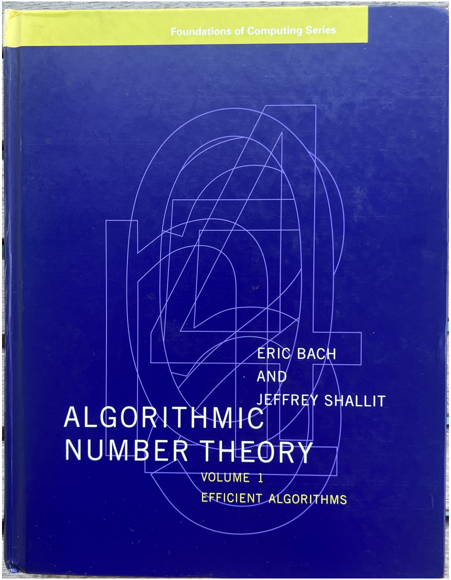
Eric Bach, Berkeley, 1981

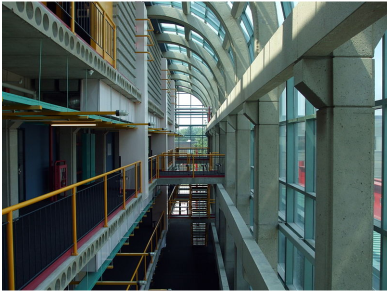


Algorithmic Number Theory, 1996

Using Automata to Prove Theorems in Additive Number Theory

Jeffrey Shallit

School of Computer Science
University of Waterloo
Waterloo, ON N2L 3G1 Canada
shallit@uwaterloo.ca
https://cs.uwaterloo.ca/~shallit/



Additive number theory

Let S be a subset of the natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$.

Additive number theory

Let S be a subset of the natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$.

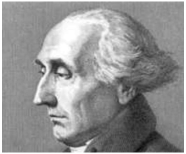
The **principal problem** of additive number theory is to determine whether every natural number (or every sufficiently large natural number) can be written as the sum of some **constant** number of elements of S.

Additive number theory

Let S be a subset of the natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$.

The **principal problem** of additive number theory is to determine whether every natural number (or every sufficiently large natural number) can be written as the sum of some **constant** number of elements of S.

Probably the most famous example is **Lagrange's theorem** (1770):



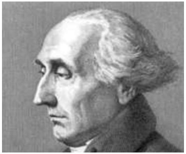
(a) every natural number is the sum of four squares; and

Additive number theory

Let S be a subset of the natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$.

The **principal problem** of additive number theory is to determine whether every natural number (or every sufficiently large natural number) can be written as the sum of some **constant** number of elements of S.

Probably the most famous example is **Lagrange's theorem** (1770):



(a) every natural number is the sum of four squares; and

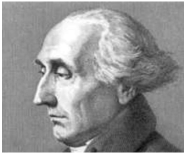
(b) three squares do not suffice for numbers of the form $4^a(8k + 7)$.

Additive number theory

Let S be a subset of the natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$.

The **principal problem** of additive number theory is to determine whether every natural number (or every sufficiently large natural number) can be written as the sum of some **constant** number of elements of S.

Probably the most famous example is **Lagrange's theorem** (1770):



(a) every natural number is the sum of four squares; and

(b) three squares do not suffice for numbers of the form $4^a(8k+7)$.

(Conjectured by Bachet in 1621.)

Additive bases

Let $S \subseteq \mathbb{N}$.

Additive bases

Let $S \subseteq \mathbb{N}$.

We say that a subset S is an **basis of order** h if every natural number can be written as the sum of h elements of S, not necessarily distinct.

Additive bases

Let $S \subseteq \mathbb{N}$.

We say that a subset S is an **basis of order** h if every natural number can be written as the sum of h elements of S, not necessarily distinct.

We say that a subset S is an **asymptotic basis of order** h if every *sufficiently large* natural number can be written as the sum of h elements of S, not necessarily distinct.

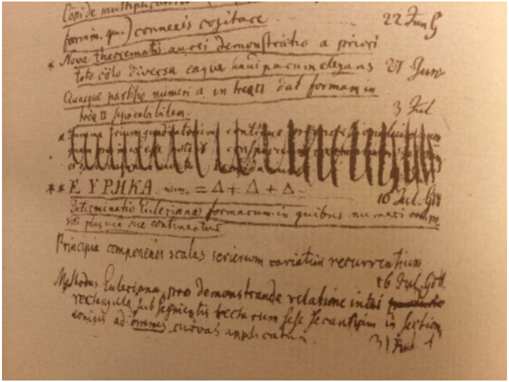
Gauss's theorem for triangular numbers

A *triangular number* is a number of the form $n(n+1)/2$.

Gauss's theorem for triangular numbers

A *triangular number* is a number of the form $n(n+1)/2$.

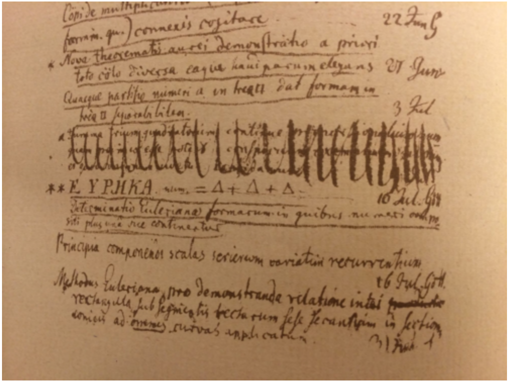
Gauss wrote the following in his diary on July 10 1796:



Gauss's theorem for triangular numbers

A *triangular number* is a number of the form $n(n+1)/2$.

Gauss wrote the following in his diary on July 10 1796:



i.e., **The triangular numbers form an additive basis of order** 3

Waring's problem for powers

Edward Waring (1770) asserted, without proof, that every natural number is
– the sum of 4 squares

Waring's problem for powers

Edward Waring (1770) asserted, without proof, that every natural number is
– the sum of 4 squares
– the sum of 9 cubes

Waring's problem for powers

Edward Waring (1770) asserted, without proof, that every natural number is
– the sum of 4 squares
– the sum of 9 cubes
– the sum of 19 fourth powers

Waring's problem for powers

Edward Waring (1770) asserted, without proof, that every natural number is
– the sum of 4 squares
– the sum of 9 cubes
– the sum of 19 fourth powers
– "and so forth".

9. Omnis integer numerus vel eſt cubus; vel e duobus, tribus, 4, 5, 6, 7, 8, vel novem cubis compoſitus: eſt etiam quadrato-quadratus; vel e duobus, tribus, &c. uſque ad novemdecim compoſitus, & ſic deinceps: conſimilia etiam affirmari poſſunt (exceptis excipiendis) de eodem numero quantitatum earundem dimenſionum.

Waring's problem

Let $g(k)$ be the least natural number m such that every natural number is the sum of m k'th powers.

Waring's problem

Let $g(k)$ be the least natural number m such that every natural number is the sum of m k'th powers.

Let $G(k)$ be the least natural number m such that every sufficiently large natural number is the sum of m k'th powers.

Waring's problem

Let $g(k)$ be the least natural number m such that every natural number is the sum of m k'th powers.

Let $G(k)$ be the least natural number m such that every sufficiently large natural number is the sum of m k'th powers.

Proving that $g(k)$ and $G(k)$ exist, and determining their values, is **Waring's problem**.

Waring's problem

Let $g(k)$ be the least natural number m such that every natural number is the sum of m k'th powers.

Let $G(k)$ be the least natural number m such that every sufficiently large natural number is the sum of m k'th powers.

Proving that $g(k)$ and $G(k)$ exist, and determining their values, is **Waring's problem**.

By Lagrange we know $g(2) = G(2) = 4$.

Waring's problem

Let $g(k)$ be the least natural number m such that every natural number is the sum of m k'th powers.

Let $G(k)$ be the least natural number m such that every sufficiently large natural number is the sum of m k'th powers.

Proving that $g(k)$ and $G(k)$ exist, and determining their values, is **Waring's problem**.

By Lagrange we know $g(2) = G(2) = 4$.

Hilbert proved in 1909 that $g(k)$ and $G(k)$ exist for all k.

Waring's problem

Let $g(k)$ be the least natural number m such that every natural number is the sum of m k'th powers.

Let $G(k)$ be the least natural number m such that every sufficiently large natural number is the sum of m k'th powers.

Proving that $g(k)$ and $G(k)$ exist, and determining their values, is **Waring's problem**.

By Lagrange we know $g(2) = G(2) = 4$.

Hilbert proved in 1909 that $g(k)$ and $G(k)$ exist for all k.

By Wieferich and Kempner we know $g(3) = 9$.

Waring's problem

Let $g(k)$ be the least natural number m such that every natural number is the sum of m k'th powers.

Let $G(k)$ be the least natural number m such that every sufficiently large natural number is the sum of m k'th powers.

Proving that $g(k)$ and $G(k)$ exist, and determining their values, is **Waring's problem**.

By Lagrange we know $g(2) = G(2) = 4$.

Hilbert proved in 1909 that $g(k)$ and $G(k)$ exist for all k.

By Wieferich and Kempner we know $g(3) = 9$.

We know that $4 \leq G(3) \leq 7$, but the true value is still unknown.

Other additive bases?

What other sets can be additive bases?

Other additive bases?

What other sets can be additive bases?

Not the powers of 2 – too sparse.

Other additive bases?

What other sets can be additive bases?

Not the powers of 2 – too sparse.

Need a set whose natural density is at least $N^{1/k}$ for some k.

Other additive bases?

What other sets can be additive bases?

Not the powers of 2 – too sparse.

Need a set whose natural density is at least $N^{1/k}$ for some k.

How about numbers with palindromic base-b expansions?

Palindromes

- A *palindrome* is any string that is equal to its reversal

Palindromes

- A *palindrome* is any string that is equal to its reversal
- Examples are `level`, `deified`, ...

Palindromes

- A *palindrome* is any string that is equal to its reversal
- Examples are `level`, `deified`, ...
- We call a natural number a *base-b palindrome* if its base-b representation (without leading zeroes) is a palindrome

Palindromes

- A *palindrome* is any string that is equal to its reversal
- Examples are `level`, `deified`, ...
- We call a natural number a *base-b palindrome* if its base-b representation (without leading zeroes) is a palindrome
- Examples are $16 = [121]_3$ and $297 = [100101001]_2$.

Palindromes

- A *palindrome* is any string that is equal to its reversal
- Examples are `level`, `deified`, ...
- We call a natural number a *base-b palindrome* if its base-b representation (without leading zeroes) is a palindrome
- Examples are $16 = [121]_3$ and $297 = [100101001]_2$.
- Binary palindromes ($b = 2$) form sequence A006995 in the *On-Line Encyclopedia of Integer Sequences* (OEIS):

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

Palindromes

- A *palindrome* is any string that is equal to its reversal
- Examples are `level`, `deified`, ...
- We call a natural number a *base-b palindrome* if its base-b representation (without leading zeroes) is a palindrome
- Examples are $16 = [121]_3$ and $297 = [100101001]_2$.
- Binary palindromes ($b = 2$) form sequence A006995 in the *On-Line Encyclopedia of Integer Sequences* (OEIS):

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

- They have density $\Theta(N^{1/2})$.

The problem

Do the base-b palindromes form an additive basis, and if so, of what order?

The problem

Do the base-b palindromes form an additive basis, and if so, of what order?

William Banks (2015) showed that every natural number is the sum of at most 49 base-10 palindromes. (*INTEGERS* **16** (2016), #A3)

The problem

Do the base-b palindromes form an additive basis, and if so, of what order?

William Banks (2015) showed that every natural number is the sum of at most 49 base-10 palindromes.
(*INTEGERS* **16** (2016), #A3)

Javier Cilleruelo, Florian Luca, and Lewis Baxter (2018) showed that for all bases $b \geq 5$, every natural number is the sum of three base-b palindromes.
(*Math. Comp.* **87** (2018), 3023–3055.)

What we proved

However, the case of bases $b = 2, 3, 4$ was left unsolved. We proved

What we proved

However, the case of bases $b = 2, 3, 4$ was left unsolved. We proved

Theorem (Rajasekaran, JOS, Smith)

Every natural number N is the sum of 4 binary palindromes. The number 4 is optimal.

What we proved

However, the case of bases $b = 2, 3, 4$ was left unsolved. We proved

Theorem (Rajasekaran, JOS, Smith)

Every natural number N is the sum of 4 binary palindromes. The number 4 is optimal.

For example,

$$\begin{aligned} 10011938 &= 5127737 + 4851753 + 32447 + 1 \\ &= [10011100011111000111001]_2 \\ &+ [10010100000100000101001]_2 \\ &+ [111111010111111]_2 + [1]_2 \end{aligned}$$

What we proved

However, the case of bases $b = 2, 3, 4$ was left unsolved. We proved

Theorem (Rajasekaran, JOS, Smith)

Every natural number N is the sum of 4 binary palindromes. The number 4 is optimal.

For example,

$$\begin{aligned}10011938 &= 5127737 + 4851753 + 32447 + 1 \\ &= [10011100011111000111001]_2 \\ &+ [10010100000100000101001]_2 \\ &+ [111111010111111]_2 + [1]_2\end{aligned}$$

4 is optimal: 10011938 is not the sum of 2 binary palindromes.

Previous proofs were complicated (1)

Excerpt from Banks (2015):

2.4. **Inductive passage from $\mathbb{N}_{\ell,k}(5^+;c_1)$ to $\mathbb{N}_{\ell-1,k+1}(5^+;c_2)$.**

Lemma 2.4. *Let $\ell, k \in \mathbb{N}$, $\ell \geqslant k+6$, and $c_\ell \in \mathcal{D}$ be given. Given $n \in \mathbb{N}_{\ell,k}(5^+;c_1)$, one can find digits $a_1, \ldots, a_{18}, b_1, \ldots, b_{18} \in \mathcal{D}\backslash\{0\}$ and $c_2 \in \mathcal{D}$ such that the number*

$$n - \sum_{j=1}^{18} q_{\ell-1,k}(a_j, b_j)$$

lies in the set $\mathbb{N}_{\ell-1,k+1}(5^+;c_2)$.

Proof. Fix $n \in \mathbb{N}_{\ell,k}(5^+;c_1)$, and let $\{\delta_j\}_{j=0}^{\ell-1}$ be defined as in (1.1) (with $L := \ell$). Let m be the three-digit integer formed by the first three digits of n; that is,

$$m := 100\delta_{\ell-1} + 10\delta_{\ell-2} + \delta_{\ell-3}.$$

Clearly, m is an integer in the range $500 \leqslant m \leqslant 999$, and we have

$$n = \sum_{j=k}^{\ell-1} 10^j \delta_j = 10^{\ell-3} m + \sum_{j=k}^{\ell-4} 10^j \delta_j. \tag{2.4}$$

Let us denote

$$\mathcal{S} := \{19, 29, 39, 49, 59\}.$$

In view of the fact that

$$9\mathcal{S} := \underbrace{\mathcal{S} + \cdots + \mathcal{S}}_{\text{nine copies}} = \{171, 181, 191, \ldots, 531\},$$

it is possible to find an element $h \in 9\mathcal{S}$ for which $m - 80 < 2h \leqslant m - 60$. With h fixed, let $s_1, \ldots, s_9$ be elements of $\mathcal{S}$ such that

$$s_1 + \cdots + s_9 = h.$$

Previous proofs were complicated (2)

Excerpt from Cilleruelo et al. (2018)

II.2 $\boldsymbol{c_m = 0}$. We distinguish the following cases:

II.2.i) $\boldsymbol{y_m \neq 0}$.

δ_m	δ_{m-1}
0	0
$*$	y_m
$*$	$*$

$\longrightarrow$

δ_m	δ_{m-1}
1	1
$*$	$y_m - 1$
$*$	$*$

II.2.ii) $\boldsymbol{y_m = 0}$.

II.2.ii.a) $\boldsymbol{y_{m-1} \neq 0}$.

δ_m	δ_{m-1}	δ_{m-2}
0	0	$*$
y_{m-1}	0	y_{m-1}
$*$	z_{m-1}	z_{m-1}

$\longrightarrow$

δ_m	δ_{m-1}	δ_{m-2}
1	1	$*$
$y_{m-1} - 1$	$g - 2$	$y_{m-1} - 1$
$*$	$z_{m-1} + 1$	$z_{m-1} + 1$

The above step is justified for $z_{m-1} \neq g - 1$. But if $z_{m-1} = g - 1$, then $c_{m-1} \geq (y_{m-1} + z_{m-1})/g \geq 1$, so $c_m = (z_{m-1} + c_{m-1})/g = (g - 1 + 1)/g = 1$, a contradiction.

II.2.ii.b) $\boldsymbol{y_{m-1} = 0,\ z_{m-1} \neq 0}$.

δ_m	δ_{m-1}	δ_{m-2}
0	0	$*$
0	0	0
$*$	z_{m-1}	z_{m-1}

$\longrightarrow$

δ_m	δ_{m-1}	δ_{m-2}
0	0	$*$
1	1	1
$*$	$z_{m-1} - 1$	$z_{m-1} - 1$

II.2.ii.c) $\boldsymbol{y_{m-1} = 0,\ z_{m-1} = 0}$.

If also $c_{m-1} = 0$, then $\delta_{m-1} = 0$, which is not allowed. Thus, $c_{m-1} = 1$.

Previous proofs were complicated (3)

- Proofs of Banks and Cilleruelo et al. were long and case-based

Previous proofs were complicated (3)

- Proofs of Banks and Cilleruelo et al. were long and case-based
- Difficult to establish

Previous proofs were complicated (3)

- Proofs of Banks and Cilleruelo et al. were long and case-based
- Difficult to establish
- Difficult to understand

Previous proofs were complicated (3)

- Proofs of Banks and Cilleruelo et al. were long and case-based
- Difficult to establish
- Difficult to understand
- Difficult to check, too: the original Cilleruelo et al. proof had some minor flaws that were only noticed when the proof was implemented as a `Python` program

Previous proofs were complicated (3)

- Proofs of Banks and Cilleruelo et al. were long and case-based
- Difficult to establish
- Difficult to understand
- Difficult to check, too: the original Cilleruelo et al. proof had some minor flaws that were only noticed when the proof was implemented as a `Python` program
- Idea: could we automate such proofs?

The main idea of our proof

- Construct a finite-state machine (automaton) that takes natural numbers as input, expressed in the desired base

The main idea of our proof

- Construct a finite-state machine (automaton) that takes natural numbers as input, expressed in the desired base
- Allow the automaton to nondeterministically "guess" a representation of the input as a sum of palindromes

The main idea of our proof

- Construct a finite-state machine (automaton) that takes natural numbers as input, expressed in the desired base
- Allow the automaton to nondeterministically "guess" a representation of the input as a sum of palindromes
- The machine accepts an input if it "verifies" its guess

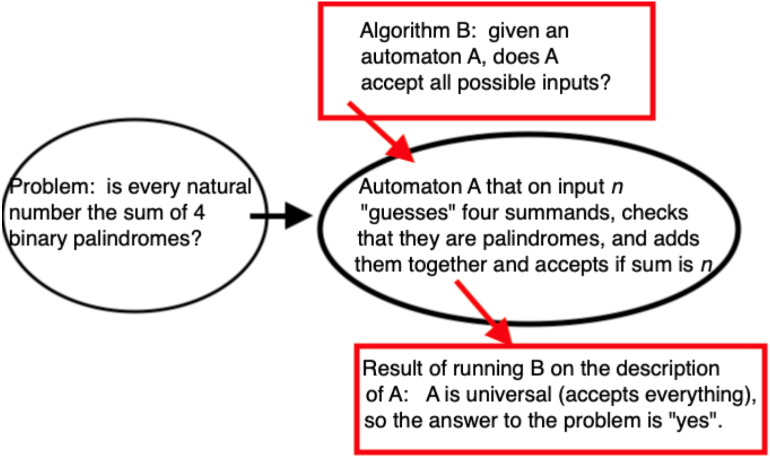
The main idea of our proof

- Construct a finite-state machine (automaton) that takes natural numbers as input, expressed in the desired base
- Allow the automaton to nondeterministically "guess" a representation of the input as a sum of palindromes
- The machine accepts an input if it "verifies" its guess
- Then use a decision procedure to establish properties about the language of representations accepted by this machine (e.g., universality – does it accept every possible input?)

The main idea of our proof

- Construct a finite-state machine (automaton) that takes natural numbers as input, expressed in the desired base
- Allow the automaton to nondeterministically "guess" a representation of the input as a sum of palindromes
- The machine accepts an input if it "verifies" its guess
- Then use a decision procedure to establish properties about the language of representations accepted by this machine (e.g., universality – does it accept every possible input?)
- We build the machine, but never run it! What we run is an algorithm that decides a property of the machine.

Our proof strategy



Basics of automata

- An automaton is a mathematical model of a very simple computer

Basics of automata

- An automaton is a mathematical model of a very simple computer
- It takes as input a finite list of symbols $x = a_1 a_2 \cdots a_n$, called a "string" or "word")

Basics of automata

- An automaton is a mathematical model of a very simple computer
- It takes as input a finite list of symbols $x = a_1 a_2 \cdots a_n$, called a "string" or "word")
- The automaton does some computation and then either "accepts" or "rejects" its input

Basics of automata

- An automaton is a mathematical model of a very simple computer
- It takes as input a finite list of symbols $x = a_1 a_2 \cdots a_n$, called a "string" or "word")
- The automaton does some computation and then either "accepts" or "rejects" its input
- The set of all accepted strings is called the *language recognized* by the automaton

Parts of an automaton

- The finite set of *states*: each state corresponds to some knowledge that has been gained about the input

Parts of an automaton

- The finite set of *states*: each state corresponds to some knowledge that has been gained about the input
- The start state

Parts of an automaton

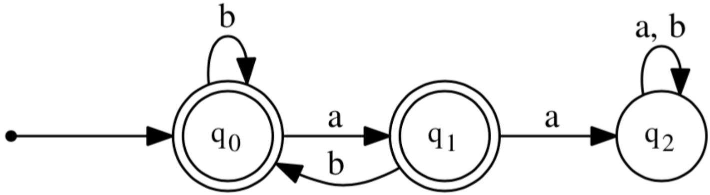
- The finite set of *states*: each state corresponds to some knowledge that has been gained about the input
- The start state
- The set of accepting states

Parts of an automaton

- The finite set of *states*: each state corresponds to some knowledge that has been gained about the input
- The start state
- The set of accepting states
- The transition function that specifies, for each state and each input symbol, which state to enter

Example of an automaton

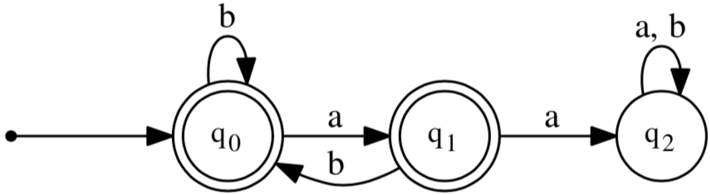
A double circle represents an accepting state.



What is the language accepted by this automaton?

Example of an automaton

A double circle represents an accepting state.



What is the language accepted by this automaton?

It is the set of all strings having no two consecutive a's.

Decision algorithms for automata

- Given an automaton, we can decide various things about the language it recognizes.

Decision algorithms for automata

- Given an automaton, we can decide various things about the language it recognizes.
- For example, is the language empty? Or infinite?

Decision algorithms for automata

- Given an automaton, we can decide various things about the language it recognizes.
- For example, is the language empty? Or infinite?
- Here "decide" means there is an algorithm that, given the automaton as input, halts and says (for example) either "language is empty" or "language is not empty".

Decision algorithms for automata

- Given an automaton, we can decide various things about the language it recognizes.
- For example, is the language empty? Or infinite?
- Here "decide" means there is an algorithm that, given the automaton as input, halts and says (for example) either "language is empty" or "language is not empty".
- In some cases, we can also decide universality: the property of accepting all strings.

Picking an automaton for palindromes

What kind of automaton should we choose?

Picking an automaton for palindromes

What kind of automaton should we choose?

- it should be possible to check if the guessed summands are palindromes

Picking an automaton for palindromes

What kind of automaton should we choose?

- it should be possible to check if the guessed summands are palindromes
  - can be done with a pushdown automaton (PDA)

Picking an automaton for palindromes

What kind of automaton should we choose?

- it should be possible to check if the guessed summands are palindromes
  - can be done with a pushdown automaton (PDA)
- it should be possible to add the summands and compare to the input

Picking an automaton for palindromes

What kind of automaton should we choose?

- it should be possible to check if the guessed summands are palindromes
  - can be done with a pushdown automaton (PDA)
- it should be possible to add the summands and compare to the input
  - can be done with a finite automaton (DFA or NFA)

Picking an automaton for palindromes

What kind of automaton should we choose?

- it should be possible to check if the guessed summands are palindromes
  - can be done with a pushdown automaton (PDA)
- it should be possible to add the summands and compare to the input
  - can be done with a finite automaton (DFA or NFA)

However

Picking an automaton for palindromes

What kind of automaton should we choose?

- it should be possible to check if the guessed summands are palindromes
  - can be done with a pushdown automaton (PDA)
- it should be possible to add the summands and compare to the input
  - can be done with a finite automaton (DFA or NFA)

However

- Can't add summands with these machine models unless they are guessed in parallel

Picking an automaton for palindromes

What kind of automaton should we choose?

- it should be possible to check if the guessed summands are palindromes
  - can be done with a pushdown automaton (PDA)
- it should be possible to add the summands and compare to the input
  - can be done with a finite automaton (DFA or NFA)

However

- Can't add summands with these machine models unless they are guessed in parallel
- Can't check if summands are palindromes if they are wildly different in length & presented in parallel

Picking an automaton for palindromes

What kind of automaton should we choose?

- it should be possible to check if the guessed summands are palindromes
  - can be done with a pushdown automaton (PDA)
- it should be possible to add the summands and compare to the input
  - can be done with a finite automaton (DFA or NFA)

However

- Can't add summands with these machine models unless they are guessed in parallel
- Can't check if summands are palindromes if they are wildly different in length & presented in parallel
- Universality is not decidable for PDA's

Picking an automaton for palindromes

What kind of automaton should we choose?

- it should be possible to check if the guessed summands are palindromes
  - can be done with a pushdown automaton (PDA)
- it should be possible to add the summands and compare to the input
  - can be done with a finite automaton (DFA or NFA)

However

- Can't add summands with these machine models unless they are guessed in parallel
- Can't check if summands are palindromes if they are wildly different in length & presented in parallel
- Universality is not decidable for PDA's

What to do?

A trick

There is a trick that allows us to use finite automata.

The basic idea is the following: there is an 8-state finite automaton that takes four inputs in parallel—numbers represented by strings x, y, z, w—and accepts if and only if $x = ta$ and $y = ub$ and $[tat^R]_2 + [ubu^r] = [zw^R]_2$. Here x and y must start with 1 and so must z. What we get is a description of those n bit numbers that are the sum of two $(n-1)$-bit palindromes.

So we can implicitly represent palindromes by xx^R and xax^R and add them implicitly.

Idea of the proof

- To prove our result, we built two automata:

Idea of the proof

- To prove our result, we built two automata:
    - A accepts all n-bit odd integers, $n \geq 8$, that are the sum of three binary palindromes of length either

Idea of the proof

- To prove our result, we built two automata:
  - A accepts all n-bit odd integers, $n \geq 8$, that are the sum of three binary palindromes of length either
    - n, $n-2$, $n-3$, or

Idea of the proof

- To prove our result, we built two automata:
  - A accepts all n-bit odd integers, $n \geq 8$, that are the sum of three binary palindromes of length either
    - n, $n-2$, $n-3$, or
    - $n-1$, $n-2$, $n-3$.
  - B accepts all valid representations of odd integers of length $n \geq 8$

Idea of the proof

- To prove our result, we built two automata:
  - A accepts all n-bit odd integers, $n \geq 8$, that are the sum of three binary palindromes of length either
    - n, $n-2$, $n-3$, or
    - $n-1$, $n-2$, $n-3$.
  - B accepts all valid representations of odd integers of length $n \geq 8$
- We then prove that all inputs accepted by B are accepted by A

Idea of the proof

- To prove our result, we built two automata:
  - A accepts all n-bit odd integers, $n \geq 8$, that are the sum of three binary palindromes of length either
    - n, $n-2$, $n-3$, or
    - $n-1$, $n-2$, $n-3$.
  - B accepts all valid representations of odd integers of length $n \geq 8$
- We then prove that all inputs accepted by B are accepted by A
- Thus every odd integer ≥ 256 is the sum of three binary palindromes.

Idea of the proof

- To prove our result, we built two automata:
  - A accepts all n-bit odd integers, $n \geq 8$, that are the sum of three binary palindromes of length either
    - n, $n-2$, $n-3$, or
    - $n-1$, $n-2$, $n-3$.
  - B accepts all valid representations of odd integers of length $n \geq 8$
- We then prove that all inputs accepted by B are accepted by A
- Thus every odd integer ≥ 256 is the sum of three binary palindromes.
- For even integers, we just include 1 as one of the summands.

Idea of the proof

- To prove our result, we built two automata:
  - A accepts all n-bit odd integers, $n \geq 8$, that are the sum of three binary palindromes of length either
    - n, $n-2$, $n-3$, or
    - $n-1$, $n-2$, $n-3$.
  - B accepts all valid representations of odd integers of length $n \geq 8$
- We then prove that all inputs accepted by B are accepted by A
- Thus every odd integer ≥ 256 is the sum of three binary palindromes.
- For even integers, we just include 1 as one of the summands.
- The numbers < 256 are easily checked by brute force.

Idea of the proof

- To prove our result, we built two automata:
  - A accepts all n-bit odd integers, $n \geq 8$, that are the sum of three binary palindromes of length either
    - n, $n-2$, $n-3$, or
    - $n-1$, $n-2$, $n-3$.
  - B accepts all valid representations of odd integers of length $n \geq 8$
- We then prove that all inputs accepted by B are accepted by A
- Thus every odd integer ≥ 256 is the sum of three binary palindromes.
- For even integers, we just include 1 as one of the summands.
- The numbers < 256 are easily checked by brute force.
- And so we've proved: every natural number is the sum of four binary palindromes.

Other results

Theorem

Every natural number $N > 256$ *is the sum of at most three base-3 palindromes.*

Other results

Theorem

Every natural number $N > 256$ is the sum of at most three base-3 palindromes.

Theorem

Every natural number $N > 64$ is the sum of at most three base-4 palindromes.

Other results

Theorem

Every natural number $N > 256$ is the sum of at most three base-3 palindromes.

Theorem

Every natural number $N > 64$ is the sum of at most three base-4 palindromes.

This completes the classification for base-b palindromes for all $b \geq 2$.

More results

Using NFA's we can establish an analogue of Lagrange's four-square theorem.

More results

Using NFA's we can establish an analogue of Lagrange's four-square theorem.

- A *square* is any string that is some shorter string repeated twice

More results

Using NFA's we can establish an analogue of Lagrange's four-square theorem.

- A *square* is any string that is some shorter string repeated twice
- Examples are `hotshots` and `murmur`

More results

Using NFA's we can establish an analogue of Lagrange's four-square theorem.

- A *square* is any string that is some shorter string repeated twice
- Examples are `hotshots` and `murmur`
- We call an integer a *base-b square* if its base-b representation is a square

More results

Using NFA's we can establish an analogue of Lagrange's four-square theorem.

- A *square* is any string that is some shorter string repeated twice
- Examples are `hotshots` and `murmur`
- We call an integer a *base-b square* if its base-b representation is a square
- Examples are $36 = [100100]_2$ and $3 = [11]_2$.

More results

Using NFA's we can establish an analogue of Lagrange's four-square theorem.

- A *square* is any string that is some shorter string repeated twice
- Examples are `hotshots` and `murmur`
- We call an integer a *base-b square* if its base-b representation is a square
- Examples are $36 = [100100]_2$ and $3 = [11]_2$.
- The binary squares form sequence [A020330](https://oeis.org/A020330) in the OEIS

$$3, 10, 15, 36, 45, 54, 63, 136, 153, 170, 187, 204, 221, \ldots$$

Results

Theorem

Every natural number $N > 686$ is the sum of at most 4 binary squares.

Results

Theorem

Every natural number $N > 686$ is the sum of at most 4 binary squares.

For example:

$$\begin{aligned}10011938 &= 9291996 + 673425 + 46517\\ &= [100011011100100011011100]_2 + [10100100011010010001]_2\\ &+ [1011010110110101]_2\end{aligned}$$

Results

Theorem

Every natural number $N > 686$ is the sum of at most 4 binary squares.

For example:

$$\begin{aligned} 10011938 &= 9291996 + 673425 + 46517 \\ &= [100011011100100011011100]_2 + [10100100011010010001]_2 \\ &+ [1011010110110101]_2 \end{aligned}$$

We also have the following result

Theorem

Every natural number is the sum of at most two binary squares and at most two powers of 2.

Generalizing: Waring's theorem for binary k'th powers

Recall Waring's theorem: *for every $k \geq 1$ there exists a constant $g(k)$ such that every natural number is the sum of $g(k)$ k'th powers of natural numbers.*

Generalizing: Waring's theorem for binary k'th powers

Recall Waring's theorem: *for every $k \geq 1$ there exists a constant $g(k)$ such that every natural number is the sum of $g(k)$ k'th powers of natural numbers.*

Could the same theorem hold for binary k'th powers?

Generalizing: Waring's theorem for binary k'th powers

Recall Waring's theorem: *for every $k \geq 1$ there exists a constant $g(k)$ such that every natural number is the sum of $g(k)$ k'th powers of natural numbers.*

Could the same theorem hold for binary k'th powers?

Two issues:

Generalizing: Waring's theorem for binary k'th powers

Recall Waring's theorem: *for every* $k \geq 1$ *there exists a constant* $g(k)$ *such that every natural number is the sum of* $g(k)$ k*'th powers of natural numbers.*

Could the same theorem hold for binary k'th powers?

Two issues:

- 1 is not a binary k'th power, so it has to be "every sufficiently large natural number" and not "every natural number".

Generalizing: Waring's theorem for binary k'th powers

Recall Waring's theorem: *for every $k \geq 1$ there exists a constant $g(k)$ such that every natural number is the sum of $g(k)$ k'th powers of natural numbers.*

Could the same theorem hold for binary k'th powers?

Two issues:

- 1 is not a binary k'th power, so it has to be "every sufficiently large natural number" and not "every natural number".
- The gcd g of the binary k'th powers need not be 1, so it actually has to be "every sufficiently large multiple of g".

gcd of the binary k'th powers

Theorem

The gcd *of the binary* k*'th powers is* $\gcd(k, 2^k - 1)$.

gcd of the binary k'th powers

Theorem

The gcd *of the binary k'th powers is* $\gcd(k, 2^k - 1)$.

Example:
The binary 6'th powers are

$$63, 2730, 4095, 149796, 187245, 224694, 262143, 8947848, 10066329, \ldots$$

with gcd equal to $\gcd(6, 63) = 3$.

Very recent results

Theorem

Every sufficiently large multiple of $\gcd(k, 2^k - 1)$ *is the sum of a constant number (depending on* k*) of binary* k*'th powers.*

Very recent results

Theorem

Every sufficiently large multiple of $\gcd(k, 2^k - 1)$ *is the sum of a constant number (depending on* k*) of binary* k*'th powers.*

Obtained with Daniel Kane and Carlo Sanna.

Outline of the proof

Given a number N we wish to represent as a sum of binary k'th powers:

Outline of the proof

Given a number N we wish to represent as a sum of binary k'th powers:

- choose a suitable power of 2, say 2^n, and express N in base 2^n.

Outline of the proof

Given a number N we wish to represent as a sum of binary k'th powers:

- choose a suitable power of 2, say 2^n, and express N in base 2^n.
- use linear algebra to change the basis and instead express x as a linear combination of $c_k(n), c_k(n+1), \ldots, c_k(n+k-1)$ where

$$c_k(n) = \frac{2^{kn}-1}{2^n-1}.$$

Outline of the proof

Given a number N we wish to represent as a sum of binary k'th powers:

- choose a suitable power of 2, say 2^n, and express N in base 2^n.
- use linear algebra to change the basis and instead express x as a linear combination of $c_k(n), c_k(n+1), \ldots, c_k(n+k-1)$ where

$$c_k(n) = \frac{2^{kn} - 1}{2^n - 1}.$$

- Such a linear combination would seem to provide an expression for x in terms of binary k'th powers, but there are three problems to overcome:

Outline of the proof

Given a number N we wish to represent as a sum of binary k'th powers:

- choose a suitable power of 2, say 2^n, and express N in base 2^n.
- use linear algebra to change the basis and instead express x as a linear combination of $c_k(n), c_k(n+1), \ldots, c_k(n+k-1)$ where

$$c_k(n) = \frac{2^{kn}-1}{2^n-1}.$$

- Such a linear combination would seem to provide an expression for x in terms of binary k'th powers, but there are three problems to overcome:
  (a) the coefficients of $c_k(i)$, $n \leq i < n+k$, could be much too large;

Outline of the proof

Given a number N we wish to represent as a sum of binary k'th powers:

- choose a suitable power of 2, say 2^n, and express N in base 2^n.
- use linear algebra to change the basis and instead express x as a linear combination of $c_k(n), c_k(n+1), \ldots, c_k(n+k-1)$ where

$$c_k(n) = \frac{2^{kn} - 1}{2^n - 1}.$$

- Such a linear combination would seem to provide an expression for x in terms of binary k'th powers, but there are three problems to overcome:
  (a) the coefficients of $c_k(i)$, $n \le i < n + k$, could be much too large;
  (b) the coefficients could be too small or negative;

Outline of the proof

Given a number N we wish to represent as a sum of binary k'th powers:

- choose a suitable power of 2, say 2^n, and express N in base 2^n.
- use linear algebra to change the basis and instead express x as a linear combination of $c_k(n), c_k(n+1), \ldots, c_k(n+k-1)$ where

$$c_k(n) = \frac{2^{kn}-1}{2^n-1}.$$

- Such a linear combination would seem to provide an expression for x in terms of binary k'th powers, but there are three problems to overcome:
  (a) the coefficients of $c_k(i)$, $n \le i < n+k$, could be much too large;
  (b) the coefficients could be too small or negative;
  (c) the coefficients might not be integers.

All of these problems can be handled with some work.

Other results

Call a set S of natural numbers *b-automatic* if the language of the base-b expansions of its members is regular (accepted by a finite automaton).

Other results

Call a set S of natural numbers *b-automatic* if the language of the base-b expansions of its members is regular (accepted by a finite automaton).

Theorem (Bell, Hare, JOS)

It is decidable, given a b-automatic set S, whether it forms an additive basis (resp., asymptotic additive basis) of finite order.

Other results

Call a set S of natural numbers *b-automatic* if the language of the base-b expansions of its members is regular (accepted by a finite automaton).

Theorem (Bell, Hare, JOS)

It is decidable, given a b-automatic set S, whether it forms an additive basis (resp., asymptotic additive basis) of finite order.
If it does, the minimum order is also computable.

Other results

Call a set S of natural numbers *b-automatic* if the language of the base-b expansions of its members is regular (accepted by a finite automaton).

Theorem (Bell, Hare, JOS)

It is decidable, given a b-automatic set S, whether it forms an additive basis (resp., asymptotic additive basis) of finite order.
If it does, the minimum order is also computable.

The proof uses, in part, a decidable extension of Presburger arithmetic.

Congratulations, Eric!

Eric Bach, Gunsight Pass Trail, Glacier National Park, 1987