Finding Primes and Safe Primes
Exposition by William Gasarch

1 Introduction

In Section 2 we present mathematics that will be needed. In Section 3 give an algorithm to TEST if a number is prime. The one we give may be incorrect in two ways, though both are rare. In Section 4 we give an algorithm that will, given n, find a prime between n and $2n$. In Section 5 we give an algorithm that will, given n, find a safe prime between n and $2n$.

2 Mathematics We Will Need

Lemma 2.1 *If $n = \frac{x}{y}$ is an integer and p is a prime that divides x but not y then p divides n.*

Proof: Factor both x and y. There will be a factor of p in x but not in y. When you reduce to lowest terms all of the prime factors of y will go away. Some of the prime factors of x will go away, but not p. Hence p will remain. This yields a factorization of x where p is one of the factors. ∎

The following lemma you should know from when you studied combinatorics, though I may go over the proof in class.

Lemma 2.2 *The number of ways to choose b items from a items is $\binom{a}{b} = \frac{a!}{b!(a-b)!}$.*

Lemma 2.3 *For all primes p, for all $1 \leq y \leq p-1$ p divides $\binom{p}{y}$.*

Proof: $\binom{p}{y} = \frac{p!}{y!(p-y)!}$ is an integer where p divides the numerator but not the denominator. By Lemma 2.1 p divides $\binom{p}{y}$. ∎

The following you have surely seen. I may prove it in class

Lemma 2.4 *Let $n \in \mathsf{N}$. Then $(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i}$.*

3 Testing Primality

We need a property true of primes but not of other numbers. The following lemmas give a properties that are true of primes. We will later discuss when it is true of composites.

Lemma 3.1 *Let p be a prime and $a \in \mathsf{N}$. Then $a^p \equiv a \pmod{p}$.*

Proof: We prove this by induction on a.
Base case: If $a = 1$ then $a^p = 1^p = a \pmod{p}$.
Induction Hypothesis: Assume that $a^p \equiv 1 \pmod{p}$ and that $a + 1 \leq p - 1$.
Induction Step:

$$(a+1)^p \equiv \sum_{i=0}^{p} \binom{p}{i} a^i 1^{p-i} = \sum_{i=0}^{p} \binom{p}{i} a^i = 1 \times a^0 + \sum_{i=1}^{p-1} \binom{p}{i} + 1 \times a^p.$$

By Lemma 2.3 all of the terms in $\sum_{i=1}^{p-1} \binom{p}{i}$ are $\equiv 0 \pmod{p}$. Hence we have

$$(a+1)^p \equiv \sum_{i=0}^{p} \binom{p}{i} a^i 1^{p-i} = 1 + a^p.$$

By the induction hypothesis $a^p \equiv a \pmod{p}$, so we have $(a+1)^p \equiv a+1 \pmod{p}$.
∎

Lemma 3.2 *If $1 \leq a \leq p - 1$ and p is prime then $a^{p-1} \equiv 1 \pmod{p}$.*

Proof: By Lemma 3.1 $a^p \equiv a \pmod{p}$.

Since $1 \leq a \leq p - 1$, a has an inverse mod p. Multiply both sides by $a^{-1} \pmod{p}$ to get

$a^{p-1} \equiv 1 \pmod{p}$. ∎

In brief:

p prime implies $(\forall 1 \leq a \leq p-1)[a^{p-1} \equiv 1 \pmod{p}]$

What about the converse? Its not true, but the following is:

For most non-primes m, for most $1 \leq a \leq m - 1$, $a^{m-1} \not\equiv \pmod{m}$.

This leads to the following algorithm to test primes. We have a parameter L that is a tradeoff parameter- the larger L is the more likely the answer is correct (YEAH!) but the more time it will take (BOO!).

1. Input(m)
2. Pick L random numbers $a \in \{1, 2, 3, \ldots, m-1\}$.
3. For each a that you picked compute a^{n-1}. If any of them are NOT 1 then output NOT A PRIME. If ALL of them are 1 then output YES, A PRIME!

CORRECTNESS: If the algorithm says NO then n is definitely NOT a prime. However, if the algorithms says YES then n could still be a prime, but this is unlikely.
SPEED: Computing a^{n-1} takes $\log m$ steps, so the algorithm takes $O(L \log n)$ steps.
IMPROVEMENTS: We could speed up the algorithm in two ways.

- Before picking L we could see if some small primes divide n. It turns out we do not need to do this. We will be using this algorithm to FIND primes, and we will end up not giving this algorithm any input n where a small prime divides n.
- We could organize step 3 so that as soon as we find an a such that $a^{n-1} \not\equiv 1$ we stop.

4 Finding Primes

We will now give an algorithm that, given n, find a prime between n and $2n$.

1. Input n
2. Repeat until you find a prime:
   (a) Pick a number $p \in [n, 2n]$ at random.
   (b) Test if p is a prime. If so then output p and STOP.

SPEED: Testing primes is easy. However, are there enough primes so that we will find one fairly quickly. YES. It is known (the prime number theorem) that between n and $2n$ there are roughly $\frac{n}{\log n}$ primes. Hence if we pick numbers between n and $2n$ at random we will almost surely encounter a prime within the first $O(\log n)$ picks.

Each iteration takes roughly $\log n$ steps, and the number of iterations will be at most roughly $\log n$. Hence the algorithm takes $\log^2 n$ steps.
IMPROVEMENTS: We can speed this up by only guessing odd numbers. Do that by picking $q \in [n/2, n]$ at random and guessing $p = 2q+1$. Instead of having to guess from n numbers, we only have to guess from $n - n/2 = 0.5n$ numbers! Formally

1. Input n
2. Repeat until you find a prime:
   (a) Pick a number $q \in [n/2, n]$ at random.
   (b) Let $p = 2q + 1$.
   (c) Test if p is a prime. If so then output p and STOP.

This is faster since the algorithm never even looks at even numbers. But can we do better? What if we do not want to look at any number that has 2 or 3 as a factor? What do such numbers look like?

Lets look at those numbers mod $6 = 2 \times 3$.
If $p \equiv 0 \pmod 6$ then $p \equiv 0 \pmod 2$ (and $\equiv 0 \pmod 3$) CANNOT USE
If $p \equiv 1 \pmod 6$ then $p \not\equiv 0 \pmod 2$ and $p \not\equiv 0 \pmod 3$) CAN USE

If $p \equiv 2 \pmod 6$ then $p \equiv 0 \pmod 2$ CANNOT USE
If $p \equiv 3 \pmod 6$ then $p \equiv 0 \pmod 3$ CANNOT USE
If $p \equiv 4 \pmod 6$ then $p \equiv 0 \pmod 2$ CANNOT USE
If $p \equiv 5 \pmod 6$ then $p \not\equiv 0 \pmod 2$ and $p \not\equiv 0 \pmod 3$) CAN USE
SO we can use all numbers of the form either $6q + 1$ or $6q + 5$. Here is a new algorithm:

1. Input n
2. Repeat until you find a prime:
   (a) Pick a number $q \in [n/6, 2n/6]$ at random.
   (b) Pick a number $b \in \{1, 5\}$ at random.
   (c) Let $p = 6q + b$.
   (d) Test if p is a prime. If so then output p and STOP.

This is faster since the algorithm never even looks at numbers $p \equiv 0 \pmod 2$ or $p \equiv 0 \pmod 3$. But can we do better? What if we do not want to look at any number that has 2 or 3 or 5 as a factor? What do such numbers look like? We leave that to you; however, it will have to do with the number mod 30.

5 Safe Primes

What if we want to find SAFE primes? This is an easy modification. We modify the first algorithm to find primes and then leave it to the reader to modify the others.

1. Input n
2. Repeat until you find a safe prime:
   (a) Pick a number $p \in [n, 2n]$ at random.
   (b) Test if p is a prime. If so then test if $\frac{p-1}{2}$ is prime. If YES then output(p) and STOP.

SPEED: Testing primes is easy. However, are there enough primes so that we will find one fairly quickly. YES. It is known (the prime number theorem) that between n and $2n$ there are roughly