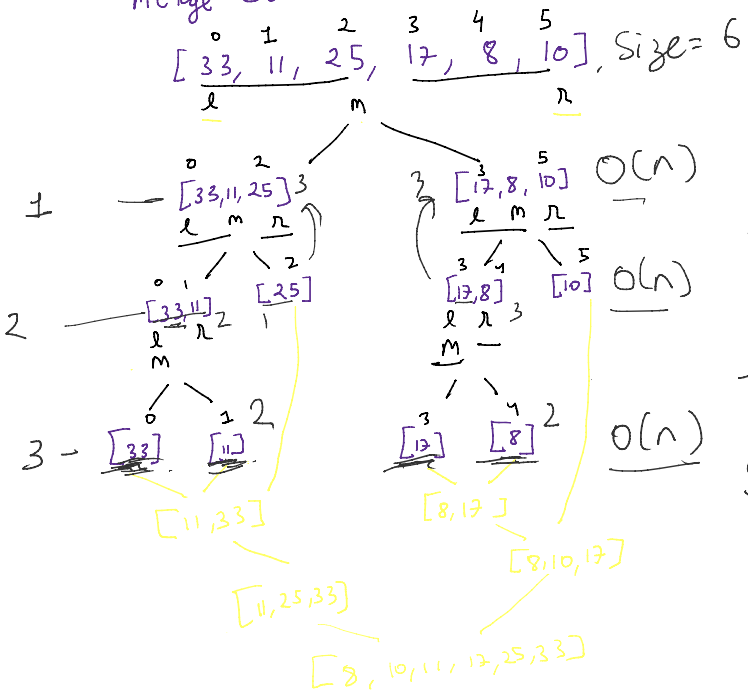
18th Feb

Merge Two Sorted Array :- [int[] merge(a1, a2)]

a = [2, 4, 6, 8] → c = [1, 2, 3, 4, 5, 6, 7, 8]

b = [1, 3, 5, 7]

Merge Sort :-



n = 6

$\log_2(n) = 2.\ldots$

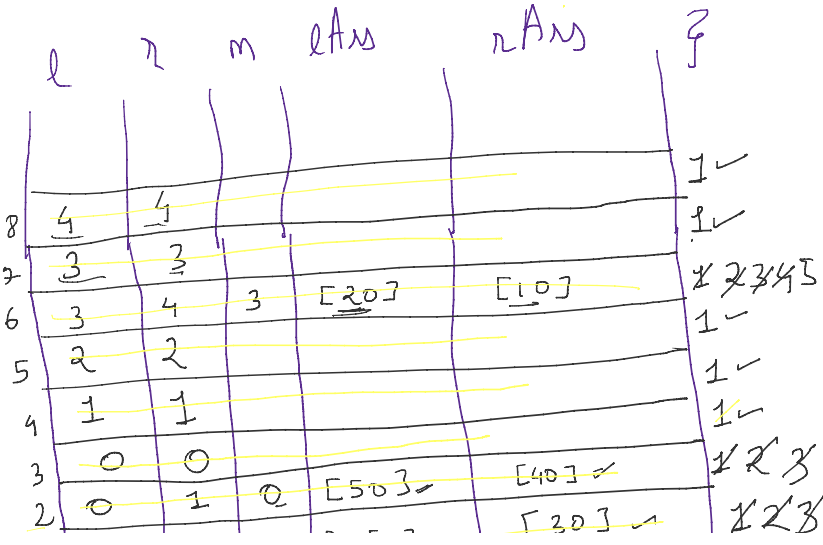
$2 \simeq 3$

T.C => $\log(n) \times (n) = O(n \log(n))$

S.C => $O(n)$

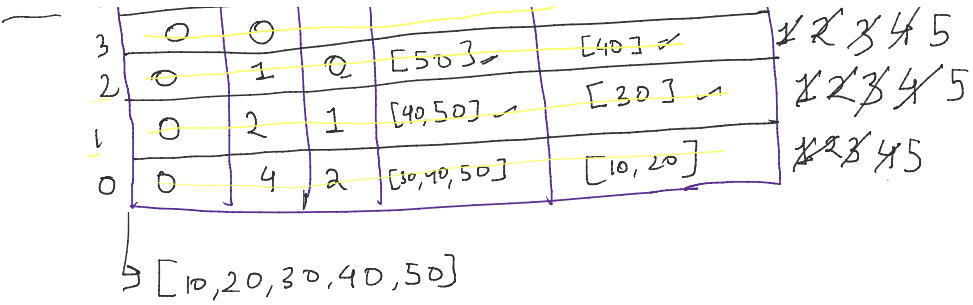
```
int[] mergeSort(arr, l, r){
if(l==r) { int[] res= new int[1];
           res[0] = arr[l];
           return res; }
```

```
mid = (l+r)/2;
lArr = mergeSort(arr, l, mid)
rArr = mergeSort(arr, mid+1, r)
res = merge(lArr, rArr),
return res
```



[50, 40, 30, 20, 10]

```
public static int[] mergeSort(int[] arr, int l, int r) {
1.  if (l == r) {...} return [l];
2.  int mid = (l + r) / 2;
3.  int[] lArr = mergeSort(arr, l, mid);
4.  int[] rArr = mergeSort(arr, mid + 1, r);
5.  return mergeTwoSortedArray(lArr, rArr);
}
```



Partion Index :-

arr = [9, 8, 6, 18, 33, 27]
0 1 2 3 4 5
i, j

Pivot = 27

=) Elements before 'i' are equal or smaller than pivot.

[9, 8, 33, 27, 6, 18, 10, 2, 3]

```
int PI(arr){
    i=0, j=0
    while(i < arr.length){
        if(arr[i] <= P){
            if(i != j) swap(arr, i, j)
            i++;
            j++;
        }else{
            i++;
        }
    }
    return j-1;
}
```

[6, 8, 9, 1, 33, 27]
0 1 2 3 4 5
P = 9
j i

```
public static int partitionIndex(int[] arr) {
    int pivot = arr[arr.length - 1];
    int i = 0, j = 0;
    while (i < arr.length) {
        if (arr[i] <= pivot) {
            if (i != j) swap(arr, i, j);
            j++;
        }
        i++;
    }
    return j - 1;
}
```

H.W =)

https://www.geeksforgeeks.org/count-number-of-occurrences-or-frequency-in-a-sorted-array/

https://www.geeksforgeeks.org/print-subsequences-string/