Project 2
Primes

KG

January 12, 2024

1 Introduction

In this project, we will study primality testing as well as algorithms for finding large probable primes.

Use uniform randomness in this problem.

2 Tasks

1. Theory We want to sample a random prime from the set $S_n = \{2^n, 2^n+1, \ldots, 2^{n+1}-1\}$ for various n.

a. Random search Suppose we search for primes by sampling candidates at random from S_n and testing for primality.

For a few different n, estimate the number of candidates we must test before we find a prime.

b. Random range search Suppose we search for primes by choosing a at random from S_n and testing the primality of the numbers in the range $\{a, a+1, \ldots, a+d-1\} \cap S_n$, for some fixed width d.

For the same n as above, estimate the size d of the small range for us to be reasonably sure that the range contains at least one prime.

This changes the distribution of the prime numbers you find. Explain how.

c. Filtering Suppose now that we only test candidates that are not divisible by some set of small primes.

For the same n as above, estimate the number of candidates we must test before we find a prime.

2. Practice

a. Implement: Primality test Use your exponentiation algorithm implementation from Project 1 to implement a Fermat test for primality (and Carmichael numbers).

Instead of the simple Fermat test, you may implement better primality tests, such as Rabin-Miller.

(If you want to, you may use any exponentiation algorithm built into your programming environment instead of your own implementation from Project 1.)

b. Implement: Finding primes Implement an algorithm for finding primes in S_n by choosing candidates at random and applying the primality test to each candidate.

c. Implement: Trial division Modify your prime-finding algorithm to do trial division by some number of small primes before applying the primality test to each candidate.

Hint: You may want to use a precomputed list of small primes.

d. Implement: Sieving Modify your prime-finding algorithm to search for primes in some range, using sieving to reject most composites in the range, applying the primality test to the remaining candidates in the range.

e. Run: Experiment Determine experimentally, for various sizes of primes, the optimal number of small primes to use, as well as the size of the sieving range, and compare the cost of the various algorithms.