Algorithms & Data Structures, Class № 3

Time complexity and Big-Oh notation

Problem 1

Work out the computational complexity of the following piece of code:

```
for( int i = n; i > 0; i /= 2 ) {
    for( int j = 1; j < n; j *= 2 ) {
        for( int k = 0; k < n; k += 2 ) {
            ... // constant number of operations
        }
    }
}
```

Problem 2

Assume that each of the expressions below gives the processing time T(n) spent by an algorithm for solving a problem of size n. Select the dominant term(s) having the steepest increase in n and specify the lowest Big-Oh complexity of each algorithm.

Expression	Dominant term(s)	$O(n)$
$5 + 0.001n^3 + 0.025n$		
$500n + 100n^{1.5} + 50n\,log_{10}n$		
$0.3n + 5n^{1.5} + 2.5n^{1.75}$		
$n^2\,log_2n + n(log_2n)^2$		
$nlog_3n + nlog_2n$		
$3log_8n + log_2log_2log_2n$		
$100n + 0.01n^2$		
$0.01n + 100n^2$		
$2n + n^{0.5} + 0.5n^{1.25}$		
$0.01nlog_2n + n(log2n)^2$		
$100nlog_3n + n^3 + 100n$		
$0.003log_4n + log_2log_2n$		

Problem 3

The statements below show some features of “Big-Oh” notation for the functions $f \equiv f(n)$ and $g \equiv g(n)$. Determine whether each statement is TRUE or FALSE and correct the formula in the latter case.

Statement	TRUE or FALSE?	If it is FALSE then write the correct formula
$O(f+g) = O(f) + O(g)$		
$O(f \cdot g) = O(f) \cdot O(g)$		
Jeżeli $g = O(f)$ i $h = O(f)$ to $g = O(h)$		
$5n + 8n^2 + 100n^3 = O(n^4)$		
$5n + 8n^2 + 100n^3 = O(n^2 log\, n)$		

Problem 4

Prove that $T(n) = a_0 + a_1 n + a_2 n^2 + a_3 n^3$ is $O(n^3)$ using the formal definition of the Big-Oh notation.
Hint: Find a constant c and threshold n_0 such that $cn^3 \geq T(n)$ for $n \geq n0$.