BAHRIA UNIVERSITY (KARACHI CAMPUS)

Discrete Structures (CSC-115)

Assignment 02
Spring 2023

Class: BSE 2B | **Shift: Morning**
Course Instructor: ENGR. FAIZ UL HAQUE ZEYA | **Due Date: 13 May 2023**
Assignment Date: 06 May 2023 | **Marks: 05 Points**
Student Name: ABDULLAH | **Registration #: 81962**

Question 01: Explain the following sorting algorithm and find their worst best and average case complexity. (**3 marks**).

1. Merge sort.
2. Quick sort.

Solution:

1) Merge Sort:

Merge Sort is a divide and conquer algorithm that divides an array into halves, sorts each half separately, and then merges the sorted halves into one sorted array. The basic steps involved in Merge Sort are:

- Divide the array into two halves until each sub-array has only one element.
- Merge the sub-arrays by comparing the elements of the sub-arrays in pairs and placing them in order.
- Repeat the merger process until there is only one sorted array.

Worst Case Complexity: O(n*logn)
Best Case Complexity: O(n*logn)
Average Case Complexity: O(n*logn)

2) Quick Sort:

Quick Sort is another divide and conquer algorithm that selects a pivot element and partitions the array around the pivot, such that all elements less than the pivot are moved to its left and all elements greater than the pivot are moved to its right. Then it recursively sorts the two partitions. The basic steps involved in Quick Sort are:

- Choose a pivot element from the array.
- Partition the array such that all elements smaller than the pivot are placed to its left and all elements greater than the pivot are placed to its right.
- Recursively repeat the above two steps on the left and right partitions until the entire array is sorted.

Worst Case Complexity: O(n^2)
Best Case Complexity: O(n*logn)
Average Case Complexity: O(n*logn)

Question 02: Choose a problem and apply amortized analysis to it. (**2 marks**)

Solution:

Consider a binary counter that can increase a binary number of length n by one bit at a time. Using the potential method of amortized analysis, we can show that the amortized cost of incrementing the binary counter is O (1) per bit. We define the potential of the binary counter as the number of 1 bit in the current binary representation minus n. Each time a 0 bit is flipped to a 1 bit, the potential increases by 1. When a 1 bit is flipped to a 0 bit, the potential decreases by the number of 1 bit to the right of the flipped bit. The total amortized cost of n bit flips is n + O (n log n) due to the potential function, resulting in an amortized cost of O (1) per bit.