Homework #2

Chris Dellomes
Professor: Ray Toal
CMSI 282: Algorithms
Loyola Marymount University

March 3, 2015

1. (a) n - 100 = Θ(n - 200)
   (b) $n^{1/2} = O(n^{2/3})$
   (c) 100n + log(n) = Θ(n + $log(n)^2$)
   (d) nlog(n) = Θ(10nlog(10n))
   (e) log(2n) = Θ(log(3n))
   (f) 10log(n) = O(log(n^2))
   (g) $n^{1.01}$ = Ω(n$(log(n))^2$)
   (h) n^2 / log(n) = Ω(n$(log(n))^2$)
   (i) $n^{0.1} = \Omega((log(n))^{10})$
   (j) $(log(n))^{log(n)}$ = Ω(n / log(n))
   (k) $n^{1/2} = \Omega((log(n))^3)$
   (l) $n^{1/2} = O(5^{log_2(n)})$
   (m) n$2^n = O(3^n)$
   (n) $2^n = \Omega(2^{n+1})$
   (o) n! = $\Omega(2^n)$
   (p) $(log(n))^{log(n)} = O(2^{(log_2(n))^2})$
   (q) $\sum\limits_{i=1}^{n} i^k = \Theta(n^{k+1})$

2. (a) $\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$

   Each entry in the matrix is calculated using 2 multiplications and 1 addition. Since there are 4 entries, that results in 8 multiplications and 4 additions.

   (b) For x^n, let n = 2^k for some positive integer k. Thenm we would calculate x^2 by repeatedly squaring.
   $x^2, x^4, ..., x^{2^k} = x^n$
   By squaring x to reach x^n, the exponent is doubled at each instance. This yields k = log(n) multiplications.

3. For a number, n, there are log_2(n + 1) binary digits and log_{10}(n + 1) decimal digits. Through conversion, we find that
   $log_{10}(n+1) = log_2(n+1)log_2(10) = 3.32log_2(n+1)$
   $log_{10}(n+1) = 4log_2(n+1)$

4. n! = (n)(n - 1)(n - 2) ... (1)
   n^n = (n)(n) ... (n)
   $n! \leq n^n$
   $(n/2)^{n/2} = ((n/2)^n)^{1/2} = (n^n/(2^n)^{1/2}$
   $(n/2)^{n/2} \leq n! \leq n^n$
   (n/2)log(n/2) $\leq$ log(n!) $\leq$ nlog(n)
   (1/2)(nlog(n) - n) $\leq$ log(n!) $\leq$ nlog(n)
   n! = Θ(nlog(n))

5. $x^{(5-1)(7-1)} = x^{(4)(6)} = x^{24}$
   $x^{24} \equiv 1mod35$
   $4^{1536} = (4^{64})^{24}$
   $(4^{64})^{24} \equiv 1mod35$
   $9^{4824} = (9^{201})^{24}$
   $(9^{201})^{24} \equiv 1mod35$
   $4^{1536} \equiv 9^{4824}mod35$
   $35 \mid (4^{1536} - 9^{4824})$

6. 31 is prime
   $x^{30} \equiv 1mod31for1 \leq x < 31$
   $5^{30000} = (5^{1000})^{30} \equiv 1mod31$
   $6^{123456} = 6123450 \cdot 6^6 = (6^{4115})^{30} \cdot 6^6$
   $(6^{4115})^{30} \equiv 1mod31$
   $6^6 = 46656 \equiv 1mod35$
   $(5^{30})^{1000} - ((6^{30})^{4115} \cdot 6^6) \equiv 1mod31$

7. Let b = 15. The given equaring algorithm gives us $a^{15} = a \cdot a^2 \cdot a^4 \cdot a^8$
   $a^{15} = a \cdot (a \cdot a) \cdot (a^2 \cdot a^2) \cdot (a^4 \cdot a^4)$
   This is a total of 6 multiplications. To find the true minimum number of multiplications, we first calculate $a^3 = a \cdot a \cdot a, a^6 = a^3 \cdot a^3, a^{12} = a^6 \cdot a^6$. Then we calculate $a^{15} = a^{12} \cdot a^3$. This shows the calculation can be done in 5 multiplications.

8. $2^{126} \equiv 1$ mod 127 by Fermat's little theorem.
   $2^{125} \cdot 2 \equiv 1$ mod 127
   Thus, 2^{125} is the inverse of 2 mod 127.
   Notice that $2^6 \cdot 2 = 128 \equiv 1$ mod 127.
   Therefore, $2^{125} \equiv 2^6$ mod 127

9. Given two n bit numbers, the running time for the algorithm used is $O(n^3)$.

```
def lcm(x, y):
        return (x * y) / gcd(x,y)

def gcd(x, y):
        while(y):
                x, y = y, x % y
        return x
```

10. Basing a primality test on Wilson's theorem would require calculating a factorial, which would be less efficient in terms of time complexity when compared to Fermat's theorem.

11. The time complexity for the program is $O(n^3)$ where n is the number of bits input.

```
def exponentMod(b, c, p):
        return (b ** c) % (p - 1)

def primaryMod(a, b, c, p):
        return (a ** exponentMod(b, c, p)) % p
```