Some Patterns for Generating Permutations on the Natural Numbers

Robert Daland
Siri Natural Language Understanding
Apple, Inc
r.daland@gmail.com

Abstract

Permutations on the natural numbers (hereafter: natperms) are theoretically uncountable. Despite this theoretical fecundity, concrete examples of natperms are not well exemplified in the literature. This paper begins by outlining some generator patterns for "simple" natperms. It is shown that for the base examples, the maximum distance between an element and its image is either constant or scales with the element's magnitude. Then the paper introduces a "nonsimple" natperm, in which the maximum displacement appears to scale supra-exponentially.

1 Introduction

What can a permutation on the natural numbers (a *natperm*) look like? And what constraints does the structure of the natural numbers impose on the space of permutations of this countably infinite set?

2 Notation and conventions

When it is necessary to distinguish the starting value for the natural numbers, the following notations will be used:

- $\mathbb{N} = \{1, 2, \ldots\}$
- $\mathbb{N}_0 = \{0, 1, \ldots\}$

Both sets will be referred to as "the natural numbers". This article will take the domain for natperms to be $\mathbb{N}$, i.e. starting from 1 rather than 0.

The symbol π will be used to denote generic natperms. Lowercase Greek letters will be used to denote specific natperms, with two notable exceptions: the add-1 map is denoted with $\sigma(n) = n + 1$, and the add-k map with $\sigma_k(n) = n + k$. These are exceptions because σ and σ_k are not natperms themselves (although this article will construct a variety of simple natperms from slight modifications of them).

The term **orbit** will refer to the collection of values generated by repeated application of a map, starting from some reference value. Orbits may include preimages of the reference value, i.e. a sequence of values may extend "backwards" when that is well-defined. The term **cycle** will be reserved for orbits in which the same value repeats after finitely many iterations of the map, i.e. $\pi^k(n) = n$ for some positive k. The **support** of a natperm is the set of values which are not fixed points.

Some of the examples discussed in this article can be represented as a product of *disjoint* permutations. However, composition of permutations with overlapping support is generally out of scope.

3 Patterns for generating natperms

This section introduces various means of constructing "simple" natperms, usually with some kind of variation of σ.

3.1 Finite natperms

Let ξ_0 be a bijection on a subset of the natural numbers. Define the *lifting* operation by constructing a natperm ξ whose value is the same as ξ_0 on its domain, and the identity mapping elsewhere:

$$\xi(n) = \begin{cases} \xi_0(n), & n \in \mathit{domain}(\xi_0) \\ n, & \text{otherwise} \end{cases} \tag{1}$$

A finite permutation is normally understood as a bijection on a set like $\{1, 2, \ldots, m\}$ for some m, i.e. what makes it finite is that the domain is finite. Every such bijection can be lifted to a natperm. In fact, it will prove convenient to apply the *lifting convention* throughout this article: lifting will be understood to apply automatically to eligible mappings (those denoted by lowercase Greek letters, with the exception of σ/σ_k). As a result of this convention, the domain of a bijection on a subset of the natural numbers is effectively the entire natural numbers, which is not a finite set. In this context, the notion of finiteness must be scoped to the support of the natperm rather than its domain. Accordingly, ξ is a *finite natperm* if and only if its support is a finite set.

3.2 Block cyclic natperms

Let $(b_j)_{j=1}^{\infty}$ be an increasing sequence of natural numbers. The reader may visualize this sequence as partitioning the natural numbers into a sequence of contiguous blocks: the starting block $B_1 = \{1 \le n < b_1\}$, and successor blocks $B_j = \{b_j \le n < b_{j+1}\}$ for all $j > 1$. A natperm is obtained by defining some kind of permutation on each block. A simple approach is to use the add-1 mapping on each block, "gluing" the top of each block to its bottom. Formally, given the sequence values (b_j), this is achieved as follows:

$$\beta(n) = \begin{cases} b_j, & n = b_{j+1} - 1 \\ n+1, & \text{otherwise} \end{cases} \tag{2}$$

This article will refer to such mappings as **block cyclic** natperms.

The simplest example is the parity-swapping map, which can be compactly represented with the cycle notation as $\phi = (1\ 2)(3\ 4)(5\ 6)\cdots$. Note that in the parity-swapping map, there is an upper bound on block size, i.e. it is a **bounded block cyclic** natperm.

It is also possible to define an **unbounded block cyclic** natperm. One way to do this is by setting the blocks to grow exponentially, e.g. $b_j = 2^j$.

3.3 Shift-and-glue natperms

Shift-and-glue natperms are closely related to the modulus group $\mathbb{Z}_k$, with some critical differences introduced by the fact that subsets of the natural numbers have a least element.

Let $\lfloor n \rfloor_k$ denote the orbit of the natural number n under the add-k map σ_k, i.e. the set $\{n, n+k, n+2k, \ldots\}$. This article will refer to every such set as a **strand**. Note that $\lfloor n \rfloor_k \supset \lfloor n+k \rfloor_k$. More generally, if n_1 and n_2 are two distinct natural numbers, either $\lfloor n_1 \rfloor_k$ and $\lfloor n_2 \rfloor_k$ are disjoint, or they are in a subset-superset relationship. A **maximal strand** is a strand which is not contained in any other strand. With the convention that natural numbers start at 1, the maximal strands for σ_k are $\lfloor 1 \rfloor_k$, $\lfloor 2 \rfloor_k$, …, $\lfloor k \rfloor_k$.

A shift-and-glue natperm is formed by gluing pairs of disjoint strands together. Gluing is accomplished by traversing one strand in descending order, switching over at the least elements, and traversing the other strand in ascending order. Here, "traversing" means that the natperm maps elements within a strand to the next larger or smaller element within the same strand. A simple example is given below:

$$\nu(n) = \begin{cases} n-2, & \text{if } n \text{ is even and greater than 2} \\ 1, & \text{if } n = 2 \\ n+2, & \text{if } n \text{ is odd} \end{cases} \tag{3}$$

The natperm can be more concisely represented by adopting a notational convention: if a strand appears on the left-hand side of a $\rightarrow$, it is traversed in decreasing order, while if it occurs on the right-hand side, it is traversed in ascending order. Thus, the natperm above can be written $\nu = \lfloor 2 \rfloor_2 \rightarrow \lfloor 1 \rfloor_2$.

Note that this convention excludes formulae like $\lfloor 1 \rfloor_3 \rightarrow \lfloor 2 \rfloor_3 \rightarrow \lfloor 3 \rfloor_3$, because that would imply that the same strand $\lfloor 2 \rfloor_3$ is traversed in both ascending and descending orders. However, because of the lifting convention, the formula $\lfloor 1 \rfloor_3 \rightarrow 2 \rightarrow \lfloor 3 \rfloor_3$ is allowed. This notation indicates that all elements of $\lfloor 2 \rfloor_3$ are subject to the identity mapping, except for 2 itself.

This notation highlights that shift-and-glue natperms with σ_k-strands are closely related to permutations on the mod group $\mathbb{Z}_k$. The key difference is that strands have a traversal order (ascending, descending, or identity), and all non-identity strands must be paired off in ascending/descending pairs. A slightly more complex example is shown in Fig. **??**, representing $\gamma = (\lfloor 1 \rfloor_6 \rightarrow \lfloor 5 \rfloor_6)(\lfloor 2 \rfloor_6 \rightarrow \lfloor 3 \rfloor_6)(\lfloor 6 \rfloor_6 \rightarrow \lfloor 4 \rfloor_6)$.

4 Maximum displacement

All of the examples presented in the previous section have the property that values are mapped to other, closeby values. The purpose of this section is to formalize what is meant by "closeby".

4.1 Definition

Let π be a natperm. Define the *displacement* of π at an input n as the distance between n and $\pi(n)$. Then the maximum displacement (up to a given value m) is $\Delta_m(\pi) = \max_{n \leq m} |\pi(n) - n|$. This article will summarize the limit behavior using big-O notation, as follows.

verbal description	big-O notation	definition
constant	$\pi \in \Theta(1)$	$\exists C \ni \lim_{m\rightarrow\infty} \Delta_m(\pi) = C$
scales linearly	$\pi \in \Theta(n)$	$\exists C \ni \lim_{m\rightarrow\infty} \frac{\Delta_m(\pi)}{m} = C$

4.2 Finite natperms

Claim: If π is a finite natperm, then $\pi \in \Theta(1)$.

Proof: By definition, the support of a finite natperm is a finite set N. If N is empty (because π is the identity mapping), the displacement is 0 everywhere, and $\pi \in \Theta(1)$. Otherwise, N must be nonempty. Since it is a finite, nonempty set of natural numbers, the max and min must be well-defined. The value $(\max(N) - \min(N))$ is an upper bound for the maximum displacement of π on N. Thus, π must achieve a maximum value $C \leq \max(N) - \min(N)$ for some value $m \in [\min(N), \max(N)]$, and the displacement is zero for all other values. Thus, the maximum displacement is C and $\pi \in \Theta(1)$.

4.3 Block cyclic natperms

Claim: If π is a bounded block cyclic natperm, then $\pi \in \Theta(1)$.

Proof: Let $(b_j)_{j=1}^{\infty}$ be the sequence of boundaries that define π. By the definition of bounded block cyclic natperm, there is an finite upper bound $C \ni |b_{j+1} - b_j| \leq C \; \forall j$. Since the natperm only remaps elements within a block, the maximum displacement can never exceed the maximum block size (which must be less than or equal to C).

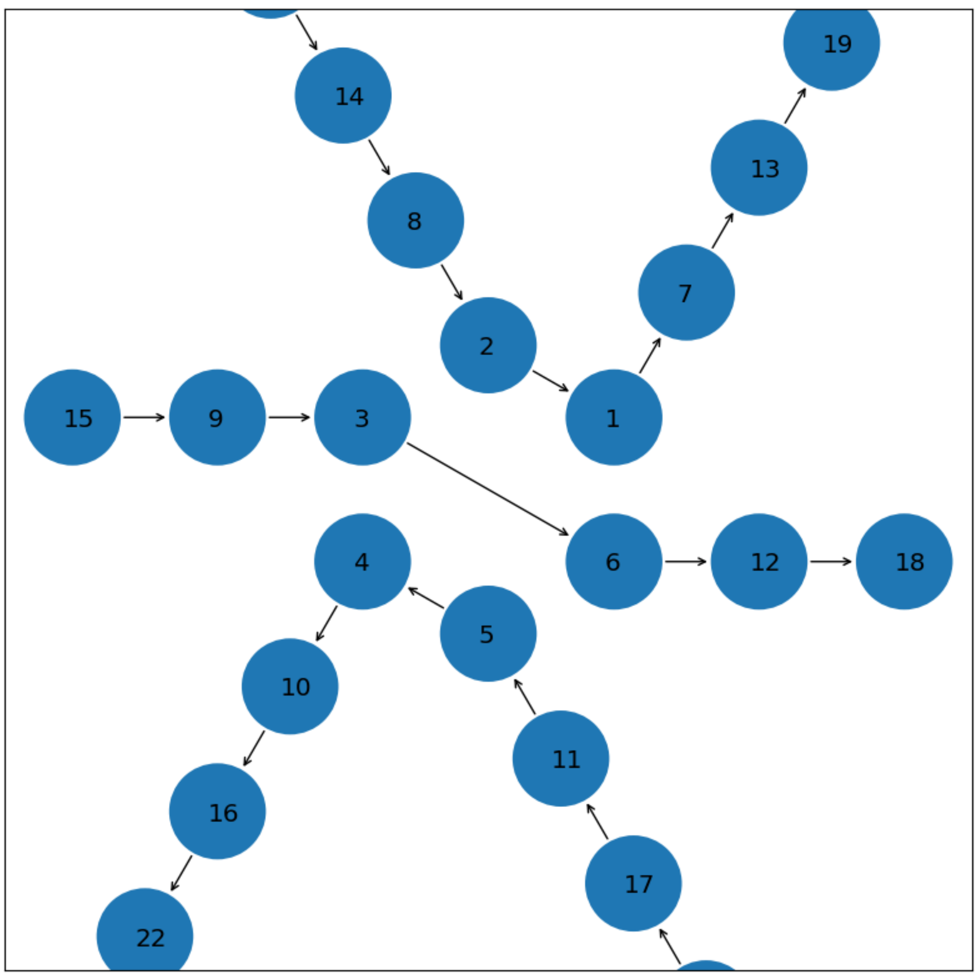

Figure 1: A shift-and-glue permutation.

Conjecture: If π is an unbounded block cyclic natperm, then $\pi \in \Theta(n)$ (or $\pi \in \Theta(1)$).

Rationale: Consider the example provided earlier, where the blocks are defined by boundary sequence $b_j = 2^j$ and the mapping on each block is the shift map (except gluing the top of the block to its bottom). The first two blocks are (1) and (2). For all higher blocks $j \geq 2$, the displacement is 1 within the block (the shift amount), and then $2^j - 2^{j-1} = 2^{j-1}$ at the glue point. Since 2^{j-1} is increasing in j (and greater than 1 for all $j > 1$), the maximum displacement is 2^{j-1} for values up to 2^j. Therefore, at the glue points, the ratio of displacement to value is $2^{j-1}/2^j = \frac{1}{2}$ (and elsewhere it tends to zero as j increases). This shows that one-half is the limit value for the ratio of maximum displacement to value.

This unbounded block cyclic natperm was constructed to have the greatest possible displacement within blocks (by gluing top to bottom). And therefore the blockwise displacement scales linearly with the size of the blocks, even though the blocks themselves grow exponentially in size. And since the block sizes accumulate exponentially, the resulting ratio of maximum displacement to input size is linear too. This article will not attempt to prove that this property holds for all possible choices of block sizes. Instead, this is left as a conjecture for future research.

Finally note that by varying the blockwise permutation, it is possible to construct a block cyclic natperm with unbounded blocks, but finite maximum displacement. Let $[b_j, b_{j+1})$ be an arbitrary block. Partition the elements of this block by their parity (evens and odds), and define a blockwise permutation which traverses the evens in ascending order and the odds in descending order, gluing to the closest element at the top and bottom as necessary. For example, the block interval $[2, 8)$ would have the blockwise cycle $(2, 4, 6, 7, 5, 3)$ and the block interval $[1, 9)$ would instead have the cycle $(1, 2, 4, 6, 8, 7, 5, 3)$. The displacement is 2 when traversing the interior of the block, and 1 at glue/switch points. Therefore, it is always possible to make a blockwise natperm have finite/constant maximum displacement. The force of the conjecture is in the upper bound: regardless of the block boundaries and the blockwise permutations, the maximum displacement cannot be supra-linear in the input.

4.4 Shift-and-glue natperms

Claim: If π is a shift-and-glue natperm, $\pi \in \Theta(1)$.

Proof: Let σ_k be the shift map that generates the strands of π. For identity strands, the displacement is 0, and for both ascending and descending strands, it is the constant value k. Recall that the glue points are in the interval $[1, k]$. The maximum possible displacement between glue points is therefore $|k - 1|$. Then k is an upper bound for the displacement for all input values.

Note that this proof is designed for the base example presented earlier. It is also possible to generate shift-and-glue permutation which have more complex behavior in the "center", and eventually settle into similar behavior as the basic example. The details of the proof would need to be adjusted to handle such cases, although the general structure would remain the same.

4.5 Summary

This section has defined displacement as the distance between an element and its image, and proposed that the limit behavior of maximum displacement (as input value grows) is a way to classify natperms. The maximum displacement is of course finite in finite natperms. Block cyclic natperms can be considered as an infinite product of finite permutations. If the block size is upper bounded, the maximum displacement is too; if not, it is conjectured that the maximum displacement scales linearly with the input (or less; but crucially, not supra-linearly). Shift-and-glue natperms can be considered as a finite product of permutations on countably nonfinite subsets of $\mathbb{N}$; here, too, the maximum distance is finite. These "simple" examples are constructed from familiar elements (finite blocks, finite permutations, the shift map, and some light gluing operations). This raises an interesting question: is it possible to construct a natperm which is essentially different from these simple examples? For example, can there be a natperm π which is not in $\Theta(n)$ or $\Theta(1)$? Can a natperm be a product of a mix of orbits which belong to different classes?

5 A natperm with nonlocal displacement

This section describes a natperm which is seemingly not $\Theta(1)$ or $\Theta(n)$.

5.1 Overview

The general approach will start with a set $\mathbb{B}$ that is homeomorphic to $\mathbb{N}$. This article will refer to an invertible mapping from $\mathbb{N}$ onto $\mathbb{B}$ as an **encoder**, and its inverse as the corresponding **decoder**. Let `fenc`, `fdec` be an encoder-decoder pair, and `genc`, `gdec` another, different encoder-decoder pair. Then `gdec`$\circ$`fenc` is a bijection from $\mathbb{N}$ to $\mathbb{N}$, in other words a natperm. This section will define the bijective space $\mathbb{B}$, an encoder-decoder pair, a different encoder-decoder pair; finally it will conclude with some analysis.

5.2 Canonical binary strings

Define the set of **canonical binary strings** as follows:

$$\mathbb{B} = \{\langle 1s\rangle \mid s \in \{0,1\}^*\} \tag{4}$$

where $*$ is the Kleene star operator (meaning any number of repetitions of any elements, including 0 repetitions). For clarity, strings will be delimited by angle brackets; thus $\langle\rangle$ denotes the empty string (which is not a canonical binary string), while $\langle 1\rangle$ denotes the string consisting only of the symbol 1.

5.3 Binary encoder/decoder

It is well known that a natural number can be represented as a sum of all powers of 2 up to some k, each multiplied by a coefficient that is either 0 or 1. For example:

$$13 = (1 \cdot 2^3) + (1 \cdot 2^2) + (0 \cdot 2^1) + (1 \cdot 2^0) \tag{5}$$

Define the binary encoder `benc` as mapping each natural number to the shortest such coefficient sequence (i.e. with no extraneous leading zeros), e.g. $\texttt{benc}(13) = \langle 1101 \rangle$.

The binary decoder `bdec` is simply the inverse of `benc`. Note that standard computational implementations of `benc` allow extraneous leading zeros. Items like $\langle 01101 \rangle$ and $\langle 001101 \rangle$ would otherwise map to the same value as $\langle 1101 \rangle$ does (namely, 13). But allowing this would prevent invertibility. The exclusion of leading zeros in $\mathbb{B}$, and the restriction of `benc` to the shortest binary coefficient sequence, is all so that `benc` and `bdec` can be inverses.

5.4 Primefac Peano encoder/decoder

This encoder will be defined by composing a number of simpler, invertible mappings (none of which are encoders themselves). Because each mapping in the encoder is invertible, it is straightforward the verify the encoder is 1-1. Finally it is demonstrated that the composed mapping is onto.

5.4.1 Prime factorization

It is well-known that every positive natural number can be factored into a product of primes. The factorization can be expressed in various ways, for example as a multiset:

$$28 = 7^1 \cdot 2^2 \rightarrow \begin{Bmatrix} 2: & 2, \\ 7: & 1 \end{Bmatrix} \tag{6}$$

Since the primes are enumerable and totally ordered by $<$, it is also possible to represent the prime factorization with a **multiplicity vector**. In a multiplicity vector, the prime values are implicitly represented by the vector index; the corresponding value indicates the multiplicity. For example, $28 = (7^1) \cdot (5^0) \cdot (3^0) \cdot (2^2)$, so the multiplicity vector is $(1, 0, 0, 2)$. In this article, multiplicity vectors will be written with the final element corresponding to the lowest prime (2); values which appear to the left correspond to increasingly larger primes. In this way, leading zeros on the multiplicity vector correspond to leading zeros in a binary string: they do not change the value, and should be suppressed. Thus, a well-defined (and invertible) mapping is obtained by defining $primefac : \mathbb{N} \rightarrow \mathbb{N}_0^*$ as mapping integers to the shortest multiplicity vector that includes all nonzero multiplicities (in other words, with no leading zeros).

5.4.2 Matchstick mapping

A primitive method for representing positive integers is with a sequence of strokes; each stroke looks like the numeral 1, and the number of strokes indicates the magnitude of the number. For example, $3 \to \langle 111 \rangle$. This paper will define the matchstick mapping $\mathit{stick} : \mathbb{N}_0 \to \mathbb{B}$ as mapping an integer n to a string consisting of n copies of the symbol 1. Note that $\mathit{stick}(0) = \langle\rangle$. The Peano axioms define the natural numbers in a way that is closely related to the matchstick mapping.

Some papers follow the convention that a scalar function applies element-wise, if its argument is a vector. This paper will instead explicitly indicate that the function is meant to apply element-wise, by defining $\overline{\mathit{stick}} : \mathbb{N}_0^* \to \mathbb{B}^*$ explicitly: $\overline{\mathit{stick}}((n_1, n_2, \ldots, n_k)) = (\mathit{stick}(n_1), \mathit{stick}(n_2), \ldots, \mathit{stick}(n_k))$.

5.4.3 String joining

It is common to compose a sequence of strings into a single string. For example, simple concatenation is often represented with the + operator, e.g. $\langle 1 \rangle + \langle 0 \rangle = \langle 10 \rangle$. Simple concatenation is not generally invertible. An alternative is to inserting a delimiter character at each join point. This paper will denote such a joining operation with the notation $\bigoplus_d$ symbol, where d indicates the delimiter character. In this paper, the delimiter will be 0. For example, $\bigoplus_0(\langle 1 \rangle, \langle\rangle, \langle\rangle, \langle 11 \rangle) = \langle 100011 \rangle$.

5.4.4 Putting it all together

The almost-correct definition of $\texttt{penc}(n) = \bigoplus_0 (\overline{\mathit{stick}} \circ \mathit{primefac})(n)$. This mapping is illustrated with the ongoing example of $n = 28$.

input					28				
primefac	(	1	,	0	,	0	,	2	)
stick		1						11	
delimiters			0		0		0		
concatenation	⟨				100011				⟩

There is an unfortunate problem with 1, however. The unit 1 has no prime factors, so its multiplicity vector is empty, so the concatenation is empty, and therefore does not belong to $\mathbb{B}$. This can be worked around by adding 1 to the number before applying $\mathit{primefac}$. In other words, the final definition must be $\texttt{penc} = \bigoplus_0 (\overline{\mathit{stick}} \circ \mathit{primefac} \circ \sigma_1)$.

5.4.5 Invertibility

It is straightforward to show that `penc` is 1-1:

- σ_1 is 1-1
- $\mathit{primefac}$ is 1-1

- $\overline{stick}$ is 1-1
- $\bigoplus_0$ is 1-1 (since *stick* outputs only the empty string and strings of 1s)

To demonstrate invertibility, it is sufficient to show that `penc` is onto, i.e. an arbitrary element $b \in \mathbb{B}$ must have a preimage. Observe that $b = \langle 1 s_k s_{k-1} \cdots s_0 \rangle$, where each s_i is either the delimiter 0 or the matchstick 1. If b contains no zeros, it must be a string consisting of k copies of 1, where $k \geq 1$. In this case, $stick^{-1}(b) = k$, and $2^k \geq 2$. Then $\sigma^{-1}(2^k) = 2^k - 1 \geq 1 \in \mathbb{N}$. Otherwise, the split-by-0 operation (which is the inverse of the join-with-0 operation) must produce a sequence of strings of the form $\langle 1* \rangle$. This follows because b can consist only of 1s and 0s, and each 0 is a delimiter. The matchstick mapping is invertible, e.g. $stick^{-1}(\langle\rangle) = 0$ and $stick^{-1}$ applied to a sequence of k 1s is simply k. Then $\overline{stick}^{-1}$ produces a sequence of integers, of which the leftmost must be nonzero, and all values must be nonnegative integers. In other words, $\overline{stick}^{-1}$ yields a multiplicity vector. The leftmost entry corresponds to a prime $p > 2$, and its multiplicity must be greater than or equal to 1. Therefore, the product (the inverse of $primefac$) is well-defined and greater than 1. As a result, σ^{-1} can be applied, yielding a value that is greater than or equal to 1. In other words, every value $b \in \mathbb{B}$ has a preimage under `penc`. Since `penc` is 1-1, this preimage is unique; in other words, $\texttt{pdec} = \texttt{penc}^{-1}$ is well-defined and `penc` is an encoder.

5.4.6 Encoder behavior demonstrated with selected examples

This subsection illustrates the encoding/decoding for the first several natural numbers. The Pythonic notation $(x,)$ is used to indicate a vector consisting of the single element x (as distinct from a scalar x which happens to be enclosed in parentheses for grouping clarity).

$$
\begin{array}{lll}
\texttt{penc}(0) & & (undefined) \\
\\
\texttt{penc}(1) & = & \bigoplus_0 (\overline{stick} \circ primefac \circ \sigma)(1) = \bigoplus_0 (\overline{stick} \circ primefac)(2) \\
 & = & \bigoplus_0 (\overline{stick}(\,(1,\,)\,)) = \bigoplus_0 (\,stick(1),\,) = \bigoplus_0 (\,\langle 1\rangle,\,) \\
 & = & \langle 1\rangle \\
\\
\texttt{penc}(2) & = & \bigoplus_0 (\overline{stick} \circ primefac \circ \sigma)(2) = \bigoplus_0 (\overline{stick} \circ primefac)(3) \\
 & = & \bigoplus_0 (\overline{stick}(\,(1,0)\,)) = \bigoplus_0 (\langle 1\rangle, \langle\rangle) \\
 & = & \langle 10\rangle \\
\\
\texttt{penc}(3) & = & \bigoplus_0 (\overline{stick} \circ primefac)(4) = \bigoplus_0 (\overline{stick}(\,(2,\,)\,)) \\
 & = & \langle 11\rangle \\
\\
\texttt{penc}(4) & = & \bigoplus_0 (\overline{stick} \circ primefac)(5) = \bigoplus_0 (\overline{stick}(\,(1,0,0)\,)) \\
 & = & \langle 100\rangle \\
\\
\texttt{penc}(5) & = & \bigoplus_0 (\overline{stick} \circ primefac)(6) = \bigoplus_0 (\overline{stick}(\,(1,1)\,)) \\
 & = & \langle 101\rangle \\
\\
\texttt{penc}(6) & = & \bigoplus_0 (\overline{stick} \circ primefac)(7) = \bigoplus_0 (\overline{stick}(\,(1,0,0,0)\,)) \\
 & = & \langle 1000\rangle \\
\\
\texttt{penc}(7) & = & \bigoplus_0 (\overline{stick}(\,(3,\,)\,)) = \langle 111\rangle
\end{array}
$$

This sample is enough to show that at small values, the lexicographic order of the natural numbers is strongly correlated with lexicographic order of their encodings. However, the final two values in the table above illustrate a pair where $n_1 < n_2$ but $\texttt{penc}(n_1) > \texttt{penc}(n_2)$, i.e. a pair where the encoder does not preserve lexicographic order. In fact, it is not hard to induce a recipe from this example for finding many other pairs where `penc` does not preserve lexicographic order. Observe that the final two values in the table above can be represented as $(n_1, n_2) = (7^1 - 1, 2^3 - 1)$. It is not difficult to show that $\texttt{penc}(2^k - 1) = k \cdot \langle 1\rangle$ (where multiplication has the standard meaning, of iterated string concatenation). Similarly, if p_j is the j^{th} prime, $\texttt{penc}(p_j^k - 1) = k \cdot \langle 1\rangle + (j-1) \cdot \langle 0\rangle$ (where addition has the standard meaning of string concatenation). By selecting a low exponent for the prime greater than 2, and a higher exponent for 2, it is easy to find additional examples where `penc` does not preserve lexicographic order.

5.5 Orbits of natperm

This section illustrates all orbits which pass through the interval [1, 100], terminating whenever the value exceeds 1e6.

The mixed occurrence of cycles and not obviously overlapping, not obviously finite orbits is pointed out.

It is highlighted that a few examples illustrate supra-exponentially increasing orbits.

6 Conclusion

Some words please.