Algorithmic Analysis Practice Problems

Problem 1: Asymptotic Analysis
For each of the following, prove or disprove using the definitions of O, Ω, or Θ.

(a) Is $3^n = O(3^{n-1})$?

(b) Is $3^{3n} = O(3^n)$?

(c) Is $n\ log(n^3) = \Theta(n\ log(n))$?

(d) Is $2^{2n} = \Omega(3^n)$?

Problem 2: Ordering Functions

Given the following functions, arrange the functions in increasing order of growth. That is, arrange them as a sequence $f_1, f_2, ...$ Such that $f_1 = O(f_2), f_2 = O(f_3), ...$ Give a short explanation to your ordering.

$$f_1 = n^2$$
$$f_2 = n$$
$$f_3 = nlog(n)$$
$$f_4 = 2^n$$
$$f_5 = (log(n^2))^2$$

Problem 3: Ordering Functions [Take 2]
Given the following functions, arrange the functions in increasing order of growth. That is, arrange them as a sequence $f_1, f_2, ...$ Such that $f_1 = O(f_2), f_2 = O(f_3), ...$ Give a short explanation to your ordering.

$$f_1 = 4^{3n}$$
$$f_2 = 2^{n^4}$$
$$f_3 = 2^{3^{n+1}}$$
$$f_4 = 3^{3^n}$$
$$f_5 = 2^{5n}$$

Problem 4: Algorithm Analysis

Consider the following basic problem. You're given an array A consisting of n integers $A[1], A[2], \ldots, A[n]$. You'd like to output a two-dimensional n-by-n array B in which $B[i, j]$ (for $i < j$) contains the sum of array entries $A[i]$ through $A[j]$—that is, the sum $A[i]+A[i+1]+\cdots+A[j]$. (The value of array entry $B[i, j]$ is left unspecified whenever $i \geq j$, so it doesn't matter what is output for these values.) Below is a simple algorithm to solve this problem:

```
1  for i ← 1 to n
2        do for j ← i + 1 to n
3             Add entries A[i] through A[j]
4             Store the result in B[i, j]
```

(a) Give a function $f(n)$ that is an asymptotically tight bound on the running time of the algorithm above. Using the pseudocode above, argue that the algorithm is, in fact $\Theta(f(n))$.

(b) Although the algorithm you analyzed above is the most natural way to solve the problem, it contains some highly unnecessary sources of inefficiency. Give a different algorithm to solve this problem with an asymptotically better running time than the provided algorithm.

(c) What is the running time of your new algorithm?

Problem 5: Running Times

A given algorithm takes 2^n microseconds to solve an instance of size n (where n is an integer).

(a) What size instance could this algorithm solve in a year? There are $31556926 \times 10^6 \mu s$ in a year.

(b) Suppose you design a second algorithm that takes $10n^2$ microseconds to solve an instance of size n. What size instance can this second algorithm solve in a year?

(c) Suppose you design a third algorithm that takes $100n$ microseconds to solve an instance of size n. What size instance can this third algorithm solve in a year?

(d) Suppose instead of trying to improve our algorithm we stick with the original one that takes 2^n microseconds for an instance of size n and just wait for our machine to get faster. If the hardware is 10,000 times faster, what size instance could our original algorithm solve in a year?