

**Symbiosis Institute of Technology**

**Faculty of Engineering**

**CSE- Academic Year 2024-25**

**Data Structures – Lab Batch 2023-27**

|  |  |
| --- | --- |
|  | **Lab Assignment No:- 3** |
|  |  |
| **Name of Student** | Laxmi kumari sah |
| **PRN No.** | 23070122125 |
| **Batch** | 2023-27 |
| **Class** | CS-B1 |
| **Academic Year & Semester** | 2024, SEM-III |
| **Date of Performance** | 6/08/2024 |
|  |  |
| **Title of Assignment:** | Implement following sorting techniques find the time complexity: Merge |
| **Theory Questions:** | 1. Apply merge Sort on 9 input items and show the partial pass-wise sorting done. Analyze its Time Complexity (Best, Worst, and Average Case) & Space Complexity    Input List:  [38, 27, 43, 3, 9, 82, 10, 5, 29]    Step 1: Divide the list into smaller sublists.  -Divide the list into two halves:  Left: [38, 27, 43, 3, 9]  Right: [82, 10, 5, 29]    Step 2: Recursively divide each sublist until each sublist contains only one element. - Divide Left half:  Left: [38, 27]  Right: [43, 3, 9]    - Divide Right half:  Left: [82, 10]  Right: [5, 29]    Continue dividing until all sublists contain only one element: |

|  |  |
| --- | --- |
|  | * [38, 27] -> [38], [27] * [43, 3, 9] -> [43], [3, 9] -> [3], [9] * [82, 10] -> [82], [10] * [5, 29] -> [5], [29]     Step 3: Merge the sublists by comparing and sorting them.    1. First merge:   * Merge [38] and [27] -> [27, 38] * Merge [3] and [9] -> [3, 9] * Merge [82] and [10] -> [10, 82] * [5] and [29] remain as is since they are single-element lists.     2. Second merge:   * Merge [27, 38] with [43] -> [27, 38, 43] * Merge [3, 9] with [27, 38, 43] -> [3, 9, 27, 38, 43] - Merge [10, 82] with [5, 29] -> [5, 10, 29, 82]     3. Final merge:   * Merge [3, 9, 27, 38, 43] with [5, 10, 29, 82] -> [3, 5, 9, 10, 27, 29, 38, 43, 82]     Final Sorted List:  [3, 5, 9, 10, 27, 29, 38, 43, 82]    Time Complexity Analysis:     * Best Case: (O(n log n)) * Merge Sort always divides the array into two halves and then merges them, regardless of the initial order of elements. So, the best-case time complexity is (O(n log n)).      * Worst Case: (O(n log n)) * The worst-case occurs when the array is sorted in reverse order, but since the divide and merge process remains consistent, the time complexity is still (O(n log n)).      * Average Case: (O(n log n)) * Regardless of the input, the process of dividing and merging takes O(n log n) time.     Space Complexity Analysis:     * Space Complexity: O(n) * Merge Sort requires additional space proportional to the size of the input array due to the need to create temporary arrays during the merge process. Hence, the space complexity is O(n). |

|  |  |  |
| --- | --- | --- |
|  | 2. | Discuss time complexity of merge sort and quick sort in detail. |

|  |  |
| --- | --- |
| **Source**  **Code/Algorithm/Flow Chart:** |  |
| **Output Screenshots** |  |
| **Practice questions** | 1. Implement Quick sort |
|  | 2. o/p screenshot |
| **Conclusion** | Thus we have studied different sorting algorithms and their time complexities. |