Algo Assignment

Analyzing the Growth of Functions and Running Time of Loops

1. Theoretical Questions

1.1. For each of the following functions, determine the asymptotic complexity in Big-O notation. Provide a clear explanation of why your answer is correct.

- (f(n) = 3n^3 + 5n^2 + 7)
- (g(n) = 2^{\sqrt{n}})
- (h(n) = n \log^2(n))
- (k(n) = n!)

1.2. Given two functions (f(n)) and (g(n)), where (f(n)) is said to be (O(g(n))), explain what this implies about the growth rates of (f(n)) and (g(n)). Provide an example with specific functions where (f(n)) is (O(g(n))) and one where (f(n)) is not (O(g(n))).

1.3. Consider the following function definitions. Determine whether they are polynomial, exponential, or neither. Provide your reasoning.

- (p(n) = 5n^2 + 3n + 1)
- (q(n) = 4^n)
- (r(n) = \log(n) . n)
- (s(n) = \sqrt{n} . 2^n)

2. Practical Coding Problems

2.1. Determine the time complexity of the following python code.

~~~python
def loop_count(n):
    count = 0
    for i in range(n):
        for j in range(i, n):
            count += 1
    return count
    ```
~~~

~~~
**2.2** Analyze the following code snippet and provide its time complexity:
```python
def nested_loops(n):
    total = 0
    for i in range(1, n):
        for j in range(i, n):
            for k in range(n):
                total += 1
    return total
~~~

2.3. Write a Python function to compute the number of basic operations performed by a nested loop structure with the following code, and determine its time complexity.

```python
def nested_count(n):
    total = 0
    for i in range(n):
        for j in range(n):
            for k in range(i + 1):
                total += 1
    return total
```

2.4. Given the following code snippet with two nested loops, analyze its time complexity:

```python
def double_nested_loops(n):
    total = 0
    for i in range(n):
        for j in range(2 * i, n):
            total += 1
    return total
```

2.5. Analyze the following code snippet and determine its time complexity:

```python
def multi_loops(n):
    total = 0
    for i in range(1, n):
        for j in range(1, i):
            for k in range(1, j):
                total += 1
    return total
```

3. Advanced Analysis

3.1. Prove or disprove that the function T(n)=5·2^n/2+3·n2 is O(2^n)

Submission Guidelines

- Provide clear and detailed explanations for all theoretical questions.
- Include Python code for practical problems, ensuring it runs correctly.
- Submit your solutions in a well-organized format, with code and explanations properly documented.