MERGE SORT

Merge sort is a sorting algorithm that follows the divide-and-conquer approach. It works by recursively dividing the input array into smaller subarrays and sorting those subarrays then merging them back together to obtain the sorted array.

Advantages of Merge Sort:

1.Stability: Merge sort is a stable sorting algorithm, which means it maintains the relative order of equal elements in the input array.

2.Simple to implement: The divide-and-conquer approach is straightforward.

Disadvantage of Merge Sort:

1.Space complexity: Merge sort requires additional memory to store the merged sub-arrays during the sorting process.

2.Not in-place: Merge sort is not an in-place sorting algorithm, which means it requires additional memory to store the sorted data. This can be a disadvantage in applications where memory usage is a concern.

```
def merge_sort(a,n):
  if n > 1:
    m=n // 2
    # divide the list in two sub lists
    l1=a[:m]
    n1=len(l1)
    l2=a[m:]
    n2=len(l2)
    # reccursively calling the function for sub lists
    merge_sort(l1,n1)
    merge_sort(l2,n2)
    i = j = k = 0
    while i < n1 and j < n2:
      if l1[i] <= l2[j]:
        a[k] = l1[i]
        i = i + 1
      else:
        a[k]=l2[j]
        j = j + 1
      k = k + 1
    while i < n1:
      a[k] = l1[i]
      i = i + 1
      k = k + 1
    while j < n2:
      a[k]=l2[j]
      j = j + 1
      k = k + 1

a = [10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0]
n = len(a)
print("Array before Sorting")
print(a)
merge_sort(a, n)
print("Array after Sorting")
print(a)
```

```
Array before Sorting
[10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0]
Array after Sorting
[0, 10, 14, 19, 26, 27, 31, 33, 35, 42, 44]
```