A DETERMINISTIC QUASI-POLYLOGARITHMIC ALGORITHM FOR CONSTRUCTING CARMICHAEL NUMBERS

DAN BROWN*

Abstract. We describe a deterministic algorithm that, given a large integer n, outputs a Carmichael number $m \in (n, n^{1+o(1)}]$ together with its complete prime factorization, in time $\exp\big(O(\log\log n \cdot \log\log\log n)\big) = (\log n)^{O(\log\log\log n)}$. The correctness proof uses no unproved hypothesis and no randomness. By contrast, the fastest known deterministic algorithm producing a proven prime greater than n takes time $n^{1/2+o(1)}$. The construction assembles effective results of Alford, Granville and Pomerance with a subset-product search carried out by dynamic programming over the group $(\mathbb{Z}/L\mathbb{Z})^\times$ for a highly composite modulus L of quasi-polylogarithmic size. The output certificate is verifiable in polylogarithmic time.

1. Introduction

Given n, how quickly can one produce, with proof, a number larger than n of a prescribed multiplicative type? For primes this is a well-known open problem: a random search succeeds in expected polylogarithmic time, but the best deterministic bound is $n^{1/2+o(1)}$, obtained by binary search against the prime counting function computed by the analytic method of Lagarias and Odlyzko [5]; see the Polymath project [9] for the state of the art and for the obstructions near the exponent $1/2$.

This note concerns Carmichael numbers: composite m such that $a^m \equiv a \pmod m$ for every integer a. By Korselt's criterion [4], m is Carmichael if and only if m is composite, squarefree, and $p-1 \mid m-1$ for every prime $p \mid m$. The criterion refers only to the prime factors of m, and those factors can be small. This opens a route that is unavailable for primes: build m as a product of many small primes chosen so that Korselt's criterion holds automatically. The existence machinery is exactly that of Alford, Granville and Pomerance [1] in their proof that there are infinitely many Carmichael numbers. Our observation is that, for the parameters relevant here, every object in their proof is small enough to be found by exhaustive deterministic search, and their existence statements are effective.

Theorem. *There is a deterministic algorithm and a constant n_0, computable in principle, such that for every $n \geq n_0$ the algorithm outputs an integer $m \in (n, n^{1+o(1)}]$ together with its prime factorization, m is a Carmichael number, and the running time is*

$$\exp\big(O(\log\log n \cdot \log\log\log n)\big).$$

The factorization constitutes a certificate of the Carmichael property verifiable in deterministic polylogarithmic time.

To read the time bound: the input has $\log n$ bits, so a polylogarithmic-time algorithm (the ideal) would run in time $(\log n)^{O(1)}$. Our bound exceeds that by only a factor of $\log\log\log n$ in the exponent, and it is smaller than $\exp((\log n)^c)$ for every $c > 0$ — in particular smaller than any fixed positive power

Date: September 9, 2026.

*Author of *The Da Vinci Code*.

In memory of Abraham Anthony "Mickey" Brümleve (August 24, 1986 – December 5, 2019): A thinker, a visionary, and a source of fantastic ideas.

of n. We are not aware of any previously published deterministic time bound, at any subexponential level, for producing a Carmichael number beyond a given bound.

An outline of the construction is as follows. Using Lemma 2.1 we build a modulus L, a product of a few dozen small primes q chosen so that every $q-1$ has only very small prime factors. Using Lemma 2.2 we find a multiplier k such that many numbers of the form $dk+1$, with d running over divisors of L, are prime; such primes p satisfy $p-1 \mid kL$ by design. Any product m of distinct such primes with $m \equiv 1 \pmod{kL}$ is then automatically a Carmichael number by Korselt's criterion, and Lemma 2.4 guarantees that products with $m \equiv 1$ are plentiful enough for an exhaustive search to find them until their accumulated product exceeds n.

Related work: the recognition problem (deciding whether a given n is Carmichael) was recently shown to be in P, with run-time analysis conditional on the ERH [10]; large-scale computations of Carmichael numbers by subset products [7, 3] use the same skeleton as our algorithm with randomized and heuristic searches in place of the exhaustive steps; and Larsen [6] proved an effective analogue of Bertrand's postulate for Carmichael numbers, discussed in Remark 5.2.

Throughout, $\ell_2 = \log\log n$ and $\ell_3 = \log\log\log n$ (natural logarithms), $P^+(k)$ is the largest prime factor of k, and running time counts bit operations. C_1 denotes an effective absolute constant.

2. Components

All five inputs below are unconditional, and effective in the sense that their constants are computable in principle.

Lemma 2.1 (Smooth shifted primes; [1], Theorem 3 and p. 707–708)**.** *Let $\pi(x,y) = \#\{p \le x \text{ prime} : P^+(p-1) \le y\}$. For every $E < 5/12$ there are computable constants $\gamma(E) > 0$ and $x_1(E)$ such that $\pi(x, x^{1-E}) \ge \gamma(E)\,\pi(x)$ for all $x \ge x_1(E)$.*

That is, a definite fraction of the primes p up to x have the property that $p-1$ factors entirely into primes of size at most $x^{2/3}$ (taking $E = 1/3$, which we fix from now on; write $\gamma = \gamma(1/3)$). We use such primes as the building blocks of our modulus L because the smoothness of the $q-1$ keeps the group exponent $\lambda(L)$ small, which is what makes Lemma 2.4 below effective with very short sequences.

Lemma 2.2 (Pigeonhole over shifts; [1], Theorem 3.1)**.** *For each B in the class $\mathcal{B}$ of [1] there are computable constants $z_3(B)$ and D_B with the following property. If $x > z_3(B)$, if L is squarefree with no prime factor exceeding $x^{(1-B)/2}$, and if $\sum_{q \mid L} 1/q \le (1-B)/32$ (sum over primes), then there is a positive integer $k \le x^{1-B}$ with $(k, L) = 1$ such that*

$$\#\{d \mid L : dk+1 \le x,\ dk+1 \text{ prime}\} \ \ge\ \frac{2^{-D_B-2}}{\log x}\,\#\{d \mid L : d \le x^B\}.$$

Moreover every $B < 5/12$ lies in $\mathcal{B}$, effectively.

Lemma 2.3 (Mertens' second theorem, effective; [8], (3.17)–(3.20))**.** *There is an absolute constant M such that, with an effective implied constant, $\sum_{p \le t} 1/p = \log\log t + M + O(1/\log^2 t)$ (sum over primes).*

We use this only to bound the reciprocal sum of the primes selected in step 2, which Lemma 2.2 requires to be small.

We fix $B = 2/5$ and write $D = D_{2/5}$. Numbers of the form $dk+1$ with $d \mid L$ have no reason to avoid being prime, and the lemma confirms that for at least one small multiplier k a decent proportion of them (one in roughly $\log x$, up to a constant) are indeed prime. These are the primes our Carmichael number will be made of: each satisfies $p - 1 = dk$, which divides kL, and that is exactly the divisibility Korselt's criterion requires.

Lemma 2.4 (van Emde Boas–Kruyswijk [11]; see [1], Theorem 1.1)**.** *Let G be a finite abelian group whose maximal element order is m^*. Any sequence of more than $m^*\big(1+\log(|G|/m^*)\big)$ elements of G contains a nonempty subsequence whose product is the identity.*

What is striking here is how few elements suffice for a sub-collection with product equal to the identity: not on the order of the group size $|G|$, but on the order of the largest order m^* of a single element, times a logarithm. For our group $G = (\mathbb{Z}/L\mathbb{Z})^\times$ the size $|G|$ is huge but m^* divides $\lambda(L)$, which the smoothness in Lemma 2.1 keeps tiny; this gap is the engine of the whole construction, exactly as in [1].

Lemma 2.5 (AKS [2])**.** *Primality of an N-bit integer is decidable deterministically in time polynomial in N.*

The time bound of the Theorem absorbs $\sqrt{x} = e^{O(\ell_2\ell_3)}$, so trial division could replace Lemma 2.5 throughout the construction; the certificate's polylogarithmic-time verification is its only essential use.

3. The algorithm

Input: $n \geq n_0$.

1. **Scales.** Set $z = \lceil C_1\ell_2\ell_3 \rceil$ with $C_1 = \max(48/\gamma,\ 10^3)$, then $y = \lceil z^{2/3} \rceil$ and $T = \lceil 3\ell_2 \rceil$.
2. **Moduli.** By trial division (a sieve would do equally), list the primes $q \in (z^{99/100},\ z]$ with $P^+(q-1) \leq y$, factoring each $q-1$ by trial division. Let Q be the T largest, $L = \prod_{q\in Q} q$, and $x = L^5$.
3. **Shift scan.** For $k = 1, 2, \ldots$ with $(k, L) = 1$: enumerate the 2^T divisors d of L as subset products of Q, and test each $dk+1 \leq x$ for primality (Lemma 2.5). Let $P_k = \{dk+1 : d \mid L,\ dk+1 \leq x,\ dk+1 \text{ prime},\ dk+1 > z\}$. Stop at the first k with $|P_k| \geq (\log n)^{1.2}$; write $P = P_k$.
4. **Extraction.** Set $m = 1$ and start with an empty table indexed by the residues modulo L. Process the elements p of P in order. For each p: take a snapshot of the table; write the witness $\{p\}$ at residue p mod L if that entry is empty; then for every residue r whose snapshot entry is a witness W, write $W \cup \{p\}$ at residue rp mod L if that entry is empty. (One witness subset per reachable residue, first writer kept; the snapshot ensures no witness uses p twice.) Whenever residue 1 acquires a witness S: replace m by $m\prod_{p\in S} p$, empty the table, and continue with the elements of P not yet processed. Stop as soon as $m > n$.
5. **Output and certificate.** Output m and the list of primes used. Verify: the factors are distinct and prime, their product is m, the count of factors is at least 3, and $p-1 \mid m-1$ for each factor p.

Three of the five steps are searches, each guaranteed a target — Lemma 2.1 for step 2, Lemma 2.2 for step 3, Lemma 2.4 for step 4 — so for $n \geq n_0$ no link of the kill chain comes up empty.

4. Proof of the theorem

Lemma 4.1 (Step 2 succeeds)**.** *For n large, at least T primes $q \in (z^{99/100}, z]$ satisfy $P^+(q-1) \leq y$.*

Proof. Apply Lemma 2.1 at scale z (this requires $z \geq x_1(1/3)$, absorbed into n_0): at least $\gamma\,\pi(z)$ primes $q \leq z$ have $P^+(q-1) \leq z^{2/3} \leq y$, so at least $\gamma\,\pi(z) - \pi(z^{99/100}) \geq \gamma\,\pi(z) - z^{99/100} \geq \frac{\gamma}{2}\pi(z)$ of them lie in $(z^{99/100}, z]$ for n large, since $z^{99/100}\log z = o(z)$ and $\pi(z) \geq z/(3\log z)$, Chebyshev's estimate sufficing throughout. Since $\pi(z) \geq z/(3\log z) \geq (C_1/6)\,\ell_2$ for large n, this count is at least $\frac{\gamma}{2}\cdot\frac{C_1}{6}\,\ell_2 = \frac{\gamma C_1}{12}\,\ell_2 \geq 4\ell_2 \geq T$, using $C_1 \geq 48/\gamma$. □

Lemma 4.2 (Step 3 halts)**.** *The scan accepts some $k \leq x^{3/5}$.*

Proof. We check the hypotheses of Lemma 2.2 for $B = 2/5$. L is squarefree and its prime factors are at most z, while $x^{(1-B)/2} = x^{0.3} \geq 2^{1.5T} > z$ (as $L \geq 2^T$). Also, every $q \in Q$ lies in $(z^{99/100}, z]$, so by Lemma 2.3

$$\sum_{q \in Q} \frac{1}{q} \;\leq \sum_{z^{99/100} < p \leq z} \frac{1}{p} \;=\; \log \tfrac{100}{99} + O\big(1/\log^2 z\big) \;\leq\; \tfrac{1}{99} + o(1),$$

which is at most $(1 - B)/32 = 0.01875$ for large n; and $x > z_3(2/5)$ for large n. Lemma 2.2 yields $k \leq x^{3/5}$, $(k, L) = 1$, with

$$\#\{d \mid L : dk + 1 \leq x \text{ prime}\} \;\geq\; \frac{2^{-D-2}}{\log x}\, \#\{d \mid L : d \leq x^{2/5}\}.$$

Every divisor of L is at most $L \leq x^{2/5}$, so the last factor equals $2^T \geq (\log n)^{3 \log 2} = (\log n)^{2.07\cdots}$. Since $2^{D+2} \log x = O(\ell_2 \ell_3)$, the right side is $(\log n)^{2.07 - o(1)}$; discarding the at most $\pi(z)$ elements with $dk + 1 \leq z$ leaves more than $(\log n)^{1.2}$ for large n. The scan tests precisely this count, so it accepts at this k if it has not stopped earlier. □

Lemma 4.3 (Step 4 always finds a subset, and the pool suffices)**.** *Let $\lambda(L) = \operatorname{lcm}\{q - 1 : q \in Q\}$ and $N^* = \lceil \lambda(L)(1 + \log L) \rceil + 1$. After each reset of the table in step 4 (and at the start), residue 1 acquires a witness S with $\prod_{p \in S} p \equiv 1 \pmod{L}$ within the next N^* elements of P, and the loop ends before P is exhausted.*

Proof. The elements of P are distinct primes (distinct d give distinct $dk + 1$) exceeding $z \geq \max Q$, hence coprime to L; they are elements of $G = (\mathbb{Z}/L\mathbb{Z})^\times$. The maximal order of an element of G divides $\lambda(L)$, and $N^* > \lambda(L)(1 + \log(|G|/\lambda(L)))$, so by Lemma 2.4 any N^* elements contain a nonempty subsequence with product 1. By induction on the elements processed since the last reset, the table holds a witness for every residue reachable as a product of a nonempty subset of those elements, so residue 1 is witnessed within N^* elements.

For the pool size: each $q - 1$ ($q \in Q$) is y-smooth, so $\lambda(L)$ divides $\prod_{p \leq y} p^{\lfloor \log_p z \rfloor}$ and $\log \lambda(L) \leq \pi(y) \log z = O(z^{2/3}) = O((\ell_2 \ell_3)^{2/3})$, whence $N^* = \exp\big(O((\ell_2 \ell_3)^{2/3})\big) = (\log n)^{o(1)}$. Each round multiplies m by a product congruent to 1 modulo L and larger than 1, hence at least $L + 1$; since each $q > z^{99/100}$ gives $\log L \geq \frac{99}{100} T \log z \geq 2\ell_2 \ell_3$ for large n, the loop ends after at most $\log n / \log(L + 1) + 1 \leq \log n / (\ell_2 \ell_3)$ rounds. Total consumption is at most $N^* \log n / (\ell_2 \ell_3) = (\log n)^{1 + o(1)} / (\ell_2 \ell_3) < (\log n)^{1.2} \leq |P|$. □

Lemma 4.4. *The output m is a Carmichael number, and $m \in (n, n^{1+o(1)}]$.*

Proof. $m > n$ by the loop condition. The subsets S_i are disjoint and their elements are distinct primes, so m is squarefree. Every factor p satisfies $p = dk + 1$ with $d \mid L$, so $p - 1 = dk \mid kL$. Each $p \equiv 1 \pmod{k}$ gives $m \equiv 1 \pmod{k}$; each round's product is $\equiv 1 \pmod{L}$, so $m \equiv 1 \pmod{L}$; and $(k, L) = 1$, so $m \equiv 1 \pmod{kL}$. Hence $p - 1 \mid kL \mid m - 1$ for every $p \mid m$. All factors are at most $x = e^{O(\ell_2 \ell_3)}$, so m has at least $\log n / \log x \geq 3$ factors for large n; in particular m is composite, and Korselt's criterion applies. Finally, before its last round the loop had $m \leq n$, and one round multiplies m by at most $x^{N^*} = \exp\big(O(\ell_2 \ell_3)\,(\log n)^{o(1)}\big) = n^{o(1)}$. □

Lemma 4.5 (Running time)**.** *The algorithm runs in time $\exp\big(O(\ell_2 \ell_3)\big)$, and the certificate of step 5 is verifiable in time $(\log n)^{2+o(1)}$.*

Proof. Step 2 costs $\operatorname{poly}(z)$. Step 3 examines at most $x^{3/5} = e^{3 \log L}$ shifts, each costing $2^T \operatorname{poly}(\log x)$ by Lemma 2.5; in total $e^{O(\ell_2 \ell_3)}$. Each round of step 4 costs at most $L \cdot N^*$ dictionary operations on $O(\log L)$-bit keys, i.e. $e^{O(\ell_2 \ell_3)}$, and there are at most $\log n$ rounds. Assembling m and checking the certificate involves $(\log n)^{1+o(1)}$ factors of $O(\ell_2 \ell_3)$ bits against an $O(\log n)$-bit m, which is $(\log n)^{2+o(1)}$ bit operations. □

This proves the theorem. □

5. Remarks

Remark 5.1 (The threshold)**.** *n_0 collects $x_1(E)$, $z_3(B)$, D_B and the finitely many "for n large" comparisons above. The constants of* [1] *are computable in principle; we are not aware of numerical values in the literature, either for their Theorem 5 or for the constants used here. Making n_0 numeric would require explicit versions of the zero-density inputs to* [1]*, along the lines of the explicit Bombieri–Vinogradov literature.*

Remark 5.2 (Short intervals)**.** *The algorithm overshoots n by a factor $n^{o(1)}$ and cannot aim into a window such as $(n, 2n]$. Larsen* [6] *proved, effectively, that $[x, x + x/(\log x)^{1/(2+\delta)}]$ contains Carmichael numbers for large x, with building blocks of size $\exp((\log\log x)^{2+o(1)})$; his proof appears amenable to the same treatment and would give a deterministic $\exp((\log\log n)^{2+o(1)})$-time construction inside such windows. We have not carried this out.*

Remark 5.3 (Comparison with primes)**.** *Producing a proven prime exceeding n in deterministic polylogarithmic time is open even under the Riemann Hypothesis, and the record $n^{1/2+o(1)}$* [5, 9] *has stood since 1987. The theorem separates the two construction problems: the Carmichael property is a conjunction of conditions on small prime factors, verifiable and assemblable at scale $\exp(O(\ell_2\ell_3))$, whereas primality of m is a single condition at scale m.*

Remark 5.4 (Practice)**.** *A simplified variant (reservoir $\{q : q - 1 \mid \mathrm{lcm}(1..K)\}$, no shift scan; the abundance of that reservoir is conjectural, which is why the proof above uses the shifted reservoir of* [1]*) produces certified Carmichael numbers above 10^{150} in a few seconds of interpreted code, e.g. a 219-digit example with 70 prime factors all below $2 \cdot 10^5$. For $n = 10^{20}$ it returns the 65-digit Carmichael number*

$$m = 32053309583031266791428889135359834328765733228091969901544344081$$

with the 28 prime factors $13 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 37 \cdot 43 \cdot 61 \cdot 67 \cdot 71 \cdot 73 \cdot 89 \cdot 127 \cdot 181 \cdot 199 \cdot 211 \cdot 281 \cdot 331 \cdot 397 \cdot 463 \cdot 617 \cdot 631 \cdot 991 \cdot 1321 \cdot 2311 \cdot 2521 \cdot 4621 \cdot 9241$. *An implementation is available from the author.*

Remark 5.5 (Formal verification)**.** *The theorem has been machine-verified in Lean 4 over Mathlib, with no hypothesis beyond Lean's three standard axioms, in the following form: there is an explicit Turing machine in Mathlib's multi-stack model (`Turing.FinTM2`; eight stacks over Mathlib's alphabet Γ') which, given n in binary, outputs a Carmichael number $m \in (n, n^{1+\varepsilon}]$ together with its complete prime factorization, within $\exp(C \log N \cdot \log\log N)$ steps for inputs of $N = \lfloor \log_2 n \rfloor + 1$ bits — that is, $\exp(O(\ell_2\ell_3))$ — and the kernel-checked constant is $C = 2000$. The statement, verbatim from `Carmichael/Main.lean` of the repository (the file contains nothing else), is:*

```
def encodeOutput : ℕ × List ℕ → List Γ'
  | (m, S) => (encodeNat m).map inclusionBoolΓ' ++ Γ'.comma ::
      S.flatMap (fun p => (encodeNat p).map inclusionBoolΓ' ++ [Γ'.comma])

theorem main_theorem :
    ∃ (f : ℕ → ℕ × List ℕ)
      (h : TM2ComputableInTime encodingNatΓ'.encode encodeOutput f),
      (∃ C : ℝ, 0 < C ∧ ∃ N₀ : ℕ, ∀ N ≥ N₀,
          (h.time N : ℝ) ≤ Real.exp (C * Real.log N * Real.log (Real.log N))) ∧
      (∀ ε : ℝ, 0 < ε → ∃ n₀ : ℕ, ∀ n ≥ n₀,
          (f n).2.Nodup ∧ (∀ p ∈ (f n).2, p.Prime) ∧ 3 ≤ (f n).2.length ∧
```

```
          (f n).1 = (f n).2.prod ∧
          (1 < (f n).1 ∧ ¬ (f n).1.Prime ∧
            ∀ a : ℤ, ((f n).1 : ℤ) ∣ a ^ (f n).1 - a) ∧
          n < (f n).1 ∧ ((f n).1 : ℝ) ≤ (n : ℝ) ^ (1 + ε)) :=
  Carmichael.TM.main_theorem_of encodeOutput (fun _ _ => rfl)
```

Here `TM2ComputableInTime` *is Mathlib's notion: a machine, a time function of the input length, and a proof that on every input the machine halts with the encoded output within that time;* `encodingNatΓ'` *is Mathlib's binary encoding of* $\mathbb{N}$*, and* `Nodup`*,* `Prime`*,* `prod` *are Mathlib's. Korselt's criterion is not invoked: the Carmichael property appears inlined as "composite and* $m \mid a^m - a$ *for every integer* a*." Mathlib charges one step per transition of the finite control, and the polynomial overhead of a stack machine over bit operations is absorbed in* C*.*

The machine implements Section 3 with two differences. The two results cited from [1] *are replaced by weaker statements proved inside the formalization from a zero-density estimate for Dirichlet L-functions established there: Lemma 2.1 for some exponent* $E \in (0, 1/2]$ *in place of every* $E < 5/12$*, so the machine uses* $y = \lceil z^{1-E} \rceil$ *for that* E*, and Lemma 2.2 at* $B = 21/100$ *in place of* $2/5$ *(any* $B \geq 1/5$ *serves, since every divisor of* $L = x^{1/5}$ *is at most* x^B*), so the scan bound becomes* $k \leq x^{79/100}$*. And every primality test, including the certificate check of step 5, is trial division, which the time bound absorbs; the polylogarithmic verification of the certificate rests on Lemma 2.5 and is not formalized. The constants* C_1 *and* E *enter the machine through the axiom of choice, so "computable in principle" is a claim about the proof above; the formalization asserts existence only. Two intermediate layers are also proved and audited: the existence statement alone, and the algorithm as a Lean function with an explicit operation count at most* $e^{100\,\ell_2\ell_3}$*. Korselt's criterion, Lemma 2.3, Lemma 2.4 (to our knowledge the first mechanization of the van Emde Boas–Kruyswijk bound in any proof assistant), and the Chebyshev estimate used in Section 4 are all proved inside the formalization. The repository is* https://github.com/jdb19937/carmichael.

References

[1] W. R. Alford, A. Granville, C. Pomerance, *There are infinitely many Carmichael numbers*, Ann. of Math. (2) **140** (1994), 703–722. doi:10.2307/2118576.

[2] M. Agrawal, N. Kayal, N. Saxena, *PRIMES is in P*, Ann. of Math. (2) **160** (2004), 781–793. doi:10.4007/annals.2004.160.781.

[3] W. R. Alford, J. Grantham, S. Hayman, A. Shallue, *Constructing Carmichael numbers through improved subset-product algorithms*, Math. Comp. **83** (2014), 899–915. doi:10.1090/S0025-5718-2013-02737-8.

[4] A. Korselt, *Problème chinois*, L'intermédiaire des mathématiciens **6** (1899), 142–143.

[5] J. C. Lagarias, A. M. Odlyzko, *Computing* $\pi(x)$*: an analytic method*, J. Algorithms **8** (1987), 173–191. doi:10.1016/0196-6774(87)90037-X.

[6] D. Larsen, *Bertrand's postulate for Carmichael numbers*, Int. Math. Res. Not. IMRN **2023**, no. 15, 13072–13098. doi:10.1093/imrn/rnac203.

[7] G. Löh, W. Niebuhr, *A new algorithm for constructing large Carmichael numbers*, Math. Comp. **65** (1996), 823–836. doi:10.1090/S0025-5718-96-00692-8.

[8] J. B. Rosser, L. Schoenfeld, *Approximate formulas for some functions of prime numbers*, Illinois J. Math. **6** (1962), 64–94. doi:10.1215/ijm/1255631807.

[9] D. H. J. Polymath, *Deterministic methods to find primes*, Math. Comp. **81** (2012), 1233–1246. doi:10.1090/S0025-5718-2011-02542-1.

[10] A. Shallue, J. Webster, *Algorithms for Carmichael numbers*, arXiv:2506.09903 (2025).

[11] P. van Emde Boas, D. Kruyswijk, *A combinatorial problem on finite abelian groups III*, Report ZW-1969-008, Mathematisch Centrum, Amsterdam, 1969. ir.cwi.nl/pub/7217.

Email address: jdb1729@gmail.com