1 Primality Testing

First, we will treat the various methods of determining whether or not a number is prime, or, as is usually easier, determining whether a number is composite. At first glance, it seems strange to have methods for such a thing that are distinct from factorization methods - after all, if you can obtain the prime factorization of a number, it is trivial to determine primality from that factorization. However, factorization methods are in general very computationally expensive compared to some of the methods we will examine in this section.

1.1 Primality Tests and Composite Tests

Any viable computational method for testing primality explicitly is composed of a condition on a number n that, when met, necessitates that n is prime. Thus if the condition is not met, n is composite. Such tests are certainly extremely convenient to directly determine the primality of n, but unfortunately methods of this form are usually a combination of very complex and restricted to n of a particular form or within some bounded range.

In addition to these primality tests, which never fail on determining primality, we also have many tests that are comparatively simple and computationally efficent, which occasionally fail to identify primes, but never indicate that a composite number is prime. We will call these tests, which never fail on determining compositeness, compositeness tests.

It is of the utmost importance to note, before moving on, that while the conditions of a primality or compositeness test being met guarantees that the number is either prime or composite respectively, and a failed primality test proves compositeness, a failed compositeness test does not necessitate that the number is prime.

1.2 The Sieve of Eratosthenes

Sieving is a process where a series of operations are applied to every number in a large, regularly spaced set of integers in order to find numbers with certain characteristics.

The Sieve of Eratosthenes is among the first algorithmic methods for factorization and primality testing. Admittedly, it is very crude, but the theoretical basis of the method gives us some of the fundamental ideas for developing further primality tests. Additionally, the Sieve of Eratosthenes also can be used for algorithmic factorization of a number - as such, we will treat it rather lightly here and revisit it in more detail when we examine factorization methods.

The core use of the Eratosthenes' sieve in primality testing stems from the idea of *trial division*, where given an integer n, we attempt to divide n by every integer that could possibly be a factor. If the number divides n, then it is a factor of n, and because possible factors must be bounded above by n itself, we are guaranteed a finite number of computations. In fact, we need only test factors less than $\sqrt{n}$, recovering greater factors as the quotient of a successful

division by a lesser factor. Additionally, after each successful division, only the quotient need be tested further, resulting in easier computation as more factors are found. None of these facts are particularly reassuring computationally, but they provide some background for the use of our sieve.

The sieve can be thought of as a list or array of $N-1$ consecutive integers, beginning with 2. We recognize that 2 is prime, and thus every multiple of 2 is composite. We then remove 4,6, etc. from our list, and are left with 2 and the odd integers. We then recognize 3 as prime, and remove all multiples of 3 in the same way. 5 has not been removed from the list, and so it is prime. We then repeat our method. Continuing this way until the end of the set has been reached, we have constructed a set of all prime numbers up to N.

Note that the sieve has other modifications that can be made to find things like – the least prime factor of each composite up to N, or even the complete factorizations of the numbers in the sieve.

It is a fair observation that this is not at all an individualized test, but rather construction of a set that will prove a number n prime if n is an element of the set, and composite if $n < N$ and n is not in the set. This is not ideal, but combined with our method of trial division, and given that 76% of odd integers have a prime factor less than 100, precompiling a list of primes up to a certain bound and performing trial divisions of only those prime numbers on a given n can prove many numbers composite without having to resort to more rigorous primality tests.

1.3 Fermat's Theorem and Resulting Methods

Among the many limitations of Eratosthenes' sieve is its lack of precision - it is impossible to simply prove a given number n prime or composite with the sieve, you must construct the entire set of primes up to n in the worst case to say anything about n at all. We will next move to our very first formal test, a compositeness test based on a theorem of Fermat.

In passing through the list, we are adding p to find the next multiple. In addition to that, we noted earlier that we are guaranteed a finite number of computations. Therefore, this sieve has the benefit of not being particularly computationally intensive. In practice, the problem is the amount of memory it can consume. This is dependent upon the size of the list of numbers. There are work-arounds for this, like array segmentation, but this may affect the efficiency of the sieve.

1.3.1 Fermat's Theorem

Fermat's Theorem. *If a positive integer p is a prime and $(a,p)=1$, then*

$$a^{p-1} \equiv 1 \pmod{p}$$

Fermat's theorem cannot be used as a primality test, as we will examine further when we define pseudoprimes, but we can use the converse of the theorem to develop a test for compositeness.

Converse of Fermat's Theorem. *If n and a are positive integers such that $(a, n) = 1$ and*

$$a^{n-1} \not\equiv 1 \pmod{n},$$

then n is not a prime. Hence n is composite.

1.3.2 Pseudoprimes and Carmichael Numbers

Pseudoprimes are composite numbers which behave like primes for some theorem .

1.3.3 Computational Viability of Fermat's Theorem Methods

1.3.4 Improvements due to Eulers Criterion

1.3.5 Euler Pseudoprimes

1.3.6 Strong Pseudoprimes

1.3.7 Remarks and Numerical Data