CSCI 3110 Assignment 2 Solutions

October 13, 2012

1.4 (2 pts) Show that

$$\log(n!) = \Theta(n \log n).$$

To show the upper bound, we compare $n!$ with n^n. By definition, $n! = \prod_{i=1}^{n} i$ while $n^n = \prod_{i=1}^{n} n$ so $n! \le n^n$ for all $n \ge 1$. This implies that $\log(n!) \le \log(n^n) = n \log n$ for all $n \ge 1$ and, thus, $\log(n!) = O(n \log n)$.

Lower Bound (Method 1):
$n! = (1 \cdot 2 \cdot \ldots n) \ge (\lfloor n/2 \rfloor) \cdots n \ge (n/2) \cdots (n/2)$ $(n/2$ terms$) = (n/2)^{n/2}$
Hence: $\log(n!) \ge (n/2)\log(n/2) = (1/2) \cdot n(\log n - log 2) = (1/2) \cdot (nlogn - 1)$ which is $\Omega(n \log n)$

Lower Bound (Method 2): Compare $n!$ with $(n/2)^{n/2}$. $n! = (1 \cdot 2 \cdot \ldots n) =$ $((1 \cdot n) \cdot (2 \cdot (n-1)) \cdot \ldots \cdot (\lceil n/2 \rceil)$ if n is odd and is times $\lfloor n/2 \rfloor$ if n is even, so $n! = \Omega((n/2)^{n/2}$, and, thus $\log(n!) = \Omega(\log((n/2)^{n/2}))$. Since $\log((n/2)^{n/2}) = (n/2)\log(n/2) = \Theta(n \log n)$, we have that $n! = \Omega(n \log n)$.

1.31 (3 pts) Consider the problem of computing $N! = 1 \cdot 2 \cdot 3 \ldots N$.

(a) If N is a n-bit number, how many bits long is $N!$, approximately in $\Theta(\cdot)$ form)?

$N!$ is $\Theta(\log_2(N!))$ bits long. From 1.4, we know that $\log_2(N!) = \Theta(N \log_2 N)$, so $N!$ is $\Theta(N \log_2 N)$ bits long. Since N is an n-bit number, this can also be written as $\Theta(n2^n)$ bits long.

(b) Give an algorithm to compute $N!$ and analyze its running time.

```
FACTORIAL(N)
if (N = 0)
        Return 1
else
        Return N · FACTORIAL(N − 1)
```

This algorithm directly computes $N!$ using its definition. The algorithm runs for $\Theta(N)$ iterations, and does a multiplication of two $O(N \log N)$-bit numbers and some $O(N \log N)$ time work at each iteration. Thus, the running time is $O(NM(N \log N))$, where $M(N \log N)$ is the time required to multiply two numbers with $O(N \log N)$ bits.

1.8 (4 pts) Justify the correctness of the recursive division algorithm given in page 15, and show that it takes time $O(n^2)$ on n-bit inputs. Proof by Induction on x.

Base case: $x = 0$, alg. returns $q = 0, r = 0$

Inductive hypothesis: The recursive division algorithm works correctly for $0‘x < X$,

i.e. alg. correctly returns (q, r) such that $\lfloor X/2 \rfloor = qy + r$

Inductive step: If X even; then $X = 2\lfloor X/2 \rfloor = 2qy + 2r$ and alg. returns $Q = 2q$ and $R = 2r$ if $R > y$ alg returns instead $Q = 2q + 1$ and $R = 2r - y$

If X odd; alg. returns $X = 2\lfloor X/2 \rfloor + 1 = 2qy + 2r + 1$ nd alg. returns $Q = 2q$ and $R = 2r + 1$ again if $R > y$ alg returns instead $Q = 2q + 1$ and $R = 2r + 1 - y$

The algorithm terminates after n recursive calls, because each call halves x, reducing the number of bits by one. Each recursive call requires a total of $O(n)$ bit operations, so the total time taken is $O(n^2)$

1.19 (3 pts) The *Fibonacci numbers* $F_0, F_1, \ldots$ are given by the recurrence $F_{n+1} = F_n + F_{n-1}, F_0 = 0, F_1 = 1$. Show that for any $n \geq 1$, $\gcd(F_n{+}1, F_n) = 1$

We will prove this by induction. Our base case is $n = 1$, where $\gcd(F_2, F_1) = \gcd(1, 1) = 1$. Now, assume that the claim holds for all $1 \leq n \leq k$. By the definition of F, $\gcd(F_{k+1}, F_k) = \gcd(F_k + F_{k-1}, F_k)$. By definition, this is the largest number d such that $d|(F_k + F_{k-1})$ and $d|F_k$. Then $xd = F_k + F_{k-1}$ and $yd = F_k$, for some integers $y > x$. Subtracting these two equations gives that $(x - y)d = F_{k-1}$, so we also have that $F_{k-1}|d$. Since $\gcd(F_k, F_{k-1}) = 1$, by the inductive hypothesis, it must be the case that $d = 1$.

1.27 (3 pts) Consider an RSA key set with $p = 17$, $q = 23$, $N = 391$, and $e = 3$ (as in Figure 1.9). What value of d should be used for the secret key? What is the encryption of the message $M = 41$?

The value of d should be the inverse of $e \mod (p-1)(q-1)$, calculated by the extended Euclid algorithm. Following the algorithm forward, we get:

$$\begin{aligned} \gcd(e, (p-1)(q-1)) &= ex + (p-1)(q-1)y \\ \gcd(3, 352) &= 3x + 352y \\ 352 &= 3(117) + 1 \\ 3 &= (1)(3) + 0 \end{aligned}$$

Now, substituting backwards to find d, we get:

$$1 = -117(3) + 352$$

So, tghe multiplicative inverse of 3 mod 352 is -117. Which is in the same equivalence class as -117 + and multiple of 352:

$$\begin{aligned} -117 + 352 &= 235 \\ 1 &= 3(235) - 2(352) \\ d &= 235 \end{aligned}$$

The encryption of the message $M = 41$ should be

$$\begin{aligned} y &= M^e \mod N \\ &= 41^3 \mod 391 \\ &= 68291 \mod 391 \\ &= 105. \end{aligned}$$

1. greedyGCD

(a) Algorithm

```
GREEDYGCD(a, b)
if (b == 0)
    Return a
r = MIN(a mod b, b - a mod b)
Return greedyGCD(b, r)
```

2. Correctness
   Similar to textbook page 30. Note that $gcd(a, b)$ is the same as $gcd(a, -b)$

3. Running time
   (Refer to the Euclid Complexity hand-out - for details, I will simply highlight the differences here.)
   We have as before, $r < b$ and hence $2r < b + r \leq a$
   BUT we now have, due to the greedy choice that ensures: $r \leq b/2$
   Add r to each side: $\frac{3r}{2} \leq \frac{b+r}{2}$ using the original inequalities $\leq \frac{a}{2}$
   Hence $3r \leq a$. Multiply by b; giving $3rb \leq ab$ or $rb \leq \frac{ab}{3}$
   So, here we have the product of the arguments is a THIRD of the product of the arguments previous calls. The rest of the analysis follows that of the normal Euclid - except everything is $\log_3$.
   (A tighter argument seems possible. I think, it is possible to make everything work out such that $rb \leq \frac{ab}{4}$ but as the rabbit said: "I'm late! I'm late!")