The University of Sydney
School of Mathematics and Statistics

Solutions to Tutorial 7 (Week 8)

MATH2068/2988: Number Theory and Cryptography — Semester 2, 2017

Web Page: http://www.maths.usyd.edu.au/u/UG/IM/MATH2068/
Lecturer: Dzmitry Badziahin

Tutorial Exercises:

1. Let $a \geq 2$ be an integer. A composite number $n > 1$ is said to be a *pseudoprime for the base a* if $a^{n-1} \equiv 1 \pmod{n}$. Find the prime factorization of 341, and hence show that 341 is a pseudoprime for the base 2 but not for the base 3.

Solution: The prime factorization of 341 is 11×31. To find the residue of 2^{340} modulo 341, we first find its residues modulo 11 and 31. The order of 2 modulo 11 is 10, because it must divide 10 by Fermat's Little Theorem and is not either 2 or 5 because $4 \not\equiv 1 \pmod{11}$ and $32 \not\equiv 1 \pmod{11}$. So $2^{340} \equiv 1 \pmod{11}$. The order of 2 modulo 31 is clearly 5, so $2^{340} \equiv 1 \pmod{31}$. The solution of the simultaneous congruences $x \equiv 1 \pmod{11}$ and $x \equiv 1 \pmod{31}$ is clearly $x \equiv 1 \pmod{341}$, so we can conclude that $2^{340} \equiv 1 \pmod{341}$, showing that 341 is indeed a pseudoprime for the base 2.

The order of 3 modulo 11 is 5, because $3^5 = 243 \equiv 1 \pmod{11}$. So $3^{340} \equiv 1 \pmod{11}$. The order of 3 modulo 31 must be a divisor of 30; after calculating that $3^5 \equiv 26 \pmod{31}$, $3^6 \equiv 16 \pmod{31}$, $3^{10} \equiv 25 \pmod{31}$, and $3^{15} \equiv 30 \pmod{31}$, we know that in fact $\mathrm{ord}_{31}(3) = 30$. Since $340 \equiv 10 \pmod{30}$, we conclude that $3^{340} \equiv 3^{10} \equiv 25 \pmod{31}$. So $3^{340} \not\equiv 1 \pmod{31}$, and it follows that $3^{340} \not\equiv 1 \pmod{341}$. Thus 341 is not a pseudoprime for the base 3.

2. A composite number $n > 1$ is called a *Carmichael number* if it is a pseudoprime for any base a such that $\gcd(a, n) = 1$.

(a) Show that any Carmichael number must be odd. (Hint: consider $a = n - 1$.)

Solution: Let n be a Carmichael number; then since $\gcd(n-1, n) = 1$, n must be a pseudopime for the base $n - 1$, which means that $(n-1)^{n-1} \equiv 1 \pmod{n}$. But $(n-1)^{n-1} \equiv (-1)^{n-1} \pmod{n}$, so we have $(-1)^{n-1} \equiv 1 \pmod{n}$. This forces n to be either odd or equal to 2, and 2 is prime so it is by definition not a Carmichael number.

(b) Find the prime factorization of 561 and show that, for each of its prime factors p, we have $p - 1 \mid 560$.

Solution: The prime factorization of 561 is $3 \times 11 \times 17$, and each of 2, 10, 16 does indeed divide 560.

(c) Hence show that 561 is a Carmichael number.

Solution: We need to show that $a^{560} \equiv 1 \pmod{561}$ for any integer a such that $\gcd(a, 561) = 1$. Since $561 = 3 \times 11 \times 17$, it is enough to show that $a^{560} \equiv 1 \pmod{p}$ for each of the primes $p \in \{3, 11, 17\}$. (This is because we know by the Chinese Remainder Theorem that the unique solution of the simultaneous congruences $x \equiv 1 \pmod{p}$ for $p \in \{3, 11, 17\}$ is $x \equiv 1 \pmod{561}$.) But the assumption that $\gcd(a, 561) = 1$ implies that $\gcd(a, p) = 1$, so Fermat's Little Theorem tells us that $a^{p-1} \equiv 1 \pmod{p}$. We have already seen that 560 is a multiple of $p-1$ in each case, so it follows that $a^{560} \equiv 1 \pmod{p}$ as desired.

(d) Similarly, show that 6601 is a Carmichael number.

Solution: The prime factorization of $n = 6601$ is $7 \times 23 \times 41$, so once again it is the product of distinct primes p, and again for each of these prime factors we have $p - 1 \mid n - 1$, because all of 6, 22, and 40 divide 6600. The argument in the previous part applies to any n with these properties.

3. Let n be an odd integer greater than 1. To try to decide whether n is prime, we could test whether $a^{n-1} \equiv 1 \pmod{n}$ for various $a \in \{2, 3, \cdots, n-1\}$. A prime number will always pass this test, by Fermat's Little Theorem; but as seen in the previous questions, there are some composite numbers which will pass this test for many values of a. This question suggests a slight improvement to the test.

(a) Show that if n is prime and $a \in \{2, \cdots, n-1\}$, then $a^{(n-1)/2} \equiv \pm 1 \pmod{n}$.

Solution: Let $x = a^{(n-1)/2}$. Then $x^2 = a^{n-1} \equiv 1 \pmod{n}$ by Fermat's Little Theorem. Since n is prime, we can conclude from $x^2 \equiv 1 \pmod{n}$ that $x \equiv \pm 1 \pmod{n}$ by the argument in Question 2 of Tutorial 2.

(b) Show that when $n = 561$ and $a = 5$, we have $a^{(n-1)/2} \not\equiv \pm 1 \pmod{n}$, despite the fact that, as seen in the previous question, $a^{n-1} \equiv 1 \pmod{n}$.

Solution: We need to compute the residue of 5^{280} modulo 561. If we were really using this as a test of the primality of 561, then we wouldn't already know that 561 is $3 \times 11 \times 17$. So we would work out the following numbers, by successive squaring, reducing mod 561 at each step: 5^2, 5^4, 5^8, 5^{16}, 5^{32}, 5^{64}. Thus we could get $5^{70} = 5^{64} \times 5^4 \times 5^2$, and then go back to squaring to get 5^{140} and 5^{280}.

Knowing the factorization of 561, the easiest thing to do is compute 5^{280} modulo 3, 11 and 17, and then use the Chinese Remainder Theorem. Now $5^2 \equiv 1 \pmod{3}$; so $5^{280} \equiv 1 \pmod{3}$. And $5^{10} \equiv 1 \pmod{11}$; so $5^{280} \equiv 1 \pmod{11}$. Working modulo 17 we get $5^2 \equiv 8$, then $5^4 \equiv 64 \equiv 13$, and $5^8 \equiv 169 \equiv -1$. So $5^{280} = (5^8)^{35} \equiv -1$. Hence 5^{280} is congruent modulo 561 to the solution of $x \equiv 1 \pmod{3}$, $x \equiv 1 \pmod{11}$ and $x \equiv -1 \pmod{17}$. This solution can be found using the methods of Tutorial 4: it is $x \equiv 67 \pmod{561}$. So $5^{280} \equiv 67 \pmod{581}$.

This slightly better version of the test is used by the `IsProbablyPrime` function in MAGMA, which calculates the residue of $a^{(n-1)/2}$ modulo n for some randomly chosen values of a. If $a^{(n-1)/2} \equiv \pm 1 \pmod{n}$ for enough different values of a, then n is very likely to be prime.

4. Show that the function $f(k) = k^4 + k^3 + 2068k + 2988$ is $O(k^4)$.

Solution: We have that for $k \geq 1$, $k^3 \leq k^4$, $2068k \leq 2068k^4$, $2988 \leq 2988k^4$. Therefore we can choose $C = 2 + 2068 + 2988$ and $N = 1$ and get that

$$f(k) \leq (2 + 2068 + 2988)k^4 \quad \text{for all} \quad k \geq 1.$$

*5. Which of the following functions of a positive integer variable k are $O(k^a)$ for some positive integer a?

$$\log_2(k), \quad k\log_2(k), \quad k!, \quad \log_2(k!), \quad k^{\log_2(k)}, \quad \frac{(1.01)^k}{k^2}.$$

For the last function you can use the result from analysis: for any $c > 1$ and $b > 0$,

$$\lim_{k\to\infty} \frac{c^k}{k^b} = \infty.$$

Solution: We have $\log_2(k) < k$ for all positive integers k, by taking logarithms of both sides of the inequality $k < 2^k$ (which is obvious, say, from the fact that a set with k elements has 2^k subsets). So $\log_2(k)$ is $O(k)$. (In fact, if we allowed non-integral exponents a, we could prove that $\log_2(k)$ is $O(k^a)$ for any positive real number a, but we do not need this more powerful information.)

Since $\log_2(k)$ is $O(k)$, $k\log_2(k)$ is $O(k^2)$.

To show that $k!$ is not $O(k^a)$ for any positive integer a, we need a more tractable lower bound for $k!$. A convenient one (certainly not the only one that would work) is that $k! \geq (k/2)^{(k/2)-1}$, because at least $(k/2) - 1$ of the factors in the product $1 \times 2 \times \cdots \times (k-1) \times k$ are at least $k/2$. So it suffices to show that $(k/2)^{(k/2)-1}$ is not $O(k^a)$ for any positive integer a. Assume for a contradiction that there were positive a, C, N such that $(k/2)^{(k/2)-1} \leq Ck^a$ for all $k \geq N$. Rearranging the inequality, we deduce that $(k/2)^{(k/2)-1-a} \leq 2^aC$ for all $k \geq N$. But 2^aC is constant, so once k is sufficiently large, $(k/2)^{(k/2)-1-a}$ will certainly exceed it: more formally, if $k \geq \max\{2(2^aC+1), 2(a+2)\}$ then we have $(k/2) - 1 - a \geq 1$ and $k/2 > 2^aC$, so $(k/2)^{(k/2)-1-a} > 2^aC$. This gives the desired contradiction.

Since $k! \leq k^k$ (because $k!$ is the product of k numbers less than or equal to k), we have $\log_2(k!) \leq k\log_2(k)$. Since $k\log_2(k)$ is $O(k^2)$ as seen before, so is $\log_2(k!)$.

To show that $k^{\log_2(k)}$ is not $O(k^a)$ for any positive integer a, we assume for a contradiction that there were positive a, C, N such that $k^{\log_2(k)} \leq Ck^a$ for all $k \geq N$. Rearranging the inequality, we deduce that $k^{\log_2(k)-a} \leq C$ for all $k \geq N$. But once k is sufficiently large, $k^{\log_2(k)-a}$ will certainly exceed the constant C: more formally, if $k \geq \max\{C+1, 2^{a+1}\}$ then we have $\log_2(k) - a \geq 1$ and $k > C$, so $k^{\log_2(k)-a} > C$. This gives the desired contradiction.

For the last function we take $c = 1.01$, $b = a + 2$ and apply the proposition from lectures. We get:

$$\lim_{k\to\infty} \frac{(1.01)^k/k^2}{k^a} = \infty \quad \Rightarrow \quad \frac{(1.01)^k}{k^2} \ \text{is not} \ O(k^a).$$

**6. Recall the Fibonacci numbers F_n from Tutorial 3. Describe a polynomial-time algorithm which determines, for given positive integers n and m, the residue of

F_n modulo m. To say that the algorithm is polynomial-time means that there is some positive integer a such that the maximum number of bit operations it requires when n and m have k bits is $O(k^a)$.

Solution: Notice that simply computing the residues mod m of F_0, F_1, F_2 and so on up to F_n using the Fibonacci recurrence is not polynomial-time, because it involves about n additions, and n is exponential in the number of bits. We saw in examples in Tutorial 3 that the sequence of residues of Fibonacci numbers is periodic, but that knowledge doesn't necessarily help here, because the period could conceivably be of the order of m^2 which might well be bigger than n.

One of various possible polynomial-time algorithms uses the matrix-power formula

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

proved in Question 2 of Tutorial 3. We can talk about the residue of a matrix modulo m, which just means that we take the residue of each entry modulo m. Since matrix multiplication is defined using ordinary addition and multiplication which respect congruence modulo m, it is still true that we can compute the residue modulo m of a power A^n of a matrix by successively squaring and reducing modulo m to find the residue of the powers A^{2^i}, then multiplying appropriate powers (and reducing modulo m as we go) to compute the residue modulo m of A^n. If the size of the matrix is bounded, the time complexity of this procedure is essentially a constant multiple of the time complexity for numbers (i.e. 1×1 matrices), and so it is polynomial-time. Applying this when $A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$, we find the residue of F_n modulo m as one of the entries of the residue of A^n modulo m.

Extra Exercises:

7. It was shown in lectures that if $\frac{f(n)}{g(n)} \to L$ as $n \to \infty$ for some (finite) real number L, then $f(n)$ is $O(g(n))$. As an example to show that the converse doesn't hold, prove that $\phi(n)$ is $O(n)$, but $\frac{\phi(n)}{n}$ does not tend to any limit as $n \to \infty$.

Solution: It is obvious that $\phi(n)$ is $O(n)$, because $\phi(n) \leq n$ for all positive integers n. To prove that $\frac{\phi(n)}{n}$ does not tend to any limit as $n \to \infty$, note that if $n = p^k$ where p is a prime and $k \geq 1$, then $\frac{\phi(n)}{n} = \frac{p^{k-1}(p-1)}{p^k} = \frac{p-1}{p}$. So $\frac{\phi(n)}{n}$ takes each of the values $\frac{p-1}{p}$, where p is a prime, infinitely often (so it takes the value $\frac{1}{2}$ infinitely often, the value $\frac{2}{3}$ infinitely often, the value $\frac{4}{5}$ infinitely often, and so on), which means that it does not tend to a limit.

8. Show that $2047 = 2^{11} - 1$ is a pseudoprime for the base 2.

Solution: The compositeness of 2047 came up in Question 5 of Tutorial 2, where we saw that it is the smallest composite number of the form $2^p - 1$ (p a prime): we have $2047 = 23 \times 89$. The order of 2 modulo 2047 is clearly 11, so $2^{2046} = (2^{11})^{186} \equiv 1 \pmod{2047}$ as required.

***9.** Suppose that $n > 1$ is odd and $2^{n-1} \equiv 1 \pmod{n}$. (This implies that n is either prime or a pseudoprime for the base 2.) Let $m = 2^n - 1$.

(a) Show that $2^{m-1} \equiv 1 \pmod{m}$. (Hint: Question 6 of Tutorial 1 showed that $b \mid a$ implies $2^b - 1 \mid 2^a - 1$.)

Solution: Since $2^{n-1} \equiv 1 \pmod{n}$, we have $2^n \equiv 2 \pmod{n}$, so $m \equiv 1 \pmod{n}$. That is, $n \mid m - 1$. By the recalled result from Tutorial 1, this implies that $m = 2^n - 1 \mid 2^{m-1} - 1$, or in other words $2^{m-1} \equiv 1 \pmod{m}$.

(b) Hence show that there are infinitely many pseudoprimes for the base 2.

Solution: Start with any pseudoprime n for the base 2 (for example, $n = 341$). By Question 6 of Tutorial 1 again, since n is composite we have that $m = 2^n - 1$ is composite, so the previous part shows that m is a pseudoprime for the base 2 also. Then we can apply the same argument starting with m to deduce that $2^m - 1$ is a pseudoprime for the base 2, and so on indefinitely. (This certainly will produce infinitely many different examples, because $2^n - 1$ is always bigger than n for $n > 1$. Indeed, the growth rate of this particular sequence of pseudoprimes is colossal: each term dictates the number of bits of the next. This is far from giving the complete list of pseudoprimes for the base 2.)

***10.** Describe a polynomial-time algorithm which determines, for a given positive integer n, whether n is a Fibonacci number.

Solution: This is almost a trick question, because this is one computational task for which the first algorithm you would think of is polynomial-time: namely, start from $F_0 = 0$ and $F_1 = 1$ and apply the Fibonacci recurrence to successively compute all the terms in the Fibonacci sequence, stopping when you first reach either n (in which case, n is a Fibonacci number) or something bigger than n (in which case, n is obviously not a Fibonacci number). Each addition of F_{i-1} and F_{i-2} to produce F_i can certainly be done in polynomial time, since the numbers involved are all less than n. All we need to consider is how many F_i this algorithm ends up computing.

We need an estimate of how big F_m is for $m \geq 1$. Recall from Tutorial 3 the closed formula for the Fibonacci number F_m:

$$F_m = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^m - \left(\frac{1-\sqrt{5}}{2}\right)^m\right).$$

Since $-1 < \frac{1-\sqrt{5}}{2} < 0$, the contribution of the second term is negligible (in fact, one can easily show that F_m is always the nearest integer to $\frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^m$, but we will not need that specific fact). Letting $\tau = \frac{1+\sqrt{5}}{2}$, we have that $\sqrt{5}F_m$ is approximately equal to τ^m: more precisely, they differ by at most 1. Taking logarithms to the base 2, we deduce that $\frac{\log_2(5)}{2} + \log_2(F_m)$ is approximately equal to $m \log_2(\tau)$: more precisely, they again differ by at most 1 (the difference cannot increase since the logarithm function is concave). Hence m is approximately equal to

$$\frac{\log_2(5)}{2\log_2(\tau)} + \frac{\log_2(F_m)}{\log_2(\tau)}.$$

We conclude that, in running the above algorithm to determine whether n is a Fibonacci number, the number of steps required is within a fixed constant of a constant multiple of $k = \lfloor \log_2(n) \rfloor + 1$, the number of bits in the binary representation of n. So the algorithm will terminate in polynomial time.

****11.** Show that every Carmichael number n is squarefree, i.e. n is the product of distinct primes. (Hint: suppose for a contradiction that $n = p^k m$ where p is prime, $k \geq 2$ and $\gcd(p, m) = 1$. Consider $a = (n/p) + 1$, and the residue of a^p modulo p^k.)

Solution: As in the hint, we suppose for a contradiction that $n = p^k m$ where p is prime, $k \geq 2$ and $\gcd(p, m) = 1$. Let $b = n/p = p^{k-1}m$ and $a = b + 1$. Since every prime factor of n is also a prime factor of b and hence cannot divide a, we have $\gcd(a, n) = 1$, so the assumption that n is a Carmichael number says that $a^{n-1} \equiv 1 \pmod{n}$, and hence $a^n \equiv a \pmod{n}$. It follows that $a^n \equiv a \pmod{p^k}$.

On the other hand, the binomial theorem tells us that

$$a^p = 1 + pb + \sum_{i=2}^{p} \binom{p}{i} b^i.$$

Since $p^{k-1} \mid b$, we have $p^k \mid pb$ and $p^{(k-1)i} \mid b^i$. As $k \geq 2$, all the terms on the right-hand side are divisible by p^k except the initial 1, so $a^p \equiv 1 \pmod{p^k}$, and it follows that $a^n \equiv 1 \pmod{p^k}$. Combining our two congruences gives that $a \equiv 1 \pmod{p^k}$, a contradiction since p^k does not divide $a - 1 = p^{k-1}m$.