2016

Automated Conjecturing Approach to the Discrete Riemann Hypothesis

Alexander Bradford
Virginia Commonwealth University, bradforda2@vcu.edu

Automated Conjecturing Approach to the Discrete Riemann Hypothesis

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

by

Alexander Bradford
Master of Science

Director: Craig Larson, Associate Professor
Department of Mathematics and Applied Mathematics

Committee Members:
J. Paul Brooks, Associate Professor
Department of Statistics and Operations Research
Sean Cox, Assistant Professor
Department of Mathematics and Applied Mathematics

Virginia Commonwealth University
Richmond, Virginia
August 2016

Acknowledgments

First, I would like to thank my adviser, Dr. Craig Larson, for all of his guidance, assistance, and patience throughout this project. You pushed me when I did not want to be pushed, and you kept me going when I had difficulties motivating myself. You inspired me to explore the world of mathematics in new and exciting ways, and you reminded me that I have the power and the ability to be successful, as long as I am willing to put in the labor and push myself to always do my best work, no matter the situation.

I would also like to extend an enormous amount of gratitude to my mom, Beth, and my brother, Austin, for their unrelenting support on this journey. Without your love and support, I would not have been able to complete this project. I love you both and cannot thank you enough for what you have done for me, not only for the last 2 years, but the last 24 and counting.

Last, but not least, I would like to thank Dr. Paul Brooks and Dr. Sean Cox for serving on my defense committee. Thanks so much for taking the time out of your schedule to hear, read, and critique my project.

Table of Contents

Abstract

The Möbius function, $\mu(x)$, is defined for all positive integers by $\mu(x) = (-1)^k$ if x is a product of k distinct primes and $\mu(x) = 0$ if the prime factorization of x contains a prime factor to any power greater than 1 (with special case $\mu(1) = 1$). The Mertens function $M(x)$, defined as the sum of the Möbius function of the first x positive integers, is an extraordinary function in the theory of numbers that is closely related to the Riemann Zeta function.

This paper is a study on some upper bounds of the Mertens function, which is often considered somewhat of a "mysterious" function in mathematics. We discuss some known bounds of the Mertens function, and also seek new bounds with the help of an automated conjecture-making program named CONJECTURING, which was created by C. Larson and N. Van Cleemput, and inspired by Fajtowicz's Dalmatian Heuristic. By utilizing this powerful program, we were able to form, validate, and disprove hypotheses regarding the Mertens function and how it is bounded.

1. Introduction

By the Fundamental Theorem of Arithmetic, any positive integer $n > 1$ can be written uniquely in its prime factorization as $n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$, where $p_1, p_2, \dots, p_k$ are primes such that $p_1 < p_2 < \dots < p_k$, and $\alpha_1, \alpha_2, \dots, \alpha_k$ are positive integers. (All terms not specifically defined here can be found in *An Introduction to the Theory of Numbers* [1].) Using this factorization, for all $x \in \mathbb{N}$, the Mertens function $M(x)$ counts the number of square-free positive integers with an even number of prime factors minus the number of square-free positive integers with an odd number of prime factors, up to x. So,

$$M(x) = \sum_{n=1}^{x} \mu(n) \tag{1}$$

where $\mu(n)$ is the Möbius function, again defined over the positive integers,

$$\mu(n) = \begin{cases} 0 & \text{if } n \text{ is not square-free} \\ (-1)^k & \text{if } n \text{ is square-free} \\ 1 & \text{if } n = 1 \end{cases} \tag{2}$$

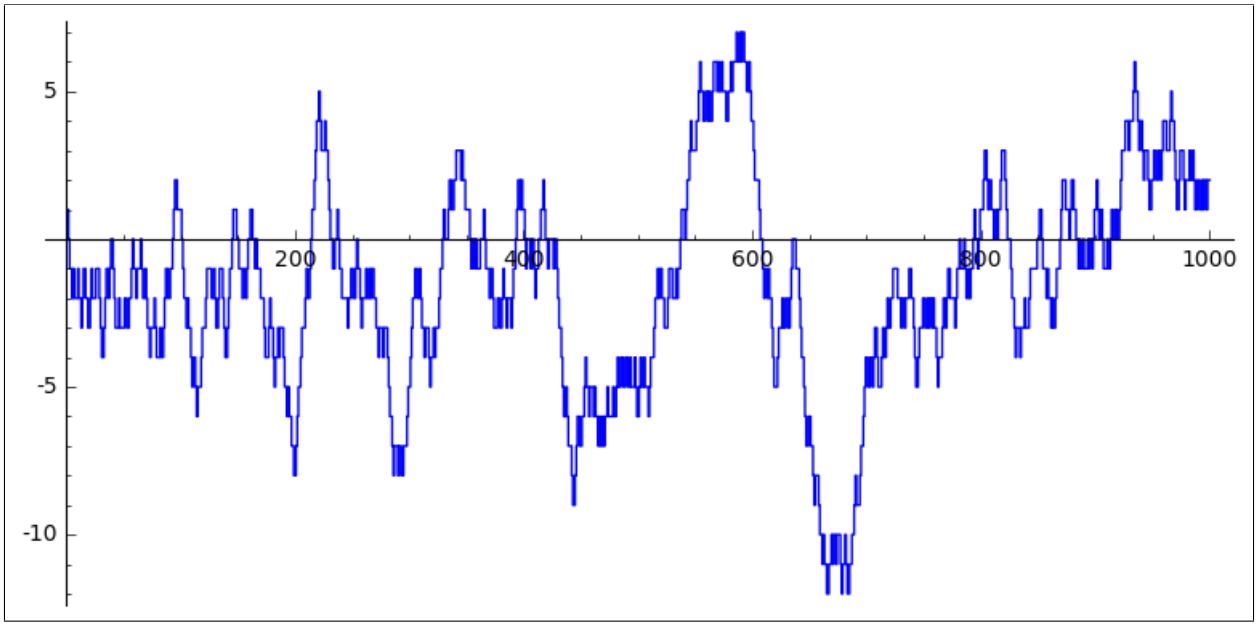
with k being the number of distinct primes in the factorization of n. The following table shows some small values of the Mertens function:

x	Prime Factorization	$\mu(x)$	$M(x)$
1	-	1	1
2	2	-1	0
3	3	-1	-1
4	2^2	0	-1
5	5	-1	-2
6	$2 \cdot 3$	1	-1
7	7	-1	-2
8	2^3	0	-2
9	3^2	0	-2
10	$2 \cdot 5$	1	-1
11	11	-1	-2
12	$2^2 \cdot 3$	0	-2
13	13	-1	-3
14	$2 \cdot 7$	1	-2
15	$3 \cdot 5$	1	-1

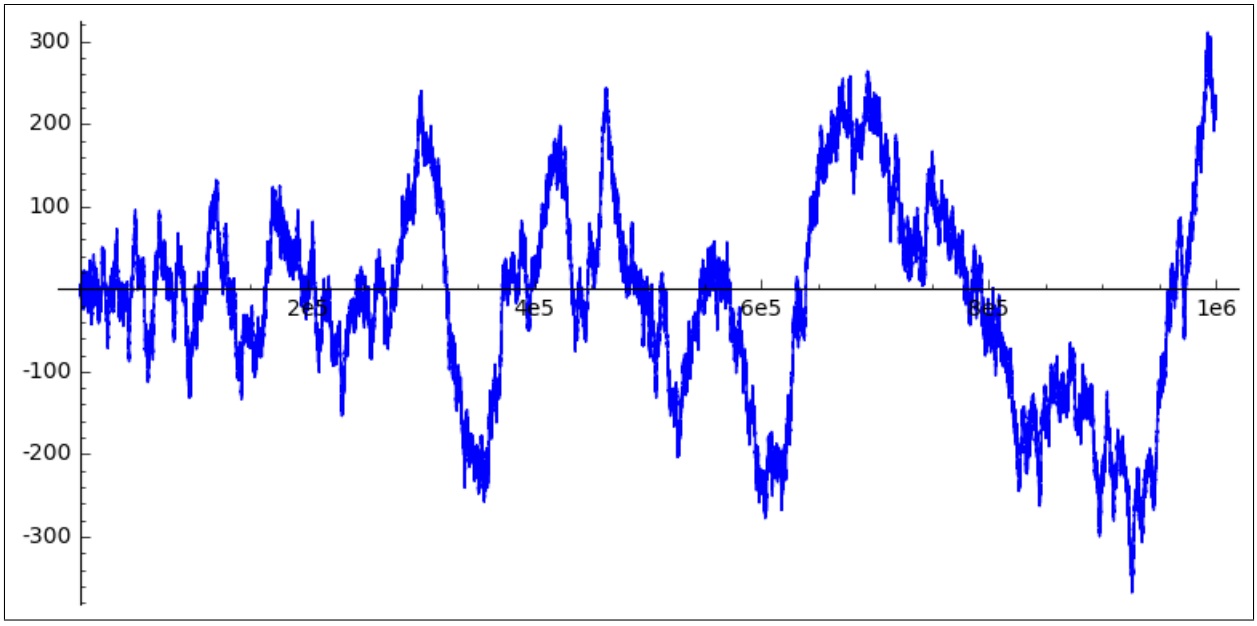
x	Prime Factorization	$\mu(x)$	$M(x)$
16	2^4	0	-1
17	17	-1	-2
18	$2 \cdot 3^2$	0	-2
19	19	-1	-3
20	$2^2 \cdot 5$	0	-3
21	$3 \cdot 7$	1	-2
22	$2 \cdot 11$	1	-1
23	23	-1	-2
24	$2^3 \cdot 3$	0	-2
25	5^2	0	-2
26	$2 \cdot 13$	1	-1
27	3^3	0	-1
28	$2^2 \cdot 7$	0	-1
29	29	-1	-2
30	$2 \cdot 3 \cdot 5$	-1	-3

Table 1: Small values of $M(x)$

Finding relative bounds for $M(x)$ is a very difficult mathematical problem, but it is easy to see that the function grows slowly and what appears to be chaotically in both the positive and negative directions as x increases to infinity. Below, for reference, are two plots of the first 1,000 and 1,000,000 values of the Mertens function, respectively:



Plot 1: $M(x)$, $x \in [1, 10^3]$



Plot 2: $M(x)$, $x \in [1, 10^6]$

2. Background & Motivations

The Mertens Conjecture (1887), from Franz Mertens, was inspired by an 1885 hypothesis from Thomas Stieltjes that $m(x) = M(x) \cdot x^{-\frac{1}{2}}$ is bounded. Mertens strengthened this claim by conjecturing that $m(x)$ is strictly bounded by ± 1. Stieltjes did not prove his claim, but by calculating values of $M(x)$ up to 10,000, Mertens, supposed that his conjecture was likely true for all values of x greater than 1 [4]. This claim is in fact true for all values of $M(x)$ up to 10,000, and in 1912, R.D. von Sterneck extended this result by calculating $M(x)$ up to 500,000 to show that it is true for this range as well. As computational power has increased over the last century, various results have shown that $-1 < m(x) < 1$ is true for larger ranges of x, up to 10^{14} by Tadej Kotnik and Jan van de Lune in 2004 [2].

In 1985, however, Andrew Odlyzko and Herman te Reile showed that $\limsup_{x\to\infty} m(x) > 1.06$ and $\liminf_{x\to\infty} m(x) < -1.009$. Although they did not provide a specific value for which $|m(x)| > 1$, their result implies a theoretical value for which the Mertens Conjecture is false. Even though this result seems to disprove the Mertens Conjecture, Odlyzko and te Reile did suppose that the claim from Steiltjes and Mertens is true for all values of $M(x)$ up to 10^{20} and possibly even 10^{30} [5]. Since the results of Odlyzko and te Reile, the collective work of Kotnik, te Reile, and van de Lune has proved that a counter-example to the Mertens Conjecture can be found between the lower bound of 10^{14} and a best known upper bound of $e^{1.59\times 10^{40}}$ [2]. A specific counter-example to the Mertens Conjecture is still unknown, but in 2006, Kotnik and te Reile showed that there

are infinitely many x for which $m(x) > 1.2184$, providing another proof that the original Mertens Conjecture is false. It is now thought that a more likely conjecture regarding the bounds of $M(x)$ is the following:

$$M(x) = \mathcal{O}(x^{\frac{1}{2}+\epsilon}) \text{ for all } \epsilon > 0 \tag{3}$$

or, for all $x > x_0$ for sufficiently large x_0, and for all $\epsilon > 0$, there exists some constant c such that $M(x) \leq c \cdot (x^{\frac{1}{2}+\epsilon})$. For $m(x)$, this conjecture is equivalent to $m(x) = \mathcal{O}(x^{\epsilon})$ for all $\epsilon > 0$. It turns out that this conjecture, which is referred to as the discrete equivalent of the Riemann Hypothesis, is even more interesting than the Mertens Conjecture. In particular, a proof of (3) is equivalent to showing that the Riemann Hypothesis is true, which claims that the zeros of the Riemann Zeta Function $\zeta(s) = \sum_{i=1}^{\infty} \frac{1}{i^s}$ are exactly the even negative integers as well as the complex numbers with real part $\frac{1}{2}$. This result is monumentally important in mathematics because its proof implies results about the distribution of prime numbers, another largely unsolved and important problem in number theory. Specifically, it would give the following approximation for $\pi(x)$, the number of primes less than or equal to x:

$$\pi(x) = \text{Li}(x) + \mathcal{O}(\sqrt{x} \cdot ln(x)) \tag{4}$$

where $\text{Li}(x) = \int_0^x \frac{dt}{ln(t)}$ is the logarithmic integral function, defined over all $x \in \mathbb{R}^+ \setminus \{1\}$. Thus, a proof of this conjecture, or equivalently a proof of (3), is one of the most famous open problems in mathematics, and it is included in the Millennium Prize Problem set from the Clay Mathematics Institute. It is for these reasons that we are most interested in exploring bounds for the Mertens function.

3. Automated Conjecturing

3.1 Dalmatian Heuristic

A critical tool in our investigation of the Mertens function is the automated conjecture-making program CONJECTURING created by Larson and Van Cleemput. A brief explanation of the program is included in this section, but for a detailed description, see [3].

CONJECTURING is an open-source implementation of Fajtlowicz's Dalmatian heuristic, which can be used to propose relations between real number invariants of mathematical objects (e.g. matrices, graphs, integers, etc.) using algebraic operators (+/-, $\cdot/\div$, $\wedge$). For this project, we use a small, selectively chosen subset of the natural numbers as our object set and a variety of invariants on $\mathbb{N}$, particularly from number theory. A few such invariants are the number $x \in \mathbb{N}$ itself, its value in the Mertens function, $M(x)$, its number of factors, its number of prime factors, its value for Euler's Totient function (counts natural numbers $n < x$ such that $\gcd(n, x) = 1$), $\phi(x)$, and its value for the prime counting function $\pi(x)$. A full list of invariants used in our experiments is available in the Glossary.

The Dalmatian heuristic is powerful because it not only guarantees the quality of the conjectures but also limits the output of conjectures to a reasonable number, effectively outputting just those that are deemed significant. For the program to output a conjecture, the conjecture must pass a two-step verification: the Truth Test and the Significance Test.

The Truth Test checks to see that a proposed conjecture is true for all of the stored values in the objects list. The Significance Test, which is what makes the conjecturing engine particularly powerful, guarantees that a proposed conjecture results in a better bound for at least one stored object than any previous conjecture. These two steps, which Fajtlowicz refers to as the "Principle of the Strongest Conjecture," are summarized by Larson as "output the strongest conjecture for which no counterexample is known." [3]

Therefore, there are two key ways to increase the quality of the conjectures: increasing the size of the invariants list, or increasing the size of the objects list. However, both of these strategies should be chosen selectively, with respect to available computing power and time; the more objects and invariants that are included in each respective list, the more time will be required by CONJECTURING to form new conjectures. In particular, it is most effective to add objects that are counter-examples to previous trials, rather than choosing arbitrary or random objects for the list, in order to guarantee that each additional object will be useful in forming new bounds. Including such counter-examples provides new and significant information to the CONJECTURING since, again, the conjectures output by the program are only necessarily true for the values in the objects list, not all general objects (for our case all $x \in \mathbb{N}$). To demonstrate this process in detail, the first few iterations of our conjecture testing for $M(x)$ are shown below.

3.2 Demonstrating CONJECTURING

In searching for upper bounds for $M(x)$, we first must supply the conjecturing program with some initial objects and some initial invariants. For the initial objects, since we desire bounds for $M(x)$, objects whose Mertens values are easy to compute are preferred. Initially, we arbitrarily choose 5 and 20, for which we can easily compute $M(5) = -2$ and $M(20) = -3$. For the initial invariants, we choose to make conjectures about the main invariant $M(x)$ in terms of 3 other invariants on natural numbers. We call two

of these `number_of_prime_factors` and `number_of_distinct_prime_factors` where the former counts the total number of prime factors of x (given by the sum of the exponents in the prime factorization, $\alpha_1 + \alpha_2 + \ldots + \alpha_k$), and the latter counts only those prime factors of x which are distinct (given by k in the prime factorization). The third invariant mentioned is simply x itself. Since we seek upper bounds for the Mertens function, we designate $M(x)$ as our invariant of interest, and we are able to conjecture upper bounds for $M(x)$. We note here that the number of output conjectures is limited to the number of objects due to the Significance Test in the Dalmatian Heuristic. With the initial objects 5 and 20, the input and output in Sage using the CONJECTURING program appear as the following:

Input:

```
load('~/conjecturing.py')
load('~/mertens2.py')
invariants = [number, mertens_function, number_of_distinct_prime_factors,
              number_of_prime_factors]
objects = [5, 20]
main_invariant = invariants.index(mertens_function)
conjecture(objects, invariants, main_invariant)
```

Output:

```
[mertens_function(x) <= -number_of_prime_factors(x),
 mertens_function(x) <= -number_of_distinct_prime_factors(x) - 1]
```

The first two lines load the `conjecturing.py` program file from Larson and van Cleemput, as well as the `mertens2.py` file which contains the definition of the Mertens function as well as definition functions for the other invariants. The following three lines contain the invariants list, the objects list, and the designation of `mertens_function` as the main

invariant for which we would like to make conjectures. The final line of input contains the function call to the `conjecturing.py` file using the lists described above as its arguments.

We interpret these output conjectures, and all future conjectures, as being supposed bounds for all $x \in \mathbb{N}$. Of course, both of these output conjectures hold true for our initial objects list, which verifies the Truth Test.

For `mertens_function(x)<= -number_of_prime_factors(x)`, which we call Conj. 1 in the table below, the bound is sharp for the object 20, but exact equality does not hold for the object 5, since $M(5) = -2$ and the value of the bound at 5 is -1. However, for `mertens_function(x)<= -number_of_distinct_prime_factors(x)-1` , or Conj. 2 in the table, exact equality holds for both objects; therefore the bound for the object 5 has been tightened, satisfying the Significance Test. Since equality holds for Conj. 2 for both 5 and 20, neither bound can be further improved, and `conjecturing.py` quits searching for new conjectures. This table shows the conjectures evaluated for each of the two initial objects, along with at 2:

x	Prime Factorization	Conj. 1	Conj. 2
5	5	$-2 \leq -1$	$-2 \leq -2$
20	$2^2 \cdot 5$	$-3 \leq -3$	$-3 \leq -3$
2	2	$0 \not\leq -1$	$0 \not\leq -2$

Table 2: First Round Conjectures

Now, it is obvious that these conjectures are not true for all $x \in \mathbb{N}$. In particular, $M(2) = 0$, so 2 is a counter-example to both of these initial conjectures. By now adding 2 to the list of stored objects, we will get new conjectures that are not only guaranteed true for the values of 5 and 20, but 2 as well. As Sage input and output, this appears as the following:

Input:

```
# 2 is a counter-example to both conjectures
invariants = [number, mertens_function,
              number_of_distinct_prime_factors, number_of_prime_factors]
objects = [5, 20, 2]
```

Conjectures:

```
[mertens_function(x) <= number_of_distinct_prime_factors(x) - 1,
 mertens_function(x) <= (-number_of_distinct_prime_factors(x))^number(x)
                         - 1,
 mertens_function(x) <= -number_of_distinct_prime_factors(x)^2 + 1]
```

We note that each of the previous conjectures for which we provided a counter-example is no longer included in the output list, and again, it is easy to see that each of these new conjectures is true for our input objects. Once more, these conjectures are not necessarily true for all values of x. To illustrate the Significance Test for these conjectures, we have the following table, again numbering the conjectures in the order of the list output:

x	Prime Factorization	Conj. 1	Conj. 2	Conj. 3
5	5	$-2 \leq 0$	$-2 \leq -2$	$-2 \leq 0$
20	$2^2 \cdot 5$	$-3 \leq 1$	$-3 \leq 1048575$	$-3 \leq -3$
2	2	$0 \leq 0$	$0 \leq 0$	$0 \leq 0$

Table 3: Second Round Conjectures

As we can see, Conj. 1 gives a sharp bound for only the object 2; however, this is enough to deem this conjecture significant per the Significance Test. For Conj. 2, again we get a sharp bound for 2, but for this conjecture there is exact equality for the object 5 as well.

Even though the bound for the object 20 is much weaker and less useful for Conj. 2 than Conj. 1, the improvement of the bound on 5 is enough to deem Conj. 2 significant as well. Finally, for Conj. 3, the bound on 20 is improved to exact equality, and therefore it is significant with respect to 20. Now, we have at least 1 sharp bound for each of the objects, and CONJECTURING halts and outputs these 3 conjectures.

We proceed in this manner, disproving at least one conjecture from each output conjecture list, and adding the appropriate counter-example to the objects list. These trials are documented from section 1.1 to 1.10 in the Index.

4. Methods

4.1 `theory` Bounds

Another powerful component of CONJECTURING is the `theory` argument. By including a bound in the `theory` list, we require that CONJECTURING only checks for upper bounds that are better than the `theory` bound, so in our case, bounds which are strictly less than the `theory` bound(s) for at least one object in the `objects` list. This allows us to improve the quality of conjectures not only by disproving output conjectures and adding the counter-examples to the `objects` list, but also by proving output conjectures and adding these bounds to the `theory` list. We note here that it also may be useful to include published bounds which are known to be true in this `theory` list. This would guarantee that all conjectures from the program that can be proved are improvements over existing publications.

For conjecturing rounds 1.6 through 1.10, the following conjecture is among the output list:

```
mertens_function(x) <= number(x)
```

It is obvious that this conjecture is true for all values of x, so we formally prove this conjecture and introduce our first `theory` bound, `number`(x) (the identity function), in round 1.11 of our CONJECTURING trials.

Theorem 1. *For all* $x \in \mathbb{N}$, `mertens_function(x) <= number(x)`.

Proof. By definition, $\mu(n) \in \{1, -1, 0\}\ \forall n \in \mathbb{N}$, so $\mu(n) \leq 1$. Then, $M(x) = \sum_{n=1}^{x} \mu(n) = \mu(1) + \mu(2) + \ldots + \mu(x) \leq x \cdot 1 = x$. □

Now that this conjecture is validated, we change our input for trial 1.11 to the following:

Input:

```
invariants = [number, mertens_function, number_of_distinct_prime_factors,
              number_of_prime_factors, euler_phi_function]
theory = [number]
objects = [5, 20, 19, 2, 97, 999983, 1, 12, 6, 1000000]
conjecture(objects, invariants, main_invariant, theory=theory)
```

Here, we let the list `theory` contain just the `number` invariant and also include this information in the function call of `conjecturing.py` by setting the `theory` argument to be the list which we called `theory`. Although it may not be immediately obvious, we can check that the output conjectures are all bounds that are strictly sharper than `number` for some value in the objects list. The chart on the following page demonstrates this requirement for each of the conjectures in our trial round 1.11:

	Conjectured Bound	Example Object, x	Conj. Bound at x vs. `number(x)`
1.	`euler_phi_function(x)`	20	$8 < 20$
2.	`4*(log(2*number(x) - 2)/log(10) + 1)^2`	97	$\sim 43 < 97$
3.	`euler_phi_function(x)` `- number_of_distinct_prime_factors(x)`	2	$0 < 2$

4.	`maximum(`
	`number_of_distinct_prime_factors(x),`
	`(log(number(x))/log(10) - 1)^4)`
		5	$1 < 5$

5.	`(-sqrt(euler_phi_function(x)))`
	`^number_of_prime_factors(x)`
	`+ number(x)`
		5	$3 < 5$

6.	`2*sqrt(number(x))` `- 2*log(euler_phi_function(x)^2)`	5	$\sim -1 < 5$

7.	`euler_phi_function(x)`
	`/number_of_distinct_prime_factors(x)`
	`- number_of_prime_factors(x)`
		97	$95 < 97$

8.	`-4*sqrt(euler_phi_function(x) - 1)` `+ number(x)`	5	$\sim -2 < 5$

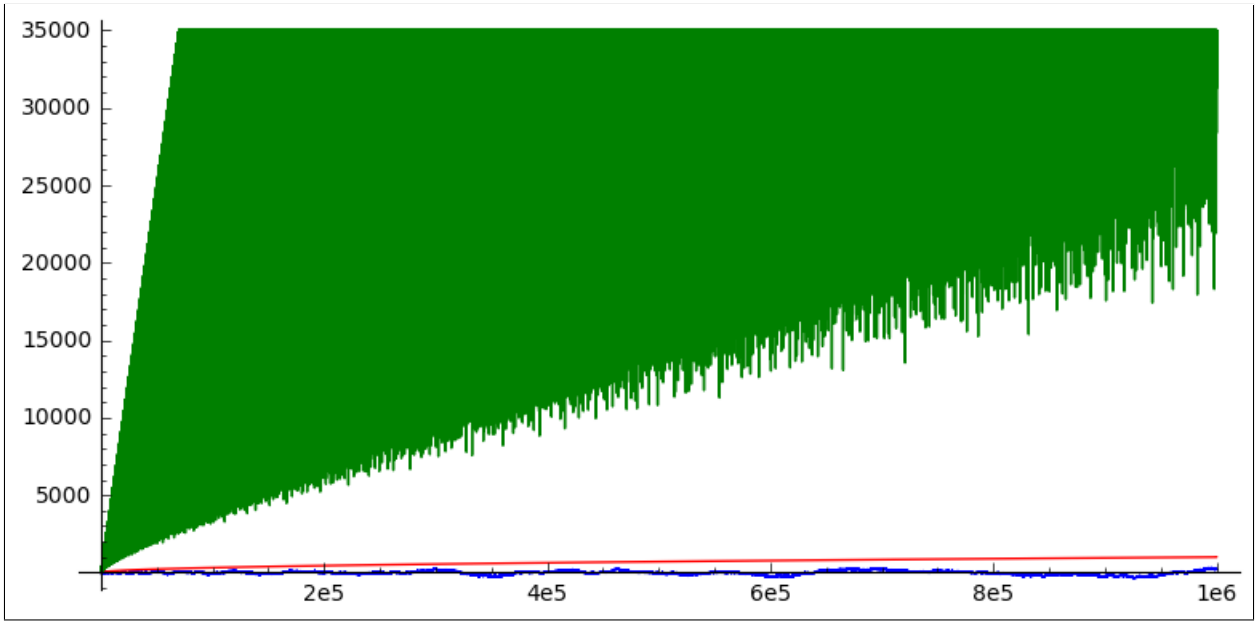
9.	`(sqrt(euler_phi_function(x)) + 1)`
	`/(number_of_distinct_prime_factors(x)`
	`+ 1) + 1`
		5	$2.5 < 5$

10.	`-number_of_prime_factors(x)^2` `+ 1/2*number(x)+ 1`	1000000	$499857 < 1000000$

Table 4: Demonstrating `theory` Bound `number(x)`

We continue to use this `theory` bound for each of the remaining CONJECTURING trials in Section 1 of the Index, until a better bound is realized.

For all other CONJECTURING trials (Sections 2 through 4), we replace `number` with `divisor_mean` as the `theory` bound, since this new bound is tighter for all natural numbers (note: we also could simply add to the `theory` list so that it contains multiple bounds). Below is a plot that compares `divisor_mean(x)` with $\sqrt{x}$ and $M(x)$: As we



Plot 3: `divisor_mean(x)` [green], $\sqrt{x}$ [red], $M(x)$ [blue], for $x \in [1, 10^6]$. Note: `divisor_mean(x)` exceeds 35,000 for many values of x, but the maximum value of the vertical axis is set to 35,000 to provide a more illuminating picture of the relationship between this function and $M(x)$.

can see, `divisor_mean(x)` (green) grows much more quickly than $M(x)$ (blue) and also $\sqrt{x}$ (red). Although we were unable to prove that `divisor_mean(x)` bounds $M(x)$ for all x, this bound is thought to be true, and it can be included in the `theory` list regardless in an attempt obtain even better bounds. Using the Arithmetic Mean-Geometric Mean theorem, however, it *can* be shown that `divisor_mean(x)` bounds $\sqrt{x}$, the supposed bound from Merten's Conjecture:

Lemma 1. *For any* $x_1, x_2 \in \mathbb{N}$, $\frac{x_1+x_2}{2} \geq \sqrt{x_1 x_2}$.

Proof. Let $x_1, x_2 \in \mathbb{N}$. Then,

$$
\begin{aligned}
(x_1 - x_2)^2 \geq 0 \quad &\Leftrightarrow \quad {x_1}^2 - 2x_1x_2 + {x_2}^2 \geq 0 \\
&\Leftrightarrow \quad {x_1}^2 + {x_2}^2 \geq 2x_1x_2 \\
&\Leftrightarrow \quad {x_1}^2 + 2x_1x_2 + {x_2}^2 \geq 4x_1x_2 \\
&\Leftrightarrow \quad \tfrac{{x_1}^2+2x_1x_2+{x_2}^2}{4} \geq x_1x_2 \\
&\Leftrightarrow \quad (\tfrac{x_1+x_2}{2})^2 \geq x_1x_2 \\
&\Leftrightarrow \quad \tfrac{x_1+x_2}{2} \geq \sqrt{x_1x_2}
\end{aligned}
$$

□

Theorem 2. *For all* $x \in \mathbb{N}$, `divisor_mean(x)` $\geq \sqrt{x}$.

Proof.

Case 1. *Assume* x *is not a perfect square, so* x *has* $2n$ *divisors* $d_1, d_2, \ldots, d_{2n}$*, with* $1 = d_1 < d_2 < \ldots < d_{2n} = x$ *such that* $d_1d_{2n} = d_2d_{2n-1} = \ldots = d_nd_{n+1} = x$*. By Lemma 1, for any pair* d_i, d_{2n+1-i}*, with* $1 \leq i \leq n$*, we know* $\frac{d_i+d_{2n+1-i}}{2} \geq \sqrt{d_id_{2n+1-i}} = \sqrt{x}$*. So,* $\frac{d_1+d_{2n}}{2} + \frac{d_2+d_{2n-1}}{2} + \ldots + \frac{d_n+d_{n+1}}{2} \geq n\sqrt{x} \Leftrightarrow \frac{d_1+d_{2n}}{2n} + \frac{d_2+d_{2n-1}}{2n} + \ldots + \frac{d_n+d_{n+1}}{2n} \geq \sqrt{x} \Leftrightarrow \frac{d_1+d_2+\ldots+d_{2n}}{2n} \geq \sqrt{x}$*.*

Case 2. *Now, assume* x *is a perfect square, so* x *has* $2n+1$ *divisors* $d_1, d_2, \ldots, d_{2n+1}$*, with* $1 = d_1 < d_2 < \ldots < d_{2n+1} = x$ *such that* $d_1d_{2n+1} = d_2d_{2n} = \ldots = d_nd_{n+2} = {d_{n+1}}^2 = x$*. By Lemma 1, for any pair* d_i, d_{2n+2-i}*, with* $1 \leq i \leq n$*, we know* $\frac{d_i+d_{2n+2-i}}{2} \geq \sqrt{d_id_{2n+2-i}} = \sqrt{x} = d_{n+1}$*. So,* $\frac{d_1+d_{2n+1}}{2} + \frac{d_2+d_{2n}}{2} + \ldots + \frac{d_n+d_{n+2}}{2} \geq n\sqrt{x} = n \cdot d_{n+1} \Leftrightarrow d_1 + d_2 + \ldots + d_n + d_{n+2} + \ldots + d_{2n+1} \geq 2n \cdot d_{n+1} \Leftrightarrow \frac{d_1+d_2+\ldots+d_n+d_{n+2}+\ldots+d_{2n+1}}{2n+1} \geq \frac{2n \cdot d_{n+1}}{2n+1} = \frac{(2n+1)d_{n+1}}{2n+1} - \frac{d_{n+1}}{2n+1} \Leftrightarrow \frac{d_1+d_2+\ldots+d_n+d_{n+2}+\ldots+d_{2n+1}}{2n+1} + \frac{d_{n+1}}{2n+1} \geq d_{n+1} = \sqrt{x} \Leftrightarrow \frac{d_1+d_2+\ldots+d_n+d_{n+1}+d_{n+2}+\ldots+d_{2n+1}}{2n+1} \geq \sqrt{x}$*.*

Therefore, `divisor_mean(x)` $\geq \sqrt{x}$. □

We note here that even though it was never included as a `theory` bound in our trials, we also proved the following conjecture:

Theorem 3. *For all* $x \in \mathbb{N}$, `mertens_function(x)<= (number(x)-3)^2-1`.

Proof. By Table 1 and brute force, we can see that for $x \leq 5$, this conjecture is true. Then, for $x \geq 6$, this conjecture is simply a consequence of Theorem 1. □

4.2 `special` Functions

As previously mentioned, computing time is a significant concern when using CONJECTURING and working with the Mertens Function. For most of our CONJECTURING trials (1.1-4.1), we were only able to test the truth of output conjectures for values of x up to 10,000 due to the excessive time required to calculate $M(x)$ (`mertens_function(x)`), as well as other functions such as $\pi(x)$ (`pi_function(x)`). In order to increase our testable range, we introduce what we call `special` functions in trial round 4.2. For each of these functions, we store a list of the computed values, rather than recomputing all values of x each time the function is called. The definition of `mertens_special(x)`, the `special` equivalent of `mertens_function(x)`, is:

```
def mertens_special(n):
    current = len(mertens_data)
    if n <= current:
        return mertens_data[n]
    for i in range(current,n+1):
        mertens_data.append(mertens_data[i-1]+moebius(i))
    save(mertens_data,'mertens_data.sobj')
    return mertens_data[n]
```

The `mertens_data` file is a list, saved as a Sage Object, which holds all previously computed values of `mertens_special(x)`. If a value which is already in the `mertens_data` list is required, the `mertens_special` function simply pulls from the list at the appropriate x. If a value which is not in the `mertens_data` list is required, `mertens_special` computes all unknown values up to the desired x and adds these to the `mertens_data` list so they also can be pulled from the list the next time the `mertens_special` function is called. All other `special` functions follow this form, and are substituted in CONJECTURING trial 4.2 for their appropriate counterparts. The table below contains a list of the `special`

functions we used to facilitate larger computations of x, up to 1,000,000:

	Invariant	special Equivalent	data List
1.	mertens_function	mertens_special	mertens_data
2.	divisor_mean	divisor_mean_special	divisor_mean_data
3.	count_divisors	count_divisors_special	count_divisors_data
4.	sum_divisors	sum_divisors_special	sum_divisors_data
5.	euler_phi_function	euler_phi_special	euler_phi_data
6.	pi_function	pi_function_special	pi_function_data
7.	upper_prime _remainder	upper_prime _remainder_special	upper_prime _remainder_data
8.	upper_prime	upper_prime_special	upper_prime_data
9.	upper_prime _adjusted	upper_prime _adjusted_special	upper_prime _adjusted_data
10.	sum_nontrivial _divisors	sum_nontrivial _divisors_special	sum_nontrivial _divisors_data

Table 5: List of special Functions

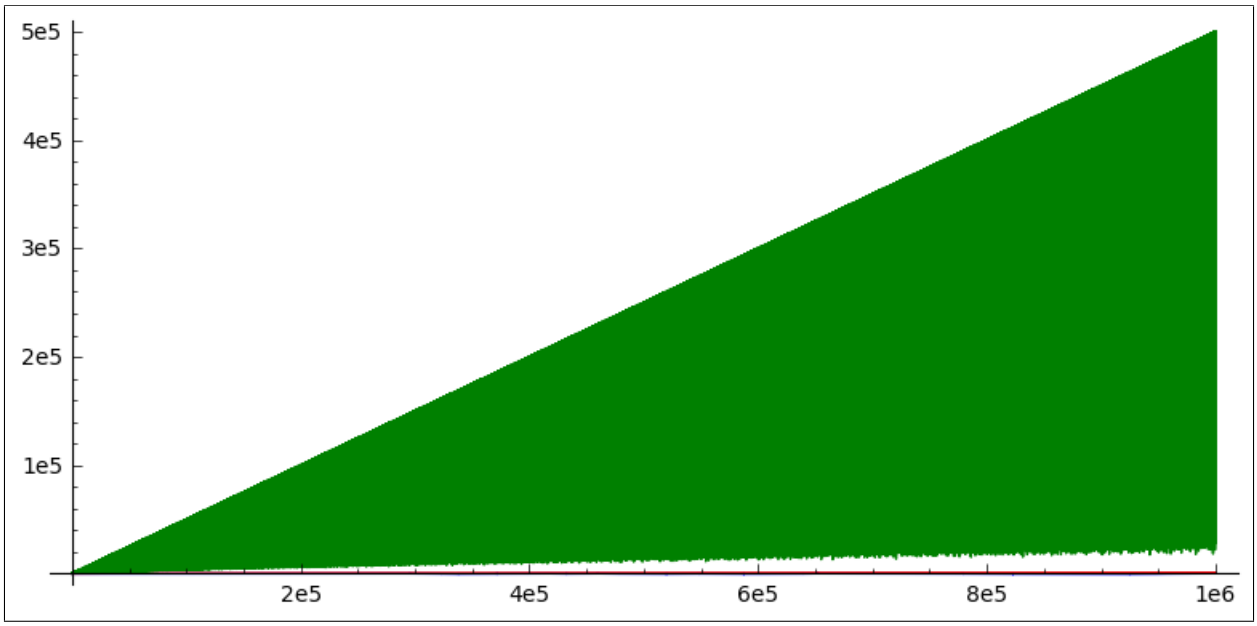
5. Notable Conjectures

Throughout our research of the Mertens function, we have discovered quite a few interesting conjectures. Although the randomness of the prime factorization of the elements of $\mathbb{N}$ - and in particular the randomness of $\pi(x)$ - makes these conjectures very difficult to prove, the conjectures themselves are quite important. As previously mentioned, adding to the known theory of the Mertens function could lead to a proof (or disproof) of the Riemann Hypothesis. In this section, we list some of these conjectures which remain unproved yet are believed to likely be true and/or particularly significant. All conjectures in this section have been tested by brute force to bound $M(x)$ for at least all values of $x \leq 1000000$.

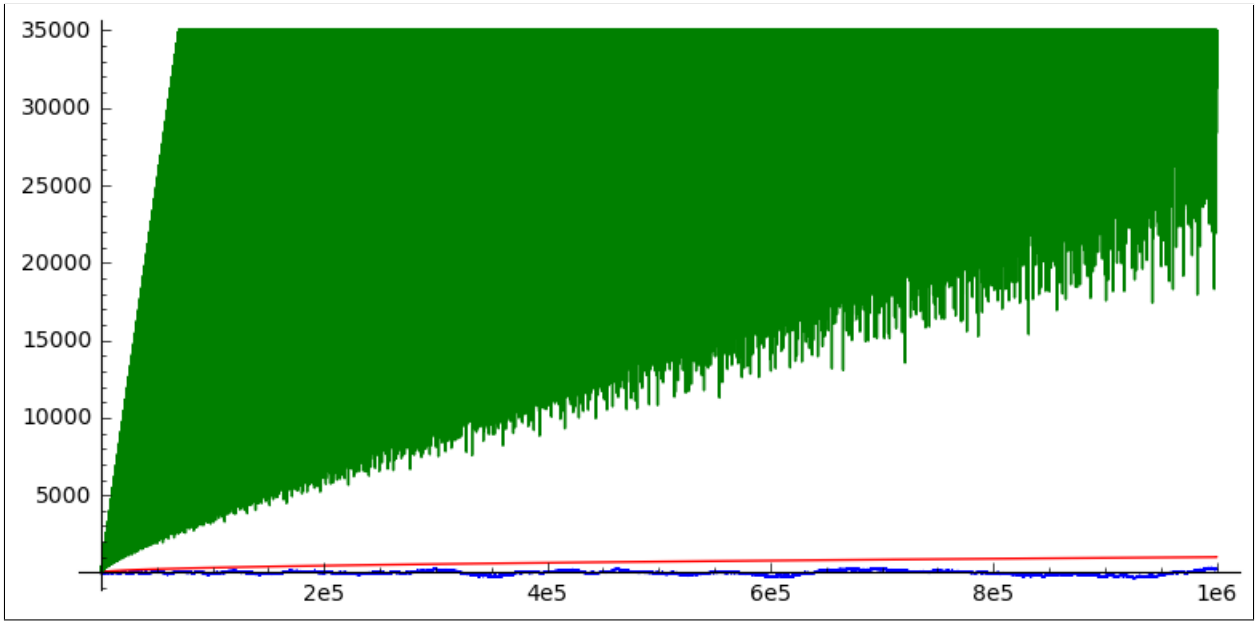
	Upper Bound
1.	`divisor_mean(x)`
2.	`euler_phi_function(x)`
3.	`pi_function(x)`
4.	`euler_phi_function(x)-pi_function(x)`
5.	`divisor_mean(x)/digits2(x) - number_squarefull_pf(x)`
6.	`(pi_function(x) + 1)/number_of_distinct_prime_factors(x)^2`

Table 6: Significant & Interesting Conjectures

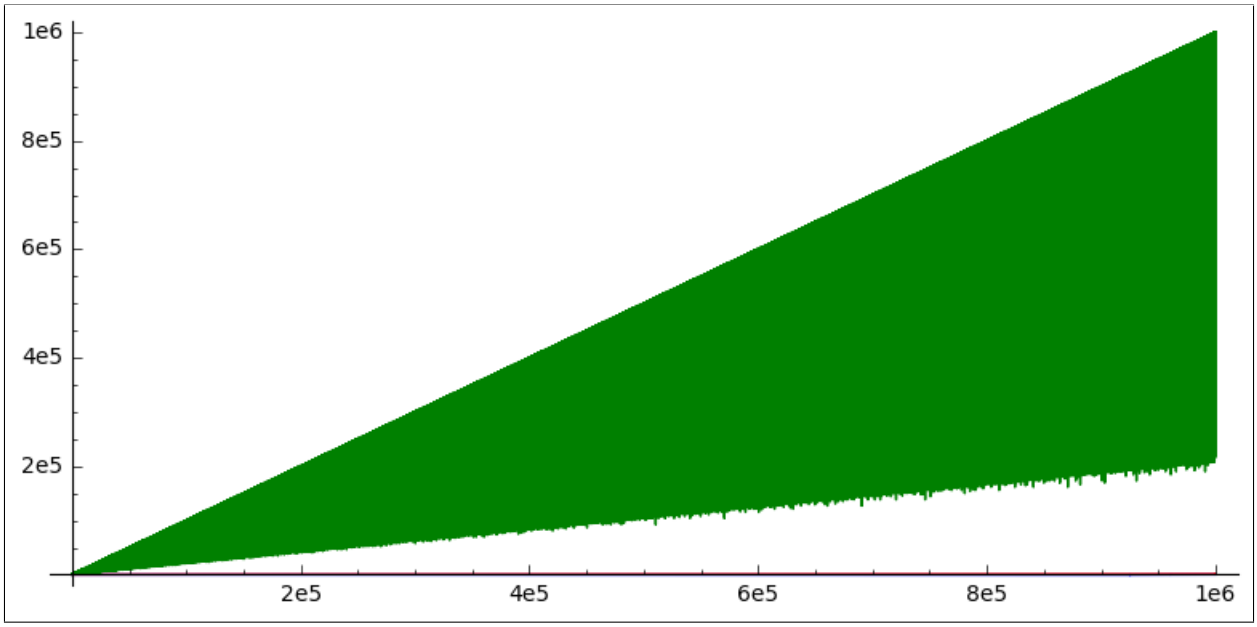
Below are a number of plots of the Significant & Interesting Conjectures. All conjectured bounds are shown in green, and are plotted along with $M(x)$ in blue and $\sqrt{x}$ in red. Plots marked with ** in the caption have truncated vertical axes to better show detail of the relationship between the conjecture and $M(x)$.



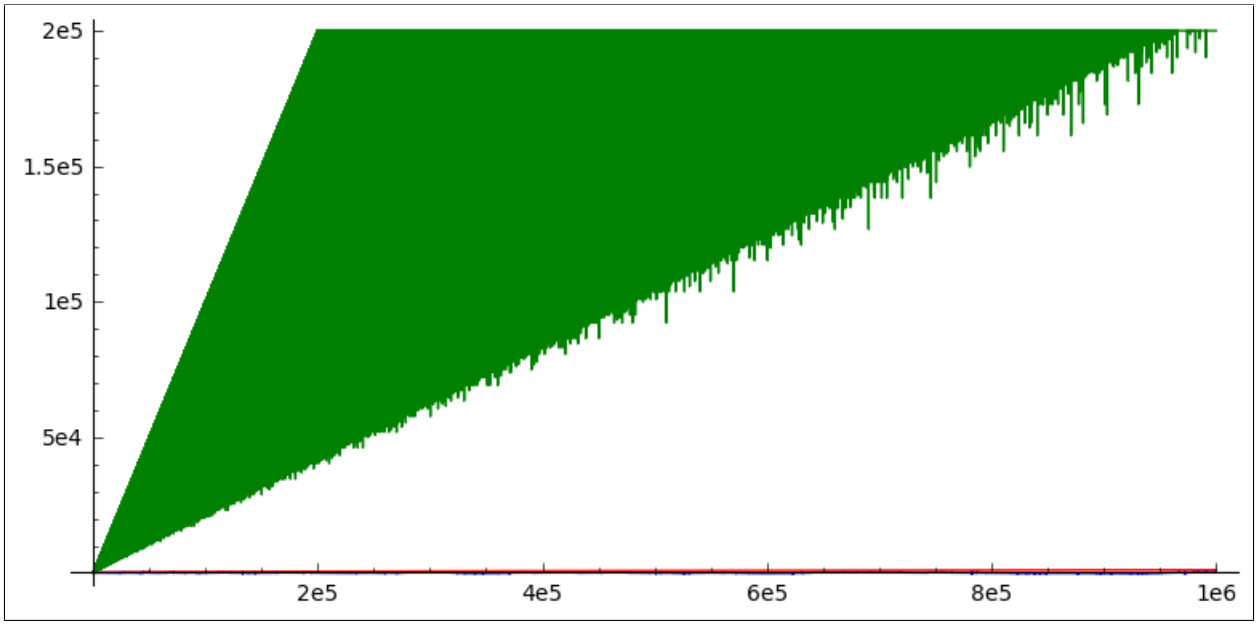
Plot 4: `divisor_mean(x)` [green], $\sqrt{x}$ [red], $M(x)$ [blue], for $x \in [1, 10^6]$.



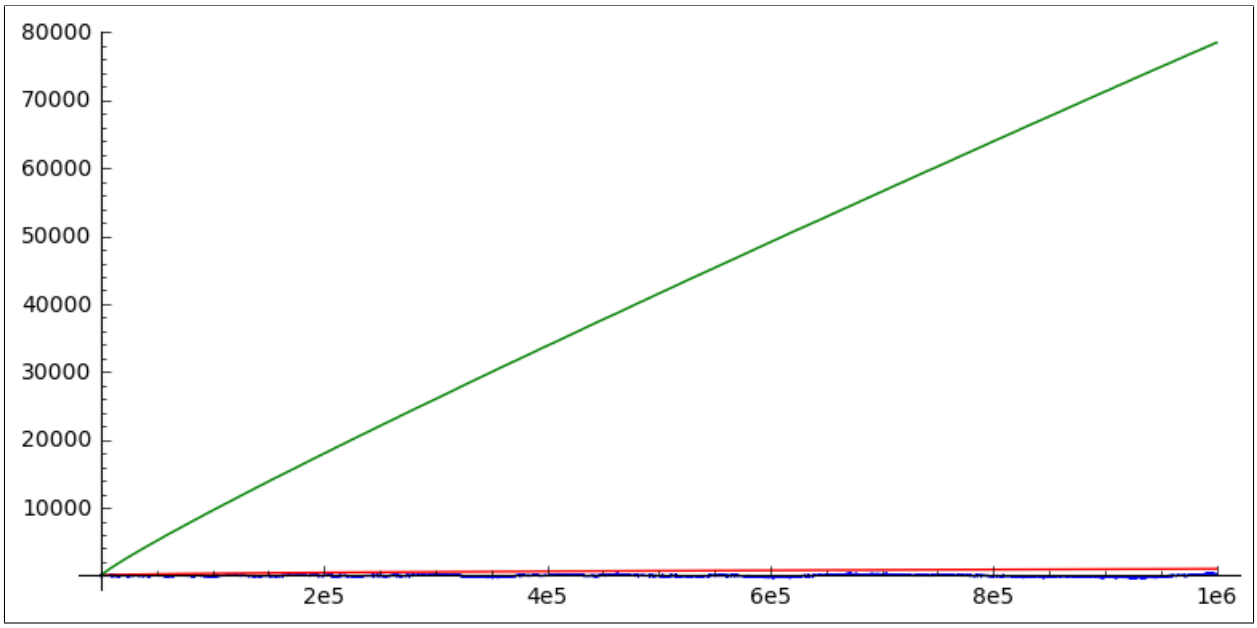
Plot 5**: `divisor_mean(x)` [green], $\sqrt{x}$ [red], $M(x)$ [blue], for $x \in [1, 10^6]$. Note: `divisor_mean(x)` exceeds 35,000 for many values of x, but the maximum value of the vertical axis is set to 35,000 to better show the relationship between this function and $M(x)$.



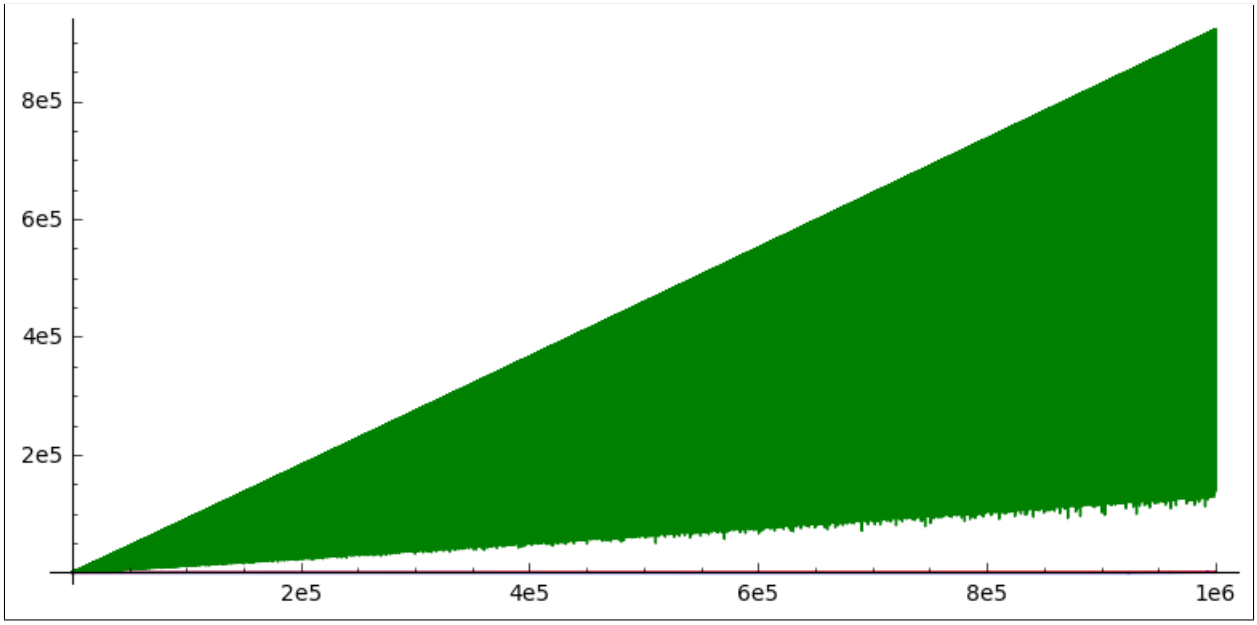
Plot 6: `euler_phi_function`(x) [green], $\sqrt{x}$ [red], $M(x)$ [blue], for $x \in [1, 10^6]$.



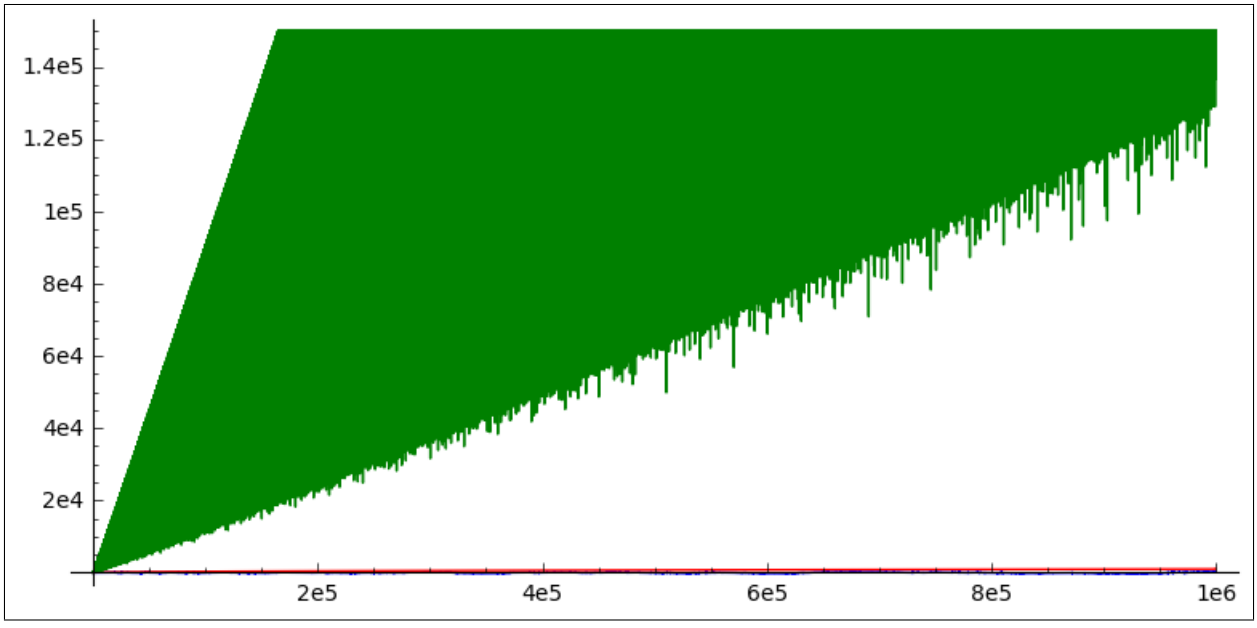
Plot 7**: `euler_phi_function`(x) [green], $\sqrt{x}$ [red], $M(x)$ [blue], for $x \in [1, 10^6]$. Note: `euler_phi_function`(x) exceeds 200,000 for many values of x, but the maximum value of the vertical axis is set to 200,000 to better show the relationship between this function and $M(x)$.



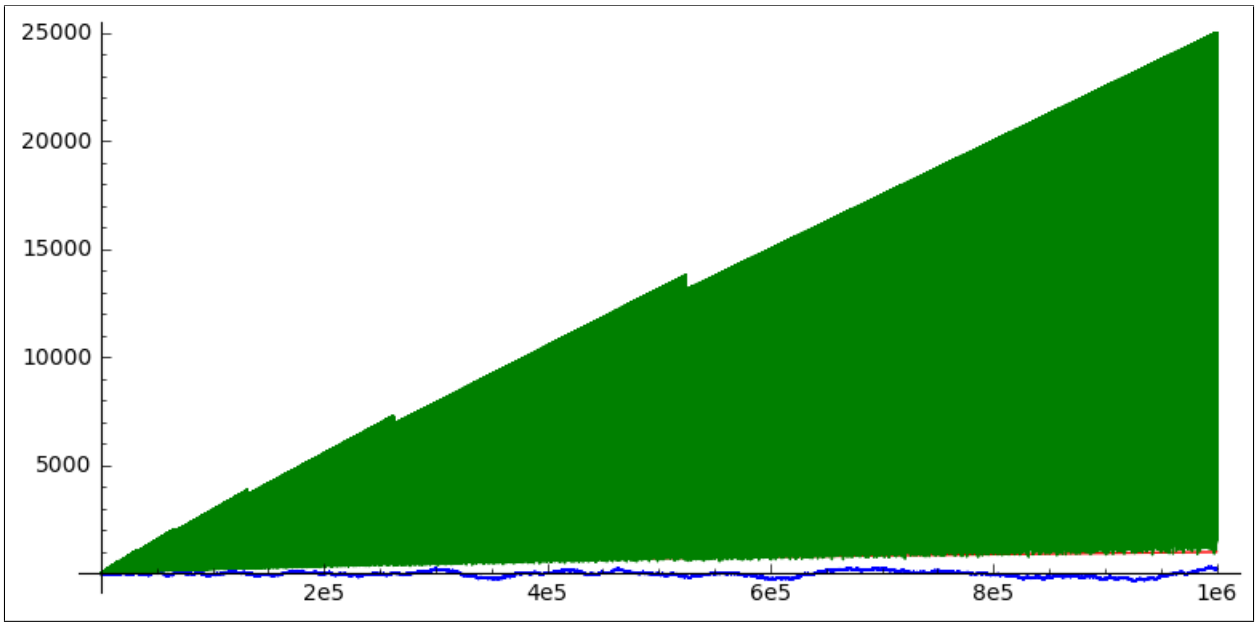
Plot 8: `prime_pi_function`(x) [green], $\sqrt{x}$ [red], $M(x)$ [blue], for $x \in [1, 10^6]$.



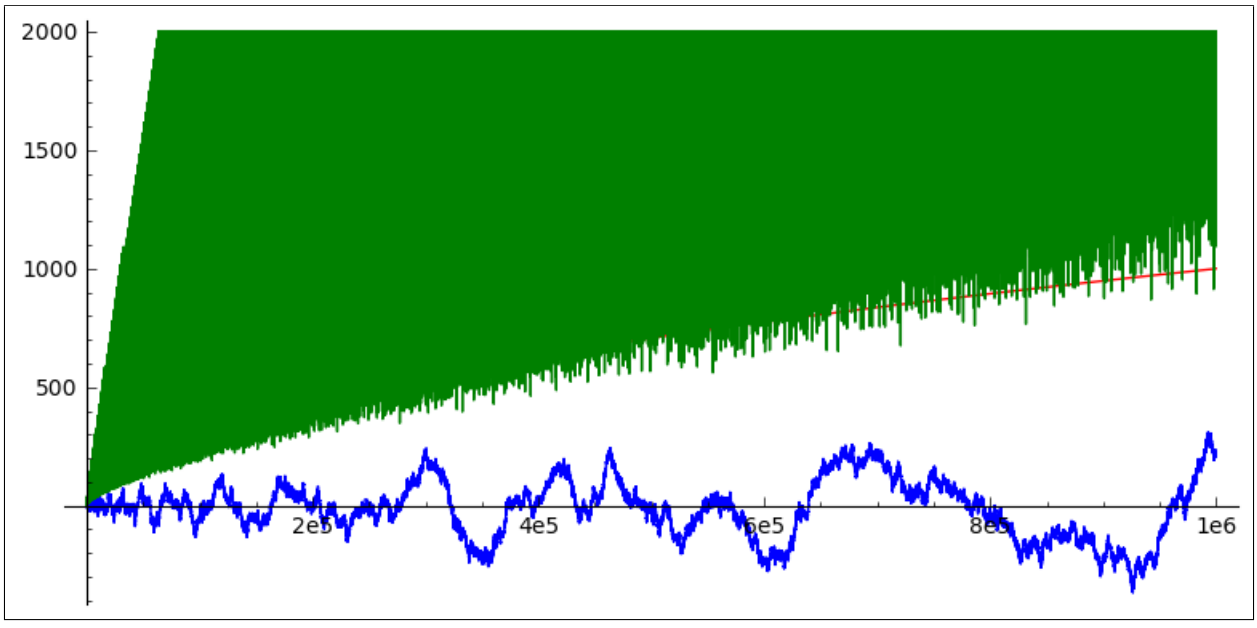
Plot 9: `euler_phi_function`$(x)-$ `prime_pi_function`(x) [green], $\sqrt{x}$ [red], $M(x)$ [blue], for $x \in [1, 10^6]$.



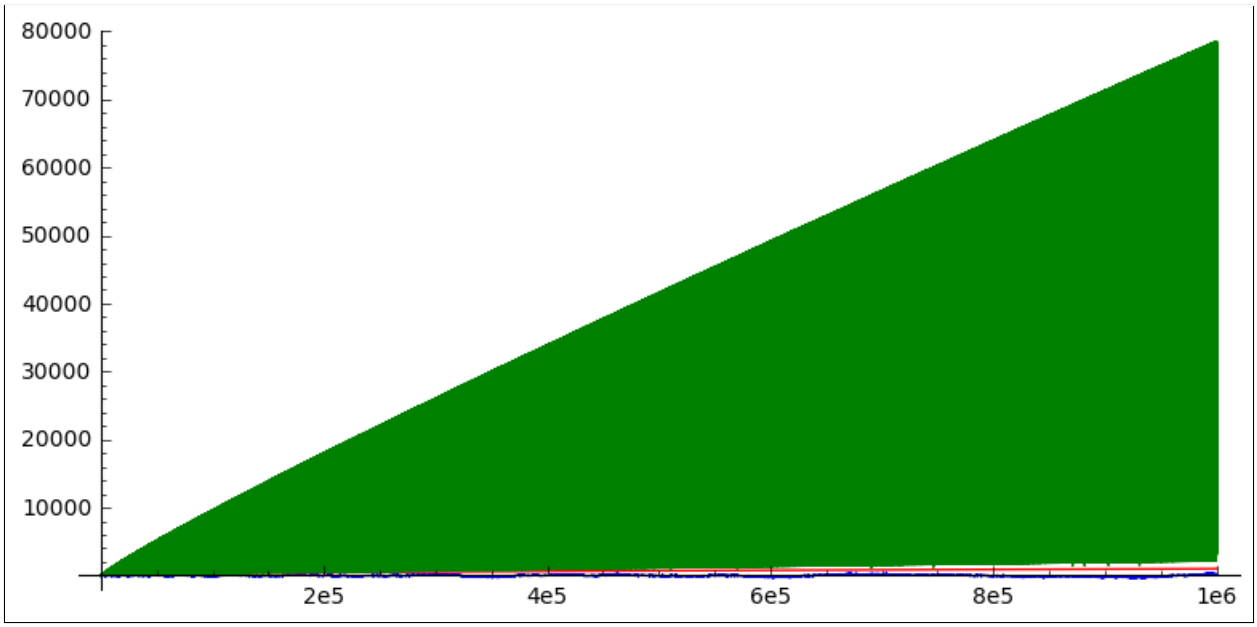
Plot 10**: `euler_phi_function`$(x)-$ `prime_pi_function`(x) [green], $\sqrt{x}$ [red], $M(x)$ [blue], for $x \in [1, 10^6]$. Note: `euler_phi_function`$(x)-$ `prime_pi_function`(x) exceeds 150,000 for many values of x, but the maximum value of the vertical axis is set to 150,000 to better show the relationship between this function and $M(x)$.



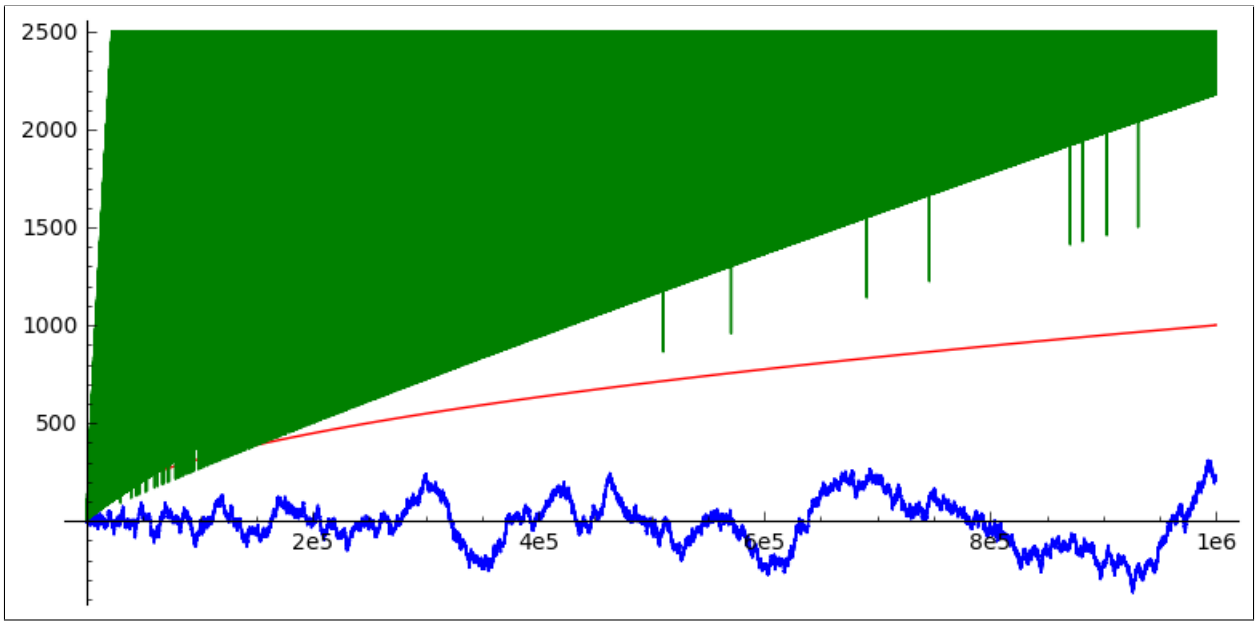
Plot 11: `divisor_mean(x)/digits2`$(x)-$`number_squarefull_pf`(x) [green], $\sqrt{x}$ [red], $M(x)$ [blue], for $x \in [1, 10^6]$.



Plot 12**: `divisor_mean(x)/digits2`$(x)-$ `number_squarefull_pf`(x) [green], $\sqrt{x}$ [red], $M(x)$ [blue], for $x \in [1, 10^6]$. Note: `divisor_mean(x)/digits2`$(x)-$`number_squarefull_pf`(x) exceeds 2,000 for many values of x, but the maximum value of the vertical axis is set to 2,000 to better show the relationship between this function and $M(x)$.



Plot 13: (pi_function(x) + 1)/number_of_distinct_prime_factors(x)^2 [green], $\sqrt{x}$ [red], $M(x)$ [blue], for $x \in [1, 10^6]$.



Plot 14**: (pi_function(x) + 1)/number_of_distinct_prime_factors(x)^2 [green], $\sqrt{x}$ [red], $M(x)$ [blue], for $x \in [1, 10^6]$. Note: (pi_function(x) + 1)/number_of_distinct_prime_factors(x)^2 exceeds 2,500 for many values of x, but the maximum value of the vertical axis is set to 2,500 to better show the relationship between this function and $M(x)$.

Glossary of Invariants

The following is a list of all invariants used in our CONJECTURING trials, along with their definitions. For each definition, we include evaluations for $30 = 2 \cdot 3 \cdot 5$ and $60 = 2^2 \cdot 3 \cdot 5$ to provide an example:

Definition 1. *For all $x \in \mathbb{N}$,* `number`(x) *is the identity function, returns x*
(e.g. `number`$(30) = 30$, `number`$(60) = 60$*)*

Definition 2. *For all $x \in \mathbb{N}$,* `mertens_function`*(x) is the Mertens function of x,*

$$M(x) = \sum_{n=1}^{x} \mu(n)$$

where $\mu(n)$ is the Möbius function,

$$\mu(n) = \begin{cases} 0 & \text{if } n \text{ is not square-free} \\ (-1)^k & \text{if } n \text{ is square-free} \\ 1 & \text{if } n = 1 \end{cases}$$

and k is the number of distinct primes in the factorization of x by the Fundamental Theorem of Arithmetic (e.g. `mertens_function`$(30) = -3$, `mertens_function`$(60) = -1$*)*

Definition 3. *For all $x \in \mathbb{N}$,* `euler_phi_function`(x) *is Euler's totient function, which is the number of elements in the set* $\{n \leq x : n \in \mathbb{N}, gcd(n, x) = 1\}$
(e.g `euler_phi_function`$(30) = 8$, `euler_phi_function`$(60) = 16$*)*

Definition 4. *For all $x \in \mathbb{N}$, given $x = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$,*
`number_of_distinct_prime_factors`(x) *is k, the number of distinct primes, and* `number_of_prime_factors`(x) *is $\alpha_1 + \alpha_2 + \ldots + \alpha_k$, the total number of (not necessarily distinct) primes, in the unique prime factorization of x*
(e.g. `number_of_distinct_prime_factors`$(30) = 3$, `number_of_prime_factors`$(30) = 3$, `number_of_distinct_prime_factors`$(60) = 3$, `number_of_prime_factors`$(60) = 4$*)*

Definition 5. *For all $x \in \mathbb{N}$, given $x = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$,* `number_squarefull_pf`$(x)$ *counts the number of primes p_i ($1 \leq i \leq k$) for which $\alpha_i > 1$, and* `number_squarefree_pf`(x) *counts the number of primes p_i for which $\alpha_i = 1$, in the unique prime factorization of x*
(e.g. `number_squarefull_pf`$(30) = 0$, `number_squarefree_pf`$(30) = 3$, `number_squarefull_pf`$(60) = 1$, `number_squarefree_pf`$(60) = 2$*)*

Definition 6. *For all $x \in \mathbb{N}$,* `pi_function`(x) *is the prime-counting function, which for given x, is the number of primes p such that $p \leq x$ (e.g.* `pi_function`$(30) = 10$, `pi_function`$(60) = 17$*)*

Definition 7. *For all $x \in \mathbb{N}$,* `count_divisors`(x) *counts the divisors of x, or $d \in \mathbb{N}$ such that d divides x.* `sum_divisors`(x) *is the sum of these divisors, and* `sum_nontrivial_divisors`(x) *is the sum of all divisors except 1 and x.* `divisor_mean(x)` *is the arithmetic mean of all divisors.*
(e.g. `count_divisors`$(30) = 8$, `sum_divisors`$(30) = 72$, `sum_nontrivial_divisors`$(30) = 41$, `divisor_mean`$(30) = 9$, `count_divisors`$(60) = 12$, `sum_divisors`$(60) = 168$, `sum_nontrivial_divisors`$(60) = 107$, `divisor_mean`$(60) = 14$*)*

Definition 8. *For all $x \in \mathbb{N}$,* `number_factorizations`(x) *is the number of pairs of distinct $n_1, n_2 \in \mathbb{N}$ such that $n_1 n_2 = x$, plus 1 if x is a perfect square*
(e.g. `number_factorizations`$(30) = 4$, `number_factorizations`$(60) = 6$*)*

Definition 9. *For all $x \in \mathbb{N}$,* `digits10`(x) *is the number of digits of x when represented in base 10 (e.g.* `digits10`$(30) = 2$, `digits10`$(60) = 2$*)*

Definition 10. *For all $x \in \mathbb{N}$,* `digits2`(x) *is the number of digits of x when represented in base 2 (e.g.* `digits2`$(30) = 5$, `digits2`$(60) = 6$*)*

Definition 11. *For all $x \in \mathbb{N}$,* `lower_prime`(x) *is the largest prime p such that $p < x$*
(e.g. `lower_prime`$(30) = 29$, `lower_prime`$(60) = 59$*)*

Definition 12. *For all $x \in \mathbb{N}$,* `upper_prime`(x) *is the smallest prime p such that $p > x$*
(e.g. `upper_prime`$(30) = 31$, `upper_prime`$(60) = 61$*)*

Definition 13. *For all $x \in \mathbb{N}$,* `lower_prime_remainder`(x) *is $x - p$, where p is largest prime $< x$ (e.g.* `lower_prime_remainder`$(30) = 1$, `lower_prime_remainder(x)`$(60) = 1$*)*

Definition 14. *For all $x \in \mathbb{N}$,* `upper_prime_remainder`(x) *is $p - x$, where p is smallest prime $> x$ (e.g.* `upper_prime_remainder`$(30) = 1$, `upper_prime_remainder`$(60) = 1$*)*

Definition 15. *For all $x \in \mathbb{N}$,* `lower_prime_adjusted`(x) *is the largest prime p such that $p \leq x$*
(e.g. `lower_prime_adjusted`$(30) = 29$, `lower_prime_adjusted`$(60) = 59$*)*

Definition 16. *For all $x \in \mathbb{N}$,* `upper_prime_adjusted`(x) *is the smallest prime p such that $p \geq x$ (e.g.* `upper_prime_adjusted`$(30) = 31$, `upper_prime_adjusted`$(60) = 61$*)*

Definition 17. *For all* $x \in \mathbb{N}$, `lower_prime_remainder_adjusted`(x) *is* $x - p$*, where* p *is largest prime* $\leq x$*. (e.g.* `lower_prime_remainder_adjusted`$(30) = 1$, `lower_prime_remainder_adjusted`$(60) = 1$*)*

Definition 18. *For all* $x \in \mathbb{N}$, `upper_prime_remainder_adjusted`(x) *is* $p - x$*, where* p *is smallest prime* $\geq x$*. (e.g.* `upper_prime_remainder_adjusted`$(30) = 1$, `upper_prime_remainder_adjusted`$(60) = 1$*)*

Definition 19. *For all* $x \in \mathbb{N}$, `value_pi`(x) *returns the irrational number* π *(e.g.* `value_pi`$(30) = \pi$, `value_pi`$(60) = \pi$*)*

Definition 20. *For all* $x \in \mathbb{N}$, `value_e`(x) *returns the irrational number* e *(e.g.* `value_e`$(30) = e$, `value_e`$(60) = e$*)*

Definition 21. *For all* $x \in \mathbb{N}$, `value_golden_ratio`(x) *returns the irrational number* $\Phi = \frac{1+\sqrt{5}}{2}$ *(e.g.* `value_golden_ratio`$(30) = \frac{1+\sqrt{5}}{2}$, `value_golden_ratio`$(60) = \frac{1+\sqrt{5}}{2}$*)*

Index of CONJECTURING Trials Arguments and Results

Again, we interpret all output conjectures in this index as being supposed bounds for all $x \in \mathbb{N}$.

Section 1

In Section 1 of this Index, trials are conducted exactly as described in Chapter 3. At each trial round, either 1 object is added to the `objects` list (typically as a result of being as counter-example to one or more conjectures from the previous round), or an invariant is added to the `invariants` list. Section 1 also contains trials which use the `theory` bound `number`(x).

1.1 Input:

```
invariants=[number, mertens_function,
            number_of_distinct_prime_factors, number_of_prime_factors]
objects=[5, 20]
```

Conjectures:

```
[mertens_function(x)<= -number_of_prime_factors(x),
 mertens_function(x)<= -number_of_distinct_prime_factors(x) - 1]
```

2 is a counter-example to both conjectures in the previous trial (e.g. `mertens_function`$(2) = 0$, `-number_of_prime_factors`$(2) = -1$), so we add it to the `objects` list and proceed to the next trial. In doing so, we guarantee that the previous conjectures will not be output by the program again, as they have now been disproved, and output conjectures must be true for all objects in the list.

1.2 Input:

```
invariants=[number, mertens_function,
            number_of_distinct_prime_factors, number_of_prime_factors]
objects=[5, 20, 2]
```

Conjectures:

```
[mertens_function(x)<= number_of_distinct_prime_factors(x) - 1,
 mertens_function(x)<= (-number_of_distinct_prime_factors(x))^number(x)
                         - 1,
 mertens_function(x)<= -number_of_distinct_prime_factors(x)^2 + 1]
```

Here, we add 19 to the `objects` list and proceed to the next trial.

1.3 Input:

```
invariants=[number, mertens_function,
            number_of_distinct_prime_factors, number_of_prime_factors]
objects=[5, 20, 2, 19]
```

Conjectures:

```
[mertens_function(x)<= 4*(number_of_distinct_prime_factors(x) + 1)^2
                        - number(x),
```

```
 mertens_function(x)<= number_of_distinct_prime_factors(x) - 1,
 mertens_function(x)<= -number_of_distinct_prime_factors(x)^2 + 1,
 mertens_function(x)<= (-number_of_distinct_prime_factors(x))^number(x)
                         - 1]
```

97 is a counter-example to all conjectures in the previous trial,
(e.g. mertens_function$(97) = 1$, number_of_distinct_prime_factors$(97) - 1 = 0$) so we add it to the objects list and proceed to the next trial. In doing so, we guarantee that the previous conjectures will not be output by the program again, as they have now been disproved, and output conjectures must be true for all objects in the list.

1.4 Input:

```
invariants=[number, mertens_function,
            number_of_distinct_prime_factors, number_of_prime_factors]
objects=[5, 20, 19, 2, 97]
```

Conjectures:

```
[mertens_function(x)<= number_of_distinct_prime_factors(x),
 mertens_function(x)<= log(log(log(number(x))^2/log(10)^2)^4),
 mertens_function(x)<= (-number_of_prime_factors(x))
                        ^(number_of_distinct_prime_factors(x) - 1),
 mertens_function(x)<= number(x) - 2,
 mertens_function(x)<= log(log(1/4*number(x))^2) + 1]
```

999983 is a counter-example to the first conjecture in the previous trial,
(mertens_function$(999983) = 213$, number_of_distinct_prime_factors$(999983) = 1$) so we add it to the objects list and proceed to the next trial. In doing so, we

guarantee that this conjecture will not be output by the program again, as it has now been disproved, and output conjectures must be true for all objects in the list.

1.5 Input:

```
invariants=[number, mertens_function,
            number_of_distinct_prime_factors, number_of_prime_factors]
objects=[5, 20, 19, 2, 97, 999983]
```

Conjectures:

```
[mertens_function(x)<= number_of_distinct_prime_factors(x),
 mertens_function(x)<= log(log(log(number(x))^2/log(10)^2)^4),
 mertens_function(x)<= (-number_of_prime_factors(x))^
                        (number_of_distinct_prime_factors(x) - 1),
 mertens_function(x)<= number(x) - 2,
 mertens_function(x)<= log(log(1/4*number(x))^2) + 1]
```

1 is a counter-example to the fourth conjecture in the previous trial, (`mertens_function`$(1) = 1$, `number`$(1) - 2 = -1$) so we add it to the `objects` list and proceed to the next trial. In doing so, we guarantee that this conjecture will not be output by the program again, as it has now been disproved, and output conjectures must be true for all objects in the list.

1.6 Input:

```
invariants=[number, mertens_function,
            number_of_distinct_prime_factors, number_of_prime_factors]
objects=[5, 20, 19, 2, 97, 999983, 1]
```

Conjectures:

```
[mertens_function(x)<= number(x),
 mertens_function(x)<= maximum(number_of_distinct_prime_factors(x),
                        (log(number(x))/log(10) - 1)^4),
 mertens_function(x)<= 2/(number_of_distinct_prime_factors(x)
                        - number_of_prime_factors(x)) - 1,
 mertens_function(x)<= number(x)
                        - 2*number_of_distinct_prime_factors(x),
 mertens_function(x)<= 2*(log(1/2*number(x)) - 1)^2 - 2,
 mertens_function(x)<= 4*(log(2*number(x) - 2)/log(10) + 1)^2,
 mertens_function(x)<= 1/4*(sqrt(number(x)) - 4)^2 - 1]
```

Here, we add `euler_phi_function` to the `invariants` list and proceed to the next trial.

1.7 Input:

```
invariants=[number, mertens_function,
            number_of_distinct_prime_factors,
            number_of_prime_factors, euler_phi_function]
objects=[5, 20, 19, 2, 97, 999983, 1]
```

Conjectures:

```
[mertens_function(x)<= number(x),
 mertens_function(x)<= euler_phi_function(x)
                       - number_of_distinct_prime_factors(x),
 mertens_function(x)<= 4*(log(2*euler_phi_function(x))/log(10) + 1)^2,
 mertens_function(x)<= 2*euler_phi_function(x) - number(x) + 1,
 mertens_function(x)<= 2*sqrt(number(x))
                       - 2*log(euler_phi_function(x)^2),
```

```
 mertens_function(x)<= maximum(number_of_distinct_prime_factors(x),
                       (log(number(x))/log(10) - 1)^4),
 mertens_function(x)<= 2*(log(1/2*number(x)) - 1)^2 - 2]
```

12 is a counter-example to the fourth conjecture in the previous trial, (mertens_function$(12) = -2$, 2*euler_phi_function$(12)-$number$(12) + 1 = -3$) so we add it to the objects list and proceed to the next trial. In doing so, we guarantee that this conjecture will not be output by the program again, as it has now been disproved, and output conjectures must be true for all objects in the list.

1.8 Input:

```
invariants=[number, mertens_function,
            number_of_distinct_prime_factors,
            number_of_prime_factors, euler_phi_function]
objects=[5, 20, 19, 2, 97, 999983, 1, 12]
```

Conjectures:

```
[mertens_function(x)<= number(x),
 mertens_function(x)<= euler_phi_function(x)
                       - number_of_distinct_prime_factors(x),
 mertens_function(x)<= 4*(log(2*euler_phi_function(x))/log(10) + 1)^2,
 mertens_function(x)<= -number_of_distinct_prime_factors(x)
                       *number_of_prime_factors(x)
                       + euler_phi_function(x),
 mertens_function(x)<= -euler_phi_function(x)
                        /(number_of_prime_factors(x) - 1) + 1,
 mertens_function(x)<= 2*sqrt(number(x))
                       - 2*log(euler_phi_function(x)^2),
```

```
 mertens_function(x)<= maximum(number_of_distinct_prime_factors(x),
                        (log(number(x))/log(10) - 1)^4),
 mertens_function(x)<= -4*sqrt(euler_phi_function(x) - 1) + number(x)]
```

6 is a counter-example to the fourth conjecture in the previous trial, (mertens_function$(6) = -1$, -number_of_distinct_prime_factors(6) $*$number_of_prime_factors$(6)+$euler_phi_function$(6) = -2$) so we add it to the objects list and proceed to the next trial. In doing so, we guarantee that this conjecture will not be output by the program again, as it has now been disproved, and output conjectures must be true for all objects in the list.

1.9 Input:

```
invariants=[number, mertens_function,
            number_of_distinct_prime_factors,
            number_of_prime_factors, euler_phi_function]
objects=[5, 20, 19, 2, 97, 999983, 1, 12, 6]
```

Conjectures:

```
[mertens_function(x)<= number(x),
 mertens_function(x)<= euler_phi_function(x)
                        - number_of_distinct_prime_factors(x),
 mertens_function(x)<= 4*(log(2*euler_phi_function(x))/log(10) + 1)^2,
 mertens_function(x)<= 2*euler_phi_function(x) - number(x) + 2,
 mertens_function(x)<= -4*sqrt(euler_phi_function(x) - 1) + number(x),
 mertens_function(x)<= 2*sqrt(number(x))
                         - 2*log(euler_phi_function(x)^2),
 mertens_function(x)<= maximum(number_of_distinct_prime_factors(x),
                        (log(number(x))/log(10) - 1)^4),
```

```
 mertens_function(x)<= -1/(number_of_distinct_prime_factors(x) - 1),
 mertens_function(x)<= -euler_phi_function(x)
                        /(number_of_prime_factors(x) - 1) + 1]
```

1000000 is a counter-example to the eighth conjecture in the previous trial, (mertens_function$(1000000) = 212$, -1/(number_of_distinct_prime_factors$(1000000) - 1) = -1$) so we add it to the objects list and proceed to the next trial. In doing so, we guarantee that this conjecture will not be output by the program again, as it has now been disproved, and output conjectures must be true for all objects in the list.

1.10 Input:

```
invariants=[number, mertens_function,
            number_of_distinct_prime_factors,
            number_of_prime_factors, euler_phi_function]
objects=[5, 20, 19, 2, 97, 999983, 1, 12, 6, 1000000]
```

Conjectures:

```
[mertens_function(x)<= number(x),
 mertens_function(x)<= euler_phi_function(x)
                        - number_of_distinct_prime_factors(x),
 mertens_function(x)<= 4*(log(2*number(x))/log(10) + 1)^2,
 mertens_function(x)<= maximum(number_of_distinct_prime_factors(x),
                        (log(number(x))/log(10) - 1)^4),
 mertens_function(x)<= (-sqrt(euler_phi_function(x)))
                        ^number_of_prime_factors(x) + number(x),
 mertens_function(x)<= 2*sqrt(number(x))
                         - 2*log(euler_phi_function(x)^2),
```

```
 mertens_function(x)<= euler_phi_function(x)
                       /number_of_distinct_prime_factors(x)
                       - number_of_prime_factors(x),
 mertens_function(x)<= -4*sqrt(euler_phi_function(x) - 1) + number(x),
 mertens_function(x)<= (sqrt(euler_phi_function(x)) + 1)
                       /(number_of_distinct_prime_factors(x) + 1) + 1,
 mertens_function(x)<= -number_of_prime_factors(x)^2 + 1/2*number(x)
                        + 1]
```

Here, we prove the first conjecture from the previous trial, and add this bound, `number`, to the `theory` list and proceed to the next trial.

1.11 Input:

```
invariants=[number, mertens_function,
            number_of_distinct_prime_factors,
            number_of_prime_factors, euler_phi_function]
theory=[number]
objects=[5, 20, 19, 2, 97, 999983, 1, 12, 6, 1000000]
```

Conjectures:

```
[mertens_function(x)<= euler_phi_function(x),
 mertens_function(x)<= 4*(log(2*number(x) - 2)/log(10) + 1)^2,
 mertens_function(x)<= euler_phi_function(x)
                       - number_of_distinct_prime_factors(x),
 mertens_function(x)<= maximum(number_of_distinct_prime_factors(x),
                       (log(number(x))/log(10) - 1)^4),
 mertens_function(x)<= (-sqrt(euler_phi_function(x)))
                       ^number_of_prime_factors(x) + number(x),
```

```
mertens_function(x)<= 2*sqrt(number(x))
                       - 2*log(euler_phi_function(x)^2),
mertens_function(x)<= euler_phi_function(x)
                      /number_of_distinct_prime_factors(x)
                      - number_of_prime_factors(x),
mertens_function(x)<= -4*sqrt(euler_phi_function(x) - 1) + number(x),
mertens_function(x)<= (sqrt(euler_phi_function(x)) + 1)
                      /(number_of_distinct_prime_factors(x) + 1) + 1,
mertens_function(x)<= -number_of_prime_factors(x)^2 + 1/2*number(x)
                       + 1]
```

Here, we add `pi_function` to the `invariants` list and proceed to the next trial.

1.12 Input:

```
invariants=[number, mertens_function,
            number_of_distinct_prime_factors,
            number_of_prime_factors, euler_phi_function, pi_function]
theory=[number]
objects=[5, 20, 19, 2, 97, 999983, 1, 12, 6, 1000000]
```

Conjectures:

```
[mertens_function(x)<= euler_phi_function(x),
 mertens_function(x)<= 4*(log(2*number(x))/log(10) + 1)^2,
 mertens_function(x)<= euler_phi_function(x)
                       - number_of_distinct_prime_factors(x),
 mertens_function(x)<= euler_phi_function(x) - pi_function(x),
 mertens_function(x)<= 2*number(x) - 4*pi_function(x),
 mertens_function(x)<= 1/2*euler_phi_function(x)
```

```
                        - pi_function(x) + 1,
mertens_function(x)<= pi_function(x)^(log(number(x))/log(10))
                        - euler_phi_function(x) + 1,
mertens_function(x)<= maximum(number_of_distinct_prime_factors(x),
                        (-number(x))^pi_function(x)),
mertens_function(x)<= (log(2*euler_phi_function(x) - 2) + 1)^2]
```

Section 2

Section 2 includes trials with our full list of invariants (all those listed in the glossary). Also, all trials in Section 2 include the `theory` bound `divisor_mean`(x).

2.1 Input:

```
invariants=[number, mertens_function, number_of_distinct_prime_factors,
            number_of_prime_factors, number_squarefull_pf,
            number_squarefree_pf, count_divisors, sum_divisors,
            sum_nontrivial_divisors, number_factorizations,
            divisor_mean, digits10, digits2, lower_prime,
            lower_prime_remainder, upper_prime, upper_prime_remainder,
            lower_prime_adjusted, upper_prime_adjusted,
            lower_prime_remainder_adjusted,
            upper_prime_remainder_adjusted, pi_function, value_pi,
            value_e, value_golden_ratio, euler_phi_function]
objects=[1, 2, 3, 4, 5, 19, 20, 30, 31, 40, 60, 95, 96, 97, 100, 218,
         999983, 1000000]
theory=[divisor_mean]
```

Conjectures:

```
mertens_function(x)<= euler_phi_function(x)
mertens_function(x)<= (number_of_prime_factors(x)^2
                        - euler_phi_function(x))^2
mertens_function(x)<= (digits2(x) + value_golden_ratio(x))
                       *value_pi(x)^2
mertens_function(x)<= euler_phi_function(x) - pi_function(x)
mertens_function(x)<= -count_divisors(x) + upper_prime_adjusted(x)
mertens_function(x)<= -upper_prime_remainder(x)^2 + number(x) + 1
mertens_function(x)<= sum_nontrivial_divisors(x)^(count_divisors(x)
                        - digits10(x))
mertens_function(x)<= number(x)/number_of_prime_factors(x)
                        - count_divisors(x)
mertens_function(x)<= upper_prime_remainder(x)^4
                       *value_golden_ratio(x)^2
mertens_function(x)<= digits10(x)^upper_prime_remainder(x)
mertens_function(x)<= number(x) - 2*pi_function(x)
mertens_function(x)<= sqrt((euler_phi_function(x)
                           - sum_nontrivial_divisors(x))^2)
mertens_function(x)<= digits10(x)^number(x) - pi_function(x)
mertens_function(x)<= -number_factorizations(x)*number_squarefree_pf(x)
                           + divisor_mean(x)
mertens_function(x)<= number_factorizations(x)^upper_prime_remainder(x)
                           /upper_prime_remainder_adjusted(x)
mertens_function(x)<= divisor_mean(x)/upper_prime_remainder(x)
                           - count_divisors(x) + 1
mertens_function(x)<= upper_prime_remainder(x)^number(x)
                           + number_squarefree_pf(x)
```

```
mertens_function(x)<= -digits2(x)^2 + count_divisors(x)
                        + sum_divisors(x)
```

6, 7, 8, 9, 17, 23, 24, 48, 54, 84, 90, 113, 114, 115, 221, 222, 226, 228, 346, 553, 554, 556, 562, 566, 568, 570, 576, 586, 926, 961, 1007, and 1263 are various counter-examples to the conjectures in the previous trial, so we add them to the `objects` list and proceed to the next trial. In doing so, we guarantee that many of the above conjectures will not be output by the program again, as they have now been disproved, and output conjectures must be true for all objects in the list.

2.2 Input:

```
invariants=[number, mertens_function, number_of_distinct_prime_factors,
            number_of_prime_factors, number_squarefull_pf,
            number_squarefree_pf, count_divisors, sum_divisors,
            sum_nontrivial_divisors, number_factorizations,
            divisor_mean, digits10, digits2, lower_prime,
            lower_prime_remainder, upper_prime, upper_prime_remainder,
            lower_prime_adjusted, upper_prime_adjusted,
            lower_prime_remainder_adjusted,
            upper_prime_remainder_adjusted, pi_function, value_pi,
            value_e, value_golden_ratio, euler_phi_function]
objects=[1, 2, 3, 4, 5, 19, 20, 30, 31, 40, 60, 95, 96, 97, 100,
         218, 999983, 1000000, 6, 7, 8, 9, 17, 23, 24, 48, 54, 84, 90,
         113, 114, 115, 221, 222, 226, 228, 346, 553, 554, 556, 562,
         566, 568, 570, 576, 586, 926, 961, 1007, 1263]
theory=[divisor_mean]
```

Conjectures:

```
mertens_function(x)<= euler_phi_function(x)

mertens_function(x)<= (log(2*euler_phi_function(x)) + 1)^2

mertens_function(x)<= digits10(x)^value_pi(x)*log(10)/log(digits2(x))

mertens_function(x)<= number(x)/value_pi(x) - pi_function(x) + 1

mertens_function(x)<= euler_phi_function(x) - pi_function(x)

mertens_function(x)<= sqrt(euler_phi_function(x)) - digits2(x)
                        + number_factorizations(x)

mertens_function(x)<= (divisor_mean(x)/upper_prime_remainder(x))
                        ^(log(digits10(x))/log(10))

mertens_function(x)<= count_divisors(x)^digits2(x)
                        /sum_nontrivial_divisors(x)^2

mertens_function(x)<= number(x)/number_of_prime_factors(x)
                        - count_divisors(x)

mertens_function(x)<= 2*digits10(x)/(number_squarefree_pf(x)
                        *number_squarefull_pf(x))

mertens_function(x)<= number(x) - 2*pi_function(x)

mertens_function(x)<= -count_divisors(x) + upper_prime_adjusted(x)

mertens_function(x)<= number(x)/count_divisors(x)
                         - number_of_prime_factors(x)

mertens_function(x)<= (digits2(x)/number_squarefull_pf(x))
                         ^upper_prime_remainder(x)

mertens_function(x)<= sqrt(-divisor_mean(x) + euler_phi_function(x)
                        + 1)

mertens_function(x)<= divisor_mean(x)/upper_prime_remainder(x)
                         - log(number(x))

mertens_function(x)<= number_factorizations(x)^2
                        /number_squarefull_pf(x)
```

```
mertens_function(x)<= 2*digits10(x)^lower_prime_remainder(x)
mertens_function(x)<= divisor_mean(x)/digits2(x)
                       - number_squarefull_pf(x)
mertens_function(x)<= upper_prime_adjusted(x)/digits2(x)
                         - upper_prime_remainder(x)
mertens_function(x)<= digits10(x)^number(x)
                       - number_of_prime_factors(x)
mertens_function(x)<= count_divisors(x)/(number_squarefree_pf(x)
                         *number_squarefull_pf(x))
mertens_function(x)<= count_divisors(x)^upper_prime_remainder(x)
                         + digits10(x)
mertens_function(x)<= maximum(lower_prime_remainder(x),
                       count_divisors(x)^upper_prime_remainder(x))
mertens_function(x)<= sqrt(euler_phi_function(x))
                         - upper_prime_remainder(x) + 1
mertens_function(x)<= number_of_prime_factors(x)^value_e(x)
                         + 1/upper_prime_remainder_adjusted(x)
mertens_function(x)<= number(x)/count_divisors(x) - digits2(x) + 1
mertens_function(x)<= digits10(x)^lower_prime_remainder(x)
                         /number_squarefull_pf(x)
mertens_function(x)<= (number_of_prime_factors(x)^2
                         - euler_phi_function(x))^2
mertens_function(x)<= (count_divisors(x)^upper_prime(x))
                         ^(1/euler_phi_function(x))
mertens_function(x)<= -number_factorizations(x)*number_squarefree_pf(x)
                         + divisor_mean(x)
mertens_function(x)<= upper_prime(x)/(sqrt(number(x))
```

```
                          *upper_prime_remainder_adjusted(x))
mertens_function(x)<= 1/2*(divisor_mean(x)
                         - sum_nontrivial_divisors(x))^2
mertens_function(x)<= -upper_prime_remainder(x)^2 + 2*number(x)
mertens_function(x)<= (digits10(x) - 1)^number(x) + 1
mertens_function(x)<= divisor_mean(x)/upper_prime_remainder(x)
                          - number_of_prime_factors(x) + 1
mertens_function(x)<= sqrt(divisor_mean(x)) + count_divisors(x)
                          - upper_prime_remainder(x)
mertens_function(x)<= (divisor_mean(x) - 2*pi_function(x))^2
mertens_function(x)<= sqrt(divisor_mean(x)) + number_factorizations(x)
                          - upper_prime_remainder_adjusted(x)
mertens_function(x)<= count_divisors(x)*number_factorizations(x)
                          - log(upper_prime_remainder_adjusted(x))
mertens_function(x)<= digits10(x)^number(x) - digits2(x) + 1
```

Section 3

For the trial in Section 3, rather than disproving a single conjecture and adding the counter-example to the `objects` list, we find all counter-examples within a testable range ($x \leq 10000$), and add the most commonly occurring counter-examples to the list. This allows us to add counter-examples that are more significant than simply the smallest available counter-example, or worse, a randomly chosen counter-example. Also, all trials in Section 3 include the `theory` bound `divisor_mean`(x).

216, 219, 220, 223, 551, 552, 557, 558, 560, 569, 572, 573, 578, 579, 580, 588, 589, 594, 595, 600, 934, 1327, 1328, 1329, 1330, 1351, 1366, 1372, 1383, 1386, 1410, 1412, 1413, 1422, 1426, 1427, 1994, 3251, 3270, 3276, 3280, 3282, 3299, 8518, 8598 are the most commonly-

occurring counter-examples to the conjectures in the previous trial, so we add them to the `objects` list and proceed to the next trial. In doing so, we guarantee that many of the conjectures from the previous round will not be output by the program again, as they have now been disproved, and output conjectures must be true for all objects in the list.

3.1 Input:

```
invariants=[number, mertens_function, number_of_distinct_prime_factors,
            number_of_prime_factors, number_squarefull_pf,
            number_squarefree_pf, count_divisors, sum_divisors,
            sum_nontrivial_divisors, number_factorizations,
            divisor_mean, digits10, digits2, lower_prime,
            lower_prime_remainder, upper_prime, upper_prime_remainder,
            lower_prime_adjusted, upper_prime_adjusted,
            lower_prime_remainder_adjusted,
            upper_prime_remainder_adjusted, pi_function, value_pi,
            value_e, value_golden_ratio, euler_phi_function]
objects=[1, 2, 3, 4, 5, 19, 20, 30, 31, 40, 60, 95, 96, 97, 100, 218,
         999983, 1000000, 6, 7, 8, 9, 17, 23, 24, 48, 54, 84, 90, 113,
         114, 115, 221, 222, 226, 228, 346, 553, 554, 556, 562, 566,
         568, 570, 576, 586, 926, 961, 1007, 1263, 216, 219, 220, 223,
         551, 552, 557, 558, 560, 569, 572, 573, 578, 579, 580, 588,
         589, 594, 595, 600, 934, 1327, 1328, 1329, 1330, 1351, 1366,
         1372, 1383, 1386, 1410, 1412, 1413, 1422, 1426, 1427, 1994,
         3251, 3270, 3276, 3280, 3282, 3299, 8518, 8598]
theory=[divisor_mean]
```

Conjectures:

```
mertens_function(x)<= euler_phi_function(x)
```

```
mertens_function(x)<= sqrt(2*digits2(x) + pi_function(x))
mertens_function(x)<= number(x)/number_of_prime_factors(x)
                       - count_divisors(x)
mertens_function(x)<= euler_phi_function(x) - pi_function(x)
mertens_function(x)<= divisor_mean(x)/digits2(x)
                      - number_squarefull_pf(x)
mertens_function(x)<= divisor_mean(x)
                      ^number_of_distinct_prime_factors(x)
                      /upper_prime_remainder(x)^2
mertens_function(x)<= number(x)/count_divisors(x)
                       - number_of_prime_factors(x)
mertens_function(x)<= (sum_divisors(x)/digits2(x))
                       ^(log(digits10(x))/log(10))
mertens_function(x)<= number(x)^(log(log(euler_phi_function(x))
                       /log(10))/log(10))
mertens_function(x)<= (log(number(x))/log(10))^log(digits2(x))
mertens_function(x)<= -count_divisors(x) + upper_prime_adjusted(x)
mertens_function(x)<= 1/2*sqrt(1/2)*sqrt(upper_prime(x)) + 1/2
mertens_function(x)<= divisor_mean(x)^(digits10(x)
                       /number_of_prime_factors(x))
mertens_function(x)<= divisor_mean(x)
                      /number_of_distinct_prime_factors(x)
                      - upper_prime_remainder(x)
mertens_function(x)<= sqrt(sum_divisors(x)/(upper_prime_remainder(x)
                      - 1))
mertens_function(x)<= (count_divisors(x)^divisor_mean(x))
                       ^(1/pi_function(x))
```

```
mertens_function(x)<= maximum(count_divisors(x), sqrt(divisor_mean(x)))
mertens_function(x)<= maximum(number_factorizations(x),
                        sqrt(divisor_mean(x))) + 1
mertens_function(x)<= number(x)/value_pi(x) - pi_function(x) + 1
mertens_function(x)<= number(x) - 2*pi_function(x)
mertens_function(x)<= (log(sum_nontrivial_divisors(x))/log(10))
                        ^(value_pi(x) + 1)
mertens_function(x)<= digits10(x)^number(x)
                       - number_of_prime_factors(x)
mertens_function(x)<= (1/2*divisor_mean(x))^(log(digits10(x))/log(10))
mertens_function(x)<= number(x)/(digits2(x)*log(upper_prime(x)))
mertens_function(x)<= (digits10(x)^number_of_distinct_prime_factors(x))
                        ^lower_prime_remainder(x)
mertens_function(x)<= (log(2*euler_phi_function(x)) + 1)^2
mertens_function(x)<= (number(x)/upper_prime_remainder(x))
                        ^(log(digits10(x))/log(10))
mertens_function(x)<= number_factorizations(x)^lower_prime_remainder(x)
                        + 1/number_squarefull_pf(x)
mertens_function(x)<= sum_divisors(x)/(digits10(x)^2
                        *upper_prime_remainder(x))
mertens_function(x)<= number(x)^2/upper_prime_remainder(x)^4
mertens_function(x)<= number_factorizations(x)
                       ^number_of_prime_factors(x)
                       /number_squarefull_pf(x)
mertens_function(x)<= (2*digits2(x) + 2)^upper_prime_remainder(x)
mertens_function(x)<= 1/2*euler_phi_function(x)/sqrt(pi_function(x))
mertens_function(x)<= digits2(x)^digits10(x)/pi_function(x)
```

```
mertens_function(x)<= 2*digits2(x)^2/number_squarefree_pf(x)^2
mertens_function(x)<= upper_prime_adjusted(x)/digits2(x)
                        - upper_prime_remainder(x)
mertens_function(x)<= upper_prime(x)/upper_prime_remainder(x)
                        - upper_prime_remainder_adjusted(x)
mertens_function(x)<= upper_prime_remainder(x)^2 + count_divisors(x)
                        ^digits10(x)
mertens_function(x)<= (number(x)^euler_phi_function(x))
                        ^(1/sum_nontrivial_divisors(x))
mertens_function(x)<= number(x)^number_of_distinct_prime_factors(x)
                        /digits2(x)^2
mertens_function(x)<= value_golden_ratio(x)^(divisor_mean(x)
                        /count_divisors(x))
mertens_function(x)<= count_divisors(x)^upper_prime_remainder(x)
                        /sqrt(number_squarefull_pf(x))
mertens_function(x)<= (2*count_divisors(x) - euler_phi_function(x))^2
mertens_function(x)<= number(x)^(count_divisors(x)
                        /upper_prime_remainder_adjusted(x))
mertens_function(x)<= minimum(number(x),
                       sqrt(sum_nontrivial_divisors(x) - 1))
mertens_function(x)<= -number_factorizations(x)*number_squarefree_pf(x)
                        + divisor_mean(x)
mertens_function(x)<= (log(1/2*pi_function(x))/log(10))
                         ^upper_prime_adjusted(x)
mertens_function(x)<= -upper_prime_remainder(x)^2 + 2*number(x)
mertens_function(x)<= number(x)/count_divisors(x) - digits2(x) + 1
mertens_function(x)<= sqrt(number(x)) - upper_prime_remainder(x) + 1
```

```
mertens_function(x)<= maximum(1/number_squarefull_pf(x),
                        count_divisors(x)^upper_prime_remainder(x))
mertens_function(x)<= (pi_function(x) - 1)^2/sum_divisors(x)
mertens_function(x)<= 4*euler_phi_function(x) - number(x)
mertens_function(x)<= digits10(x)^(log(1/2*divisor_mean(x))/log(10))
mertens_function(x)<= (number(x)^divisor_mean(x))
                        ^(1/euler_phi_function(x))
mertens_function(x)<= sum_divisors(x)/digits2(x) - 1/2*pi_function(x)
mertens_function(x)<= (number_of_prime_factors(x)^2
                        - divisor_mean(x))^2
mertens_function(x)<= (number_of_prime_factors(x)^2
                        - euler_phi_function(x))^2
mertens_function(x)<= (digits10(x) - 1)^number(x) + 1
mertens_function(x)<= (divisor_mean(x) - 2*pi_function(x))^2
mertens_function(x)<= sqrt(euler_phi_function(x)) - digits2(x)
                        + number_factorizations(x)
mertens_function(x)<= (-euler_phi_function(x)
                        + sum_nontrivial_divisors(x))
                        ^(2*upper_prime_remainder(x))
```

Section 4

For trials in Section 4, like Section 3, we find all counter-examples within a testable range and add the most commonly occurring counter-examples to the list. However, for Section 4, this testable range was raised to $x \leq 1000000$ due to the substitution of the `special` functions. Also, again all trials in Section 4 include the `theory` bound `divisor_mean`(x).

592, 593, 594, 1011, 1369, 1408, 1409, 3166, 3274, 3293, 3294, 3295, 3296, 3300, 3360, 4904, 7522, 8389, 8394, 8510, 8512, 8514, 8520, 8526, 8542, 8554, 8580, 8627 are the most commonly-occurring counter-examples to the conjectures in the previous trial, so we add them to the `objects` list and proceed to the next trial. In doing so, we guarantee that many of the conjectures from the previous round will not be output by the program again, as they have now been disproved, and output conjectures must be true for all objects in the input list.

4.1 Input:

```
invariants=[number, mertens_function, number_of_distinct_prime_factors,
            number_of_prime_factors, number_squarefull_pf,
            number_squarefree_pf, count_divisors, sum_divisors,
            sum_nontrivial_divisors, number_factorizations,
            divisor_mean, digits10, digits2, lower_prime,
            lower_prime_remainder, upper_prime, upper_prime_remainder,
            lower_prime_adjusted, upper_prime_adjusted,
            lower_prime_remainder_adjusted,
            upper_prime_remainder_adjusted, pi_function, value_pi,
            value_e, value_golden_ratio, euler_phi_function]
objects=[1, 2, 3, 4, 5, 19, 20, 30, 31, 40, 60, 95, 96, 97, 100, 218,
         999983, 1000000, 6, 7, 8, 9, 17, 23, 24, 48, 54, 84, 90, 113,
         114, 115, 221, 222, 226, 228, 346, 553, 554, 556, 562, 566,
         568, 570, 576, 586, 926, 961, 1007, 1263, 592, 593, 594, 1011,
         1369, 1408, 1409, 3166, 3274, 3293, 3294, 3295, 3296, 3300,
         3360, 4904, 7522, 8389, 8394, 8510, 8512, 8514, 8520, 8526,
         8542, 8554, 8580, 8627]
theory=[divisor_mean]
```

Conjectures:

```
mertens_function(x)<= euler_phi_function(x)

mertens_function(x)<= (sum_divisors(x)/digits2(x))
                       ^(log(digits10(x))/log(10))

mertens_function(x)<= (digits2(x) - number_squarefull_pf(x))
                       ^log(digits10(x))

mertens_function(x)<= euler_phi_function(x) - pi_function(x)

mertens_function(x)<= digits2(x)^maximum(number_squarefree_pf(x)
                                        , upper_prime_remainder(x))

mertens_function(x)<= divisor_mean(x)/(sqrt(number_factorizations(x))
                                      *upper_prime_remainder(x))

mertens_function(x)<= sqrt(pi_function(x)) + value_golden_ratio(x) + 1

mertens_function(x)<= (pi_function(x) + 1)/digits10(x)^2

mertens_function(x)<= number(x)/count_divisors(x)
                        - number_of_prime_factors(x)

mertens_function(x)<= digits10(x)^value_pi(x)*log(10)/log(digits2(x))

mertens_function(x)<= -count_divisors(x) + upper_prime_adjusted(x)

mertens_function(x)<= sqrt(pi_function(x)) + number_factorizations(x)

mertens_function(x)<= number(x)/number_of_prime_factors(x)
                        - count_divisors(x)

mertens_function(x)<= euler_phi_function(x)/count_divisors(x)

mertens_function(x)<= (digits2(x)/number_of_prime_factors(x))
                        ^log(sum_divisors(x))

mertens_function(x)<= count_divisors(x)*number_of_prime_factors(x)
                        + 1/number_squarefull_pf(x)

mertens_function(x)<= (log(2*euler_phi_function(x)) + 1)^2

mertens_function(x)<= -sqrt(sum_nontrivial_divisors(x))
```

```
                      + divisor_mean(x)/number_squarefull_pf(x)
mertens_function(x)<= number(x)/value_pi(x) - pi_function(x) + 1
mertens_function(x)<= (pi_function(x) + 1)
                      /(upper_prime_remainder(x) - 1)
mertens_function(x)<= number(x) - 2*pi_function(x)
mertens_function(x)<= sqrt(-divisor_mean(x) + euler_phi_function(x)
                      + 1)
mertens_function(x)<= (number(x)^euler_phi_function(x))
                       ^(1/sum_nontrivial_divisors(x))
mertens_function(x)<= value_golden_ratio(x)^(1/2*digits10(x)^2)
mertens_function(x)<= divisor_mean(x)/digits2(x)
                       - number_of_distinct_prime_factors(x) + 1
mertens_function(x)<= sqrt((divisor_mean(x)
                      - sum_nontrivial_divisors(x))^2)
mertens_function(x)<= digits10(x)^number(x)
                      - number_of_prime_factors(x)
mertens_function(x)<= sqrt(euler_phi_function(x)
                      /number_squarefull_pf(x))
mertens_function(x)<= divisor_mean(x)/upper_prime_remainder(x)
                       - log(number(x))
mertens_function(x)<= 2*count_divisors(x)^value_e(x)
                      *upper_prime_remainder(x)
mertens_function(x)<= divisor_mean(x)/upper_prime_remainder(x)
                       - number_of_prime_factors(x) + 1
mertens_function(x)<= sqrt(1/2)*sqrt(sum_divisors(x)
                                    /number_of_prime_factors(x))
mertens_function(x)<= (divisor_mean(x)/value_golden_ratio(x))
```

```
                        ^(log(digits10(x))/log(10))
mertens_function(x)<= digits2(x)^digits10(x)/pi_function(x)
mertens_function(x)<= number(x)/count_divisors(x) - digits2(x) + 1
mertens_function(x)<= (sum_nontrivial_divisors(x)
                       + upper_prime_remainder(x))
                       /upper_prime_remainder_adjusted(x)
mertens_function(x)<= upper_prime_adjusted(x)/digits2(x)
                        - upper_prime_remainder(x)
mertens_function(x)<= digits2(x)*number_of_prime_factors(x)
                        + 1/upper_prime_remainder_adjusted(x)
mertens_function(x)<= (log(sum_nontrivial_divisors(x))/log(10))
                        ^log(divisor_mean(x))
mertens_function(x)<= (count_divisors(x)
                        + number_of_distinct_prime_factors(x))
                        ^(log(pi_function(x))/log(10))
mertens_function(x)<= -number_factorizations(x)*number_squarefree_pf(x)
                        + divisor_mean(x)
mertens_function(x)<= -upper_prime_remainder(x)^2 + 2*number(x)
mertens_function(x)<= sqrt(number(x)) - upper_prime_remainder(x) + 1
mertens_function(x)<= sqrt(euler_phi_function(x)) - digits2(x)
                        + number_factorizations(x)
mertens_function(x)<= number_factorizations(x)^2
                        + digits10(x)/number_squarefull_pf(x)
mertens_function(x)<= digits10(x)^number(x) - digits2(x) + 1
mertens_function(x)<= (number_of_prime_factors(x)
                       /number_squarefull_pf(x))^(digits10(x) - 1)
mertens_function(x)<= digits2(x)^digits10(x) - divisor_mean(x) + 1
```

```
mertens_function(x)<= divisor_mean(x)*log(10)/(log(sum_divisors(x))
                      *number_squarefree_pf(x))
mertens_function(x)<= divisor_mean(x)/number_squarefree_pf(x)
                       - upper_prime_remainder_adjusted(x) - 1
mertens_function(x)<= (number_of_prime_factors(x)^2
                       - euler_phi_function(x))^2
mertens_function(x)<= digits10(x)^lower_prime_remainder(x)
                       + 2*count_divisors(x)
mertens_function(x)<= (digits10(x) - 1)^number(x) + 1
```

216, 220, 600, 3481, 8522, 8568, 11760, 11776, 11777, 11793, 11794, 11881 are the most commonly-occurring counter-examples to the conjectures in the previous trial, so we add them to the `objects` list and proceed to the next trial. In doing so, we guarantee that many of the conjectures from the previous round will not be output by the program again, as they have now been disproved, and output conjectures must be true for all objects in the input list. In this trial, we also added 24185, 48433, and 300551 to the `objects` list. Rather than coming from conjecture testing, these three objects are significant values of $M(x)$ which are discussed in [2]. Additionally, the `special` functions were used in trial 4.2, the final round of our investigation. As this was the last trial we conducted, the conjectures below have not been tested for counter-examples for any range.

The following trial was our

4.2 Input:

```
invariants=[number, mertens_function, number_of_distinct_prime_factors,
            number_of_prime_factors, number_squarefull_pf,
            number_squarefree_pf, count_divisors, sum_divisors,
            sum_nontrivial_divisors, number_factorizations,
```

```
            divisor_mean, digits10, digits2, lower_prime,
            lower_prime_remainder, upper_prime, upper_prime_remainder,
            lower_prime_adjusted, upper_prime_adjusted,
            lower_prime_remainder_adjusted,
            upper_prime_remainder_adjusted, pi_function, value_pi,
            value_e, value_golden_ratio, euler_phi_function]
objects=[1, 2, 3, 4, 5, 19, 20, 30, 31, 40, 60, 95, 96, 97, 100, 218,
         999983, 1000000, 6, 7, 8, 9, 17, 23, 24, 48, 54, 84, 90, 113,
         114, 115, 221, 222, 226, 228, 346, 553, 554, 556, 562, 566,
         568, 570, 576, 586, 926, 961, 1007, 1263, 592, 593, 594, 1011,
         1369, 1408, 1409, 3166, 3274, 3293, 3294, 3295, 3296, 3300,
         3360, 4904, 7522, 8389, 8394, 8510, 8512, 8514, 8520, 8526,
         8542, 8554, 8580, 8627, 216, 220, 600, 3481, 8522, 8568,
         11760, 11776, 11777, 11793, 11794, 11881, 24185, 48433,
         300551]
theory=[divisor_mean]
```

Conjectures:

```
mertens_special(x)<= euler_phi_function(x)
mertens_special(x)<= number(x)^(log(log(euler_phi_function(x))
                                    /log(10))/log(10))
mertens_special(x)<= sqrt(divisor_mean(x)) + 2*count_divisors(x)
mertens_special(x)<= euler_phi_function(x) - pi_function(x)
mertens_special(x)<= divisor_mean(x)/digits2(x)
                     - number_squarefull_pf(x)
mertens_special(x)<= (euler_phi_function(x) - pi_function(x))
                       /upper_prime_remainder(x)
```

```
mertens_special(x)<= 1/2*(pi_function(x) + 1)
                     /number_of_prime_factors(x)
mertens_special(x)<= 2*(digits2(x) + 1)*digits10(x)
mertens_special(x)<= divisor_mean(x)
                     /number_of_distinct_prime_factors(x)
                     - upper_prime_remainder(x)
mertens_special(x)<= digits10(x)^(count_divisors(x) + 1)
mertens_special(x)<= -count_divisors(x) + upper_prime_adjusted(x)
mertens_special(x)<= log(digits2(x))^(log(number(x))/log(10))
mertens_special(x)<= number(x)/number_of_prime_factors(x)
                       - count_divisors(x)
mertens_special(x)<= euler_phi_function(x)/count_divisors(x)
mertens_special(x)<= divisor_mean(x)
                     ^number_of_distinct_prime_factors(x)
                     /sum_nontrivial_divisors(x)
mertens_special(x)<= number(x) - 2*pi_function(x)
mertens_special(x)<= digits10(x)^number(x) - number_of_prime_factors(x)
mertens_special(x)<= upper_prime_adjusted(x)/upper_prime_remainder(x)
                      - digits2(x)
mertens_special(x)<= (log(sum_divisors(x))/log(10))^(digits10(x) - 1)
mertens_special(x)<= (pi_function(x) + 1)
                     /(upper_prime_remainder(x) - 1)
mertens_special(x)<= digits2(x)^digits10(x)/pi_function(x)
mertens_special(x)<= (log(pi_function(x))/log(10))^(upper_prime(x)^2)
mertens_special(x)<= sqrt(value_golden_ratio(x))^(digits10(x)^2)
mertens_special(x)<= (pi_function(x) + 1)
                       /number_of_distinct_prime_factors(x)^2
```

```
mertens_special(x)<= sqrt(1/2)*sqrt(euler_phi_function(x)) + value_e(x)
mertens_special(x)<= (upper_prime_remainder(x) + 1)^(2*value_e(x))
mertens_special(x)<= (2*upper_prime_remainder(x))
                     ^(count_divisors(x) + 1)
mertens_special(x)<= digits2(x)*number_factorizations(x)
                       /number_squarefull_pf(x)
mertens_special(x)<= upper_prime_adjusted(x)/digits2(x)
                       - upper_prime_remainder(x)
mertens_special(x)<= number(x)/count_divisors(x)
                       - number_of_prime_factors(x)
mertens_special(x)<= upper_prime(x)/upper_prime_remainder(x)
                       - upper_prime_remainder_adjusted(x)
mertens_special(x)<= 2*sqrt(divisor_mean(x)) - upper_prime_remainder(x)
mertens_special(x)<= -(euler_phi_function(x) - upper_prime(x))
                     *digits2(x)
mertens_special(x)<= (pi_function(x) + 1)/digits10(x)^2
mertens_special(x)<= (count_divisors(x) + 1)^(digits10(x) - 1)
mertens_special(x)<= sqrt(value_golden_ratio(x))^sqrt(pi_function(x))
mertens_special(x)<= (2*sum_nontrivial_divisors(x) + 1)
                       /upper_prime_remainder_adjusted(x)
mertens_special(x)<= (log(1/2*pi_function(x))/log(10))
                       ^upper_prime_adjusted(x)
mertens_special(x)<= number(x)^(value_e(x)/digits10(x))
mertens_special(x)<= value_pi(x)^digits10(x)/value_e(x)
mertens_special(x)<= value_e(x)^digits10(x) - upper_prime_remainder(x)
mertens_special(x)<= (pi_function(x) - 1)^2/sum_divisors(x)
mertens_special(x)<= -number_factorizations(x)*number_squarefree_pf(x)
```

```
                        + divisor_mean(x)
mertens_special(x)<= 1/2*sqrt(euler_phi_function(x))
                       + count_divisors(x)
mertens_special(x)<= (log(sum_nontrivial_divisors(x))/log(10))
                        ^log(divisor_mean(x))
mertens_special(x)<= -upper_prime_remainder(x)^2 + 2*number(x)
mertens_special(x)<= sqrt(number(x)) - upper_prime_remainder(x) + 1
mertens_special(x)<= (log(sum_nontrivial_divisors(x))/log(10))
                        ^(2*value_e(x))
mertens_special(x)<= (2*divisor_mean(x))^(1/number_squarefull_pf(x))
mertens_special(x)<= maximum(count_divisors(x)^2,
                       sqrt(divisor_mean(x)))
mertens_special(x)<= sum_divisors(x)/(digits2(x) - 1)^2
```

Bibliography

[1] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers, 4th ed*. Oxford University Press, London (1975)

[2] T. Kotnik and J. van de Lune, On the Order of the Mertens Function, *Experimental Mathematics* **13:4** (2000) 473-481.

[3] C. E. Larson and N. Van Cleemput, Automated Conjecturing I: Fajtlowicz's Dalmatian Heuristic Revisited, *Artificial Intelligence* **231** (2016) 17-38.

[4] F. Mertens, Über eine zahlentheoretische Funktion, *Akademie Wissenschaftlicher Wien Mathematik-Naturlich Kleine Sitzungsber*, **IIa 106**, (1897) 761–830.

[5] A. M. Odlyzko and Herman te Riele, Disproof of the Mertens Conjecture, *Journal für die reine und angewandte Mathematik* **357**, (1985) 138–160.