|  |
| --- |
| **Subject: DAA Class: S.E.(Comp)**    **Practical No.: 1 Date:** |

**AIM: Implement divide and conquer method by using merge sort.**

**Title:** Write algorithm and program for divide and conquer method using merge sort.

**Theory:**

Merge sort is a divide-and-conquer algorithm based on the idea of breaking down a list into several sub-lists until each sub list consists of a single element and merging those sub lists in a manner that results into a sorted list.

**How Merge Sort Works?**

To understand merge sort, we take an unsorted array as the following –

Unsorted Array

We know that merge sort first divides the whole array iteratively into equal halves unless the atomic values are achieved. We see here that an array of 8 items is divided into two arrays of size 4.

Merge Sort Division

This does not change the sequence of appearance of items in the original. Now we divide these two arrays into halves.

Merge Sort Division

We further divide these arrays and we achieve atomic value which can no more be divided.

Merge Sort Division

Now, we combine them in exactly the same manner as they were broken down. Please note the color codes given to these lists.

We first compare the element for each list and then combine them into another list in a sorted manner. We see that 14 and 33 are in sorted positions. We compare 27 and 10 and in the target list of 2 values we put 10 first, followed by 27. We change the order of 19 and 35 whereas 42 and 44 are placed sequentially.

Merge Sort Combine

In the next iteration of the combining phase, we compare lists of two data values, and merge them into a list of found data values placing all in a sorted order.

Merge Sort Combine

After the final merging, the list should look like this −

Merge Sort

Now we should learn some programming aspects of merge sorting.

**Time Complexity:**

The list of size N is divided into a max of logN parts, and the merging of all sublists into a single list takes O(N)time, the worst case run time of this algorithm is O(NLogN).

**Algorithm:**

Merge sort keeps on dividing the list into equal halves until it can no more be divided. By definition, if it is only one element in the list, it is sorted. Then, merge sort combines the smaller sorted lists keeping the new list sorted too.

1. If it is only one element in the list it is already sorted, return.
2. Divide the list recursively into two halves until it can no more be divided.
3. Merge the smaller lists into new list in sorted order.

**Source Code:**

/\* Simple Merge Sort Program Using Functions and Array in C\*/

/\* Data Structure Programs,C Functions and Array Examples \*/

#include<stdio.h>

#include<conio.h>

#define MAX\_SIZE 5

void merge\_sort(int, int);

void merge\_array(int, int, int, int);

int arr\_sort[MAX\_SIZE];

int main()

{

int i;

clrscr();

printf("\nEnter %d Elements for Sorting\n", MAX\_SIZE);

for (i = 0; i < MAX\_SIZE; i++)

scanf("%d", &arr\_sort[i]);

printf("\nYour Data :");

for (i = 0; i < MAX\_SIZE; i++) {

printf("\t%d", arr\_sort[i]);

}

merge\_sort(0, MAX\_SIZE - 1);

printf("\n\nSorted Data :");

for (i = 0; i < MAX\_SIZE; i++) {

printf("\t%d", arr\_sort[i]);

}

getch();

}

void merge\_sort(int i, int j) {

int m;

if (i < j) {

m = (i + j) / 2;

merge\_sort(i, m);

merge\_sort(m + 1, j);

// Merging two arrays

merge\_array(i, m, m + 1, j);

}

}

void merge\_array(int a, int b, int c, int d) {

int t[50];

int i = a, j = c, k = 0;

while (i <= b && j <= d) {

if (arr\_sort[i] < arr\_sort[j])

t[k++] = arr\_sort[i++];

else

t[k++] = arr\_sort[j++];

}

//collect remaining elements

while (i <= b)

t[k++] = arr\_sort[i++];

while (j <= d)

t[k++] = arr\_sort[j++];

for (i = a, j = 0; i <= d; i++, j++)

arr\_sort[i] = t[j];

}

**Output:**



**Conclusion:** In this experiment, We study divide and conquer method by using merge sort and implement merge sort.