

Conjectures.io

Formal proof verified through Conjectures

ERDŐS PROBLEM 196: A PERMUTATION AVOIDING FOUR-TERM ARITHMETIC PROGRESSIONS

LIAM KRUEER AND JENSEN KOHLMAYER

ABSTRACT. We construct a permutation of the natural numbers having no four-term arithmetic progression as a subsequence, in either increasing or decreasing value order. This answers Erdős Problem 196 negatively. The construction extends finite lists without changing earlier positions. Each list carries a closure condition on three-term progressions and a compatible family of binary ordering preferences. A finite saturation lemma supplies the extra values needed at each step; a lexicographic order on binary digits arranges them. The resulting nested lists eventually contain every natural number, and their limit is the required permutation. We give the extension argument in detail and identify its correspondence with the accepted Lean proof.

1. THE PROBLEM AND THE RESULT

We use $\mathbb{N} = \{0, 1, 2, \dots\}$. A permutation is a bijection $f : \mathbb{N} \rightarrow \mathbb{N}$, regarded as the sequence $f(0), f(1), \dots$. A four-term arithmetic progression in this sequence means indices $i < j < k < \ell$ satisfying

$$f(i) + f(k) = 2f(j), \quad f(j) + f(\ell) = 2f(k). \quad (1)$$

Because f is injective, its common difference is nonzero; the equations therefore cover both increasing and decreasing progressions. The chosen indices need not be consecutive or equally spaced.

Theorem 1.1. *There is a bijection $f : \mathbb{N} \rightarrow \mathbb{N}$ for which no indices $i < j < k < \ell$ satisfy (1).*

Replacing $f(n)$ by $f(n-1)+1$ for $n \geq 1$ gives the corresponding permutation of the positive integers. Translation preserves the equations, so the choice of zero-based indexing has no mathematical effect.

1.1. Historical context. Davis, Entringer, Graham, and Simmons [1] studied permutations avoiding long arithmetic progressions. In the first concluding question on printed page 88 of their 1977 paper, they explicitly asked whether every singly infinite permutation of the positive integers must contain a monotone four-term progression. This is the question now recorded as Erdős Problem 196 [2]. Their work distinguishes singly infinite permutations from doubly infinite orderings; the theorem here concerns the former. The date 1977 is a documented publication date for the exact question, rather than a claim about its first private proposal.

Date: Working draft — 15 September 2026.

Key words and phrases. Arithmetic progressions, permutations, binary order, finite extension, Lean formalization.

This working manuscript is a prose exposition prepared from the accepted Lean source; it is not an author-approved journal publication.

1.2. Why an extension argument is needed. Binary-digit orders readily arrange any finite set without an ordered three-term progression. Such an order on all of $\mathbb{N}$, however, need not have the order type of $\mathbb{N}$: an element may have infinitely many predecessors. Simply sorting all natural numbers in a binary order does not construct the required permutation.

Instead we append one finite block at a time. The old block keeps its positions, while a new binary order arranges the new block. A closure condition prevents four-term progressions from straddling the boundary. A second condition guarantees that this process can continue and can include any prescribed finite set. Those conditions are the substance of the construction.

2. BINARY PREFERENCES AND ARITHMETIC PROGRESSIONS

For $a \in \mathbb{N}$, let $a_k \in \{0, 1\}$ be its k th binary digit, starting at the least significant digit, and let $\text{low}_k(a) = (a_0, \dots, a_{k-1}) \in \{0, 1\}^k$. Write $a \equiv_k b$ when these words agree, equivalently when $a \equiv b \pmod{2^k}$.

A *preference system* chooses a bit $e(k, w) \in \{0, 1\}$ for every $k \geq 0$ and $w \in \{0, 1\}^k$. Define the key of a at level k by

$$\kappa_e(a)_k = \begin{cases} 0, & a_k = e(k, \text{low}_k(a)), \\ 1, & a_k \neq e(k, \text{low}_k(a)). \end{cases}$$

Order natural numbers by lexicographically comparing their keys from the least significant level upward; denote this strict order by $<_e$. Distinct natural numbers have a first differing digit, so this is a strict total order. More explicitly, if k is the first differing digit of a, b , and $w = \text{low}_k(a) = \text{low}_k(b)$, then

$$a <_e b \iff a_k = e(k, w). \quad (2)$$

This comparison rule also proves injectivity of the key encoding.

Lemma 2.1 (Alternating bits). *Suppose $a, b, c, d \in \mathbb{N}$ satisfy*

$$a + c = 2b, \quad b + d = 2c, \quad (3)$$

and k is the first differing binary digit of a and b . All four values have the same word w of length k , and their bits at level k alternate:

$$(a_k, b_k, c_k, d_k) = (\alpha, \beta, \alpha, \beta), \quad \alpha \neq \beta.$$

Moreover $c \leq 2b$ and $d \leq 2c$.

Proof. In integer arithmetic, put $h = b - a$. Its exact power of two divisor is 2^k , since a, b agree below k and disagree at k . The four terms are $a, a + h, a + 2h, a + 3h$. They have the same residue modulo 2^k , and adding h toggles bit k each time. This reasoning also applies to negative h . Finally $c = 2b - a \leq 2b$ and $d = 2c - b \leq 2c$, since $a, b \geq 0$. $\square$

Lemma 2.2 (Binary order excludes three terms). *There are no $a <_e b <_e c$ with $a + c = 2b$.*

Proof. At the first differing digit of a, b , the endpoints a, c have the same bit and b the opposite bit. All three share the preceding word. The two order comparisons would require both opposite bits to be preferred at that same word, contradicting (2). $\square$

For later use, truncate the key to K digits and let $\rho_{e,K}(a)$ be the rank of that word in lexicographic order on $\{0, 1\}^K$. Thus $0 \leq \rho_{e,K}(a) < 2^K$. If a, b first differ at a level $k < K$ and a_k is preferred, then

$$\rho_{e,K}(a) < \rho_{e,K}(b). \quad (4)$$

3. FINITE LISTS AND FINITE SATURATION

For a finite list P without repetitions, write $a \triangleleft_P b$ when a, b occur in P and the occurrence of a precedes that of b . Write $a \in P$ for membership, without reference to position, and $\text{supp}(P)$ for the set of its entries.

Definition 3.1 (Branch). A word (k, w) is a *branch* of P if two elements of P have lower word w and opposite bits at level k .

Only finitely many words are branches of a finite list. In particular, choose K with $a < 2^K$ for every $a \in P$. At every level $k \geq K$, all elements have bit zero, so each branch has level less than K .

Definition 3.2 (Closure and compatibility). The list P is *closed* if, whenever (3) holds and $a \triangleleft_P b \triangleleft_P c$, one has $d \triangleleft_P c$. A preference system e is *compatible with P* if the following condition holds: whenever (3) holds, $a \triangleleft_P b$, and $c, d \notin P$, then at the first differing digit k of a, b ,

$$e(k, \text{low}_k(a)) = b_k. \quad (5)$$

A list is *good* if it has no repetitions, is closed, and has a compatible preference system.

Closure implies that P has no ordered four-term progression: its first three entries would force the fourth to precede the third. Compatibility prescribes the future ordering of two missing terms. It is only a restriction at branches already present in P . The empty list is good.

Lemma 3.3 (Finite saturation). *For every map $\rho : \mathbb{N} \rightarrow \mathbb{N}$ and finite $F \subset \mathbb{N}$, there is a finite set $A \supseteq F$ such that*

$$\rho(a) < \rho(b), \quad a \leq 2b, \quad b \in A \implies a \in A. \quad (6)$$

Proof. Set $W(n) = 2^{\rho(n)}n$ and let $M = \max(\{W(n) : n \in F\} \cup \{0\})$. Take $A = \{n \in \mathbb{N} : W(n) \leq M\}$. Since $n \leq W(n)$, this set is finite, and it contains F . Under the hypotheses in (6),

$$W(a) \leq 2^{\rho(a)}2b = 2^{\rho(a)+1}b \leq 2^{\rho(b)}b = W(b) \leq M.$$

Thus $a \in A$, as required. $\square$

Definition 3.4 (Saturation relative to a list). A finite set A is *saturated for (P, e)* if, at every branch (k, w) of P , the conditions

$$\text{low}_k(x) = \text{low}_k(y) = w, \quad x_k = e(k, w) \neq y_k, \quad x \leq 2y, \quad y \in A$$

imply $x \in A$.

Corollary 3.5. *For any P, e and finite F , a finite set $A \supseteq F$ saturated for (P, e) exists.*

Proof. Choose K above all branch levels of P and apply Lemma 3.3 with $\rho = \rho_{e,K}$. The two values in Definition 3.4 first differ at the branch level; hence (4) gives $\rho(x) < \rho(y)$. $\square$

4. EXTENDING A GOOD LIST

Theorem 4.1 (Finite extension). *If P is good and $F \subset \mathbb{N}$ is finite, then a good list Q exists which has P as a prefix and contains every member of F .*

Fix a compatible preference system for P . We first adapt it at words which are not branches of P . If elements of P have lower word w at such a level, they all have the same next bit; prefer that bit. If no element has that word, choose either bit, say zero. Leave preferences at branches unchanged. Denote the adapted system by e . It is still compatible, since the conditions in (5) concern two old elements splitting at a branch. The adaptation has the additional property

$$(k, w) \text{ is not a branch of } P, a \in P, \text{low}_k(a) = w \implies e(k, w) = a_k. \quad (7)$$

Choose by Corollary 3.5 a finite saturated set $A \supseteq \text{supp}(P) \cup F$. List the elements of $A \setminus \text{supp}(P)$ in increasing $<_e$ order, obtaining a list B . Set $Q = PB$, the concatenation. It has no repetitions, has support A , and extends P . We must show closure and compatibility.

4.1. Closure of the extended list. Suppose (3) holds and $a \triangleleft_Q b \triangleleft_Q c$. If $c \in P$, then $a, b \in P$ as well, and closure of P gives $d \triangleleft_P c$, hence $d \triangleleft_Q c$. We may therefore assume $c \in B$. If $d \in P$, it automatically precedes c , so also assume $d \notin P$.

Let k be the first differing digit of a, b , put $w = \text{low}_k(a)$, and write the alternating bits as $(\alpha, \beta, \alpha, \beta)$ as in Lemma 2.1. The following observation handles the nontrivial cases:

$$(k, w) \text{ is a branch of } P \text{ and } e(k, w) = \beta \implies d \triangleleft_Q c. \quad (8)$$

Indeed, $c \in A$, $d \leq 2c$, and d has the preferred bit while c has the other bit. Saturation gives $d \in A$. Since $d \notin P$, it lies in B , where (2) orders d before c .

There are three possibilities for a, b :

- (1) If $b \in P$, then $a \in P$. Their differing bits give a branch of P , and compatibility gives $e(k, w) = \beta$, since $c, d \notin P$. Apply (8).
- (2) If $a \in P$ and $b \in B$, then $b, c \in B$, so $b <_e c$. Their bits are β, α , which implies $e(k, w) = \beta$. The word must be a branch of P : otherwise (7) applied to a gives $e(k, w) = \alpha$, a contradiction. Again apply (8).
- (3) If $a, b \in B$, then all three of a, b, c are in increasing $<_e$ order, contradicting Lemma 2.2.

This proves that Q is closed. No assumption about the sign of the common difference was used.

4.2. Preferences for the next extension. Define a new system by retaining preferences at old branches and flipping them elsewhere:

$$e'(k, w) = \begin{cases} e(k, w), & \text{if } (k, w) \text{ is a branch of } P, \\ 1 - e(k, w), & \text{otherwise.} \end{cases} \quad (9)$$

We prove it is compatible with Q . Suppose (3) holds, $a \triangleleft_Q b$, and $c, d \notin Q$. Let k, w, α, β be as above. We need $e'(k, w) = \beta$.

First suppose (k, w) is an old branch. If $b \in P$, then also $a \in P$, so compatibility for P already gives $e(k, w) = \beta$. If $b \in B$ and $e(k, w) \neq \beta$, then $e(k, w) = \alpha$. The value c has this preferred bit and satisfies $c \leq 2b$. Since $b \in A$, saturation forces $c \in A = \text{supp}(Q)$, a contradiction. Thus again $e'(k, w) = e(k, w) = \beta$.

Now suppose (k, w) is not an old branch. We cannot have $b \in P$, since then $a, b \in P$ would make it a branch. Hence $b \in B$. If $a \in P$, adaptation gives $e(k, w) = \alpha$. If $a \in B$, the order $a <_e b$ gives the same conclusion. In either case the flip in (9) gives $e'(k, w) = 1 - \alpha = \beta$. Compatibility is proved. Together with closure and absence of repetitions, this completes the proof of Theorem 4.1.

5. THE INFINITE PERMUTATION

Begin with P_0 empty. Applying Theorem 4.1 at step n with $F = \{n\}$ gives a good list P_{n+1} extending P_n and containing n . Thus

$$P_0 \text{ is a prefix of } P_1 \text{ is a prefix of } P_2 \text{ is a prefix of } \dots, \quad \{0, \dots, n-1\} \subseteq \text{supp}(P_n).$$

Since the lists have no repetitions, $|P_n| \geq n$. Define

$$f(i) = \text{the entry in position } i \text{ of } P_{i+1}, \quad (10)$$

using zero-based positions. The length bound makes this well-defined. Prefix compatibility implies that whenever $i < |P_t|$, the entry in position i of P_t equals $f(i)$: compare the two nested lists P_t and P_{i+1} .

Proof of Theorem 1.1. For distinct indices i, j , choose $t > \max\{i, j\}$. The two values $f(i), f(j)$ are distinct entries of the repetition-free list P_t . Thus f is injective. For every $x \in \mathbb{N}$, the value x occurs in P_{x+1} , say at position i . Agreement with that finite list gives $f(i) = x$. Thus f is surjective.

Finally suppose $i < j < k < \ell$ satisfy (1). In $P_{\ell+1}$, let a, b, c, d be the values in those four positions. Then $a \triangleleft b \triangleleft c \triangleleft d$ and (3) holds. Closure of that list forces $d \triangleleft c$, a contradiction. Hence f avoids all four-term progressions in index order. $\square$

Remark 5.1. The proof supplies a finite extension for any good prefix and any finite set of required values. The Lean implementation selects extensions using classical choice. No numerical efficiency bound or closed formula for the resulting permutation is asserted here.

6. FORMAL PROOF AND VERIFICATION

The accepted source consists of 635 lines of Lean and proves

```
Construction196.counterexample :
   $\exists f : \mathbb{N} \simeq \mathbb{N}, \neg \text{HasMonotoneAP } f \ 4.$ 
```

The source is placed in the validator’s `Bounty` namespace. Its final `target` declaration refutes the exact pinned `Erdos196.erdos_196` statement. In that environment, `HasMonotoneAP` requires increasing indices and accepts both increasing and decreasing arithmetic progressions of values. This is exactly the condition expressed by (1).

Exposition	Principal Lean declarations
Binary comparison and Lemma 2.1	<code>first_difference</code> , <code>code_lt_iff_split</code> , <code>progression_bits</code>
Lemma 2.2	<code>binary_no_three</code>
Lemma 3.3	<code>finite_saturation</code>
Good lists and saturated sets	<code>Closed</code> , <code>Compatible</code> , <code>Good</code> , <code>Saturated</code>
Corollary 3.5	<code>saturated_exists</code>
Preparation of the new block	<code>adapt</code> , <code>adapt_compatible</code> , <code>sorted_block</code>
Closure and compatibility of the extension	<code>extend_closed</code> , <code>extend_compatible</code>
Theorem 4.1	<code>finite_extension</code>
Construction of the limit	<code>stage</code> , <code>permFun</code> , <code>permFun_agrees</code>
Bijection and progression avoidance	<code>permFun_injective</code> , <code>permFun_surjective</code> , <code>permFun_no_four</code> , <code>counterexample</code>

6.1. Artifact identity and scope. The public result [4] records the submission as verified, approved, and rewarded. Its exact source was retrieved on 15 September 2026 and matched byte-for-byte against the retained accepted submission. The production verification report records successful statement comparison, task and dependency integrity, permitted-axiom checks, and Lean-kernel acceptance. A retained isolated replay from 10 September 2026 also reports acceptance. An additional integer diagnostic checks eight finite extension stages, including closure, compatible preferences, and preservation of earlier positions. These finite checks supplement the infinite argument. The verification records concern the Lean proof. The present prose manuscript was prepared separately and is not itself a machine-checked artifact.

Submission.: e73b95f7-1d1b-42b5-a442-c07077741d73.

Lean.: leanprover/lean4:v4.33.1.

Pinned source revision.:

8432eac998110a563e03df65a28c117e97c8c142.

Proof SHA-256.:

e65c98a926ce4eca5df30277790c8b3dbcc630074ceaa24a551464951b3d2826.

Task bundle SHA-256.:

db4b31179fca49ad899184caa87b901b2c11143b8769bfebae5120e6a155523d.

Permitted axioms.: propext, Quot.sound, Classical.choice. The recorded runs used Lean’s default kernel; the optional independent kernel was not enabled.

6.2. Public proof and report. The [result page](#) links the accepted proof and the review rationale. The exact source is available from the [public solution endpoint](#); the [public verification report](#) records the checks and immutable task identity.

REFERENCES

- [1] J. A. Davis, R. C. Entringer, R. L. Graham, and G. J. Simmons, *On permutations containing no long arithmetic progressions*, Acta Arithmetica **34** (1977), 81–90. See Concluding remarks, question 1, p. 88. [Journal archive scan](#).
- [2] T. F. Bloom (ed.), *Erdős Problem 196*, <https://www.erdosproblems.com/196>.
- [3] Google DeepMind and contributors, *Formal Conjectures*, FormalConjectures/ErdosProblems/196.lean, revision 8432eac998110a563e03df65a28c117e97c8c142. [Pinned statement](#).
- [4] Liam Krueer and Jensen Kohlmeier, *Accepted Lean submission for Erdős Problem 196*, Conjectures.io, September 2026. [Public result and proof](#).