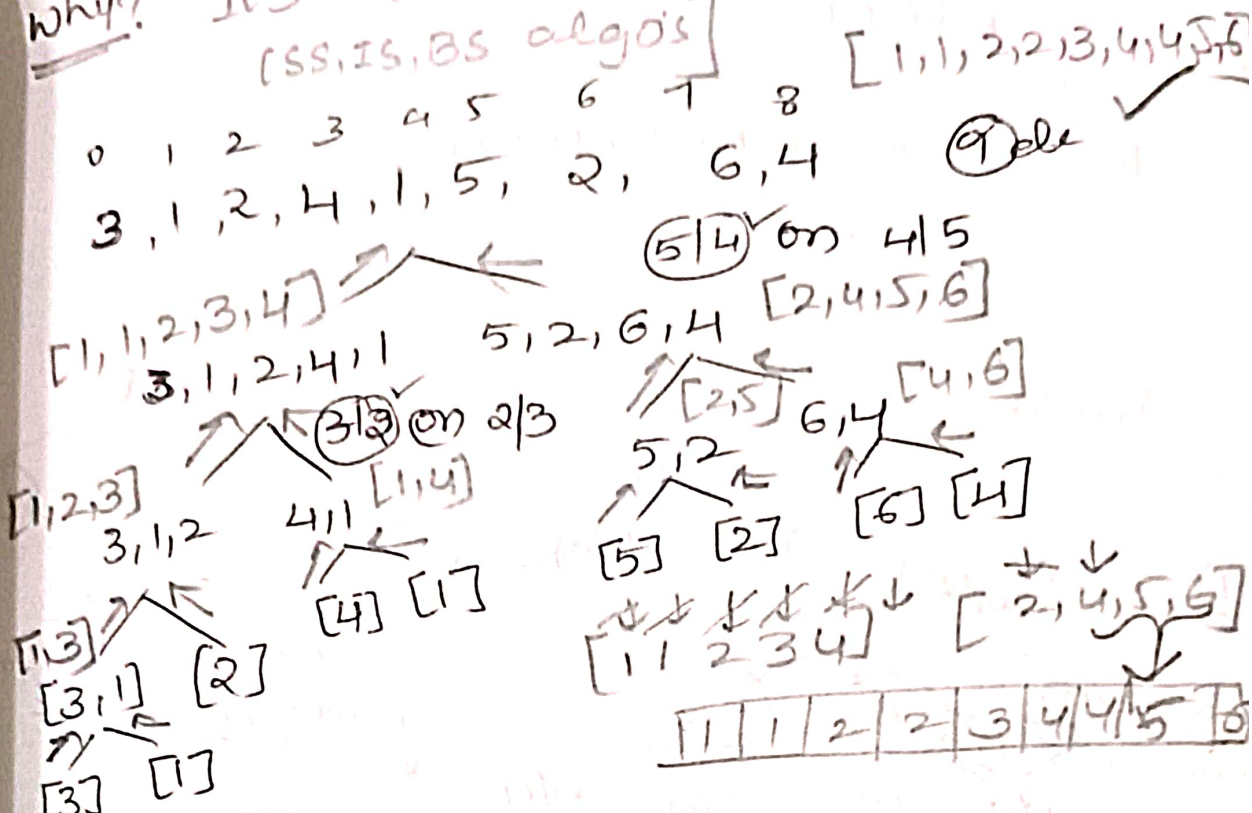
47 MergeSort: [divide n Merge]

Why? It's O(N) better than other O(N²) [SS, IS, BS algo's]



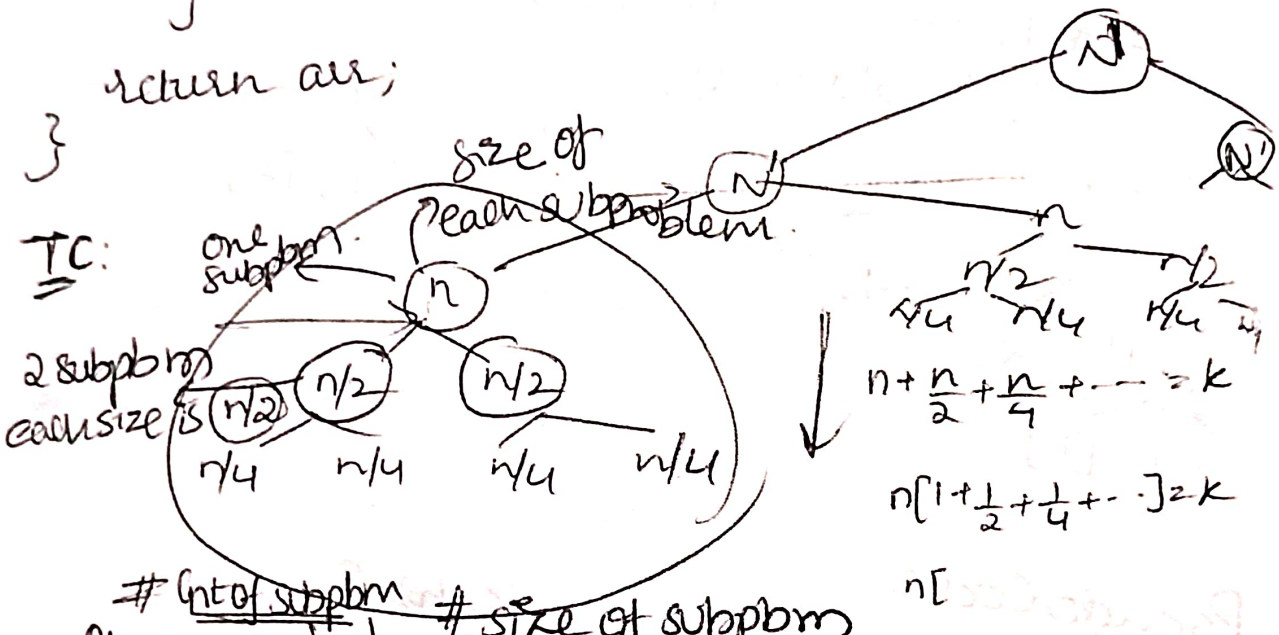
Pseudo Code:

```
mergeSort(arr, low, high) {
    if(low >= high) return;
        mid = low+high/2;
    mergeSort(arr, low, mid);
    mergeSort(arr, mid+1, high);
    merge(arr, low, mid, high);
}
```

```
int[] merge (int[] arr, int low, int mid, int high){
    List<Integer> temp = new ArrayList<>();
    int left = low, right = mid+1;
    while(left ≤ mid && right ≤ high){
        if (arr[left] ≤ arr[right]) { temp.add(arr[left]);
left++; }
else { temp.add(arr[right]); right++; }
    }
while(left≤mid) { temp.add(arr[left]) left++; }
```

```
while(right <= high) {
    temp.add(arr[right]);
    right++;
}
for(int i=low; i<=high; i++) {
    arr[i] = temp.get(i-low);
}
return arr;
}
```

TC:



	# Cnt of subpbm	# size of subpbm
At level 0	1	n
level 1	2	n/2 (∵ n/2 + n/2 = 1)
level 2	4	n/4 (n/4 + n/4 + n/4 + n/4 = 1)
:	:	:
level l	2^l	$n/2^l$

for $1 = n/2^L$ ← $2^L/2^L$

$\therefore n/2^L = 1$ ← at each level of recursion it adds upto 1

$n = 2^L$

$L = \log_2 n$

again, at each level it takes 'n' merges:

$\therefore O(L*n)$

1. n
2. $n/2 + n/2 = 2n/2 \rightarrow n$
3. $n/4 \times 4 = n$

$\therefore O(n \log_2 n)$

SC: O(N)