Number Theory and Primality Tests

UCLA Math Circle Advanced 1B

by Jack Fasching

Week 2: January 19, 2025

Introduction

Primarily Test Recap

Definition 0.1 *A natural number greater than 1 is* ***prime*** *if it cannot be written as a product of two natural numbers that are both greater than 1. Otherwise, it is said to be* ***composite****.*

Primarily tests are algorithms that determine whether a number is prime or not. Last packet we looked at one primality test called **trivial division**, which checks if a number n is divisible by all natural numbers less than n, excluding 1. We can write it out in pseudocode:

```
Let integer i = 2;
While i is not greater than n-1, do the following:

    If the remainder of n divided by i is 0:
        Return "n is composite" and exit code.
    Otherwise, increase i by 1.

If code goes through all values of i:
    Return "n is prime" and exit code.
```

We made this algorithm more efficient by only checking divisors up to $\sqrt{n}$, then only checking prime numbers up to $\sqrt{n}$, then checking 2, 3, and 5 before checking all numbers greater than 5 of the form $30k + i$ for certain i up to $\sqrt{n}$. After all this, our new trial division algorithm (in pseudocode) is:

```
Let L be {1, 7, 11, 13, 17, 19, 23, 29}

For each element p in list {2, 3, 5}:
    If the remainder of n divided by p is 0:
        Return "n is composite" and exit code.

Let integer k = 0
Let integer j = 2
Let x = 30 * k + (jth element of L)
While x is not greater than sqrt(n), do the following:
    If the remainder of n divided by x is 0:
        Return "n is composite" and exit code.
    Otherwise, if j < 9:
        Increase j by 1.
    Otherwise, set j to 0 and increase k by 1.

If code goes through all values of x:
    Return "n is prime" and exit code.
```

Number Theory Recap

Last packet on primality tests, we introduced the following:

Definition 0.2 *The **greatest common divisor** of nonzero integers a and b is the greatest integer that divides both a and b. It is denoted by* $\gcd(a, b)$.

We also proved the following:

Definition 0.3 (The Division Algorithm) *Given two integers a, b, we can find two integers q, r, where $0 \leq r < b$ and $a = qb + r$. In other words, we can divide a by b to get q remainder r.*

Theorem 0.1 (Bezout's Identity) *Let a, b be integers with greatest common divisor d. Then there exist integers x, y such that $ax + by = d$.*

Lemma 0.1 (Euclid's Lemma) *If a prime p divides the product ab of two integers a and b, then p must divide at least one of those integers a or b.*

Theorem 0.2 (Fundamental Theorem of Arithmetic) *Every integer greater than 1 can be represented uniquely as a product of prime numbers, up to the order of their factors (in other words, this unique representation $p_1^{e_1}...p_k^{e_k}$ has $p_1 < ... < p_k$).*

We will not go over the proofs we did last packet. Instead, here are a few simple warm-up problems:

Problem 0.1 *Prove that, for any integers a, b, c,*
$\gcd(ac + b, a) = gcd(a, b)$.

Problem 0.2 *Using the previous problem and the Division Algorithm (Definition 0.3), detail an algorithm (pseudocode or not) for finding* $\gcd(a, b)$. *Then use it to compute $gcd(1610, 207)$ by hand.*

1 Number Theory: Modular Arithmetic

The next primality tests rely on **modular arithmetic**:

Definition 1.1 *We say that a, b are equivalent to mod m if m divides $a - b$.*
If a is equivalent to b mod m, we write $a \equiv b \pmod{m}$.

For example:
$32 \equiv 2 \pmod{6}$
$4 \equiv 4 \pmod{6}$
$-2 \equiv 4 \pmod{6}$

In this system of arithmetic, numbers "wrap around" after reaching a certain value, called the **modulus** (this is the number in parenthesis after "mod").

One famour example of modular arithmetic is determining AM/PM time on a 12-hour clock: for example, if the time now is 5:00, then after 9 hours it will be 2:00, just like how $5 + 9 \equiv 2 \pmod{12}$.

Problem 1.1 *Complete the following:*
$87 \equiv$ ___ $\pmod{12}$ *(your answer should be between 0 and 12)*
$13 \equiv 2 \pmod{\text{___}}$
___ $\equiv 1 \pmod{9}$

Problem 1.2 *Fix a nonzero integer n. Prove that congruence $\pmod{n}$ is an equivalence relation. In other words, prove that:*

- *For any integer a, $a \equiv a \pmod{n}$.*
- *For any integers a, b, if $a \equiv b \pmod{n}$, then $b \equiv a \pmod{n}$.*
- *For any integers a, b, c, if $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$.*

Problem 1.3 *Prove that congruence mod n is a congruence relation, an equivalence relation that is compatible with addition and multiplication. In other words, prove that if $a_1 \equiv a_2 \pmod{n}$ and $b_1 \equiv b_2 \pmod{n}$, then $a_1 + b_1 \equiv a_2 + b_2 \pmod{n}$ and $a_1 b_1 \equiv a_2 b_2 \pmod{n}$.*

Problem 1.4 *How does the congruence relation of modular arithmetic relate to Euclidean Division (Definition 0.3)?*

Using our new knowledge of modular arithmetic, we can create a second primality test other than trial division using **Wilson's Theorem**.

Theorem 1.1 (Wilson's Theorem) *Natural number p is prime if and only if $(p-1)! \equiv -1 \pmod{p}$.*

1.1 Proving Wilson's Theorem

To succinctly prove Wilson's theorem, we need to introduce another concept in number theory:

Definition 1.2 *The **inverse** of $a \pmod m$ is an integer a^* so that $a \times a^* \equiv 1 \pmod m$.*
Note that not every integer a has an inverse mod m.

Problem 1.5 *Find each of the following, if it exists:*

- *The inverse of* $3 \pmod 7$
- *The inverse of* $5 \pmod{10}$
- *The inverse of* $20 \pmod{14}$
- *The inverse of* $34 \pmod{541}$

Problem 1.6 *For some prime p and natural number $a < p$, we assume the inverse of $a \pmod p$ exists. In general, how do we find it? Give an example for* $3 \pmod 7$.
Hint: The answer involves something we covered last packet in the "Number Theory Recap" section.

Problem 1.7 *Prove the following theorem:*
a has an inverse mod m if and only if $\gcd(a, m) = 1$.
Hint: The proof involves two things we covered last packet.

A corollary to this theorem is that for prime p any integer a where $0 < a < p$ has an inverse mod p.

Problem 1.8 *If a is its own inverse mod p, then what is $a \pmod p$?*

We should have enough practice for the following:

Problem 1.9 *Prove Wilson's Theorem.*

1.2 Back to Primality Tests

Problem 1.10 *Write an algorithm, similar in spirit to the one for trial division, that uses Wilson's Theorem to check whether the number is prime or composite.*

Problem 1.11 *Compared to our new algorithm for trial division, does this algorithm require more or less computing power? Is it more or less efficient?*

Wilson's theorem is very important in finding quadratic residues in number theory (this is related to the first challenge problem).

2 Probabilistic Primality Tests

Previously, we have looked at examples of ***deterministic*** *algorithms, meaning they give a definitive result every time they are run.*

In this section, however, we will look at primality tests that are not deterministic, instead called ***probabilistic****. Probabilistic primality tests usually have an element of randomization, run many iterations for required accuracy, and only determine that a number is composite or probably prime.*

2.1 Fermat primality test

This is the simplest probabilistic primality test. Before we get into the pseudocode, we must define the following:

Definition 2.1 *Two integers a, b are* ***coprime*** *if $\gcd(a, b) = 1$.*

Given natural number $n > 1$, we can write out the algorithm in pseudocode:

```
Let a = randomly generated integer from 2 to n-2, inclusive.

If a^(n-1) is not 1 (mod n):
     Return "n is composite" and exit code.
Otherwise, return "n is probably prime" and exit code.
```

This primality test utilizes Fermat's Little Theorem:

Theorem 2.1 (Fermat's Little Theorem) *If a is an integer, p is a prime number and a is not divisible by p, then $a^{p-1} \equiv 1 \ (mod\ p)$.*

Problem 2.1 *Show that Fermat's theorem holds for $n = 7, a = 5$. Then show it holds for $n = 13, a = 3$.*

Problem 2.2 *Prove Fermat's Little Theorem.*

Problem 2.3 *For the program above, why would it be better not to include the possibility of checking $a = 1$ or $a = p - 1$?*

Problem 2.4 *Suppose we wish to determine whether $n = 221$ is prime or not, and say we randomly pick $a = 38$. What will the program output? What about for $a = 24$? Justify your answers.*
Hint: Try to use some tricks to make your calculations more efficient.

Definition 2.2 *Compisite number n is called a* ***Fermat pseudoprime*** *to base a if $a^{n-1} \equiv 1 \pmod{n}$. Unlike the pseucode in our algorithm, a does not have to be in the range $[2, n-2]$.*

Problem 2.5 *For what bases are each of the numbers pseudoprime?*

- *n=9*
- *n=15*
- *n=21*
- *n=25*
- *n=27*

3 Challenge Problems

Problem 3.1 *There are certain things whose number is unknown. If we count them by threes, we have two left over; by fives, we have three left over; and by sevens, two are left over. How many things are there?*

Problem 3.2 *Prove that, for any prime p such that $p \equiv 1 \pmod 4$, the number -1 is a square mod p.*
Hint: Use algebraic manipulation on Wilson's Theorem.

Problem 3.3 *Prove that there are infinitely many Fermat pseudoprimes.*