University of Nevada, Las Vegas

University of Nevada, Las Vegas Computer Science 477/677 Fall 2019

Answers to Assignment 2: Due Wednesday September 4, 2019

I made an important mistake in my lecture when I told you that the secret key should be chosen so that $de \,\%\, N = 1$ **That is incorrect.**

The correction is: if $N = pq$ where p and q are primes, then the public key e must be relatively prime to $(p-1)(q-1)$, and the private key d must be chosen so that $de \,\%\, (p-1)(q-1) = 1$

Here are some additional helpful facts.

- If N is prime and $gcd(N, a) = 1$, *i.e.* a is relatively prime to N, then $a^{N-1} \,\%\, N = 1$.
- If $N = pq$, where p and q are large primes, and if a is relatively prime to N, then $a^{(p-1)(q-1)} \,\%\, N = 1$.

1. (a) Work problem 1.11 on page 39 of the textbook.

   The question reduces to whether $4^{1536} \,\%\, 35 = 9^{4824} \,\%\, 35$.

   $35 = pq$ where $p = 5$ and $q = 7$ are primes. Thus, $\phi(35) = (p-1)(q-1) = 24$. If n is relatively prime to 35, then $n^{24} \,\%\, 35 = 1$, and $n^m \,\%\, 35 = n^{m \,\%\, 24} \,\%\, 35$ for any m. Since $1536 \,\%\, 24 = 0$, we have $4^{1536} \,\%\, 35 = 4^0 \,\%\, 35 = 1$. Since $9834 \,\%\, 24 = 0$, we have $9^{4824} = 9^0 \,\%\, 35 = 1$. Hence the answer is yes.

   (b) Work problem 1.14 on page 39 of the textbook, which is to design a fast algorithm for computing $F_n \,\%\, p$, where F_n is the n^{th} Fibonacci number and p is a positive integer. The case $p = 1$ is trivial, hence we assume $p \geq 2$. We let $\overline{F}_n = F_n \,\%\, p$ for short. Here are three algorithms for the problem.

   i. Dynamic programming. $\overline{F}_1 = \overline{F}_2 = 1$ and $\overline{F}_n = \overline{F}_{n-2} + \overline{F}_{n-1}$ for all $n \geq 2$. The dynamic program has $O(n)$ steps, and each step takes $O(\log p)$ time. Thus the dynamic program computes $\overline{F}_n$ is $O(n \log p)$ time.

   ii. Matrix multiplication. We can use the equation

   $$\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

   on page 9 of our textbook and the method of calculating powers by repeated squaring, taking everything modulo p. There are $O(\log n)$ matrix multiplications needed, each of which takes $M(logp)$ time. (Recall that $M(n)$ is the time to multiply two n-bit numbers.) Thus, the time to compute $\overline{F}_n$ by this method is $O(\log n M(\log p))$.

   iii. Memoization. There is different recurrence relation for Fibonacci numbers. For any $n \geq 2$:

   $$F_n = F_{(n-1)/2} F_{n/2} + F_{(n+1)/2} F_{(n+2)/2}$$

   where division is truncated integer division. Note that $F_5 = F_2^2 + F_3^2$, while $F_6 = F_2 F_3 + F_3 F_4$. Using the recurrence

   $$\overline{F}_n = \overline{F}_{(n-1)/2} \overline{F}_{n/2} + \overline{F}_{(n+1)/2} \overline{F}_{(n+2)/2}$$

   and using memoization: `https://en.wikipedia.org/wiki/Memoization` we compute values from top down, starting with $\overline{F}_n$, computing only the values that will be needed, and storing

them in a search structure. This way, the number of values of $\overline{F}_i$ that need to be computed is $O(\log n)$.

Memoization requires a search structure to hold intermediate values. The time to search for each item is proportional to the logarithm of the number of values times the size of one value, hence is $O(\log\log n \log p)$. Thus, the time to compute $\overline{F}_n$ by this method is $O(\log nM(\log p) + \log n \log\log p) = O(\log nM(\log p))$.

iv. Periodicity. The values of $\overline{F}_n$ are periodic: there is a number $k = k(p) < p^2$ such that $\overline{F}_n = \overline{F}_{n\,\%\,k}$ for all n. For example, $k(2) = 3$, $k(3) = 8$, $k(4) = 6$, $k(5) = 20$, $k(6) = 24$. We can find the value of k, as well as the values of $\overline{F}_i$ for all $1 \le i \le k$, by linear search in $O(p^2 \log p)$ time. (There might be a faster method.) One of those values will be $\overline{F}_{n\,\%\,p}$. We can compute $n\,\%\,p$ in $O(\log n \log p)$ time. Thus this method takes $O((p^2 + \log n)\log p)$ time.

2. Work problem 1.27 on page 40 of your textbook. The public key is $(391, 3)$, and $(p-1)(q-1) = 352$. The private key is 235 since $235 \cdot 3\,\%\,352 = 1$. The message is 41, and the crypt is $41^3\,\%\,391 = 105$

3. Work problem 1.33 on page 41 of your textbook. Use Euclid's algorithm to compute $g = gcd(x, y)$, the greatest common divisor of x and y. This takes $O(n^2)$ time, unless there is a faster algorithm for integer division. Then $lcm(x, y)$, the least common multiple of x and y is $(x/g)y$, Hence the time is $O(n^2)$. Can you do better?