NATIONAL UNIVERSITY OF MODERN LANGUAGES

Analysis Of Algorithm Assignment No 1

NAME: Abdul Rafay

ROLL NO: FL23791

PROGRAM: BSSE (5th Semester)

COURSE: Analysis of Algorithm

SUBMITTED TO: Sir Waris Ali

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

Q1: Merge Sort Algorithm (C++ Form)

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

    int L[n1 +1], R[n2+1];

    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 = 0;

    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: Two Examples Using Merge Sort

Initial Unsorted List

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

Step 1: Divide

The list is split down until each sublist has only 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]

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

Step 2: Merge (The Sorting Process)

The sublists are merged back together in sorted order.

Pass 1: Merge lists of 1 element [27, 38] | [3, 43] | [9, 82] | [10, 19] | [4, 15] | [66, 70]

Pass 2: Merge lists of 2 elements [3, 27, 38, 43] | [9, 10, 19, 82] | [4, 15, 66, 70]

Pass 3: Merge lists of 4 and 4 elements [3, 9, 10, 19, 27, 38, 43, 82] | [4, 15, 66, 70]

Pass 4: Final Merge The two final sorted lists are combined into one fully Sorted List.

Final Sorted List

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

Example: 2

Initial Unsorted List

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

Step 1: Divide

The list is continuously split into halves until only single elements remain.

Split 1 (6 elements each): [65, 10, 5, 75, 20, 30] | [90, 45, 80, 50, 25, 35]

Split 2 (3 elements each): [65, 10, 5] | [75, 20, 30] | [90, 45, 80] | [50, 25, 35]

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

Step 2: Merge (The Sorting Process)

The sublists are merged back, performing the sort.

Pass 1: Merge lists of 1 element [10, 65] | [5, 75] | [20, 30] | [45, 90] | [50, 80] | [25, 35]

Pass 2: Merge lists (to form lists of 3, 4, or 6 elements) [5, 10, 65, 75] | [20, 30] | [45, 80, 90] | [25, 35, 50]

Pass 3: Merge lists of 4 and 2 (and 3 and 3) [5, 10, 20, 30, 65, 75] | [25, 35, 45, 50, 80, 90]

Pass 4: Final Merge The two sorted lists of 6 elements are combined into one fully Sorted List.

Final Sorted List

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

Q3: Difference Between In-Place, In-Memory, and External Sorting

1. In-Place Sorting

- **Definition:**
  In-place sorting algorithms perform sorting within the same memory space as the input data. They do **not require extra memory**.
- **Memory Usage:** Very low (O(1) or constant extra space).
- **Examples:** Bubble Sort, Insertion Sort, Selection Sort,.

2. In-Memory Sorting

- **Definition:**
  In-memory sorting algorithms sort all data **that fits completely into main memory (RAM)**.
- **Memory Usage:** Moderate; data must reside in RAM for processing.
- **Examples:** Merge Sort,, Counting Sort, Radix Sort.

3. External Sorting

- **Definition:**
  External sorting is used when the data **does not fit into the main memory** and must be sorted using **secondary storage (like disk or SSD)**.
- **Memory Usage:** High — uses files and external memory for intermediate sorting steps.
- **Examples:** 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