NAME:- SAGAR ADHIKARI
ROLL NO:- PUR078BEI034
FACULTY:- ELECTRONICS(BEI)

DSA LAB SHEET NO. 7

TITLE:- IMPLEMENTATION OF SORTING (QUICK AND MERGE)

THEORY:-

Sorting:-A Sorting Algorithm is used to rearrange a given array or list of elements according to a comparison operator on the elements.
Quick Sort:- Quick Sort is a sorting algorithm based on the Divide and Conquer algorithm that picks an element as a pivot and partitions the given array around the picked pivot by placing the pivot in its correct position in the sorted array.
Merge Sort:-Merge sort is defined as a sorting algorithm that works by dividing an array into smaller sub arrays, sorting each sub array, and then merging the sorted sub arrays back together to form the final sorted array.

1.Quick Sort

PROGRAM CODE:-

```
#include <iostream>
#include <cmath>
#include <chrono>
using namespace std;
void swapp(int *x, int *y)
{
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
}

void display(int A[], int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << A[i] << " ";
    }
    cout << endl;
}

int partition(int A[], int l, int r)
{
    int x = l;
    int y = r;
    int pivot = A[l];
```

NAME:- SAGAR ADHIKARI
ROLL NO:- PUR078BEI034
FACULTY:- ELECTRONICS(BEI)

DSA LAB SHEET NO. 7

```
    while (x < y)
    {
        while (A[x] <= pivot)
        {
            x++;
        }
        while (A[y] > pivot)
        {
            y--;
        }
        if (x < y)
        {
            swapp(&A[x], &A[y]);
        }
    }
    A[l] = A[y];
    A[y] = pivot;
    return y;
}

void quicksort(int A[], int l, int r)
{
    if (l < r)
    {
        int p = partition(A, l, r);
        quicksort(A, l, p - 1);
        quicksort(A, p + 1, r);
    }
}

int main()
{
    int A[10000], n, i, key;
    cout << "Enter n: ";
    cin >> n;

    for (int i = 0; i < n; i++)
    {
        A[i] = rand();
    }

    cout << "Before sorting" << endl;
```

```
    display(A, n);
    auto t1 = chrono::high_resolution_clock::now();
    quicksort(A, 0, n - 1);
    auto t2 = chrono::high_resolution_clock::now();
    auto duration = chrono::duration_cast<chrono::microseconds>(t2 - t1);
    cout << "Time=" << duration.count() << " microseconds" << endl;
    cout << "After sorting" << endl;
    display(A, n);

    return 0;
}
```

OUTPUT:-

Enter n: 20
Before sorting
41 18467 6334 26500 19169 15724 11478 29358 26962 24464 5705 28145 23281 16827 9961 491 2995 11942 4827 5436
Time=2 microseconds
After sorting
41 491 2995 4827 5436 5705 6334 9961 11478 11942 15724 16827 18467 19169 23281 24464 26500 26962 28145 29358

2.Merge Sort

PROGRAM CODE:-

```
#include <iostream>
#include <chrono>
#include <cmath>
#include <cstdlib>
using namespace std;

void display(int A[], int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << A[i] << " ";
    }
    cout << endl;
}

void merge(int A[], int l, int m, int r)
{
```

NAME:- SAGAR ADHIKARI
ROLL NO:- PUR078BEI034
FACULTY:- ELECTRONICS(BEI)

DSA LAB SHEET NO. 7

```
    int i = l;
    int k = l;
    int j = m + 1;
    int c[100000];
    while (i <= m && j <= r)
    {
        if (A[i] < A[j])
            c[k++] = A[i++];
        else
            c[k++] = A[j++];
    }
    for (; i <= m; i++, k++)
        c[k] = A[i];
    for (; j <= r; j++, k++)
        c[k] = A[j];
    for (i = l; i <= r; i++)
        A[i] = c[i];
}

void mergeSort(int A[], int l, int r)
{
    if (l < r)
    {
        int m = floor((l + r) / 2);
        mergeSort(A, l, m);
        mergeSort(A, m + 1, r);
        merge(A, l, m, r);
    }
}

int main()
{
    int A[100000], n, i;
    cout << "Enter n: ";
    cin >> n;

    for (int i = 0; i < n; i++)
    {
        A[i] = rand();
    }

    cout << "Before sorting" << endl;
```

NAME:- SAGAR ADHIKARI
ROLL NO:- PUR078BEI034
FACULTY:- ELECTRONICS(BEI)


```
    display(A, n);

    auto t1 = chrono::high_resolution_clock::now();
    mergeSort(A, 0, n - 1);
    auto t2 = chrono::high_resolution_clock::now();
    auto duration = chrono::duration_cast<chrono::microseconds>(t2 - t1);
    cout << "Time taken for sorting: " << duration.count() << " microseconds" << endl;

    cout << "After sorting" << endl;
    display(A, n);

    return 0;
}
```

OUTPUT:-

Enter n: 20
Before sorting
41 18467 6334 26500 19169 15724 11478 29358 26962 24464 5705 28145 23281 16827 9961 491 2995 11942 4827 5436
Time taken for sorting: 426 microseconds
After sorting
41 491 2995 4827 5436 5705 6334 9961 11478 11942 15724 16827 18467 19169 23281 24464 26500 26962 28145 29358