Z. Abel, B. Chapman, E. Demaine
revised Tuesday 12th March, 2024

Problem Set 5

- **Due Date:** 11:59pm on **Monday 18th March, 2024**
- **Days Covered**: 08, 09, and 10 (including Lecture, Warm-Up, and Recitation)

Problem 1. Fibonacci Divisibility [13 points]

Recall the Fibonacci numbers $(F_0, F_1, F_2, \ldots) = (0, 1, 1, 2, 3, 5, \ldots)$, where F_n equals $F_{n-1} + F_{n-2}$ for every $n \geq 2$. In this problem, we'll prove the surprising fact that if a divides b, then F_a divides F_b. For example, $F_6 = 8$ is a factor of $F_{12} = 144$.

Let $a > 0$ be a fixed (but unspecified) integer.

(a) [6 pts] Prove by (regular or strong) induction on n that

$$F_{n+a} \equiv F_n \cdot F_{a+1} \bmod F_a$$

for every $n \geq 0$.

In other words, when looking modulo F_a, the subsequence

$$(F_a,\ F_{a+1},\ F_{a+2},\ \ldots)$$

is congruent, term-by-term, to the original Fibonacci sequence after scaling by F_{a+1}:

$$(F_0F_{a+1},\ F_1F_{a+1},\ F_2F_{a+1},\ \ldots).$$

(b) [7 pts] Use part (a), and another induction, to prove that $F_a \mid F_b$ whenever $b \in \mathbb{N}$ is a multiple of a.

Hint: What will you induct on?

Problem 2. Computing Modular Inverses [10 points]

Let's see a few common techniques for computing modular inverses! Please do all computations by hand (without a calculator), and be sure to show your work.

(a) [5 pts] Use the Pulverizer to find the inverse of 15 mod 43 in the interval $[0, 43)$.

(b) [5 pts] Use Fermat's Little Theorem to find the inverse of 15 mod 43 in $[0, 43)$.

Problem 3. Primality Testing [12 points]

With modern hardware and (publically known) algorithms, factoring large numbers into their component prime factors seems intractable for integers with more than a few hundred digits[1]. It may be surprising, then, that the related task of **primality testing**—determining whether a given number is prime or composite—can be accomplished efficiently and practically, using clever but not complicated randomized algorithms algorithms. Let's investigate a commonly-used primality testing algorithm, due to Gary Miller and Michael Rabin, that can test numbers with hundreds of digits in fractions of a second on a typical laptop[2].

For this problem, let n be the integer we wish to test, to answer the question "is n prime or composite"?

(a) [2 pts] Here is a first attempt, known as the Fermat test:

> Choose an integer $1 \leq a \leq n-1$, and compute $a^{n-1} \bmod n$ using the repeated squaring technique. If this result is not 1, return **Composite**. Otherwise, return **I don't know**.

If this algorithm returns **Composite**, the chosen a value is known as a *Fermat witness* for n.

Prove that a Fermat witness is enough to prove that n is composite. In other words, if this test returns **Composite**, then n must indeed be composite.

Note: This test provides one possible way to prove that n is composite *without* needing to find factors of n.

One approach is to try many values of a, hoping to find a Fermat witness in a reasonable number of tries. However, this does not always work: some composite numbers can "fool" the Fermat test for too many values of a. For example, a composite number n is known as a *Carmichael number* iff $a^{n-1} \equiv 1 \bmod n$ for every a, $1 \leq a \leq n-1$, that is *relatively prime* to n. For these numbers, the only possible Fermat witnesses are numbers that share factors with n, which suggests that finding Fermat witnesses for n may be as hard as factoring n.

It is known that infinitely many Carmichael numbers exist. The three smallest Carmichael numbers are $561 = 3 \cdot 11 \cdot 17$, $1105 = 5 \cdot 13 \cdot 17$, and $1729 = 7 \cdot 13 \times 19$.

(b) [5 pts] En route to a better test, prove the following lemma: if p is prime and $x^2 \equiv_p 1$, then $x \equiv_p \pm 1$. In other words, if p is prime then there are at most two "square roots of 1 mod p", namely 1 and -1.

Hint: Use Lemma 9.4.2 in the textbook.

(c) [5 pts] The Miller-Rabin primality test is a strengthening of the Fermat test:

---

[1] The largest number in the RSA Factoring Challenge that has been successfully factored has 250 digits (in base 10), requiring approximately 2700 CPU-years of computational power. https://en.wikipedia.org/wiki/RSA_numbers#RSA-250

[2] Try it for yourself! Here is an interactive Javascript implemenation https://planetcalc.com/8995/, and here are some large primes to test https://primes.utm.edu/lists/small/small2.html.

Divide $n-1$ by 2 as many times as possible, so that $n-1 = 2^e \cdot k$ where k is odd. Choose an integer $1 \leq a \leq n-1$ as before, and compute the mod n remainders of

$$\begin{aligned}
x_0 &= a^k, \\
x_1 &= x_0^2 = a^{2k}, \\
x_2 &= x_1^2 = a^{4k}, \\
&\vdots \\
x_e &= x_{e-1}^2 = a^{2^e k} = a^{n-1}.
\end{aligned}$$

If $x_e \not\equiv_n 1$, or if there is a pair of consecutive terms (x_i, x_{i+1}) where $x_{i+1} \equiv_n 1$ but $x_i \not\equiv_n \pm 1$, return **Composite**. Otherwise return **I don't know**.

Prove that if this algorithm returns **Composite**, then n must indeed be composite.

Note: We won't prove this here, but it can be shown that for every composite number n (including Carmichael numbers!), at least 3/4 of the values $1 \leq a \leq n-1$ lead to a conclusive **Composite** proof under this test! In practice, you can repeat this with (say) 100 randomly-chosen a-values. If any say **Composite** then n is definitely composite; otherwise, you'll conclude that n is **Probably Prime**. The chance of a composite number returning the wrong answer (i.e., getting unlucky with every randomly-selected a-value) is at most $1/4^{100}$; I'll take those odds!

Problem 4. Pulverizer State Machine [15 points]

Define the Pulverizer State machine to have:

$$\begin{aligned}
\text{states} &::= \mathbb{N}^2 \times \mathbb{Z}^4 \\
\text{start state} &::= (a, b, 1, 0, 0, 1) \\
\text{transitions} &::= \text{If } y > 0, \text{ then } (x, y, s, t, u, v) \to \\
&\quad (y,\ r,\ u,\ v,\ s - qu,\ t - qv) \qquad (\text{where } q = (x \text{ div } y), r = (x \operatorname{rem} y)).
\end{aligned}$$

Here, $(x \text{ div } y)$ indicates the quotient when dividing with remainder, so $q = (x \text{ div } y)$ is the integer q that satisfies $x = qy + (x \operatorname{rem} y)$.

(a) [7 pts] Define the state predicates

$$\gcd(x, y) = \gcd(a, b), \tag{Pres1}$$
$$sa + tb = x, \text{ and} \tag{Pres2}$$
$$ua + vb = y. \tag{Pres3}$$

Show that "(Pres1) AND (Pres2) AND (Pres3)" is *preserved* by the Pulverizer machine across transitions.

(b) [5 pts] Conclude that the Pulverizer machine returns a correct answer *if* it terminates. Correctness means that it computes a pair of coefficients s, t satisfying Bézout's identity

$$\gcd(a, b) = sa + tb.$$

(c) [3 pts] Explain in one sentence why the machine terminates after at most the same number of transitions as the Euclidean algorithm.

MIT OpenCourseWare
https://ocw.mit.edu

6.1200J Mathematics for Computer Science
Spring 2024

For information about citing these materials or our Terms of Use, visit: https://ocw.mit.edu/terms