**COMSATS Institute of Information Technology, Islamabad Campus**

**Department of Computer Science**

**Course title & Code: Design and Analysis of Algorithms – CSC301 Course Instructor: Memoona Malik**

**Date: 8/5/21 Time: One hour Max Marks: 15**

**Name: Mohammad Haris Zia Registration#FA19-bcs-037**



**CLO3**

**Q1: a) How quick sort executes for the sorted input data if first element is chosen as the pivot. [2]**

In this case where we choose the pivot as the first element, we will have very unbalanced arrays as a result. After the first partition one array will have one element and the other will have N-1 elements as it continues so on. The first partition will take N times to perform the partition step after that each partition is invoked recursively from the previous one if we sum up the complexity of each partition using the analysis of a recurrence tree, we will get the time complexity of this case O(n2) which is the worst-case complexity of quick sort algorithm.

**b) Suppose you must sort any given data using the Merge sort algorithm and search a key in that data using binary search algorithm. What would be the time complexity of this task? Justify your answer. Consider the data in ascending, descending, and random order. [4]**

As we must implement the working of merge sort in order to sort the data.

If it is already sorted or not sorted or partially ported, in all cases the sorting will take O (n log n) time complexity.

Second, we need to search for some key in the array. As Binary search requires a sorted array which we have already provided so the searching will take O (log n) complexity.

**Ascending Sorted:**

If the input is in sorted ascending order, Merge sort takes O (n log n) and Binary Search Takes O (log n).

**Descending Sorted:**

If the input is in sorted Descending order, Merge sort takes O (n log n) and Binary Search Takes O (log n).

**Random Order:**

If the input is in random sorted order, Merge sort takes O (n log n) and Binary Search Takes O (log n).

**Justification:**

Whether the array is unsorted or sorted, Merge sort takes O (n log n) in its best or worst case or average time complexity because it is a divide and conquer technique and takes O (log n) to divide and linear time O(n) to merge. In order to search using binary search time complexity becomes O (log n). Even if we find the element at the start the complexity will still stay O (n log n) as we required sorting for binary search.

**So, the time complexity become O (n log n) in all the above-mentioned cases.**

**c) Solve the best and worst case of Quick sort algorithm using the master method. [3]**

**Best Case Partition Analysis:**

2T(n/2) + O(n)

a=2, b=2

log 2 2=1

T(n)=O(nlog 2 2)

So, f(n)=T(n)

T(n) = O (nlogn)

**Worst Case Partition Analysis:**

Let a = 2, b = 1, f(n) = O(n)

Since a > 1 then case 3 is applicable

T(n) = O( an \* f(n))

T(n) = O (2n \* n)

T(n) = O (n \* n)

T(n) = n2

**CLO4**

**Q2. (a) Suppose you are provided with a graph of any social networking application. Which method would you use to store this graph in computer and why?     [2]**

I will use an **undirected graph** in this case. A line between two people will represent that they know each other and a relationship goes both ways. For instance, if a person A knows a person B, that means that person B knows person A as well. If there is no line i.e., an **edge** present between two people that simply signifies that they do not know each other. The people names represent the **vertices** of the graph and the edges represent the relationship between them. An undirected graph is suitable in this case representing two-way relationships. Also, the vertices connected by edge are referred as **neighbors** or **adjacent** and the number of edges mapped on a certain vertex represent the **degree** of the vertex.

**E.g.:** If a person A does not know a person D i.e., there is no direct edge between the two. We suppose that the person A knows person B who in turn know person C who has a direct edge to person D. That is how person A can approach person D by traversing through a set of vertices. And since it is the most direct way to approach person D, we refer it as **shortest path**. Secondly, we can use BFS algorithm in order to find mutual friends in the social networking application.

Moreover, there also might be cycles in this graph, for instance, if a person A knows person B who in turn knows person C and surprisingly person C knows person A. So, we say that we have a **cycle** in your graph.

**b) Explain the time complexity of depth first search? [4]**

Depth First Algorithm (DFS) traverses a graph in depth and maintains a record of the next visited node using a stack data structure.

**Time Complexity:**

***V are the vertices and E represents the edges of a graph.***

In Depth First Search, for each node discover all its neighbors by traversing the adjacency list.

**Formula:**

Sum of sizes of adjacency list = Edges (E)

Thus, **time complexity is: O (V) + O(E)**

So, we can conclude that,

**O(V) represents** the time complexity for traversing through the vertices.

**O(E) represents** the DFS visit for neighboring nodes of a vertex v that belongs to V (Vertices)