Problem Set 4

CS381-Cryptography

Due October 14, 2014

One hundred points is a 'perfect' paper, but there are 140 points worth of problems. By the way, the last programming problem is a completely cookbook calculation, but it's something everyone in the class should be completely familiar with. The large point value is intended as an enticement to work on it.

1 Written Problems

1.1 Routine problems

1. *(10 points each)* Do problems 1,5,10,12 from Section 3.13 of the text. Show your work carefully and explain how you get your conclusions (you may cite any theorem we stated in class). You can use a calculator or a computer to do the arithmetic, but you should note that *all of the arithmetic can be done by hand,* and you should show your calculations at this level of detail.

Solution: Problem 1. This problem (both parts) is solved by applying the extended Euclid algorithm to 17 and 101. Here it is:

$$101 = 5 \times 17 + 16, 16 = -5 \times 17 + 1 \times 101,$$

$$17 = 1 \times 16 + 1, 1 = 6 \times 17 - 1 \times 101.$$

This is an answer to part (a). We also get as an immediate consequence the solution to part (b):

$$17^{-1} \bmod 101 = 6.$$

Solution: Problem 5. The computation of the gcd is done through Euclid's algorithm:

$$4883 = 4369 + 514$$

$$4369 = 8 \times 514 = 257$$

$$514 = 2 \times 257 + 0.$$

So $\gcd(4883, 4369) = 257$. Dividing 257 into each of these values gives

$$4369 = 17 \times 257, 4883 = 19 \times 257.$$

A quick check (division by primes up to 17) shows that 257 is prime, so these are the prime factorizations.

Solution: Problem 10. The problem asks us to solve the simultaneous congruences

$$\begin{aligned} x &\equiv 1 \pmod 3 \\ x &\equiv 2 \pmod 4 \\ x &\equiv 3 \pmod 5. \end{aligned}$$

The Chinese Remainder Theorem guarantees a unique solution between 0 and 59 inclusive (because $3 \times 4 \times 5 = 60$). We can use the explicit formula for solving these systems (given in class, and also in Problem 24), but here the numbers are small enough to let us use trial and error: The second congruence says the number has to be even, and the third then implies that the last decimal digit must be 8. So the solution is either 8,18,28,...,58. Since 58 is the only one of these that leaves a remainder of 1 upon division by 3, that is the smallest solution.

The next smallest solution is $58 + 60 = 118$.

Solution: Problem 12. 101 is prime, so by Fermat's Theorem,

$$2^{100} \equiv 1 \pmod{101}.$$

Thus

$$2^{10203} = (2^{100})^{102} \cdot 2^3 \equiv 1^{102} \cdot 2^3 \equiv 8 \pmod{101}.$$

1.2 More involved problems

(30 points) 2. *Fibonacci numbers as worst case for Euclid's Algorithm.* In class we showed that Euclid's Algorithm is *easy*: Applied to integers of size N it requires no more than $2\log_2 N$ divisions. The *Fibonacci numbers,* as you probably know, are defined by starting from $F_0 = 0, F_1 = 1$, and defining $F_{n+1} = F_n + F_{n-1}$ for $n > 1$. This gives the sequence 0,1,1,2,3,5,8,13,21,34,...

(a) Prove that applying Euclid's Algorithm to (F_n, F_{n+1}) for $n \geq 3$ gives $\gcd(F_n, F_{n+1}) = 1$ and requires $n-2$ divisions to reach the last remainder 1. (All these problems asking for a proof of something involving Fibonacci numbers can be done by mathematical induction.)

Solution. For the case $n = 3$, we have $F_n = 2$ and $F_{n+1} = 3$. Euclid's Algorithm gives

$$3 = 1 \times 2 + 1,$$

so the gcd is 1 and $n - 2 = 1$ division was performed.

Now assume $n > 3$, and for all $3 \leq m < n$, the claim is satisfied. We have

$$F_{n+1} = 1 \cdot F_n + F_{n-1},$$

and this is the first division performed in Euclid's Algorithm, since $F_{n-1} < F_n$. The subsequent steps are just Euclid's Algorithm applied to the pair (F_{n-1}, F_n). By the inductive hypothesis, this terminates after $n-1-2 = n-3$ subsequent divisions with a remainder of 1. So in total, n_2 divisions are performed, and the gcd is 1.

(b) Suppose that $1 \leq m < n$ are integers with $\gcd(m,n) = 1$, and that Euclid's algorithm applied to m and n reaches the last nonzero remainder in k steps. Prove that $n \geq F_{k+3}$. Combined with (a), this shows that the Fibonacci numbers are the worst case for Euclid's algorithm. (Induct on k. The induction starts with $k = 0$, when Euclid's algorithm requires no divisions. This forces $m = 1$, and hence $n \geq 2 = F_3$. You will also have to consider the case $k = 1$ separately before you do the inductive step.)

Solution. We demonstrated the case $k = 0$ above. If $k = 1$, then Euclid's algorithm requires 1 division to reach a remainder of 1. This means $2 \leq m < n$, so $n \geq 3 = F_4 = F_{k+3}$, so the theorem holds in this case as well. Now assume $k > 1$, and that the claim holds for all $0 \leq k' < k$. We write

$$n = q \cdot m + r.$$

Euclid's algorithm applied to (r, m) requires $k-1$ steps, and thus $m \geq F_{k+2}$. The remainder r' from this division is at least 1, so $1 \leq r' < r$, thus the inductive hypothesis again applies: Since Euclid's algorithm applied to (r', r) requires $k-2$ steps and thus $r \geq F_{k+1}$. We now have

$$n = q \cdot m + r \geq m + r \geq F_{k+2} + F_{k+1} = F_{k+3},$$

as required.

(c) Show that $F_k \geq 1.6^{k-2}$ for $k \geq 2$. Use this and parts (a,b) to prove that the number of divisions in Euclid's algorithm required to compute $\gcd(m,n)$ with $m < n$ is no more than $5 \cdot \log_{10} n$.

Solution. For $k = 2$ we have $F_k = 1 = 1.6^0 = 1.6^{2-2}$. For $k = 3$, $F_3 = 2 > 1.6 = 1.63 - 2$. For the inductive step, assume $k > 3$ and that the claim holds for all k' with $2 \leq k' < k$. Application of the inductive hypothesis gives

$$\begin{aligned}
F_k &= F_{k-1} + F_{k-2} \\
&\geq 1.6^{k-3} + 1.6k - 4 \\
&= 1.6^{k-4} \times (1.6 + 1) \\
&> 1.6^{k-4} \times 2.56 \\
&= 1.6^{k-4} \times 1.6^2 \\
&= 1.6^{k-2}
\end{aligned}$$

This establishes the first part of the claim. Now suppose that Euclid's algorithm applied to m, n with $m < n$ requires k steps. By the result of part (b) and the above inequality.we have

$$n \geq F_{k+3} > 1.6^{k+1}.$$

Taking base 10 logarithms of both sides

$$\log_{10} n > (k+1) \cdot \log_{10} 1.6 > (k+1) \cdot 0.2,$$

so
$$k < k+1 < 5 \cdot \log_{10} n.$$