|  |  |
| --- | --- |
|  | DEPARTMENT OF ARTIFICIAL INTELLIGNECE & DATA SCIENCE |

|  |  |
| --- | --- |
| Subject: Analysis of Algorithm | Course Code: CSC402 |
| Semester: 4 | Course: AI & DS |
| Laboratory No: 205 | Name of Subject Teacher: Pramod Bhavarthe |
| Name of Student: Sahil Shaikh | Roll Id: VU2S2223012 |

|  |  |
| --- | --- |
| Title of Practical | Implement Merge sort. |

**Theory –**

**Merge sort**is defined as a sorting algorithm that works by dividing an array into smaller subarrays, sorting each subarray, and then merging the sorted subarrays back together to form the final sorted array.

### Need for Merge Sort

* Merge sort has a time complexity of O(n log n), which makes it efficient for sorting large arrays relatively quickly.
* It is a stable sort, which means that the order of elements with equal values is preserved during the sort.
* Merge sort is a popular choice for sorting large datasets because of its efficiency and ease of implementation.
* It can be used in combination with other algorithms, such as quicksort, to improve the overall performance of a sorting routine.

**Algorithm:**

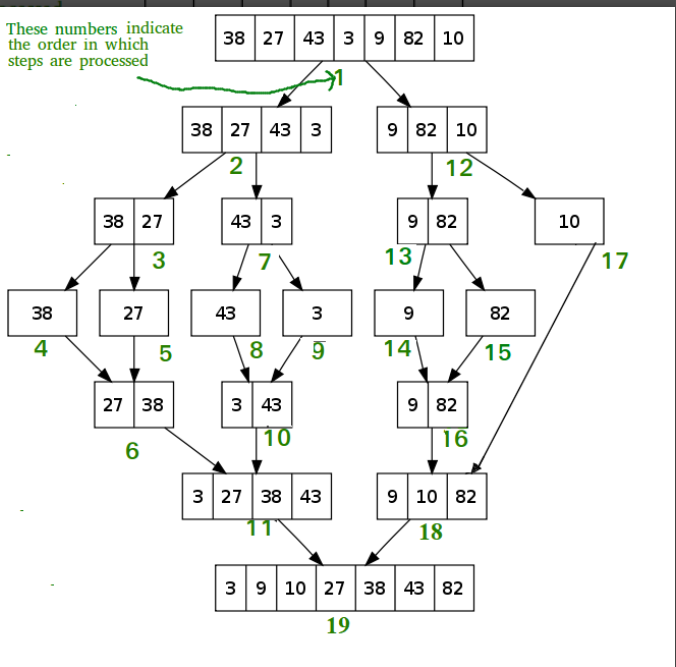
step 1: start

step 2: declare array and left, right, mid variable

step 3: perform merge function.  
    if left > right  
        return  
    mid= (left+right)/2  
    mergesort(array, left, mid)  
    mergesort(array, mid+1, right)  
    merge(array, left, mid, right)

step 4: Stop

## **Merge Sort Working Process:**

****

**Program –**

def mergeSort(arr):

    if len(arr) > 1:

        # Finding the mid of the array

        mid = len(arr)//2

        # Dividing the array elements

        L = arr[:mid]

        # into 2 halves

        R = arr[mid:]

        # Sorting the first half

        mergeSort(L)

        # Sorting the second half

        mergeSort(R)

        i = j = k = 0

        # Copy data to temp arrays L[] and R[]

        while i < len(L) and j < len(R):

            if L[i] <= R[j]:

                arr[k] = L[i]

                i += 1

            else:

                arr[k] = R[j]

                j += 1

            k += 1

        # Checking if any element was left

        while i < len(L):

            arr[k] = L[i]

            i += 1

            k += 1

        while j < len(R):

            arr[k] = R[j]

            j += 1

            k += 1

# Code to print the list

def printList(arr):

    for i in range(len(arr)):

        print(arr[i], end=" ")

    print()

# Driver Code

if \_\_name\_\_ == '\_\_main\_\_':

    arr = [12, 11, 13, 5, 6, 7]

    print("Given array is", end="\n")

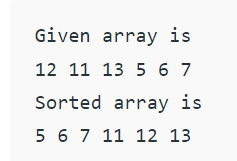
    printList(arr)

    mergeSort(arr)

    print("Sorted array is: ", end="\n")

    printList(arr)

**Output –**

****

**Conclusion –**

**Therefore, we have successfully understood and Implemented Merge sort.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Grade and Dated Signature of Teacher** | **Total (10)** | **Remark** | **Dated signature of teacher** |
|  |  |  |