NATIONAL UNIVERSITY OF MODERN LANGUAGES

Analysis Of Algorithm Assignment No 1

NAME: Raja Umer Shehzad

ROLL NO: FL23803

PROGRAM: BSSE (5^{th} Semester)

COURSE: Analysis of Algorithm

SUBMITTED TO: Sir Waris Ali

DATE: 21, Oct/2025 **DAY: Tuesday**

Q1: Merge Sort Algorithm

```
void merge(int arr[], int left, int mid, int right) {
  int n1 = mid - left + 1;
  int n2 = right - mid;

  int L[n1], R[n2];

  for (int i = 0; i < n1; i++)
    L[i] = arr[left + i];
  for (int j = 0; j < n2; j++)
    R[j] = arr[mid + 1 + j];

  int i = 0, j = 0, k = left;

  while (i < n1 && j < n2) {
    if (L[i] <= R[j]) {
      arr[k] = L[i];
      i++;
    } else {
      arr[k] = R[j];
      j++;
    }
    k++;
  }

  while (i < n1) {
    arr[k] = L[i];
    i++;
```

```
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = (left + right) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}
```

Q2: Merge Sort Examples

Example 1

Initial List:
[38, 27, 43, 3, 9, 82, 10, 19, 15, 4, 70, 66]

Step 1 – Divide:
Split the array into halves repeatedly until each sublist has one element.

- Split 1: [38, 27, 43, 3, 9, 82] | [10, 19, 15, 4, 70, 66]
- Split 2: [38, 27, 43] | [3, 9, 82] | [10, 19, 15] | [4, 70, 66]
- Single elements: [38], [27], [43], [3], [9], [82], [10], [19], [15], [4], [70], [66]

Step 2 – Merge:
Combine sublists in sorted order.

- Pass 1: [27, 38], [3, 43], [9, 82], [10, 19], [4, 15], [66, 70]
- Pass 2: [3, 27, 38, 43], [9, 10, 19, 82], [4, 15, 66, 70]
- Pass 3: [3, 9, 10, 19, 27, 38, 43, 82], [4, 15, 66, 70]
- Final Merge: [3, 4, 9, 10, 15, 19, 27, 38, 43, 66, 70, 82]

Final Sorted List:
[3, 4, 9, 10, 15, 19, 27, 38, 43, 66, 70, 82]

Example 2

Initial List:
[65, 10, 5, 75, 20, 30, 90, 45, 80, 50, 25, 35]

Step 1 – Divide:

- Split 1: [65, 10, 5, 75, 20, 30] | [90, 45, 80, 50, 25, 35]
- Split 2: [65, 10, 5] | [75, 20, 30] | [90, 45, 80] | [50, 25, 35]
- Final single elements: [65], [10], [5], [75], [20], [30], [90], [45], [80], [50], [25], [35]

Step 2 – Merge:

- Pass 1: [10, 65], [5, 75], [20, 30], [45, 90], [50, 80], [25, 35]
- Pass 2: [5, 10, 65, 75], [20, 30], [45, 80, 90], [25, 35, 50]
- Pass 3: [5, 10, 20, 30, 65, 75], [25, 35, 45, 50, 80, 90]
- Final Merge: [5, 10, 20, 25, 30, 35, 45, 50, 65, 75, 80, 90]

Final Sorted List:
[5, 10, 20, 25, 30, 35, 45, 50, 65, 75, 80, 90]

Q3: Difference Between Sorting Types

Type	Definition	Memory Usage	Examples
In-Place Sorting	Sorts data in the same memory space without using extra storage.	Very low (O(1))	Bubble Sort, Insertion Sort, Selection Sort

Type	Definition	Memory Usage	Examples
In-Memory Sorting	Sorts all data that fits in main memory (RAM).	Moderate	Merge Sort, Counting Sort, Radix Sort
External Sorting	Used when data is too large to fit in memory, and files/disks are used.	High (uses secondary storage)	Merge Sort

Q4: Comparison of Sorting Algorithms

Algorithm	Best Case	Average Case	Worst Case	Space Complexity	Type
Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$	O(1)	In-place
Insertion Sort	O(n)	$O(n^2)$	$O(n^2)$	O(1)	In-place
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)	In-place
Merge Sort	O(n log n)	O(n log n)	O(n log n)	O(n)	In-memory
Counting Sort	O(n + k)	O(n + k)	O(n + k)	O(k)	In-memory
Radix Sort	O(nk)	O(nk)	O(nk)	O(n + k)	In-memory
Bucket Sort	O(n + k)	O(n + k)	$O(n^2)$	O(n + k)	In-memory