Expt. No. 3 Date :

* Aim → Implement using divide and conquer technique and analyze time complexity of it (merge sort)

* Requirement → Computer system & TurboC

* Theory →

Divide and conquer is a technique of designing algorithm that consist of decomposing the problem to the solved into small number of sub problems. Divide & conquer approach involves

① divide the instance of problems into 2 or more small instances.

② solve smaller to original instance by solution

Steps

① divide & break : step involves breaking of problem that represents a part of original problem. This step generally take recurrsive approach.

② Conquer / Save : step receives a lot of smaller problems to be solved generally as this level the problem are consist solved by their own.

* Aim → Implement using divide & conquer technique & analyze time complexity of it (merge sort)

* Program →

```
# include <stdio.h>
void merge sort (int a[] ; int i ; int j);
void merge ( int a[], int i1, int j1, int i2, int j2);
int main()
{
  int a[30], n, i;
  printf("Enter number of elements");
  scanf("%d", &n);
  printf("Enter array elements");
  for( i=0 ; i<n ; i++)
  {
    scanf("%d", &a[i]);
    mergesort (a, 0, n-1);
    printf("\n sorted array i");
    for (i=0; i<n ; i++)
    printf("%d", a[i]);
    return 0;
  }
    void mergesort (int a[], int i, int j)
  { int mid;
    if (i<j)
    {mid = (i+j)/2;
     mergesort (a, i, mid);
     mergesort (a, mid+1, j);
     mergesort (a, i, mid+1 >j);
    }
  }
```

Expt. No. 3 Date :

③

Merge / combine → When smaller sub programs are solved, this step recursively combine them until they formulate the solution of original problem.

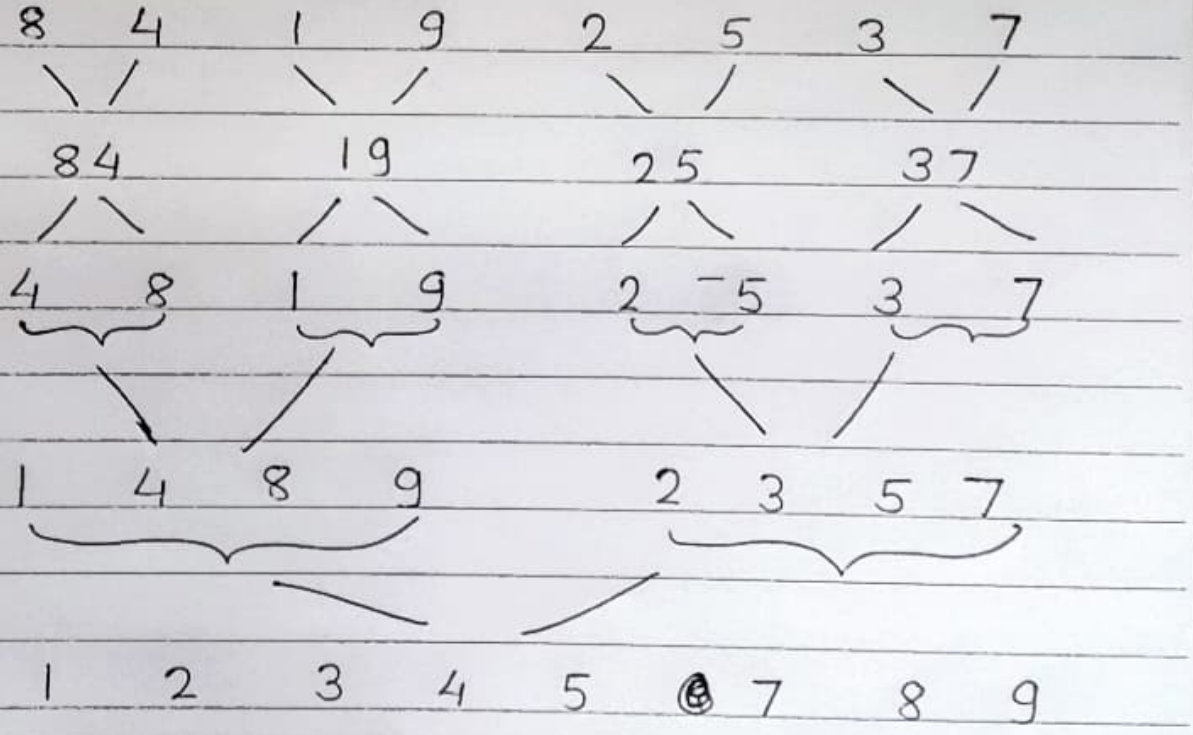
Following algorithms are based on divide and conquer programming approach :

ⓐ Merge sort ③ Binary sort (search)

ⓑ Quick sort ④ Closest path (points)

Merge sort is a divide & conquer based technique which divides the array into parts whose size are sorting these parts by recursive call & then merge the solution for even parts.

Ex →



```
void merge (int a[], int i1, int j1, int i2, int j2)
{
  int temp[50];
  int i, j, k;
  i = i1;
  j = j1;
  k = 0;
  while (i <= j1 && j <= j2)
  {
    if (a[i] < a[j])
      temp[k++] = a[i++];
    else
      temp[k++] = a[j++];
  }
  while (i <= j1)
  temp[k++] = a[i++];
  while (j <= j2)
  temp[k++] = a[j++];
  for (i = i1; j = a; i <= j2; i++; j++)
    a[i] = temp[j];
}
```

output →

```
Enter number of elements : 5
Enter array elements : 36  6  12  0  22
Sorted array is : 0  6  12  22  36
Process returned  0(0x0)
Execution time :  10.219
```

* Conclusion → We have successfully implement merge sort algorithm using divide and conquer technique and analyze complexity

Expt. No. 3 Date :

* Time complexity →

Merge sort works on dividing nodes in half of each level, the total time taken n(logn)

* Algorithm →

Step 1 : It is only one element in the list. It is already sorted.

Step 2 : Divide the list recursively into 2 halves until it can no more be divided.

Step 3 : Merge smaller list into new list.

* Conclusion →

We have successfully implement merge sort algorithm using divide and conquer technique and analyze complexity.