**Data Structures and Algorithms**

Logo, company name

Description automatically generated

**Lab report: 8**

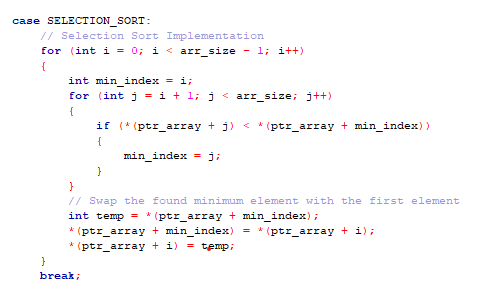
|  |  |
| --- | --- |
| **Name:** | **Ali Salman** |
| **Reg no:** | **FA22-BCE-005** |
| **Class:** | **BCE-3A** |
| **Lab Instructor:** | **Dr. Ali Mustafa** |

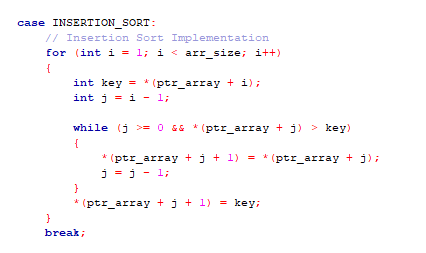
**Lab 08**

**Implementation of Sorting Algorithms**

**In-Lab Task 1:**

**Complete the functions for Selection Sort and Insertion Sort**





**In-Lab Task 2:**

**Empirically compute the execution times for the three sorting methods**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S#** | **Data Size ↓** | **Bubble Sort** | **Selection Sort** | **Insertion Sort** | **Merge Sort** |
| 1 | 16 | 0 | 0 | 0 | 0 |
| 2 | 128 | 0 | 0 | 0 | 0 |
| 3 | 1024 | 0 | 0.007991 | 0 | 0 |
| 4 | 16384 | 1.062421 | 0.531214 | 0.437465 | 0 |
| 5 | 131072 | 74.962894 | 32.810202 | 28.720268 | 0.046872 |

**Post Lab Task:**

**Complete the Merge Sort Function and empirically determine its time complexity.**

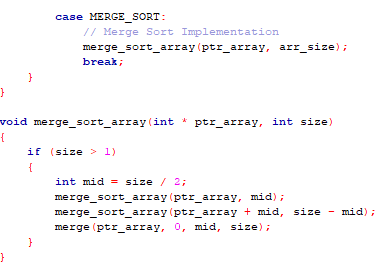
Merge Sort is a recursive algorithm and time complexity can be expressed as following recurrence relation. T(n) = 2T(n/2) + O(n) The solution of the above recurrence is O(nLogn) The list of size N is divided into a max of Logn parts, and the merging of all sublists into a single list takes O(N) time, the worst-case run time of this algorithm is O(nLogn)

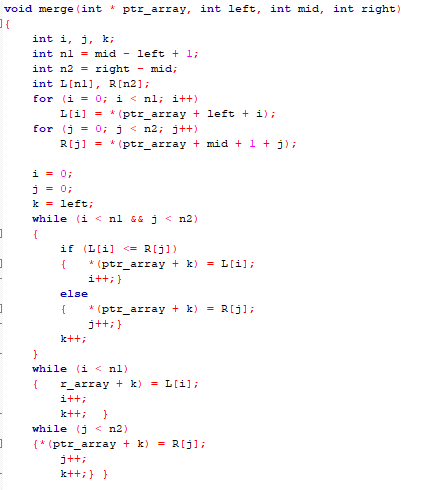
Best Case Time Complexity: O(n\*log n)

Worst Case Time Complexity: O(n\*log n)

Average Time Complexity: O(n\*log n)

The time complexity of MergeSort is O(n\*Log n) in all the 3 cases (worst, average and best) as the mergesort always divides the array into two halves and takes linear time to merge two halves.

****

****

**Critical Analysis:**

This lab provides a framework for benchmarking and comparing the performance of different sorting algorithms, including Bubble Sort, Selection Sort, Insertion Sort, and Merge Sort. The use of a time measurement technique employing the `gettimeofday` function enables the empirical evaluation of sorting algorithm efficiency. However, it's important to note that the program generates an array of random numbers for each run, which might affect the accuracy of time complexity measurements. Additionally, As we have studied the time complexities of different algorithm, we came to know that merging algorithm is the best algorithm for large size of data