COMP 2119A Introduction to Data Structures and Algorithms
Assignment One
Due Date: 04 October 2016 5:00pm

Assignment box: A3 (also submit a scanned copy to moodle)

[Note that for the questions involving algorithm design, before you present your algorithm, try to describe your idea first! More marks will be given to faster algorithms.]

1) [26%] Prove or disprove each of the followings.
   a) $100n^2 + 20n = \Theta(1000n)$.
   b) $99n^2 = O(n^2)$.
   c) $0.5n^3 = \Omega(n^{2.999})$.
   d) $O(\log n) + O(n) = \Theta(n)$.
   e) $O(0.0001n) = O(1)$.
   f) $O(1) + O(1) + \ldots + O(1)$ [9,000,000 terms] $= O(1)$.
   g) $\min\{f(n), g(n)\} = O(f(n) + g(n))$, where $f(n)$ and $g(n)$ are positive functions.
   h) $\min\{f(n), g(n)\} = \Theta(f(n) + g(n))$, where $f(n)$ and $g(n)$ are positive functions.
   i) $6000n + \Theta(n^2) = \Omega(n^2)$.
   j) If $f(n) = O(n)$, then $2^{f(n)} = O(2^n)$, where $f(n)$ is a positive function.
   k) $105n = o(n^2)$.
   l) $\log_{200} n = o(\log_2 n)$.
   m) If $f(n) = \omega(g(n))$, then $\log_2(f(n)) = \omega(\log_2 g(n))$, where $f(n)$ and $g(n)$ are positive functions.

2) [14%] Rank the following functions by order of growth and partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$. NO NEED to prove your answer. [log n stands for $\log_2 n$.]

$(\sqrt{2})^{\log n}$	9999^{999}	2^n	$200n$
$2^{\log n}$	2^{3n}	n^{ε} $(0 < \varepsilon < 0.5)$	$\frac{n}{\log\log n^2}$
$(n + \sqrt{n}\,)\log n$	1.001^n	$\log n$	$\log_{30} n$

3) [10%] Assume that we have 12 coins of which one of them is counterfeit and is either heavier or lighter than others. You are given a balance, which can only tell which side of the balance is heavier or indicate that both sides are of equal weight. Design an algorithm using only 3 weightings to identify the counterfeit coin and tell whether it is heavier or lighter.

   [Try to outline the key idea(s) on how you solve the problem before you present the pseudocode. Hint: You may need to make use of real coins to help once you identify them from the given coins.]

4) [10%] Consider the following algorithm:

```
F(A, B) {                    // A, B are two positive integers with B > 1
        while A > 0 do
                print (A mod B)
                A = A div B
}
```

a) [2%] What will be the output if (i) $A = 10$, $B = 2$; (ii) $A = 134$, $B = 6$

b) [3%] Guess what the algorithm does.

c) [5%] Give an asymptotically (worst case) tight bound for the running time of the algorithm. Explain your answer.

5) [15%]

We have m coins on the table, with labels from 1 to m. All coins show the head initially. At each round, we will select some coins to flip (i.e., from head to tail or from tail to head) according to the following rule. At the i-th round ($i = 1, 2, \ldots, m$), we flip the coins whose labels are multiple of i, and we are interested to know how many coins show the head after the m-th round.

We can compute the solution using the following code:

```
Initialize A[1..m] = 0 (0: head, 1: tail);
for (round i = 1, 2, …, m) {
        for (position j = i, 2i, 3i, …) {
                if (j ≤ m) A[j] = abs(A[j] – 1);   // abs – absolute value
                else     break;
        }
}
return(the number of zeros in A);
```

(a) [10%] Show that the above code runs in $\Theta(m \log m)$ time. If you can show that the above code runs in $O(m^2)$ time, you can get partial credits.

(b) [5%] Design a faster algorithm that can solve the same problem in $O(m)$ time.

6) [10%] Let $f(n) = n$. Find the error in the following "proof" by mathematical induction that $f(n) \in \Omega(n^2)$.

- Basis: The case for $n = 1$ is trivially satisfied since
  $f(1) = 1 = cn^2$, where $c = 1$.
- Induction Step: Consider any $n > 1$. Assume by the induction hypothesis the existence of a positive constant c such that $f(n\text{-}1) \geq c(n\text{-}1)^2$.

Then, $f(n) = n = (n\text{-}1) + 1 = f(n\text{-}1) + 1$
$\geq c(n\text{-}1)^2 + 1 = cn^2 - 2cn + c + 1$
Take $c \leq 1/(2n - 1)$, $-2cn + c + 1 \geq 0$, so $f(n) \geq cn^2$

Thus we have shown that no matter how large n is, we can always find a value of c such that $f(n) \geq cn^2$. It follows that $f(n) \in \Omega(n^2)$.

7) [10%] Define a number sequence G_i ($i \geq 0$) as follows. $G_0 = 0$; $G_1 = 1$; $G_2 = 1$; $G_i = G_{i-1} + G_{i-2} + G_{i-3}$ for all $i > 2$. Design a recursive algorithm to compute G_n ($n \geq 0$) and analyze the time complexity of your algorithm. If you can only give a non-recursive algorithm together with its time complexity, you can get partial credits.

8) [5%]

(a) Is the assignment (1) too difficult; (2) too easy; (3) about right?

(b) How many hours you spend on the assignment?

(1) Less than 5 hours
(2) 5 – 10 hours
(3) 10 – 20 hours
(4) More than 20 hours

(c) [Self assessment] Do you consider yourself understand the topics of this assignment?
(1) Yes; (2) not 100% sure; (3) No
If your answer is (2) or (3), please elaborate (at least indicate which part you do not understand).

(d) Other comments?

--- End of Assignment ---