Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

UCID: 2024510001 BATCH: A NAME: Atharva Vasant Angre

EXPERIMENT NO: 01

EXPERIMENT TITLE: To implement Divide and conquer method

1.1 To Study different Time and space complexity tools

1.2 To Implement Quick sort algorithm and determine the Time and space complexity

1.3 To Implement Merge sort algorithm and determine the Time and space complexity

Objective:

1. To study various tools for time and space complexity.
2. To understand Big O, omega and Theta notations.
3. To implement the divide and conquer method and calculate the time and space complexity.

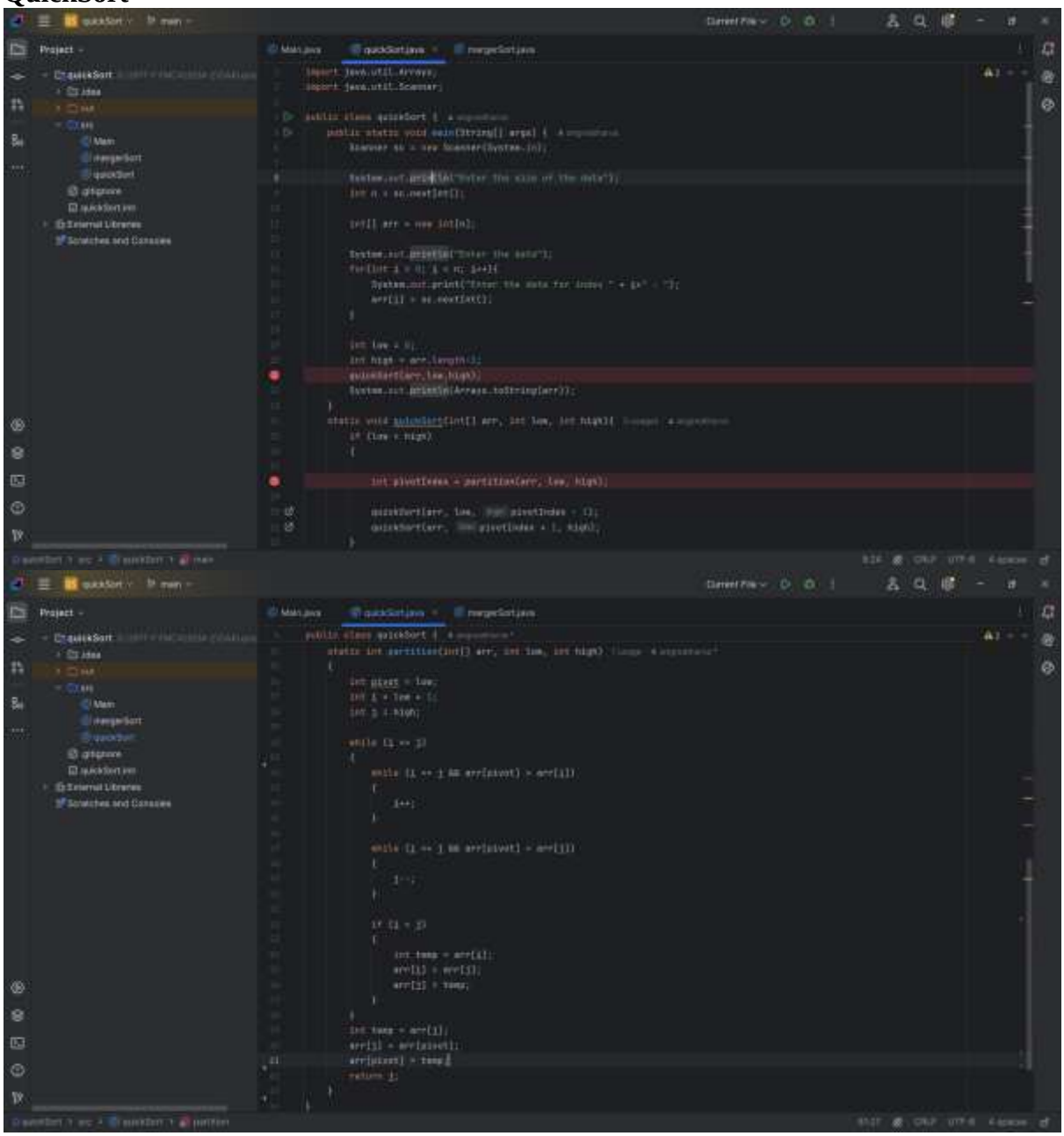
BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
MUNSHI NAGAR, ANDHERI (WEST), MUMBAI – 400 058.
(Autonomous College Affiliated to University of Mumbai)
MASTER OF COMPUTER APPLICATIONS

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

Program code: -
QuickSort



UCID: 2024510001

BHARATIYA VIDYA BHAVAN'S

SARDAR PATEL INSTITUTE OF TECHNOLOGY

MUNSHI NAGAR, ANDHERI (WEST), MUMBAI – 400 058.

(Autonomous College Affiliated to University of Mumbai)

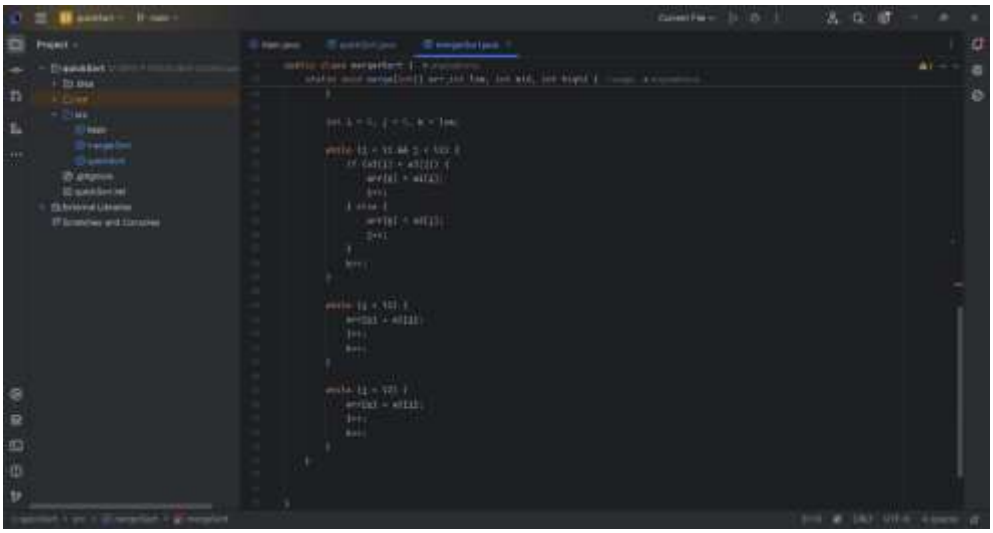
MASTER OF COMPUTER APPLICATIONS

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

MergeSort



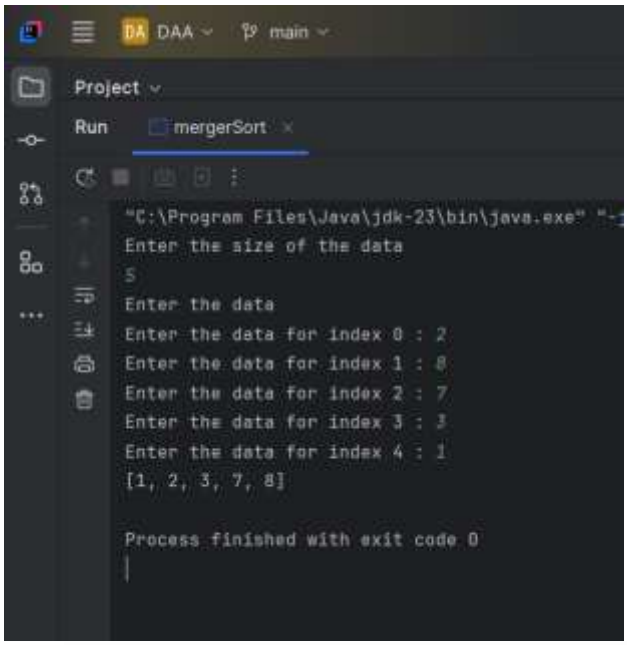
UCID: 2024510001

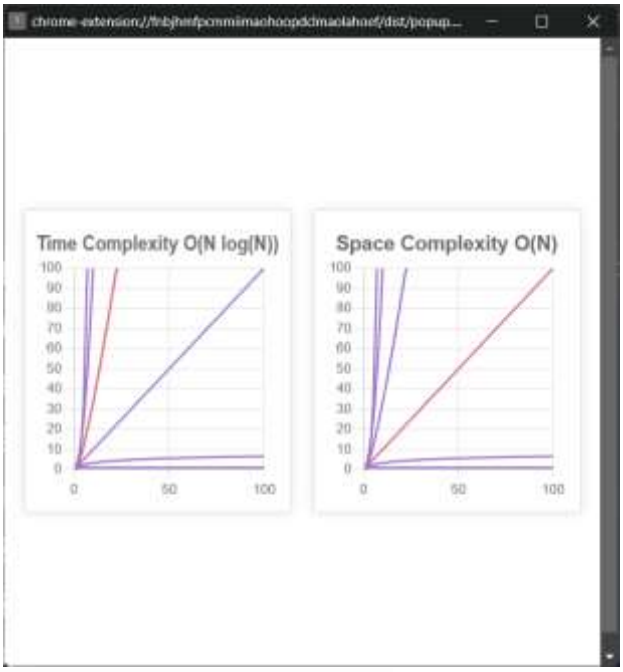
Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

Output:
After executing the Merge Sort and Quick Sort implementations with different input cases, the results display a sorted array. Below is the time complexity analysis for each sorting algorithm:
Merge Sort:





- Best Case Complexity: O (n log n) (Occurs when the array is already sorted)
- Worst Case Complexity: O (n log n) (Occurs for any random order of elements since Merge Sort always divides the array into two halves)
- Average Case Complexity: O (n log n)

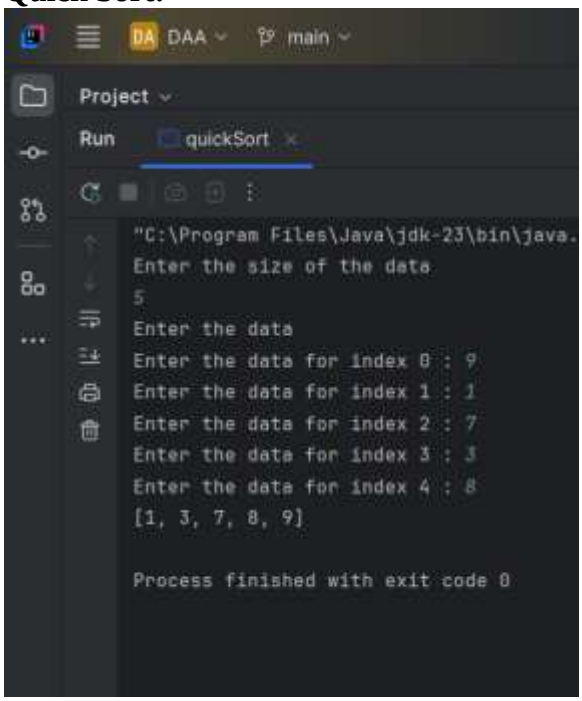
Time and Space complexity of merge sort using the extension Time and Space complexity.

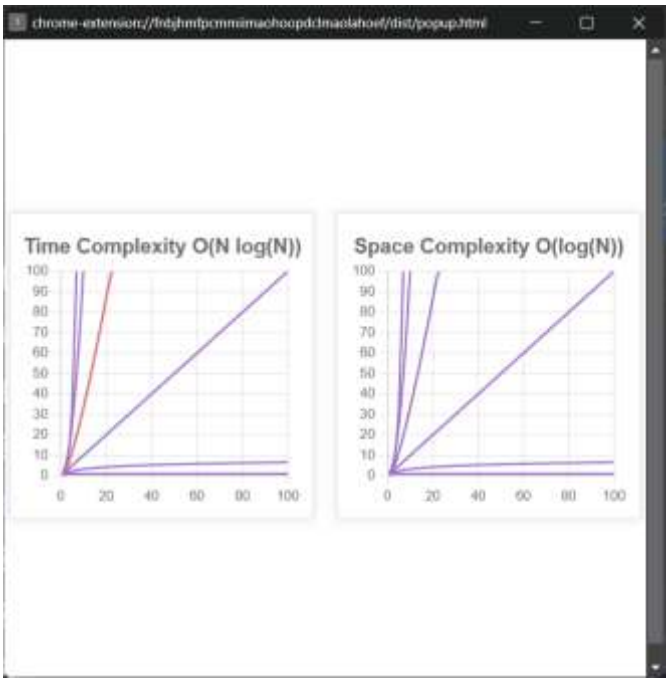
Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

Quick Sort:





- Best Case Complexity: O (n log n) (Occurs when the pivot divides the array into nearly equal halves)
- Worst Case Complexity: O (n^2) (Occurs when the smallest or largest element is always chosen as the pivot, resulting in an unbalanced partition)
- Average Case Complexity: O (n log n)

Time and Space complexity of quick sort using the extension Time and Space complexity.

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

Result:

Merge Sort:

- Best Case: Even when the array is sorted, Merge Sort still recursively divides the array and performs merge operations, leading to O (n log n).
- Worst Case: It performs the same number of comparisons and merge steps regardless of the order of input, so the worst case remains O (n log n)
- Average Case: The division and merging operations make the complexity O (n log n) in all cases.

Quick Sort:

- Best Case: Achieved when the pivot splits the array into two nearly equal halves, making the complexity O (n log n).
- Worst Case: If the pivot is always the smallest or largest element, it results in O (n^2) due to highly unbalanced partitions.
- Average Case: The expected time complexity remains O (n log n) assuming the pivot is chosen randomly or efficiently.

Conclusion:

This experiment helped in understanding:

1. Divide and Conquer Approach – Both Merge Sort and Quick Sort divide the problem into smaller subproblems, solve them recursively, and combine the results.
2. Time Complexity Notations – Best, worst, and average case complexities were analyzed using Big (O) notations.
3. Comparison of Sorting Algorithms – Merge Sort provides stable sorting and consistent performance, while Quick Sort is faster in practice but suffers from poor worst-case scenarios if not optimized.

Applications of These Sorting Techniques:

1. Merge Sort Applications:
    - Sorting linked lists (as merging is easy in linked lists)
    - External sorting (large datasets that don't fit into memory)
    - Data processing applications (e.g., log file analysis)
2. Quick Sort Applications:
    - Database sorting (due to fast in-memory sorting)
    - Used in standard libraries like Java's Arrays.sort()