CS383, Algorithms
Spring 2009
HW4

1. Suppose that \( f(n) \) and \( g(n) \) are strictly positive functions defined for all positive integers \( n \), such that
\[
f(n) = O(g(n))
\]  
For each of the following, state whether the given relationship is implied by the assumption in Eq. 1 alone or not. If you state that the stated relationship follows from Eq. 1, provide a proof. Otherwise, provide a specific counterexample.

(a) \( f(n) = \Theta(g(n)) \)

(b) \( \frac{1}{g(n)} = O\left(\frac{1}{f(n)}\right) \)

(c) \( f(n)^2 = O\left(g(n)^2\right) \)

(d) \( 2^f(n) = O\left(2^g(n)\right) \)

2. Algorithm 1 draws a recursive pattern (finite fractal) based on a given image. An example based on a photo of the Prague Astronomical Clock is displayed approximately in Fig. 1, where finer details have been suppressed (the recursion depth has been limited to three).

**Algorithm 1: Recursive Image Generation**

**Input:** An image \( im \), location coordinates \( x, y \), display size \( n \).

**Output:** Displays recursive image based on \( im \) of size \( n \times n \) at \( (x, y) \) (e.g., Fig. 1).

\[
\text{DRAW}(im, x, y, n)
\]

(1) \textbf{if} \( n > 0 \)

(2) \textbf{DRAW_ONCE}(im, x, y, n)

(3) \text{DRAW}(im, x, y + n, \lfloor n/2 \rfloor)

(4) \text{DRAW}(im, x + n, y, \lfloor n/2 \rfloor)

(5) \text{DRAW}(im, x, y - n, \lfloor n/2 \rfloor)

(6) \text{DRAW}(im, x - n, y, \lfloor n/2 \rfloor)

Let \( C(n) \) denote the total number of times that the \text{DRAW_ONCE} function is called by the invocation \text{DRAW}(im, x, y, n).
(a) Write a recurrence relation for $C(n)$. Include a base case. Explain how to derive the recurrence relation from the pseudocode for Algorithm 1.

(b) Solve the recurrence relation to find a big $\Theta$ expression for the number of drawOnce calls as a function of $n$. Explain.

(c) How would the recurrence relation for the number of drawOnce calls and the resulting big $\Theta$ growth rate change if there were five recursive calls to draw instead of four in Algorithm 1 (with the rest of the algorithm remaining the same)? Explain.

3. You are designing a divide and conquer algorithm for a certain computational task. Your approach will involve dividing an input instance of size $n$ into a certain number of subproblems, each of size $n/3$. Assuming that the gluing time needed to recombine the subsolutions is $\Theta(n^2)$, what is the largest number of subproblems for which the overall running time of the algorithm on instances of size $n$ will be $O(n^2)$? Give a concise explanation, and state your answer clearly.

4. Consider the task of detecting whether a given array has an element that is repeated in more than half of the positions of the array. For example, the value 2 is the majority element in the array $\{3, 2, 5, 2, 3, 2, 7, 2, 2\}$, while the array $\{5, 8, 8, 3, 10, 8, 5, 8\}$ has no majority element. In the present task you will develop a divide and conquer algorithm for solving this problem, and you will analyze its running time.

(a) Suppose that an array $a[1...n]$ has an element $v_L$ that occurs in strictly more than half of the first $\lfloor n/2 \rfloor$ positions of $a$, and an element $v_H$ that occurs in strictly more than half of the remaining $\lceil n/2 \rceil$ positions. Argue carefully why if there is an element $v$ of $a$ that occurs in over half of all $n$ positions of $a$, then $v$ must be either $v_L$ or $v_H$. Note that it
would not be enough for \( v \) to occur more times in \( a \) than either \( v_L \) or \( v_H \). We explicitly require that \( v \) occur in strictly more than half of all positions of the entire array.

(b) Using 4a, design a divide and conquer algorithm that returns the majority element of an array or the value NO_SUCH_ELEMENT as appropriate. Give detailed pseudocode.

(c) Analyze the running time of your algorithm from 4b. Provide a careful explanation, including the relevant recurrence relation.