Q3.2) Let M(n) be the minimum number of comparisons needed to sort an array A with exactly n elements. For example, M(1)=0, M(2)=1, and M(4)=4. If n is an even number, clearly explain why M(n) = 2M(n/2) + n/2.

Merge Sort Algorithm uses divide and conquer to sort the array using the following steps-

- It divides the given unsorted array into two halves- left and right sub arrays.
- The sub arrays are divided recursively.
- This division continues until the size of each sub array becomes 1.
- After each sub array contains only a single element, each sub array is sorted trivially.
- The merge procedure combines these trivially sorted arrays to produce a final sorted array.
- Merge procedure is when we merge two sorted sub-arrays, we only compare the left-most element of each sub-array, since one of these two elements is guaranteed to be the smallest. We then repeat the process until one of the two sub-arrays is empty. Then there is nothing left to compare, and we will have our desired merged array.

Given, M(n) be the minimum number of comparisons needed to sort an array A with exactly n elements, and n is an even number.

Let M*(n)* be a function telling us the number of comparisons necessary to mergeSort an array with *n* elements. Also, all the comparisons take place in the merge method. If we are trying to merge two sorted lists, every time we compare two elements from the lists we will put one in its correct position. When we run out of the elements in one of the lists, we put the remaining elements into the last remaining slots of the sorted list. As a result, merging two lists which have a total of *n* elements requires at most *n-1* comparisons.

As we noted above, we break the list in half, mergeSort each half, and then merge the two pieces. Thus the total amount of comparisons needed are the number of comparisons to mergeSort each half plus the number of comparisons necessary to merge the two halves.

Now, in each step in merge sort, we divide the array into two (nearly) equal halves and solve them recursively using merge sort only. So now we can think of the

number of comparisons in mergeSort on an n-element subarray as being the sum of twice the number of comparisons in mergeSort on an (n/2)-element subarray.
So, we have,

$$M(n_L/2) + M(n_R/2) = 2M(n/2)$$

where,

n_L = Left half of array
n_R = Right half of array
$n_L = n_R$ (n is even)

Now dividing the problem into two halves every time will have two recursive calls on n/2 elements each. Each of these two recursive calls makes twice of the number of comparisons in mergeSort on an (n/4)-element subarray (because we have to halve n/2) plus n/2 to merge, and so the total time we spend merging for subproblems of size n/2 is 2*n/2 = n.

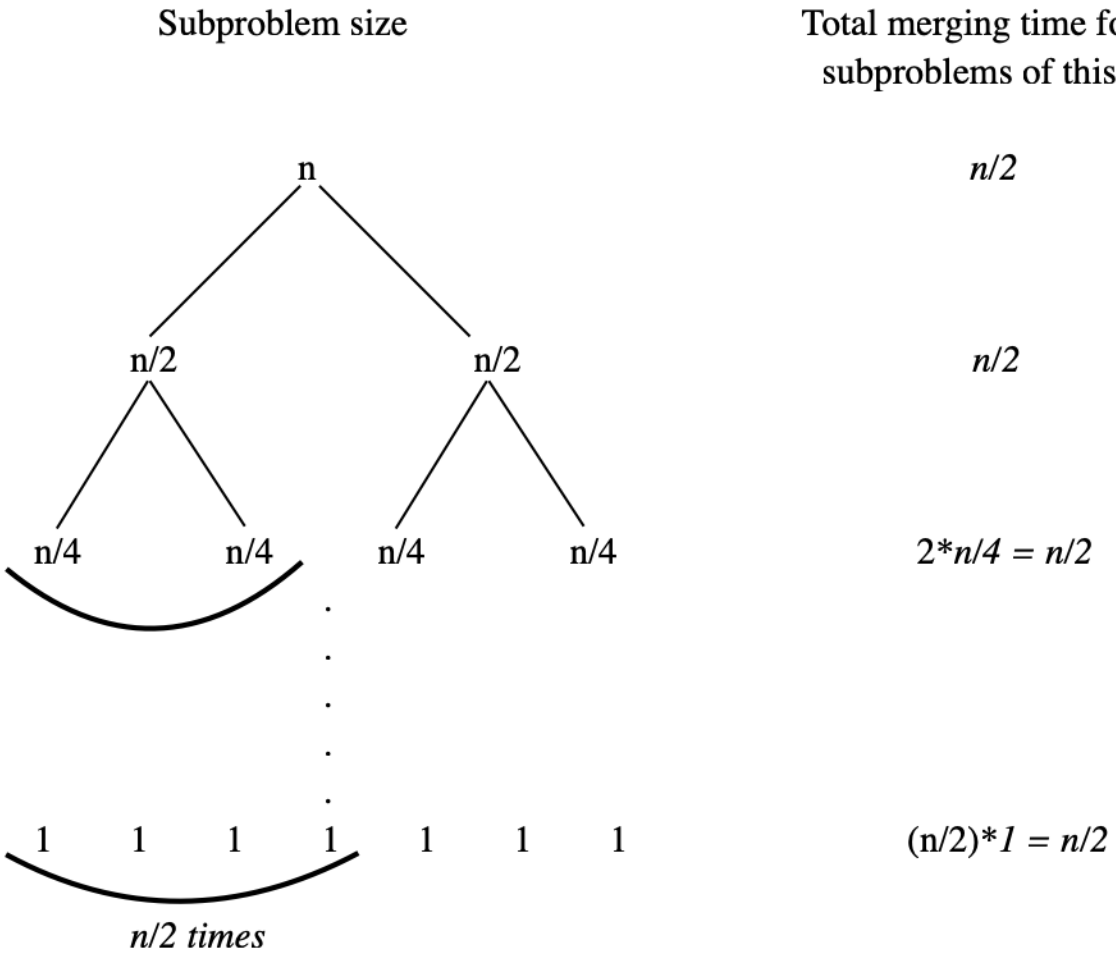
However, mergeSort will have **minimum number of comparisons** when the largest element of one sorted sub-list is smaller than the first element of its opposing sub-list, for every merge step that occurs. Only one element from the opposing list is compared, which reduces the number of comparisons in each merge step from n to **n/2**.

So, finally the minimum number of comparisons at each step will be, n/2.

Subproblem size	Total merging time for all subproblems of this size
n	*n/2*
n/2 n/2	*n/2*

As the sub-problems get smaller, the number of sub-problems doubles at each "level" of the recursion, but only one element from the opposing list is compared,

which reduces the number of comparisons in each merge step from n to **n/2**. So, the minimum number of comparisons is *n/2* at each level of recursion.



So, minimum number of comparisons in mergeSort will be,

$$M(n) = 2M(n/2) + n/2$$

Looking at the mergeSort algorithm, no comparisons are necessary when the size of the array is 0 or 1. Thus, M(0) = M(1) = 0. Solving this for different values of n, where n is an even number, we get,

$$n = 1$$
$$M(1) = 0$$

$$n = 2$$

$$M(2) = 2*M(2/2) + 2/2 = 2*M(1) + 1 = 2*0 + 1 = 1 \text{ comparison}$$

$$n = 4$$
$$M(4) = 2*M(4/2) + 4/2 = 2*M(2) + 2 = 2*1 + 2 = 4 \text{ comparisons}$$

$$n = 8$$
$$M(8) = 2*M(8/2) + 8/2 = 2*M(4) + 2 = 2*4 + 4 = 12 \text{ comparisons}$$

$$n = 16$$
$$M(16) = 2*M(16/2) + 16/2 = 2*M(8) + 2 = 2*12 + 8 = 24 \text{ comparisons}$$

So, when $n = 2^k$ will have $k*2^{k-1}$ minimum number of comparisons.

Hence, minimum number of comparisons in mergeSort will be,

$$\boldsymbol{M(n) = 2M(n/2) + n/2}$$