MA/CSSE 473 – Design and Analysis of Algorithms **In-class Quiz Day 03**

Name:_____ Score:____/ 16 circle your Section # 01(3rd) 02 (4th)

1. (1) With another student, try to write a precise definition of "f(n) is in O(g(n))"

   Any answer gets credit. Here is a good one:
   We say that **f(n) ∈ O(g(n))** iff there exist two positive constants **c** and $\mathbf{n_0}$ such that
   for all $n \geq n_0$, $f(n) \leq c\, g(n)$

2. (1) Prove using the formal definition: If $f(n) \in O(h(n))$ and $g(n) \in O(h(n))$, then $f(n)+g(n) \in O(h(n))$

   Since $f(n) \in O(h(n))$, there are constants n_1 and c such that for all $n \geq n_1$, $f(n) \leq c\, h(n)$. Similarly,
   Since $g(n) \in O(h(n))$, there are constants n_2 and d such that for all $n \geq n_2$, $g(n) \leq c\, h(n)$.
   Let $n_3 = \max(n_1, n_2)$, and let $b = c+d$. Then for all $n \geq n_3$, $f(n) + g(n) \leq b\, h(n)$. So $f(n)+ g(n) \in O(h(n))$.

3. (3) For each of the cases of $\lim_{n\to\infty} \frac{f(n)}{g(n)}$
   What can you conclude about asymptotics?

   Limit is 0 ➔ f(n) is O(g(n)), but f(n) is not Θ(g(n)).

   Limit is a non-zero positive constant ➔ f(n) is Θ(g(n)).

   Limit is infinite ➔ f(n) is Ω(g(n)), but f(n) is not Θ(g(n)).

4. (1) How many matrix multiplications are needed in order to compute the nth Fibonacci number using the matrix approach? Explain briefly.
   For each recursive call in the code, at most two multiplications get done, thus ≤ 2 lg n [lg n means $\log_2 n$]

5. (1) Why can't we just compute Fibonacci numbers by using the formula (seen in CSSE 230) that uses powers of the golden ratio?
   Taking powers of floating point numbers that are stored with enough accuracy to compute large Fibonacci numbers is prohibitively expensive.

6. (1) When we add three 1-digit integers, how many digits can be in the answer? Is this independent of the base (i.e, the same for decimal, binary, hexadecimal, etc.).
   At most 2 digits. Yes, independent of base.

7. (1) How does the previous question apply to the analysis of the addition of two n-bit non-negative integers?

   There can be at most one digit in the carry from each digit addition, so the amount of work for each digit of the answer is constant time. Thus the whole algorithm is O(n), where n is the number of bits.

8. (1) What is the running time of the "standard" algorithm for multiplying multiple-digit numbers?
n^2

9. (1) What is the running time of the "European" algorithm for multiplying multiple-digit numbers?

n^2

10. (1) What is the recurrence for the first Divide and Conquer multiplication algorithm?

$T(n) = 4T(n/2) + O(n)$.

What is its solution?

Solution is case 3 of Master Theorem.
$\text{Log}_2 4$ is 2, so it is n^2.

11. (1) Gauss's algorithm for multiplying two complex numbers replaces __4__ integer multiplications by _3__.

12. (1) What is the recurrence for the Gaussian Divide and Conquer multiplication algorithm?

$T(n) = 3\ T(n/2) + \Theta(n)$

What is its solution?

In Master theorem, a = 3, b = 2, k = 1.
So we have the third case: $T(N) = \Theta(n^{\lg 3})$,
Which is approximately $n^{1.59}$.

13. (1) Tell me about anything from today's lecture that you found confusing or feel that we need to spend more time on?. (or write N/A).

Any answer is okay; must write something.

14. (1) What questions do you have (from today's lecture, from the reading, or form the course in general), or write N/A.
Any answer is okay; must write something.