|  |  |
| --- | --- |
|  | **Design & Analysis of Algorithm**  **BSCS – 5 A**  **Department of Computer Science**  **Bahria University, Lahore Campus** |

**Solution Assignment: 1**

**Q1: Find out the growth rate using asymptotic analysis for the following pseudo segments, and write the Worst Case analysis.**

**a) [1 mark]**

for (int i = n; i > 1; i = fun(i)) {

// some O(1) expressions

}

The growth rate of the loop hinges on the behavior of the fun(i) function. If fun(i) consistently reduces i by a constant factor, the loop has a logarithmic growth rate (O(**log n**)). Otherwise, the growth rate can vary depending on the function's specific implementation.

**b) [1.5 mark]**

for (int i = 1; i <=m; i += c) {

// some O(1) expressions }

for (int i = 1; i <=n; i += c) {

// some O(1) expressions }

* The first loop iterates from 1 to m with a step of c.
* The second loop iterates from 1 to n with a step of c.

Therefore, the dominant factor in terms of growth rate is whichever upper bound (m or n) has the larger value. If m >= n, the growth rate is O(m), If n > m, the growth rate is O(n).

**c) [2.5 mark]**

**bubbleSort(array)**

**for i <- 1 to indexOfLastUnsortedElement-1**

**if leftElement > rightElement**

**swap leftElement and rightElement**

**end bubbleSort**

The bubble sort algorithm has a growth rate analysis of O(n^2) in the worst case, average case, and best case, where n is the number of elements in the array. This means the number of comparisons (and swaps) grows quadratically with the input size.

 Bubble sort makes several comparisons in each pass, even if the array is already sorted (best case) or partially sorted (average case).

 The number of comparisons needed is still proportional to n^2, resulting in the same **O(n^2)** growth rate.