Name: _____ **Date:** _____ **Period:** _____

Big-O Notation Practice Worksheet

Instructions: For each problem, determine the tightest asymptotic bound (Big-O, Big-Θ, or Big-Ω) for the given function or algorithm. Show your reasoning for multi-part problems. For multiple-choice questions, circle the correct answer. Write your final answer in the space provided.

---

1. Determine the Big-O complexity of the following function: $f(n) = 3n^3 + 2n^2 + 5n + 1$. Provide a brief justification.

2. Sort the following functions in increasing order of asymptotic growth rate (from slowest to fastest): $n \log n$, 2^n, n^2, $\log n$, $n!$, n, $n^{0.5}$. Write your final ordering.

3. Consider the following code fragment:

```
for i = 1 to n:
    for j = 1 to i:
        print(i + j)
```

What is the Big-O time complexity of this code? Explain your reasoning.

4. True or False: $2^{n+1} = O(2^n)$. Justify your answer.

5. Multiple Choice: Which of the following is equivalent to $\Theta(n^2)$?

   (A) $O(n^2)$ and $\Omega(n^2)$

   (B) $O(n^2)$ only

   (C) $\Omega(n^2)$ only

   (D) Neither $O(n^2)$ nor $\Omega(n^2)$

6. Determine the Big-O complexity of the recurrence $T(n) = 2T(n/2) + n$. Assume $T(1) = 1$. Show your work using the Master Theorem or iteration method.

7. Given $f(n) = n^2 \log n$ and $g(n) = n^2\sqrt{n}$, determine whether $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, or $f(n) = \Theta(g(n))$. Justify your answer.

8. Multiple Choice: What is the Big-O complexity of the following function?

$$f(n) = \sum_{i=1}^{n} i^2$$

(A) $O(n)$
(B) $O(n^2)$
(C) $O(n^3)$
(D) $O(\log n)$

9. Prove or disprove: $n^3 + 100n^2 = \Omega(n^3)$. Provide a formal proof using the definition of Big-Omega.

10. Consider the algorithm that finds the maximum element in an unsorted array of size n. What is the tightest asymptotic bound for its worst-case time complexity? Explain why.

Answer Key

1. $O(n^3)$. The highest-order term is $3n^3$, so the function grows as n^3. All lower-order terms are dominated.

2. $\log n$, $n^{0.5}$, n, $n \log n$, n^2, 2^n, $n!$

3. $O(n^2)$. The outer loop runs n times, and the inner loop runs an average of $n/2$ times, giving approximately $n^2/2$ operations, which is $O(n^2)$.

4. True. $2^{n+1} = 2 \cdot 2^n$, so $2^{n+1} \leq c \cdot 2^n$ for $c = 2$ and all $n \geq 0$. Thus $2^{n+1} = O(2^n)$.

5. (A) $O(n^2)$ and $\Omega(n^2)$. By definition, $\Theta(n^2)$ means the function is both $O(n^2)$ and $\Omega(n^2)$.

6. $O(n \log n)$. Using the Master Theorem with $a = 2$, $b = 2$, $f(n) = n$, we have $\log_b a = 1$, and $f(n) = n = \Theta(n^1)$, so case 2 applies: $T(n) = \Theta(n \log n)$.

7. $f(n) = O(g(n))$. Compare: $n^2 \log n$ vs $n^2\sqrt{n} = n^{2.5}$. Since $\log n$ grows slower than $n^{0.5}$, $f(n)$ is asymptotically smaller, so $f(n) = O(g(n))$ but not $\Omega(g(n))$.

8. (C) $O(n^3)$. The sum $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$, which is $\Theta(n^3)$.

9. True. We need constants $c > 0$ and n_0 such that $n^3 + 100n^2 \geq cn^3$ for all $n \geq n_0$. Choose $c = 1$ and $n_0 = 1$: $n^3 + 100n^2 \geq n^3$ for all $n \geq 1$. Thus $n^3 + 100n^2 = \Omega(n^3)$.

10. $O(n)$. The algorithm must examine each element once to find the maximum, requiring $n - 1$ comparisons in the worst case, which is $\Theta(n)$.