Data Structures and Algorithms with Java
CS2105

Complexity analysis and Big-O Notation [2 Marks]

1. **Design an algorithm to determine how many times each distinct element appears in an array of size n.[1]**

   a. **Write the pseudocode for your algorithm.**

   b. **Analyze the worst-case time complexity and express it using Big-O notation.**

   c. **Implement your algorithm in Java and print the frequency of each element.**

Example (for illustration): If the input array is [2, 3, 2, 5, 3, 3], then the output should be

```
2 appears 2 times
3 appears 3 times
5 appears 1 time
```

2- **Time Complexity [1]**

Two algorithms solve the same problem:

Algorithm A

```
for (int i = 1; i <= n; i *= 2) {
    System.out.println("Hello");
}
```

Algorithm B

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < 10; j++) {
        System.out.println("Hello");
```

a. **Determine the time complexity of each algorithm using Big-O notation.**

b. **Which algorithm has the better growth rate?**

Submission Instructions

1. Submit your written answers in a PDF file, your Java source code as a .java file, and a screenshot demonstrating the program output.
2. A 0.5-point deduction will be applied in either of the following cases:
   a. Late submission.
   b. Absence without an accepted excuse.