Sifting Squared Prime Intervals
Efficient Prime Acquisition and Counting

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Anthony Perez

Annandale-on-Hudson, New York
January, 2015

Abstract

In mathematics, a sieve is a method through which to sift out the desired numbers from a large set. The Sieve of Eratosthenes does just this: it sifts the natural numbers for primes. However, through increasing computing power and more advanced techniques, this sieve has become obsolete, remaining in the minds of mathematicians mainly for aesthetic purposes.

In this project, I try to resurrect Eratosthenes' basic concept of prime acquisition through composite elimination by sifting not the natural numbers but rather the primitive residual classes modulo the product over the first n primes, increasing the efficiency of composite elimination, which results in faster prime acquisition. We build an algorithm that does just this, then analyze its performance to determine its viability.

The Sieve of Eratosthenes involves a combinatorial method of counting prime density up to some ceiling $m \in \mathbb{N}$. We apply the notion of sifting the primitive residual classes modulo the product over the first n primes to make this combinatorial counting method more efficient by showing that less terms are necessary to evaluate $\pi(m)$, where m is restricted to the product over the first n primes for $n \geq 4$.

Contents

List of Figures

Dedication

This project is dedicated to many people, whose existence in my life create my enthusiasm to become better in everything I do. To my mother, who has believed in me since before Bard. To my father, who keeps the past there and continues to become a new man everyday. And to my daughter, the little ball of loving energy, who constantly jumps on my back and calls me horsey, the one person who I aspire to inspire.

Acknowledgments

I must acknowledge my classmates Kenneth Alger III and John Zoccoli, who by just their being encourage me to strive on in hopes of being the best. I also must acknowledge everyone who has kept me in anxiety and daily pressures: my classmates for trusting my mathematical capacity, and my professors throughout this mathematical journey for opening the doors to new (at least for me) mathematical discoveries. Specifically, I must thank Professor Lauren Wolf, whose personality makes math fun; Professor Branden Stone, who continually asks questions that deepen my understanding of whatever subject I'm working on; and Professor Malick Ndiaye, whose patience and encouragements help make this student feel better through the frustrations of self-doubt.

1
Introduction

The search for prime numbers goes back into ancient history. These numbers have piqued the interest of mathematicians eager to understand these seemingly simple building blocks of our number system. Eratosthenes, however, was the first mathematician to develop a systematic method for acquiring primes [11]. His sieve continues to influence a whole theory of mathematics dedicated to the sifting for not only primes but other types of numbers. This project investigates his sieve in hopes of answering the following questions.

Question 1. What is the Sieve of Eratosthenes?

The Sieve of Eratosthenes is a systematic method for acquiring large primes. By creating a simple recurrence sequence for each prime that gathers all positive multiples of that prime, the sieve eliminates all composites from the natural numbers up to some ceiling $m \in \mathbb{N}$. In fact, one main property of primality testing comes from this sieve: the fact that one only needs to use primes less than $\sqrt{m}$ in order to sift up to m. This type of sifting essentially disposes of primality testing by trial-division, making the process of prime acquisition as simple as addition.

However, although the Sieve of Eratosthenes is simple in its concept and execution, it has a major flaw: a composite with j distinct prime factors can be eliminated from $\mathbb{N}$ up to j times. That is, an algorithm created to acquire primes in the same way as the Sieve of Eratosthenes will execute the process of elimination up to j times for each composite. This flaw translates into major inefficiency of computing power and time, for the number of distinct composites eliminated is less than the actual number of times the process of elimination is executed. The ratio between these numbers, what we will call our efficiency rate, seems to tend to some low percentage as the ceiling of the sieve increases. Therefore, executing the Sieve of Eratosthenes as an algorithm wastes time and computing power as it runs for longer periods of time.

Question 2. Can the Sieve of Eratosthenes (SOE) be modified to be more efficient?

In this project, we present a method through which to make the SOE, and more generally any sieve that uses the same type of execution algorithm, faster by increasing the efficiency rate of elimination. This is done by creating a new algorithm that essentially reduces the amount of composites needed to be eliminated by sifting a system of numbers with "lesser" order. This algorithm is called the $\mathbb{G}_i$ sieve.

Question 3. How is the $\mathbb{G}_i$ sieve different from the Sieve of Eratosthenes?

The $\mathbb{G}_i$ sieve algorithm (see Chapter 3) is based upon the premise of composite elimination used by the SOE. The main difference between the SOE and the $\mathbb{G}_i$ sieve is *how* they sift through $\mathbb{N}$ up to some number $m \in \mathbb{N}$. Whereas the SOE sifts straight through $\mathbb{N}$, the $\mathbb{G}_i$ sieve sifts through intervals defined by the squares of consecutive primes. Sifting an interval is one iteration performed by $\mathbb{G}_i$. Two benefits of sifting in this manner are: 1) rather than holding the set $\{2, \ldots, m\}$ in memory, the $\mathbb{G}_i$ sieve needs only hold a relatively smaller interval, allowing for more efficient use of memory; and, 2) it allows for

the perpetuality of execution. That is to say, sifting intervals gives the option to sift up to a ceiling *or* sift for a given time.

Another difference is that the SOE sifts through $\mathbb{N}$, whereas the $\mathbb{G}_i$ sieve sifts through the primitive residual classes modulo P_n, where P_n is the product over the first n primes. The index "n" is dependent on the index "i", where the relation is $n = i - 1$. (For $\mathbb{G}_1$, we define $P_0 = 1$).

Question 4. How "fast" is the $\mathbb{G}_i$ algorithm?

By sifting through the primitive residual classes modulo P_n, the $\mathbb{G}_i$ sieve essentially "pre-eliminates" multiples of the first n primes. This in turn will reduce the number of eliminations made by the algorithm during execution, increasing the efficiency of prime acquisition. We first time the $\mathbb{G}_i$ sieve algorithm for $1 \leq i \leq 7$ by trial-runs. Then we time the $\mathbb{G}_i$ sieve algorithm in steps. This not only allows us to understand the $\mathbb{G}_i$ sieve's performance for $1 \leq i \leq 7$ but also to model the $\mathbb{G}_i$ sieve's performance for $i \geq 8$.

Question 5. Sifting through primitive residual classes modulo P_n to increase the efficiency of prime acquisition comes at what price?

Because the $\mathbb{G}_i$ sieve sifts through primitive residual classes modulo P_n, we are dealing with a number system of "lesser order" than $\mathbb{N}$, requiring less eliminations to sift all primes out. This decrease in the number of eliminations results in more efficient prime acquisition. However, because we are concerned with eliminating all multiples of primes congruent to the primitive residual classes modulo P_n, we must create a set for each residual class to represent these multiples. This means that any prime p will have multiple sets associated with it, requiring more memory. With this information, we try to answer the following question.

Question 6. What are the viable uses of the $\mathbb{G}_i$ sieve?

The applications associated with sieves are not many, for other methods have been created to test for primality and to factor large composites that have been proven to work incredibly fast. But this type of sieve trumps other methods in that it is capable of acquiring a large *set* of primes. If the efficiency and speed of the $\mathbb{G}_i$ sieve prove viable, then it may become a competing method to accomplish these needs.

Brief Synopsis. In Chapter 2, we introduce all the terms, definitions, and analytical concepts to be used throughout this paper, along with basic properties of primes vital to the sifting process. We first explore properties of primes and the sieve method relevant to the SOE. Then we introduce the SOE through an example: we sift the set of numbers $\{2, \ldots, 100\}$ to see how the SOE works. Afterward, we generalize the SOE for any ceiling $m \in \mathbb{N}$. We use the concept of composite elimination used by the SOE to explore the method of counting primes up to some ceiling $m \in \mathbb{N}$. After exploring this counting method, we go on to describe more properties of primes that are relevant to the $\mathbb{G}_i$ sieve. Also, we introduce the tools to be used to model the $\mathbb{G}_i$ sieve algorithm's performance, namely, the Prime Number Theorem (PNT). We also introduce the notion of "average spacing" between primes. In Chapter 4, these two concepts aid in our discoveries of prime and composite densities of the sets we sift.

In Chapter 3, we introduce the sieves with which this project is concerned. We first work with the $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_3$ and $\mathbb{G}_4$ sieves to understand how they work and to see how they contrast with the SOE as well as with each other. We then segue into the abstract $\mathbb{G}_i$ sieve and its execution. Next, we use the premise of the $\mathbb{G}_i$ sieve to acquire a more efficient and thus quicker method of counting primes. Moreover, we conjecture a newer, more "closer" upper bound than the one found in Chapter 2.

In Chapter 4, we empirically and theoretically analyze the $\mathbb{G}_i$ sieve algorithm. That is, we first run the $\mathbb{G}_i$ sieve algorithm for $1 \leq i \leq 7$ and observe the behavior of three statistics: the number of primes acquired during a certain length of time; the number of iterations performed during a certain length of time; and the efficiency rate at the end of run-time. Next, we use the PNT and the concept of average spacing, introduced in Chapter 2, to estimate the prime density of the set $\{2, \ldots, p_k^2 - 1\}$, where p_k is the k^{th} prime for some arbitrarily large $k \in \mathbb{N}$. We then use this estimate to approximate the composite density of said set, thus giving us an idea of how many composite eliminations occur during the $\mathbb{G}_i$ sieve algorithm's run-time. Knowing the number of eliminations executed aids in timing the $\mathbb{G}_i$ sieve algorithm in steps and obtain a function in terms of some ceiling $m \in \mathbb{N}$ that will model how long it takes to reach m, where $m = p_k^2 - 1$.

2

Notations, Definitions, and Concepts for Sifting Analysis

In venturing into any mathematical journey, one is bound to come across a multitude of terms and notations that simplify concepts and make ideas more concise. Some terms are easily understood as they are used by convention over many fields; others not so much, especially if they represent ideas specific to a certain discipline. This project is no different in that a lot of symbols are utilized for brevity's sake. With that said, the following are terms that are generally universally used and understood over a wide range of mathematical fields.

- The natural numbers, integers, rationals, and reals are represented as $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$, respectively. We use $\mathbb{Z}^*$,$\mathbb{Q}^*$, $\mathbb{R}^*$ to represent the same sets minus the 0 element. When we refer to $\mathbb{N}$, we assume that $0 \notin \mathbb{N}$.

- The greatest common divisor of two integers a and b is represented (a, b). The greatest common divisor of n integers $a_1, \ldots, a_n$ is represented as $(a_1, \ldots, a_n)$. If $(a, b) = 1$, then a and b are said to be relatively prime.

- The largest integer less than or equal to some $a \in \mathbb{R}$ is denoted as $\lfloor a \rfloor$. This is called the floor function. If $a = \frac{b}{c} \in \mathbb{Q}$, then $\lfloor a \rfloor$ represents the integer part of the quotient.
- Similarly, the smallest integer greater than or equal to some $a \in \mathbb{R}$ is denoted as $\lceil a \rceil$. This is called the ceiling function.
- For any two numbers a and b, we write $a|b$ if a evenly divides b.
- The binomial coefficient is defined as $\binom{a}{b} = \frac{a!}{b!(a-b)!}$ for $a, b \in \mathbb{Z}$, $0 \leq a \leq b$.
- The prime counting function $\pi(a)$ is defined as the number of primes less than or equal to a.
- Let $f(x), g(x)$ be two real-value functions. If $\lim_{x\to\infty} \frac{f(x)}{g(x)} = 1$, then we say that $f(x)$ is asymptotic to $g(x)$ and is denoted as $f(x) \sim g(x)$.
- Euler's Totient function (or Euler's phi function), denoted as $\phi(x)$, is the number of positive integers less than or equal to x that are relatively prime to x. That is,

$$\phi(x) = \#\{c \in \mathbb{N} | c \leq x, (c, x) = 1\}.$$

  Let $x = x_1 \cdots x_k$. If $(x_i, x_j) = 1$ for all i, j such that $1 \leq i < j \leq k$, then $\phi(x) = \phi(x_1) \cdots \phi(x_k)$. Moreover, $\phi(x) = x \prod_{p|x} \left(1 - \frac{1}{p}\right)$. (See [1]).
- Let A be a finite set. We denote the i^{th} element of A as $A(i)$. If $A(i)$ is a set, then $A(i)(j)$ is the j^{th} element of the i^{th} element of A.
- Throughout this paper, when we write $[a, b]$, $[a, b)$, $(a, b]$, and (a, b), we mean to say

$$[a, b] = \{d \in \mathbb{Z} | a \leq d \leq b\}$$

$$[a, b) = \{d \in \mathbb{Z} | a \leq d < b\}$$

$$(a, b] = \{d \in \mathbb{Z} | a < d \leq b\}$$

$$(a, b) = \{d \in \mathbb{Z} | a < d < b\}.$$

2.1 Prime Properties Relevant to the Sieve Of Eratosthenes

Grasping the elusive nature of primes has been a goal of many mathematicians. Paramount to understanding the primes is establishing their properties. This is especially important when dealing with the sifting process. We start from the very beginning: knowing exactly what a prime number is.

Definition 2.1.1. Let $p \in \mathbb{N}$. We say p is prime if for all $c \in \mathbb{N}$, $c \leq p$, the statement $c|p$ is true only for $c \in \{1, p\}$. △

Basically, a number is prime if its only divisors are 1 and itself. The reason why a prime gets the name "prime" is because it is essentially the first in a sequence composed of all its positives multiples, itself being the first one. These positive multiples greater than p compose the set counterpart to the primes.

Definition 2.1.2. Let $c \in \mathbb{N}$. If there exists a $d \in \mathbb{N}$, $d \notin \{1, c\}$, such that $d|c$ is true, that is, the quotient $\frac{c}{d} \in \mathbb{Z}$, then c is called composite. △

Composites themselves are just the product of primes, as seen in the following theorem. For a proof, the interested reader should see [7].

Theorem 2.1.3 (Fundamental Theorem of Arithmetic)**.** *Let $c \in \mathbb{Z}$. Then*

$$c = p_1 \cdots p_k,$$

where $p_1, \ldots, p_k$ are primes (not necessarily distinct). Furthermore, this factorization is unique; that is, if

$$c = q_1 \cdots q_l,$$

then $k = l$ and the q_i's are just the p_i's rearranged.

Although we restrict our investigation to $c \in \mathbb{N}$, we see that Theorem 2.1.3 still applies. Implicit in Theorem 2.1.3's notation is that c is composite, with $k \geq 2$. However, if c were

prime, then $c = p_1$.

Of interest is the fact that there exists no largest prime. That is, there is an infinite number of primes in $\mathbb{N}$.

Proposition 2.1.4. *There are an infinite number of primes in* $\mathbb{N}$.

Proof. We prove by contradiction. That is, we assume that there are only a finite number of primes. This implies that there is a largest prime p_N. Let $Q = p_1 \cdots p_N + 1$. Obviously Q is not divisible by any $p_i \in \{p_1, \ldots, p_N\}$ or else $Q = p_i \left[(p_1 \cdots p_{i-1} p_{i+1} \cdots p_N) + \frac{1}{p_i} \right]$, which would imply that $\frac{1}{p_i} \in \mathbb{N}$. Thus, Q is not divisible by any of the primes in $\{p_1, \ldots, p_N\}$, which means that Q is either prime itself or a composite whose prime factors are not in $\{p_1, \ldots, p_N\}$. This leads to a contradiction since we assumed that the number of primes is finite. □

We now proceed to derive some fundamental properties of primes relevant to the sifting process.

Proposition 2.1.5. *Given any number* $g \in \mathbb{N}$*, if* g *is composite, that is, if* $g = g_1 \ldots g_j$*, then* g *has at least one factor* $g_i \in \{g_1, \ldots, g_j\}$ *such that* $g_i \leq \sqrt{g}$.

Proof. We will prove this by contradiction. We have that $g = g_1 \cdots g_j$. We assume that for all factors $g_i \in \{g_1, \ldots, g_j\}$, we have that $g_i \nleq \sqrt{g}$. That is, $g_i > \sqrt{g}$. We thus have that

$$\begin{aligned} g &= g_1 \cdots g_j \\ &> \underbrace{\sqrt{g} \cdots \sqrt{g}}_{j \text{ times}} \\ &= (\sqrt{g})^j. \end{aligned}$$

We see that $j \geq 2$ since g is composite. Then

$$\begin{aligned} g &> (\sqrt{g})^j \\ &= g(\sqrt{g})^{j-2}. \end{aligned}$$

This implies that $g > g(\sqrt{g})^{j-2}$, which is false since $\sqrt{g} \geq 1$ and $(\sqrt{g})^{j-2} \geq 1$. Therefore, there must exist at least one factor of g such that $g_i \leq \sqrt{g}$. □

Using Theorem 2.1.3 and Proposition 2.1.5, we derive the following:

Corollary 2.1.6. *If $g \in \mathbb{N}$ is composite, then g has at least one prime factor p such that $p \leq \sqrt{g}$.*

Proof. Let $g = g_1 \cdots g_j$. By Proposition 2.1.5, there exists at least one factor of g such that $g_i \leq \sqrt{g}$. If g_i is prime, we are done. If g_i is composite, then we go further. By Theorem 2.1.3, all positive integers can be written as a product of prime numbers. That is, $g_i = p_1 \cdots p_k$, where $p_1, \ldots, p_k$ are prime, although not necessarily distinct. Therefore, we have

$$\begin{aligned} g &= g_1 \cdots (g_i) \cdots g_j \\ &= g_1 \cdots (p_1 \cdots p_k) \cdots g_j. \end{aligned}$$

But $p_1, \ldots, p_k < g_i \leq \sqrt{g}$. Thus, g has a prime factor $p \in \{p_1, \ldots, p_k\}$ such that $p \leq \sqrt{g}$. □

With Corollary 2.1.6, we establish a fundamental rule in primality testing:

Corollary 2.1.7. *Let $g \in \mathbb{N}$. Then the only primes needed to test the primality of g are all $p \leq \sqrt{g}$. If no such p divides g, then g is prime.*

Example 2.1.8. Let $g = 881$, and let p be a prime number. Then the only primes needed to test the primality of g are those in the set $\{p | p \leq \sqrt{g}\}$. In this case, $\sqrt{g} \approx$

29.68. Therefore, we only need to use the primes $2, 3, 5, 7, 11, 13, 15, 17, 19, 23, 29$ to test the primality of g. By trial-division, we find out that none of these primes divides g, which shows that g is prime.

We can see from Example 2.1.8 that testing the primality of some large $g \in \mathbb{N}$ can be tedious. It is even more so if we had to test the primality of a whole set of numbers or a significantly larger number. This is where the process of sifting comes into play. Rather than trail-dividing every single number in some set by all relevant primes, we instead eliminate the multiples of all relevant primes from the set, which is essentially the purpose of sifting: to remove unwanted numbers and retain desired ones up to some number (what is called the ceiling). In the context of prime acquisition, this process "tests" primality by eliminating composites. Whichever numbers are left in the set are prime. This sieve method is a simple algorithm that goes as follows:

Sieve Method 2.1.9.

Input: Some $m \in \mathbb{N}$.

Output: Set of primes up to m.

This sieve method is simple in that it removes multiples of the "next" number until all numbers have been used.

Algorithm:[1]

```
m = ceiling
G = {2, 3, ..., m}
i = 1
while i < #G:
    h = 2G(i)
    while h ≤ m:
        if h ∈ G:
            G = G\{h}
            h = h + G(i)
```

[1]Indentation indicates a "Do" command. Nested blocks, or rather nested Do's, are thus indented multiple times. This pseudo-code is python language based.

else:
$h = h + G(i)$
$i = i + 1$
G is the set of primes less than or equal to m. △

Using Corollary 2.1.6 and Corollary 2.1.7, we can derive a fundamental rule of sifting used by the SOE and reduce the number of steps in Sieve Method 2.1.9.

Proposition 2.1.10. *Let $P = \{p_1, \ldots, p_j\}$ be the set of the first j primes, and let $p_j^2 \leq m < p_{j+1}^2$. Then the only primes needed to sift the set $G = \{2, \ldots, m\}$ are all $p \in P$.*

Proof. We have that $p_j^2 \leq m < p_{j+1}^2$, which implies that $p_j \leq \sqrt{m} < p_{j+1}$. By Corollary 2.1.7, we know that only primes less than or equal to $\sqrt{m}$ are needed to test the primality of m. These primes are precisely those in P. For all other composites $g \in G$, we have that $g < m$. Since $\sqrt{g} < \sqrt{m}$, then, by Corollary 2.1.6, g has a prime factor $p \in \{p_1, \ldots, p_j\}$ such that $p \leq \sqrt{g} < \sqrt{m}$, which implies that g has a prime factor in P, resulting in its elimination. □

Using Proposition 2.1.10, we transform Sieve Method 2.1.9 into the SOE's sieve method.

Sieve Method 2.1.11 (Sieve of Eratosthenes)**.**

Input: Some $m \in \mathbb{N}$.

Output: Set of primes up to m.

The SOE sifts the set $\{2, 3, \ldots, m\}$ using the "next" number less than $\sqrt{m}$.

Algorithm:[2]
$m =$ ceiling
$G = \{2, 3, \ldots, m\}$
$i = 1$
while $G(i) \leq \sqrt{m}$:
$h = 2G(i)$
while $h \leq m$:

[2]Indentation indicates a "Do" command. Nested blocks, or rather nested Do's, are thus indented multiple times. This pseudo-code is python language based.

if $h \in G$:
$G = G \backslash \{h\}$
$h = h + G(i)$
else:
$h = h + G(i)$
$i = i + 1$
G is the set of primes less than or equal to m. △

It is Sieve Method 2.1.11 and Proposition 2.1.10 that the Sieve of Eratosthenes uses to acquire primes.

Before we introduce the SOE and how it functions, we first define the terms that will be used.

2.2 Terms Used in the Sieve of Eratosthenes

In order to understand the sifting process of the SOE, we must define terms integral to sifting.

The SOE sifts a set of numbers for primes. We define this set first.

Definition 2.2.1. We define the set of natural numbers up to some ceiling m that the SOE sifts as

$$\mathcal{G}_m = \{g \in \mathbb{N} | 2 \leq g \leq m\}.$$

△

The sifting process essentially eliminates multiples of primes, leaving behind more primes. Therefore, we must gather the multiples of relevant primes into sets.

Definition 2.2.2. Let p_j be the j^{th} prime. Then the multiples of p_j needed to be removed from the set $\mathcal{G}_m$ form the set

$$\mathcal{S}_{j,m} = \{s \in p_j\mathbb{Z} | p_j < s \leq m\}.$$

The elements of $\mathcal{S}_{j,m}$ are obtained from the following recurrence sequence:

$$s_1 = 2p_j,$$

$$s_{j_{k+1}} = s_{j_k} + p_j.$$

△

We start the recurrence sequence at $s_1 = 2p_j$ because we want to eliminate the *multiples* of the prime p_j, and not p_j itself. We then add p_j to any s_{j_k} to obtain the next multiple of p_j.

We next define the set of primes as follows.

Definition 2.2.3. Let $\mathcal{P}[a, b]$ be the set of all primes in the interval $[a, b]$, where $a, b \in \mathbb{N}$. △

By sifting the set $\mathcal{G}_m$, we acquire the set of primes $\mathcal{P}[1, m]$.

2.3 Sieve of Eratosthenes

To clarify how the SOE's sifting process works, we work through an example.

Example 2.3.1 (Sieve of Eratosthenes)**.** Many textbooks introduce the Sieve of Eratosthenes visually, using a 10 x 10 grid containing the natural numbers $\{2, \ldots, 100\}$. This set of numbers is $\mathcal{G}_{100} = \{g \in \mathbb{N} | 2 \leq g \leq 100\}$. (See Figure 2.3.1.)

The use of this grid aids in quickly introducing the process of sifting with the SOE to obtain primes. In this specific grid, we sift $\mathcal{G}_{100}$ to obtain all primes $p \in \mathcal{G}_{100}$. The process begins by taking the first number, which is the first prime number, $p_1 = 2$, and then building the set of all its positive multiples. This set is $\mathcal{S}_{1,100} = \{s \in 2\mathbb{Z} | p_1 = 2 < s \leq 100\}$. To

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Figure 2.3.1. A Visualization of Pre-sifted $\mathcal{G}_{100}$

eliminate these multiples from $\mathcal{G}_{100}$, we simply subtract $\mathcal{S}_{1,100}$ from $\mathcal{G}_{100}$, resulting in the set of numbers in the grid in Figure 2.3.2.

	2	3		5		7		9	
11		13		15		17		19	
21		23		25		27		29	
31		33		35		37		39	
41		43		45		47		49	
51		53		55		57		59	
61		63		65		67		69	
71		73		75		77		79	
81		83		85		87		89	
91		93		95		97		99	

Figure 2.3.2. $\mathcal{G}_{100}$ After Being Sifted by 2

The grid in Figure 2.3.2 represents the new set $\mathcal{G}_{100}\backslash\mathcal{S}_{1,100}$. The following number after 2 that is not eliminated is our next prime number. In fact, this rule follows after the elimination of the multiples of any prime number, p. We now take 3 and eliminate all its multiples from $\mathcal{G}_{100}\backslash\mathcal{S}_{1,100}$. We define the set of positive multiples of $p_2 = 3$ as $\mathcal{S}_{2,100} = \{s \in 3\mathbb{Z} | p_2 = 3 < s \leq 100\}$. To eliminate these multiples, we substract $\mathcal{S}_{2,100}$ from $\mathcal{G}_{100}\backslash\mathcal{S}_{1,100}$, resulting in the set of numbers in the grid in Figure 2.3.3.

	2	3		5		7			
11		13				17		19	
		23		25				29	
31				35		37			
41		43				47		49	
		53		55				59	
61				65		67			
71		73				77		79	
		83		85				89	
91				95		97			

Figure 2.3.3. $\mathcal{G}_{100}$ After Being Sifted by 2 and 3

We are now left with the set $(\mathcal{G}_{100}\backslash\mathcal{S}_{1,100})\backslash\mathcal{S}_{2,100} = \mathcal{G}_{100}\backslash(\mathcal{S}_{1,100} \cup \mathcal{S}_{2,100})$. Continuing this process of eliminating all composites within this range, we are left with the set in the grid in Figure 2.3.4

	2	3		5		7			
11		13				17		19	
		23						29	
31						37			
41		43				47			
		53						59	
61						67			
71		73						79	
		83						89	
						97			

Figure 2.3.4. $\mathcal{G}_{100}$ After Being Fully Sifted

The grid in Figure 2.3.4 represents the fully sifted set $\mathcal{G}_{100}\backslash(\mathcal{S}_{1,100}\cup\mathcal{S}_{2,100}\cup\mathcal{S}_{3,100}\cup\mathcal{S}_{4,100})$. We know that we need only use the four sets $\mathcal{S}_{1,100}$, $\mathcal{S}_{2,100}$, $\mathcal{S}_{3,100}$, and $\mathcal{S}_{4,100}$ to sift $\mathcal{G}_{100}$ since, by Proposition 2.1.10, we need only use primes less than the square root of our ceiling, which in $\mathcal{G}_{100}$ is $\sqrt{100} = 10$. These four sets are the sets that represent the multiples of the primes 2, 3, 5, and 7.

We can generalize the Sieve of Eratosthenes for any ceiling m using sets analogous to those used in Example 2.3.1. Let the set that the SOE sifts be $\mathcal{G}_m = \{g \in \mathbb{N} | 2 \leq g \leq m\}$. To sift the primes out of $\mathcal{G}_m$, the SOE uses Sieve Method 2.1.11 to eliminate composites. By building the set $\mathcal{S}_{j,m}$ for each prime $p_j \leq \sqrt{m}$ consisting of p_j's positive multiples, we are gathering every composite in $\mathcal{G}_m$. To sift $\mathcal{G}_m$, we need only subtract the sets $\mathcal{S}_{j,m}$ from $\mathcal{G}_m$.

Proposition 2.3.2. *The sets sufficient to sift $\mathcal{G}_m$ via the SOE are the sets $\mathcal{S}_{j,m} = \{s \in p_j\mathbb{Z} | p_j < s \leq m\}$ that correspond to the primes $p_j \leq \sqrt{m}$.*

Proof. By Proposition 2.1.10, we know that we need only use primes less than or equal to $\sqrt{m}$ to sift $\mathcal{G}_m$. □

Recalling that $\mathcal{P}[a, b]$ is the set of all primes in the interval $[a, b]$, we have that the SOE provides the set $\mathcal{P}[1, m]$ after execution is complete. By Proposition 2.1.10, we know that the primes in $\mathcal{P}[1, \sqrt{m}\,]$ are sufficient to sift up to m. After sifting, the SOE acquires all other primes in $\mathcal{P}[1, m]$ not in $\mathcal{P}[1, \sqrt{m}\,]$. That is, SOE acquires $\mathcal{P}[1, m] \backslash \mathcal{P}[1, \sqrt{m}\,] = \mathcal{P}[\sqrt{m}, m]$.

We see that the SOE acquires $\mathcal{P}[1, m]$ by subtracting the set $\bigcup_{j=1}^{\#\mathcal{P}[1,\sqrt{m}\,]} \mathcal{S}_{j,m}$ from $\mathcal{G}_m$. That is,

Proposition 2.3.3. *The set of primes $\mathcal{P}[1, m]$ can be written as*

$$\mathcal{P}[1, m] = \mathcal{G}_m \backslash \bigcup_{j=1}^{\#\mathcal{P}[1,\sqrt{m}\,]} \mathcal{S}_{j,m}. \tag{2.3.1}$$

Proof. This is a direct consequence of Propositions 2.1.10 and 2.3.2. □

In executing the SOE, we can follow the procedure outlined in Example 2.3.1, in which we assume only the first prime and then follow the iterative process detailed in Sieve

Method 2.1.11 to eliminate composites. However, Proposition 2.3.3 assumes that $\mathcal{P}[1, \sqrt{m}\,]$ is already known. Using either procedure, we ultimately end up with the set equality in (2.3.1).

2.4 Bounding Prime Density Estimates Using The SOE

Although acquiring primes is the main goal of the SOE, we can use it to count the number of primes less than or equal to some $m \in \mathbb{N}$. To do so, we use a combinatorial counting method called the Principle of Inclusion-Exclusion.

Theorem 2.4.1 ([10, Theorem 7.5.1])**.** *Let $A_1, A_2, \ldots, A_n$ be finite sets. Then*

$$\begin{aligned}\#(A_1 \cup A_2 \cup \cdots \cup A_n) = &\sum_{1\le i\le n} \#A_i - \sum_{1\le i<j\le n} \#(A_i \cap A_j) \\ &+ \sum_{1\le i<j<k\le n} \#(A_i \cap A_j \cap A_k) - \cdots + (-1)^{n+1} \cdot \#(A_1 \cap A_2 \cap \cdots \cap A_n).\end{aligned} \tag{2.4.1}$$

Proof. We must show that for any element $a \in (A_1 \cup A_2 \cup \cdots \cup A_n)$, a is counted exactly once by the right-hand side of (2.4.1). Suppose that a is in exactly r sets of the sets $A_1, A_2, \ldots, A_n$, where $1 \le r \le n$. Then a is counted $\binom{r}{1}$ times by $\sum \#A_i$. It is counted $\binom{r}{2}$ times by $\sum \#(A_i \cap A_j)$. In general, it is counted $\binom{r}{s}$ times by the summation involving s of the sets A_i. Then a is counted

$$\binom{r}{1} - \binom{r}{2} + \binom{r}{3} - \cdots + (-1)^{r+1}\binom{r}{r}$$

times by the right-hand side of (2.4.1). But

$$\begin{aligned}\binom{r}{0} - \binom{r}{1} + \binom{r}{2} - \binom{r}{3} + \cdots + (-1)^{r+1}\binom{r}{r} &= \sum_{i=0}^{r}(-1)^i\binom{r}{i} \\ &= \sum_{i=0}^{r}(-1)^i(1)^r\binom{r}{i} \\ &= (1-1)^r \\ &= 0.\end{aligned}$$

Then

$$\begin{aligned}\binom{r}{1} - \binom{r}{2} + \binom{r}{3} - \cdots + (-1)^{r+1}\binom{r}{r} &= \binom{r}{0} \\ &= 1.\end{aligned}$$

Therefore, each element in the union is counted exactly once by the expression on the right-hand side of (2.4.1). □

In the context of counting primes, Theorem 2.4.1 allows us to count the number of composites in the set $\{2, \ldots, m\}$. By subtracting this value from the ceiling m, we come closer to deriving the value $\pi(m)$. We elucidate the use of Theorem 2.4.1 through an example.

Example 2.4.2. Let us find $\pi(100)$. That is, we count the primes in $\mathcal{G}_{100}$. From Proposition 2.1.10, we need only eliminate the multiples of the first four primes $2, 3, 5$, and 7. Let

$$A_1 = \{a \in 2\mathbb{Z} | 1 < a \leq 100\},$$

$$A_2 = \{a \in 3\mathbb{Z} | 1 < a \leq 100\},$$

$$A_3 = \{a \in 5\mathbb{Z} | 1 < a \leq 100\}, \text{ and}$$

$$A_4 = \{a \in 7\mathbb{Z} | 1 < a \leq 100\}.$$

The set A_i represents the multiples of p_i in the set $\mathcal{G}_{100}$, where $\#A_i = \left\lfloor \frac{100}{p_i} \right\rfloor$. We must find $\#(A_1 \cup A_2 \cup A_3 \cup A_4)$. By Theorem 2.4.1, we know that

$$\begin{aligned}\#(A_1 \cup A_2 \cup A_3 \cup A_4) = {} & \#(A_1) + \#(A_2) + \#(A_3) + \#(A_4) \\ & - \#(A_1 \cap A_2) - \#(A_1 \cap A_3) - \#(A_1 \cap A_4) \\ & - \#(A_2 \cap A_3) - \#(A_2 \cap A_4) - \#(A_3 \cap A_4) \\ & + \#(A_1 \cap A_2 \cap A_3) + \#(A_1 \cap A_2 \cap A_4) \\ & + \#(A_2 \cap A_3 \cap A_4) - \#(A_1 \cap A_2 \cap A_3 \cap A_4).\end{aligned}$$

We know the value for $\#A_i$. To obtain the cardinality of the intersections, we note that the composites in any intersection are precisely those that are divisible by the primes that compose the elements of the sets that are intersected. That is, if a composite c is in $(A_i \cap A_j)$, $1 \leq i < j \leq 4$, then c is divisible by p_i and p_j, and $\#(A_i \cap A_j) = \left\lfloor \frac{100}{p_i \cdot p_j} \right\rfloor$. Likewise, if c is in $(A_i \cap A_j \cap A_k)$, $1 \leq i < j < k \leq 4$, then c is divisible by p_i, p_j, and p_k, and $\#(A_i \cap A_j \cap A_k) = \left\lfloor \frac{100}{p_i \cdot p_j \cdot p_k} \right\rfloor$. The values of the cardinality of all intersections are similarly defined. Then

$$\begin{aligned}\#(A_1 \cup A_2 \cup A_3 \cup A_4) = {} & \left\lfloor \frac{100}{2} \right\rfloor + \left\lfloor \frac{100}{3} \right\rfloor + \left\lfloor \frac{100}{5} \right\rfloor + \left\lfloor \frac{100}{7} \right\rfloor \\ & - \left\lfloor \frac{100}{2 \cdot 3} \right\rfloor - \left\lfloor \frac{100}{2 \cdot 5} \right\rfloor - \left\lfloor \frac{100}{2 \cdot 7} \right\rfloor - \left\lfloor \frac{100}{3 \cdot 5} \right\rfloor - \left\lfloor \frac{100}{3 \cdot 7} \right\rfloor - \left\lfloor \frac{100}{5 \cdot 7} \right\rfloor \\ & + \left\lfloor \frac{100}{2 \cdot 3 \cdot 5} \right\rfloor + \left\lfloor \frac{100}{2 \cdot 3 \cdot 7} \right\rfloor + \left\lfloor \frac{100}{2 \cdot 5 \cdot 7} \right\rfloor + \left\lfloor \frac{100}{3 \cdot 5 \cdot 7} \right\rfloor \\ & - \left\lfloor \frac{100}{2 \cdot 3 \cdot 5 \cdot 7} \right\rfloor \\ = {} & 50 + 33 + 20 + 14 - 16 - 10 - 7 - 6 - 4 - 2 + 3 + 2 + 1 + 0 - 0 \\ = {} & 78.\end{aligned}$$

To obtain $\pi(100)$, we first note that $\#\mathcal{G}_{100} = 99$. Next, we subtract $\#(A_1 \cup A_2 \cup A_3 \cup A_4)$ from this value. However, the four primes that we used to sift have been eliminated by the sets A_i, $1 \leq i \leq 4$. Therefore, we must add 4 to our result. This yields

$\pi(100) = 99 - 78 + 4 = 25$.

In order to come up with a generalized method of obtaining $\pi(m)$ for any ceiling $m \in \mathbb{N}$, we first define the following.

Definition 2.4.3. Let A be a set. Then the power set of A is the set of all subsets of A. We denote the power set of A as $X(A)$. We denote the power set of A minus the empty set as $X^*(A) = X(A)\backslash\{\}$. △

We can now generalize the counting method used in Example 2.4.2 to evaluate $\pi(m)$ for any $m \in \mathbb{N}$.

Proposition 2.4.4. *Let $\mathcal{P}[1, \sqrt{m}\,]$ be the set of primes we use to sift the set $\mathcal{G}_m = \{2, \ldots, m\}$. Also, let $X(\mathcal{P}[1, \sqrt{m}\,])$ be the power set of $\mathcal{P}[1, \sqrt{m}\,]$, with x signifying an element of $X(\mathcal{P}[1, \sqrt{m}\,])$. Then we can count the primes up to m using the following formula.*

$$\pi(m) = (m-1) + \#\mathcal{P}[1, \sqrt{m}\,] - \sum_{x \in X^*(\mathcal{P}[1, \sqrt{m}\,])} (-1)^{\#x+1} \left\lfloor \frac{m}{\prod_{p \in x} p} \right\rfloor. \tag{2.4.2}$$

Proof. We build the sets $A_i = \{a \in p_i\mathbb{Z} | 1 < a \leq m\}$ for $1 \leq i \leq \#\mathcal{P}[1, \sqrt{m}\,]$. We have the values

$$\begin{aligned}
\#(A_i) &= \left\lfloor \frac{m}{p_i} \right\rfloor,\ 1 \leq i \leq \#\mathcal{P}[1, \sqrt{m}\,], \\
\#(A_i \cap A_j) &= \left\lfloor \frac{m}{p_i \cdot p_j} \right\rfloor,\ 1 \leq i < j \leq \#\mathcal{P}[1, \sqrt{m}\,], \\
\#(A_i \cap A_j \cap A_k) &= \left\lfloor \frac{m}{p_i \cdot p_j \cdot p_k} \right\rfloor,\ 1 \leq i < j < k \leq \#\mathcal{P}[1, \sqrt{m}\,], \\
&\vdots \qquad\qquad\qquad \vdots \\
\#(A_i \cap \cdots \cap A_{\#\mathcal{P}[1, \sqrt{m}\,]}) &= \left\lfloor \frac{m}{p_i \cdots p_{\#\mathcal{P}[1, \sqrt{m}\,]}} \right\rfloor.
\end{aligned}$$

For every term above, the values in the denominators are the products of the elements in the subsets of $X^*(\mathcal{P}[1,\sqrt{m}\,])$. Then by the inclusion-exclusion principle,

$$\#(A_1 \cup \cdots \cup A_{\mathcal{P}[1,\sqrt{m}\,]}) = \sum_{x \in X^*(\mathcal{P}[1,\sqrt{m}\,])} (-1)^{(\#x)+1} \left\lfloor \frac{m}{\prod_{p\in x} p} \right\rfloor.$$

To obtain $\pi(m)$, we first note that $\#\mathcal{G}_m = (m-1)$. We subtract $\#(A_1 \cup \cdots \cup A_{\mathcal{P}[1,\sqrt{m}\,]})$ from this value. However, the primes in $\mathcal{P}[1,\sqrt{m}\,]$ have been eliminated by the sets A_i for $1 \le i \le \#\mathcal{P}[1,\sqrt{m}\,]$. Therefore, we must add $\#\mathcal{P}[1,\sqrt{m}\,]$ to our result. Thus, we have derived (2.4.2). □

This counting method requires the knowledge of the set $X(\mathcal{P}[1,\sqrt{m}\,])$ before execution, which is not hard to obtain using a standard program like Sage. As the ceiling m increases, so does the cardinality of the set of primes required to sift $\mathcal{G}_m$. In turn, this formula becomes increasingly difficult and improbable to compute by hand, requiring many terms to be evaluated. By using power sets to obtain the denominator in each term, we can count the number of terms that the sum in (2.4.2) has.

Proposition 2.4.5. *For any $m \in \mathbb{N}$, the number of terms needed to evaluate $\pi(m)$ using (2.4.2) is $2^{\#\mathcal{P}[1,\sqrt{m}\,]} + 1$.*

Proof. We have that

$$\begin{aligned}
\#X^*(\mathcal{P}[1,\sqrt{m}\,]) &= \sum_{i=1}^{\#\mathcal{P}[1,\sqrt{m}\,]} \binom{\#\mathcal{P}[1,\sqrt{m}\,]}{i} \\
&= \sum_{i=1}^{\#\mathcal{P}[1,\sqrt{m}\,]} \binom{\#\mathcal{P}[1,\sqrt{m}\,]}{i} + 1 - 1 \\
&= \sum_{i=1}^{\#\mathcal{P}[1,\sqrt{m}\,]} \binom{\#\mathcal{P}[1,\sqrt{m}\,]}{i} + \binom{\#\mathcal{P}[1,\sqrt{m}\,]}{0} - 1 \\
&= \sum_{i=0}^{\#\mathcal{P}[1,\sqrt{m}\,]} \binom{\#\mathcal{P}[1,\sqrt{m}\,]}{i} - 1 \\
&= 2^{\#\mathcal{P}[1,\sqrt{m}\,]} - 1.
\end{aligned}$$

Then the sum in (2.4.2) has $2^{\#\mathcal{P}[1,\sqrt{m}]} - 1$ terms. Since we also have the terms "$(m-1)$" and "$\#\mathcal{P}[1,\sqrt{m}]$", then the total number of terms needed to evaluate $\pi(m)$ is $2^{\#\mathcal{P}[1,\sqrt{m}]} - 1 + 2 = 2^{\#\mathcal{P}[1,\sqrt{m}]} + 1$. □

It is the counting method in Proposition 2.4.4 that we use to show the increase in efficiency of execution of the $\mathbb{G}_i$ algorithm later on. That is, we show that we can reduce the number of terms needed to evaluate $\pi(m)$.

Another counting method, taken from [1], which does not require the use of power sets, uses a tool defined as follows.

Definition 2.4.6. The Möbius function is the function $\mu : \mathbb{N} \to \{0, \pm 1\}$ defined by

$$\mu(n) = \begin{cases} 1 & \text{if } n = 1, \\ 0 & \text{if } \exists\ p \text{ such that } \ p^2 | n, \\ (-1)^r & \text{if } n = p_1 \cdots p_r \text{ where } p_1, \ldots, p_r \text{ are distinct primes.} \end{cases}$$

△

It is easy to see that

$$\mu(mn) = \mu(m)\mu(n) \qquad \text{if } (m,n) = 1$$

and

$$\mu(mn) = 0 \qquad \text{if } (m,n) > 1.$$

One very interesting property of the Möbius function is the following:

Proposition 2.4.7. *For all $n \in \mathbb{N}$, we have*

$$\sum_{d|n} \mu(d) = \begin{cases} 1 & \textit{if } n = 1, \\ 0 & \textit{otherwise}\ . \end{cases}$$

Proof. Suppose $n = p_1^{e_1} \cdots p_r^{e_r}$ where $p_1, \ldots, p_r$ are distinct primes and $e_1, \ldots, e_r \in \mathbb{N}$. Then

$$\sum_{d|n} \mu(d) = \sum_{d|p_1 \cdots p_r} \mu(d) = 1 + (-1)\binom{r}{1} + \cdots + (-1)^r \binom{r}{r} = (1-1)^r = 0.$$

□

Using the Möbius function, we can find the inverse of a function that is in terms of its divisors using the following property. We state the result and proof as found in [1].

Proposition 2.4.8 (The Möbius Inversion)**.** *If*

$$f(m) = \sum_{d|m} g(d),$$

then

$$\begin{aligned} g(m) &= \sum_{d|m} \mu(d) f(\frac{m}{d}) \\ &= \sum_{d|m} \mu(\frac{m}{d}) f(d). \end{aligned}$$

Proof. For the first equality, we have

$$\begin{aligned} \sum_{d|m} \mu(d) f\left(\frac{m}{d}\right) &= \sum_{d|m} \mu(d) \sum_{l|(\frac{m}{d})} g(l) \\ &= \sum_{l|m} g(l) \left(\sum_{d|(\frac{m}{l})} \mu(d) \right) \\ &= \sum_{l=m} g(l) \\ &= g(m). \end{aligned}$$

For the second equality, we have

$$\begin{aligned}\sum_{d|m}\mu(\frac{m}{d})f(d) &= \sum_{d|m}\mu\left(\frac{m}{d}\right)\sum_{l|d}g(l)\\ &= \sum_{l|m}g(l)\sum_{d|(\frac{m}{l})}\mu\left(\frac{m}{dl}\right)\\ &= \sum_{l=m}g(l)\\ &= g(m).\end{aligned}$$

□

Example 2.4.9. To see how the two inversions in Proposition 2.4.8 function, we use a well-known identity of Euler's function that is very interesting, as mentioned in [11]. For any $m \in \mathbb{N}$, we have

$$m = \sum_{d|m}\phi(d). \tag{2.4.3}$$

Applying both inversions in Proposition 2.4.8 to (2.4.3) results in

$$\begin{aligned}\phi(m) &= \sum_{d|m}\mu\left(\frac{m}{d}\right)d\\ &= m\sum_{d|m}\mu(d)\frac{1}{d}.\end{aligned} \tag{2.4.4}$$

Recalling that $\phi(m) = m\prod_{p|m}\left(1-\frac{1}{p}\right)$, we therefore have that

$$m\sum_{d|m}\mu(d)\frac{1}{d} = m\prod_{p|m}\left(1-\frac{1}{p}\right). \tag{2.4.5}$$

Since we use the primes in $\mathcal{P}[1,\sqrt{m}\,]$ to sift up to m, we can begin to count the primes in the set $\{2,\ldots,m\}$ by first eliminating those numbers not relatively prime to $P_{\#\mathcal{P}[1,\sqrt{m}\,]}$, where $P_{\#\mathcal{P}[1,\sqrt{m}\,]} = \prod_{i=1}^{\#\mathcal{P}[1,\sqrt{m}\,]} p_i$. These numbers are the multiples of the first n primes. The

rest of the primes in $\{2, \ldots, m\}$ are precisely those used to create the product $P_{\#\mathcal{P}[1,\sqrt{m}]}$, i.e. $p_1, \ldots, p_{\#\mathcal{P}[1,\sqrt{m}]}$. Therefore, we begin by counting all numbers relatively prime to $P_{\#\mathcal{P}[1,\sqrt{m}]}$. Let the set of these numbers be

$$S = \{l | l \leq m, (l, P_{\#\mathcal{P}[1,\sqrt{m}]}) = 1\}.$$

Then

$$\#S = \pi(m) - \#\mathcal{P}[1, \sqrt{m}\,] + 1.$$

Note that the number $1 \in S$. Therefore, we add 1 to account for this. In order to count this value, we create the characteristic function $s(l)$ of the set S, where

$$s(l) = \begin{cases} 1 & \text{if } l \in S, \\ 0 & \text{otherwise.} \end{cases}$$

Using the Möbius property from Proposition 2.4.7, we can restate $s(l)$ as

$$s(l) = \sum_{d|(l, P_{\#\mathcal{P}[1,\sqrt{m}\,]})} \mu(d). \tag{2.4.6}$$

With (2.4.6), we derive the following count for $\#S$.

Theorem 2.4.10 ([1])**.** *Let $S = \{l | l \leq m, (l, P_{\#\mathcal{P}[1,\sqrt{m}]}) = 1\}$, where $m \in \mathbb{N}$. An upper bound for the cardinality of S is*

$$\#S \leq m \prod_{p|P_{\#\mathcal{P}[1,\sqrt{m}\,]}} \left(1 - \frac{1}{p}\right) + 2^{\#\mathcal{P}[1,\sqrt{m}\,]}.$$

Proof. We begin by expressing $\#S$ in terms of (2.4.6).

$$\begin{aligned} \#S &= \sum_{l \leq m} s(l) \\ &= \sum_{l \leq m} \left(\sum_{d|(l, P_{\#\mathcal{P}[1,\sqrt{m}\,]})} \mu(d) \right). \end{aligned} \tag{2.4.7}$$

The sum in the parenthesis essentially counts every number relatively prime to $P_{\#\mathcal{P}[1,\sqrt{m}]}$ by using the fact that if $(l, P_{\#\mathcal{P}[1,\sqrt{m}]}) \neq 1$, then the sum over l's divisors will be 0, whereas

if $(l, P_{\#\mathcal{P}[1,\sqrt{m}]}) = 1$, then the sum over l's divisors will equal 1, which is a consequence of the definition of the Möbius function. For l such that $(l, P_{\#\mathcal{P}[1,\sqrt{m}]}) = 1$, we note that l is in fact prime. Therefore, this sum counts the primes less than m but greater than $p_{\#\mathcal{P}[1,\sqrt{m}]}$. However, counting the primes in S is equivalent to eliminating the composites from S. By this logic, we see that $\sum_{l\leq m}\left(\sum_{d|(l,P_{\#\mathcal{P}[1,\sqrt{m}]})}\mu(d)\right) = \sum_{d|P_{\#\mathcal{P}[1,\sqrt{m}]}}\mu(d)\left(\sum_{\substack{l\leq m\\ d|l}}1\right)$ by Inclusion-Exclusion. Therefore, we have

$$\begin{aligned}
\#S &= \sum_{l\leq m}\left(\sum_{d|(l,P_{\#\mathcal{P}[1,\sqrt{m}]})}\mu(d)\right)\\
&= \sum_{d|P_{\#\mathcal{P}[1,\sqrt{m}]}}\mu(d)\left(\sum_{\substack{l\leq m\\ d|l}}1\right)\\
&= \sum_{d|P_{\#\mathcal{P}[1,\sqrt{m}]}}\mu(d)\left\lfloor\frac{m}{d}\right\rfloor \qquad (2.4.8)\\
&= \sum_{d|P_{\#\mathcal{P}[1,\sqrt{m}]}}\mu(d)\left(\frac{m}{d}+\left\lfloor\frac{m}{d}\right\rfloor-\frac{m}{d}\right)\\
&= \sum_{d|P_{\#\mathcal{P}[1,\sqrt{m}]}}\mu(d)\frac{m}{d}+\sum_{d|P_{\#\mathcal{P}[1,\sqrt{m}]}}\mu(d)\left(\left\lfloor\frac{m}{d}\right\rfloor-\frac{m}{d}\right). \qquad (2.4.9)
\end{aligned}$$

We first bound the second sum in (2.4.9). In this sum, we are summing over the divisors of the product over the first $\#\mathcal{P}[1,\sqrt{m}]$ primes. Obviously, 1 is a divisor. We can count the rest of the divisors the same way we counted the terms for the sum in (2.4.2). These other divisors of $P_{\#\mathcal{P}[1,\sqrt{m}]}$ are precisely the products over the elements of the subsets of $X^*(\mathcal{P}[1,\sqrt{m}])$. Since $\#X^*(\mathcal{P}[1,\sqrt{m}]) = \sum_{i=1}^{\#\mathcal{P}[1,\sqrt{m}]}\binom{\#\mathcal{P}[1,\sqrt{m}]}{i}$, the total number of terms in the second sum of (2.4.9) is

$$\begin{aligned}
\#A^*(\mathcal{P}[1,\sqrt{m}])+1 &= \sum_{i=1}^{\#\mathcal{P}[1,\sqrt{m}]}\binom{\#\mathcal{P}[1,\sqrt{m}]}{i}+1\\
&= \sum_{i=0}^{\#\mathcal{P}[1,\sqrt{m}]}\binom{\#\mathcal{P}[1,\sqrt{m}]}{i}\\
&= 2^{\#\mathcal{P}[1,\sqrt{m}]}.
\end{aligned}$$

Given the fact that $\left|\left\lfloor \frac{m}{d} \right\rfloor - \frac{m}{d}\right| \leq 1$, we can bound the second sum by $2^{\#\mathcal{P}[1,\sqrt{m}\,]}$. Using the equality in (2.4.5), we get the upper bound

$$\begin{aligned} \#S &= \sum_{d|P_{\#\mathcal{P}[1,\sqrt{m}\,]}} \mu(d)\frac{m}{d} + \sum_{d|P_{\#\mathcal{P}[1,\sqrt{m}\,]}} \mu(d)\left(\left\lfloor \frac{m}{d} \right\rfloor - \frac{m}{d}\right) \\ &\leq m \sum_{d|P_{\#\mathcal{P}[1,\sqrt{m}\,]}} \frac{\mu(d)}{d} + 2^{\#\mathcal{P}[1,\sqrt{m}\,]} \\ &= m \prod_{p|P_{\#\mathcal{P}[1,\sqrt{m}\,]}} \left(1 - \frac{1}{p}\right) + 2^{\#\mathcal{P}[1,\sqrt{m}\,]}. \end{aligned} \tag{2.4.10}$$

□

Given the bound obtained in Theorem 2.4.10, we also have an upper bound for the prime counting function since $\pi(m) = \#S + \#\mathcal{P}[1, \sqrt{m}\,] - 1$. However, this bound is quite inferior since (2.4.10) increases much faster than $\pi(m)$ as m increases. We make this upper bound better in Section 3.7 using concepts to be introduced later.

2.5 Prime Properties Relevant to $\mathbb{G}_i$

Before we introduce the $\mathbb{G}_i$ sieve in Chapter 3, we first go over certain properties of primes that justify the sifting process that $\mathbb{G}_i$ uses to obtain primes.

Definition 2.5.1. We define the product over the first n primes to be

$$P_n = \prod_{j=1}^{n} p_j.$$

We define P_0 to equal 1. △

Because we deal with residual classes in the $\mathbb{G}_i$ algorithm, we must introduce the concept of modulo arithmetic.

Definition 2.5.2. Let $a, b \in \mathbb{Z}$, $c \in \mathbb{N}$. Then when a divided by c leaves the remainder b, we write

$$a \equiv b \bmod c.$$

△

Definition 2.5.2 implies that if $a \equiv b \bmod c$, then a can be written as $a = ck + b$ for some $k \in \mathbb{Z}$.

Using the concept of modulo arithmetic, we derive the following property of primes.

Proposition 2.5.3. *Let p be prime. Then for all primes $q > p$, one of the following congruences are true.*

$$\begin{aligned} q &\equiv 1 \bmod p, \\ q &\equiv 2 \bmod p, \\ &\vdots \qquad \vdots \\ q &\equiv (p-1) \bmod p. \end{aligned}$$

Proof. We prove by contradiction. Let $q \equiv 0 \bmod p$. This implies that $q = kp + 0 = kp$ for some $k \in \mathbb{N}$. But this implies that $p|q$, which contradicts the fact that q is prime. □

Let p_n be the n^{th} prime. From Proposition 2.5.3, we see that for any prime $p > p_n$, one of the congruences from each column in Figure 2.5.1 is true. That is, for each modulo

$1 \bmod 2$	$1 \bmod 3$	$1 \bmod 5$	$\cdots$	$1 \bmod p_n$
	$2 \bmod 3$	$2 \bmod 5$	$\cdots$	$2 \bmod p_n$
		$3 \bmod 5$	$\cdots$	$3 \bmod p_n$
		$4 \bmod 5$	$\cdots$	$\cdots$
				$\cdots$
				$(p_n - 1) \bmod p_n$

Figure 2.5.1. All possible systems of modulo congruences

$p_i \leq p_n$, we have $(p_i - 1) = \phi(p_i)$ possible congruences for $p > p_n$. Since $(p_i, p_j) = 1$ for $1 \leq i < j \leq n$, then there are $\phi(p_1)\phi(p_2)\cdots\phi(p_n) = \phi(P_n)$ possible combinations of

congruences for any given $p > p_n$. That is,

$$\begin{aligned} p &\equiv t_1 \bmod p_1, \\ &\equiv t_2 \bmod p_2, \\ &\vdots \\ &\equiv t_n \bmod p_n, \end{aligned}$$

where $1 \leq t_i \leq (p_i - 1)$. We can express the above system of congruences as a sequence of t_i's by writing $p \equiv (t_1, \ldots, t_n)$. These $\phi(P_n)$ distinct sequences of t_i's does not elucidate much since we are expressing p in terms of multiple linear congruences. We must find a way to solve this system so that we can express the congruency of p in terms of one linear congruence. Solving each one of these possible systems brings us one step closer to understanding the possible congruences of all $p > p_n$ modulo one value. To solve these systems, we need more tools. The first is the following theorem, as taken from [7].

Theorem 2.5.4. *Let a and b be nonzero integers. Then there exist integers r and s such that*

$$(a, b) = ar + bs$$

where (a, b) is the unique greatest common divisor of a and b.

A specific case of Theorem 2.5.4 follows here.

Corollary 2.5.5. *Let a and b be two integers that are relatively prime. Then there exist integers r and s such that $ar + bs = 1$.*

The method through which to acquire the greatest common divisor of two integers a and b along with the two integers r and s such that $ar + bs = (a, b)$ requires the use of the well-known Euclidean Algorithm, as taken from [3].

Theorem 2.5.6 (Euclidean Algorithm)**.** *The minimal nonzero remainder r_m in the following set of equations is the greatest common divisor of the two positive integers a and b:*

$$\begin{aligned} b &= aq_1 + r_1, & 0 < r_1 < a, \\ a &= r_1q_2 + r_2, & 0 < r_2 < r_1, \\ r_1 &= r_2q_3 + r_3, & 0 < r_3 < r_2, \\ &\vdots \quad \vdots & \vdots \\ r_{m-2} &= r_{m-1}q_m + r_m, & 0 < r_m < r_{m-1}, \\ r_{m-1} &= r_mq_{m+1}. \end{aligned}$$

Through back-substitution, we obtain the linear combination in Theorem 2.5.4.

Because we need to solve systems of linear congruences, we introduce the following method of solving these systems, whose proof is found in [7].

Theorem 2.5.7 (Chinese Remainder Theorem)**.** *Let $m, n \in \mathbb{N}$ such that $(m, n) = 1$. That is, there exist $r, s \in \mathbb{Z}$ such that $mr + ns = 1$. Then for $a, b \in \mathbb{N}$, the system*

$$x \equiv a \bmod m,$$

$$x \equiv b \bmod n,$$

has a unique solution modulo mn. This solution is $x = a + (b - a)rm \bmod mn$.

The Chinese Remainder Theorem can be generalized to systems with more than two linear congruences by performing the CRT to two linear congruences at a time.

Example 2.5.8. Say we have the following system of two linear congruences,

$$x \equiv 3 \bmod 5,$$

$$x \equiv 4 \bmod 6.$$

Since $(5, 6) = 1$, then the CRT states that there exists a unique solution modulo 30. Notice that $r = -1$ and $s = 1$ is a solution set to $5r + 6s = 1$. Knowing that $a = 3$, $b = 4$, $m = 5$, $n = 6$, and $r = -1$, we have that

$$\begin{aligned} x &\equiv a + (b - a)rm \bmod mn \\ &\equiv 3 + (4 - 3)(-1)(5) \bmod 30 \\ &\equiv 3 - 5 \bmod 30 \\ &\equiv 28 \bmod 30. \end{aligned}$$

We see that $28 \equiv 3 \bmod 5 \equiv 4 \bmod 6$.

To solve each system of linear congruences formed by choosing a congruence from each column in Figure 2.5.1, we use the Chinese Remainder Theorem to get the solutions modulo P_n. What the CRT tells us is that every one of the combinations $(t_1, \ldots, t_n)$, $1 \leq t_i \leq (p_i - 1)$, gives a *unique* solution modulo P_n. Since there are $(p_1 - 1)(p_2 - 1) \cdots (p_n - 1) = \phi(p_1)\phi(p_2) \cdots \phi(p_n) = \phi(P_n)$ combinations, then there are $\phi(P_n)$ unique solutions. Therefore, we now know that for any prime $p > p_n$, p is congruent to one of $\phi(P_n)$ residual classes modulo P_n. The next statement demonstrates what these $\phi(P_n)$ residual classes look like.

Proposition 2.5.9. *Let p be a prime greater than the n^{th} prime p_n. Then p has the following property:*

$$p \equiv t \bmod P_n,$$

for some $t \in \mathbb{N}$ such that $(t, P_n) = 1$.

Proof. We prove by contradiction. Let p be a prime such that $p \equiv t \bmod P_n$, where $t \in \mathbb{N}$, $1 \leq t < P_n$, and $(t, P_n) \neq 1$. This implies that $(t, P_n) = p_i$ for some $p_i \leq p_n$. Let $t = p_i u$

and $P_n = p_i v$ for some $u, v \in \mathbb{N}$. Since $p = P_n w + t$ for some $w \in \mathbb{N}$, we have that

$$\begin{aligned} p &= P_n w + t \\ &= p_i v w + p_i u \\ &= p_i(vw + u). \end{aligned}$$

Since $(vw + u) \in \mathbb{N}$, let $(vw + u) = x$. Note that $x > 1$. Then

$$\begin{aligned} p &= p_i(vw + u) \\ &= p_i x. \end{aligned}$$

This implies that $p_i | p$, which contradicts the fact that p is prime. □

Proposition 2.5.9 asserts that for $p > p_n$, we have that $p \equiv t \bmod P_n$, where $(t, P_n) = 1$. But there are exactly $\phi(P_n)$ such t's that have this property. Therefore, to find primes $p > p_n$, we need look only at the $\phi(P_n)$ residual classes of the form $t \bmod P_n$, $(t, P_n) = 1$. To formalize this idea, we define one more concept.

Definition 2.5.10. Let $a, b \in \mathbb{N}$. The set of all $a \leq b$ such that $(a, b) = 1$ forms the primitive residual classes modulo b. △

Since we know that for all $p > p_n$, we have $p \equiv t \bmod P_n$, where $(t, P_n) = 1$, then p is congruent to some primitive residual class modulo P_n.

Definition 2.5.11. We write the set of all $t \in \mathbb{N}$ that represents the set of primitive residual classes modulo P_n as

$$\mathcal{T}^{(P_n)} = \{t \in \mathbb{N} | t < P_n, (t, P_n) = 1\}.$$

△

Whenever we refer to the primitive residual classes modulo P_n, we shall use $\mathcal{T}^{(P_n)}$ or $t \in \mathcal{T}^{(P_n)}$. Moreover, we also state that $p \equiv \mathcal{T}^{(P_n)} \bmod P_n$ to signify that p is congruent

to some primitive residual class modulo P_n.

Remark. By the CRT, we found that the system of linear congruences $p \equiv (t_1, \ldots, t_n)$ yields a unique solution t^* mod P_n, where $t^* \in \mathcal{T}^{(P_n)}$, for each of the $\phi(P_n)$ combinations. One question arises: how many primes, if any, are actually congruent to any given $t^* \in \mathcal{T}^{(P_n)}$ modulo P_n? In other words, given any combination of t_i's, can we find a prime such that $p \equiv (t_1, \ldots, t_n)$, where $1 \leq t_i < p_i$? We answer this using a discovery made by Dirichlet in 1837.

When looking at the primitive residual classes modulo P_n, we can see these $\phi(P_n)$ congruences as each being a linear progression. That is, for each $t^* \in \mathcal{T}^{(P_n)}$, we can see t^* mod P_n as representing the linear progression

$$a_k = P_n k + t^*, \qquad k = 0, 1, 2, \ldots \tag{2.5.1}$$

Dirichlet showed that each linear progression of the form in (2.5.1), where $(t^*, P_n) = 1$, contains an equal number of primes. Since primes are infinite, then each progression contains an infinite number of primes [6].

Since an infinite number of primes are congruent t^* mod P_n for each $t^* \in \mathcal{T}^{(P_n)}$, and since each $t^* \in \mathcal{T}^{(P_n)}$ is obtained by using the CRT upon each sequence $(t_1, \ldots, t_n)$, then it follows that there are an infinite number of primes that satisfy each combination of t_i's. That is, we have that the CRT creates a one-to-one correspondence between the $\phi(P_n)$ number of distinct sequences $(t_1, \ldots, t_n)$ and the $\phi(P_n)$ number of distinct $t^* \in \mathcal{T}^{(P_n)}$, which, in the context of Dirichlet's discovery, implies that there exist infinite primes p such that $p \equiv (t_1, \ldots, t_n)$ for each possible sequence. This observation justifies the sifting

of the set of natural numbers congruent $\mathcal{T}^{(P_n)}$ mod P_n: because it is in this set that all the primes greater than p_n lie.

We now concentrate our sifting efforts to the set of natural numbers congruent $\mathcal{T}^{(P_n)}$ mod P_n to find the primes $p > p_n$. This is precisely what the $\mathbb{G}_i$ sieve does.

2.6 $\mathbb{G}_i$ Sieve Terms

Here, we refine the definitions in Section 2.2 to fit the algorithmic needs of the $\mathbb{G}_i$ sieve.

To introduce the sets that the $\mathbb{G}_i$ sieve sifts, we first define the following.

Definition 2.6.1. Let $a, k \in \mathbb{N}$. The set whose elements are all values in $\mathbb{N}$ that are congruent a mod k is defined as

$$\mathcal{G}^{(a,k)} = \{g \in \mathbb{N} | g \equiv a \bmod k\}.$$

We further refine these sets into the j^{th} intervals such that

$$\mathcal{G}_j^{(a,k)} = \{g \in \mathcal{G}^{(a,k)} | p_j^2 \leq g < p_{j+1}^2\}.$$

△

We have already seen an example of a set $\mathcal{G}^{(a,k)}$. Note that every number $n \in \mathbb{N}$ is congruent 0 mod 1. Since the SOE sifts $\mathbb{N}$, and since $\mathcal{G}^{(0,1)} = \mathbb{N}$, then the SOE essentially sifts the set $\mathcal{G}^{(0,1)}$.

Next, we use the properties of primes obtained in Section 2.5 to define the sets the $\mathbb{G}_i$ sieve sifts. These sets are all $\mathcal{G}^{(t,P_n)}$ such that $t \in \mathcal{T}^{(P_n)}$. That is,

$$\mathcal{G}^{(t,P_n)} = \{g \in \mathbb{N} | g \equiv t \bmod P_n\},$$

where $t \in \mathcal{T}^{(P_n)}$. Then the j^{th} interval of these sets are

$$\mathcal{G}_j^{(t,P_n)} = \{g \in \mathcal{G}^{(t,P_n)} | p_j^2 \leq g < p_{j+1}^2\}.$$

Moreover, we denote the union sets $\mathcal{G}^{(P_n)}$ and $\mathcal{G}_j^{(P_n)}$ as

$$\mathcal{G}^{(P_n)} = \bigcup_{t \in \mathcal{T}^{(P_n)}} \mathcal{G}^{(t,P_n)},$$
$$\mathcal{G}_j^{(P_n)} = \bigcup_{t \in \mathcal{T}^{(P_n)}} \mathcal{G}_j^{(t,P_n)}.$$

We will often refer to $\mathcal{G}_j^{(P_n)}$ as the j^{th} interval. However, this informality should not lead to confusion since context will make this reference clear.

Vital to sifting any set of numbers is being able to eliminate the composites in that set. To do this, we simply gather the multiples of relevant primes, then form them into sets.

Definition 2.6.2. We define the set of positive multiples of the j^{th} prime p_j congruent $a \bmod k$ as

$$\mathcal{S}_j^{(a,k)} = \{s \in p_j\mathbb{Z} | s \equiv a \bmod k, s > p_j\}.$$

The elements of $\mathcal{S}_j^{(a,k)}$ are obtained from the following recurrence sequence.

$$s_{j_1}^{(a,k)} = p_j^{(a,k)},$$
$$s_{j_{u+1}}^{(a,k)} = s_{j_u}^{(a,k)} + k \cdot p_j,$$

where $p_j^{(a,k)}$ is the least positive solution greater than p_j to the system of linear congruences

$$p_j^{(a,k)} \equiv 0 \bmod p_j,$$
$$p_j^{(a,k)} \equiv a \bmod k.$$

$\triangle$

We see that $\mathcal{S}_j^{(0,1)}$ represents *all* positive multiples of p_j greater than p_j.

When sifting with the $\mathbb{G}_i$ sieve, we need eliminate only the multiples of any prime p that are congruent $\mathcal{T}^{(P_n)}$ mod P_n. Let p_j be the j^{th} prime. Then we need use only the sets $\mathcal{S}_j^{(t,P_n)}$ for all $t \in \mathcal{T}^{(P_n)}$. That is,

$$S_j^{(t,P_n)} = \{s \in p_j\mathbb{Z} | s \equiv t \bmod P_n, s > p_j\},$$

where the values of $S_j^{(t,P_n)}$ are obtained by the sequence

$$\begin{aligned} s_{j_1}^{(t,P_n)} &= p_j^{(t,P_n)}, \\ s_{j_{k+1}}^{(t,P_n)} &= s_{j_k}^{(t,P_n)} + P_n \cdot p_j, \end{aligned}$$

where $p_j^{(t,P_n)}$ is the least positive solution greater than p_j to the system of linear congruences

$$\begin{aligned} p_j^{(t,P_n)} &\equiv 0 \bmod p_j, \\ p_j^{(t,P_n)} &\equiv t \bmod P_n. \end{aligned}$$

The values $p_j^{(t,P_n)}$ for all $t \in \mathcal{T}^{(P_n)}$ are simple to obtain. Using Corollary 2.5.5 and Theorem 2.5.7, we first find $r, s \in \mathbb{Z}$ such that $rp_j + sP_n = 1$ (we know that r and s exist since $(p_j, P_n) = 1$). Then $p_j^{(t,P_n)} = (trp_j) \bmod p_jP_n$. Since r and p_j do not change, then we can find the values of $p_j^{(t,P_n)}$ for all $t \in \mathcal{T}^{(P_n)}$ simply by plugging in the value t.

2.7 Concepts for Analysis

Analysis of the $\mathbb{G}_i$ sieve in Chapter 4 will have two aspects: the empirical and the theoretical. First, we examine the actual performance of the first seven $\mathbb{G}_i$ sieves, examining the number of primes acquired through run-time, the number of iterations performed, and their efficiency rate (this rate will be made clear in Chapter 4). Afterward, we model the

$\mathbb{G}_i$ sieve's performance for higher i's using estimates of prime and composite density. To do this, we need two concepts: the Prime Number Theorem, and "average" spacing between primes.

Fundamental to the analysis of the $\mathbb{G}_i$ sieve algorithm is the well-known Prime Number Theorem, first proved by Hadamard [5] and de la Vallée-Poussin [9] in 1896, almost a hundred years after Gauss's and Legendre's conjectures. In 1948, Erdös [4] proved the prime number theorem using "elementary" number theory tools.

Theorem 2.7.1 (Prime Number Theorem)**.** *Let $\pi(x)$ denote the number of primes less than or equal to x. Then we have that*

$$\pi(x) \sim \frac{x}{\ln x}$$

since

$$\lim_{x\to\infty} \frac{\pi(x)}{\frac{x}{\ln x}} = 1.$$

The Prime Number Theorem allows us to estimate prime density given a significantly large x. To estimate the prime density of $[p_j^2, p_{j+1}^2)$, we note that there are approximately $\pi(p_j^2)$ primes in $[2, p_j^2)$ and $\pi(p_{j+1}^2)$ in $[2, p_{j+1}^2)$. Then we say that the prime density of $[p_j^2, p_{j+1}^2)$ is $\pi(p_{j+1}^2) - \pi(p_j^2) \approx \frac{p_{j+1}^2}{\ln p_{j+1}^2} - \frac{p_j^2}{\ln p_j^2}$. However, we further estimate this value by making it a function of only one variable: p_j. To do so, we make use of the concept of average spacing between primes. The following progression of manipulations are taken from [11]. Many liberties are taken, such as: 1) probability of divisibility by different primes is considered independent; 2) the sum taken in (2.7.1) is over *all* primes less than x since restricting the sum to primes less than $\sqrt{x}$ will not make much difference.

Proposition 2.7.2. *Let $x \in \mathbb{N}$. Then the average spacing of primes around x is $ln(x)$.*

Proof. The probability that any given integer x is divisible by a prime $p < x$ is $\frac{1}{p}$. Therefore, the probability of x *not* being divisible by p is $(1 - \frac{1}{p})$. Then the probability that x is not divisible by *any* prime $p < x$, which we label $W(x)$, is

$$W(x) \approx \left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right)\left(1 - \frac{1}{5}\right)\cdots \approx \prod_{p_j<x}\left(1 - \frac{1}{p_j}\right). \tag{2.7.1}$$

Taking the natural logarithm of each end of (2.7.1) results in

$$\ln W(x) \approx \sum_{p_j<x} \ln\left(1 - \frac{1}{p_j}\right) \approx -\sum_{p_j<x} \frac{1}{p_j}. \tag{2.7.2}$$

Since a given term $\frac{1}{n}$ in the sum occurs with probability $W(n)$, we can write the last sum in (2.7.2) over all integers, obtaining

$$\ln W(x) \approx -\sum_{n=2}^{x} \frac{W(n)}{n}. \tag{2.7.3}$$

Next, we convert the sum in (2.7.3) into an integral.

$$\ln W(x) \approx -\int_2^x \frac{W(n)}{n} dn. \tag{2.7.4}$$

Letting $A(x) = \frac{1}{W(x)}$, which is the average *spacing* between primes around x, we substitute the terms in (2.7.4) as follows.

$$\begin{aligned} \ln W(x) &\approx -\int_2^x \frac{W(n)}{n} dn \\ &= -\int_2^x \frac{1}{A(x)n} dn. \end{aligned}$$

But $\ln W(x) = -\ln A(x)$, which implies that

$$\ln A(x) \approx \int_2^x \frac{1}{A(x)n} dn. \tag{2.7.5}$$

Differentiating both sides of (2.7.5) gives

$$\begin{aligned} \frac{A'(x)}{A(x)} &\approx \frac{1}{A(x)x}, \\ \Rightarrow A'(x) &\approx \frac{1}{x}. \end{aligned} \tag{2.7.6}$$

Integrating both sides of (2.7.6) yields $A(x) \approx \ln x$, which is the average spacing of primes around x. □

We now have sufficient information for the introduction and analysis of the $\mathbb{G}_i$ sieve.

3
The $\mathbb{G}_i$ Sieves

In Chapter 2, we introduced the Sieve of Eratosthenes in a few ways. First, we introduced Sieve Method 2.1.11, the sieve method used by the SOE. Then, we worked through Example 2.3.1 to see how the SOE actually sifted the primes out of the set $\mathcal{G}_{100} = \{2, \ldots, 100\}$. This example made use of sets to represent the sifting process. After this example came Section 2.3, in which we generalized the sifting process for any ceiling $m \in \mathbb{N}$ through the use of sets. We used the SOE, in conjunction with the Inclusion-Exclusion Principle, to build the counting method in Proposition 2.4.4. Finally, we bounded the prime counting function. In this chapter, we follow a similar progression to introduce the $\mathbb{G}_i$ sieve.

First, we work through four concrete examples using $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_3$, and $\mathbb{G}_4$ to understand how the $\mathbb{G}_i$ sieve's sifting method functions. These examples make use of the sets introduced in Section 2.6 to streamline the sifting process. We then generalize the $\mathbb{G}_i$ sieve's process, first presenting $\mathbb{G}_i$'s sieve method, then going through the abstract process of sifting. After this generalization, we propose a counting method analogous to Proposition 2.4.4 using not only the Inclusion-Exclusion Principle but also the concept of sifting the

"lesser order" primitive residual classes modulo P_n to show efficiency in the number of terms needed to compute $\pi(P_n)$. Then, we conjecture a new bound for $\pi(P_n)$.

In the following four sections, we introduce the first four $\mathbb{G}_i$ sieves. We begin each exploration by understanding what occurs before the sifting process begins. Then, we perform the first few iterations, each iteration being broken down into four steps:

- **Step 1:** Building the set to be sifted, which is the j^{th} interval.
- **Step 2:** Building the set of composites to be eliminated from the j^{th} interval.
- **Step 3:** Performing the set subtraction to obtain the set of primes in the j^{th} interval.
- **Step 4:** Unioning this set with our base set of primes.

These four steps repeat until the interval that contains our ceiling m is reached and sifted.

3.1 $\mathbb{G}_1$

We first obtain the base information needed to begin sifting with the $\mathbb{G}_1$ algorithm. Since $i = 1$, then $n = i - 1 = 0$. We therefore have $P_0 = 1$ (as defined in Definition 2.5.1). We now obtain

$$\mathcal{T}^{(P_0)} = \{t \in \mathbb{N} | t < 1, (t, 1) = 1\} = \{0\}.$$

What values are congruent 0 mod 1? The answer to this is simple: the natural numbers. Since every natural number is congruent 0 mod 1, then $\mathbb{G}_1$ essentially sifts the natural numbers. The $\mathbb{G}_1$'s method of sifting through primitive residual classes is relatively similar the SOE. The only difference is the iterative process of sifting the j^{th} interval. Before sifting begins, we assume the set of primes $\mathcal{P}[2, 3]$.

First Iteration.

Step 1: We begin the sifting process by creating the first interval. Since $i = j = 1$, then $\mathbb{G}_1$ begins by sifting the interval

$$\mathcal{G}_1^{(0,1)} = \{g \in \mathbb{N} | g \equiv 0 \bmod 1, p_1^2 = 4 \leq g < p_2^2 = 9\} = [p_1^2, p_2^2) = [4, 9).$$

Step 2: We now build the sets sufficient to sift $\mathcal{G}_1^{(0,1)}$. By Proposition 2.1.10, we need only eliminate multiples of $p_1 = 2$ from $\mathcal{G}^{(0,1)}$. Therefore, we build the set

$$\mathcal{S}_1^{(0,1)} = \{s \in 2\mathbb{Z} | s \equiv 0 \bmod 1, s > 2\}.$$

Step 3: We now sift $\mathcal{G}_1^{(0,1)}$ by performing the set subtraction $\mathcal{G}_1^{(0,1)} \backslash \mathcal{S}_1^{(0,1)} = \mathcal{P}[4, 8]$.

Step 4: We update our set of primes by unioning $\mathcal{P}[4, 8]$ with our already obtained primes, giving us $\mathcal{P}[1, 8]$. We now sift the next interval.

Second Iteration.

Step 1: We build the second interval

$$\mathcal{G}_2^{(0,1)} = \{g \in \mathbb{N} | g \equiv 0 \bmod 1, p_2^2 = 9 \leq g < p_3^2 = 25\} = [p_2^2, p_3^2) = [9, 24).$$

Step 2: We build the sets sufficient to sift $\mathcal{G}_2^{(0,1)}$. By Proposition 2.1.10, we need only eliminate multiples of the first two primes 2 and 3. We already have $\mathcal{S}_1^{(0,1)}$. We therefore need to build the set

$$\mathcal{S}_2^{(0,1)} = \{s \in 3\mathbb{Z} | s \equiv 0 \bmod 1, s > 3\}.$$

Step 3: We now sift $\mathcal{G}_2^{(0,1)}$ with *both* sets $\mathcal{S}_1^{(0,1)}$ and $\mathcal{S}_2^{(0,1)}$, giving us

$$\mathcal{G}_2^{(0,1)} \backslash (\mathcal{S}_1^{(0,1)} \cup \mathcal{S}_2^{(0,1)}) = \mathcal{P}[9, 24].$$

Step 4: Combining our new set of primes with our old ones, we get $\mathcal{P}[1, 8] \cup \mathcal{P}[9, 24] = \mathcal{P}[1, 24]$.

j^{th} Iteration.

Step 1: We begin the j^{th} iteration by building the j^{th} interval

$$\mathcal{G}_j^{(0,1)} = \{g \in \mathbb{N} | g \equiv 0 \bmod 1, p_j^2 \leq g < p_{j+1}^2\} = [p_j^2, p_{j+1}^2).$$

Step 2: We build the set of multiples of the j^{th} prime,

$$\mathcal{S}_j^{(0,1)} = \{s \in p_j\mathbb{Z} | s \equiv 0 \bmod 1, s > p_j\}.$$

Note that the sets of multiples of the first $j-1$ primes have already been built.

Step 3: We sift $\mathcal{G}_j^{(0,1)}$ by subtracting $\bigcup_{i=1}^{j} \mathcal{S}_i^{(0,1)}$ from it, yielding

$$\mathcal{G}_j^{(0,1)} \setminus \bigcup_{i=1}^{j} \mathcal{S}_i^{(0,1)} = \mathcal{P}[p_j^2, p_{j+1}^2 - 1].$$

Step 4: Unioning this set of primes with our old set, we now have

$$\mathcal{P}[1, p_j^2 - 1] \cup \mathcal{P}[p_j^2, p_{j+1}^2 - 1] = \mathcal{P}[1, p_{j+1}^2 - 1].$$

This iterative process continues until $\mathbb{G}_1$ reaches and sifts the interval that contains the desired ceiling $m \in \mathbb{N}$.

3.2 $\mathbb{G}_2$

We begin the sifting process using the $\mathbb{G}_2$ sieve algorithm by first obtaining the base information needed. We have that $i = 2$. Then $n = 1$, and $P_1 = 2$. Next, we determine $\mathcal{T}^{(P_1)}$, which is

$$\mathcal{T}^{(2)} = \{t \in \mathbb{N} | t < 2, (t, 2) = 1\} = \{1\}.$$

Therefore, $\mathbb{G}_2$ sifts the residual class 1 mod 2. Intuitively, we can see $\mathbb{G}_2$ as sifting through the odd numbers, which essentially means that all multiples of two have been pre-eliminated before the sifting process even begins. This implies that we have no need to use $p_1 = 2$ or the set $\mathcal{S}_1^{(1,2)}$ for sifting.

	0 mod 2	0	**2**	4	6	8	10	12	14	16	18	20	22	24	26	28	30
$\Rightarrow$	1 mod 2	1	**3**	**5**	**7**	9	**11**	**13**	15	**17**	**19**	21	**23**	25	27	**29**	**31**

Figure 3.2.1. Visual Representation of the Number System Sifted by $\mathbb{G}_2$

In Figure 3.2.1, we can visualize the number system that $\mathbb{G}_2$ sifts. The prime numbers greater than 2 lie in the bottom row, the row which holds values congruent to 1 mod 2. This is precisely the row that we are now sifting. Note that since all multiples of $p_1 = 2$ have already been eliminated, then we assume that the interval $\mathcal{G}_1^{(1,2)}$ contains only primes. This is indeed the case since $\mathcal{G}_1^{(1,2)} = \{5, 7\}$. Thus, $\mathbb{G}_2$ begins the sifting process already knowing the set $\mathcal{P}[1, 8]$.

As we did with $\mathbb{G}_1$, we detail the first few iterations of $\mathbb{G}_2$'s execution to understand how it works.

First Iteration.

Step 1: We begin sifting by creating the first interval. Since $i = j = 2$, then $\mathbb{G}_2$ begins by sifting the interval

$$\begin{aligned}\mathcal{G}_2^{(1,2)} &= \{g \in \mathbb{N} | g \equiv 1 \bmod 2, p_2^2 = 9 \leq g < p_3^2 = 25\} \\ &= \{9, 11, 13, 15, 17, 19, 21, 23\}.\end{aligned}$$

Step 2: We now build the sets sufficient to sift $\mathcal{G}_2^{(1,2)}$. By Proposition 2.1.10, we need only eliminate multiples of the primes $p_1 = 2$ and $p_2 = 3$ from $\mathcal{G}_2^{(1,2)}$. However, recall the no multiples of $p_1 = 2$ are congruent 1 mod 2. We therefore need not use $p_1 = 2$ from now on. We need only eliminate the multiples of $p_2 = 3$ to sift $\mathcal{G}_2^{(1,2)}$. We build the set

$$\mathcal{S}_2^{(1,2)} = \{s \in 3\mathbb{Z} | s \equiv 1 \bmod 2, s > 3\}.$$

Step 3: We sift $\mathcal{G}_2^{(1,2)}$ by subtracting $\mathcal{S}_2^{(1,2)}$ from it, obtaining $\mathcal{G}_2^{(1,2)} \backslash \mathcal{S}_2^{(1,2)} = \mathcal{P}[9, 24]$.

Step 4: We add this new set of primes to our already assumed set to obtain $\mathcal{P}[1, 8] \cup \mathcal{P}[9, 24] = \mathcal{P}[1, 24]$.

Second Iteration.

Step 1: We are now ready to sift the next interval, which is

$$\begin{aligned}\mathcal{G}_3^{(1,2)} &= \{g \in \mathbb{N} | g \equiv 1 \bmod 2, p_3^2 = 25 \leq g < p_4^2 = 49\} \\ &= \{25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47\}.\end{aligned}$$

Step 2: We now find the sets sufficient to sift the set $\mathcal{G}_3^{(1,2)}$. We already have $\mathcal{S}_2^{(1,2)}$. We therefore build the set

$$\mathcal{S}_3^{(1,2)} = \{s \in 5\mathbb{Z} | s \equiv 1 \bmod 2, s > 5\}.$$

Step 3: We sift $\mathcal{G}_3^{(1,2)}$ by subtracting $\mathcal{S}_2^{(1,2)}$ and $\mathcal{S}_3^{(1,2)}$ from it, yielding

$$\mathcal{G}_3^{(1,2)} \backslash (\mathcal{S}_2^{(1,2)} \cup \mathcal{S}_3^{(1,2)}) = \mathcal{P}[25, 48].$$

Step 4: Combining our new primes with the old, we get $\mathcal{P}[1, 24] \cup \mathcal{P}[25, 48] = \mathcal{P}[1, 48]$.

$(j-1)^{th}$ **Iteration.**

Step 1: We start the $(j-1)^{th}$ iteration by building the j^{th} interval

$$\mathcal{G}_j^{(1,2)} = \{g \in \mathbb{N} | g \equiv 1 \bmod 2, p_j^2 \leq g < p_{j+1}^2\}.$$

Step 2: We next find the sets sufficient to sift $\mathcal{G}_j^{(1,2)}$. We already have $\bigcup_{i=2}^{j-1} \mathcal{S}_i^{(1,2)}$. We therefore need

$$\mathcal{S}_j^{(1,2)} = \{s \in p_j\mathbb{Z} | s \equiv 1 \bmod 2, s > p_j\}.$$

Step 3: We sift $\mathcal{G}_j^{(1,2)}$ by subtracting $\bigcup_{i=2}^{j} \mathcal{S}_i^{(1,2)}$, obtaining

$$\mathcal{G}_j^{(1,2)} \setminus \bigcup_{i=2}^{j} \mathcal{S}_i^{(1,2)} = \mathcal{P}[p_j^2, p_{j+1}^2 - 1].$$

Step 4: Unioning this new set of primes with the old, we get

$$\mathcal{P}[1, p_j^2 - 1] \cup \mathcal{P}[p_j^2, p_{j+1}^2 - 1] = \mathcal{P}[1, p_{j+1}^2 - 1].$$

Note that the only difference between $\mathbb{G}_1$ and $\mathbb{G}_2$ is the number system that they sift. However, the iteration process is exactly the same.

3.3 $\mathbb{G}_3$

As with the prior two sieves, we begin $\mathbb{G}_3$'s sifting process by first obtaining the base information. Since $i = 3$, then $n = 2$ and $P_2 = 6$. We therefore have

$$\mathcal{T}^{(6)} = \{t \in \mathbb{N} | t < 6, (t, 6) = 1\} = \{1, 5\},$$

which means that $\mathbb{G}_3$ sifts the two primitive residual classes $1, 5 \bmod 6$. The $\mathbb{G}_3$ algorithm seemingly makes the process of sifting through primitive residual classes a bit more complicated.

	0 mod 6	0	6	12	18	24	30	36	42	48	54	60	66	72	78	84
⇒	1 mod 6	1	**7**	**13**	**19**	25	**31**	**37**	**43**	49	55	**61**	**67**	73	**79**	85
	2 mod 6	**2**	8	14	20	26	32	38	44	50	56	62	68	74	80	86
	3 mod 6	**3**	9	15	21	27	33	39	45	51	57	63	69	75	81	87
	4 mod 6	4	10	16	22	28	34	40	46	52	58	64	70	76	82	88
⇒	5 mod 6	**5**	**11**	**17**	**23**	**29**	35	**41**	**47**	**53**	**59**	65	**71**	77	**83**	**89**

Figure 3.3.1. Visual Representation of the Number System Sifted by $\mathbb{G}_3$

By creating a grid with six rows, each row representing a residual class modulo 6, we can see how the primes are distributed (see Figure 3.3.1). Specifically, we note that all primes greater than 3 lie in the two rows representing $1, 5 \bmod 6$. These are the rows that $\mathbb{G}_3$ sifts. Formally, we are unioning the sets $\mathcal{G}^{(1,6)}$ and $\mathcal{G}^{(5,6)}$ to form the set of all numbers congruent $\mathcal{T}^{(6)} \bmod 6$, which is $\mathcal{G}^{(6)}$.

Looking at the visual representation of the six residual classes modulo 6 in Figure 3.3.1, we see that the first two primes, $p_1 = 2$ and $p_2 = 3$, nor any of their multiples are in $\mathcal{G}^{(6)}$. We therefore need not bother with them; rather, we start sifting with the prime $p_3 = 5$. However, we do not need to start sifting with $p_3 = 5$ until we reach the interval $\mathcal{G}_3^{(6)} = \{g \in \mathcal{G}^{(6)} | 25 \leq g < 49\}$. But this means that the first two intervals $\mathcal{G}_1^{(6)}$ and $\mathcal{G}_2^{(6)}$ haven't been sifted, nor do they need to be, since $\mathcal{G}_1^{(6)} \cup \mathcal{G}_2^{(6)} = \mathcal{P}[4, 24]$. By assuming the primes $p_1 = 2$ and $p_2 = 3$ and determining $\mathcal{T}^{(6)}$, we have essentially gained the set of primes $\mathcal{P}[1, 24]$ before beginning the actual sifting process.

First Iteration.

Step 1: As stated already, the first interval that $\mathbb{G}_3$ sifts is $\mathcal{G}_3^{(6)} = \mathcal{G}_3^{(1,6)} \cup \mathcal{G}_3^{(5,6)}$, where

$$\begin{aligned}
\mathcal{G}_3^{(1,6)} &= \{g \in \mathbb{N} | g \equiv 1 \bmod 6, 25 \leq g < 49\} \\
&= \{25, 31, 37, 43\}, \\
\mathcal{G}_3^{(5,6)} &= \{g \in \mathbb{N} | g \equiv 5 \bmod 6, 25 \leq g < 49\} \\
&= \{29, 35, 41, 47\}.
\end{aligned}$$

Step 2: We now determine the set sufficient to sift $\mathcal{G}_3^{(6)}$. We need only use the set $\mathcal{S}_3^{(6)} = \mathcal{S}_3^{(1,6)} \cup \mathcal{S}_3^{(5,6)}$, where

$$\mathcal{S}_3^{(1,6)} = \{s \in p_3\mathbb{Z} | s \equiv 1 \bmod 6, s > 5\},$$
$$\mathcal{S}_3^{(5,6)} = \{s \in p_3\mathbb{Z} | s \equiv 5 \bmod 6, s > 5\}.$$

Step 3: We now perform the set subtraction to obtain the primes in $\mathcal{G}_3^{(6)}$.

$$\mathcal{G}_3^{(6)} \backslash \mathcal{S}_3^{(6)} = \mathcal{P}[25, 48].$$

Step 4: We union these primes with the ones we already have, giving us

$$\mathcal{P}[1, 24] \cup \mathcal{P}[25, 48] = \mathcal{P}[1, 48].$$

Second Iteration.

Step 1: We now sift the next interval, $\mathcal{G}_4^{(6)} = \mathcal{G}_4^{(1,6)} \cup \mathcal{G}_4^{(5,6)}$, where

$$\begin{aligned}\mathcal{G}_4^{(1,6)} &= \{g \in \mathbb{N} | g \equiv 1 \bmod 6, 49 \leq g < 121\} \\ &= \{49, 55, 61, 67, 73, 79, 85, 91, 97, 103, 109, 115\}, \\ \mathcal{G}_4^{(5,6)} &= \{g \in \mathbb{N} | g \equiv 5 \bmod 6, 49 \leq g < 121\} \\ &= \{53, 59, 65, 71, 77, 83, 89, 95, 101, 107, 113, 119\}.\end{aligned}$$

Step 2: We now determine the sets sufficient to sift $\mathcal{G}_4^{(6)}$. We need only use the primes $p_3 = 5$ and $p_4 = 7$. Since we already have the set $\mathcal{S}_3^{(6)}$, we need only build the set $\mathcal{S}_4^{(6)} = \mathcal{S}_4^{(1,6)} \cup \mathcal{S}_4^{(5,6)}$, where

$$\mathcal{S}_4^{(1,6)} = \{s \in p_4\mathbb{Z} | s \equiv 1 \bmod 6, s > 7\},$$
$$\mathcal{S}_4^{(5,6)} = \{s \in p_4\mathbb{Z} | s \equiv 5 \bmod 6, s > 7\}.$$

Step 3: We are now ready to sift the set $\mathcal{G}_4^{(6)}$. We now have

$$\mathcal{G}_4^{(6)} \backslash (\mathcal{S}_3^{(6)} \cup \mathcal{S}_4^{(6)}) = \mathcal{P}[49, 120].$$

Step 4: Bringing all of our primes together gives us

$$\mathcal{P}[1,48] \cup \mathcal{P}[49,120] = \mathcal{P}[1,120]$$

$(j-2)^{th}$ Iteration.

Step 1: In the $(j-2)^{th}$ iteration, we are sifting the set $\mathcal{G}_j^{(6)} = \mathcal{G}_j^{(1,6)} \cup \mathcal{G}_j^{(5,6)}$, where

$$\mathcal{G}_j^{(1,6)} = \{g \in \mathbb{N} | g \equiv 1 \bmod 6, p_j^2 \leq g < p_{j+1}^2\},$$
$$\mathcal{G}_j^{(5,6)} = \{g \in \mathbb{N} | g \equiv 5 \bmod 6, p_j^2 \leq g < p_{j+1}^2\}.$$

Step 2: We now build the set whose elements are the multiples of the j^{th} prime that are congruent $\mathcal{T}^{(6)}$ mod 6. This set is $\mathcal{S}_j^{(6)} = \mathcal{S}_j^{(1,6)} \cup \mathcal{S}_j^{(5,6)}$, where

$$\mathcal{S}_j^{(1,6)} = \{s \in p_j\mathbb{Z} | s \equiv 1 \bmod 6, s > p_j\},$$
$$\mathcal{S}_j^{(5,6)} = \{s \in p_j\mathbb{Z} | s \equiv 5 \bmod 6, s > p_j\}.$$

Step 3: We now sift $\mathcal{G}_j^{(6)}$, getting

$$\mathcal{G}_j^{(6)} \setminus \bigcup_{i=3}^{j} S_j^{(6)} = \mathcal{P}[p_j^2, p_{j+1}^2 - 1].$$

Step 4: Combining our primes, we get

$$\mathcal{P}[1, p_j^2 - 1] \cup \mathcal{P}[p_j^2, p_{j+1}^2 - 1] = \mathcal{P}[1, p_{j+1}^2 - 1].$$

3.4 $\mathbb{G}_4$

Without going into too detailed an account of $\mathbb{G}_4$'s execution, it will suffice to show visually the inspiration behind sifting $\mathcal{T}^{(30)}$ mod 30 (recall that $\mathbb{G}_4$ sifts the primitive residual classes modulo $P_3 = 30$, where $\mathcal{T}^{(30)} = \{1, 7, 11, 13, 17, 19, 23, 29\}$).

	0 mod 30	0	30	60	90	120	150	180	210	240	270
⇒	1 mod 30	1	**31**	**61**	91	121	**151**	**181**	**211**	**241**	**271**
	2 mod 30	**2**	32	62	92	122	152	182	212	242	272
	3 mod 30	**3**	33	63	93	123	153	183	213	243	273
	4 mod 30	4	34	64	94	124	154	184	214	244	274
	5 mod 30	**5**	35	65	95	125	155	185	215	245	275
	6 mod 30	6	36	66	96	126	156	186	216	246	276
⇒	7 mod 30	**7**	**37**	**67**	**97**	**127**	**157**	187	217	247	**277**
	8 mod 30	8	38	68	98	128	158	188	218	248	278
	9 mod 30	9	39	69	99	129	159	189	219	249	279
	10 mod 30	10	40	70	100	130	160	190	220	250	280
⇒	11 mod 30	**11**	**41**	**71**	**101**	**131**	161	**191**	221	**251**	**281**
	12 mod 30	12	42	72	102	132	162	192	222	252	282
⇒	13 mod 30	**13**	**43**	**73**	**103**	133	**163**	**193**	**223**	253	**283**
	14 mod 30	14	44	74	104	134	164	194	224	254	284
	15 mod 30	15	45	75	105	135	165	195	225	255	285
	16 mod 30	16	46	76	106	136	166	196	226	256	286
⇒	17 mod 30	**17**	**47**	77	**107**	**137**	**167**	**197**	**227**	**257**	287
	18 mod 30	18	48	78	108	138	168	198	228	258	288
⇒	19 mod 30	**19**	49	**79**	**109**	**139**	169	**199**	**229**	259	289
	20 mod 30	20	50	80	110	140	170	200	230	260	290
	21 mod 30	21	51	81	111	141	171	201	231	261	291
	22 mod 30	22	52	82	112	142	172	202	232	262	292
⇒	23 mod 30	**23**	**53**	**83**	**113**	143	**173**	203	**233**	**263**	**293**
	24 mod 30	24	54	84	114	144	174	204	234	264	294
	25 mod 30	25	55	85	115	145	175	205	235	265	295
	26 mod 30	26	56	86	116	146	176	206	236	266	296
	27 mod 30	27	57	87	117	147	177	207	237	267	297
	28 mod 30	28	58	88	118	148	178	208	238	268	298
⇒	29 mod 30	**29**	**59**	**89**	119	**149**	**179**	209	**239**	**269**	299

Figure 3.4.1. Visual Representation of the Number System Sifted by $\mathbb{G}_4$.

Figure 3.4.1 shows in which residual classes primes greater than 5 lie. Using $\mathbb{G}_4$, we are only concerned with $\phi(30) = 8$ residual classes, which dramatically decreases the number of composites needed to be eliminated.

3.5 The $\mathbb{G}_i$ Sieve Method

The $\mathbb{G}_i$ sieve uses the following sieve method to sift the primes out of the set $\mathcal{G}^{(P_n)} = \bigcup_{t \in \mathcal{T}^{(P_n)}} \mathcal{G}^{(t,P_n)}$.

Sieve Method 3.5.1 (The $\mathbb{G}_i$ Sieve)**.**

Input: The set of the first n primes $\{p_1, \cdots, p_n\}$,

Input: Some $m \in \mathbb{N}$.

Output: Set of primes up to the least prime squared greater than m.

The $\mathbb{G}_i$ sieve requires the first n primes in order to determine the modulo whose primitive residual classes it sifts. It then begins to sift through the values in the j^{th} interval congruent to these classes until it reaches the interval $[p_j^2, p_{j+1}^2)$ containing m, after which $\mathbb{G}_i$ outputs the set of primes up to p_{j+1}^2.

Algorithm:‡[1]
$\mathcal{I} = \{p_1, \cdots, p_n\}$ (Set of inputed primes.)
$m =$ Ceiling of Sifting Process
$P_n = p_1 \cdots p_n$
$\mathcal{T}^{(P_n)} = \{t | t < P_n, (t, P_n) = 1\}$
$\mathcal{P} = \mathcal{I} \cup \{p \in \mathcal{T}^{(P_n)} | p < p_i^2\}$‡[2]
$\mathcal{S} = \{\underbrace{\mathcal{I}, \mathcal{I}, \ldots}_{\phi(P_n) \text{ sets}}\}$‡[3]
$i = \#\mathcal{I} + 1$
$j = i$
$A = \{a \in \mathbb{Z} | 1 \leq a \leq \phi(P_n)\}$
while $\mathcal{P}(j) \leq \sqrt{m}$:
  for t in $\mathcal{T}^{(P_n)}$:
    $\mathcal{G}_j^{(t,P_n)} = \{g \in \mathbb{N} | g \equiv t \bmod P_n, p_j \leq g < p_{j+1}^2\}$
  $\mathcal{G}_j^{(P_n)} = \bigcup_{t \in \mathcal{T}^{(P_n)}} \mathcal{G}_j^{(t,P_n)}$

[1] A quick note on the syntax of this pseudo-code. Indentation indicates a "Do" command. Nested blocks, or rather nested Do's, are thus indented multiple times. This pseudo-code is python language based.

[2] If using $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_3$, or $\mathbb{G}_4$, the set $\mathcal{P}$ must be inputed manually since the set $\mathcal{T}$ will not contain all primes up to p_i^2.

[3] The elements of $\mathcal{S}$ will themselves be sets that contain the "current" multiples of relevant primes that are congruent $\mathcal{T}^{(P_n)}$ mod P_n. Each element represents the multiples of each prime congruent to a primitive residual class. S begins with $\phi(P_n)$ sets to represent these classes. We place $\mathcal{I}$'s in S so that the index of the element $S(i)(j)$ will correspond to the index of the prime p_j in $\mathcal{P}$. For example, the element $S(1)(4)$ will be the "current" multiple of $p_4 = 7$ congruent to the first primitive residual class modulo P_n, which is 1 mod P_n.

$B = \{b \in \mathbb{Z} | i \leq b \leq j\}$
Run Euclidean Algorithm to find $r, s \in \mathbb{Z}$ s.t. $rp_j + sP_n = 1$
for a in A:
  $\mathcal{S}(a) = \mathcal{S}(a) \cup \{p_j^{\mathcal{T}^{(P_n)}(a),P_n)}\}$‡[4], where $p_j^{(t,P_n)} = trp_j \bmod p_j P_n$
for $a \in A$:
  for $b \in B$:
    if $S(a)(b) \in \mathcal{G}_j^{(P_n)}$:
      $\mathcal{G}_j^{(P_n)} = \mathcal{G}_j^{(P_n)} \backslash \{\mathcal{S}(a)(b)\}$
      $\mathcal{S}(a)(b) = \mathcal{S}(a)(b) + \mathcal{P}(a) \cdot P_n$
    else:
      $\mathcal{S}(a)(b) = \mathcal{S}(a)(b) + \mathcal{P}(a) \cdot P_n$
$\mathcal{P}[p_j^2, p_{j+1}^2] = \mathcal{G}_j^{(P_n)}$
$\mathcal{P} = \mathcal{P} \cup \mathcal{P}[p_j^2, p_{j+1}^2]$
$j = j + 1$

3.6 $\mathbb{G}_i$

We now introduce the abstraction of the $\mathbb{G}_i$ sieve. We restate the sets mentioned in Section 2.6 for clarity's sake. Recall that $\mathbb{G}_i$ does not sift straight up to a ceiling $m \in \mathbb{N}$. Rather, it sifts through intervals defined by the squares of consecutive primes. Because we are sifting through the primitive residual classes modulo P_n, we must create the set whose values are congruent to these classes. This set will be the counterpart to the SOE's $\mathcal{G}_m$. We have that $\mathbb{G}_i$ sifts the sets

$$\mathcal{G}^{(t,P_n)} = \{g \in \mathbb{N} | g \equiv t \bmod P_n, t \in \mathcal{T}^{(P_n)}\},$$

for all $t \in \mathcal{T}^{(P_n)}$. Since $\mathbb{G}_i$ works in iterations based upon the squares of the j^{th} and $(j+1)^{th}$ primes, we refine these sets into j^{th} intervals such that

$$\mathcal{G}_j^{(t,P_n)} = \{g \in \mathcal{G}^{(t,P_n)} | p_j^2 \leq g < p_{j+1}^2\},$$

where $j \geq i$. The reason why $j \geq i$ is that, when working modulo P_n, the multiples of the first $n = i - 1$ primes are already eliminated and don't need to be sifted out, which

[4]Here, we add a new element to each set in $\mathcal{S}$ to hold the current multiples of the current j^{th} prime that are congruent to $\mathcal{T}^{(P_n)}$ mod P_n. It is to be understood that when an element is added to a set, it is placed as the last element in the set, and its "place" never changes. Thus, the a^{th} element will always be the a^{th} element, regardless of the subsequent changes to the value of this element.

implies that $\bigcup_{j=1}^{n} \mathcal{G}_j^{(P_n)} = \mathcal{P}[p_1^2, p_{n+1}^2 - 1] = \mathcal{P}[p_1^2, p_i^2 - 1]$ is known before sifting begins.[5]

Sifting the set $\mathcal{G}_j^{(P_n)} = \bigcup_{t \in \mathcal{T}^{(P_n)}} \mathcal{G}_j^{(t,P_n)}$ requires the elimination of all multiples of *relevant* primes. Recalling that the set $\mathcal{G}_j^{(P_n)}$ goes up to p_{j+1}^2 (not inclusive), then by Proposition 2.1.10, we need only use the primes $p_i, \ldots, p_j$ to sift $\mathcal{G}_j^{(P_n)}$. However, we must build the set of multiples of all primes $p \in \{p_i, \ldots, p_j\}$ that represent p's multiples congruent $\mathcal{T}^{(P_n)}$ mod P_n. Since there are $\phi(P_n)$ primitive residual classes, we assign $\phi(P_n)$ sets to all relevant primes. These sets are of the form $\mathcal{S}_j^{(t,P_n)} = \{s \in p_j\mathbb{Z} | s \equiv t \bmod P_n, s > p_j\}$ for all $t \in \mathcal{T}^{(P_n)}$.

Theorem 3.6.1. *Let $\mathcal{S}_j^{(P_n)} = \bigcup_{t \in \mathcal{T}^{(P_n)}} \mathcal{S}_j^{(t,P_n)}$ be the set whose values are the multiples of the j^{th} prime p_j congruent $\mathcal{T}^{(P_n)}$ mod P_n. Also, let $\mathcal{G}_j^{(P_n)} = \bigcup_{t \in \mathcal{T}^{(P_n)}} \mathcal{G}_j^{(t,P_n)}$. Then sifting $\mathcal{G}_j^{(P_n)}$ with the set $\bigcup_{k=i}^{j} \mathcal{S}_k^{(P_n)}$ results in the set $\mathcal{P}[p_j^2, p_{j+1}^2 - 1]$. That is,*

$$\mathcal{G}_j^{(P_n)} \setminus \bigcup_{k=i}^{j} \mathcal{S}_k^{(P_n)} = \mathcal{P}[p_j^2, p_{j+1}^2 - 1].$$

Proof. We must sift $\mathcal{G}_j^{(P_n)}$, which consists only of values in $\mathbb{N}$ congruent $\mathcal{T}^{(P_n)}$ mod P_n. Then we need only eliminate the composites congruent to these residual classes. These composites are multiples of primes greater than the n^{th} prime since multiples of primes less than or equal to p_n are not congruent $\mathcal{T}^{(P_n)}$ mod P_n. We therefore need only use primes greater than p_n. By Proposition 2.1.10, these are the primes up to p_j. Therefore, we have that $\mathcal{G}_j^{(P_n)} \setminus \bigcup_{k=i}^{j} \mathcal{S}_k^{(P_n)} = \mathcal{P}[p_j^2, p_{j+1}^2 - 1]$. □

After every interation, we obtain the set $\mathcal{P}[p_j^2, p_{j+1}^2 - 1]$. As noted earlier, $\mathbb{G}_i$ assumes the set $\mathcal{P}[1, p_i^2 - 1]$ before sifting begins. After every iteration, $\mathbb{G}_i$ adds the set $\mathcal{P}[p_j^2, p_{j+1}^2 - 1]$

[5] Algorithmically, for $i \geq 5$, $\mathbb{G}_i$ can acquire the set $\mathcal{P}[1, p_i^2 - 1]$ simply by taking the values in $\mathcal{T}^{(P_n)}$ less than p_i^2. See footnote 2 on Page 58.

to the set of already acquired primes to form the set of all primes up to p_k^2 for some large $k \in \mathbb{N}$. That is, $\mathcal{P}[1, p_i^2 - 1] \cup \bigcup_{j=i}^{k-1} \mathcal{P}[p_j^2, p_{j+1}^2 - 1] = \mathcal{P}[1, p_k^2 - 1]$.

3.7 Counting Primes using $\mathbb{G}_i$

How does the concept of sifting primitive residual classes modulo P_n help us estimate prime density? We proceed as we did in Section 2.4 with a concrete example of counting. In this example, we count the number of primes less than or equal to $P_4 = 210$. (From now on, we will restrict our ceilings to be the product over the first n primes, with $n \geq 4$.)

Example 3.7.1. We want to find the value $\pi(P_4) = \pi(210)$. Since $\mathcal{P}[1, \sqrt{210}\,] = \{2, 3, 5, 7, 11, 13\}$, then we concern ourselves with ensuring that the positive multiples of the six primes in $\mathcal{P}[1, \sqrt{210}\,]$ are eliminated from the set $\mathcal{G}_{210}$. Let

$$A_1 = \{a \in 2\mathbb{Z} | 1 < a \leq 210\},$$

$$A_2 = \{a \in 3\mathbb{Z} | 1 < a \leq 210\},$$

$$A_3 = \{a \in 5\mathbb{Z} | 1 < a \leq 210\},$$

$$A_4 = \{a \in 7\mathbb{Z} | 1 < a \leq 210\},$$

$$A_5 = \{a \in 11\mathbb{Z} | 1 < a \leq 210\}, \text{ and}$$

$$A_6 = \{a \in 13\mathbb{Z} | 1 < a \leq 210\}.$$

The set A_i represents the multiples of p_i in the set $\mathcal{G}_{210}$, where, recalling from Section 2.4,

$$\begin{aligned}
\#A_i &= \left\lfloor \frac{210}{p_i} \right\rfloor, \quad 1 \le i \le 6, \\
\#(A_i \cap A_j) &= \left\lfloor \frac{210}{p_i \cdot p_j} \right\rfloor, \quad 1 \le i < j \le 6, \\
\#(A_i \cap A_j \cap A_k) &= \left\lfloor \frac{210}{p_i \cdot p_j \cdot p_k} \right\rfloor, \quad 1 \le i < j < k \le 6, \\
\#(A_i \cap A_j \cap A_k \cap A_l) &= \left\lfloor \frac{210}{p_i \cdot p_j \cdot p_k \cdot p_l} \right\rfloor, \quad 1 \le i < j < k < l \le 6, \\
\#(A_i \cap A_j \cap A_k \cap A_l \cap A_m) &= \left\lfloor \frac{210}{p_i \cdot p_j \cdot p_k \cdot p_l \cdot p_m} \right\rfloor, \quad 1 \le i < j < k < l < m \le 6, \\
\#(A_i \cap A_j \cap A_k \cap A_l \cap A_m \cap A_n) &= \left\lfloor \frac{210}{p_i \cdot p_j \cdot p_k \cdot p_l \cdot p_m \cdot p_n} \right\rfloor, \quad 1 \le i < j < k < l < m < n \le 6.
\end{aligned}$$

We must find $\#(A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5 \cup A_6)$. By Theorem 2.4.1, we have that

$$\begin{aligned}
\#(A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5 \cup A_6) &= \#(A_1) + \#(A_2) + \#(A_3) + \#(A_4) + \#(A_5) + \#(A_6) \\
&\quad - \#(A_1 \cap A_2) - \#(A_1 \cap A_3) - \#(A_1 \cap A_4) - \#(A_1 \cap A_5) - \#(A_1 \cap A_6) \\
&\quad - \#(A_2 \cap A_3) - \#(A_2 \cap A_4) - \#(A_2 \cap A_5) - \#(A_2 \cap A_6) - \#(A_3 \cap A_4) \\
&\quad - \#(A_3 \cap A_5) - \#(A_3 \cap A_6) - \#(A_4 \cap A_5) - \#(A_4 \cap A_6) - \#(A_5 \cap A_6) \\
&\quad + \#(A_1 \cap A_2 \cap A_3) + \#(A_1 \cap A_2 \cap A_4) + \#(A_1 \cap A_2 \cap A_5) + \#(A_1 \cap A_2 \cap A_6) \\
&\quad + \#(A_1 \cap A_3 \cap A_4) + \#(A_1 \cap A_3 \cap A_5) + \#(A_1 \cap A_3 \cap A_6) + \#(A_1 \cap A_4 \cap A_5) \\
&\quad + \#(A_1 \cap A_4 \cap A_6) + \#(A_1 \cap A_5 \cap A_6) + \#(A_2 \cap A_3 \cap A_4) + \#(A_2 \cap A_3 \cap A_5) \\
&\quad + \#(A_2 \cap A_3 \cap A_6) + \#(A_2 \cap A_4 \cap A_5) + \#(A_2 \cap A_4 \cap A_6) + \#(A_2 \cap A_5 \cap A_6)
\end{aligned}$$

$$
\begin{aligned}
&+ \#(A_3 \cap A_4 \cap A_5) + \#(A_3 \cap A_4 \cap A_6) + \#(A_3 \cap A_5 \cap A_6) + \#(A_4 \cap A_5 \cap A_6) \\
&- \#(A_1 \cap A_2 \cap A_3 \cap A_4) - \#(A_1 \cap A_2 \cap A_3 \cap A_5) - \#(A_1 \cap A_2 \cap A_3 \cap A_6) \\
&- \#(A_1 \cap A_2 \cap A_4 \cap A_5) - \#(A_1 \cap A_2 \cap A_4 \cap A_6) - \#(A_1 \cap A_2 \cap A_5 \cap A_6) \\
&- \#(A_1 \cap A_3 \cap A_4 \cap A_5) - \#(A_1 \cap A_3 \cap A_4 \cap A_6) - \#(A_1 \cap A_3 \cap A_5 \cap A_6) \\
&- \#(A_1 \cap A_4 \cap A_5 \cap A_6) - \#(A_2 \cap A_3 \cap A_4 \cap A_5) - \#(A_2 \cap A_3 \cap A_4 \cap A_6) \\
&- \#(A_2 \cap A_3 \cap A_5 \cap A_6) - \#(A_2 \cap A_4 \cap A_5 \cap A_6) - \#(A_3 \cap A_4 \cap A_5 \cap A_6) \\
&+ \#(A_1 \cap A_2 \cap A_3 \cap A_4 \cap A_5) + \#(A_1 \cap A_2 \cap A_3 \cap A_4 \cap A_6) \\
&+ \#(A_1 \cap A_2 \cap A_3 \cap A_5 \cap A_6) + \#(A_1 \cap A_2 \cap A_4 \cap A_5 \cap A_6) \\
&+ \#(A_1 \cap A_3 \cap A_4 \cap A_5 \cap A_6) + \#(A_2 \cap A_3 \cap A_4 \cap A_5 \cap A_6) \\
&- \#(A_1 \cap A_2 \cap A_3 \cap A_4 \cap A_5 \cap A_6). \qquad (3.7.1)
\end{aligned}
$$

Since (3.7.1) represents the number of composites in the set $\mathcal{G}_{210}$, then we must subtract this value from $\#\mathcal{G}_{210}$, which is $\#\mathcal{G}_{210} = 210 - 1$. However, (3.7.1) also counts the 6 primes whose multiples are the composites being eliminated. We therefore must add $6 = \mathcal{P}[1, \sqrt{210}\,]$ back this value. We thus have that

$$\pi(210) = 210 - \#(A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5 \cup A_6) + \mathcal{P}[1, \sqrt{210}\,] - 1.$$

Obviously (3.7.1) is extremely tedious to evaluate. Let us concentrate only on the terms in (3.7.1) that include A_i for $1 \leq i \leq 4$ and gather them together. Let

$$
\begin{aligned}
B =& \#(A_1) + \#(A_2) + \#(A_3) + \#(A_4) - \#(A_1 \cap A_2) - \#(A_1 \cap A_3) - \#(A_1 \cap A_4) \\
&- \#(A_2 \cap A_3) - \#(A_2 \cap A_4) - \#(A_3 \cap A_4) + \#(A_1 \cap A_2 \cap A_3) + \#(A_1 \cap A_2 \cap A_4) \\
&+ \#(A_1 \cap A_3 \cap A_4) + \#(A_2 \cap A_3 \cap A_4) - \#(A_1 \cap A_2 \cap A_3 \cap A_4).
\end{aligned}
$$

Then

$$B = \left\lfloor \frac{210}{2} \right\rfloor + \left\lfloor \frac{210}{3} \right\rfloor + \left\lfloor \frac{210}{5} \right\rfloor + \left\lfloor \frac{210}{7} \right\rfloor - \left\lfloor \frac{210}{2 \cdot 3} \right\rfloor - \left\lfloor \frac{210}{2 \cdot 5} \right\rfloor - \left\lfloor \frac{210}{2 \cdot 7} \right\rfloor - \left\lfloor \frac{210}{3 \cdot 5} \right\rfloor - \left\lfloor \frac{210}{3 \cdot 7} \right\rfloor - \left\lfloor \frac{210}{5 \cdot 7} \right\rfloor + \left\lfloor \frac{210}{2 \cdot 3 \cdot 5} \right\rfloor + \left\lfloor \frac{210}{2 \cdot 3 \cdot 7} \right\rfloor + \left\lfloor \frac{210}{2 \cdot 5 \cdot 7} \right\rfloor + \left\lfloor \frac{210}{3 \cdot 5 \cdot 7} \right\rfloor - \left\lfloor \frac{210}{2 \cdot 3 \cdot 5 \cdot 7} \right\rfloor.$$

Let $B^* = 210 - B = \left\lfloor \frac{210}{1} \right\rfloor - B$. We notice that the denominators in the terms in B^* are the divisors of 210. Then, using (2.4.4), we get

$$\begin{aligned} B^* &= \sum_{d|210} \mu(d) \frac{210}{d} \\ &= 210 \sum_{d|210} \mu(d) \frac{1}{d} \\ &= \phi(210). \end{aligned}$$

Therefore, we have simplified 16 terms out of the 66 terms required to evaluate $\#(A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5 \cup A_6)$. (To evaluate $\pi(210)$, we see that the intersection of the six sets contributes 63 terms. We also have the three terms "210", "$\mathcal{P}[1, \sqrt{210}\,]$", and "$-1$".) By evaluating $\phi(210)$, we essentially eliminate the positive multiples of the first four primes from 210; that is, we count the number of elements in $\mathcal{G}_{210}$ that are relatively prime to $P_4 = 210$, leaving behind the set of numbers in $\mathcal{T}^{(P_4)}$ that are either primes, multiples of 11, and/or multiples of 13. To finish the count $\pi(210)$, we evaluate the rest of the terms

in (3.7.1).

$$
\begin{aligned}
\pi(210) =& \phi(210) - \left\lfloor \frac{210}{11} \right\rfloor - \left\lfloor \frac{210}{13} \right\rfloor + \left\lfloor \frac{210}{11 \cdot 13} \right\rfloor \\
&+ \left\lfloor \frac{210}{11 \cdot 2} \right\rfloor + \left\lfloor \frac{210}{11 \cdot 3} \right\rfloor + \left\lfloor \frac{210}{11 \cdot 5} \right\rfloor + \left\lfloor \frac{210}{11 \cdot 7} \right\rfloor \\
&- \left\lfloor \frac{210}{11 \cdot 2 \cdot 3} \right\rfloor - \left\lfloor \frac{210}{11 \cdot 2 \cdot 5} \right\rfloor - \left\lfloor \frac{210}{11 \cdot 2 \cdot 7} \right\rfloor - \dots \\
&+ \left\lfloor \frac{210}{11 \cdot 2 \cdot 3 \cdot 5} \right\rfloor + \left\lfloor \frac{210}{11 \cdot 2 \cdot 3 \cdot 7} \right\rfloor + \left\lfloor \frac{210}{11 \cdot 2 \cdot 5 \cdot 7} \right\rfloor + \left\lfloor \frac{210}{11 \cdot 3 \cdot 5 \cdot 7} \right\rfloor \\
&- \left\lfloor \frac{210}{11 \cdot 2 \cdot 3 \cdot 5 \cdot 7} \right\rfloor \\
&+ \left\lfloor \frac{210}{13 \cdot 2} \right\rfloor + \left\lfloor \frac{210}{13 \cdot 3} \right\rfloor + \left\lfloor \frac{210}{13 \cdot 5} \right\rfloor + \left\lfloor \frac{210}{13 \cdot 7} \right\rfloor \\
&- \left\lfloor \frac{210}{13 \cdot 2 \cdot 3} \right\rfloor - \left\lfloor \frac{210}{13 \cdot 2 \cdot 5} \right\rfloor - \left\lfloor \frac{210}{13 \cdot 2 \cdot 7} \right\rfloor - \dots \\
&+ \left\lfloor \frac{210}{13 \cdot 2 \cdot 3 \cdot 5} \right\rfloor + \left\lfloor \frac{210}{13 \cdot 2 \cdot 3 \cdot 7} \right\rfloor + \left\lfloor \frac{210}{13 \cdot 2 \cdot 5 \cdot 7} \right\rfloor + \left\lfloor \frac{210}{13 \cdot 3 \cdot 5 \cdot 7} \right\rfloor \\
&- \left\lfloor \frac{210}{13 \cdot 2 \cdot 3 \cdot 5 \cdot 7} \right\rfloor \\
&- \left\lfloor \frac{210}{11 \cdot 13 \cdot 2} \right\rfloor - \left\lfloor \frac{210}{11 \cdot 13 \cdot 3} \right\rfloor - \left\lfloor \frac{210}{11 \cdot 13 \cdot 5} \right\rfloor - \left\lfloor \frac{210}{11 \cdot 13 \cdot 7} \right\rfloor \\
&+ \left\lfloor \frac{210}{11 \cdot 13 \cdot 2 \cdot 3} \right\rfloor + \left\lfloor \frac{210}{11 \cdot 13 \cdot 2 \cdot 5} \right\rfloor + \left\lfloor \frac{210}{11 \cdot 13 \cdot 2 \cdot 7} \right\rfloor + \dots \\
&- \left\lfloor \frac{210}{11 \cdot 13 \cdot 2 \cdot 3 \cdot 5} \right\rfloor - \left\lfloor \frac{210}{11 \cdot 13 \cdot 2 \cdot 3 \cdot 7} \right\rfloor - \left\lfloor \frac{210}{11 \cdot 13 \cdot 2 \cdot 5 \cdot 7} \right\rfloor - \left\lfloor \frac{210}{11 \cdot 13 \cdot 3 \cdot 5 \cdot 7} \right\rfloor \\
&+ \left\lfloor \frac{210}{11 \cdot 13 \cdot 2 \cdot 3 \cdot 5 \cdot 7} \right\rfloor + 6 - 1 \\
&= 48 - 19 + 9 + 6 + 3 + 2 - 3 - 1 - 1 - 1 - 0 - 0 + 0 + 0 + 0 + 0 - 0 \\
&- 16 + 8 + 5 + 3 + 2 - 2 - 1 - 1 - 1 - 0 - 0 + 0 + 0 + 0 + 0 - 0 \\
&+ 1 - 0 - 0 - 0 - 0 + 0 + 0 + 0 + 0 + 0 + 0 - 0 - 0 - 0 - 0 + 0 + 6 - 1 \\
&= 46.
\end{aligned}
$$

Thus, $\pi(210) = 46$. We can also express $\pi(210)$ using summation notation. Let $C = \{11, 13\}$. Then $Y^*(C)$ is the power set of C minus the empty set, where y^* is an element

of Y^*. Then

$$\pi(210) = \phi(210) + \mu(210) \sum_{d|210} \sum_{y^* \in Y^*(C)} \left((-1)^{\#y^*} \mu(d) \left\lfloor \frac{d}{\prod_{p \in y^*} p} \right\rfloor \right) + \#\mathcal{P}[1, \sqrt{210}] - 1. \tag{3.7.2}$$

We generalize (3.7.2) for any ceiling P_n.

Proposition 3.7.2. *Let Y^* be the power set of $\mathcal{P}[p_{n+1}, \sqrt{P_n}\,]$ minus the empty set, where y^* is an element of Y^*. Then*

$$\pi(P_n) = \phi(P_n) + \mu(P_n) \sum_{d|P_n} \sum_{y^* \in Y^*} \left((-1)^{\#y^*} \mu(d) \left\lfloor \frac{d}{\prod_{p \in y^*} p} \right\rfloor \right) + \#\mathcal{P}[1, \sqrt{P_n}] - 1. \tag{3.7.3}$$

Proof. Recall that in order to sift up to P_n, we need use the primes in $\mathcal{P}[1, \sqrt{P_n}\,]$. Let $A_i = \{a \in p_i\mathbb{Z} | 1 < a \leq P_n\}$ for $1 \leq i \leq \#\mathcal{P}[1, \sqrt{P_n}\,]$. To begin evaluating $\pi(P_n)$, we must find $\#(A_1 \cup A_2 \cup \cdots \cup A_{\#\mathcal{P}[1,\sqrt{P_n}\,]})$ since $\pi(P_n) = P_n - \#(A_1 \cap A_2 \cap \cdots \cap A_{\#\mathcal{P}[1,\sqrt{P_n}\,]}) + \#\mathcal{P}[1, \sqrt{P_n}\,] - 1$. By Theorem 2.4.1, we have that

$$\begin{aligned} \#(A_1 \cup A_2 \cup \cdots \cup A_{\#\mathcal{P}[1,\sqrt{P_n}\,]}) = & \sum_{1 \leq i \leq \#\mathcal{P}[1,\sqrt{P_n}\,]} \#A_i - \sum_{1 \leq i < j \leq \#\mathcal{P}[1,\sqrt{P_n}\,]} \#(A_i \cap A_j) \\ & + \sum_{1 \leq i < j < k \leq \#\mathcal{P}[1,\sqrt{P_n}\,]} \#(A_i \cap A_j \cap A_k) - \cdots \\ & + (-1)^{\#\mathcal{P}[1,\sqrt{P_n}\,]+1} \cdot \#(A_1 \cap A_2 \cap \cdots \cap A_{\#\mathcal{P}[1,\sqrt{P_n}\,]}), \end{aligned} \tag{3.7.4}$$

where

$$\begin{aligned} \#A_i &= \left\lfloor \frac{P_n}{p_i} \right\rfloor, \quad 1 \leq i \leq \#\mathcal{P}[1, \sqrt{P_n}\,], \\ \#(A_i \cap A_j) &= \left\lfloor \frac{P_n}{p_i \cdot p_j} \right\rfloor, \quad 1 \leq i \leq \#\mathcal{P}[1, \sqrt{P_n}\,], \\ \#(A_i \cap A_j \cap A_k) &= \left\lfloor \frac{P_n}{p_i \cdot p_j \cdot p_k} \right\rfloor, \quad 1 \leq i \leq \#\mathcal{P}[1, \sqrt{P_n}\,], \\ &\vdots \qquad\qquad \vdots \\ \#(A_1 \cap A_2 \cap \cdots \cap A_{\#\mathcal{P}[1,\sqrt{P_n}\,]}) &= \left\lfloor \frac{P_n}{p_i \cdot p_j \cdot \cdots \cdot p_{\#\mathcal{P}[1,\sqrt{P_n}\,]}} \right\rfloor. \end{aligned}$$

Some of the terms in (3.7.4) can easily be evaluated. We concentrate on the terms in (3.7.4) that involve only the sets A_i for $1 \leq i \leq n$, which are the sets associated with the first n primes $p_1, \ldots, p_n$. We sum these terms up and label the value B. That is,

$$\begin{aligned} B &= \#(A_1) + \#(A_2) + \cdots + \#(A_n) - \#(A_1 \cap A_2) - \#(A_1 \cap A_3) - \cdots \\ &\quad - \#(A_{n-1} \cap A_n) + \#(A_1 \cap A_2 \cap A_3) + \cdots + \#(A_{n-2} \cap A_{n-1} \cap A_n) \cdots \\ &\quad + (-1)^{n+1}(A_1 \cap A_2 \cap \cdots \cap A_n) \\ &= \left\lfloor \frac{P_n}{p_1} \right\rfloor + \left\lfloor \frac{P_n}{p_2} \right\rfloor + \cdots + \left\lfloor \frac{P_n}{p_n} \right\rfloor - \left\lfloor \frac{P_n}{p_1 \cdot p_2} \right\rfloor - \cdots - \left\lfloor \frac{P_n}{p_{n-1} \cdot p_n} \right\rfloor + \cdots . \end{aligned}$$

Now let $B^* = P_n - B = \left\lfloor \frac{P_n}{1} \right\rfloor - B$. The denominators in the terms in B^* are the divisors of P_n. Then

$$\begin{aligned} B^* &= \sum_{d|P_n} \mu(d) \frac{P_n}{d} \\ &= P_n \sum_{d|P_n} \mu(d) \frac{1}{d} \\ &= \phi(P_n). \end{aligned}$$

We now need to evaluate the rest of the terms in (3.7.4). We first solve for the cardinality of the sets in (3.7.4) that involve only the primes in $\mathcal{P}[p_{n+1,\sqrt{P_n}\,]}$. Recalling that Y^* is the power set of $\mathcal{P}[p_{n+1}, \sqrt{P_n}]$, we have that the cardinality of these sets give the value

$$\#(A_{n+1} \cup \cdots \cup \#A_{\#\mathcal{P}[1,\sqrt{P_n}\,]}) = \sum_{y^* \in Y^*} (-1)^{\#y^*+1} \left\lfloor \frac{P_n}{\prod_{p \in y^*} p} \right\rfloor .$$

The rest of the terms in (3.7.4) involve primes in both $\mathcal{P}[1, p_n]$ and $\mathcal{P}[p_{n+1}, \sqrt{P_n}\,]$. Let X^* be the power set of $\mathcal{P}[1, p_n]$. Then the value for the rest of the terms in (3.7.4) is

$$\sum_{x^* \in X^*} \sum_{y^* \in Y^*} (-1)^{x^*+y^*+1} \left\lfloor \frac{P_n}{\prod_{p \in x^*} p \prod_{p \in y^*} p} \right\rfloor .$$

Bringing these values together, we evaluate $\pi(P_n)$ as

$$\pi(P_n) = \phi(P_n) + \sum_{y^* \in Y^*} (-1)^{\#y^*} \left\lfloor \frac{P_n}{\prod_{p \in y^*} p} \right\rfloor \tag{3.7.5}$$

$$+ \sum_{x^* \in X^*} \sum_{y^* \in Y^*} (-1)^{\#x^* + \#y^*} \left\lfloor \frac{P_n}{\prod_{p \in x^*} p \prod_{p \in y^*} p} \right\rfloor + \#\mathcal{P}[1, \sqrt{P_n}\,] - 1. \tag{3.7.6}$$

Upon closer investigation, we readily see in (3.7.6) that for all $x^* \in X^*$, every possible value of $\prod_{p \in x^*} p$ is a divisor of P_n. The only divisor of P_n that we are missing is 1. However, looking at (3.7.5), we see that the values in the denominators of the sum are trivially multiplied by 1. We can therefore restate (3.7.5) and (3.7.6) in terms of the divisors of P_n with the help of the Möbius function.

$$\begin{aligned}\pi(P_n) &= \phi(P_n) + \sum_{d|P_n} \sum_{y^* \in Y^*} \left((-1)^{\#y^*} \mu(d) \left\lfloor \frac{P_n}{d \prod_{p \in y^*} p} \right\rfloor \right) + \#\mathcal{P}[1, \sqrt{P_n}\,] - 1 \\ &= \phi(P_n) + \sum_{d|P_n} \sum_{y^* \in Y^*} \left((-1)^{\#y^*} \mu\left(\frac{P_n}{d}\right) \left\lfloor \frac{d}{\prod_{p \in y^*} p} \right\rfloor \right) + \#\mathcal{P}[1, \sqrt{P_n}\,] - 1.\end{aligned}$$

Looking at the term $\mu\left(\frac{P_n}{d}\right)$, we see a curious fact. Let $\mu(P_n) = (-1)^r$ and $\mu(d) = (-1)^s$, where $r, s \geq 1$. Since $d|P_n$, then $\mu\left(\frac{P_n}{d}\right) = (-1)^{r-s}$. However, note that if $r - s$ is even, then $r + s$ is even. Moreover, if $r - s$ is odd, then $r + s$ is odd. Therefore, $(-1)^{r-s} = (-1)^{r+s} = (-1)^r(-1)^s = \mu(P_n)\mu(d)$, which implies that $\mu\left(\frac{P_n}{d}\right) = \mu(P_n)\mu(d)$. We now have

$$\begin{aligned}\pi(P_n) &= \phi(P_n) + \sum_{d|P_n} \sum_{y^* \in Y^*} \left((-1)^{\#y^*} \mu(P_n)\mu(d) \left\lfloor \frac{d}{\prod_{p \in y^*} p} \right\rfloor \right) + \#\mathcal{P}[1, \sqrt{P_n}\,] - 1 \\ &= \phi(P_n) + \mu(P_n) \sum_{d|P_n} \sum_{y^* \in Y^*} \left((-1)^{\#y^*} \mu(d) \left\lfloor \frac{d}{\prod_{p \in y^*} p} \right\rfloor \right) + \#\mathcal{P}[1, \sqrt{P_n}\,] - 1.\end{aligned} \tag{3.7.7}$$

□

We see that (3.7.7) gives an exact count to the prime density up to the ceiling P_n. The number of terms needed to completely solve (3.7.7) for some ceiling P_n is relatively easy

to determine. We use the same technique to count the terms as we used to count the terms in (2.4.2). We first note that P_n has $\sum_{i=0}^{n}\binom{n}{i} = 2^n = 2^{\#\mathcal{P}[1,p_n]}$ divisors. We also have that

$$\begin{aligned}\#Y^*(\mathcal{P}[p_{n+1},\sqrt{P_n}]) &= \sum_{i=0}^{\#\mathcal{P}[1,\sqrt{P_n}]-\#\mathcal{P}[1,p_n]}\binom{\#\mathcal{P}[1,\sqrt{P_n}]-\#\mathcal{P}[1,p_n]}{i}-1\\ &= 2^{\#\mathcal{P}[1,\sqrt{P_n}]-\#\mathcal{P}[1,p_n]}-1.\end{aligned}$$

Then the double sum in (3.7.7) has

$$\begin{aligned}2^{\#\mathcal{P}[1,p_n]}\left(2^{\#\mathcal{P}[1,\sqrt{P_n}]-\#\mathcal{P}[1,p_n]}-1\right) &= 2^{\#\mathcal{P}[1,p_n]}2^{\#\mathcal{P}[1,\sqrt{P_n}]-\#\mathcal{P}[1,p_n]}-2^{\#\mathcal{P}[1,p_n]}\\ &= 2^{\#\mathcal{P}[1,\sqrt{P_n}]}-2^n\end{aligned}$$

terms. Therefore, (3.7.7) has a total of $2^{\#\mathcal{P}[1,\sqrt{P_n}]}-2^n+3$ terms. If we were to use (2.4.2) to evaluate $\pi(P_n)$, then we would have $2^{\#\mathcal{P}[1,\sqrt{P_n}]}+1$ terms to evaluate, which, for all $n\in\mathbb{N}$, is more terms than (3.7.7).

	$2^{\#\mathcal{P}[1,\sqrt{P_n}]}+1$	$2^{\#\mathcal{P}[1,\sqrt{P_n}]}-2^n+3$	Difference(2^n-2)
n=4	65	51	14
n=5	32769	32739	30
n=6	$2^{40}+1$	$2^{40}-61$	62
n=7	$2^{127}+1$	$2^{127}-125$	126

Figure 3.7.1. Number of Terms in Both Prime Density Counts

Figure 3.7.1 has the total number of terms contained in the prime density counts of (2.4.2) and (3.7.7). Notice that the differences in the number of terms as n grows is relatively small. Therefore, the efficiency of our second count, although saving some terms, is not a significant improvement in the number of terms needed to be evaluated. Nevertheless, there *is* an improvement.

As done in Section 2.4, we bound the prime counting function $\pi(m)$. However, we restrict our ceiling m to P_n. We first show that, using the concept of pre-elimination of the

first n primes, we derive the exact upper bound as found in Theorem 2.4.10.

To begin deriving an upper bound for $\pi(P_n)$, we first must find a way to count the primes within the set $\mathcal{G}_{P_n}$. By pre-eliminating the multiples of the first n primes, we are left with the numbers that form the set $S^* = \{l | l \leq P_n, (l, P_n) = 1\}$. We must now eliminate the remaining composites in S^*, which are multiples of the primes $p_{n+1}, \ldots, p_{\#\mathcal{P}[1,\sqrt{P_n}]}$. We accomplish this by refining S^* to be

$$S^* = \{l | l \leq P_n, (l, P_n) = 1, \left(l, \frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right) = 1\}.$$

To count the number of elements in S^*, we create the characteristic function $s^*(l)$ of S^*, where

$$s^*(l) = \begin{cases} 1 & \text{if } l \in S^*, \\ 0 & \text{otherwise.} \end{cases}$$

Then we have the following proposition.

Proposition 3.7.3. *Let* $S^* = \{l | l \leq P_n, (l, P_n) = 1, \left(l, \frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right) = 1\}$. *Then*

$$\#S^* \leq P_n \prod_{p | P_{\#\mathcal{P}[1,\sqrt{P_n}]}} \left(1 - \frac{1}{p}\right) + 2^{\#\mathcal{P}[1,\sqrt{P_n}]}. \tag{3.7.8}$$

Proof. Note that (3.7.8) is the same bound obtained in Theorem 2.4.10. We will use the method that helped us derive the bound (2.4.10) in order to argue the following upper bound. Using the Möbius property from Proposition 2.4.7, we can restate s^* as

$$s^*(l) = \sum_{d_1 | (l, P_n)} \sum_{d_2 | \left(l, \frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)} \mu(d_1)\mu(d_2).$$

Then we get

$$
\begin{aligned}
\#S^* &= \sum_{l\le P_n} s^*(l) \\
&= \sum_{l\le P_n}\left(\sum_{d_1|(l,P_n)}\sum_{d_2|\left(l,\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)}\mu(d_1)\mu(d_2)\right) \\
&= \sum_{d_1|P_n}\sum_{d_2|\left(\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)}\mu(d_1)\mu(d_2)\left(\sum_{\substack{l\le P_n\\ d_1d_2|l}}1\right) \\
&= \sum_{d_1|P_n}\sum_{d_2|\left(\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)}\mu(d_1)\mu(d_2)\left\lfloor\frac{P_n}{d_1d_2}\right\rfloor \qquad (3.7.9)\\
&= \sum_{d_1|P_n}\sum_{d_2|\left(\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)}\mu(d_1)\mu(d_2)\left(\frac{P_n}{d_1d_2}+\left\lfloor\frac{P_n}{d_1d_2}\right\rfloor-\frac{P_n}{d_1d_2}\right) \\
&= \sum_{d_1|P_n}\sum_{d_2|\left(\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)}\mu(d_1)\mu(d_2)\frac{P_n}{d_1d_2}+\sum_{d_1|P_n}\sum_{d_2|\left(\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)}\mu(d_1)\mu(d_2)\left(\left\lfloor\frac{P_n}{d_1d_2}\right\rfloor-\frac{P_n}{d_1d_2}\right) \\
&= P_n\sum_{d_1|P_n}\frac{\mu(d_1)}{d_1}\sum_{d_2|\left(\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)}\frac{\mu(d_2)}{d_2}+\sum_{d_1|P_n}\sum_{d_2|\left(\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)}\mu\left(\frac{P_n}{d_1}\right)\mu(d_2)\left(\left\lfloor\frac{d_1}{d_2}\right\rfloor-\frac{d_1}{d_2}\right) \\
&= P_n\prod_{p|P_n}\left(1-\frac{1}{p}\right)\prod_{p|\left(\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)}\left(1-\frac{1}{p}\right)+\sum_{d_1|P_n}\sum_{d_2|\left(\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)}\mu(P_n)\mu(d_1)\mu(d_2)\left(\left\lfloor\frac{d_1}{d_2}\right\rfloor-\frac{d_1}{d_2}\right) \\
&= P_n\prod_{p|P_{\#\mathcal{P}[1,\sqrt{P_n}]}}\left(1-\frac{1}{p}\right)+\mu(P_n)\sum_{d_1|P_n}\sum_{d_2|\left(\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)}\mu(d_1)\mu(d_2)\left(\left\lfloor\frac{d_1}{d_2}\right\rfloor-\frac{d_1}{d_2}\right). \qquad (3.7.10)
\end{aligned}
$$

Looking at the last term in (3.7.10), we have that $\left|\left\lfloor\frac{d_1}{d_2}\right\rfloor-\frac{d_1}{d_2}\right|\le 1$. We now need only determine how many terms lie in the double sum in (3.7.10). Since P_n has $\sum_{i=0}^{n}\binom{n}{i}=2^n$ divisors and $\left(\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)$ has $\sum_{i=0}^{\#\mathcal{P}[1,\sqrt{P_n}]-n}\binom{\#\mathcal{P}[1,\sqrt{P_n}]-n}{i}=2^{\#\mathcal{P}[1,\sqrt{P_n}]-n}$ divisors, then the double sum has $2^n\cdot 2^{\#\mathcal{P}[1,\sqrt{P_n}]-n}=2^{\#\mathcal{P}[1,\sqrt{P_n}]}$ terms, and $\#S^*$ can be bounded

from above as follows:

$$\#S^* \leq P_n \prod_{p|P_{\#\mathcal{P}[1,\sqrt{P_n}]}} \left(1 - \frac{1}{p}\right) + 2^{\#\mathcal{P}[1,\sqrt{P_n}]}. \tag{3.7.11}$$

□

It seems as if we are not any closer to improving the bound in Theorem 2.4.10. However, this is not true. By restricting the ceiling of the prime counting function to P_n, we observe a certain behavior: Because of the double sum in (3.7.9), we can assume certain values in order to reduce the bound (3.7.11).

Looking at the sum in (3.7.9), we see that many zeros occur. Since $P_n = p_1 p_2 \dots p_n$, then for any divisor d of $P_{\#\mathcal{P}[1,\sqrt{P_n}]}$, if $\omega(d) > n$, then $\left\lfloor \frac{P_n}{d} \right\rfloor = 0$. There are some d such that if $\omega(d) \leq n$ then $\left\lfloor \frac{P_n}{d} \right\rfloor = 0$. However, since not all these d are guaranteed to produce zero terms, we restrict our investigation to those d such that $\omega(d) > n$. In (3.7.9), we express the divisors of $P_{\#\mathcal{P}[1,\sqrt{P_n}]}$ as combinations of the divisors d_1 and d_2 of P_n and $\left(\frac{P_{\#\mathcal{P}[1,\sqrt{P_n}]}}{P_n}\right)$, respectively. However, we return to the sum in (2.4.7) and deal with the divisors of $P_{\#\mathcal{P}[1,\sqrt{P_n}]}$. We do this so that we can propose a better upper bound.

We begin by with determining the cardinality of the set $S = \{l | l \leq P_n, (l, P_{\#\mathcal{P}[1,\sqrt{P_n}]}) = 1\}$.

Proposition 3.7.4. *The cardinality of S is*

$$\#S = P_n \prod_{p|P_{\#\mathcal{P}[1,\sqrt{P_n}]}} \left(1 - \frac{1}{p}\right) + \sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]} \\ \omega(d)\leq n}} \mu(d) \left(\left\lfloor \frac{P_n}{d} \right\rfloor - \frac{P_n}{d}\right) - P_n \sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]} \\ n<\omega(d)\leq \#\mathcal{P}[1,\sqrt{P_n}]}} \frac{\mu(d)}{d}.$$

Proof. We begin as we did in Theorem 2.4.10.

$$\begin{aligned}
\#S &= \sum_{l \le P_n} s(l) \\
&= \sum_{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]}} \left(\sum_{\substack{l \le P_n \\ d \mid l}} 1 \right) \\
&= \sum_{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]}} \mu(d) \left\lfloor \frac{P_n}{d} \right\rfloor \\
&= \sum_{\substack{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]} \\ \omega(d) \le n}} \mu(d) \left\lfloor \frac{P_n}{d} \right\rfloor . \qquad (3.7.12)
\end{aligned}$$

Since (2.4.8) and (3.7.12) are equal, we can derive

$$\begin{aligned}
\sum_{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]}} \mu(d) \left\lfloor \frac{P_n}{d} \right\rfloor &= \sum_{\substack{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]} \\ \omega(d) \le n}} \mu(d) \left\lfloor \frac{P_n}{d} \right\rfloor \\
\Rightarrow \sum_{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]}} \mu(d) \left(\frac{P_n}{d} + \left\lfloor \frac{P_n}{d} \right\rfloor - \frac{P_n}{d} \right) &= \sum_{\substack{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]} \\ \omega(d) \le n}} \mu(d) \left(\frac{P_n}{d} + \left\lfloor \frac{P_n}{d} \right\rfloor - \frac{P_n}{d} \right) \\
\Rightarrow \sum_{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]}} \mu(d) \frac{P_n}{d} + \sum_{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]}} \mu(d) \left(\left\lfloor \frac{P_n}{d} \right\rfloor - \frac{P_n}{d} \right) &= \sum_{\substack{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]} \\ \omega(d) \le n}} \mu(d) \frac{P_n}{d} \\
&\quad + \sum_{\substack{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]} \\ \omega(d) \le n}} \mu(d) \left(\left\lfloor \frac{P_n}{d} \right\rfloor - \frac{P_n}{d} \right) .
\end{aligned}$$

Since

$$\sum_{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]}} \mu(d) \frac{P_n}{d} = \sum_{\substack{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]} \\ \omega(d) \le n}} \mu(d) \frac{P_n}{d} + \sum_{\substack{d \mid P_{\#\mathcal{P}[1,\sqrt{P_n}\,]} \\ n < \omega(d) \le \#\mathcal{P}[1,\sqrt{P_n}\,]}} \mu(d) \frac{P_n}{d},$$

then

$$\begin{aligned}\sum_{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}} \mu(d)\left(\left\lfloor\frac{P_n}{d}\right\rfloor-\frac{P_n}{d}\right) &= \sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}\\ \omega(d)\le n}} \mu(d)\frac{P_n}{d}+\sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}\\ \omega(d)\le n}} \mu(d)\left(\left\lfloor\frac{P_n}{d}\right\rfloor-\frac{P_n}{d}\right)\\ &\quad-\sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}\\ \omega(d)\le n}} \mu(d)\frac{P_n}{d}-\sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}\\ n<\omega(d)\le\#\mathcal{P}[1,\sqrt{P_n}]}} \mu(d)\frac{P_n}{d}\\ &= \sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}\\ \omega(d)\le n}} \mu(d)\left(\left\lfloor\frac{P_n}{d}\right\rfloor-\frac{P_n}{d}\right)-\sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}\\ n<\omega(d)\le\#\mathcal{P}[1,\sqrt{P_n}]}} \mu(d)\frac{P_n}{d}.\end{aligned} \tag{3.7.13}$$

Substituting (3.7.13) into (2.4.9), we get

$$\begin{aligned}\#S &= \sum_{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}} \mu(d)\frac{P_n}{d}+\sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}\\ \omega(d)\le n}} \mu(d)\left(\left\lfloor\frac{P_n}{d}\right\rfloor-\frac{P_n}{d}\right)-\sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}\\ n<\omega(d)\le\#\mathcal{P}[1,\sqrt{P_n}]}} \mu(d)\frac{P_n}{d}\\ &= P_n\prod_{p|P_{\#\mathcal{P}[1,\sqrt{P_n}]}}\left(1-\frac{1}{p}\right)+\sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}\\ \omega(d)\le n}} \mu(d)\left(\left\lfloor\frac{P_n}{d}\right\rfloor-\frac{P_n}{d}\right)-P_n\sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}\\ n<\omega(d)\le\#\mathcal{P}[1,\sqrt{P_n}]}}\frac{\mu(d)}{d}.\end{aligned} \tag{3.7.14}$$

□

We are now ready to conjecture an inequality that will allow us to refine the upper bound for $\#S$.

Conjecture 3.7.5. *For $n \ge 4$, we have*

$$\begin{aligned}&P_n\prod_{p|P_n}\left(1-\frac{1}{p}\right)-P_n\prod_{p|P_{\#\mathcal{P}[1,\sqrt{P_n}]}}\left(1-\frac{1}{p}\right)\\ &\qquad\ge\sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}\\ \omega(d)\le n}} \mu(d)\left(\left\lfloor\frac{P_n}{d}\right\rfloor-\frac{P_n}{d}\right)-P_n\sum_{\substack{d|P_{\#\mathcal{P}[1,\sqrt{P_n}]}\\ n<\omega(d)\le\#\mathcal{P}[1,\sqrt{P_n}]}}\frac{\mu(d)}{d}.\end{aligned} \tag{3.7.15}$$

Theorem 3.7.6. *Let $n \ge 0$. Given the truth of Conjecture 3.7.5, we derive the following.*

$$\pi(P_n) \le \phi(P_n) + n - 1. \tag{3.7.16}$$

Proof. Using (3.7.15) on (3.7.14), we get

$$\begin{aligned} \#S &\leq \prod_{p|P_n} \left(1 - \frac{1}{p}\right) \\ &= \phi(P_n). \end{aligned}$$

Since $\pi(P_n) = \#S + n - 1$, then $\pi(P_n) \leq \phi(P_n) + n - 1$. □

Looking at Figure 3.7.2, we can see the difference in the estimates between the upper bound in (2.4.10) and our new one in (3.7.16).

	$\pi(P_n)$	$P_n \prod_{p\mid P_{\#\mathcal{P}[1,\sqrt{P_n}]}} \left(1 - \frac{1}{p}\right) + 2^{\#\mathcal{P}[1,\sqrt{P_n}]}$	$\phi(P_n) + n - 1$
n=4	46	104.2797	51
n=5	343	33088.4065	484
n=6	3248	$3239.21 + 2^{40}$	5765
n=7	42331	$43381.9512 + 2^{127}$	92166
n=8	646029	$6757603.3492 + 2^{443}$	1658887
n=9	12283531	$13021785.8987 + 2^{1748}$	36495368

Figure 3.7.2. Estimates of $\pi(P_n)$ Using Two Upper Bounds

To continue exploring this upper bound would be to delve too deep into the subject of counting, which is not the sole goal of this project; rather, we have already shown that using the base concept of sifting $\mathcal{T}^{(P_n)}$ mod P_n refines the upper bounds for S and $\pi(P_n)$ obtained in Chapter 2. However, one possible exploration of (3.7.16) would be to take away the restriction of our ceiling being P_n but still keeping the product over primes of the term $\prod_{p|P_n} \left(1 - \frac{1}{p}\right)$ intact. That is, let $m \in \mathbb{N}$. If $P_n \leq m < P_{n+1}$, then the form of the upper bound in Theorem 3.7.6 can still be used, still assuming Conjecture 3.7.5 is true.

Conjecture 3.7.7. *Let $m \in \mathbb{N}$, where $P_n \leq m < P_{n+1}$. Then*

$$\pi(m) \leq m \prod_{p|P_n} \left(1 - \frac{1}{p}\right) + n - 1.$$

The motivation behind generalizing Theorem 3.7.6 in this way can been seen in the behavior of $\phi(m)$. Figure 3.7.3 contains a scatter plot of both functions $\phi(m)$ and $\pi(m)$.

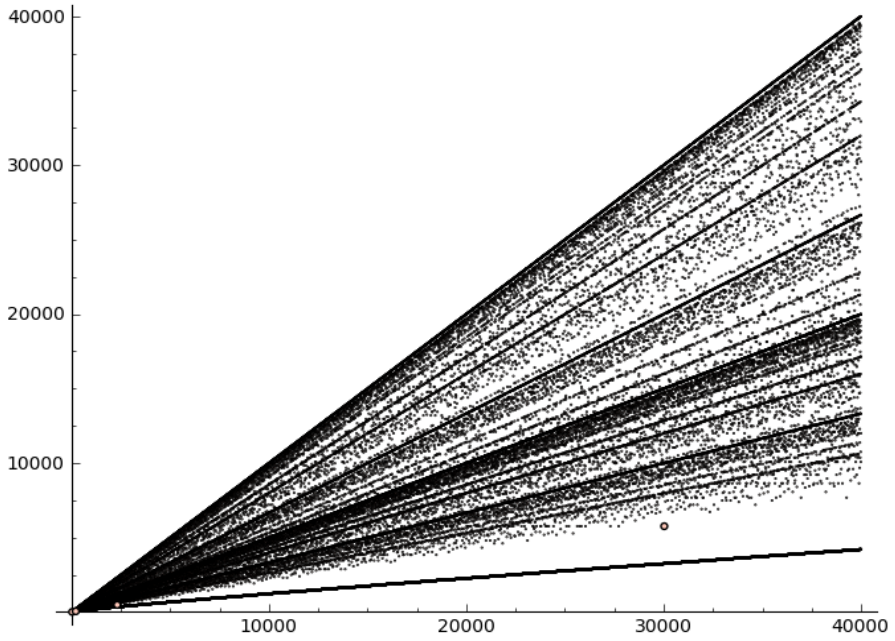

Figure 3.7.3. A Plot of the Erratic Behavior of $\phi(m)$ for $m \leq 40000$ Compared to $\pi(m)$.

The bottom line-like set of points are values of $\pi(m)$, and the set of erratic cone-shaped points are values of $\phi(m)$. There are four points of interest, points that are slightly larger than the rest. These are $(P_n, \phi(P_n) + n - 1)$ for $n = 3, 4, 5, 6$. These seem to be the lowest points in the cone shape, but are still well above the values of $\pi(m)$. This seemingly implies that the points $(P_n, \phi(P_n) + n - 1)$, $i \geq 7$, will also be the bottom-most of this increasingly wide cone. Conjecture 3.7.7 takes these points and creates a linear function between each, then uses this linear function as an upper bound for $\pi(m)$. A possible proof for this idea would probably involve a deeper investigation into the behavior of the unique factorization of numbers, along with an exploration of the asymptotic behavior of the product $\prod_{p|P_n} \left(1 - \frac{1}{p}\right)$ as $n \to \infty$, a daunting task not to be undertaken at the moment.

4
Analysis

4.1 Performace Analysis of $\mathbb{G}_i$

In order to analyze the $\mathbb{G}_i$ algorithm, we first observe its performance. We use three statistics (primes acquired vs. time, iterations performed vs. time, efficiency rate vs. time) to do so.

	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_3$	$\mathbb{G}_4$	$\mathbb{G}_5$	$\mathbb{G}_6$	$\mathbb{G}_7$
Primes Acquired	518,134	1,085,262	2,613,152	7,745,089	39,081,631	45,057,152	10,519,320
Largest Prime Acquired	7,656,281	43,151,743	67,749,349	136,866,599	757,735,687	880,368,221	189,310,063
Efficiency Rate	41.12%	46.34%	51.05%	53.74%	56.97%	57.8%	60.56%
Iterations Performed	402	564	846	1400	3001	3214	1621

Figure 4.1.1. Data on the First Seven $\mathbb{G}$ Algorithms After One-Hour Run-Time

Figure 4.1.1 shows the basic stats of the performance of $\mathbb{G}_i$ for $1 \leq i \leq 7$ after one hour of run-time. Notice that the number of primes acquired by each $\mathbb{G}_i$ grows immensely up to $\mathbb{G}_6$. However, we see that $\mathbb{G}_7$ doesn't perform as expected, since it didn't exceed the number of primes acquired by the $\mathbb{G}_5$ and $\mathbb{G}_6$ sieves in the time alloted. Figure 4.1.2 contains a graph that shows the number of primes acquired by the first seven $\mathbb{G}_i$ sieves in

one hour.

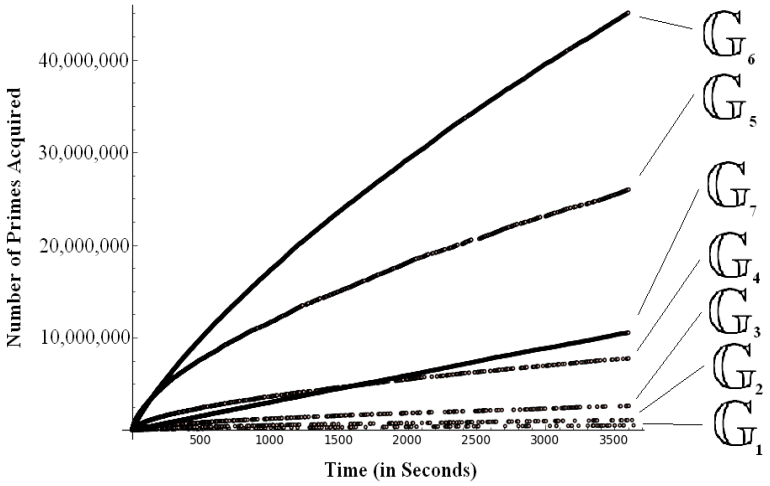

Figure 4.1.2. Primes Acquired By $\mathbb{G}_i$, $1 \leq i \leq 7$ After One Hour

Of interest is the behavior of the first six $\mathbb{G}_i$ sieves compared to $\mathbb{G}_7$. The prime acquisition rate of $\mathbb{G}_i$ for $1 \leq i \leq 6$ decreases, whereas $\mathbb{G}_7$'s acquisition rate seems to be steady. This possibly implies that the number of primes acquired by $\mathbb{G}_7$ may match and even surpass the number acquired by $\mathbb{G}_5$ and $\mathbb{G}_6$ if given sufficient time. This is in fact the case, as we will see in Section 4.3.

Figure 4.1.3 shows the performance of $\mathbb{G}_i$ for $1 \leq i \leq 7$ in terms of iterations performed after one hour of run-time. Looking closely at the area around the origin, we notice that $\mathbb{G}_7$ actually performs less iterations than even $\mathbb{G}_1$ during the beginning of execution. Yet, it "catches" up with $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ within the first few minutes of execution, and surpasses $\mathbb{G}_4$ after about 25 minutes. As already stated, $\mathbb{G}_7$ will outperform $\mathbb{G}_5$ and $\mathbb{G}_6$ given more run-time. Viewing Figure 4.1.3 does not give more information than Figure 4.1.2 other than highlighting certain behaviors not obvious in the first graph, especially during the

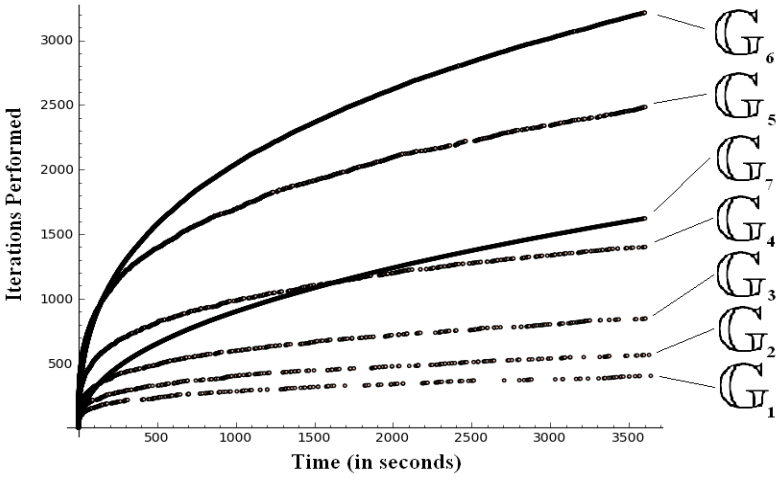

Figure 4.1.3. Iterations Performed By $\mathbb{G}_i$, $1 \leq i \leq 7$ After One Hour

beginning of execution.

The graph in Figure 4.1.4 shows the efficiency rate of $\mathbb{G}_i$ for $1 \leq i \leq 7$. This statistic seemingly goes against the information obtained from the first two graphs because it shows an increase in efficiency of composite elimination as i increases. There seems to be a correlation between increased efficiency and poorer short-term performance. The question of why this happens is related to why $\mathbb{G}_7$'s performance is so poor relative to the other sieves in the short-term.

The reason why $\mathbb{G}_7$ performs worse than $\mathbb{G}_5$ and $\mathbb{G}_6$ after one hour of run-time is that it sifts $\phi(P_6) = 5760$ primitive residual classes, whereas $\mathbb{G}_5$ and $\mathbb{G}_6$ each sift $\phi(P_4) = 48$ and $\phi(P_5) = 480$ primitive residual classes, respectively. Sifting an increased number of primitive residual classes, although pre-eliminating many composites, results in extra steps necessary in preparation. In Section 4.3, we clarify this idea by timing the $\mathbb{G}_i$ algorithm in steps and deriving a function that estimates the number of steps necessary to sift the

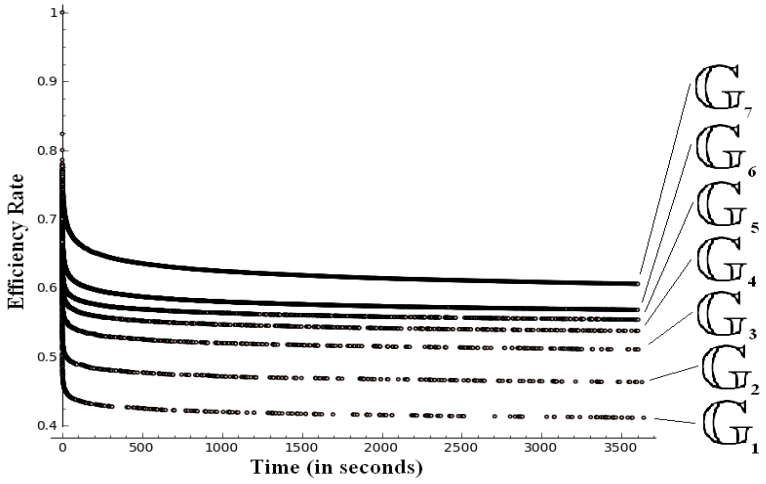

Figure 4.1.4. Efficiency Rate of $\mathbb{G}_i$, $1 \leq i \leq 7$ After One Hour

primes out of $\{2, 3, \ldots, m\}$ for any ceiling $m \in \mathbb{N}$, which gives a clearer picture of $\mathbb{G}_i$'s short- and long-term performance.

4.2 Prime and Composite Density of $[p_j^2, p_{j+1}^2)$ and $[p_i^2, p_k^2)$

We must determine how many composites are eliminated, and the total number of times the process of elimination is executed, in order to further analyze $\mathbb{G}_i$. We first investigate the characteristics of the j^{th} interval (i.e. $\mathcal{G}_j^{(P_n)}$) with respect to prime and composite density using the analytical tools mentioned in Section 2.7. We restate certain ideas already visited in Section 2.7 to refreshen them in our minds.

The Prime Number Theorem states that the prime counting function $\pi(m) \sim \frac{m}{\ln m}$ as $n \to \infty$. The PNT allows us to estimate the prime density of the j^{th} interval as

$$\pi(p_{j+1}^2) - \pi(p_j^2) \approx \frac{p_{j+1}^2}{\ln p_{j+1}^2} - \frac{p_j^2}{\ln p_j^2}.$$

This estimate requires knowledge of the consecutive primes p_j and p_{j+1}. However, by Proposition 2.7.2, we have that $p_{j+1} \approx p_j + \ln p_j$, We therefore use this estimate of the

"average" spacing of primes around a prime p_j to say that prime density of the j^{th} interval is approximately

$$\begin{aligned}\pi(p_{j+1}^2) - \pi(p_j^2) &\approx \pi[(p_j + \ln p_j)^2] - \pi(p_j^2) \\ &\approx \frac{(p_j + \ln p_j)^2}{\ln\left[(p_j + \ln p_j)^2\right]} - \frac{p_j^2}{\ln p_j^2}. \end{aligned} \tag{4.2.1}$$

Figure 4.2.1 contains a scatter plot showing two things: 1) the actual number of primes in the j^{th} intervals for $1 \leq j \leq 3100$; and, 2) the prime density of the j^{th} interval using the estimate in (4.2.1) (the darkened line-like set of points).

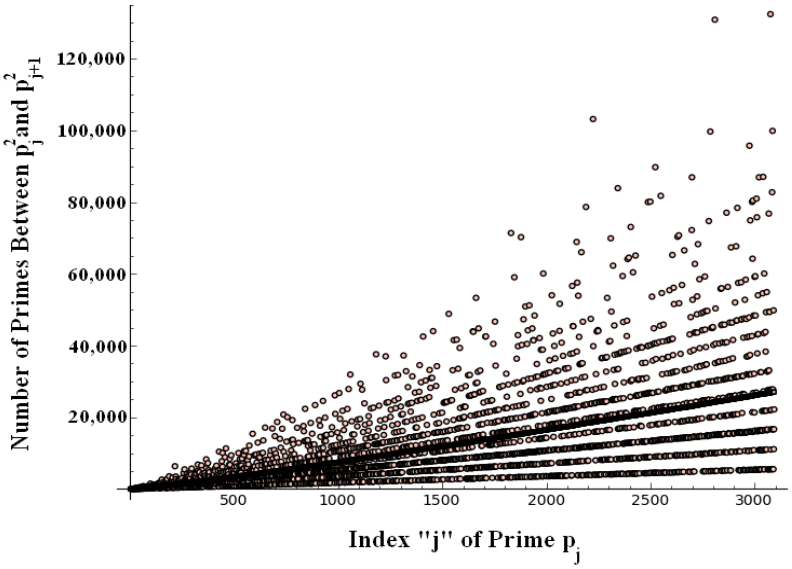

Figure 4.2.1. The number of primes in the j^{th} interval, along with the estimate $\pi(p_{j+1}^2) - \pi(p_j^2) \approx \pi[(p_j + \ln p_j)^2] - \pi(p_j^2) \approx \frac{(p_j+\ln p_j)^2}{\ln[(p_j+\ln p_j)^2]} - \frac{p_j^2}{\ln p_j^2}$.

We have that $\#\mathcal{P}[p_j^2, p_{j+1}^2 - 1]$ is the prime density of the j^{th} interval. We see in Figure 4.2.1 that $\#\mathcal{P}[p_j^2, p_{j+1}^2 - 1]$ behaves wildly, obviously due to the irratic distribution of the primes. That is, not only is the length of the j^{th} interval fluctuating, but also the distribution of primes isn't uniform, thus resulting in a frantic scatter plot.

Let the estimate (4.2.1) be denoted as $P^*(j)$. The darkened line of points in Figure 4.2.1 are the values of $P^*(j)$ for $1 \leq j \leq 3100$. Although the values of $\#\mathcal{P}[p_j^2, p_{j+1}^2 - 1]$ cover a wide range, the estimate $P^*(j)$ seems to be a good weighted mean.

Tied closely to knowing how many primes are in the j^{th} interval is knowing how many composites there are that are congruent $\mathcal{T}^{(P_n)}$ mod P_n. This tells us how many distinct composites are eliminated. This estimate can be obtained by subtracting our estimate of the j^{th} interval's prime density from the cardinality of said interval. Obtaining the cardinality of the j^{th} interval, which is equivalent to finding $\#\mathcal{G}_j^{(P_n)}$, can be difficult since the value fluctuates wildly because of the inconsistency of the spacing of primes; however, we can use the average spacing of primes to deduce that there should be about $(p_j + \ln p_j)^2 - p_j^2$ numbers in the j^{th} interval. But since we concentrate only on numbers congruent $\mathcal{T}^{(P_n)}$ mod P_n, of which only $\phi(P_n)$ out of every P_n are, then we estimate $\#\mathcal{G}_j^{(P_n)}$ as

$$\begin{aligned}\#\mathcal{G}^{(P_n)} &\approx [(p_j + \ln p_j)^2 - p_j^2] \cdot \frac{\phi(P_n)}{P_n} \\ &= [2 \ln p_j + (\ln p_j)^2] \cdot \frac{\phi(P_n)}{P_n}.\end{aligned}$$

Thus, there are approximately

$$[2 \ln p_j + (\ln p_j)^2] \cdot \frac{\phi(P_n)}{P_n} - \left(\frac{(p_j + \ln p_j)^2}{\ln (p_j + \ln p_j)^2} - \frac{p_j^2}{\ln p_j^2} \right) \tag{4.2.2}$$

composites in the j^{th} interval. This value tells us how many distinct numbers are eliminated when sifting the j^{th} interval.

This investigation into the prime and composite density of the j^{th} interval is purely for curiosity's sake since it leads to difficult and convoluted equations. For instance, using (4.2.2) to aid in finding out how many *times* composites are eliminated, which is vital to modeling the efficiency rate (this rate is defined soon) and timing $\mathbb{G}_i$, is seemingly

impossible since the next possible manipulation of (4.2.2) is difficult to see. Therefore, we consolidate the j^{th} intervals into the set $\bigcup_{j=i}^{k-1}[p_j^2, p_{j+1}^2) = [p_i^2, p_k^2)$, where p_k^2 functions as the ceiling of our sieve for some arbitrarily large k. (Note that this interval begins at p_i^2 since this is where $\mathbb{G}_i$ begins sifting.)

We begin with the following definition.

Definition 4.2.1. Let $C(m)$ be the set of all composites $c < m$. △

We know that the $\mathbb{G}_i$ sieve begins sifting at p_i^2. Then $\mathbb{G}_i$ eliminates at most $\#C(p_k^2) - \#C(p_i^2)$ distinct composites (recall that our ceiling must be the square of some large prime p_k). However, many of these composites are pre-eliminated by sifting $\mathcal{T}^{(P_n)}$ mod P_n. Specifically, the multiples of the first n primes are gone. We therefore refine Definition 4.2.1 to fit our analytical needs.

Definition 4.2.2. Let $C(i,k)$ denote the set of composites in the interval $[p_i^2, p_k^2)$ congruent $\mathcal{T}^{(P_n)}$ mod P_n. We define the value $C_i(p_k^2) = \#C(i,k)$. △

We estimate the value $C_i(p_k^2)$ in order to determine approximately how many distinct composites are eliminated during execution.

Conjecture 4.2.3. *We estimate the value $C_i(p_k^2)$ as*

$$C_i(p_k^2) \approx \frac{\phi(P_n)}{P_n}(p_k^2 - p_i^2) - \frac{p_k^2}{2\ln p_k} + \frac{p_i^2}{2\ln p_i}.$$

We estimate the refined composite density of the interval $[p_i^2, p_k^2)$ by first approximating how many numbers in $[p_i^2, p_k^2)$ are congruent $\mathcal{T}^{(P_n)}$ mod P_n. We estimate this value as $\#[p_i^2, p_k^2)^* \approx \frac{\phi(P_n)}{P_n} \cdot (p_k^2 - p_i^2)$ since only $\phi(P_n)$ out of every P_n numbers are in these residual classes. We then subtract the prime density of $[p_i^2, p_k^2)$ from this value. Since the prime density of $[p_i^2, p_k^2)$ is $\frac{p_k^2}{\ln p_k^2} - \frac{p_i^2}{\ln p_i^2}$, then subtracting $\pi(p_k^2) - \pi(p_i^2)$ from $\#[p_i^2, p_k^2)^*$

yields

$$C_i(p_k^2) \approx \frac{\phi(P_n)}{P_n}(p_k^2 - p_i^2) - \frac{p_k^2}{2\ln p_k} + \frac{p_i^2}{2\ln p_i}.$$

Throughout execution, $\mathbb{G}_i$ keeps track of the actual value $C_i(p_k^2)$ by counting throughout each iteration. The value $C_i(p_k^2)$ tells us how many distinct composites have been eliminated after execution is complete.

Although $C_i(p_k^2)$ composites are eliminated, $\mathbb{G}_i$ actually executes the elimination of many of these composites more than once. This happens because a composite number c is eliminated by each of its distinct prime factors that are less than or equal to its square root. That is, we have that c will be eliminated by the prime factors $p \leq \sqrt{c}$ (we need not worry about composites with a prime factor in $\{p_1, \ldots, p_n\}$ since these composites are not congruent $\mathcal{T}^{(P_n)} \bmod P_n$). We must count the number of actual eliminations executed by $\mathbb{G}_i$ during run-time. This number is defined as follows.

Definition 4.2.4. Let $C(i,k)$ be the set as defined in Definition 4.2.2, and, for all $c \in C(i,k)$, let $\omega^*(c)$ be the number of distinct prime factors of c less than or equal to $\sqrt{c}$. We define the value $S_i(p_k^2)$ as

$$S_i(p_k^2) = \sum_{c \in C(i,k)} \omega^*(c).$$

△

$S_i(p_k^2)$ represents the number of actual eliminations executed by $\mathbb{G}_i$ upon reaching the ceiling p_k^2. We make the following conjecture on the behavior of $S_i(p_k^2)$.

Conjecture 4.2.5. *We estimate the value $S_i(p_k^2)$ as*

$$S_i(p_k^2) \approx \frac{\phi(P_n)}{P_n}(p_k^2 - p_i^2) \cdot (\ln\ln p_k - \ln\ln p_i).$$

To estimate this value, we observe the following: for any prime p_j such that $p_i \leq p_j < p_k$, we have that p_j has approximately $\frac{\phi(P_n)}{P_n} \cdot \left(\frac{p_k^2 - p_i^2}{p_j}\right)$ multiples in the interval $[p_i^2, p_k^2)$ that are congruent $\mathcal{T}^{(P_n)}$ mod P_n. That is, we wish only to count the multiples that are congruent $\mathcal{T}^{(P_n)}$ mod P_n, of which only $\phi(P_n)$ out of every P_n are. To get the total number of eliminations actually executed by $\mathbb{G}_i$, we sum this value over all primes p_j for $i \leq j < k$, which yields the estimate

$$\begin{aligned}\sum_{j=i}^{k-1} \frac{\phi(P_n)}{P_n} \cdot \left(\frac{p_k^2 - p_i^2}{p_j}\right) &= \frac{\phi(P_n)}{P_n}(p_k^2 - p_i^2) \sum_{j=i}^{k-1} \frac{1}{p_j} \\ &= \frac{\phi(P_n)}{P_n}(p_k^2 - p_i^2) \left(\sum_{j=1}^{k-1} \frac{1}{p_j} - \sum_{j=1}^{n} \frac{1}{p_j}\right). \end{aligned} \tag{4.2.3}$$

To estimate the difference inside the parenthesis of (4.2.3), we recall from (2.7.2) that $-\sum_{p_j<x} \frac{1}{p_j} \approx \ln W(x)$. Then

$$\begin{aligned}\sum_{p_j<x} \frac{1}{p_j} &\approx -\ln W(x) \\ &\approx -\ln \frac{1}{\ln x} \\ &= \ln \ln x. \end{aligned} \tag{4.2.4}$$

Using (4.2.4), we further (4.2.3) and derive

$$\sum_{j=i}^{k-1} \frac{\phi(P_n)}{P_n} \cdot \left(\frac{p_k^2 - p_i^2}{p_j}\right) = \frac{\phi(P_n)}{P_n}(p_k^2 - p_i^2)\left(\ln \ln p_k - \ln \ln p_i\right). \tag{4.2.5}$$

Throughout execution, $\mathbb{G}_i$ also keeps track of this statistic by counting through each iteration. The value $S_i(p_k^2)$ tells us how many times composites (though not necessarily distinct) are eliminated.

Now that $C_i(p_k^2)$ and $S_i(p_k^2)$ are defined, we define the efficiency rate statistic: the ratio between $C_i(p_k^2)$ and $S_i(p_k^2)$.

Definition 4.2.6. We define $E_i(p_k^2)$ as the efficiency rate of $\mathbb{G}_i$ upon reaching the ceiling p_k^2, where

$$E_i(p_k^2) = \frac{C_i(p_k^2)}{S_i(p_k^2)}$$

△

The efficiency rate $E_i(p_k^2)$ tells us how much of the computing power used for composite eliminations is actually used on eliminating *distinct* composites, whereas $1 - E_i(p_k^2)$ is the percent of this computing power used for redundant eliminations. Using the estimates from Conjectures 4.2.3 and 4.2.5, we conjecture an estimate for $E_i(p_k^2)$.

Conjecture 4.2.7. *We estimate the value* $E_i(p_k^2)$ *as*

$$E_i(p_k^2) \approx \frac{\frac{\phi(P_n)}{P_n}(p_k^2 - p_i^2) - \frac{p_k^2}{2\ln p_k} + \frac{p_i^2}{2\ln p_i}}{\frac{\phi(P_n)}{P_n}(p_k^2 - p_i^2) \cdot (\ln\ln p_k - \ln\ln p_i)}.$$

For the sake of analyzing the $\mathbb{G}_i$ sieve, the efficiency rate, $E_i(p_k^2)$, is calculated after the completion of every iteration.

Figure 4.2.2 contains a graph that shows the modeled efficiency rate, $E_i(p_k^2)$, for $1 \leq i \leq 11$. There are eleven curves, the top-most representing the efficiency rate of $\mathbb{G}_{11}$ and the bottom-most of $\mathbb{G}_1$.

Comparing the graphs in Figure 4.1.4 and Figure 4.2.2, we note that our model fits our empirical data. We can therefore confidently conclude that the $\mathbb{G}_i$ sieve is definitively more efficient in composite elimination as i increases. That is to say, less computing is wasted on redundant composite eliminations. However, increased efficiency seems to have a negative impact on the short-term prime acquisition rate for higher i's. We determine the reason for this behavior by timing $\mathbb{G}_i$ in steps.

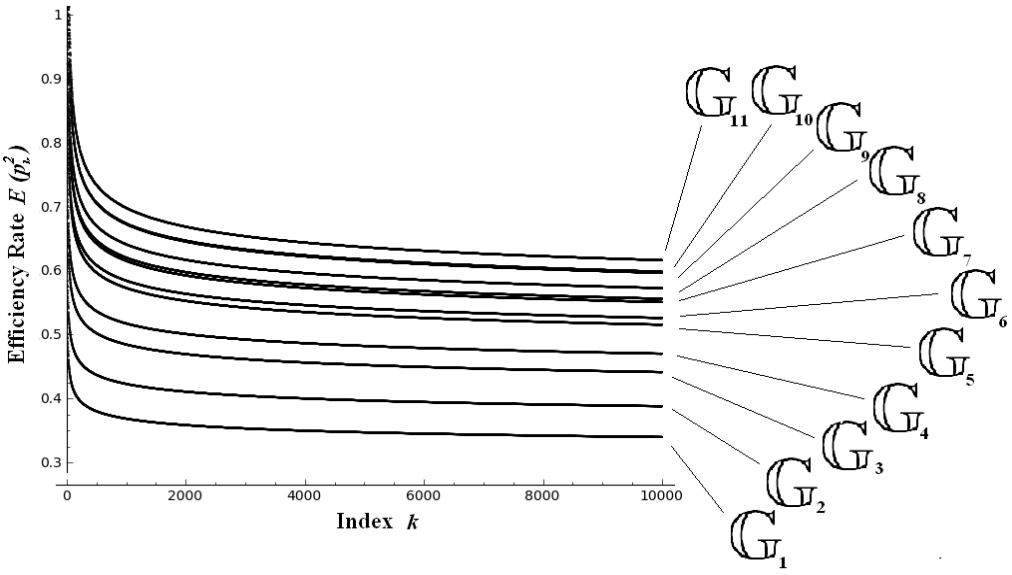

Figure 4.2.2. Behavior of $E_i(p_k^2)$ for $1 \leq i \leq 11, k \leq 10000$

4.3 Timing the $\mathbb{G}_i$ Algorithm

Our goal in this section is to create a function that models the time that the $\mathbb{G}_i$ takes to sift up to a ceiling p_k^2.

Conjecture 4.3.1. *Let $T_i(p_k^2)$ be the total number of steps needed for $\mathbb{G}_i$ to sift up to the ceiling p_k^2. We approximate $T_i(p_k^2)$ as*

$$\begin{aligned} T_i(p_k^2) \approx 2 + \frac{k+3n}{2} + \phi(P_n)\left(\frac{p_k^2 - p_i^2}{P_n}\right) + (k-n)\left(20 + 13\phi(P_n) + \frac{k-n}{2}\right) \\ + (P_n - 1)(5.4 \ln P_n) + \frac{2\phi(P_n)}{P_n}(p_k^2 - p_i^2)\left(2 \cdot (\ln\ln p_k - \ln\ln p_i^2) + 1\right) \\ - \frac{p_k^2}{\ln p_k^2} + \frac{p_i^2}{\ln p_i} \end{aligned}$$

Before we begin timing $\mathbb{G}_i$ in steps, we first make certain assumptions on the number of steps this algorithm takes for certain operations.[1]

[1] As the author is not familiar with this aspect of computer science, assumptions made in this section are derived from an intuitive standpoint.

- The process of "looking up", adding, and/or removing an element in a set requires one step.
- Assigning equality to a parameter and checking the value of a parameter requires one step.
- The operations of addition, subtraction, multiplication, and division each take one step.
- Determining the modular congruence of a number requires three steps: 1) division, 2) multiplication, and 3) subtraction.
- Determining the integer remainder of a quotient requires one step: division.
- Performing the Euclidean Algorithm on two integers y and x, where $y < x$, to determine (y, x) and the two integers r, s such that $yr + xs = (y, x)$ requires approximately $5.4 \ln x$ steps.

The justification for the timing of the last step was derived from empirical data. Recalling that the Euclidean Algorithm functions in iterations until (y, x) is found, we determined the average number of iterations needed to derive (y, x) by performing the algorithm for each $y < x$. That is, we fixed an integer $x \geq 10$. We then ran the Euclidean Algorithm for all $y < x$, counting the number of iterations needed to determine (y, x) for each y. After this, we took the total number of iterations performed over all y and divided this by $(x - 1)$ to find the mean value. We then increased x by one and began this process again, up until we reached $x = 130000$.

Figure 4.3.1 contains a scatter plot of the values attained from our data. Alongside our data, the function $.9 \ln x$ is plotted. We see that $.9 \ln x$ is a decent model for our data.

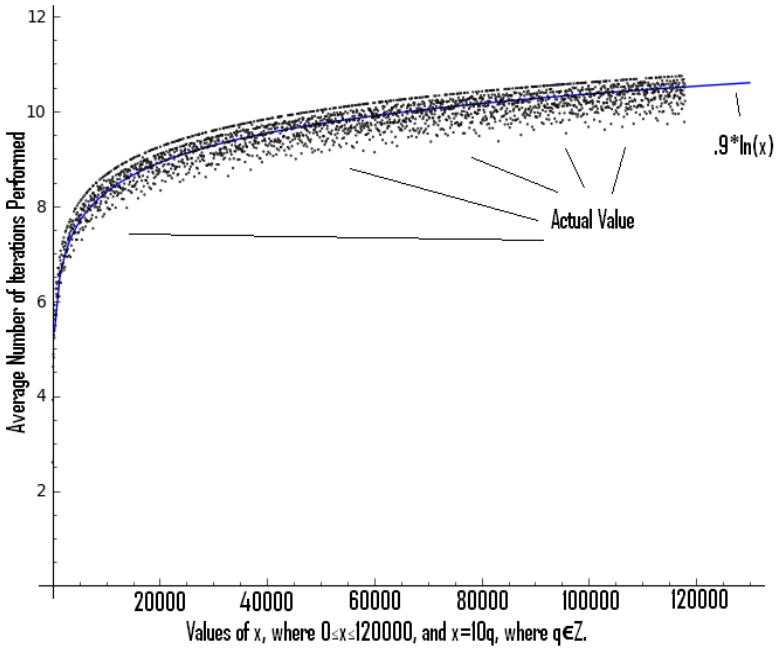

Figure 4.3.1. Average Number of Iterations Performed Using the Euclidean Algorithm over all Integers $y < x$, along with the Estimate $.9 \ln x$.

To estimate the number of steps needed to complete the Euclidean Algorithm, we must also account for the number of steps needed to perform each iteration. This is a simpler value to attain. We know that each iteration of the Euclidean Algorithm has four terms: 1) $a :=$ the number to be expressed; 2) $d :=$ the divisor; 3) $q :=$ the integer quotient of $\frac{a}{d}$; and 4) $r :=$ the integer remainder of the quotient $\frac{a}{d}$. We therefore have $a = dq + r = (a \text{ div } d) \cdot d + a \bmod d$. Then each iteration performs 6 operations. Altogether, the Euclidean Algorithm takes approximately

$$6 \cdot (.9 \ln x) = 5.4 \ln x$$

steps to determine (y, x) and to find two integers r, s such that $yr + xs = (y, x)$.

We proceed to time the $\mathbb{G}_i$ algorithm as seen in Appendix A, while referring to Sieve Method 3.5.1 in Section 3.5 to clarify the process being executed. When possible, we model certain steps in a simplified manner, preferring to time steps per prime rather than per

iteration. That is, at times, it may be easier to time certain steps over k iterations rather than first time per iteration and multiply by k since the latter may not be plausible. (An example of this idea is seen when we estimated the two values $C_i(p_k^2)$ and $E_i(p_k^2)$ by examining the set $C(i,k)$ instead of the j^{th} intervals.)

We first introduce the lines to be timed as seen verbatim in Appendix A, then pair them with their equivalent, simplified lines in Sieve Method 3.5.1. We then give an explanation of what operations are being performed. After this, we proceed to determine the number of steps necessary to accomplish these operations.

To begin, we note that certain lines apply only to $\mathbb{G}_i$ for $i \geq 5$. These steps will be marked by ‡. However, we nevertheless treat these lines as existing in all $\mathbb{G}_i$'s as this does not lead to much of a difference in the timing of the first four. Also note that the algorithm in Appendix A is written so that $\mathbb{G}_i$ terminates after a certain *time* has elapsed. This was done so that our analysis of $\mathbb{G}_i$ in Section 4.1 could be made uniform across time. Sieve Method 3.5.1, on the other hand, is written to terminate after reaching a ceiling. We will time the algorithm as if it terminates upon reaching a ceiling. This should not lead to ambiguity as only one "while" statement is affected. We will assume termination occurs after reaching a ceiling for the sake of analyzing $\mathbb{G}_i$ in terms of the time needed to complete the sifting process.

Lines.

`Pn=1`
`for j in primes:`
`  Pn=Pn*j`
	$P_n = p_1 \cdots p_n$

These three lines evaluate P_n, which is the modulus $\mathbb{G}_i$ works with. Setting `Pn` $= 1$ requires one step. Next, we are multiplying the first n primes to derive the modulo that we are dealing with. Since multiplication is a binary operation, then we have $n-1$ operations being performed.

Total Steps. $1 + n - 1 = n$.

Lines. ‡

`T=[]`
`Range=list(range(1,Pn))`
`for j in Range:`
`  if XGCD(j,Pn)[0]==1:`
`  T.append(j)`
	$\mathcal{T}^{(P_n)} = \{t \vert t < P_n, (t, P_n) = 1\}$

These lines build the set $\mathcal{T}^{(P_n)} = \{t \in \mathbb{N} \vert t < P_n, (t, P_n) = 1\}$. To build it, we must find the gcd between $P_n - 1$ pairs of numbers. We therefore perform the Euclidean Algorithm $P_n - 1$ times, performing approximately $(P_n - 1) \cdot 5.4 \ln P_n$ steps to complete.

Total Steps. $(P_n - 1) \cdot 5.4 \ln P_n$.

Lines. ‡

`P=[]`
`for j in primes:`
`  P.append(j)`
`for j in T:`
`  if j>1 and j<T[1]^2:`
`  P.append(j)`
	$\mathcal{P} = \mathcal{I} \cup \{p \in \mathcal{T}^{(P_n)} \vert p < p_i^2\}$

These lines create the set of primes that we can assume before the process of composite elimination begins. Since we assume the primes less than p_i^2, then we are adding approximately $\frac{p_i^2}{\ln p_i^2}$ elements to P. But we are checking two inequalities for $\phi(P_n)$ elements in T. This adds $2 \cdot \phi(P_n)$ steps.

Total Steps. $\frac{p_i^2}{\ln p_i^2} + 2 \cdot \phi(P_n)$.

Lines.

`S=[]`
 	
`for j in list(range(0,len(T)))):`
`  S.append(list(primes))`
	$\underbrace{\mathcal{S} = \{\mathcal{I}, \mathcal{I}, \ldots\}}_{\phi(P_n) \text{ sets}}$

These lines create the set that will hold the current multiple of relevant primes needed to be eliminated from any given interval. We are adding $\phi(P_n)$ elements to the set `S`, each requiring only one step.

Total Steps. $\phi(P_n)$.

Lines.

`i=len(primes)`	$i = \#\mathcal{I} + 1$
`index=len(primes)`	$j = i$
`SS=list(range(0,len(T)))`	$A = \{a \in \mathbb{Z} \vert 1 \leq a \leq \phi(P_n)\}$

The first two lines set equalities to two parameters: `i` and `index`. Throughout execution, `index` maintains the value that i holds in the beginning, whereas `i` increases by one every iteration and functions as j. These two parameters are in charge of creating the bounds for indexing the multiples of relevant primes. These equalities requires 2 steps. The set `SS` is also used for indexing and is the equivalent of the set A. This set works in conjunction with `i` and `index`. Creating the set `SS` requires $\phi(P_n)$ steps.

Total Steps. $2 + \phi(P_n)$.

Note that the aforementioned lines are executed only once during run-time. The following lines are performed every iteration. Therefore, total steps will account for the $k - i + 1 = k - n$ iterations performed during execution.

Lines.

`G=[]`
`Pisq=(P[i])**2`
`Pi1sq=(P[i+1])**2`
`n=(Pisq)%Pn`
 	
`nn=Pn-n`
`for j in T:`
`  G.append(list(range(`
`  Pisq+((nn+j)%Pn),Pi1sq,Pn)))`
	for t in $\mathcal{T}^{(P_n)}$:
	$\mathcal{G}_j^{(t,P_n)} = \{g \in \mathbb{N} \mid g \equiv t \text{ Pn } P_n,$
	$p_j^2 \leq g < p_{j+1}^2\}$
	$\mathcal{G}_j^{(P_n)} = \bigcup_{t \in \mathcal{T}^{(P_n)}} \mathcal{G}_j^{(t,P_n)}$

These lines create the j^{th} interval. `Pisq` and `Pi1sq` are the bounds p_j^2 and p_{j+1}^2. Determining these values requires 6 steps: 4 to look up the values $P[i]$ and $P[i+1]$ twice each, and 2 to find the squares. Determining `n` requires that we perform `Pisq` mod P_n, which requires 3 steps. The value `nn` needs only 1 step. To determine the number of steps required to build `G`, we first note that there are $\phi(P_n)$ elements being added to `G`, each of which will be a set $\mathcal{G}_j^{(t,P_n)}$. This requires $\phi(P_n)$ steps. Next, for each set we build, we first determine the first term, which is `Pisq+((nn+j)%Pn)`. This takes 5 steps. The algorithm then adds the value $P_n =$ `Pn` until the end point is reach. We consolidate these steps in terms of the whole interval $[p_i^2, p_k^2)$ and say that it takes approximately

$$\phi(P_n)\left(\frac{p_k^2 - p_i^2}{P_n} + k - n\right)$$

steps to complete.

Total Steps.

$$(k-n)(10 + 6 \cdot \phi(P_n)) + \phi(P_n)\left(\frac{p_k^2 - p_i^2}{P_n} + k - n\right).$$

Lines.

`ST=list(range(index,i+1))`
`V=XGCD(P[i],Pn)`
`PnPi=Pn*P[i]`
`V1P1=V[1]*P[i]`
`for j in SS:`
`  S[j].append(`
`  (T[j]*V1P1)%PnPi))`
	$B = \{b \in \mathbb{Z} \mid i \leq b \leq j\}$
	Run Euclidean Algorithm to find $r, s \in \mathbb{Z}$ s.t. $rp_j + sP_n = 1$
	for a in A:
	$\mathcal{S}(a) = \mathcal{S}(a) \cup \{p_j^{(\mathcal{T}^{(P_n)}(a),P_n)}\}$
	where $p_j^{(\mathcal{T}^{(P_n)}(a),P_n)} = trp_j \text{ Pn } p_jP_n$

The set $\texttt{ST} = B$ is used to index the relevant primes in the set `primes`. It is created $k - n$ times during execution, requiring a total of

$$\begin{aligned}\sum_{i=n}^{k} i - n &= \sum_{i=n}^{k} i - \sum_{i=n}^{k} n \\ &= \sum_{i=1}^{k} i - \sum_{i=1}^{n-1} i - n(k-n) \\ &= \frac{k(k+1)}{2} - \frac{(n-1)n}{2} - \frac{2n(k-n)}{2} \\ &= \frac{k^2 + k + n^2 + n - 2nk}{2} \\ &= \frac{(k-n)^2 + k + n}{2}\end{aligned}$$

steps. Note that the parameter `V` is in fact a set. We have $\texttt{V} = \{(\texttt{P[i]}, \texttt{Pn}), r, s\}$, where $r\texttt{P[i]} + s\texttt{Pn} = 1$. We have that $\texttt{V[0]} = (\texttt{P[i]}, \texttt{Pn}) = (p_i, P_n)$, $\texttt{V[1]} = r$, and $\texttt{V[2]} = s$. (Indexing in Python begins at 0). We only concern ourselves with the value $r \in \texttt{V}$ for the sake of solving for the value $p_j^{(\mathcal{T}^{(P_n)}(a), P_n)}$. We must solve for the set `V` for each prime $p_i, \ldots, p_k$. We will assume that $p_i, \ldots, p_k < \texttt{Pn}$ to make timing simpler. Then $(k-n)5.4 \ln P_n$ steps are required to determine `V`, plus $(k - n)$ steps to "look up" `P[i]`. We then assign the value `PnPi=Pn*P[i]`, requiring another $3(k - n)$. After this, we assign the equality to the parameter `V1P1`, which requires 2 steps to look up the values `V[1]` and `P[i]`, plus an additional 2 for the operation of multiplication and assignment, for a total of $4(k - n)$.

We now add the first values of the recurrence sequences, $p_j^{(\mathcal{T}^{(P_n)}(a), P_n)}$, that define the sets $\mathcal{S}_j^{(t, P_n)}$ to the set `S`, which, as stated earlier, holds the "current" multiples of all relevant primes. To determine these values and then add them to the proper set, we must perform $4\phi(P_n)(k - n)$ steps throughout execution.

Total Steps. $\frac{(k-n)^2+k+n}{2} + (k-n)(8 + 4\phi(P_n))$

Lines.

`for j in SS:`
` for k in ST:`
`  PnPk=Pn*P[k]`
`  while S[j][k]<Pi1sq:`
`   if S[j][k] in G[j]:`
`    G[j].remove(S[j][k])`
`    S[j][k]=S[j][k]+PnPk`
`   else:`
`    S[j][k]=S[j][k]+PnPk`
	for $a \in A$:
	  for $b \in B$:
	  if $S(a)(b) \in \mathcal{G}_j^{(P_n)}$:
	  $\mathcal{G}_j^{(P_n)} = \mathcal{G}_j^{(P_n)} \backslash \{\mathcal{S}(a)(b)\}$
	  $\mathcal{S}(a)(b) = \mathcal{S}(a)(b) + \mathcal{P}(a)P_n$
	  else:
	  $\mathcal{S}(a)(b) = \mathcal{S}(a)(b) + \mathcal{P}(a)P_n$

These lines are a bit more complicated to time. We time these lines per prime. We have that 2 steps are needed to determine the value `PnPk=Pn*P[k]`. This occurs $(k-n)\phi(P_n)$ times during run-time. The following lines are responsible for composite eliminations. Here, we over-simplify and rely on our estimates of the values $C_i(p_k^2)$ and $S_i(p_k^2)$. In the context of our algorithm, $S_i(p_k^2)$ is the number of times we perform `S[j][k]=S[j][k]+PnPk`. The line `G[j].remove(S[j][k])` occurs $C_i(p_k^2)$ times since this operation occurs once per distinct composite.

Total Steps. $2\phi(P_n)(k-n) + 4(S_i(p_k^2)) + 2(C_i(p_k^2))$.

Lines.

`G1=[]`	$\mathcal{P}[p_j^2, p_{j+1}^2]$
`for j in SS:`	$\mathcal{P} = \mathcal{P} \cup \mathcal{P}[p_j^2, p_{j+1}^2]$
`  G1=G1+G[j]`	$j = j+1$
`G1.sort()`	
`P=P+G1`	
`i=i+1`	

We now have the primes left in the j^{th} interval. We group the sifted sets together, requiring approximately $\dfrac{p_{j+1}^2}{\ln p_{j+1}^2} - \dfrac{p_j^2}{\ln p_j^2}$ steps each iteration. We can assume that unioning this new set of primes with the old ones requires no steps. We also need 2 steps to perform `i=i+1`.

Total Steps. $\displaystyle\sum_{j=i}^{k} \left(\frac{p_{j+1}^2}{\ln p_{j+1}^2} - \frac{p_j^2}{\ln p_j^2} + 2 \right) = \frac{p_k^2}{\ln p_k^2} - \frac{p_i^2}{\ln p_i^2} + 2(k-n).$

Summing the total steps derived from each part yields the estimate as seen in Conjecture 4.3.1.

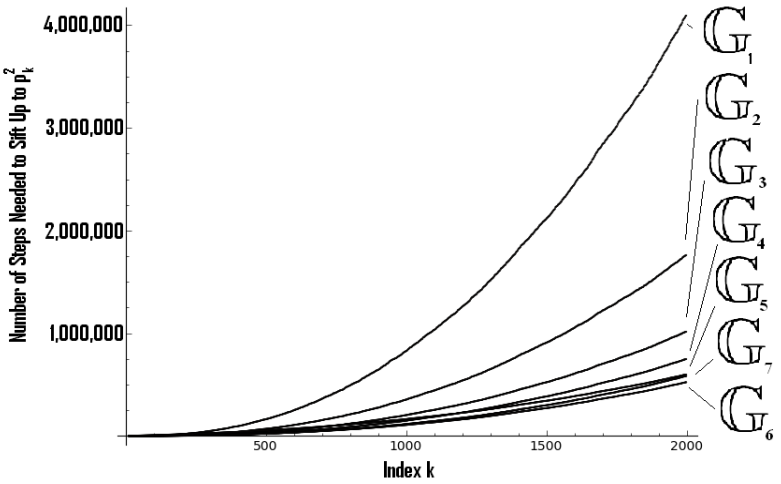

Figure 4.3.2. Steps Needed to Sift up to p_k^2 for $10 \leq k \leq 2000$.

Figure 4.3.2 contains a graph of our approximation of the value $T_i(p_k^2)$. Recall the graph in Figure 4.1.2 that shows the number of primes acquired by the first seven $\mathbb{G}_i$ sieves. Although that graph plots time vs. primes acquired whereas the graph in Figure 4.3.2 plots ceiling vs. steps, we can still compare the two graphs.

There is agreement between the two graphs in that they both show $\mathbb{G}_7$ performing poorly relative to $\mathbb{G}_6$ in the short-term. However, when we plot the same estimate $T_i(p_k^2)$ for higher k, we see something different.

Looking at the graph in Figure 4.3.3, we see that $\mathbb{G}_7$ "catches up" to $\mathbb{G}_6$, surpassing it in terms of prime acquisition after the $30,000^{th}$ iteration. That is, it sifts up to p_{30000}^2

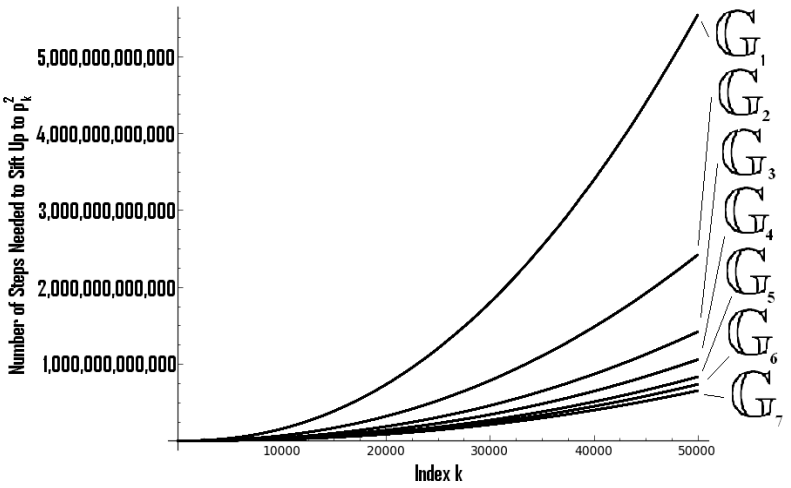

Figure 4.3.3. Steps Needed to Sift up to p_k^2 for $10 \leq k \leq 50000$.

faster than $\mathbb{G}_6$ does. We therefore can conjecture that, given enough time, $\mathbb{G}_i$ for higher i will always perform better than lower $\mathbb{G}_i$'s given enough time.

We can now determine why the $\mathbb{G}_i$ algorithm for higher i have an inferior performance in the short-term. It is because of the fact that the value $\phi(P_n)$ increases incredibly as i increases, thus requiring more preparations not only before execution begins but also before *every* iteration. Nevertheless, the reduction in the number of composites needed to be eliminated coupled with the increase in composite elimination efficiency seem to more than compensate for the extra steps needed for preparation. In essence, $\mathbb{G}_i$ is a marathon runner, not a sprinter.

4.4 The Price of an Increased Efficiency Rate

Up until now, we have investigated the $\mathbb{G}_i$ without considering a major factor in algorithms: not only should the algorithm be efficient in its execution but also in memory. In our analysis of $\mathbb{G}_i$, we have so far left memory unaccounted for, creating a scenario

analogous to the physicist's "vacuum, frictionless" world condusive to modeling.

Since this is not an investigation in computer science, we will not delve into the topic of memory usage too deeply. However, it will suffice to show that the $\mathbb{G}_i$ algorithm, for higher i, is very inefficient in terms of memory usage. Recall that the set `S` holds the current multiples of all relevant primes. Since $\#\texttt{S} = \phi(P_n)$ and, for every element $s \in \texttt{S}$, we have that $\#s = k$, then at any time during execution, `S` holds a total of $k \cdot \phi(P_n)$ numbers in memory. Looking once again at the graph in Figure 4.3.3, we see that at the approximate time when $\mathbb{G}_7$ surpasses the performance of $\mathbb{G}_6$, it is holding more than

$$30,000 \cdot \phi(P_n) = 30,000 \cdot 30,030 = 900,900,000$$

numbers in memory. This is on top of the numbers in the set `G` being sifted ($p_{30000}^2 - p_{29999}^2 = 122,764,042,129 - 122,745,823,201 = 18,218,928$). Needless to say, the memory needed to hold this much is not readily available on a regular home computer.

4.5 Conclusion and Further Discussion

The goal in this project was to resurrect the Sieve of Eratosthenes's concept of sifting in hopes of determining whether or not the process of composite elimination could be made faster. We tried to do this through increase efficiency. As we saw in our analysis, we can make composite elimination much faster; however, this speed came with the price of increased memory usage. This is the reason why we did not empirically run $\mathbb{G}_8$ and higher: because after a very short time, our computer's memory shortage prevented any tangible progress in prime acquisition. Nevertheless, we *were* able to show that we *could* increase prime acquisition with the condition that there is no shortage in memory.

The other aspect of this project examined the counting method based upon the SOE in hopes of making it, too, more efficient. We indeed showed that we could determine $\pi(P_n)$ using less terms. However, the number of terms reduced is seemingly trivial compared to the number of terms left to be evaluated. Nevertheless, we *did* efficientize the counting method.

Appendix A
Python

A.1 $\mathbb{G}_i$ Algorithm Verbatim

The $\mathbb{G}_i$ sieve algorithm, as follows, is the form used during our trial-runs. Lines are included that were responsible for keep track of our analytical data. We present here the algorithm as used.

```
import os
def XGCD(a, b):
     if a == 0 and b == 0:
         return (0,0,1)
     if a == 0:
         return (abs(b), 0, b/abs(b))
     if b == 0:
         return (abs(a), a/abs(a), 0)
     psign = 1
     qsign = 1
     if a < 0:
         a = -a
         psign = -1
     if b < 0:
         b = -b
         qsign = -1
     p = 1; q = 0; r = 0; s = 1
     while b != 0:
         c = a % b
         quot = a/b
```

```
        a = b; b = c
        new_r = p - quot*r
        new_s = q - quot*s
        p = r; q = s
        r = new_r; s = new_s
    return (a, p*psign, q*qsign)
print("Enter set of first n primes.")
primes=(input())
Pn=1
for j in primes:
  Pn=Pn*j
if len(primes)==4 or len(primes)>4:
  T=[]
  Range=list(range(1,Pn))
  for j in Range:
    if XGCD(j,Pn)[0]==1:
      T.append(j)
  P=[]
  for j in primes:
    P. append(j)
  for j in T:
    if j>1 and j<T[1]**2:
      P.append(j)
elif len(primes)==3:
  P=[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47]
  T=[1,7,11,13,17,19,23,29]
elif len(primes)==2:
  P=[2,3,5,7,11,13,17,19,23]
  T=[1,5]
elif len(primes)==1:
  P=[2,3,5,7]
  T=[1]
elif len(primes)==0:
  P=[2,3]
  T=[0]
S=[]
for j in list(range(0,len(T))):
  S.append(list(primes))
R=[]
E=[]
I=[]
i=len(primes)
index=len(primes)
c=0
s=0.0
V=os.times()
print("How long to run G?")
x=(input())
SS=list(range(0,len(T)))
while V[0]<x:
  G=[]
```

```
  Pisq=(P[i])**2
  Pi1sq=(P[i+1])**2
  n=(Pisq)%Pn
  nn=Pn-n
  for j in T:
    G.append(list(range(Pisq+((nn+j)%Pn),Pi1sq,Pn)))
  ST=list(range(index,i+1))
  V=XGCD(P[i],Pn)
  PnPi=Pn*P[i]
  V1P1=V[1]*P[i]
  for j in SS:
    S[j].append((T[j]*V1P1)%(PnPi))
  for j in SS:
    for k in ST:
      PnPk=Pn*P[k]
      while S[j][k]<Pi1sq:
        if S[j][k] in G[j]:
          G[j].remove(S[j][k])
          S[j][k]=S[j][k]+PnPk
          c=c+1
          s=s+1
          continue
        else:
          S[j][k]=S[j][k]+PnPk
          s=s+1
          continue
  G1=[]
  for j in SS:
    G1=G1+G[j]
  G1.sort()
  P=P+G1
  V=os.times()
  i=i+1
  R.append([V[0],len(P)])
  E.append([V[0],float(c/s)])
  I.append([V[0],i-index])
  print(V[0])
f=open('amount','w')
f.write(str(R))
f.close()
f=open('efficiency','w')
f.write(str(E))
f.close()
f=open('iteration','w')
f.write(str(I))
f.close()
f=open('lastprime','w')
f.write(str(P[len(P)-1])+','+str(len(P)))
f.close()
f=open('prime','w')
f.write(str(P))
```

```
f.close()
```

Bibliography

[1] Denis Xavier Charles, *Sieve Methods*, Master's Thesis, University of Buffalo (SUNY), 2000.

[2] Alina Carmen Cojocaru, *An Introduction to Sieve Methods and Their Applications*, Cambridge University Press, Cambridge, 2005.

[3] W.E. Deskins, *Abstract Algebra*, Dover Publications, Inc., New York, 1995.

[4] Erdös, *On a new method in elementary number theory which leads to an elementary proof of the prime number theorem*, Proc. Nat. Acad. Sci. U.S.A. **35** (1949), 374-384.

[5] J Hadamard, *Sur la distribution des zéros de la fonction $\zeta(s)$ et ses conséquences arithmétiques*, Bull. Soc. Math. **24** (1896), 199-200.

[6] C. Hooley Halberstam, *Progress in Analystic Number Theory, Vol. I*, Academin, London, 1981.

[7] Thomas W. Judson, *Abstract Algbebra: Theory and Applications*, Stephen F. Austin State University, Texas, 1997.

[8] Oystein Ore, *Number Theory and Its History*, Dover Publications, Inc., New York, 1948.

[9] C. de la Vallée Poussin, *Recherces analytiques sur la théorie des nombres premiers*, Ann. Soc. Sci. Bruxells (1897).

[10] Kenneth H. Rosen, *Discrete Mathematics and Its Applications*, McGraw Hill Higher Education, Boston, 2007.

[11] M.R. Schroeder, *Number Theory in Science and Communication*, Springer-Verlag, Berlin, 1984.