Design Analysis and Algorithm – Lab Work

Week 3

1.MERGE SORT

CODE:

```c
#include <stdio.h>
void swap(int arr[], int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
void merge(int arr[], int low, int mid, int high) {
    int i = low, j = mid + 1, k = 0;
    int size = high - low + 1;
    int tempArray[size];
    while (i <= mid && j <= high) {
        if (arr[i] <= arr[j]) tempArray[k++] = arr[i++];
        else tempArray[k++] = arr[j++];
    }
    while (i <= mid) tempArray[k++] = arr[i++];
    while (j <= high) tempArray[k++] = arr[j++];
    for (i = 0; i < size; i++) arr[low + i] = tempArray[i];
}
void mergeSort(int arr[], int low, int high) {
    if (low < high) {
        int mid = low + (high - low) / 2;
        mergeSort(arr, low, mid);
        mergeSort(arr, mid + 1, high);
        merge(arr, low, mid, high);
    }
}
int main() {
    int n, i;
    printf("Enter number of elements: ");
    if (scanf("%d", &n) != 1) return 1;
    int arr[n];
    printf("Enter %d elements: ", n);
    for (i = 0; i < n; i++) scanf("%d", &arr[i]);
    mergeSort(arr, 0, n - 1);
    printf("Sorted array: ");
    for (i = 0; i < n; i++) printf("%d ", arr[i]);
    printf("\n");
    return 0;
}
```

OUTPUT:

```
Enter number of elements: 5
Enter 5 elements: 89 12 64 90 33
Sorted array: 12 33 64 89 90
```

Time complexity: O (n log n)Same number of comparisons are made.

Space Complexity: O(n)

2.QUICK SORT

CODE:

```c
#include <stdio.h>
void swap(int arr[], int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);
    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(arr, i, j);
        }
    }
    swap(arr, i + 1, high);
    return (i + 1);
}
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
int main() {
    int n, i;
    printf("Enter size: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter elements: ");
    for (i = 0; i < n; i++) scanf("%d", &arr[i]);
    quickSort(arr, 0, n - 1);
    printf("Sorted array: ");
    for (i = 0; i < n; i++) printf("%d ", arr[i]);
    return 0;
}
```

OUTPUT:

```
Enter size: 5
Enter elements: 67 12 90 1 43
Sorted array: 1 12 43 67 90
```

Time complexity**:** O(n^2)-Occurs when the array is already sorted and we pick the last element as pivot.

Space Complexity: O (log n) it uses O (log n) space on the recursion stack for the function calls.