Solutions to Homework 1

Debasish Das
EECS Department, Northwestern University
ddas@northwestern.edu

1 Problem 1

The algorithm will print GCD and LCM of X and Y.
To prove: GCD(X,Y) is given by $\frac{x+y}{2}$ while LCM(X,Y) is given by $\frac{u+v}{2}$
Proof: Invariant for GCD computation are $x > 0 \wedge y > 0 \wedge GCD(X, Y) = GCD(x, y)$. Verify both steps maintain this invariant using the number theory result that GCD(x,y) = GCD(x-y,y) = GCD(x,y-x). At termination we have $x = y$ and therefore $GCD(x, y) = GCD(x, x) = \frac{x+x}{2} = \frac{x+y}{2}$.
Invariant for LCM computation is $uy + vx$. Verify that both steps of algorithm maintain $uy+vx$ as invariant. Therefore from the initialization conditions when algorithm terminates $uy + vx = XY + XY$. Now since $x = y = GCD(X, Y)$ we get $\frac{u+v}{2} \times GCD(X, Y) = XY$. Using the number theory result we know that $\frac{u+v}{2}$ is the LCM.

2 Problem 2(0.1)

(a)n-100 = Θ(n-200)
(b)$n^{\frac{1}{2}}$ = O($n^{\frac{2}{3}}$)
(c)100n+log(n) = Θ(n + log$(n)^2$)
(d)nlogn = Θ(10nlog10n)
(e)log2n = Θ(log3n)
(f)10logn = Θ(log(n^2)
(g)$n^{1.01}$ = Ω(nlog^2(n))
(h)$\frac{n^2}{\log n}$ = Ω(n($\log n^2$))
(i)$n^{0.1}$ = Ω($(\log n)^{10}$)
(j)$(\log n)^{\log n}$ = Ω($\frac{n}{\log n}$)
(k)$n^{\frac{1}{2}}$ = Ω($\log n^3$)
(l)$n^{\frac{1}{2}}$ = Θ($5^{\log n}$)
(m)$n2^n$ = O(3^n)
(n)2^n = Θ(2^{n+1})
(o)$n!$ = Ω(2^n)
(p)$\log n^{\log n}$ = O($2^{\log n^2}$)
(g)$\sum_{i=1}^{n} i^k$ = O(n^{k+1})

3 Problem 3(0.2)

The given function $g(n)$ turns out to be a geometic series and it evaluates to

$$g(n) = \begin{cases} \frac{c^{n+1}-1}{c-1} & \text{if c} > 1 \\ \frac{1-c^{n+1}}{1-c} & \text{if c} < 1 \\ n & \text{if c} = 1 \end{cases} \tag{1}$$

(a) When $c < 1$, lower bound on Equation 1 can be obtained by analyzing the case when $n = 0$ which is 1. Similarly upper bound is obtained by analyzing the case when $n \to \infty$ which comes out to be $\frac{1}{1-c}$. Since we got two constants bounding Equation 1 g(n) is $\Theta(1)$
(b) When $c > 1$ similar analysis will produce upper and lower bounds as c^n which makes g(n) as $\Theta(c^n)$
(c) When $c = 1$, the bound of $\Theta(n)$ is straight forward as each term of $g(n)$ evaluates to 1 and there are n terms.

4 Problem 4(0.3)

(a) Base case : Fib[6] $= 8 \geq 2^3$
Hypothesis : Fib[i] $\geq 2^{i/2} \forall i \in (6, .., k)$
Induction : Fib[k+1] = Fib[k] + Fib[k-1] $\geq 2^{k/2}(1 + \frac{1}{\sqrt{2}})$
Fib[k+1] $\geq 2^{k/2} \times \sqrt{2}$
Therefore Fib[k+1] $\geq 2^{\frac{k+1}{2}}$
(b)Using the generating function derivation shown in class Fibonacci numbers can be represented as

$$Fib[n] = \frac{1}{\sqrt{5}}(\phi^n - \varphi^n) \tag{2}$$

where $\phi = \frac{1+\sqrt{5}}{2}$ and $\varphi = \frac{1-\sqrt{5}}{2}$ Now choosing c as $\log \phi$ we need to prove that Fib[n] $\leq \phi^n$. We use induction to prove that.
Base case : Fib[0] $= 1 \leq \phi^0 = 1$
Hypothesis : Fib[i] $\leq \phi^i \ \ \forall i \in (0, .., k-1)$
Induction : $\phi^n = \phi^2 \phi^{n-2} = (1 + \phi)\phi^{n-2}$
Using induction hypothesis $\phi^n \geq Fib[n-1] + Fib[n-2] = Fib[n]$
Therefore c is given as $\log \phi$
(c)Any number in between 0.5 and $\log \phi$ will suffice. Largest is $\log \phi$.

5 Problem 5(1.31)

(a) N is an n-bit number. N! is given by 1.2.3...N.
Upper Bound : Assuming each number 1,2,3,...,N is represented by n bits, the result of multiplying N n-bit number will give a number of $N * n$ bits where $N = 2^n$. Hence it is O(N*n)
Lower Bound : Since each number $i \in (1, 2, ..., N)$ can be optimally represented

by $\log i$ bits, total number of bits in N! is given by $\sum_{i=1}^{N} \log i$ which is $\log N!$. Using Sterling's approximation or using a factor argument we know $N! \geq \frac{N}{2}^{\frac{N}{2}}$ which implies that total number of bits in N! is lower bounded by $N \log N$. It turns out to be Ω(N*n). Combining both we get Θ(N*n)
(b) A simple iterative algorithm to solve the problem is given by:

```
Input :  N
Output :  N!
prod = 1
for i = 1 to N
  prod = prod * i
return prod
```

Complexity analysis:We present a worst case bound. Assuming each of the number 1,2,3,...,N are n-bit long each multiplication computes product of a $i \times n$ bit number with n bit number. Therefore total time taken by the for-loop is given by $\sum_{i=1}^{N}(n \times in)$ which turns out to be O($N^2 \times n^2$)

6 Problem 6(1.32)

Given numbers X and Y, apply Euclid algorithm (page 20) to compute GCD(X,Y). Followed by that get LCM(X,Y) by $\frac{X \times Y}{GCD(X,Y)}$. GCD computation takes O(n^3) where n are the number of bits in X, Y. Final LCM computation takes O(n^2). Therefore the algorithm is bounded by O(n^3).