10. IMPLEMENTATION OF MERGE SORT

Preamble

Merge sort is similar to the quick sort algorithm as it uses the divide and conquer approach to sort the elements. It is one of the most popular and efficient sorting algorithm. It divides the given list into two equal halves, calls itself for the two halves and then merges the two sorted halves. We have to define the merge() function to perform the merging.

The sub-lists are divided again and again into halves until the list cannot be divided further. Then we combine the pair of one element lists into two-element lists, sorting them in the process. The sorted two-element pairs is merged into the four-element lists, and so on until we get the sorted list.

Steps

```
MERGE_SORT(arr, beg, end)
    if beg < end
        set mid = (beg + end)/2
        MERGE_SORT(arr, beg, mid)
        MERGE_SORT(arr, mid + 1, end)
        MERGE (arr, beg, mid, end)
    end of if
END MERGE_SORT
```

Implementation in C

```
#include <stdio.h>
/* Function to merge the subarrays of a[] */
void merge(int a[], int beg, int mid, int end)
{
    int i, j, k;
    int n1 = mid - beg + 1;
    int n2 = end - mid;

    int LeftArray[n1], RightArray[n2]; //temporary arrays
```

```
 /* copy data to temp arrays */
  for (int i = 0; i < n1; i++)
        LeftArray[i] = a[beg + i];
for (int j = 0; j < n2; j++)
        RightArray[j] = a[mid + 1 + j];
i = 0; /* initial index of first sub-array */
j = 0; /* initial index of second sub-array */
k = beg;  /* initial index of merged sub-array */
while (i < n1 && j < n2)
{
        if(LeftArray[i] <= RightArray[j])
        {
              a[k] = LeftArray[i];
              i++;
        }
        else
        {
              a[k] = RightArray[j];
              j++;
        }
        k++;
}
while (i<n1)
{
        a[k] = LeftArray[i];
        i++;
        k++;
}
while (j<n2)
{
        a[k] = RightArray[j];
        j++;
```

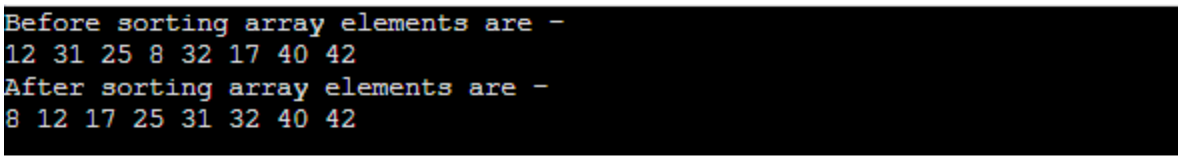
```
            k++;
        }
}

void mergeSort(int a[], int beg, int end)
{
      if (beg < end)
      {
            int mid = (beg + end) / 2;
            mergeSort(a, beg, mid);
            mergeSort(a, mid + 1, end);
            merge(a, beg, mid, end);
      }
}

/* Function to print the array */
void printArray(int a[], int n)
{
      int i;
      for (i = 0; i < n; i++)
            printf("%d ", a[i]);
      printf("\n");
}
int main()
{
      int a[] = { 12, 31, 25, 8, 32, 17, 40, 42 };
      int n = sizeof(a) / sizeof(a[0]);
      printf("Before sorting array elements are - \n");
      printArray(a, n);
      mergeSort(a, 0, n - 1);
      printf("After sorting array elements are - \n");
      printArray(a, n);
```

```
        return 0;
}
```

Sample Input and Output



```
Before sorting array elements are -
12 31 25 8 32 17 40 42
After sorting array elements are -
8 12 17 25 31 32 40 42
```