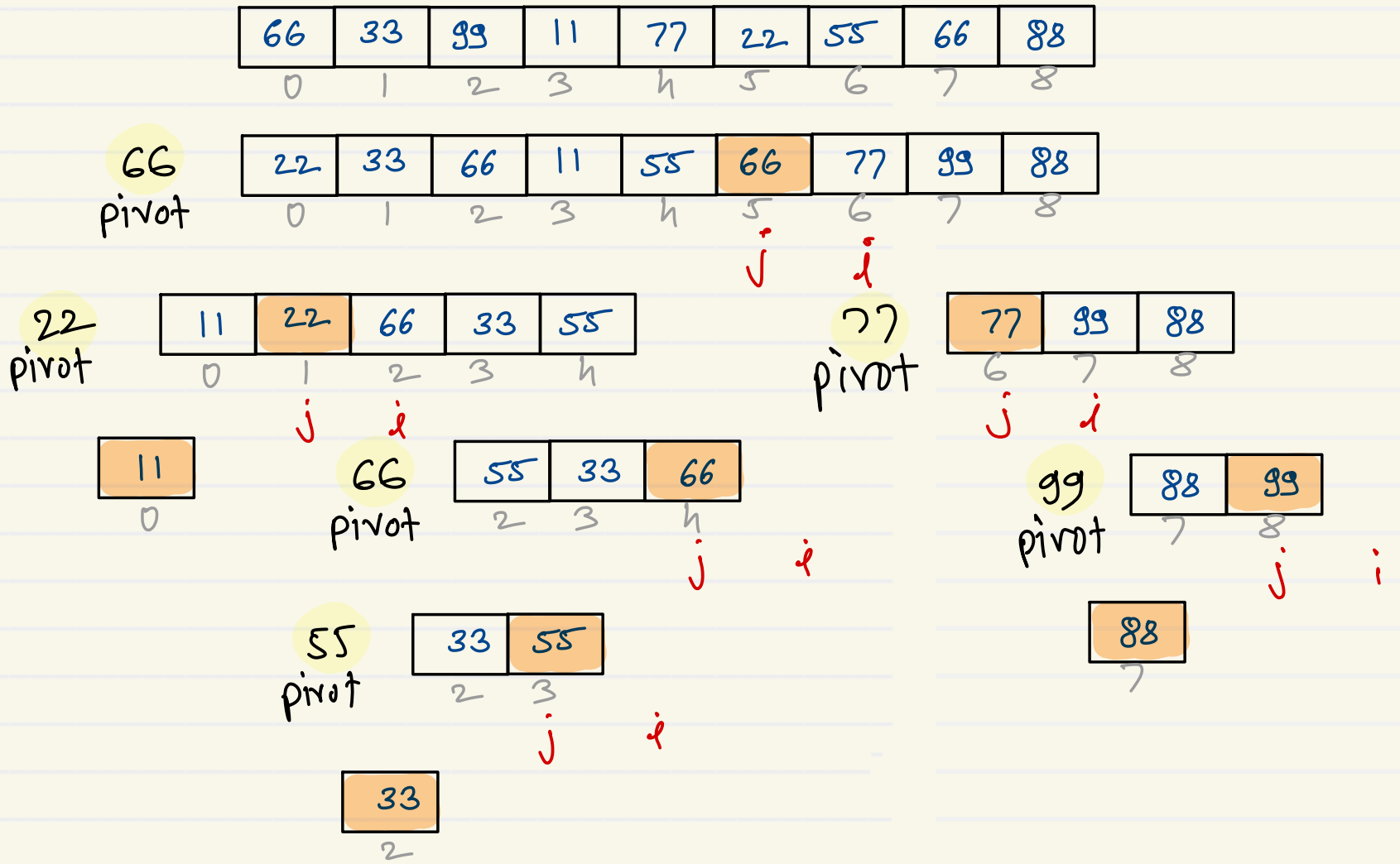
Quick sort

1. Select pivot/axis/reference element from array
2. Arrange lesser elements on left side of pivot
3. Arrange greater elements on right side of pivot
4. Sort left and right side of pivot again (by quick sort)

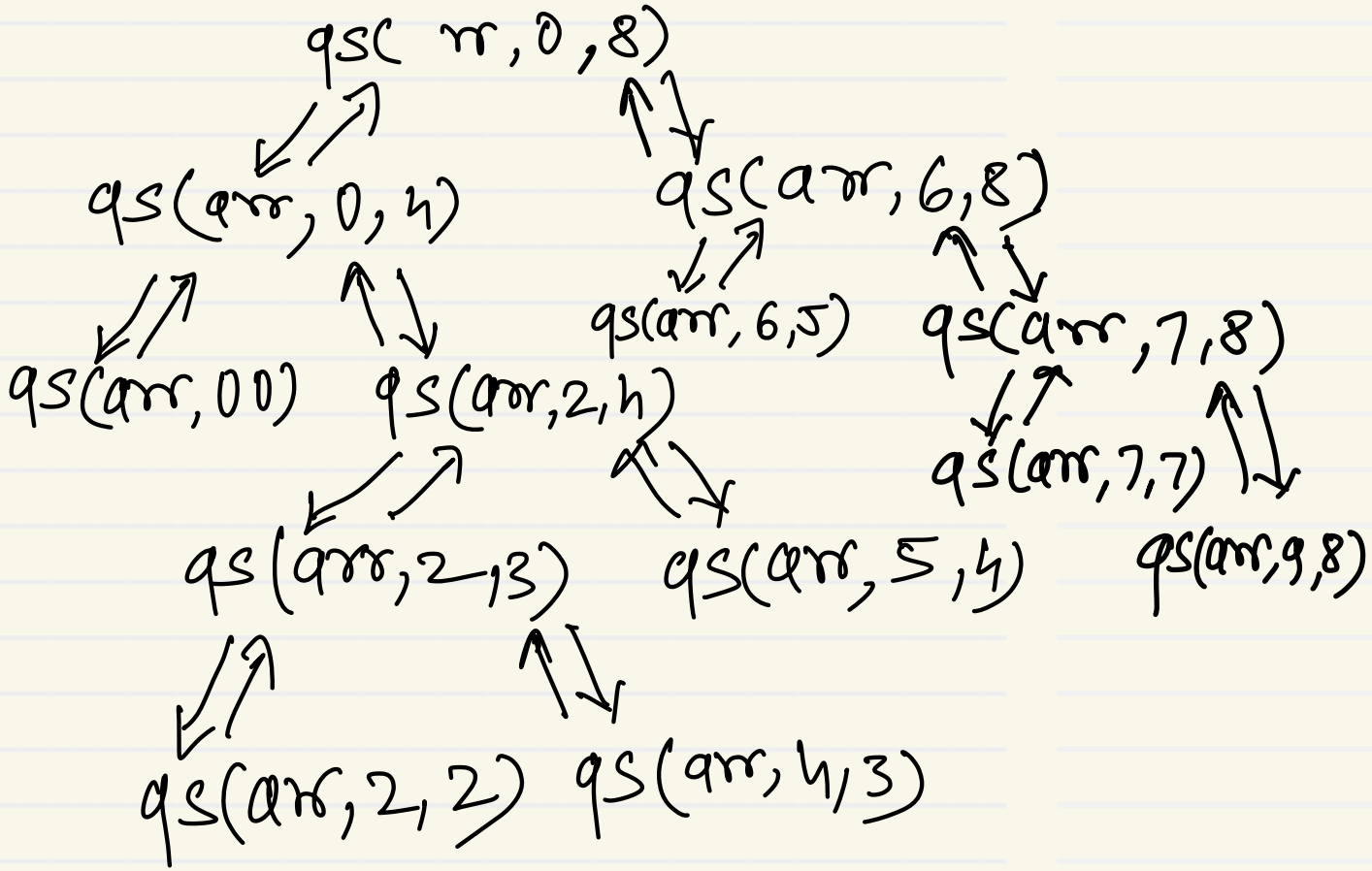
Selection pivot :

1. extreme left or right
2. middle element
3. median
   - random 3 element
   - random 5 element
   - random 7 element
4. dual pivot

Quick sort


66 33 99 11 77 22 55 66 88
0 1 2 3 4 5 6 7 8
66
pivot
22 33 66 11 55 66 77 99 88
0 1 2 3 4 5 6 7 8
j i
22
pivot
11 22 66 33 55
0 1 2 3 4
j i
11
0
66
pivot
55 33 66
2 3 4
j i
55
pivot
33 55
2 3
j i
33
2
77
pivot
77 99 88
6 7 8
j i
99
pivot
88 99
7 8
j i
88
7

Quick sort


qs(arr, 0, 8)
qs(arr, 0, 4)
qs(arr, 6, 8)
qs(arr, 0, 0)
qs(arr, 2, 4)
qs(arr, 6, 5)
qs(arr, 7, 8)
qs(arr, 2, 3)
qs(arr, 5, 4)
qs(arr, 7, 7)
qs(arr, 9, 8)
qs(arr, 2, 2)
qs(arr, 4, 3)