Design and Analysis of Algorithms 2009
(Practice Problems 1)

October 2, 2009

- You do not need to submit solutions to these problems.
- Some of these problems are likely to be discussed on Oct. 5 in the class.
- You are expected to try all of them.

1. Let $H_n = 1 + \frac{1}{2} + \frac{1}{3} \cdots \frac{1}{n}$. Show that $H_n = \Theta(\log n)$
2. Show that $\log(n!) = \Theta(n \log n)$
3. Show that for any base $b > 2$, the sum of any three single digit numbers is at most two digits long.
4. Suppose we have available a black box $\mathcal{B}(.)$, which when given as input an n bit integer a returns a^2 in $O(n)$ time. Use this black-box $\mathcal{B}(.)$ to multiply two n bit numbers a and b in $O(n)$ time.
5. Prof. Calculus claims that there is an algorithm for squaring integers which is asymptotically faster than multiplying two integers. Argue that this claim of Prof. Calculus is false.
6. Compute the following:
   (a) 3^{80} mod 5.
   (b) The multiplicative inverse of 26 modulo 677.
   (c) $4^{200} - 9^{100}$ mod 35
   (d) $3^{3^{100}}$ mod 5 (as usual a^{b^c} means a raised to b^c).
7. Prove that for every integer x, either $x^2 \equiv 0 \bmod 4$ or $x^2 \equiv 1 \bmod 4$.
8. Solve the following system of congruences

$$\begin{aligned} x &\equiv 10 (\mathrm{mod} 11) \\ x &\equiv 11 (\mathrm{mod} 12) \\ x &\equiv 12 (\mathrm{mod} 13) \end{aligned}$$

9. Show that the following rule is true

$$\gcd(a,b) = \begin{cases} 2\gcd(\frac{a}{2}, \frac{b}{2}) & \text{if } a, b \text{ are even} \\ \gcd(a, \frac{b}{2}) & \text{if } a \text{ is odd and } b \text{ is even} \\ \gcd(\frac{a-b}{2}, b) & \text{if } a, b \text{ are odd} \end{cases}$$

10. Using the above rule give an efficient algorithm to compute $\gcd(a, b)$. Discuss the complexity of your algorithm.

11. Given two heaps A and B each of size n, give an efficient algorithm to merge these two heaps into a single heap C.

12. You are given an array A of size n which contains only zeros and ones. Give a linear time algorithm to sort A.

13. Given two arrays A and B of size n and a number c, design an algorithm which decides whether there exists $i, j \in \{1, 2, \ldots, n\}$, such that $A[i] + B[j] = c$. Your algorithm should run in $O(n \log n)$ time.