Sums of Palindromes: an Approach via Nested-Word Automata

Aayush Rajasekaran, Jeffrey Shallit, and Tim Smith
School of Computer Science
University of Waterloo
Waterloo, ON N2L 3G1
Canada
{arajasekaran,shallit,timsmith}@uwaterloo.ca

June 30, 2017

Abstract

We prove, using a decision procedure based on nested-word automata, that every natural number is the sum of at most 9 natural numbers whose base-2 representation is a palindrome. We consider some other variations on this problem, and prove similar results. We argue that heavily case-based proofs are a good signal that a decision procedure may help to automate the proof.

1 Introduction

In this paper we combine three different themes: (i) additive number theory; (ii) numbers with special kinds of representations in base k; (iii) use of a decision procedure to prove theorems. We prove, for example, that every natural number is the sum of at most 9 numbers whose base-2 representation is a palindrome.

Additive number theory is the study of the additive properties of integers. For example, Lagrange proved (1770) that every natural number is the sum of four squares [33]. In additive number theory, a subset $S \subseteq \mathbb{N}$ is called a *additive basis of order* h if every element of $\mathbb{N}$ can be written as a sum of at most h members of S, not necessarily distinct.

Waring's problem asks for the smallest value $g(k)$ such that the k'th powers form a basis of order $g(k)$. In a variation on Waring's problem, one can ask for the smallest value $G(k)$ such that every *sufficiently large* natural number is the sum of $G(k)$ k'th powers [63]. This kind of representation is called an *asymptotic additive basis* of order $G(k)$.

Quoting Nathanson [50, p. 7],

> *"The central problem in additive number theory is to determine if a given set of integers is a basis of finite order."*

In this paper we show how to solve this central problem for certain sets, *using almost no number theory at all*.

Our second theme concerns numbers with special representations in base k. For example, numbers of the form $11 \cdots 1$ in base k are sometimes called *repunits* [64], and special effort has been devoted to factoring such numbers, with the Mersenne numbers $2^n - 1$ being the most famous examples. The *Nagell-Ljunggren problem* asks for a characterization of those repunits that are integer powers (see, e.g., [57]).

Another interesting class, and the one that principally concerns us in this article, consists of those numbers whose base-k representation forms a *palindrome*: a string that reads the same forwards and backwards, like the English word `radar`. Palindromic numbers have been studied for some time in number theory; see, for example, [59, 61, 42, 7, 39, 12, 41, 34, 44, 9, 37, 10, 11, 40, 45, 23, 21, 31, 46, 13, 22, 15, 14, 54, 2].

Recently Banks initiated the study of the additive properties of palindromes, proving that every natural number is the sum of at most 49 numbers whose decimal representation is a palindrome [8]. (Also see [58].) Banks' result was recently improved by Cilleruelo, Luca, & Baxter [19, 20], who proved that for all bases $b \geq 5$, every natural number is the sum of at most 3 numbers whose base-b representation is a palindrome. The proofs of Banks and Cilleruelo, Luca, & Baxter are both rather lengthy and case-based. Up to now, there have been no proved results for bases $b = 2, 3, 4$.

The long case-based solutions to the problem of representation by sums of palindromes suggests that perhaps a more automated approach might be useful. For example, in a series of recent papers, the second author and his co-authors have proved a number of old and new results in combinatorics on words using a decision procedure based on first-order logic [1, 18, 27, 56, 29, 28, 30, 25, 49, 24]. The classic result of Thue [60, 16] that the Thue-Morse infinite word $\mathbf{t} = 0110100110010110 \cdots$ avoids overlaps (that is, blocks of the form $axaxa$ where a is a single letter and x is a possibly empty block) is an example of a case-based proof that can be entirely replaced [1] with a decision procedure based on the first-order logical theory $\text{FO}(\mathbb{N}, +, V_2)$.

Inspired by these and other successes in automated deduction and theorem-proving (e.g., [47]), we turn to formal languages and automata theory as a suitable framework for expressing the palindrome representation problem. Since we want to make assertions about the representations of *all* natural numbers, this requires finding (a) a machine model or logical theory in which universality is decidable and (b) a variant of the additive problem of palindromes suitable for this machine model or logical theory. The model we use is the *nested-word automaton*, a variant of the more familiar pushdown automaton.

Our paper is organized as follows: In Section 2 we introduce some notation and terminology, and state more precisely the problem we want to solve. In Section 3 we recall the pushdown automaton model and give an example, and we motivate our use of nested-word automata. In Section 4 we restate our problem in the framework of nested-word automata, and the proof of a bound of 11 palindromes is given in Section 5. This bound is improved to 9 in Section 6. The novelty of our approach involves replacing the long case-based reasoning of previous proofs with an automaton-based approach using a decision procedure. In Section 7 we consider some variations on the original problem. In Section 8 we discuss possible objections to our approach. In Section 9 we describe future work. In Section 10 we mention related problems that do not seem amenable to our approach. Finally, in Section 11, we conclude our paper by stating a thesis underlying our approach.

2 The sum-of-palindromes problem

We first introduce some notation and terminology.

The natural numbers are $\mathbb{N} = \{0, 1, 2, \ldots\}$. If n is a natural number, then by $(n)_k$ we mean the string (or word) representing n in base k, with no leading zeroes, starting with the most significant digit. Thus, for example, $(43)_2 = 101011$. The alphabet Σ_k is defined to be $\{0, 1, \ldots, k-1\}$; by Σ_k^* we mean the set of all finite strings over Σ_k. If $x \in \Sigma_\ell^*$ for some ℓ, then by $[x]_k$ we mean the integer represented by the string x, considered as if it were a number in base k, with the most significant digit at the left. That is, if $x = a_1 a_2 \cdots a_n$, then $[x]_k = \sum_{1 \le i \le n} a_i k^{n-i}$. For example, $[135]_2 = 15$.

If x is a string, then x^i denotes the string $\overbrace{xx\cdots x}^{i}$, and x^R denotes the reverse of x. Thus, for example, $(\texttt{ma})^2 = \texttt{mama}$, and $(\texttt{drawer})^R = \texttt{reward}$. If $x = x^R$, then x is said to be a *palindrome*.

We are interested in integers whose base-k representations are palindromes. In this article, we routinely abuse terminology by calling such an integer a *base-k palindrome*. In the case where $k = 2$, we also call such an integer a *binary palindrome*. The first few binary palindromes are

$$0, 1, 3, 5, 7, 9, 15, 17, 21, 27, 31, 33, 45, 51, 63, \ldots;$$

these form sequence A006995 in the *On-Line Encyclopedia of Integer Sequences* (OEIS).

If $k^{n-1} \le r < k^n$ for $n \ge 1$, we say that r is an *n-bit integer* in base k. If k is unspecified, we assume that $k = 2$. Note that the first bit of an n-bit integer is always nonzero. The *length* of an integer r satisfying $k^{n-1} \le r < k^n$ is defined to be n; alternatively, the length of r is $1 + \lfloor \log_k r \rfloor$.

Our goal is to find a constant c such that every natural number is the sum of at most c binary palindromes. To the best of our knowledge, no such bound has been proved up to now. In Sections 4 and 5 we describe how we used a decision procedure for nested-word automata to prove the following result:

Theorem 1. *For all $n \ge 6$ with n even, every n-bit integer is the sum of at most 10 binary palindromes of length exactly $n - 3$.*

As a corollary, we get the following result:

Corollary 2. *Every natural number N is the sum of at most 11 binary palindromes.*

Proof. It is routine to verify the result for $N < 64$. In fact, for these numbers at most 3 palindromes are needed.

Now suppose $N \geq 64$. Say that N is an n-bit integer. Then $n \geq 6$. If n is even, then Theorem 1 states that N is the sum of at most 10 binary palindromes.

Otherwise, suppose N is an n-bit integer with n odd and $n \geq 7$. There are three cases to consider.

(i) If $2^{n-1} \leq N \leq 3 \cdot 2^{n-2} - 2$, then set $N' = N - (2^{n-2} - 1)$. Then $2^{n-2} + 1 \leq N' \leq 2^{n-1} - 1$, so N' is an $(n-1)$-bit integer. Theorem 1 gives a representation for N' as the sum of at most 10 palindromes, so since $2^{n-2} - 1$ itself is a binary palindrome, it follows that N has a representation as the sum of at most 11 palindromes.

(ii) If $3 \cdot 2^{n-2} - 1 \leq N \leq 2^n - 2$, then set $N' = N - (2^{n-1} - 1)$. Then $2^{n-2} \leq N' \leq 2^{n-1} - 1$, so again N' is an $(n-1)$-bit integer, and again Theorem 1 gives a representation for N' as the sum of 10 palindromes, so since $2^{n-1} - 1$ itself is a binary palindrome, it follows that N has a representation as the sum of at most 11 palindromes.

(iii) If $N = 2^n - 1$, then N is already a palindrome. □

3 Finding an appropriate computational model

To find a suitable model for proving Theorem 1, we turn to formal languages and automata. We seek some class of automata with the following property: for each k, there is an automaton which, given a natural number n as input, accepts the input iff n can be expressed as the sum of k palindromes. Furthermore, we would like the problem of universality ("Does the automaton accept every possible input?") to be decidable in our chosen model. By constructing the appropriate automaton and checking whether it is universal, we could then determine whether every number n can be expressed as the sum of k palindromes.

Palindromes suggest considering the model of pushdown automaton (PDA), since it is well-known that this class of machines, equipped with a stack, can accept the palindrome language $\texttt{PAL} = \{x \in \Sigma^* \ : \ x = x^R\}$ over any fixed alphabet Σ. A tentative approach is as follows: create a PDA M that, on input n expressed in base 2, uses nondeterminism to "guess" the k summands and verify that (a) every summand is a palindrome, and (2) they sum to the input n. We would then check to see if M accepts all of its inputs. However, two problems immediately arise.

The first problem is that universality is recursively unsolvable for nondeterministic PDAs [38, Thm. 8.11, p. 203], so even if the automaton M existed, there would be no algorithm guaranteed to check universality.

The second problem involves checking that the guessed summands are palindromes. One can imagine guessing the summands in parallel, or in series. If we try to check them in parallel, this seems to correspond to the recognition of a language which is not a CFL (i.e., a context-free language, the class of languages recognized by nondeterministic PDAs). Specifically, we encounter the following obstacle:

Theorem 3. *The set of strings L over the alphabet $\Sigma \times (\Sigma \cup \#)$, where the first "track" is a palindrome and the second "track" is a another, possibly shorter, palindrome, padded on the right with # signs, is not a CFL.*

Proof. Assume that it is. Consider L intersected with the regular language

$$[1,1]^+[1,0][1,1]^+[0,1][1,\#]^+,$$

and call the result L'. We use Ogden's lemma [51] to show L' is not a CFL.

Choose z to be a string where the first track is $(1^{2n}01^{2n})$ and the second track is $(1^n01^n\#^{2n})$. Mark the compound symbols $[1,\#]$. Then every factorization $z = uvwxy$ must have at least one $[1,\#]$ in v or x. If it's in v, then the only choice for x is also $[1,\#]$, so pumping gives a non-palindrome on the first track. If it's in x then v can be $[1,1]^i$ or contain $[1,0]$ or $[0,1]$. If the latter, pumping twice gives a string not in L' because there is more than one 0 on one of the two tracks. If the former, pumping twice gives a string with the second track not a palindrome. This contradiction shows that L', and hence L, is not a context-free language. □

So, using a pushdown automaton, we cannot check arbitrary palindromes of wildly unequal lengths in parallel.

If the summands were presented serially, we could check whether each summand individually is a palindrome, using the stack, but doing so destroys our copy of the summand, and so we cannot add them all up and compare them to the input. In fact, we cannot add serial summands in any case, because we have

Theorem 4. *The language*

$$L = \{(m)_2\#(n)_2\#(m+n)_2 \ : \ m,n \geq 0\}$$

is not a CFL.

Proof. Assume L is a CFL and intersect with the regular language $1^+01^+\#1^+\#1^+0$, obtaining L'. We claim that

$$L' = \{1^a01^b\#1^c\#1^d0 \ : \ b = c \text{ and } a + b = d\}.$$

This amounts to the claim, easily verified, that the only solutions to the equation $2^{a+b+1} - 2^b - 1 + 2^c - 1 = 2^{d+1} - 2$ are $b = c$ and $a + b = d$. Then, starting with the string $z = 1^n01^n\#1^n\#1^{2n}0$, an easy argument with Ogden's lemma proves that L' is not a CFL, and hence neither is L. □

So, using a pushdown automaton, we can't handle summands given in series, either.

These issues lead us to restrict our attention to representations as sums of palindromes of the same (or similar) lengths. More precisely, we consider the following variant of the additive problem of palindromes: for a length l and number of summands k, given a natural number n as input, is n the sum of k palindromes all of length exactly l? Since the palindromes are all of the same length, a stack would allow us to guess and verify them in parallel. To tackle this problem, we need a model which is both (1) powerful enough to handle our new variant, and (2) restricted enough that universality is decidable. We find such a model in the class of *nested-word automata*, described in the next section.

4 Restating the problem in the language of nested-word automata

Nested-word automata (NWAs) were popularized by Alur and Madhusudan [3, 4], although essentially the same model was discussed previously by Mehlhorn [48], von Braunmühl and Verbeek [17], and Dymond [26]. They are a restricted variant of pushdown automata. Readers familiar with visibly pushdown automata (VPA) should note that NWAs are an equally powerful machine model as VPAs [3, 4]. We only briefly describe their functionality here. For other theoretical aspects of nested-word and visibly-pushdown automata, see [43, 53, 32, 55, 52].

The input alphabet of an NWA is partitioned into three sets: a *call alphabet*, an *internal alphabet*, and a *return alphabet*. An NWA has a stack, but has more restricted access to it than PDAs do. If an input symbol is from the internal alphabet, the NWA cannot access the stack in any way. If the input symbol read is from the call alphabet, the NWA pushes its current state onto the stack, and then performs a transition, based only on the current state and input symbol read. If the input symbol read is from the return alphabet, the NWA pops the state at the top of the stack, and then performs a transition based on three pieces of information: the current state, the popped state, and the input state read. An NWA accepts if the state it terminates in is an accepting state.

As an example, Figure 1 illustrates a nested-word automaton accepting the language $\{0^n 1 2^n \ : \ n \geq 1\}$.

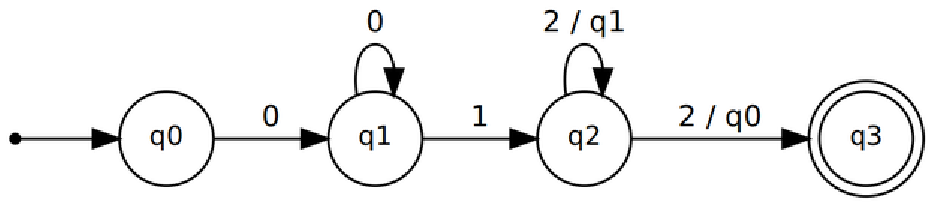

Figure 1: A nested-word automaton for the language $\{0^n 1 2^n \; : \; n \geq 1\}$

Here the call alphabet is $\{0\}$, the internal alphabet is $\{1\}$, and the return alphabet is $\{2\}$.

Nondeterministic NWAs are a good machine model for our problem, because nondeterminism allows "guessing" the palindromes that might sum to the input, and the stack allows us to "verify" that they are indeed palindromes. Deterministic NWAs are as expressive as nondeterministic NWAs, and the class of languages they accept is closed under the operations of union, complement and intersection. Finally, testing emptiness, universality, and language inclusion are all decidable problems for NWAs [3, 4]; there is an algorithm for all of them.

For a nondeterministic NWA of n states, the corresponding determinized machine has at most $2^{\Theta(n^2)}$ states, and there are examples for which this bound is attained. This very rapid explosion in state complexity potentially could make deciding problems such as language inclusion infeasible in practice. Fortunately, we did not run into determinized machines with more than 25000 states in proving our results. Most of the algorithms invoked to prove our results run in under a minute.

We now discuss the general construction of the NWAs that check whether inputs are sums of binary palindromes. We use the following observation: a number n is the sum of exactly t length-ℓ binary palindromes if and only if $n = [w]_2$, where w is a length-ℓ palindrome starting with t over the alphabet $\Sigma_{t+1} = \{0, 1, \ldots, t\}$. It therefore suffices to guess a *single* palindrome over the larger alphabet Σ_{t+1}, "normalize" it on the fly by considering it as a number in base 2, and compare the normalized result to the input symbol-by-symbol. Normalization involves propagating carries, if any, from the least-significant digit to the most significant digit.

We partition the input alphabet into the call alphabet $\{a, b\}$, the internal alphabet $\{c, d\}$, and the return alphabet $\{e, f\}$. The symbols a, c, and e correspond to 0, while b, d, and f correspond to 1. The input string is fed to the machine starting with the *least significant digit*. We provide the NWA with input strings whose first half is entirely made of call symbols, and second half is entirely made of return symbols. Internal symbols are used to create a divider between the halves (for the case of odd-length palindromes), and for the most significant digits of the input that could be produced by a terminal carry.

The idea behind the NWA is to nondeterministically guess all possible summands when reading the first half of the input string. The guessed summands are characterized by the states pushed onto the stack. The machine then checks if the guessed summands can produce the input bits in the second half of the string. The machine keeps track of any carries in the current state.

For example, to check whether $96 = [1100000]_2$ is the sum of two binary palindromes of length 6, the input to our NWA would be $aaaeefd$. (Recall that the input starts with the least significant digit. The block aaa is the call half, eef is the return half, and d is left over for the terminal carry.) As the NWA reads the first three as, it makes corresponding guesses over the alphabet $\Sigma_3 = \{0, 1, 2\}$. For example, its guesses could be 2, 1 and 1. The machine then confirms that the next three symbols, eef, correspond to 1, 1 and 2, normalizing to base 2 and propagating carries as it goes. Finally, our NWA has a carry of 1 after reading the first 6 letters of the input, which corresponds to the last input symbol, d. The machine finishes in an accepting state, corresponding (for example) to the decomposition of 96 as the sum of the two binary palindromes 51 and 45.

To check the correctness of our NWAs, we built an NWA-simulator, and then ran simulations of the machines on various types of inputs, which we then checked against experimental results.

For instance, we built the machine that accepts representations of integers that can be expressed as the sum of 2 binary palindromes. We then simulated this machine on every integer from 513 to 1024, and checked that it only accepts those integers that we experimentally confirmed as being the sums of 2 binary palindromes.

The general procedure to prove our results is to build an NWA `PalSum` accepting only those inputs that it verifies as being appropriate sums of palindromes, as well as an NWA `SyntaxChecker` accepting all valid representations. We then run the decision algorithms for language inclusion, language emptiness, etc. on `PalSum` and `SyntaxChecker` as needed. To do this, we used the Automata Library toolchain of the ULTIMATE program analysis framework [35, 36].

We have provided links to the proof scripts used to establish all of our results. To run these proof scripts, simply copy the contents of the script into `https://monteverdi.informatik.uni-freiburg.de/tomcat/Website/?ui=int&tool=automata_library` and click "execute".

5 Proving Theorem 1

In this section, we discuss construction of the appropriate nested-word automaton in more detail.

Proof. (of Theorem 2) First, we note that for an n-bit integer to be the sum of palindromes of length $n-3$, adding the palindromes must produce a carry of at least 4. The number of summands must thus lie between 5 and 10, inclusive. We build a separate automaton for each case, for a total of 6 automata. We call them `palCheckerFIVE`, `palCheckerSIX`, `palCheckerSEVEN`, `palCheckerEIGHT`, `palCheckerNINE` and `palCheckerTEN`. The total number of states in the nondeterminized versions of these 6 automata is 469. We then determinize them, and take their union, to get a single deterministic NWA, `FinalAut`, with 21519 states.

The language of valid inputs to our automata is given by

$$L = \{\{a, b\}^n\{c, d\}\{e, f\}^n\{c, d\}^2 d \ : \ n \geq 1\}.$$

The last three symbols are taken from the internal alphabet, since these bits are produced by the carry.

We detail the mechanism of `palCheckerFIVE` as an example. The states of `palCheckerFIVE` include 30 states labeled $f_{i,j}$, for $0 \leq i \leq 4$, $0 \leq j \leq 5$. The i-value represents the current carry of the automaton, while the j-coordinate represents the next guess that the automaton makes. A state f_{i_1,j_1} has a transition to f_{i_2,j_2} on input $p \in \{a, b\}$, if $i_1 + j_1$ produces an output bit corresponding to p and a new carry of i_2. Note that the value of j_2 is irrelevant to determining this transition. For example, $f_{1,3}$ has a transition to $f_{2,0}$ on input a, because adding 1 and 3 produces a 0 (which is an a), and yields a fresh carry of 2. The state $f_{1,3}$ also has transitions to $f_{2,1}$, $f_{2,2}$ $\ldots$ $f_{2,5}$ on input a.

The machine `palCheckerFIVE` also has states labeled s_i, for $0 \leq i \leq 4$. These states process the second portion of the input string. Let the input symbol be $q \in \{e, f\}$, and the state at the top of the stack be $f_{x,y}$. The state s_i has a transition to s_j if $i + y$ produces an output bit corresponding to q and a new carry of j. The NWA thus pushes all possible guesses onto the stack while processing the first portion of the input string, and then pops those guesses off the stack, one at a time, checking whether the popped guesses can produce the bits in the second portion of the string.

The machine's initial state is $f_{0,5}$, because we start with no carry, and our guessed palindromes must all start with a 1. The transitions from f-states to s-states occur on the dividing letter from the internal alphabet. On input $r \in \{c, d\}$, we take a transition from $f_{i,j}$ to s_k if there is any $0 \leq m \leq 5$ such that $i + m$ produces the bit corresponding to r with a carry of k. The machine's lone accepting state can only be reached by reading ccd from the state s_4.

The functioning of the other automata that accept integers requiring 6 to 10 summands is very similar to `palCheckerFIVE`. We also build an NWA called `syntaxChecker`, which accepts L, the language of valid inputs. Finally, we assert that L is a subset of the language accepted by `FinalAut`. This verifies that for all even $n \geq 6$, all valid representations of n-bit integers, can be expressed as the sum of at most 10 binary palindromes of length $n - 3$.

The complete script executing this proof can be found at `https://cs.uwaterloo.ca/~shallit/papers.html`. It is worth noting that a state labeled as $f_{2,3}$ in this report is labeled `f_2_THREE` in the proof script. Also, ULTIMATE does not currently have a union operation for NWAs, so we work around this by using De Morgan's laws for complement and intersection. □

Proposition 5. *Theorem 1 is optimal, in the following sense: for $n \geq 6$, the number $2^n - 2$ is not the sum of 9 binary palindromes of length $n - 3$.*

Proof. We carry out a similar machine-based proof as before. The proof script can be found at `https://cs.uwaterloo.ca/~shallit/papers.html`. There are three steps:

1. Construct the automaton, `FinalAut`, accepting integers of length n that are sums of 9 or fewer palindromes of length $n-3$. Its determinized version has 15305 states.
2. Construct the automaton, `OnesZeroAcceptor`, accepting $2^n - 2$ for even $n \geq 6$.
3. Assert that the intersection of the languages accepted by `FinalAut` and `OnesZeroAcceptor` is empty.

□

6 An improved bound

In this section we show how to improve the bound of 11 for binary palindromes, obtained in the previous section, to 9.

We start with the following:

Theorem 6. *Every length-n even integer N, where $n \geq 8$ is even, is the sum of exactly 7 numbers, six of which are binary palindromes of length $n-3$, and the seventh number of the form 2^i for $1 \leq i < n$.*

Proof. As before, we create a NWA and verify the claim. Our machine has 7582 states. The code is available at `https://cs.uwaterloo.ca/~shallit/papers.html`. □

Corollary 7. *Every natural number can be written as the sum of at most 9 binary palindromes.*

Proof. Similar to the proof of Corollary 2. As before, the assertion is easily verified for $N < 256$. So assume N is a length-n integer with $n \geq 8$.

(a) If N is a length-n integer with N odd and $n \geq 8$ even, then we get a bound of 7 by using Corollary 7 and adding one; the 7th palindrome is then $2^i + 1$.

(b) If $N \neq 2^{n-1}$ is a length-n integer with both N and $n \geq 8$ even, then we get a bound of 8 by using (a) on $N' = N - 1$ and adding 1.

(c) If N is a length-n integer with N even and $n \geq 9$ odd, then we use (a) on either $N' - (2^{n-2} - 1)$ or $N' - (2^{n-1} - 1)$ to get a representation for N using 8 palindromes.

(d) If $N \neq 2^{n-1} - 1$ is a length-n integer with both N odd and $n \geq 9$ odd, then we use (b) on either $N' - (2^{n-2} - 1)$ or $N' - (2^{n-1} - 1)$ to get a representation for N using 9 palindromes.

(e) Finally, if N is of the form 2^{n-1}, then N is the sum of the two palindromes $2^{n-1} - 1$ and 1. If N is of the form $2^{n-1} - 1$, then N is a palindrome itself.

□

Our bound of 9 is almost certainly not optimal. We may be able to improve it using a variation on these ideas. We make the following conjectures.

Conjecture 8. Every natural number is the sum of at most 4 binary palindromes, and the number 4 is optimal.

Conjecture 9. Every odd natural number is the sum of at most 3 binary palindromes, and the number 3 is optimal.

Remark 10. Sequence A261678 in the OEIS lists those even numbers that are not the sum of two binary palindromes. Sequence A261680 gives the number of distinct representations as the sum of four binary palindromes.

7 Variations on the original problem

In this section we consider some variations on the original problem. Our first variation involves the notion of generalized palindromes.

7.1 Generalized palindromes

We define a *generalized palindrome of length* n to be an integer whose base-k representation, extended, if necessary, by adding leading zeroes to make it of length n, is a palindrome. For example, 12 is a generalized binary palindrome of length 6, since its length-6 representation 001100 is a palindrome. If a number is a generalized palindrome of any length, we call it a *generalized palindrome*. The first few binary generalized palindromes are

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15, 16, 17, 18, 20, 21, 24, 27, 28, 30, 31, 32, \ldots;$$

they form sequence A057890 in the OEIS.

We can use our method to prove

Theorem 11. *Every natural number is the sum of at most 5 generalized binary palindromes.*

Proof. For $N < 16$, the result can easily be checked by hand. In fact, in this case, at most 2 generalized palindromes are needed.

We then use our method to prove the following claims:

(a) For n even, $n \geq 2$, all length-n numbers are the sum of exactly 4 generalized binary palindromes of length $n - 1$;

(b) For n odd, $n \geq 5$, all length-n numbers are the sum of exactly 5 generalized binary palindromes of length $n-2$.

To prove (a) we build the automaton accepting valid representations of n-bit integers, for an even n, that are the sum of exactly 4 generalized palindromes of length $n-1$. Note that allowing generalized palindromes requires us to allow all f-states with no carry as initial states, since we do not need the first guessed bit to be 1. The determinized automaton has 285 states. We then build an automaton accepting valid representations as before, and assert that the languages accepted by these two automata are equal. The proof script can be found at `https://cs.uwaterloo.ca/~shallit/papers.html`.

To prove (b) we build the automaton accepting valid representations of n-bit integers, for an odd n, that are the sum of exactly 5 generalized palindromes of length $n-2$. The determinized automaton has 484 states. We then build an automaton accepting valid representations as before, and assert that the languages accepted by these two automata are equal. The proof script can be found `https://cs.uwaterloo.ca/~shallit/papers.html`. □

As an aside, we also mention the following enumeration result.

Theorem 12. *There are exactly $3^{\lceil n/2 \rceil}$ natural numbers that are the sum of two generalized binary palindromes of the same length n.*

Proof. Take two generalized binary palindromes of length n and add them bit-by-bit. Then each digit position is either $0, 1$, or 2, and the result is still a palindrome. Hence there are at most $3^{\lceil n/2 \rceil}$ such numbers. It remains to see they are all distinct.

We show that if x is a length-n word that is a palindrome over $\{0, 1, 2\}$, then the map that sends x to its evaluation in base 2 is injective.

We do this by induction on the length of the palindrome. The claim is easily verified for length 0 and 1. Now suppose x and y are two distinct strings of the same length that evaluate to the same number in base 2. Then, by considering everything mod 2, we see that either

(a) x and y both end in the same number i, or

(b) (say) x ends in 0 and y ends in 2.

In case (a) we know that x and y, since they are both palindromes, both begin and end in i. So subtracting $i00 \cdots 0i$ from x and y we get words x', y' of length $n-2$ that evaluate to the same thing, and we can apply induction to see $x' = y'$.

In case (b), say $x = 0x'0$ and $y = 2y'2$, and they evaluate to the same thing. However, the largest x could be is $0222 \cdots 20$ and the smallest y could be is $200 \cdots 02$. The difference between these two numbers is 6, so we conclude that this is impossible. □

Remark 13. Not every natural number is the sum of 2 generalized binary palindromes; the smallest exception is 157441. The list of exceptions forms sequence A278862 in the OEIS. It does not seem to be known currently whether there are infinitely many exceptions. We conjecture that every natural number is the sum of 3 generalized binary palindromes.

7.2 Antipalindromes

Another generalization concerns antipalindromes. Define a map from Σ_2^* to Σ_2^* by changing every 0 to 1 and vice-versa; this is denoted $\overline{w}$. Thus, for example, $\overline{010} = 101$. An *antipalindrome* is a word w such that $w = \overline{w^R}$. It is easy to see that if a word w is an antipalindrome, then it must be of even length. We call an integer n an antipalindrome if its canonical base-2 expansion is an antipalindrome. The first few antipalindromic integers are

$$2, 10, 12, 38, 42, 52, 56, 142, 150, 170, 178, 204, 212, 232, 240, \ldots;$$

they form sequence A035928 in the OEIS. Evidently every antipalindrome must be even.

To use NWAs to prove results about antipalindromes, we slightly modify our machine structure. While processing the second portion of the input string, we complement the number of ones we guessed in the first portion. Assume we are working with n summands, and let the input symbol be $q \in \{e, f\}$, and the state at the top of the stack be $f_{x,y}$. The state s_i has a transition to s_j if $i + (n - y)$ produces an output bit corresponding to q and a new carry of j. We also set the initial state to be $f_{0,0}$, because the least significant bit of an antipalindrome must be 0.

Using our method, we would like to prove the following conjecture. However, so far we have been unable to complete the computations because our program runs out of space. In a future version of this paper, we hope to change the status of the following conjecture to a theorem.

Conjecture 14. Every even integer of length n, for n odd and $n \geq 9$, is the sum of at most 10 antipalindromes of length $n - 3$.

Corollary 15. *(Conditional only on the truth of Conjecture 14) Every even natural number N is the sum of at most* 13 *antipalindromes.*

Proof. For $N < 256$ this can be verified by a short program. (In fact, for these n, only 4 antipalindromes are needed.) Otherwise assume $N \geq 256$. Then N is of length $n \geq 9$. If n is odd, we are done by Theorem 14. Otherwise, n is even. Then, by subtracting at most 3 copies of the antipalindrome $2^{n-2} - 2^{\frac{n}{2}-1}$ from N, we obtain N' even, of length $n - 1$. We can then apply Theorem 14 to N'. □

The 13 in Corollary 15 is probably quite far from the optimal bound. Numerical evidence suggests

Conjecture 16. Every even natural number is the sum of at most 4 antipalindromes.

Conjecture 17. Every even natural number except

$$8, 18, 28, 130, 134, 138, 148, 158, 176, 318, 530, 538, 548, 576, 644, 1300, 2170, 2202, 2212, 2228,$$
$$2230, 2248, 8706, 8938, 8948, 34970, 35082$$

is the sum of at most 3 antipalindromes.

7.3 Generalized antipalindromes

Similarly, one could consider "generalized antipalindromes"; these are numbers whose base-2 expansion becomes an antipalindrome if a suitable number of leading zeroes are added. The notion of length here is the same as in Section 7.1.

Theorem 18. *Every natural number of length n, for $n \geq 6$ and even, is the sum of exactly 6 generalized antipalindromes of length $n-2$.*

Proof. Since generalized antipalindromes can have leading zeroes, we allow all f-states with no carry as initial states. We also complement the number of ones for the second half, as we did when handling regular antipalindromes.

The automated proof of (a) can be found at `https://cs.uwaterloo.ca/~shallit/papers.html`. The determinized automaton has 2254 states.

□

Corollary 19. *Every natural number is the sum of at most 7 generalized antipalindromes.*

Proof. Just like the proof of Corollary 2, using Theorem 18. □

Remark 20. Corollary 19 is probably not best possible. The correct bound seems to be 3. The first few numbers that do not have a representation as the sum of 2 generalized antipalindromes are

$$29, 60, 91, 109, 111, 121, 122, 131, 135, 272, 329, 347, 365, 371, 373, 391, 401, 429, 441, 445, 449, 469, 473, 509, 531, 539, 546, 577, 611, 660, 696, 731, 744, 791, 804, 884, 905, 940, 985, 1011, 1020, 1045, \ldots$$

8 Objections to this kind of proof

A proof based on computer calculations, like the one we have presented here, is occasionally criticized because it cannot easily be verified by hand, and because it relies on software that has not been formally proved. These kinds of criticisms are not new; they date at least to the 1970's, in response to the celebrated proof of the four-color theorem by Appel and Haken [5, 6]. See, for example, Tymoczko [62].

We answer this criticism in several ways. First, it is not reasonable to expect that every result of interest to mathematicians will have short and simple proofs. There may well be, for example, easily-stated results for which the shortest proof possible in a given axiom system is longer than any human mathematician could verify in their lifetime, even if every waking hour were devoted to checking it. For these kinds of results, an automated checker may be our only hope. There are many results for which the only proof currently known is computational.

Second, while short proofs can easily be checked by hand, what guarantee is there that any very long case-based proof — whether constructed by humans or computers — can always be certified by human checkers with a high degree of confidence? There is always the potential that some case has been overlooked. Indeed, the original proof by Appel and Haken apparently overlooked some cases. Similarly, the original proof by Cilleruelo & Luca on sums of palindromes [19] had some minor flaws that became apparent once their method was implemented as a `python` program.

Third, confidence in the correctness of the results can be improved by providing code that others may check. Transparency is essential. To this end, we have provided our code for the nested-word automata, and the reader can easily run this code on the software we referenced.

9 Future work

This is a preliminary report. In a later version of the paper, we hope to address the case of other bases. For example, a natural conjecture is the following:

Conjecture 21. Every natural number is the sum of at most 3 palindromes in bases 3 and 4.

Although we probably can cannot prove this using our methods, the following is a reasonable candidate for our approach:

Conjecture 22. Every length-n integer, n even, $n \geq 8$, is the sum of at most 28 base-3 palindromes of length $n - 3$.

10 Things we don't know how to prove using our method

We mention two claims that we do not currently know how to attack using our method.

Conjecture 23. Every natural number, except the 56 integers given below, is the sum of at most four natural numbers whose base-2 expansions are of the form xx for some string x starting with 1:

$$1, 2, 4, 5, 7, 8, 11, 14, 17, 22, 27, 29, 32, 34, 37, 41, 44, 47, 53, 62, 95, 104, 107, 113, 116, 122, 125, \\ 131, 134, 140, 143, 148, 155, 158, 160, 167, 407, 424, 441, 458, 475, 492, 509, 526, 552, 560, \\ 569, 587, 599, 608, 613, 620, 638, 653, 671, 686.$$

Remark 24. We verified this up to 2^{28}. For two reasons this problem does not seem to be amenable to our approach. For one thing, the set of strings that are squares, $\{xx \ : \ x \in \Sigma^*\}$, is not a CFL. For another, it does not appear to be the case that all or even most numbers are the sum of a constant number of numbers with representation xx of the same length.

Conjecture 25. Every natural number > 147615 is the sum of at most nine natural numbers whose base-2 expansions are of the form xxx for some strings x starting with 1. The total number of exceptions is 4921.

Remark 26. We have verified this result up to 2^{27}.

More generally one might conjecture a variant of Waring's theorem: for all $k \geq 2$ there is a constant $C(k)$ such that all sufficiently large integers are the sum of $C(k)$ integers whose base-2 representation is of the form $\overbrace{xx \cdots x}^{k}$.

10.1 Schnirelmann density

Another, more number-theoretic, approach to the problems we discuss in this paper is Schnirelmann density. Given a set $S \subseteq \mathbb{N}$, define $A_S(x) = \sum_{\substack{i \in S \\ 1 \leq i \leq x}} 1$ to be the counting function associated with S. The Schnirelmann density of S is then defined to be

$$\sigma(S) := \inf_{n \geq 1} \frac{A_s(n)}{n}.$$

Classical theorems of additive number theory (e.g., [50, §7.4]) relate the property of being an additive basis to the value of $\sigma(S)$. We pose the following open problems:

Open Problem 27. What is the Schnirelmann density d_k of those numbers expressible as the sum of at most k binary palindromes? By computation we find $d_2 < 0.443503$ and $d_3 < .942523$.

Open Problem 28. What is the smallest k such that the Schnirelmann density d'_k of those numbers expressible as the sum of at most k binary palindromes of the same length, is nonzero? By Theorem 1 we have $k \leq 10$.

It is possible that these questions, particularly the last one, could be answered using a decision procedure. To this end, we pose the following problem:

Open Problem 29. Let A be a k-automatic set of natural numbers, or its analogue using a pushdown automaton or nested-word automaton. Is $\sigma(A)$ computable?

11 Moral of the story

We conclude with the following thesis, expressed as two principles.

1. If an argument is heavily case-based, consider turning the proof into an algorithm.

2. If an argument is heavily case-based, seek a logical system or machine model where the assertions can be expressed, and prove them purely mechanically using a decision procedure.

Can other new results in number theory or combinatorics be proved using our approach? We leave this as another challenge to the reader.

Acknowledgments

We thank Dirk Nowotka and Jean-Paul Allouche for helpful discussions. We thank the creators of the ULTIMATE automaton library for their assistance.

References

[1] J.-P. Allouche, N. Rampersad, and J. Shallit. Periodicity, repetitions, and orbits of an automatic sequence. *Theoret. Comput. Sci.* **410** (2009), 2795–2803.

[2] K. Aloui, Ch. Mauduit, and M. Mkaouar. Somme des chiffres et répartition dans les classes de congruence pour les palindromes ellipséphiques. *Acta Math. Hung.* **151** (2017), 409–455.

[3] R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proc. Thirty-Sixth Ann. ACM Symp. Theor. Comput. (STOC)*, pp. 202–211. ACM, 2004.

[4] R. Alur and P. Madhusudan. Adding nested structure to words. *J. Assoc. Comput. Mach.* **56** (2009), Art. 16.

[5] K. Appel and W. Haken. Every planar map is four colorable. I. Discharging. *Illinois J. Math.* **21** (1977), 429490.

[6] K. Appel, W. Haken, and J. Koch. Every planar map is four colorable. II. Reducibility. *Illinois J. Math.* **21** (1977), 491–567.

[7] C. Ashbacher. More on palindromic squares. *J. Recreational Math.* **22** (1990), 133–135.

[8] W. D. Banks. Every natural number is the sum of forty-nine palindromes. *INTEGERS — Electronic J. Combinat. Number Theory* **16** (2016), #A3 (electronic).

[9] W. D. Banks, D. N. Hart, and M. Sakata. Almost all palindromes are composite. *Math. Res. Lett.* **11** (2004), 853–868.

[10] W. D. Banks and I. E. Shparlinski. Prime divisors of palindromes. *Period. Math. Hung.* **51** (2005), 1–10.

[11] W. D. Banks and I. E. Shparlinski. Average value of the Euler function on binary palindromes. *Bull. Pol. Acad. Sci. Math.* **54** (2006), 95–101.

[12] J. K. R. Barnett. Tables of square palindromes in bases 2 and 10. *J. Recreational Math.* **23**(1) (1991), 13–18.

[13] B. Bašić. On d-digit palindromes in different bases: the number of bases is unbounded. *Internat. J. Number Theory* **8** (2012), 1387–1390.

[14] B. Bašić. On "very palindromic" sequences. *J. Korean Math. Soc.* **52** (2015), 765–780.

[15] A. Bérczes and V. Ziegler. On simultaneous palindromes. *J. Combin. Number Theory* **6** (2014), 37–49.

[16] J. Berstel. *Axel Thue's Papers on Repetitions in Words: a Translation*. Number 20 in Publications du Laboratoire de Combinatoire et d'Informatique Mathématique. Université du Québec à Montréal, February 1995.

[17] B. von Braunmühl and R. Verbeek. Input-driven languages are recognized in $\log n$ space. In M. Karpinski, editor, *Foundations of Computation Theory, FCT 83*, Vol. 158 of *Lecture Notes in Computer Science*, pp. 40–51. Springer-Verlag, 1983.

[18] E. Charlier, N. Rampersad, and J. Shallit. Enumeration and decidable properties of automatic sequences. *Internat. J. Found. Comp. Sci.* **23** (2012), 1035–1066.

[19] J. Cilleruelo and F. Luca. Every positive integer is a sum of three palindromes. Preprint available at `https://arxiv.org/abs/1602.06208v1`, 2016.

[20] J. Cilleruelo, F. Luca, and L. Baxter. Every positive integer is a sum of three palindromes. Preprint available at `https://arxiv.org/abs/1602.06208v2`, 2017.

[21] J. Cilleruelo, F. Luca, and I. E. Shparlinski. Power values of palindromes. *J. Combin. Number Theory* **1** (2009), 101–107.

[22] J. Cilleruelo, R. Tesoro, and F. Luca. Palindromes in linear recurrence sequences. *Monatsh. Math.* **171** (2013), 433–442.

[23] S. Col. Palindromes dans les progressions arithmétiques. *Acta Arith.* **137** (2009), 1–41.

[24] C. F. Du, H. Mousavi, E. Rowland, L. Schaeffer, and J. Shallit. Decision algorithms for Fibonacci-automatic words, II: related sequences and avoidability. *Theoret. Comput. Sci.* **657** (2017), 146–162.

[25] C. F. Du, H. Mousavi, L. Schaeffer, and J. Shallit. Decision algorithms for Fibonacci-automatic words, III: enumeration and abelian properties. *Internat. J. Found. Comp. Sci.* **27** (2016), 943–963.

[26] P. W. Dymond. Input-driven languages are in $\log n$ depth. *Inform. Process. Lett.* **26** (1988), 247–250.

[27] D. Goc, D. Henshall, and J. Shallit. Automatic theorem-proving in combinatorics on words. In N. Moreira and R. Reis, editors, *CIAA 2012*, Vol. 7381 of *Lecture Notes in Computer Science*, pp. 180–191. Springer-Verlag, 2012.

[28] D. Goc, H. Mousavi, and J. Shallit. On the number of unbordered factors. In A.-H. Dediu, C. Martin-Vide, and B. Truthe, editors, *LATA 2013*, Vol. 7810 of *Lecture Notes in Computer Science*, pp. 299–310. Springer-Verlag, 2013.

[29] D. Goc, K. Saari, and J. Shallit. Primitive words and Lyndon words in automatic and linearly recurrent sequences. In A.-H. Dediu, C. Martin-Vide, and B. Truthe, editors, *LATA 2013*, Vol. 7810 of *Lecture Notes in Computer Science*, pp. 311–322. Springer-Verlag, 2013.

[30] D. Goc, L. Schaeffer, and J. Shallit. The subword complexity of k-automatic sequences is k-synchronized. In M.-P. Béal and O. Carton, editors, *DLT 2013*, Vol. 7907 of *Lecture Notes in Computer Science*, pp. 252–263. Springer-Verlag, 2013.

[31] E. H. Goins. Palindromes in different bases: a conjecture of J. Ernest Wilkins. *INTEGERS — Electronic J. Combinat. Number Theory* **9** (2009), Paper #A55, 725–734 (electronic).

[32] Y.-S. Han and K. Salomaa. Nondeterministic state complexity of nested word automata. *Theoret. Comput. Sci.* **410** (2009), 2961–2971.

[33] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, 5th edition, 1985.

[34] M. Harminc and R. Soták. Palindromic numbers in arithmetic progressions. *Fibonacci Quart.* **36** (1998), 259–262.

[35] M. Heizmann, J. Hoenicke, and A. Podelski. Software model checking for people who love automata. In N. Sharygina and H. Veith, editors, *Computer Aided Verification — 25th International Conference, CAV 2013*, Vol. 8044 of *Lecture Notes in Computer Science*, pp. 36–52. Springer-Verlag, 2013.

[36] M. Heizmann, D. Dietsch, M. Greitschus, J. Leike, B. Musa, C. Schätzle, and A. Podelski. Ultimate automizer with two-track proofs. In M. Chechik and J.-F. Raskin, editors, *Tools and Algorithms for the Construction and Analysis of Systems — 22nd International Conference, TACAS 2016*, Vol. 9636 of *Lecture Notes in Computer Science*, pp. 950–953. Springer-Verlag, 2016.

[37] S. Hernández and F. Luca. Palindromic powers. *Rev. Colombiana Mat.* **40** (2006), 81–86.

[38] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[39] M. Keith. Classification and enumeration of palindromic squares. *J. Recreational Math.* **22**(2) (1990), 124–132.

[40] P. Kidder and H. Kwong. Remarks on integer palindromes. *J. Recreational Math.* **36** (2007), 299–306.

[41] I. Korec. Palindromic squares for various number system bases. *Math. Slovaca* **41** (1991), 261–276.

[42] H. Kresová and T. Šalát. On palindromic numbers. *Acta Math. Univ. Comenian.* **42/43** (1984), 293–298.

[43] S. La Torre, M. Napoli, and M. Parente. On the membership problem for visibly pushdown languages. In S. Graf and W. Zhang, editors, *ATVA 2006*, Vol. 4218 of *Lecture Notes in Computer Science*, pp. 96–109. Springer-Verlag, 2006.

[44] F. Luca. Palindromes in Lucas sequences. *Monatsh. Math.* **138** (2003), 209–223.

[45] F. Luca and A. Togbé. On binary palindromes of the form $10^n \pm 1$. *C. R. Acad. Sci. Paris* **346** (2008), 487–489.

[46] F. Luca and P. T. Young. On the binary expansion of the odd Catalan numbers. In *Proc. 14th Int. Conf. on Fibonacci Numbers and their Applications*, pp. 185–190. 2011.

[47] W. McCune. Solution of the Robbins problem. *J. Automated Reasoning* **19**(3) (1997), 263–276.

[48] K. Mehlhorn. Pebbling mountain ranges and its application of DCFL-recognition. In *Proc. 7th Int'l Conf. on Automata, Languages, and Programming (ICALP)*, Vol. 85 of *Lecture Notes in Computer Science*, pp. 422–435. Springer-Verlag, 1980.

[49] H. Mousavi, L. Schaeffer, and J. Shallit. Decision algorithms for Fibonacci-automatic words, I: Basic results. *RAIRO Inform. Théor. App.* **50** (2016), 39–66.

[50] M. B. Nathanson. *Additive Number Theory: The Classical Bases.* Springer-Verlag, 1996.

[51] W. Ogden. A helpful result for proving inherent ambiguity. *Math. Systems Theory* **2** (1968), 191–194.

[52] A. Okhotin and K. Salomaa. State complexity of operations on input-driven pushdown automata. *J. Comput. System Sci.* **86** (2017), 207–228.

[53] X. Piao and K. Salomaa. Operational state complexity of nested word automata. *Theoret. Comput. Sci.* **410** (2009), 3290–3302.

[54] P. Pollack. Palindromic sums of proper divisors. *INTEGERS — Electronic J. Combinat. Number Theory* **15A** (2015), Paper #A13 (electronic).

[55] K. Salomaa. Limitations of lower bound methods for deterministic nested word automata. *Inform. Comput.* **209** (2011), 580–589.

[56] J. Shallit. Decidability and enumeration for automatic sequences: a survey. In A. A. Bulatov and A. M. Shur, editors, *CSR 2013*, Vol. 7913 of *Lecture Notes in Computer Science*, pp. 49–63. Springer-Verlag, 2013.

[57] T. N. Shorey. On the equation $z^q = (x^n - 1)/(x - 1)$. *Indag. Math.* **48** (1986), 345–351.

[58] M. Sigg. On a conjecture of John Hoffman regarding sums of palindromic numbers. Preprint available at `https://arxiv.org/abs/1510.07507`, 2015.

[59] G. J. Simmons. On palindromic squares of non-palindromic numbers. *J. Recreational Math.* **5**(1) (1972), 11–19.

[60] A. Thue. Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. *Norske vid. Selsk. Skr. Mat. Nat. Kl.* **1** (1912), 1–67. Reprinted in *Selected Mathematical Papers of Axel Thue*, T. Nagell, editor, Universitetsforlaget, Oslo, 1977, pp. 413–478.

[61] C. Trigg. Infinite sequences of palindromic triangular numbers. *Fibonacci Quart.* **12** (1974), 209–212.

[62] T. Tymoczko. The four-color problem and its philosophical significance. *J. Philosophy* **76**(2) (1979), 57–83.

[63] R. C. Vaughan and T. Wooley. Waring's problem: a survey. In M. A. Bennett, B. C. Berndt, N. Boston, H. G. Diamond, A. J. Hildebrand, and W. Philipp, editors, *Number Theory for the Millennium. III*, pp. 301–340. A. K. Peters, 2002.

[64] S. Yates. The mystique of repunits. *Math. Mag.* **51** (1978), 22–28.