hexagon:2611.00785v1

# ERDŐS 196: A PERMUTATION OF THE NATURAL NUMBERS WITH NO MONOTONE FOUR-TERM ARITHMETIC PROGRESSION

JENSEN KOHLMEYER AND LIAM KRUER

Abstract. We construct a permutation of the natural numbers containing no increasing or decreasing four-term arithmetic progression as a subsequence. This gives a negative answer to Erdős problem 196. The construction extends finite prefixes while preserving two conditions: every ordered three-term progression with a nonnegative fourth term has that term already placed before its third, and the requirements imposed by unfinished progressions agree at each binary residue node. To extend a prefix, we order a new finite block by an adapted binary order and impose saturation only at nodes where the old prefix branches. Finiteness follows from the weight $w(n) = 2^{\rho(n)}n$, where $\rho$ is a finite binary residue rank. The extension preserves both conditions and can include any prescribed finite set. Iteration therefore gives a bijection of order type $\omega$ on all the natural numbers. The finite extension theorem, the resulting bijection, and the exact negative answer are formalized in Lean 4.

## 1. Introduction

Write $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$. A permutation of $\mathbb{N}_0$ means a bijection $p : \mathbb{N}_0 \to \mathbb{N}_0$, viewed as the sequence $p(0), p(1), \ldots$. A monotone four-term arithmetic progression in this sequence consists of indices $i < j < k < \ell$ and integers $a, h$, with $h \neq 0$, such that

$$(p(i), p(j), p(k), p(\ell)) = (a, a + h, a + 2h, a + 3h).$$

Both signs of $h$ are allowed. Equivalently, the four values satisfy

$$p(i) + p(k) = 2p(j), \qquad p(j) + p(\ell) = 2p(k). \tag{1.1}$$

Their distinctness follows from injectivity. These equalities also avoid any ambiguity about subtraction in the natural numbers. Davis, Entringer, Graham, and Simmons [3] proved that every permutation of the positive integers contains a monotone three-term arithmetic progression and constructed a permutation containing no monotone five-term progression. The remaining four-term question is recorded as Erdős problem 196 in the Formal Conjectures library [5] and on Conjectures.io [2]. We prove the following.

**Theorem 1.1.** *There exists a bijection $p : \mathbb{N}_0 \to \mathbb{N}_0$ for which no indices $i < j < k < \ell$ satisfy both equalities in* (1.1).

Adding one to each value and shifting the starting index gives the same conclusion for permutations of the positive integers.

Binary residue orders provide an elementary way to order every finite set of integers without a monotone three-term progression. Related orders on infinite sets are classical; see Ardal, Brown, and Jungić [1]. Such an order need not enumerate its underlying set in a sequence indexed by $\mathbb{N}_0$: an element can have infinitely many predecessors. Our construction instead preserves an entire finite prefix at each step.

*Date*: September 9, 2026.

2020 *Mathematics Subject Classification.* Primary 11B25; Secondary 05A05, 68V20.

*Key words and phrases.* Erdős problem 196, infinite permutations, arithmetic progressions, binary orders, finite extension, Lean formalization.

There are also strong density results for subsets admitting permutations that avoid monotone progressions. Geneson [4] obtains density supremum results for lengths at least four and explicitly distinguishes them from an avoiding permutation of all the natural numbers. Theorem 1.1 addresses the full underlying set directly.

The main point is a finite extension theorem. A prefix records both the fourth terms that must already have appeared and the binary preferences required by progressions whose last two terms are still absent. We impose saturation at the finitely many nodes where the old prefix occupies both children. After the new block is appended, the future requirements keep their orientation at those nodes and reverse it at the other nodes. This update is what makes repeated extension possible. A weighted finite set supplies the needed saturation without a limiting argument.

Sections 2 and 3 establish the order and finiteness lemmas. Sections 4 and 5 prove the extension theorem. Section 6 constructs the permutation, and Section 7 describes the accompanying complete formalization.

## 2. Binary residue orders

A binary residue node is a pair

$$\eta = (k, r), \qquad k \in \mathbb{N}_0, \qquad 0 \le r < 2^k.$$

It represents the class $C_\eta = r + 2^k\mathbb{N}_0$. For $x \in C_\eta$, let

$$\mathrm{bit}_\eta(x) = \frac{x - r}{2^k} \pmod 2 \in \{0, 1\}.$$

The two children of $\eta$ are the subsets with the two possible values of $\mathrm{bit}_\eta$. A preference assignment $e$ chooses a value $e(\eta) \in \{0, 1\}$ at every node; its child is called preferred.

For distinct $a, b \in \mathbb{N}_0$, let

$$\eta(a, b) = (k, r), \qquad k = \nu_2(|b - a|), \qquad r = a \bmod 2^k. \tag{2.1}$$

This is their first differing binary node when digits are read from least to most significant. Define

$$a \prec_e b \iff \mathrm{bit}_{\eta(a,b)}(a) = e(\eta(a, b)). \tag{2.2}$$

**Lemma 2.1.** *The relation $\prec_e$ is a strict total order. For every $K \in \mathbb{N}_0$ there is a rank function*

$$\rho_{e,K} : \mathbb{N}_0 \longrightarrow \{0, 1, \ldots, 2^K - 1\},$$

*depending only on the residue modulo $2^K$, such that if the node $\eta(a, b)$ has depth less than $K$, then*

$$a \prec_e b \iff \rho_{e,K}(a) < \rho_{e,K}(b).$$

*Proof.* List the leaves of the binary residue tree of depth $K$ by recursively listing the preferred child before the other child. Give the leaves their ranks, starting with zero. The comparison of two distinct leaves is decided at their first differing node and is exactly (2.2). This constructs $\rho_{e,K}$. For any finite set of distinct integers, choose $K$ large enough that their residues modulo $2^K$ are distinct. The relation $\prec_e$ on that set agrees with the finite leaf order. In particular, every pair is comparable and every triple satisfies transitivity. □

The arithmetic fact used throughout the proof is the alternation of the two children along a progression. For four values, the progression equations are

$$a + c = 2b, \qquad b + d = 2c. \tag{2.3}$$

**Lemma 2.2.** *Suppose $a, b, c \in \mathbb{N}_0$, $a \ne b$, and $a + c = 2b$. Then $\eta(a, b) = \eta(b, c) =: \eta$, all three values lie in $C_\eta$, and $\mathrm{bit}_\eta(a) = \mathrm{bit}_\eta(c) \ne \mathrm{bit}_\eta(b)$. If $d \in \mathbb{N}_0$ also satisfies $b + d = 2c$, then $\eta(c, d) = \eta$, all four values lie in $C_\eta$, and*

$$\mathrm{bit}_\eta(a) = \mathrm{bit}_\eta(c) \ne \mathrm{bit}_\eta(b) = \mathrm{bit}_\eta(d). \tag{2.4}$$

*Moreover, $c \le 2b$, and $d \le 2c$ whenever the fourth term is present.*

*Proof.* In $\mathbb{Z}$, put $h = b - a$. The three values are $a, a+h, a+2h$, with $d = a+3h$ when the fourth term is present. If $k = \nu_2(|h|)$, the values have the same residue modulo $2^k$, and their normalized adjacent differences are odd. Their $k$th digits therefore alternate. Adjacent differences equal $h$, so their first differing nodes agree. Finally, $c = 2b - a \le 2b$, and $d = 2c - b \le 2c$ when present, because $a, b \ge 0$. □

**Lemma 2.3.** *There are no distinct $a, b, c \in \mathbb{N}_0$ with $a \prec_e b \prec_e c$ and $a + c = 2b$.*

*Proof.* At the first differing node of $a$ and $b$, the values $a, c$ have the same digit and $b$ has the other digit, by the three-term part of Lemma 2.2. If the digit of $b$ is preferred, then $b$ precedes both endpoints; otherwise it follows both. In neither case is $b$ between them. □

## 3. Finite saturation by a weight

We first give a finiteness lemma that applies to any natural-valued rank.

**Lemma 3.1.** *Let $\rho : \mathbb{N}_0 \to \mathbb{N}_0$ and let $F \subseteq \mathbb{N}_0$ be finite. There is a finite set $A \supseteq F$ such that*

$$\rho(x) < \rho(y), \qquad x \le 2y, \qquad y \in A \quad \Longrightarrow \quad x \in A. \tag{3.1}$$

*Proof.* Define

$$w(n) = 2^{\rho(n)} n, \qquad M = \max(\{w(n) : n \in F\} \cup \{0\}), \qquad A = \{n \in \mathbb{N}_0 : w(n) \le M\}.$$

Since $n \le w(n)$, the set $A$ is contained in $\{0, \dots, M\}$ and is finite. It contains $F$ by the choice of $M$. If $\rho(x) < \rho(y)$ and $x \le 2y$, then $\rho(x) + 1 \le \rho(y)$, so

$$w(x) = 2^{\rho(x)} x \le 2^{\rho(x)+1} y \le 2^{\rho(y)} y = w(y). \tag{3.2}$$

Thus $y \in A$ implies $x \in A$. □

**Definition 3.2.** Let $R$ be a finite set of binary residue nodes and let $e$ be a preference assignment. A set $A \subseteq \mathbb{N}_0$ is saturated at $(R, e)$ if, for every $\eta \in R$ and $x, y \in C_\eta$,

$$\left.\begin{array}{c} \mathrm{bit}_\eta(x) = e(\eta), \quad \mathrm{bit}_\eta(y) \ne e(\eta), \\ x \le 2y, \quad y \in A \end{array}\right\} \quad \Longrightarrow \quad x \in A. \tag{3.3}$$

Thus a nonpreferred value brings into $A$ every preferred value at the same specified node that is at most twice as large. The inequality concerns the values $x, y$ themselves.

**Corollary 3.3.** *For every finite $R$, preference assignment $e$, and finite seed set $F$, there is a finite superset of $F$ saturated at $(R, e)$.*

*Proof.* Choose $K$ greater than the depth of every node in $R$; take $K = 0$ if $R$ is empty. Use Lemma 3.1 with the rank $\rho_{e,K}$ from Lemma 2.1. If $x, y$ satisfy the hypotheses of (3.3), they first differ at $\eta \in R$, and the digit of $x$ is preferred. Hence $\rho_{e,K}(x) < \rho_{e,K}(y)$, and (3.1) gives $x \in A$. □

Only finitely many nodes are used in this argument. The conclusion does not require saturation at every node of the infinite binary tree.

## 4. Admissible finite prefixes

Let $P$ be a finite list of distinct members of $\mathbb{N}_0$, let $S = \mathrm{supp}(P)$ be its set of entries, and write $a <_P b$ if $a, b \in S$ and $a$ occurs before $b$ in $P$. The notation $<_P$ always refers to positions, whereas $<$ and $\le$ without a subscript refer to numerical order.

**Definition 4.1.** The list $P$ is admissible if it satisfies both conditions below.

(1) Closure. For all $a, b, c, d \in \mathbb{N}_0$ satisfying (2.3),

$$a <_P b <_P c \quad \Longrightarrow \quad d <_P c. \tag{4.1}$$

In particular, the conclusion requires $d \in S$.

(2) Compatibility. There exists a preference assignment $e_0$ such that, whenever $a <_P b$ and $c, d \in \mathbb{N}_0 \setminus S$ satisfy (2.3),

$$e_0(\eta(a,b)) = \mathrm{bit}_{\eta(a,b)}(b). \tag{4.2}$$

The compatibility condition says that all pairs imposing a requirement at a given node impose the same requirement. Indeed, for a progression beginning with $a <_P b$ and with $c, d$ both absent, any extension avoiding that progression must put $d$ before $c$. In a binary order, this is achieved by preferring the digit of $b$, which is also the digit of $d$.

The empty list is admissible. Also, every admissible list avoids monotone four-term progressions: a chain $a <_P b <_P c <_P d$ satisfying (2.3) would contradict (4.1).

Let $R(P)$ be the set of nodes at which $P$ branches, meaning that each child contains an element of $S$. Equivalently,

$$R(P) = \{\eta(a,b) : a, b \in S,\ a \neq b\}. \tag{4.3}$$

The set $R(P)$ is finite. Every compatibility requirement for $P$ occurs at one of these nodes. Fix a compatible assignment $e_0$. We use the following adapted assignment $e$:

(1) If $\eta \in R(P)$, set $e(\eta) = e_0(\eta)$.
(2) If $\eta \notin R(P)$ and $C_\eta$ meets $S$, prefer the unique child containing the old entries in that node.
(3) If $C_\eta$ does not meet $S$, set $e(\eta) = 0$.

This definition is consistent. It retains compatibility because it changes no preference at a node in $R(P)$. The list $P$ itself need not be ordered by $\prec_e$.

## 5. The finite extension theorem

**Theorem 5.1.** *Let $P$ be admissible and let $F \subseteq \mathbb{N}_0$ be finite. There is an admissible finite list $Q$ such that $P$ is an initial segment of $Q$ and $F \subseteq \mathrm{supp}(Q)$.*

*Proof.* Put $S = \mathrm{supp}(P)$ and choose the adapted assignment $e$ from Section 4. By Corollary 3.3, choose a finite set $A \supseteq S \cup F$ saturated at $(R(P), e)$. Let $B$ be the list of $A \setminus S$ in increasing $\prec_e$ order, and define

$$Q = P \parallel B. \tag{5.1}$$

Then $Q$ has distinct entries, has support $A$, extends $P$, and contains $F$. It remains to prove closure and compatibility. Call the entries of $P$ old and those of $B$ new. Every old entry precedes every new entry.

*Closure.* Suppose $a <_Q b <_Q c$ and $a, b, c, d \in \mathbb{N}_0$ satisfy (2.3). We show $d <_Q c$. If $c$ is old, then $a, b$ are old and closure of $P$ gives the conclusion. If $a, b, c$ are all new, their order agrees with $\prec_e$, contrary to Lemma 2.3. Thus we may assume that $c$ is new and $a$ is old. If $d$ is old, then it already precedes $c$, so we may also assume $d \notin S$. There are two cases, according to the status of $b$.

*Case 1: $a, b$ are old and $c$ is new.* Let $\eta = \eta(a,b)$. This is a node in $R(P)$. Since $c, d \notin S$, compatibility of $P$ and the choice of $e$ give

$$e(\eta) = \mathrm{bit}_\eta(b) = \mathrm{bit}_\eta(d).$$

By Lemma 2.2, the digit of $c$ is the other digit and $d \leq 2c$. As $c \in A$, saturation at $\eta$ gives $d \in A$. Our assumption $d \notin S$ now makes $d$ new. Its preferred digit puts it before $c$ in $\prec_e$, hence before $c$ in $B$ and in $Q$.

*Case 2: $a$ is old and $b, c$ are new.* The node $\eta = \eta(a,b) = \eta(b,c)$ is the same by Lemma 2.2. Because $b \prec_e c$, the digit of $b$ is preferred and the digits of $a, c$ are not. This forces $\eta \in R(P)$. Otherwise the adaptation rule at the nonbranching node containing the old value $a$ would prefer the digit of $a$, a contradiction. The digit of $d$ equals that of $b$, and $d \leq 2c$. Saturation at this old branching node therefore gives $d \in A$. As in Case 1, $d$ is new and precedes $c$ in $B$. All cases give $d <_Q c$, proving closure.

*Compatibility.* Define a new assignment on every node by

$$e^+(\eta) = \begin{cases} e(\eta), & \eta \in R(P), \\ 1 - e(\eta), & \eta \notin R(P). \end{cases} \tag{5.2}$$

We show that $e^+$ is compatible with $Q$. Take an ordered pair $a <_Q b$ and values $c, d \in \mathbb{N}_0 \setminus A$ satisfying (2.3), and put $\eta = \eta(a, b)$. We must prove

$$e^+(\eta) = \mathrm{bit}_\eta(b). \tag{5.3}$$

First suppose $\eta \in R(P)$. If $b$ is old, then $a$ is old as well. Since $c, d$ are outside $A \supseteq S$, compatibility of $P$ gives $e(\eta) = \mathrm{bit}_\eta(b)$, as required. If $b$ is new, suppose for a contradiction that $\mathrm{bit}_\eta(b) \neq e(\eta)$. Then $c$ has the preferred digit by Lemma 2.2, and $c \leq 2b$. The value $b$ belongs to $A$, so saturation at $\eta$ puts $c$ in $A$, contradicting its choice. Thus $\mathrm{bit}_\eta(b) = e(\eta)$ in this case also. This establishes (5.3) at every old branching node.

Now suppose $\eta \notin R(P)$. The values $a, b$ cannot both be old: their occurrence in opposite children would make $\eta$ a node of $R(P)$. Since $a <_Q b$, it follows that $b$ is new. If $a$ is old, the adapted assignment prefers the child containing $a$. Therefore the digit of $b$ is $1 - e(\eta)$. If $a$ is new, both values occur in the sorted block $B$, so $a \prec_e b$. Again the digit of $a$ is preferred and that of $b$ is $1 - e(\eta)$. In both cases (5.2) gives (5.3). Hence $e^+$ satisfies every compatibility requirement of $Q$. Together with closure and distinctness, this proves that $Q$ is admissible. □

The inequalities used in the proof are precisely $d \leq 2c$ in closure and $c \leq 2b$ in compatibility. They follow from nonnegativity of the first two terms, as recorded in Lemma 2.2, and apply to progressions with either sign of the common difference.

## 6. Construction of the infinite permutation

*Proof of Theorem 1.1.* Start with the empty list $P_0$. Given the admissible list $P_n$, apply Theorem 5.1 with $F = \{n\}$ to choose $P_{n+1}$. Thus

$$P_n \text{ is an initial segment of } P_{n+1}, \qquad n \in \mathrm{supp}(P_{n+1}). \tag{6.1}$$

Every $P_n$ is finite and admissible. Induction shows that it contains each integer less than $n$, so

$$|P_n| \geq n. \tag{6.2}$$

Here $|P_n|$ denotes the length of the list. The lengths need not increase strictly at every step; the lower bound (6.2) is sufficient. Index list positions from zero and define

$$p(i) = \text{the entry at position } i \text{ in } P_{i+1}.$$

This entry exists by (6.2). The initial-segment property ensures that its value agrees with the entry at position $i$ in any stage having that position: any two stages are comparable by inclusion as initial segments.

If $i \neq j$, choose a stage whose length exceeds both indices. Its entries are distinct and agree with $p$ at those positions, proving injectivity. For any $x \in \mathbb{N}_0$, the value $x$ belongs to $P_{x+1}$ by (6.1). If it is at position $i$ in that list, the same agreement gives $p(i) = x$. Thus $p$ is surjective. In particular, every value occupies a finite position, and this is a permutation of order type $\omega$.

Finally, suppose $i < j < k < \ell$ satisfy (1.1). The stage $P_{\ell+1}$ contains all four positions and agrees there with $p$. Its closure condition applied to the first three entries puts $p(\ell)$ before $p(k)$, contrary to $k < \ell$. This proves the theorem. □

The construction only needs an admissible extension at each step. In particular, it never reorders a previously placed value. The binary orders serve to sort the finite added blocks; the stages determine the order of the infinite sequence.

## 7. Lean formalization and reproducibility

The accompanying file `Erdos196.lean` contains the complete Lean proof, including every lemma specific to this construction. Its imports are

```
import FormalConjectures.ErdosProblems.«196»
import TaskSupport
```

The checked environment uses Lean 4.33.1 and Formal Conjectures commit `8432eac998110a563e03df65a28c117e97c8c142`.

The dependence on `TaskSupport` implements the exact target-type notation used in the submission bundle. The mathematical construction is contained in the namespace `Bounty.Construction196`. The correspondence between the paper and the principal declarations is shown below; namespace prefixes are omitted in the right-hand column.

| Mathematical component | Lean declaration |
|---|---|
| Binary comparison and rank | `code_lt_iff_split, rank_lt_of_split` |
| Lemma 2.3 | `binary_no_three` |
| Lemma 3.1 | `finite_saturation` |
| Corollary 3.3 | `saturated_exists` |
| Definition 4.1 | `Closed, Compatible, Good` |
| Theorem 5.1 | `finite_extension` |
| Nested stages and enumeration | `stage, permFun` |
| Injectivity and surjectivity | `permFun_injective, permFun_surjective` |
| Absence of four-term progressions | `permFun_no_four, counterexample` |
| Theorem 1.1 in index form | `Bounty.paper_main` |

In Lean, a binary residue is represented by its finite list of low bits. The relation `LowEq k a b` states equality of the bits below $k$; `code` uses the lexicographic order of preferred versus nonpreferred bits. The proof of `finite_saturation` is exactly the sublevel-set argument of Lemma 3.1. The stages are selected using classical choice from `finite_extension`. The formalized result is the existence theorem; it does not assert executable running-time bounds.

The main library-facing conclusion is

$$\exists f : \mathbb{N} \simeq \mathbb{N}, \qquad \neg\, \texttt{HasMonotoneAP}\ f\ 4.$$

Here Lean's $\mathbb{N}$ includes zero, and $\simeq$ denotes a bijection. The predicate `HasMonotoneAP` permits either orientation of an arithmetic progression along a strictly increasing list of indices. The proof converts that predicate to the two equalities in (1.1) and applies `permFun_no_four`. The declaration `Bounty.paper_main` states Theorem 1.1 directly in index form. The declaration `Bounty.target` proves the negation of the pinned catalogue type. In ASCII Lean notation, the final step is

```
theorem target : Not (fcTypeOfName% "Erdos196.erdos_196") := by
  intro h
  cases Construction196.counterexample with
  | intro f hf => exact hf (h.mp True.intro f)
```

The source type is `True` $\leftrightarrow \forall f : \mathbb{N} \simeq \mathbb{N}$, `HasMonotoneAP` $f$ 4; the displayed argument negates that affirmative assertion.

Local compilation, the validator's static checks, and its exact-target inspector passed for the submission. The standalone companion and its explicit index theorem were also compiled. The axiom audit for the finite extension theorem, the counterexample, the index theorem, and the final target lists only

```
propext, Classical.choice, Quot.sound
```

These are standard axioms of the Lean environment. There are no admitted proofs or new axioms in the supplied construction, and its conclusions do not depend on the admitted open conjecture in the imported catalogue.

The supplementary archive contains both the standalone source and `Erdos196-submission.lean`, the body formatted for the catalogue's Counterexample-mode wrapper. The latter omits imports and the outer namespace because the validator supplies them. The archive records the dependency revision, file hashes, and the local checking results.

## Acknowledgment of AI assistance

OpenAI's Codex assistant was used extensively in developing the mathematical argument, checking related literature, producing and debugging the Lean 4 formalization, and drafting and typesetting this manuscript. This assistance included substantive proof ideas and generated proof code. The named authors are responsible for the final manuscript and its submission.

## References

[1] H. Ardal, T. Brown, and V. Jungić, *Chaotic orderings of the rationals and reals*, Amer. Math. Monthly 118 (2011), no. 10, 921–925. doi:10.4169/amer.math.monthly.118.10.921.

[2] Conjectures.io, *Erdős problem 196*, `https://conjectures.io/problems/erdos196-erdos-196`. Accessed September 9, 2026.

[3] J. A. Davis, R. C. Entringer, R. L. Graham, and G. J. Simmons, *On permutations containing no long arithmetic progressions*, Acta Arith. 34 (1977), no. 1, 81–90. doi:10.4064/aa-34-1-81-90.

[4] J. Geneson, *Density bounds for permutations avoiding monotone arithmetic progressions*, preprint, arXiv:2608.12604v1 (2026). `https://arxiv.org/abs/2608.12604v1`.

[5] Google DeepMind, *Formal Conjectures*, file `FormalConjectures/ErdosProblems/196.lean`, revision `8432eac998110a563e03df65a28c117e97c8c142`. `https://github.com/google-deepmind/formal-conjectures`.

*Email address*: `jenwin@purdue.edu`

*Email address*: `kruerl@purdue.edu`