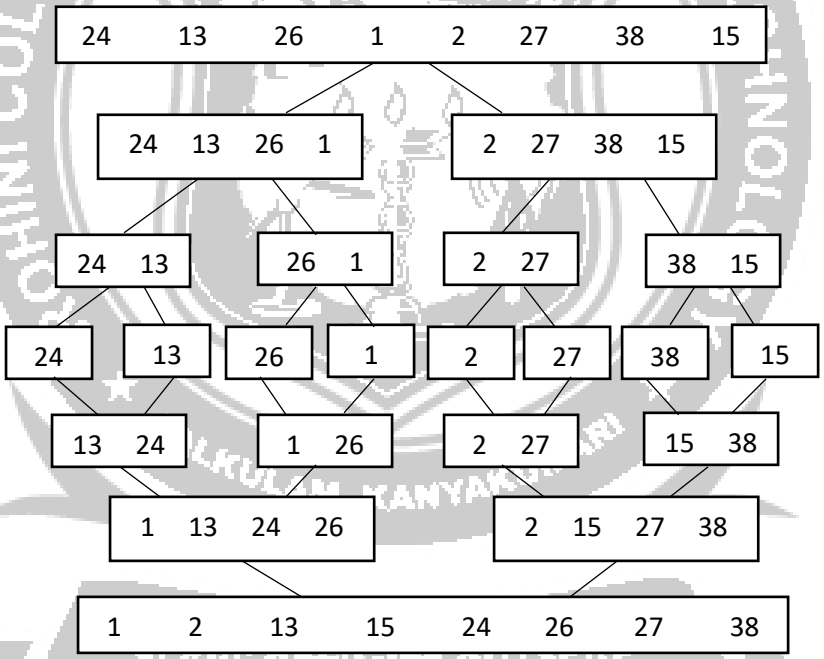
MERGE SORT

Merge Sort based on Divide and conquer strategy. The problem is divided into smaller problem and solved recursively. Finally it merges two sorted list.

Procedure:

1. Divide the given list f elements in two half.
2. Take two input lists A&B, an output list C.
3. The first element of list A &list B are compared, then the smaller element is stored in the output list C. The corresponding pointer is incremented.

Example: 24, 13, 26, 1, 2, 27, 38, 15



Routine

```
#include<stdio.h>
#include <stdlib.h>
void merge(int arr[], int l, int m, int r)
{
int i, j, k;
int n1 = m - l + 1;
```

```
int n2 = r - m;
int L[n1], R[n2];
for (i = 0; i < n1; i++)
L[i] = arr[l + i];
for (j = 0; j < n2; j++)
R[j] = arr[m + 1 + j];
i = 0;
j = 0;
k = l;
while (i < n1 && j < n2)
{
if (L[i] <= R[j])
{
arr[k] = L[i];i++;
}
else
{arr[k] = R[j];j++;
} k++;
}
while (i < n1)
{
arr[k] = L[i];i++;
k++;
}
while (j < n2)
{
arr[k] = R[j];j++;
k++;
}
}
```

```
void mergeSort(int arr[], int l, int r)
{
if (l < r)
{
int          m=l+(r-l)/2;
mergeSort(arr,l,m);
mergeSort(arr,m+1,r);
merge(arr, l, m, r);
}
}
void printArray(int A[], int size)
{
int i;
for (i = 0; i < size; i++)
printf("%d ", A[i]);
 printf("\n");
}
int main()
{
int arr[] = { 12, 11, 13, 5, 6, 7 };
intarrsize=sizeof(arr)/sizeof(arr[0]);
printf("Given array is \n");
printArray(arr,arrsize);
mergeSort(arr,0,arr_size-1);
printf("\nSorted   array   is   \n");
printArray(arr, arr_size); return 0;
}
```

Output

Given array is

12 11 13 5 6 7

Sorted array is

5 6 7 11 12 13

Time Complexity of merge sort:

Best case : O (n log n)

Average case : O (n log n) Worst case : O (n log n)