DESIGN AND ANALYSIS OF ALGORITHMS

LAB WORKBOOK

NAME : MADDU RAAHITHYA YADAV

ROLL NUMBER : CH.SC.U4CSE24124

CLASS : CSE-B

1.QUICK SORT

CODE:

```c
#include <stdio.h>
int partition(int a[], int low, int high) {
    int pivot = a[high], i = low - 1, temp;
    for (int j = low; j < high; j++) {
        if (a[j] <= pivot) {
            i++;
            temp = a[i]; a[i] = a[j]; a[j] = temp;
        }
    }
    temp = a[i+1]; a[i+1] = a[high]; a[high] = temp;
    return i + 1;
}
void quickSort(int a[], int low, int high) {
    if (low < high) {
        int p = partition(a, low, high);
        quickSort(a, low, p - 1);
        quickSort(a, p + 1, high);
    }
}
int main() {
    int n, a[50];
    printf("Enter number of elements for quick sort: ");
    scanf("%d", &n);
    printf("Enter elements:\n");
    for (int i = 0; i < n; i++) scanf("%d", &a[i]);
    quickSort(a, 0, n - 1);
    printf("Sorted array:\n");
    for (int i = 0; i < n; i++) printf("%d ", a[i]);
    printf("\n");
    return 0;
}
```

OUTPUT:

```
raahithya@ubuntu-virtualbox:~/Documents$ gcc QUICKSORT.c -o result
raahithya@ubuntu-virtualbox:~/Documents$ ./result
Enter number of elements for quick sort: 12
Enter elements:
157
110
147
122
111
149
151
141
143
112
117
133
Sorted array:
110 111 112 117 122 133 141 143 147 149 151 157
```

Time Complexity

- The pivot divides the array into two equal halves.
- At each level, all n elements are compared once.
- The array is divided log n times.
- So,
  Time = n × log n = O(n log n)

2.MERGE SORT

CODE:

```c
#include <stdio.h>

void merge(int a[], int low, int mid, int high) {
    int i = low, j = mid + 1, k = low;
    int temp[50];

    while (i <= mid && j <= high) {
        if (a[i] <= a[j])
            temp[k++] = a[i++];
        else
            temp[k++] = a[j++];
    }

    while (i <= mid)
        temp[k++] = a[i++];

    while (j <= high)
        temp[k++] = a[j++];

    for (i = low; i <= high; i++)
        a[i] = temp[i];
}

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

```c
int main() {
    int n, i;
    int a[50];

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter elements:\n");
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);

    mergeSort(a, 0, n - 1);

    printf("Sorted array:\n");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);

    return 0;
}
```

OUTPUT:

```
raahithya@ubuntu-virtualbox:~/Documents$ gcc MERGESORT.c -o result
raahithya@ubuntu-virtualbox:~/Documents$ ./result
Enter number of elements: 12
Enter elements:
157 110 147 122 111 149 151 141 143 112 117 133
Sorted array:
110 111 112 117 122 133 141 143 147 149 151 157
```

Time Complexity

- The array is always divided into two equal halves.
- Number of divisions = log n.
- At each level, all n elements are merged.
- So,
  Time = n × log n = O(n log n)