The Complexity of Number Theory

Frank Vega
Joysonic, Uzun Mirkova 5, Belgrade, 11000, Serbia
vega.frank@gmail.com

Abstract

The Goldbach's conjecture has been described as the most difficult problem in the history of Mathematics. This conjecture states that every even integer greater than 2 can be written as the sum of two primes. This is known as the strong Goldbach's conjecture. The conjecture that all odd numbers greater than 7 are the sum of three odd primes is known today as the weak Goldbach conjecture. A principal complexity class is NSPACE(S(n)) for some S(n). We show if the weak Goldbach's conjecture is true, then the problem PRIMES is not in NSPACE(S(n)) for all S(n) = o(log n). This proof is based on the assumption that if some language belongs to NSPACE(S(n)), then the unary version of that language belongs to NSPACE(S(log n)) and vice versa. However, if PRIMES is not in NSPACE(S(n)) for all S(n) = o(log n), then the strong Goldbach's conjecture is true or this has an infinite number of counterexamples. Since Harald Helfgott proved that the weak Goldbach's conjecture is true, then the strong Goldbach's conjecture is true or this has an infinite number of counterexamples, where the case of infinite number of counterexamples statistically seems to be unlikely. In addition, if PRIMES is not in NSPACE(S(n)) for all S(n) = o(log n), then the twin prime conjecture is true.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Regular languages; Theory of computation → Problems, reductions and completeness; Mathematics of computing → Number-theoretic computations

Keywords and phrases complexity classes, regular languages, number theory, conjecture, primes

1 Introduction

Number theory is a branch of pure mathematics devoted primarily to the study of the integers and integer-valued functions [17]. Goldbach's conjecture is one of the most important and unsolved problems in number theory [9]. Nowadays, it is one of the open problems of Hilbert and Landau [9]. Goldbach's original conjecture, written on 7 June 1742 in a letter to Leonhard Euler, states: "... at least it seems that every number that is greater than 2 is the sum of three primes" [6]. This is known as the ternary Goldbach conjecture. We call a prime as a natural number that is greater than 1 and has exactly two divisors, 1 and the number itself [20]. However, the mathematician Christian Goldbach considered 1 as a prime number. Euler replied in a letter dated 30 June 1742 the following statement: "Every even integer greater than 2 can be written as the sum of two primes" [6]. This is known as the strong Goldbach conjecture.

Using Vinogradov's method [19], it has been showed that almost all even numbers can be written as the sum of two primes. In 1973, Chen showed that every sufficiently large even number can be written as the sum of some prime number and a semi-prime [4]. The strong Goldbach conjecture implies the conjecture that all odd numbers greater than 7 are the sum of three odd primes, which is known today as the weak Goldbach conjecture [6]. In 2012 and 2013, Peruvian mathematician Harald Helfgott published a pair of papers claiming to improve major and minor arc estimates sufficiently to unconditionally prove the weak Goldbach conjecture [11], [12]. In this work, we prove the strong Goldbach's conjecture is true or this has an infinite number of counterexamples.

Statistical considerations that focus on the probabilistic distribution of prime numbers present informal evidence in pos of the strong conjecture for sufficiently large integers: The greater the integer, the more ways there are available for that number to be represented as the sum of two other numbers, and the more "likely" it becomes that at least one of these representations consists entirely of primes. In this way, the statement that the strong Goldbach's conjecture has an infinite number of counterexamples is certainly "unlikely". To sum up, this work represents a big step forward in showing the strong Goldbach's conjecture should be really true.

On the other hand, the question of whether there exist infinitely many twin primes has been one of the great open questions in number theory for many years. This is the content of the twin prime conjecture, which states that there are infinitely many primes p such that $p + 2$ is also prime [10]. In addition, the Dubner's conjecture is an as yet unsolved conjecture by American mathematician Harvey Dubner [7]. It states that every even number greater than 4208 is the sum of two t-primes, where a t-prime is a prime which has a twin [7]. We prove there are infinite even numbers that comply the Dubner's conjecture, where this also implies that the twin prime conjecture is true [7].

2 Background Theory

In 1936, Turing developed his theoretical computational model [18]. The deterministic and nondeterministic Turing machines have become in two of the most important definitions related to this theoretical model for computation [18]. A deterministic Turing machine has only one next action for each step defined in its program or transition function [18]. A nondeterministic Turing machine could contain more than one action defined for each step of its program, where this one is no longer a function, but a relation [18].

Let Σ be a finite alphabet with at least two elements, and let Σ^* be the set of finite strings over Σ [3]. A Turing machine M has an associated input alphabet Σ [3]. For each string w in Σ^* there is a computation associated with M on input w [3]. We say that M accepts w if this computation terminates in the accepting state, that is $M(w) = $ "*yes*" [3]. Note that M fails to accept w either if this computation ends in the rejecting state, that is $M(w) = $ "*no*", or if the computation fails to terminate, or the computation ends in the halting state with some output, that is $M(w) = y$ (when M outputs the string y on the input w) [3].

Another relevant advance in the last century has been the definition of a complexity class. A language over an alphabet is any set of strings made up of symbols from that alphabet [5]. A complexity class is a set of problems, which are represented as a language, grouped by measures such as the running time, memory, etc [5]. The language accepted by a Turing machine M, denoted $L(M)$, has an associated alphabet Σ and is defined by:

$$L(M) = \{w \in \Sigma^* : M(w) = \text{“}yes\text{”}\}.$$

Moreover, $L(M)$ is decided by M, when $w \notin L(M)$ if and only if $M(w) = $ "*no*" [5]. We use o-notation to denote an upper bound that is not asymptotically tight. We formally define $o(g(n))$ as the set

$$o(g(n)) = \{f(n) : \textit{ for any positive constant } c > 0, \textit{ there exists a constant}$$

$$n_0 > 0 \textit{ such that } 0 \leq f(n) < c \times g(n) \textit{ for all } n \geq n_0\}.$$

For example, $2 \times n = o(n^2)$, but $2 \times n^2 \neq o(n^2)$ [5].

In theoretical computer science and formal language theory, a regular language is a formal language that can be expressed using a regular expression [2]. The complexity class that contains all the regular languages is REG. The complexity class $NSPACE(f(n))$ is the set of decision problems that can be solved by a nondeterministic Turing machine M, using space $f(n)$, where n is the length of the input [13].

3 Results

▶ **Definition 1.** *We define the weak Goldbach's language L_{WG} as follows:*

$$L_{WG} = \{1^{2\times n+1}0^p0^q0^r : n \in \mathbb{N} \wedge n \geq 4 \wedge p, q \text{ and } r \text{ are odd primes } \wedge 2\times n+1 = p+q+r\}.$$

We define the strong Goldbach's language L_G as follows:

$$L_G = \{1^{2\times n}0^p0^q : n \in \mathbb{N} \wedge n \geq 3 \wedge p \text{ and } q \text{ are odd primes } \wedge 2\times n = p+q\}.$$

▶ **Theorem 2.** *If the weak Goldbach's conjecture is true, then the weak Goldbach's language L_{WG} is non-regular. Moreover, if the strong Goldbach's conjecture is true, then the strong Goldbach's language L_G is non-regular.*

Proof. If the weak Goldbach's conjecture is true, then the weak Goldbach's language L_{WG} is equal to the another language L' defined as follows:

$$L' = \{1^{2\times n+1}0^{2\times n+1} : n \in \mathbb{N} \wedge n \geq 4\}.$$

We can easily prove that L' is non-regular using the Pumping lemma for regular languages [15]. Moreover, if the strong Goldbach's conjecture is true, then the strong Goldbach's language L_G is equal to the another language L'' defined as follows:

$$L'' = \{1^{2\times n}0^{2\times n} : n \in \mathbb{N} \wedge n \geq 3\}.$$

We can easily prove that L'' is non-regular using the Pumping lemma for regular languages as well [15]. ◀

▶ **Definition 3.** *We define the weak verification Goldbach's language L_{WVG} as follows:*

$$L_{WVG} = \{(2\times n+1, p, q, r) : \text{ such that } 1^{2\times n+1}0^p0^q0^r \in L_{WG}\}.$$

We define the strong verification Goldbach's language L_{VG} as follows:

$$L_{VG} = \{(2\times n, p, q) : \text{ such that } 1^{2\times n}0^p0^q \in L_G\}.$$

▶ **Definition 4.** *We define the weak Goldbach's language with separator L_{WSG} as follows:*

$$L_{WSG} = \{0^{2\times n+1}\#0^p\#0^q\#0^r : \text{ such that } 1^{2\times n+1}0^p0^q0^r \in L_{WG}\}$$

and we define the strong Goldbach's language with separator L_{SG} as follows:

$$L_{SG} = \{0^{2\times n}\#0^p\#0^q : \text{ such that } 1^{2\times n}0^p0^q \in L_G\}$$

where $\#$ is the blank symbol.

▶ **Lemma 5.** *The weak Goldbach's language with separator L_{WSG} is the unary representation of the weak verification Goldbach's language L_{WVG}. The strong Goldbach's language with separator L_{SG} is the unary representation of the strong verification Goldbach's language L_{VG}.*

Proof. This is trivially true from the definition of these languages. ◀

▶ **Theorem 6.** *If $L_{WVG} \in NSPACE(S(n))$ for some $S(n) = o(\log n)$, then $L_{WG} \in REG$.*

Proof. In case of $L_{WVG} \in NSPACE(S(n))$ for some $S(n) = o(\log n)$, then there is a nondeterministic Turing machine which decides L_{WSG} that uses space that is smaller than $c \times \log\log n$ for all $c > 0$, because of L_{WSG} is the unary version of L_{WVG} due to Lemma 14 [8]. Certainly, the standard space translation between the unary and binary languages actually works for nondeterministic machines with small space [8]. This means that if some language belongs to $NSPACE(S(n))$, then the unary version of that language belongs to $NSPACE(S(\log n))$ [8]. In this way, we obtain that $L_{WSG} \in REG$ because of $REG = NSPACE(o(\log\log n))$ [13]. In addition, we can reduce in a nondeterministic constant space the language L_{WG} to L_{WSG} just nondeterministically inserting the blank symbol # within two arbitrary positions between the 0's on the input. Moreover, this nondeterminism reduction inserts the blank symbol # between the 1's and 0's and converts the 1's to 0's from the original input of L_{WG} just generating the final output to L_{WSG}. Consequently, we prove $L_{WG} \in REG$ under the assumption that $L_{WVG} \in NSPACE(S(n))$ for some $S(n) = o(\log n)$, since REG is also the complexity class of languages decided by nondeterministic Turing machines in constant space [16]. ◀

▶ **Theorem 7.** *$L_{WVG} \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$.*

Proof. If the weak Goldbach's conjecture is true, then $L_{WG} \notin REG$ as a consequence of Theorem 11. However, if $L_{WVG} \in NSPACE(S(n))$ for some $S(n) = o(\log n)$, then $L_{WG} \in REG$ due to Theorem 6. In this way, the weak Goldbach's conjecture cannot be true under the assumption that $L_{WVG} \in NSPACE(S(n))$ for some $S(n) = o(\log n)$. Since the weak Goldbach's conjecture is true, then we obtain that $L_{WVG} \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$ [11], [12]. ◀

The checking whether a number is prime can be decided in polynomial time by a deterministic Turing machine [1]. This problem is known as $PRIMES$ [1].

▶ **Theorem 8.** *$PRIMES \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$.*

Proof. From the Theorem 7, we obtain that $L_{WVG} \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$. However, the checking of whether the four numbers on the input are odds and proving the equality of the sum's equation can be done in $NSPACE(o(\log n))$. Certainly, the verification of the odd property could be done in constant space. In addition, the verification of the equality of the sum's equation $2 \times n + 1 = p + q + r$ can be done in $NSPACE(o(\log n))$.

Indeed, given four natural numbers p, q, r and t in binary encoding, it is obviously possible to check in $NSPACE(\log n)$ whether $p + q + r = t$. We need to go through corresponding bits from p, q, r and t starting from least significant bits to most significant bits. So for each i from 1 to n, we check if p, q, r and t have compatible/matching bits at position i (i.e. p_i, q_i, r_i, and t_i are compatible). Then, we keep track of any carry bit in constant space and move to index $i + 1$. We just need to keep track of i written in binary. If n is the greatest bit length between p, q, r and t, then we need $\log n$ bits to keep track of i. However, we can keep track of i using $o(\log n)$ space.

The position i is stored using a triple (a, b, c) of binary strings that represent positive integers. In the least significant bit position we use $(1, 0, 0)$. For a current bit position i in a triple (a, b, c), we move for the new bit position $i + 1$ using the rules of the following steps:

1. If $0 < a < \lfloor \frac{n}{\log n} \rfloor$, then the next step $i + 1$ into the new bit position is $(a + 1, b, c)$,

2. else if $a = \lfloor\frac{n}{\log n}\rfloor$, then the next step $i+1$ into the new bit position is $(0,0,1)$,
3. else if $a = 0$ then:
   a. if $c = \lfloor\log n\rfloor$, then the next step $i+1$ into the new bit position is $(a, b+1, 1)$ otherwise if $c \neq \lfloor\log n\rfloor$, then the next step $i+1$ into the new bit position is $(a, b, c+1)$.

Every triple (a, b, c) represents the bit position $a \leq \lfloor\frac{n}{\log n}\rfloor$ when $a > 0$ or $\lfloor\frac{n}{\log n}\rfloor + (\lfloor\log n\rfloor \times b) + c$ when $a = 0$. In this way, b and c always comply with $c \leq \lfloor\log n\rfloor$ and $b \leq \frac{n - \lfloor\frac{n}{\log n}\rfloor}{\lfloor\log n\rfloor}$. Certainly, this is based on the following equation

$$\lfloor\frac{n}{\log n}\rfloor + \lfloor\log n\rfloor \times \frac{n - \lfloor\frac{n}{\log n}\rfloor}{\lfloor\log n\rfloor} = n$$

However, the bit length of $\lfloor\log n\rfloor$ is bounded by $\log\lfloor\log n\rfloor$. In addition, the bit length of $\lfloor\frac{n}{\log n}\rfloor$ is bounded by $\log n - \log\log n$. Moreover, the bit length of the integer part of $\frac{n - \lfloor\frac{n}{\log n}\rfloor}{\lfloor\log n\rfloor}$ is bounded by $\log(n - \lfloor\frac{n}{\log n}\rfloor) - \log\lfloor\log n\rfloor$. Since we add the bit length of c in case of $a = 0$, then this will be $\log(n - \lfloor\frac{n}{\log n}\rfloor)$. In this way, the whole computation is bounded by $\log(n - \lfloor\frac{n}{\log n}\rfloor)$ or $\log n - \log\log n$ space. Furthermore, we use a single triple (a', b', c') to put the head into the bit positions of the binary numbers:

1. if we want to put the head of the tape into the bit position (a', b', c') inside of a binary string, then we just set the head in the least significant bit position and move to the left while we decrement in 1 the bit position using the same rules that we used for incrementing until we reach the value $(1, 0, 0)$
2. and after that, if we want to put the head of the tape into the bit position (a', b', c') inside of another binary string, then from the current position in the head tape, we move to the right while we just increment in 1 the bit position from the value $(1, 0, 0)$ using the same rules above until the head will stay in the least significant bit position of the current binary string reaching the previous value (a', b', c')
3. and while we doing that, we copy the bits p_i, q_i, r_i, and t_i to the work tapes from the bit position i that represents (a', b', c') and do the necessary verification
4. and finally, when we finish all that, then we erase the bits p_i, q_i, r_i, and t_i and create the next step $i+1$ from the value (a', b', c') into the new bit position using the same rules above.

However, we know that $\log n - \log\log n = o(\log n)$ and $\log(n - \lfloor\frac{n}{\log n}\rfloor) = o(\log n)$ for $n \geq 3$ where the whole computation can be done in a nondeterministic way because of it is indeed deterministic [14]. In addition, the ultimate remaining verification that we need to analyze in L_{WVG} is whether p, q and r are primes. Since the other properties can be done in $NSPACE(o(\log n))$ excluding the primality test and $L_{WVG} \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$, then we have as unique remaining possibility that $PRIMES \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$. ◀

▶ **Theorem 9.** *The strong Goldbach's conjecture is true or this has an infinite number of counterexamples.*

Proof. If the strong Goldbach's conjecture is false, then $L_G \in REG$ or L_G is non-regular and its complement is infinite, since every finite set is regular and REG is also closed under complement [14]. However, this implies that the exponentially more succinct version of L_G, that is L_{VG}, should be in $NSPACE(S(n))$ for some $S(n) = o(\log n)$, because of $REG = NSPACE(o(\log\log n))$ and the same algorithm that decides L_G in $NSPACE(o(\log\log n))$ could be easily transformed into a slightly modified algorithm that decides L_{VG} in $NSPACE(S(n))$

for some $S(n) = o(\log n)$ [13], [8]. Actually, L_G could be reduced to L_{SG} in a nondeterministic constant space following the steps of Theorem 6 and L_{SG} is the unary version of L_{VG} due to Lemma 14. As we mentioned before, the standard space translation between the unary and binary languages actually works for nondeterministic machines with small space [8]. This means that if some unary language belongs to $NSPACE(S(\log n))$, then the binary version of that language belongs to $NSPACE(S(n))$ [8]. It is not possible that $L_{VG} \in NSPACE(S(n))$ for some $S(n) = o(\log n)$, because of $PRIMES \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$. Certainly, the verification of whether p and q are primes need to be done in order to accept the elements of this language. Consequently, we obtain that $L_G \notin REG$, since it is not possible that $L_G \in NSPACE(o(\log \log n))$ under the result of $L_{VG} \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$. In this way, we obtain a contradiction just assuming that the strong Goldbach's conjecture is false and $L_G \in REG$. In contraposition, we have the strong Goldbach's conjecture is true or this has an infinite number of counterexamples. ◀

▶ **Definition 10.** *We define the Dubner's language L_D as follows:*

$$L_D = \{1^{2\times n}0^p 0^q : n \in \mathbb{N} \wedge n > 2104 \wedge p \textit{ and } q \textit{ are t-primes } \wedge 2 \times n = p + q\}.$$

▶ **Theorem 11.** *If the Dubner's conjecture is true, then the Dubner's language L_D is non-regular.*

Proof. If the Dubner's conjecture is true, then the Dubner's language L_D is equal to the another language L''' defined as follows:

$$L''' = \{1^{2\times n}0^{2\times n} : n \in \mathbb{N} \wedge n > 2104\}.$$

We can easily prove that L''' is non-regular using the Pumping lemma for regular languages as well [15]. ◀

▶ **Definition 12.** *We define the verification Dubner's language L_{VD} as follows:*

$$L_{VD} = \{(2 \times n, p, q) : \textit{ such that } 1^{2\times n}0^p 0^q \in L_D\}.$$

▶ **Definition 13.** *We define the Dubner's language with separator L_{SD} as follows:*

$$L_{SD} = \{0^{2\times n}\#0^p\#0^q : \textit{ such that } 1^{2\times n}0^p 0^q \in L_D\}$$

where # is the blank symbol.

▶ **Lemma 14.** *The Dubner's language with separator L_{SD} is the unary representation of the verification Dubner's language L_{VD}.*

Proof. This is trivially true from the definition of these languages. ◀

▶ **Theorem 15.** *There are infinite even numbers that comply the Dubner's conjecture.*

Proof. If the Dubner's conjecture is false, then $L_D \in REG$ or L_D is non-regular and its complement is infinite, since every finite set is regular and REG is also closed under complement [14]. However, this implies that the exponentially more succinct version of L_D, that is L_{VD}, should be in $NSPACE(S(n))$ for some $S(n) = o(\log n)$, because of $REG = NSPACE(o(\log \log n))$ and the same algorithm that decides L_D in $NSPACE(o(\log \log n))$ could be easily transformed into a slightly modified algorithm that decides L_{VD} in $NSPACE(S(n))$ for some $S(n) = o(\log n)$ [13], [8]. Actually, L_D could be reduced to L_{SD} in a nondeterministic constant space following the steps of Theorem 6 and L_{SD} is the unary version of L_{VD} due to

Lemma 14. As we mentioned before, the standard space translation between the unary and binary languages actually works for nondeterministic machines with small space [8]. This means that if some unary language belongs to $NSPACE(S(\log n))$, then the binary version of that language belongs to $NSPACE(S(n))$ [8]. It is not possible that $L_{VD} \in NSPACE(S(n))$ for some $S(n) = o(\log n)$, because of $PRIMES \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$. Certainly, the verification of whether p and q are t-primes need to be done in order to accept the elements of this language. Consequently, we obtain that $L_D \notin REG$, since it is not possible that $L_D \in NSPACE(o(\log \log n))$ under the result of $L_{VD} \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$. In this way, we obtain a contradiction just assuming that the Dubner's conjecture is false and $L_D \in REG$. In contraposition, we have the Dubner's conjecture is true or there are infinite even numbers that comply the Dubner's conjecture, since in case of L_D would be finite, then we obtain that $L_D \in REG$ and we prove that is not possible. ◀

▶ **Lemma 16.** *The twin prime conjecture is true.*

Proof. The Theorem 15 implies that there exists an infinite number of t-primes, and thus there will be an infinite number of twin prime pairs as well [7]. ◀

References

1. Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004. `doi:10.4007/annals.2004.160.781`.
2. Alfred V. Aho and John E. Hopcroft. *The Design and Analysis of Computer Algorithms*. Pearson Education India, 1974.
3. Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
4. Jing-run Chen. On the Representation of a Large Even Integer as the Sum of a Prime and the Product of Two Primes at Most. *Sci. Sinica*, 16:157–176, 1973.
5. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3 edition, 2009.
6. Leonard Eugene Dickson. *History of the Theory of Numbers: Divisibility and Primality*, volume 1. New York, Dover, 2005.
7. Harvey Dubner. Twin prime conjectures. *JOURNAL OF RECREATIONAL MATHEMATICS*, 30(3):199–205, 2000.
8. Viliam Geffert and Dana Pardubská. Unary Coded NP-Complete Languages in ASPACE (log log n). *International Journal of Foundations of Computer Science*, 24(07):1167–1182, 2013. `doi:10.1007/978-3-642-31653-1_16`.
9. Richard K. Guy. *Unsolved Problems in Number Theory*. New York, Springer-Verlag, 3 edition, 2004.
10. Godfrey Harold Hardy and Edward Maitland Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, 1979.
11. Harald A. Helfgott. Minor arcs for Goldbach's problem. *arXiv preprint arXiv:1205.5252*, 2012.
12. Harald A. Helfgott. Major arcs for Goldbach's theorem. *arXiv preprint arXiv:1305.2897*, 2013.
13. Pascal Michel. A survey of space complexity. *Theoretical computer science*, 101(1):99–132, 1992. `doi:10.1016/0304-3975(92)90151-5`.
14. Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
15. Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM journal of research and development*, 3(2):114–125, 1959. `doi:10.1147/rd.32.0114`.
16. John C. Shepherdson. The Reduction of Two-Way Automata to One-Way Automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959. `doi:10.1147/rd.32.0198`.
17. Joseph H. Silverman. *A Friendly Introduction to Number Theory*. Pearson Education, Inc., 4 edition, 2012.

18 Michael Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

19 Ivan M. Vinogradov. Representation of an Odd Number as a Sum of Three Primes. *Comptes Rendus (Doklady) de l'Académie des Sciences de l'USSR*, 15:169–172, 1937.

20 David G. Wells. *Prime Numbers, The Most Mysterious Figures in Math.* John Wiley & Sons, Inc., 2005.