Lecture - 26

Height of some students are given:

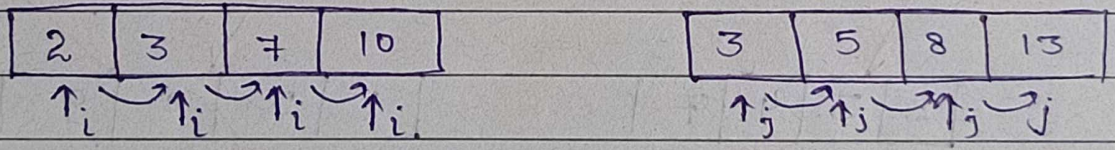
4 3 7 1 2 8 9

We have to arrang the student in ascending order by sorting.

previously, we learn Bubble sort, insertion sort, selection sort.

Now, we can sort this by a new sorting method called Merge sort.

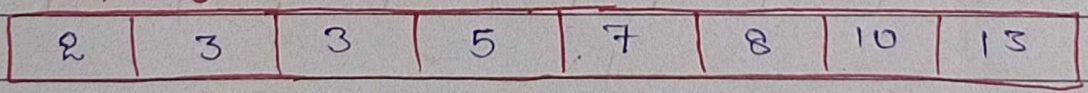
* Merge sort:



2 < 3 , 3 <= 3

3 < 7

Depth first search

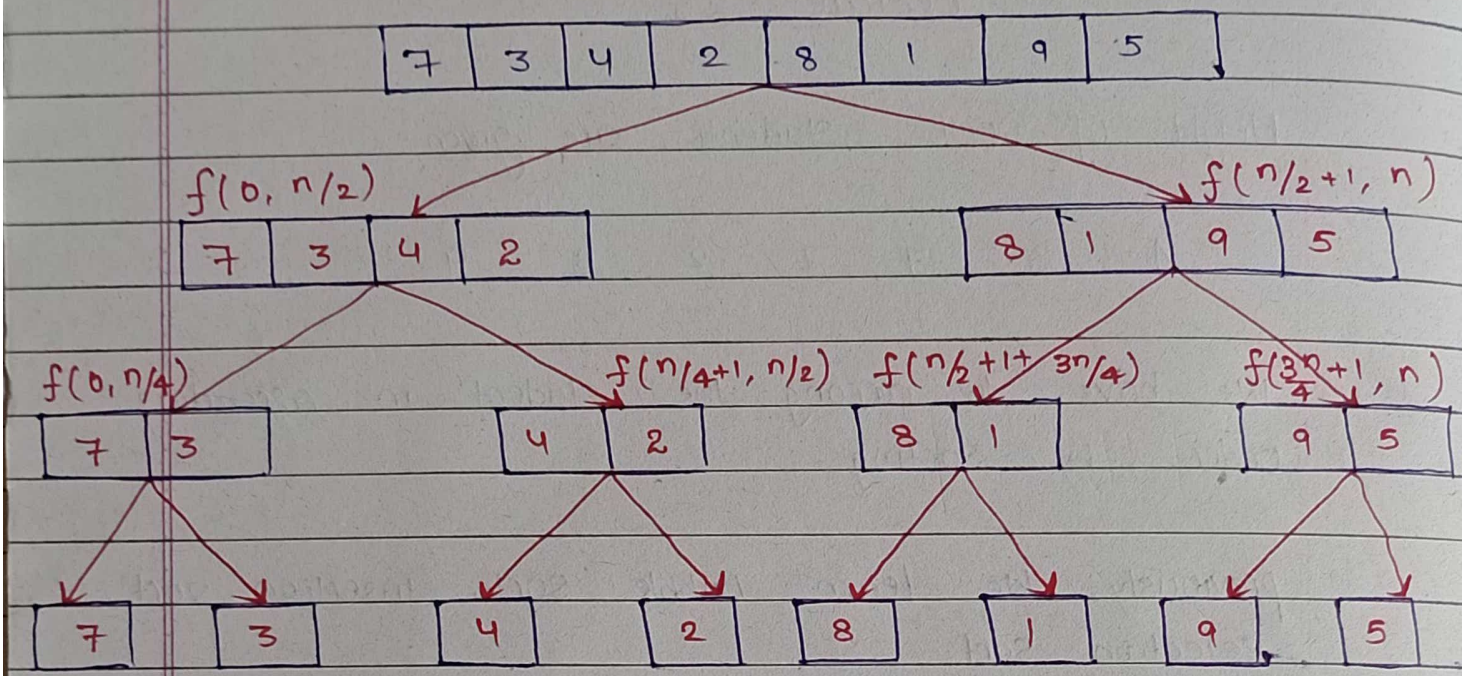


→ Divide the array in two parts of equal size.

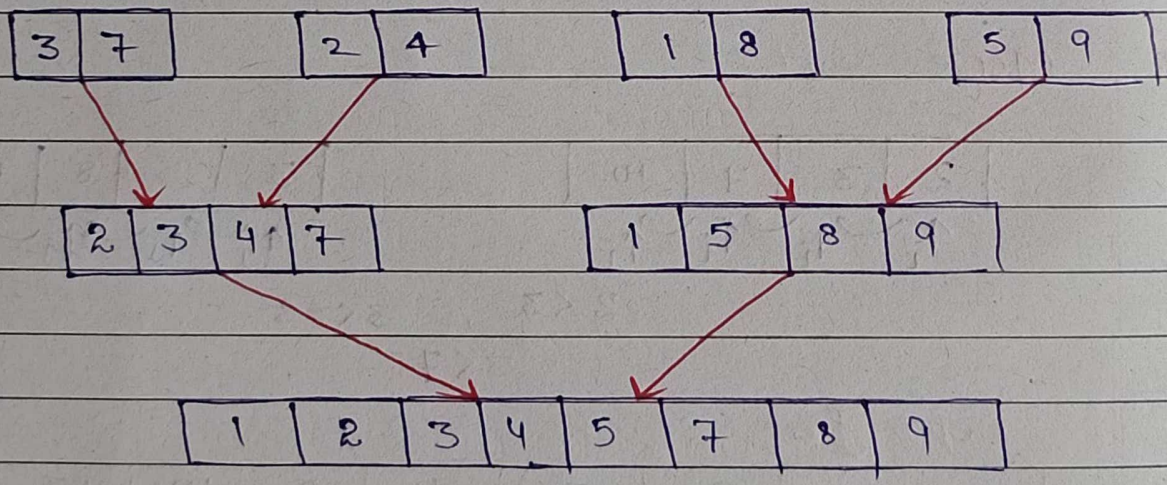
→ Divide it until at the last the size of array becomes 1.

→ Now, start Merging the single size array in sorted order and then Merge all upper step.

Teacher's Signature.....



Now, it can reverse the process with comparing and sorting.



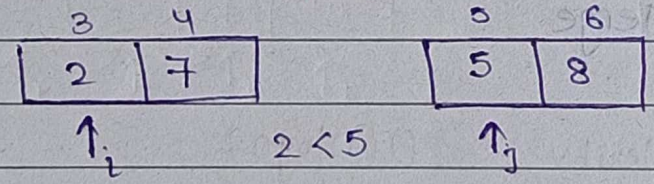
↳ This is sorted by Merge sort.

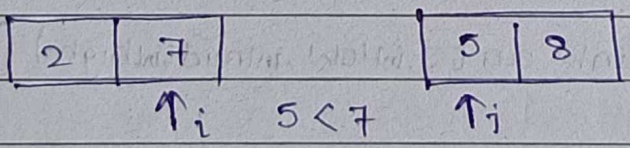
Teacher's Signature.....

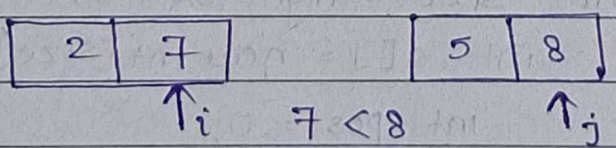
```
Merge_sort (int arr[], int low, int high) {

    if(high == low)
        return;

    mid = (low + high)/2;
    Merge_sort (arr, low, mid);
    Merge_sort (arr, mid+1, end);
     Merge_array (arr, low, mid, high);

}
```

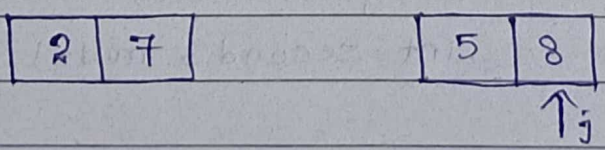
Merge_sort function can divide the array until the size of array becomes 1.

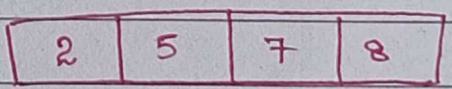
Merge_array can merge the after dividing with size = 1 and merge the element in sorted order.











↳ Final answer after sorting

Page No:
Date: / /

Size :

$n \rightarrow n/2^0$

$\downarrow$

$n/2 \rightarrow n/2^1$

$\downarrow$

$n/4 \rightarrow n/2^2$

$\downarrow$

$\vdots$

$\downarrow$

$n/2^k = 1$

$\frac{n}{2^k} = 1$

$n = 2^k$

$\boxed{k = \log_2 n}$

→ Merge sort can follow divide and conquer method.

η swap in each heigh.

height = logn.

Total Time = nlogn.
for Merge

⇒ Time Complexity : O(nlogn)

```
Merge_array (int arr[], intlow, intmid, inthigh) {

    int size = high - low + 1;
    int a[] = new int[size];
    int pos = 0;
    int first = low;
    int second = mid+1;

    while (first <= mid && second <= high) {
```

Teacher's Signature.....

```
if (arr[first] <= arr[second]) {
        a[pos] = arr[first];
            pos++;
            first++;
        }
    else {
        a[pos] = arr[second];
            pos++;
            second++;
        }
}
while (first <= mid) {
    a[pos] = arr[first];
        pos++;
        first++;
    }
while (second <= high) {
    a[pos] = arr[second];
        pos++;
        second++;
    }

for (int i=0; i<arr.length; i++) {
        arr[i] = a[i];
    }

    pos = 0, mid = low;
    while(mid <= high)
    arr[mid++] = a[pos++];
}
```

Teacher's Signature.....

Page No.:
Date: / /

Space Complexity:

At every step we make array
↳ O(n)

Stack size : log n

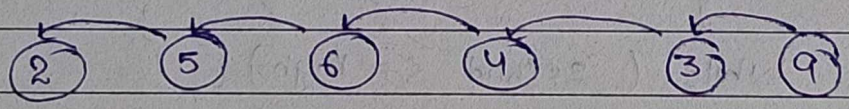
(n + log n)

* Quick Sort

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

Choose one element as pivot element

put the pivot element at Correct place.



2 3 4 5 6 9

↳ Now the array is sorted.

pivot element : Choose any element as pivot element. But we can organise in Correct way.

So, We choose last element as pivot element.

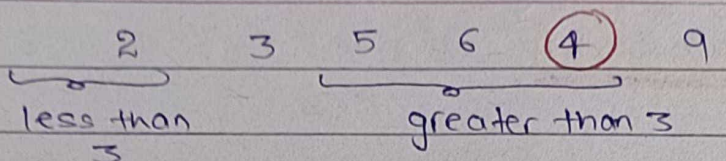
Example-1

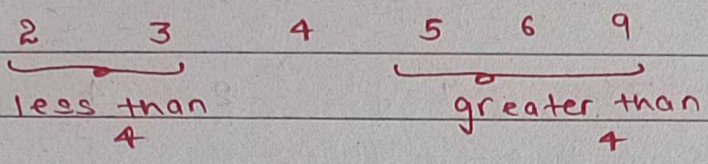
2 5 6 4 3 (9)

2 5 6 4 (3) 9

↑ pivot

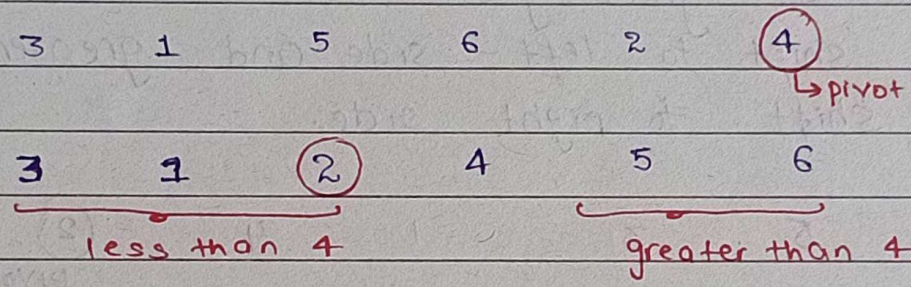
Teacher's Signature.....

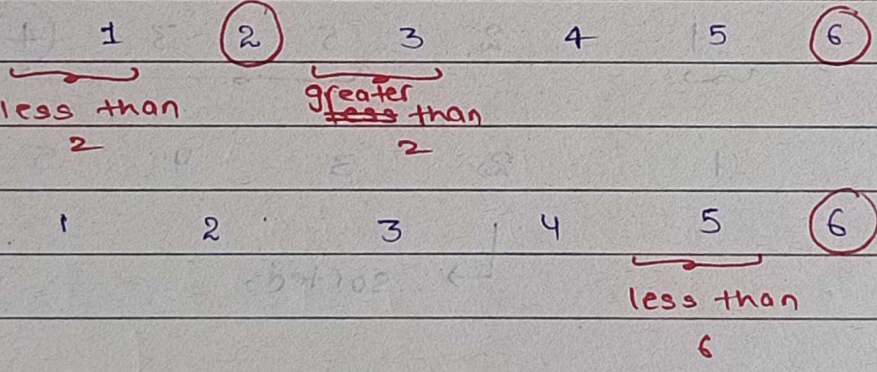




Now, Array is sorted.

Example-2





Now, Array is sorted.