Assignment Related to Time and Space Complexity

(Dr. GC Jana)

Assignment 1: Understanding Basic Concept

1.1. Define Algorithmic Complexity and explain its significance in computer science.
1.2. Differentiate between Time and Space Complexity with examples.
1.3. Provide a real-world example where time complexity is more crucial than space complexity and vice versa. Explain your reasoning.

Assignment 2: Big O Notation and Its Applications

2.1. Explain Big O Notation and its importance in analyzing algorithms.

2.2. Classify the following code snippets according to their Big O complexity:

```
for i in range(n):
    print(i)
```

```
for i in range(n):
    for j in range(n):
        print(i, j)
```

```
while n > 1:
    n = n // 2
```

2.3. Consider an algorithm with an $O(n^2)$ time complexity. If the input size is doubled, by what factor does the running time increase? Explain your answer.

Assignment 3: Omega and Theta Notations

3.1. Explain the difference between Omega (Ω) and Theta (Θ) notations with suitable examples.

3.2. Analyze the following algorithm and provide the Ω and Θ notations:

Algo:

```
def example_function(arr):
    min_val = arr[0]
    for i in arr:
        if i < min_val:
            min_val = i
    return min_val
```

What is the best-case and average-case time complexity of this algorithm?

3.3. Provide a scenario where Theta notation is more informative than Big O notation. Explain your reasoning.

Assignment 4: Practical Problem Solving

4.1. Write an algorithm to find the maximum element in an unsorted array. Analyze the algorithm's time and space complexity.

4.2. Compare the time complexities of Bubble Sort and Merge Sort. Which one would you prefer for a large dataset and why?

4.3. Design an algorithm that has a time complexity of O(log n). Provide a brief explanation of how your algorithm works.

Assignment 5: Case Study and Analysis

5.1. Select an algorithm (e.g., Quick Sort, Binary Search Tree operations) and perform a detailed analysis of its time and space complexities.

5.2. Discuss the worst-case, best-case, and average-case scenarios for your selected algorithm.

5.3. Suggest possible optimizations for the algorithm and analyze how these changes would impact the time and space complexities.