Name: B M RAUF SEC: 20 SPRING'24
ID: 22201782 CSE221

Task01

I implemented the merge-sort algorithm to sort the array. By using recursion, ~~we~~ I kept dividing the array until I reach the base case (len(arr) == 1) and then conquer the array by comparing left and right elements and append the smallest element to the array. Finally, returned the array once the sorting is done.

Task 02

I use divide and conquer algorithm to solve this problem. By using recursion, I kept dividing the array until I reach the base case and ~~the~~ I kept the maximum value from both ends and compared which one was the largest. Finally, I returned the maximum value. The overall time complexity of the code is $O(n)$.

Task03

I use divide and conquer algorith to solve this problem. By using recursion, I sort the array and while sorting the array if i-th element of the left array is greater than j-th element of the right array, I increase the counter len(left)-i times because if i-th element is greater than j-th element, all the elements of the left array after i-th element will be greater than j-th element of the right array.

Task04

I recursively divide the array into 2 parts and tried to find maximum finding from each array. Finally, returned the maximum ~~among~~ sum among the leftsum, righsum & leftmax + right max. Overall, time complexity of the code O(nlogn)

Task 05

I solve this problem using quick sort which recursively divide the array and sorts its subarray. In the partition step I select ~~e~~ a last element as pivot and then rearranges the array such that elements smaller than pivot are on the left and elements greater are on the right.

Task06

I use the parition method of quick sort to find and return the smallest element at a given index. It 1st checks the base case and return the element if the sub array has only one elements and it matches the given index. It then performs partitioning ~~using~~ and based on the pivot's position, it either returns the wanted elements or continues searching in the left or right of the subarray, by using recursion.