**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: 3/4/21 Time: 10:30-11:40 Max Marks: 10**

**Name: Mohammad Haris Zia Registration:FA19-BCS-037**



**Q1: What is the loop invariant for binary search? Explain. [3]**

**Binary Search:**

Binary search is a divide and conquer algorithm to search for a specific element in a list of elements.

**Time Complexity of binary search algorithm is O (log n).**

**Loop Invariant Conditions:**

* + The array is sorted in ascending order.
  + The Search Key is always present in the array.
  + Lower <= Higher.

**Explanation:**

* + **Initialization:**
    - Our array before the first iteration is always sorted in ascending order.
    - The search key is always present in the array before the first iteration.
    - Before the first iteration Lower index is always less than higher index
  + **Maintenance:**
    - Since our current array is sorted in ascending thus the sub array will also be sorted.
    - If the search key is present in the current array thus it will also be present in the sub array.
    - Lower index is still less than higher.
  + **Termination:**
    - During termination i.e., last iteration our sub array is still sorted.
    - Search key is found inside the array which means that the search key was present in the array.
    - Since search key was found our loop terminated thus lower is still lesser or equal to higher

**Q2: Dry run Bubble sort algorithm for the array elements (10,15,20,25). Compute the time complexity of this case and number of comparisons. [5]**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Iteration no**  0 | **10** | **15** | **20** | **25** | **No of Swaps**  **(Orange**  **Color)** | **No of Comparisons** |
| 1.1 | **10** | **15** | 20 | 25 | 0 | 1 |
| 1.2 | 10 | **15** | **20** | 25 | 0 | 1 |
| 1.3 | 10 | 15 | **20** | **25** | 0 | 1 |
| 1 | 10 | 15 | 20 | **25** | 0 | 3 |
| 2.1 | **10** | **15** | 20 | **25** | 0 | 1 |
| 2.2 | 10 | **15** | **20** | **25** | 0 | 1 |
| 2 | 10 | 15 | **20** | **25** | 0 | 2 |
| 3.1 | **10** | **15** | **20** | **25** | 0 | 1 |
| 3 | **10** | **15** | **20** | **25** | 0 | 1 |

**Time Complexity of Bubble Sort:**

**Standard Algorithm:**

In each iteration, we tend to make sequential comparisons and perform swapping whenever required. This algorithm continues to run for the N number of iterations even if the array is sorted which is average case.

The first iteration performs n-1 comparisons. The second performs n-2 till it reaches 1.

**Best Case:**

In this case, we check all the elements to see if there is any need for swaps. If there is no swapping still, we continue to make comparisons and complete N iterations.

Time complexity in Standard Bubble Sort:

(n-1) +(n-2) +(n-3) +…3+2+1=n(n-1)/2= **O(n2)**

**Optimized Binary Search Algorithm:**

In this algorithm, we assume that the array is already sorted and if at some iteration, no swaps occur for the entire array, we exit the loop and terminate as no further iterations are required because the array is already sorted.

**Best Case:**

In this case, we proceed with the first iteration to look for possible swaps and since the array is already sorted, there will be no swaps. We will exit the loop and the algorithm will terminate.

**Time complexity in Modified Bubble Sort:**

**(n-1) = O(n)**

**Q3: Name any algorithm whose best- and worst-case complexities are less than O(n^2). [2]**

**Merge Sort:**

* Merge Sort algorithm is an example of the algorithm whose best- and worst-**case complexities are less than O (n^2).**
* Merge Sort algorithm **uses Divide and Conquer Approach** to sort arrays.
* Time Complexity: Merge sort time complexity is **(n log n) in all cases (Best, Average, Worst).**