# Department of Computing

# School of Electrical Engineering and Computer Science

**CS-250: Data Structure and Algorithms**

**Class: BEEE** **13 (Grp1+2)**

**Lab 12: Merge & Quick Sort**

**Date: 29th April, 2024**

**Time: 10:00 am – 12:50 am**

# Lab Engineer: Anum Asif

# Lab 12: Quick Sort

**Introduction**

In this lab, you will implement Quick Sort.

**Objectives**

Objective of this lab is to implement quick sort then compare the running times for sorting.

**Tools/Software Requirement**

Visual Studio C++

**Helping Material**

Lecture slides, text book

**Description**

**Merge Sort:**

Merge sort is another important sorting algorithm that we have seen. Unlike insertion sort, it is not an in-place sorting algorithm. The pseudo code for merge sort is shown below:



Merge (Arr, n1, mid, n2)

a=n1, b=mid, c=n1 ,B;

while a <= mid and b<=n2

if Arr[a]<Arr[b]

B[c++]=Arr[a++];

else

B[c++]=Arr[b++];

while a<mid

B[c++]=Arr[a++];

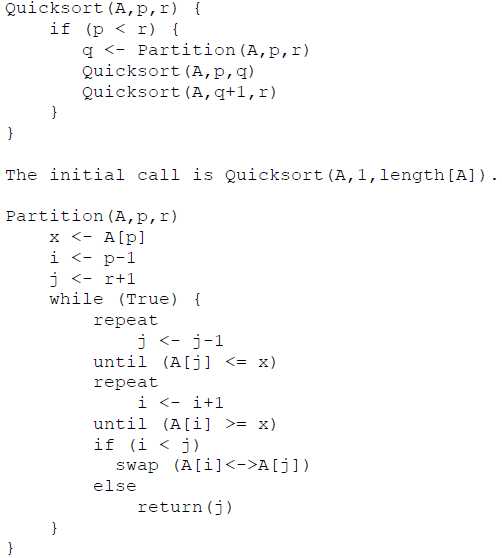
while b<n2

B[c++]=Arr[b++];

for a=n1; a<n2; a++

Arr[a]=B[a];

**Quick Sort:-**

****

**Lab Tasks**

**Task 1:**

A). Implement the merge & quick sort algorithm using C++.

**Task 2: (average case complexity):**

A). The next step is to compute the time complexity of the merge & quick sort algorithm. Generate arrays of random numbers in the range 1 to 100 with sizes 100, 1000, 10000, 100000, and 1000000. Compare the running times of the algorithm on each array. Are the results what you expected, and why? Answer the questions in the solution section.

**Task 3: (best and worst case complexity):**

A). Now sort the arrays using stl::sort, once in ascending order and then in descending order. Given both sorted arrays as inputs to the algorithm and compute their running time. Does the running time of algorithm shows variations based on the structure of the input and why? Plot the running time of the best and worst case complexities for different input sizes in a excel sheet and add it in the solution section.  
   
  
**Important Note:** Practice your knowledge of OOP with C++ when creating a solution.

**Deliverables**

This lab grading policy is as follows: Insert the solution/answer in this document. You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on the LMS.

**Note:** Students are required to upload the lab on LMS before deadline.

Use proper indentation and comments. Lack of comments and indentation will result in deduction of marks.