Extending The Prouhet-Thue-Morse Sequence

Olivia Appleton-Crocker
TMW Center for Early Learning + Public Health
University of Chicago
Chicago, Illinois, United States
ORCID: 0009-0004-2296-7033

***Abstract*—In this paper, we discuss various ways to extend the Prouhet-Thue-Morse Sequence [1] when used as a fair-share sequence. Included are 21 definitions of the original sequence, 9 extensions to n players, for a total of 30 definitions. Also included are proofs of equality for all definitions, as well as an examination of several properties of the Prouhet-Thue-Morse Sequence and their presence in the Extended Prouhet-Thue-Morse Sequence . In the appendix are several complexity analyses for both time and space of each definition.**

***Index Terms*—Combinatorics, Generating Functions, Thue-Morse, Prouhet-Thue-Morse, Formal Languages, Number Theory, Periodicity, Aperiodicity, Rotation, Concatenation**

I. Introduction

This section is too fluffy. Probably needs to be rewritten entirely

The Prouhet-Thue-Morse Sequence is a fundamental object in combinatorics and theoretical computer science, widely studied for its remarkable properties and diverse applications. It is often presented as an infinite sequence starting with 0, as shown below:

$$T=\langle 0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0,0,1,1,0,\ldots\rangle$$

This sequence arises in various fields, including automata theory, formal languages, number theory, and signal processing, due to its connection to binary operations, periodicity, and complexity. Its structure has been examined in relation to notions of minimality and non-repetitiveness, making it a natural object of study in mathematical logic and computational theory.

Over the past two centuries, the Prouhet-Thue-Morse Sequence has garnered attention for its role in constructing infinite words with specific properties (such as non-repetition over large substrings) and for its use in the design of error-correcting codes, pseudorandom number generators, and tiling problems. More recently, its applications have extended to the analysis of dynamical systems, coding theory, and the study of complexity within algorithms.

This survey primarily aims to explore the various definitions of the Prouhet-Thue-Morse Sequence found in the literature, with an emphasis on identifying distinct formulations and categorizing them. Additionally, this work seeks to extend these definitions from the binary alphabet $\{0,1\}$ to larger alphabets $\{0,1,\ldots,n\text{–}1\}$, often referred to as "integer bases." This work aims to explore how properties such as non-repetition, periodicity, and complexity are preserved under these extensions. By systematically analyzing these extensions and their associated properties, this overview hopes to present a deeper understanding of the Prouhet-Thue-Morse Sequence's generalizations and their potential applications in broader contexts.

This paper is divided into 8 sections. In Section 1, we introduce the concepts built upon in this paper. In Section 2 we present each of the 21 definitions of the standard Prouhet-Thue-Morse Sequence as seen in the literature today, divided by category of definition. We first look at numeric methods, then operations on ordered collections of numbers. We next examine definitions that relate to other sequences of integers, followed by a pair of generating functions, a hypergeometric definition, one based on cellular automatons, and a derivation from Galois Duels.

In Section 3 we prove their equivalence with each other in the standard base-2 domain. In Section 4 we present 9 definitions that extend into larger domains. Most of these can only construct sequences with integer value elements, though a select few can be extended into even broader domains, such as rational bases. In Section 5 we prove that the extensions are equivalent to each other in the domain of positive integer bases. In Section 6 we examine the preservation (or failure) of properties for the standard Prouhet-Thue-Morse Sequence when extended to larger bases. Sections 7 and 8 are acknowledgments and the appendix respectively. The appendix contains honorable mention extensions and complexity analyses (both time and space) for the different definitions. These show which may be the most computationally efficient ways to calculate the Prouhet-Thue-Morse Sequence or Extended Prouhet-Thue-Morse Sequence , given your available resources.

II. The Original Sequence

A. Definition 1 of 21 — Parity of Hamming Weight

This definition appears in [1–4].

The Hamming Weight, as typically defined, is the digit sum of a binary number. In other words, it is a count of the high bits in a given number. A common way to generate the Prouhet-Thue-Morse Sequence is to take the parity[1] of the Hamming Weight[2] for each natural number. We can define that as follows:

[1]Whether a number is odd or even
[2]The count of 1s in the binary representation of a number

$$
\begin{aligned}
p(0) &= 0 \\
p(n) &= (n \mod 2) + p\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \\
T_{2,1}(n) &= p(n) \mod 2
\end{aligned}
\tag{1}
$$

The subscript indicates that we are using 2 players (writing in base 2) and that we are using the first definition laid out in this paper. Note that when we extend to n players, the T function will get a second parameter for the number of players, so it will look like $T_{n,d}(x, s)$, where s is the size of the player pool, and therefore the base we use to define the sequence.

B. Definition 2 of 21 — Powers of Negative One

This appears in [5].

Another way to implement the modularity of the previous definition is to use the appropriate root of unity. In this paper, we use $\omega_s = \exp\left(\frac{2i\pi}{s}\right)$ to represent the primitive root of unity in base s, such that $(\omega_s)^s = 1$. Because of this, extracting the power of $(\omega_s)^x$ will give you the same value as $x \mod s$.

There are two ways this can be approached. In the first we work with these output values directly, mapping $\{1, -1\} \to \{0, 1\}$:

$$
\begin{aligned}
T_{2,2a}(n) &= \frac{1 - (\omega_2)^{p(n)}}{2} \\
&= \frac{1 - (-1)^{p(n)}}{2}
\end{aligned}
\tag{2}
$$

C. Definition 3 of 21 — Root of Unity

And in the second we extract the exponent:

$$
\begin{aligned}
T_{2,3b}(n) &= \frac{\log\left(\omega_2^{p(n)}\right)}{\log\left(\omega_2\right)} \\
&= \frac{\log\left((-1)^{p(n)}\right)}{\log(-1)} \\
&= \frac{(p(n) \mod 2) \cdot \log(-1)}{\log(-1)} \\
&= \frac{(p(n) \mod 2) \cdot i\pi}{i\pi} \\
&= p(n) \mod 2
\end{aligned}
\tag{3}
$$

D. Definition 4 of 21 — Recursion

This definition appears in [1, 4, 6].

$$
\begin{aligned}
T_{2,4}(0) &= 0 \\
T_{2,4}(2n) &= T_{2,4}(n) \\
T_{2,4}(2n+1) &= 1 - T_{2,4}(n)
\end{aligned}
\tag{4}
$$

E. Definition 5 of 21 — Floor-Ceiling Difference

This definition appears in [1]. Some of the info that's in the proof of equivalence should be lifted up into here

$$
\begin{aligned}
b(n) &= \begin{cases} n & \text{if } n \le 1 \\ b\left(\left\lceil \frac{n}{2} \right\rceil\right) - b\left(\left\lfloor \frac{n}{2} \right\rfloor\right) & \text{otherwise} \end{cases} \\
T_{2,5}(n-1) &= \frac{1 - b(2n-1)}{2} \pmod 2
\end{aligned}
\tag{5}
$$

This seems very similar to the highest bit difference definition, and I think it may be what that was derived from

F. Definition 6 of 21 — Highest Bit Difference

This definition appears in [7].

The text below is from Wiki and needs to be entirely rewritten. I was able to derive the formula on my own from translating their code. This method leads to a fast method for computing the Prouhet-Thue-Morse Sequence: start with t0 = 0, and then, for each n, find the highest-order bit in the binary representation of n that is different from the same bit in the representation of n - 1. If this bit is at an even index, tn differs from tn - 1, and otherwise it is the same as tn - 1.

```
from itertools import count

def p2_d07():
    value = 1
    for n in count():
        # Assumes that (-1).bit_length() == 1
        x = (n ^ (n - 1)).bit_length() + 1
        if x & 1 == 0:
            # Bit index even, so toggle value
            value = 1 - value
        yield value
```

Should there be a +1 inside that call to log2?

$$
\begin{aligned}
T_{2,6}(0) &= 0 \\
T_{2,6}(n) &= \begin{matrix} \lfloor \log_2(n \oplus (n-1)) \rfloor \\ + T_{2,6}(n-1) \pm 1 \end{matrix} \pmod 2
\end{aligned}
\tag{6}
$$

G. Definition 7 of 21 — Hamming-Weight Complement

This definition appears in [5]. The Hamming Weight Complement (defined as $\overline{p}(n)$) is the count of 0s in the minimal binary representation of a number, and A054429 is the sequence defined by reversing the order of integers between 2^n and $2^{n+1} - 1$ (ex: $0, 1, 3, 2, 7, 6, 5, 4, \ldots$).

$$
\begin{aligned}
\text{A054429}(n) &= \begin{cases} 0 & \text{if } n = 0 \\ 3 \cdot 2^{\lfloor \log_2(n) \rfloor} - n - 1 & \text{otherwise} \end{cases} \\
\overline{p}(x) &= \lceil \log_2(x+1) \rceil - p(x) \\
T_{2,7}(n) &= 1 - \overline{p}(\text{A054429}(n)) \mod 2
\end{aligned}
\tag{7}
$$

H. Definition 8 of 21 — Invert and Extend

This definition appears in [1, 4, 8].

This definition is more natural to think about as extending a tuple that contains the sequence. We will give a recurrence relation below, but to build an intuition we will work in this framework first.

Let $t(n)$ be the first 2^n elements of the Prouhet-Thue-Morse Sequence. Given this, we can define[3]:

[3] Using $\|$ to mean concatenation, so $\langle 0 \rangle \| \langle 1 \rangle = \langle 0, 1 \rangle$. Likewise: $\Big\|_{i=a}^{b} f(i) = f(a) \| f(a+1) \| f(a+2) \| \cdots \| f(b)$

$$\begin{aligned}\text{inv}(\mathbf{x}) &= \begin{cases}0, & \text{if } x_i = 1\\ 1, & \text{if } x_i = 0\end{cases}\\ &\quad \text{for } \mathbf{x} = \langle x_0, x_1, \ldots, x_{(|\mathbf{x}|-1)}\rangle\\ t(0) &= \langle 0\rangle\\ t(n) &= t(n-1) \parallel \text{inv}(t(n-1))\end{aligned} \tag{8}$$

Given the above, we can define a recurrence relation that will give us individual elements. It will be less efficient to compute, but will allow proofs of equivalence to be easier.

Should the below have a +1 inside the log?

$$\begin{aligned}T_{2,8}(0) &= 0\\ T_{2,8}(n) &= T_{2,8}\left(n - 2^{\lfloor \log_2(n)\rfloor}\right) + 1 \pmod 2\end{aligned} \tag{9}$$

I. Definition 9 of 21 — Substitute and Flatten

This definition appears in [1, 2, 4, 6]. It is one of the most commonly used because of its simplicity. Define the morhpism μ on the alphabet $\{0, 1\}$ by $\mu(0) = 01, \mu(1) = 10$. T is then the unique fixed point of μ that begins with 0. In other words, $T = \lim_{n\to\infty} \mu^n(0)$.

$$\begin{aligned}s(n) &= \begin{cases}\langle 0, 1\rangle, & \text{if } n = 0\\ \langle 1, 0\rangle, & \text{if } n = 1\end{cases}\\ t(0) &= \langle 0\rangle\\ t(n) &= \Big\|_{i=0}^{2^{n-1}-1} s(t(n-1)_i)\\ T_{2,9}(n) &= t(\lceil \log_2(n+1)\rceil)_n\end{aligned} \tag{10}$$

So for example, calculating $T_{2,9}(3)$ would look like:

$$\begin{aligned}t(0) &= \langle 0\rangle\\ t(1) &= \Big\|_{i=0}^{0} s(t(0)_i) = \langle 0, 1\rangle\\ t(2) &= \Big\|_{i=0}^{1} s(t(1)_i) = \langle 0, 1, 1, 0\rangle\\ T_{2,9}(3) &= t(\lceil \log_2(3+1)\rceil)_3\\ &= t(2)_3\\ &= \langle 0, 1, 1, 0\rangle_3\\ &= 0\end{aligned} \tag{11}$$

J. Definition 10 of 21 — Recursive Rotation

Another way to phrase the above definition is as recursive rotation. To the best of our knowledge, this definition is original to this paper. If we decompose s, we can instead represent it as:

$$\begin{aligned}r(\mathbf{x}, i) &= \langle x_{0+i \bmod |\mathbf{x}|}, x_{1+i \bmod |\mathbf{x}|}, \cdots\rangle\\ &\quad \text{for } \mathbf{x} = \langle x_0, x_1, \ldots, x_{(|\mathbf{x}|-1)}\rangle\\ t(0) &= \langle 0\rangle\\ t(1) &= \langle 0, 1\rangle\\ t(n) &= \Big\|_{i=0}^{1} r\left(t(n-1), i \cdot 2^{n-2}\right)\\ T_{2,10}(n) &= t(\lceil \log_2(n+1)\rceil)_n\end{aligned} \tag{12}$$

K. Definition 11 of 21 — Odious Number Derivation

This definition appears in [1].

Another way to generate the Prouhet-Thue-Morse Sequence is to take the sequence of Odious Numbers [9] mod 2. Odious numbers are those with an odd number of 1s in their binary representation. Note that the player numbers in this derivation are swapped, so when generating this for testing and extension, we add 1 to the result. Some simple generating code [10] for this is as follows:

```
from itertools import count

def seq_p2_d09():
    for i in count():
        if i.bit_count() & 1:
            yield (i + 1) & 1
```

In mathematical terms, this can be translated to:

$$\begin{aligned}no(n) &= \begin{cases}n & \text{if } p(n) \bmod 2 = 1\\ no(n+1) & \text{otherwise}\end{cases}\\ o(n) &= \begin{cases}1 & \text{if } n = 0\\ no(o(n-1)+1) & \text{if } n > 0\end{cases}\\ T_{2,11}(n) &= o(n) + 1 \pmod 2\end{aligned} \tag{13}$$

Aren't Odious Numbers exactly the numbers where the parity of the hamming weight is 1? So doesn't that mean that the Thue-Morse Sequence selects which numbers are Odious? From cursory testing, it seems to. There's something to be had there.

A possible way to extend this would be to reinterpret this as where the digit sum is not n-even

A related definition on OEIS [1] is

$$\begin{aligned}T(n) + \text{Odious}(n-1) + 1 &= 2n \text{ for } n \geq 1\\ T(n) &= 2n - \text{Odious}(n-1) - 1\end{aligned}$$

L. Definition 12 of 21 — Evil Numbers Derivation 1

This definition appears in [1]. More text

The Evil Numbers [11] are those who have an even number of 1s in their binary representation. Note that this is the opposite of the Odious Numbers referenced above.

```
from itertools import count

def evil():
    for i in count():
        if i.bit_count() & 1 == 0:
            yield i

def p2_d10():
    for n, i in enumerate(evil()):
        yield (i - 2 * n) & 1
```

In mathematical terms, this can be translated to:

$$\begin{aligned} ne(n) &= \begin{cases} n & \text{if } p(n) \bmod 2 = 1 \\ ne(n+1) & \text{otherwise} \end{cases} \\ e(n) &= \begin{cases} 1 & \text{if } n = 0 \\ ne(e(n-1)+1) & \text{if } n > 0 \end{cases} \\ T_{2,12}(n) &= e(n) - 2n \pmod 2 \end{aligned} \tag{14}$$

M. Definition 13 of 21 — Evil Numbers Derivation 2

This definition ppears in [5]. More text

A second, more-efficient derivation from the Evil Numbers is as follows, where $ce()$ is the count of Evil Numbers less than n [12], and $p()$ is the function defined in Equation 1.

$$\begin{aligned} ce(n) &= \left\lfloor \frac{n+1}{2} \right\rfloor + p(n+1) \cdot (n+1 \bmod 2) \\ T_{2,13}(n) &= 1 - ce(n+1) + ce(n) \end{aligned} \tag{15}$$

N. Definition 14 of 21 — Odious & Evil Numbers Derivation

This definition appears in [5]. More text as to why. Also, figure out why

A second, more-efficient derivation from the Evil Numbers is as follows, where $ce()$ is the count of Evil Numbers less than n [12], and $p()$ is the function defined in Equation 1.

$$\begin{aligned} oe(n) &= \begin{cases} o\left(\left\lfloor \frac{n}{2} \right\rfloor\right) & \text{if } n \bmod 2 = 0 \\ e\left(\left\lfloor \frac{n}{2} \right\rfloor\right) & \text{if } n \bmod 2 = 1 \end{cases} \\ T_{2,14}(n) &= 1 - oe(n) \pmod 2 \end{aligned} \tag{16}$$

O. Definition 15 of 21 — Gould's Sequence Derivation

This definition appears in [1].

Gould's Sequence [13] are the number of odd entries in a given row of Pascal's Triangle. Note that of the numeric methods, this is the least computationally efficient definition in this paper (see Tables XXX & XXXI).

Why mod 3? Everything else is mod 2. This definition is unlikely to be extendable, and the obvious routes fail. I don't get why this works, and I think I need to. I *think* Gould(x) always returns $2k + \{0, 1\}$

$$\begin{aligned} T_{2,15}(n) &= \text{Gould}(n) - 1 \bmod 3 \\ &= \left(\sum_{k=0}^{n} \left(\binom{n}{k} \bmod 2 \right) \right) - 1 \bmod 3 \end{aligned} \tag{17}$$

P. Definition 16 of 21 — Derivation from Blue Code

This definition appears in [1].

In the OEIS, this sequence [14] is defined as the "binary coding of a polynomial over GF(2), substitute x+1 for x". There are a number of ways to generate it. One of the more computationally-accessible ones is:

$$\begin{aligned} \text{A001317}(n) &= \sum_{k=0}^{n} \left(\binom{n}{k} \bmod 2 \right) \cdot 2^k \\ f(n,i) &= \left\lfloor \frac{n}{2^i} \bmod 2 \right\rfloor_{\lceil \log_2(n) \rceil} \\ \text{A193231}(n) &= \bigoplus_{i=0} (\text{A001317}(i \cdot f(n,i)) \cdot f(n,i)) \\ T_{2,16}(n) &= \text{A193231}(n) \pmod 2 \end{aligned} \tag{18}$$

Translated into words, this function computes the value of Sierpiński's triangle for the index of each high bit, then takes the bitwise exclusive or[4] of all such resulting values. Note that since $f(n,i) = 0$ if and only if the ith bit of n is low, each low bit can be simplified out when calculating.

It seems to me that this might be extendable by using GF(n) instead of GF(2), though I don't know of a way to efficiently compute or prove such a result

Q. Definition 17 of 21 — Generating Function 1

This generating function[5] definition appears in [1, 3].

$$\begin{aligned} G(x) &= \mathcal{G}.\mathcal{F}.\ \frac{\frac{1}{1-x} - \prod_{k\geq 0}^{\infty}(1-x^{2^k})}{2} \\ &= \mathcal{G}.\mathcal{F}.\ \frac{\sum_{k\geq 0}^{\infty} x^k - \prod_{k\geq 0}^{\infty}(1-x^{2^k})}{2} \\ T_{2,17}(n) &= [x^n]G(x) \end{aligned} \tag{19}$$

The logic of this construction is fairly clear. The component that is $\frac{1}{1-x}$ will generate coefficients of all 1s. This is then subtracted by a sequence that counts the high bits of a given index, evaluating to -1 if the number of bits is even, and 1 if it is odd. This means that the only possible outputs of the top part are 0 or 2, so the divisor corrects for that.

This means that it is equivalent to $T_{2,1}$ (prove it).

In practice, to get the coefficients up to the nth term, one needs to only expand the product to $k_{\max} = \lfloor \log_2(n+1) \rfloor - 1$.

[4] Represented in this paper as $\oplus$.

$$x \oplus y = \sum_{i=0}^{\lfloor \log_2(\max(x,y)) \rfloor} 2^i \cdot \left(\left\lfloor \frac{x}{2^i} \right\rfloor + \left\lfloor \frac{y}{2^i} \right\rfloor \bmod 2 \right)$$

$$\bigoplus_{i=a}^{b} f(i) = f(a) \oplus f(a+1) \oplus f(a+2) \oplus \cdots \oplus f(b)$$

[5] A generating function is a polynomial such that the coefficient of each term is equal to the value in a sequence. $[x^n]G(f)$ indicates "the coefficient of x^n in the function $G(x)$." So $[x^2]\left(1 + 2x + 3x^2\right) = 3$

Note that this and the next definition are by far the least computationally efficient definitions in this paper (see Figure 2 and Tables XXXIV, XXXV, XXXVI, XXXVII).[6]

R. Definition 18 of 21 — Generating Function 2

This definition appears in [17, 18]. The coefficients of this generating function are given in OEIS Sequence A309303 [19].

$$\begin{aligned} G(x) &= \mathcal{G}.\mathcal{F}.\frac{\sqrt{x+1}-\sqrt{1-3x}}{2\cdot(x+1)^{\frac{3}{2}}} \\ T_{2,18}(n) &= \frac{1-(-1)^{[x^n]G(x)}}{2} \end{aligned} \tag{20}$$

S. Definition 19 of 21 — Hypergeometry

I'm a little unclear why this works, but it certainly seems to

This definition appears in [17, 18].

$$T_{2,19}(n) = \left(1+\frac{(-1)^n}{2}+\sqrt{\pi}\cdot(-3)^n\cdot\frac{{}_2F_1(\frac{3}{2},-n;\frac{3}{2}-n;\frac{-1}{3})}{4\cdot n!}\right) \mod 2 \tag{21}$$

where ${}_2F_1(a_1,a_2;b_1;z)$ is the generalized, regularized hypergeometric function given by:

$${}_2F_1(a_1,a_2;b_1;z) = \sum_{k=0}^{\infty}\frac{\frac{(a_1)_k(a_2)_k}{(b_1)_k}\cdot\frac{z^k}{k!}}{\Gamma(b_1)}, \tag{22}$$

and $(a)_k$ is the Pochhammer symbol defined as:

$$(a)_k = \left(\prod_{i=0}^{k-1}(a+i),\ \ (a)_0=1\right) \approx \frac{(a+k-1)!}{(a-1)!} \tag{23}$$

T. Definition 20 of 21 — Cellular Automaton

This definition appears in [17, 18].

Using Mathematica-like notation,

$$\begin{aligned} T_{2,20} = \lim_{k\to\infty} 1 - \text{Flatten}[\text{CellularAutomaton}[\\ &\{69540422,2,2\}, \\ &\{\{1\},0\}, \\ &2^k-1, \\ &\{\text{All},0\}]] \end{aligned} \tag{24}$$

Here, the **CellularAutomaton** function is used to generate the sequence. The first argument, $\{69540422,2,2\}$, represents the rule for the automaton. Rule $69540422 = 10000111000010111011111010_2$ is a specific binary rule for generating the sequence, while the numbers 2 and 2 correspond to the number of states and the neighborhood range, respectively. This means that each cell will have the following transformation table, when including the state of its left and right 2 neighbors:

[6]Our implementation [10] heavily utilizes [15, 16] for these calculations

Neighborhood	Output
$\langle 0,0,0,0,0\rangle$	0
$\langle 0,0,0,0,1\rangle$	1
$\langle 0,0,0,1,0\rangle$	0
$\langle 0,0,0,1,1\rangle$	1
$\langle 0,0,1,0,0\rangle$	1
$\langle 0,0,1,0,1\rangle$	1
$\langle 0,0,1,1,0\rangle$	1
$\langle 0,0,1,1,1\rangle$	0
$\langle 0,1,0,0,0\rangle$	1
$\langle 0,1,0,0,1\rangle$	1
$\langle 0,1,0,1,0\rangle$	1
$\langle 0,1,0,1,1\rangle$	0
$\langle 0,1,1,0,0\rangle$	1
$\langle 0,1,1,0,1\rangle$	0
$\langle 0,1,1,1,0\rangle$	0
$\langle 0,1,1,1,1\rangle$	0
$\langle 1,0,0,0,0\rangle$	0
$\langle 1,0,0,0,1\rangle$	1
$\langle 1,0,0,1,0\rangle$	1
$\langle 1,0,0,1,1\rangle$	1
$\langle 1,0,1,0,0\rangle$	0
$\langle 1,0,1,0,1\rangle$	0
$\langle 1,0,1,1,0\rangle$	0
$\langle 1,0,1,1,1\rangle$	0
$\langle 1,1,0,0,0\rangle$	1
$\langle 1,1,0,0,1\rangle$	0
$\langle 1,1,0,1,0\rangle$	0
$\langle 1,1,0,1,1\rangle$	0
$\langle 1,1,1,0,0\rangle$	0
$\langle 1,1,1,0,1\rangle$	0
$\langle 1,1,1,1,0\rangle$	0
$\langle 1,1,1,1,1\rangle$	0

TABLE I
ENUMERATED 5-CELL NEIGHBORHOOD CONFIGURATIONS AND THEIR CORRESPONDING RULE BITS.

The initial configuration is specified by $\{\{1\},0\}$, meaning the sequence starts with a single 1 surrounded on either side by a field of 0s. The value 2^{k-1} specifies the number of iterations to take. The function will output the initial state of the center cell, followed by its state at each iteration. Finally, the Flatten function ensures that the sequence is flattened into a one-dimensional list. Each element x in the list is then passed through $1-x$. As $k\to\infty$, this process generates an increasingly large sequence approaching the Prouhet-Thue-Morse Sequence.

U. Definition 21 of 21 — Equitable Galois Duels

Consider a Galois Duel, where two players fire upon each other with equally improbable odds of success. At limit, the Thue-Morse Sequence is the turn order that minimizes advantage between players, breaking any ties using lexographic order [20].

Consider the following system of equations, where q is an arbitrarily low probability:

$$
\begin{aligned}
a_{n+1} &= \begin{cases} -a_n & \text{if } f_n(q) \geq 0 \\ a_n & \text{otherwise} \end{cases} \\
f_n(q) &= a_n \cdot \left(\sum_{j=0}^{n} T_{2,21}(j) \cdot q^j \right) \\
T_{2,21}(n) &= \frac{1-(-1)^{a_n}}{2}
\end{aligned}
\tag{25}
$$

V. Summary

Of the 21 we discussed above:

- 17 were initially found on the OEIS [1, 5, 18, 21, 22] ($T_{2,1\ldots2}, T_{2,4\ldots5}, T_{2,9}, T_{2,11\ldots20}$)
- 1 was found in a textbook [7] ($T_{2,6}$)
- 1 was found in a preprint paper [20] ($T_{2,21}$)
- 2 are original to this paper ($T_{2,3}, T_{2,10}$)

- 10 utilize recursion ($T_{2,1\ldots4}, T_{2,6\ldots10}, T_{2,21}$)
- 6 reference other integer sequences ($T_{2,11\ldots16}$)
- 7 utilize floor-division ($T_{2,1\ldots3}, T_{2,5}, T_{2,13\ldots14}, T_{2,16}$)
- 3 use operations on strings, not integers ($T_{2,8\ldots10}$)
- 2 use combinatoric functions ($T_{2,15\ldots16}$)
- 2 utilizes a generating function ($T_{2,17\ldots18}$)
- 1 is hypergeometric ($T_{2,19}$)
- 1 is based on a cellular automaton ($T_{2,20}$)
- 0 have closed form solutions

III. Proving Equivalence Between Standard Definitions

A. Correlating Definition 1 and Definition 2

Proof.

Observation 1: For all integers n, $(-1)^n \in \{1, -1\}$

Observation 2: For $n = \{1, -1\}$, $\frac{1-n}{2} = \{0, 1\}$

Observation 3: For all $n = 2k$, $(-1)^n = 1$

Observation 4: For all $n = 2k+1$, $(-1)^n = -1$

Inference 1: $\frac{1-(-1)^n}{2} = n \mod 2$

Conclusion:

$$
\begin{aligned}
T_{2,2}(n) &= \frac{1-(-1)^{p(n)}}{2} \\
&= p(n) \mod 2 \\
&= T_{2,1}(n)
\end{aligned}
\tag{26}
$$

□

B. Correlating Definition 1 and Definition 4

Note that in Equation 4 where we define $T_{2,4}$, we are working in mod 2, where $+1$ and -1 are logically equivalent. We can therefore simplify its definition to be:

$$
\begin{aligned}
&T_{2,4}(0) = 0 \\
&T_{2,4}(n) = n + T_{2,4}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \pmod 2
\end{aligned}
\tag{27}
$$

This is identical to Equation 1, where we define $T_{2,1}$.

C. Correlating Definition 1 and Definition 8

Proof.

Observation 1: $0 \leq n < 2^k \implies t(k+1)_{2^k+n} \equiv t(k)_n + 1$ This is because at each step in the process, you are inverting and extending. Inversion is equivalent to $x + 1 \mod 2$, and each entry keeps a relative position by adding a power of 2.

Inference 1:

$$
\begin{aligned}
&T_{2,8}(0) = 0 \\
&T_{2,8}(n) = T_{2,3}(n - 2^{\lfloor \log_2(n) \rfloor}) + 1 \mod 2
\end{aligned}
\tag{28}
$$

Observation 2: By subtracting a power of 2, you reduce the Hamming Weight by 1

Inference 2: Since you are adding 1 for each time you reduce the Hamming Weight by 1, this means $T_{2,8}$ is recursively computing the parity of the Hamming Weight.

Observation 3: $T_{2,1} = p(n)$, and $p(n)$ computes the parity of the Hamming Weight of n.

Conclusion: $T_{2,1}(n) = T_{2,8}(n)$ □

D. Correlating Definition 2 and Definition 3

Proof.

Branch 1: $p(n)$ is even

Observation 1:

$$
\begin{aligned}
T_{2,2}(n) &= \frac{1-(-1)^{2k}}{2} \\
&= \frac{1-1}{2} = 0
\end{aligned}
\tag{29}
$$

Observation 2:

$$
\begin{aligned}
T_{2,3}(n) &= \frac{\log((-1)^{2k})}{\log(-1)} \\
&= \frac{\log(1)}{\log(-1)} \\
&= \frac{0}{\log(-1)} = 0
\end{aligned}
\tag{30}
$$

Branch 2: $p(n)$ is odd

Observation 3:

$$
\begin{aligned}
T_{2,2}(n) &= \frac{1-(-1)^{2k+1}}{2} \\
&= \frac{1-(-1)}{2} \\
&= \frac{2}{2} = 1
\end{aligned}
\tag{31}
$$

Observation 4:

$$
\begin{aligned}
T_{2,3}(n) &= \frac{\log((-1)^{2k+1})}{\log(-1)} \\
&= \frac{\log(-1)}{\log(-1)} = 1
\end{aligned}
\tag{32}
$$

Conclusion: $T_{2,2}(n) = T_{2,3}(n)$ □

E. Correlating Definition 2 and Definition 5

Proof.

Observation 1: If n is even:

$$\begin{aligned} b(n=2k) &= b(\lceil \tfrac{2k}{2} \rceil) - b(\lfloor \tfrac{2k}{2} \rfloor) \\ &= b(k) - b(k) \\ &= 0 \end{aligned} \tag{33}$$

Inference 1: Therefore,

$$\begin{aligned} b(n=2k+1) &= b\left(\left\lceil \frac{2k+1}{2} \right\rceil\right) - b\left(\left\lfloor \frac{2k+1}{2} \right\rfloor\right) \\ &= b(k+1) - b(k) \\ &= \begin{cases} b(k+1) & \text{if } k \text{ is even} \\ -b(k) & \text{if } k \text{ is odd} \end{cases} \end{aligned} \tag{34}$$

Hypothesis 1: $b(2k+1) = (-1)^{p(k)}$

Assumption 1: Assume $b(2k+1) \neq (-1)^{p(k)}$

Assumption 2:

$$\begin{aligned} T_{2,2}(n) = \tfrac{1-(-1)^{p(n)}}{2} = \tfrac{1-b(2n+1)}{2} &= T_{2,5}(n) \\ 1-(-1)^{p(n)} &= 1-b(2n+1) \\ (-1)^{p(n)} &= b(2n+1) \end{aligned} \tag{35}$$

Contradiction! The only way for these definitions to be equal is for $b(2n+1) = (-1)^{p(n)}$

I don't think this proof is 100% solid □

F. Correlating Definition 9 and Definition 10

Proof.

Observation 1: $r(\mathbf{x}, i) = r(\mathbf{x}, i + k \cdot |\mathbf{x}|)$

Inference 1: $r(\mathbf{x}, i) = r(\mathbf{x}, i \mod |\mathbf{x}|)$

Observation 2: $s(0) = \langle 0, 1 \rangle = r(s(0), 0)$

Observation 3: $s(1) = \langle 1, 0 \rangle = r(s(0), 1)$

Inference 2: $s(n) = r(s(0), n)$

This is where I run into trouble. I know I can keep going from here, but I'm not quite sure how.

Conclusion: $T_{2,9}(n) = T_{2,10}(n)$ □

IV. The Extensions

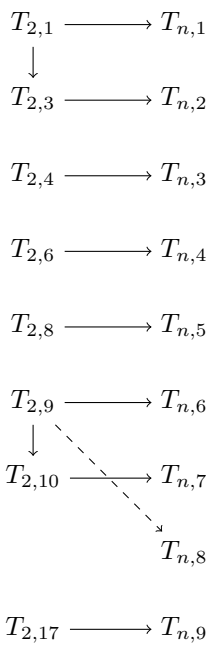


Fig. 1. Map of standard definitions to their extensions

While the Prouhet-Thue-Morse Sequence has a well-established binary form, its extension to larger alphabets introduces interpretive choices. In this work, we adopt a consistent methodology that extends all definitions equivalently. This approach ensures coherence across all extended definitions and preserves the underlying structure of the sequence. However, we acknowledge alternative constructions (such as the morphism $0 \to 012, 1 \to 02, 2 \to 1$ [4, 23–35]) which deviate from these extensions. While these may yield sequences of interest, they do not align with the equivalence criteria established in our framework.

A. Extension 1 of 9 — Modular Digit Sums

This definition appears in [21, 36–51].

To extend definition 1 from 2 to n players, we must first map our concept of parity to base n. We can do this by taking the parity equation defined above and replacing the 2s with n, for $n \in \mathbb{Z}_{\geq 2}$.

$$\begin{aligned} p_n(0) &= 0 \\ p_n(x) &= x + p_n\left(\left\lfloor \frac{x}{n} \right\rfloor\right) \pmod n \end{aligned} \tag{36}$$

Under this definition, you can construct the Prouhet-Thue-Morse Sequence using the following, starting at 0:

$$T_{n,1}(x, s) = p_s(x) \tag{37}$$

Note that this definition is trivially extensible to non-integer bases by redefining $p_n()$, though that is beyond the scope of this paper. This has been done in [21, 37]. It has also been extended to negative integer bases [38–40].

Some other works present a more generalized version, where

$$t_{b,m}(n) = p_b(n) \bmod m \tag{38}$$

This allows for increased flexibility, especially when using fractional bases. In this notation, for negative integer bases, $T_{n,1}(x,s) = p_s(x) \bmod |s|$

1) Proof of Equivalence with Original Definition 1:

Proof.

It is clear from visual inspection that p_2 is identical to our original definition of p.

$$\begin{aligned} p_2(x) &= p(x) \\ x + p_2\left(\left\lfloor \frac{x}{2} \right\rfloor\right) &= x + p\left(\left\lfloor \frac{x}{2} \right\rfloor\right) \\ x + \left\lfloor \frac{x}{2} \right\rfloor + p_2\left(\left\lfloor \frac{x}{2^2} \right\rfloor\right) &= x + \left\lfloor \frac{x}{2} \right\rfloor + p\left(\left\lfloor \frac{x}{2^2} \right\rfloor\right) \\ x + \left\lfloor \frac{x}{2} \right\rfloor + \left\lfloor \frac{x}{2^2} \right\rfloor + \ldots &= x + \left\lfloor \frac{x}{2} \right\rfloor + \left\lfloor \frac{x}{2^2} \right\rfloor + \ldots \end{aligned} \tag{39}$$

□

This definition is also trivially extended to negative integer bases.

B. Extended Definition 2 of 9 — Roots of Unity

$$T_{n,2}(x,s) = \frac{\log\left(\omega_s^{p_s(x)}\right)}{\log\left(\omega_s\right)} \tag{40}$$

1) Proof of Equivalence with Original Definition 3:

Proof.

Let's start by substituting s for 2:

$$\begin{aligned} T_{n,2}(x,2) &= \frac{\log\left(\omega_2^{p_2(x)}\right)}{\log(\omega_2)} \\ &= \frac{\log\left((-1)^{p_2(x)}\right)}{\log(-1)} \\ &= \frac{(p_2(x) \bmod 2) \cdot \log(-1)}{\log(-1)} \\ &= \frac{(p_2(x) \bmod 2) \cdot i\pi}{i\pi} \\ &= p_2(x) \bmod 2 \end{aligned} \tag{41}$$

This is identical to $T_{2,1}$, which we earlier proved is equivalent to $T_{2,2}$. □

C. Extended Definition 3 of 9 — Recursion

This definition is found in [52], which entered preprint around the time this paper was being drafted.

$$\begin{aligned} T_{n,6}(0,s) &= 0 \\ T_{n,6}(s \cdot x, s) &= T_{n,6}(x,s) \\ T_{n,6}(s \cdot x + k, s) &= k + T_{n,6}(x,s) \pmod s \end{aligned} \tag{42}$$

1) Proof of Equivalence with Original Definition 4:

Proof.

Let's begin by substituting s for 2:

$$\begin{aligned} T_{n,3}(0,2) &= 0 \\ T_{n,3}(2 \cdot x, 2) &= T_{n,3}(x,2) \\ T_{n,3}(2 \cdot x + k, 2) &= k + T_{n,3}(x,2) \pmod 2 \end{aligned} \tag{43}$$

Note that that only values for k that fit in this definition are 0 and 1. This means we can further simplify to:

$$T_{n,3}(2 \cdot x + 1, 2) = 1 + T_{n,3}(x,2) \pmod 2 \tag{44}$$

This is very similar to the definition found in equation 4, except that one is adding and the other subtracting. Fortunately, we know that the only values that $T_{2,4}$ will return are 0 and 1, which means that these operations will be completely equivalent.

$$\begin{aligned} 1 - 0 \bmod 2 &= 1 + 0 \bmod 2 \\ 1 - 1 \bmod 2 &= 1 + 1 \bmod 2 \end{aligned}$$

□

D. Extended Definition 4 of 9 — Highest Digit Difference

$$\begin{aligned} \text{XOR}_n(a,b) &= \sum_{i=0}^{\lceil \log_n(\max(a,b)+1) \rceil} n^i \left(\left\lfloor \frac{a}{n^i} \right\rfloor - \left\lfloor \frac{b}{n^i} \right\rfloor \bmod n \right) \\ T_{n,4}(0,s) &= 0 \\ T_{n,4}(x,s) &= \begin{matrix} \lfloor \log_s(\text{XOR}_s(x, x-1)) \rfloor \\ + T_{n,4}(x-1,s) + 1 \end{matrix} \pmod s \end{aligned} \tag{45}$$

Substitute n for 2, then simplify, plus a bit

1) Proof of Equivalence with Original Definition 6:

E. Extended Definition 5 of 9 — Increment and Extend

[53] gives a good example on how to possibly adapt inversion to incrementing

In the original version of this definition, we inverted the elements. In base 2, this is the same thing as adding 1 (mod 2). Given that, let $t(x,n)$ be the first n^x elements of the Extended Prouhet-Thue-Morse Sequence , for $n \in \mathbb{Z}_{\geq 2}$.

$$\text{inc}(\mathbf{x}, n) = \begin{matrix} x_i + 1 \pmod n \\ \text{for } \mathbf{x} = (x_0, x_1, \ldots, x_{(|\mathbf{x}|-1)}) \end{matrix} \tag{46}$$

$$\begin{aligned} t(0,n) &= \langle 0 \rangle \\ t(1,n) &= \langle 0, 1, \ldots, n-1 \rangle \\ t(x,n) &= t(x-1,n) \cdot \text{inc}(t(x-1,n), n) \end{aligned} \tag{47}$$

Given the above, we can define a recurrence relation that will give us individual elements. It will be less efficient to compute, but will allow proofs of equivalence to be easier.

$$\begin{aligned} T_{n,5}(0,s) &= 0 \\ T_{n,5}(x,s) &= T_{n,5}\left(x - s^{\lfloor \log_s(x) \rfloor}, s\right) + 1 \pmod s \end{aligned} \tag{48}$$

1) Proof of Equivalence with Original Definition 8:

F. Extended Definition 6 of 9 — Substitute and Flatten

This definition appears in [44, 45, 54, 55]

There's a bit of a leap here, since we have to explain why the rotation is equivalent to the binary choice presented in the original. There also might be a better syntax to define the rotation, perhaps using the format used in inv and inc.

$$\begin{aligned} b(s) &= \langle 0, 1, \cdots, s-2, s-1 \rangle \\ r(\mathbf{x}, i) &= \begin{array}{l} \langle x_{0+i \bmod |\mathbf{x}|}, x_{1+i \bmod |\mathbf{x}|}, \ldots \rangle \\ \text{for } \mathbf{x} = \langle x_0, x_1, \ldots, x_{(|\mathbf{x}|-1)} \rangle \end{array} \\ s(x, s) &= r(b(s), x) \\ t(0) &= \langle 0 \rangle \\ t(x, s) &= \prod_{i=0}^{2^{x-1}-1} s(t(x-1)_i, s) \\ T_{n,6}(x, s) &= t(\lceil \log_s(x+1) \rceil, s)_x \end{aligned} \tag{49}$$

1) Proof of Equivalence with Original Definition 9:

G. Extended Definition 7 of 9 — Recursive Rotation

$$\begin{aligned} r(\mathbf{x}, i) &= \begin{array}{l} \langle x_{0+i \bmod |\mathbf{x}|}, x_{1+i \bmod |\mathbf{x}|}, \ldots \rangle \\ \text{for } \mathbf{x} = \langle x_0, x_1, \ldots, x_{(|\mathbf{x}|-1)} \rangle \end{array} \\ t(0, s) &= \langle 0 \rangle \\ t(1, s) &= \langle 0, 1, \ldots, s-1 \rangle \\ t(x, s) &= \prod_{i=0}^{s-1} r\left(t(x-1, s), i \cdot s^{x-2}\right) \\ T_{n,7}(x, s) &= t(\lceil \log_s(x+1) \rceil, s)_x \end{aligned} \tag{50}$$

1) Proof of Equivalence with Original Definition 10:

H. Extended Definition 8 of 9 — Latin Square Constructions

This definition appears in [8, 56]

Let $L(n)$ be the reduced-form Latin Square with a first row of $\langle 0, 1, \ldots, n-1 \rangle$, and where each row progresses from one entry to the next as $L(n)_{a,x+1} \equiv L(n)_{a,x} + 1 \pmod n$. For each iteration t_n, substitute each entry x for the string $L(n)_{x,*}$.

Needs more explanation, largely copying from std def 8

$$L(N) = \begin{pmatrix} 0 & 1 & 2 & \ldots & N\text{–}1 \\ 1 & 2 & \ddots & N\text{–}1 & 0 \\ 2 & \ddots & N\text{–}1 & 0 & 1 \\ \vdots & N\text{–}1 & 0 & 1 & \ddots \\ N\text{–}1 & 0 & 1 & \ddots & \ddots \end{pmatrix} \tag{51}$$

1) Proof of Equivalence with Original Definition 9:

I. Extended Definition 9 of 9 — Generating Functions

$$\begin{aligned} G_s(x) &= \mathcal{G}.\mathcal{F}. \prod_{k \geq 0}^{\infty} \sum_{i=0}^{s-1} \omega_s^i \cdot x^{i \cdot s^k} \\ \omega_s^{T_{n,9}(j,s)} &= [x^j] G_s(x) \\ T_{n,9}(j, s) &= \frac{\log\left([x^j] G_s(x)\right)}{\log(\omega_s)} \\ &= \frac{\log\left([x^j] G_x(x)\right) \cdot s}{2i\pi} \end{aligned} \tag{52}$$

A similar definition to the below is found in [57] for $x \in \mathbf{Q}((x^{-1}))$. While that definition is equivalent to T_2, it does not match the values for the other generalizations in this paper. For a specific example:

$$\begin{aligned} T_3 &= \langle 0, 1, 2, 1, 2, 0, 2, 0, 1, \ldots \rangle \\ T_3' &= \langle 0, 2, 0, 2, 1, 0, 0, 0, 0, \ldots \rangle \pmod 3 \end{aligned} \tag{53}$$

Above needs checking. I am only 80% confident in my analysis here.

1) Proof of Equivalence with Original Definition 17:

Proof.

Observation 1: to start, let us rephrase definition 17 slightly

$$\begin{aligned} T_{2,17}(n) &= [x^n] G(x) \\ &= [x^n] \left(\frac{\sum_{k \geq 0}^{\infty} x^k - \prod_{k \geq 0}^{\infty} (1 - x^{2^k})}{2} \right) \\ &= \frac{[x^n] \left(\sum_{k \geq 0}^{\infty} x^k - \prod_{k \geq 0}^{\infty} (1 - x^{2^k}) \right)}{2} \\ &= \frac{1 - [x^n] \prod_{k \geq 0}^{\infty} (1 - x^{2^k})}{2} \end{aligned} \tag{54}$$

Observation 2: the aparatus around the infinite product exists entirely to translate $\{1, -1\} \to \{0, 1\}$. Another way to do that is to take the complex log of this output: for $x = \{1, -1\} : \frac{\log(x)}{\log(-1)} = \{0, 1\}$

Inference 1:

$$\frac{1 - [x^n] \prod_{k \geq 0}^{\infty} (1 - x^{2^k})}{2} = \frac{\log\left([x^n] \prod_{k \geq 0}^{\infty} (1 - x^{2^k}) \right)}{\log(-1)} \tag{55}$$

Observation 3: $\omega_2 = -1$

Observation 4: If we take $T_{n,9}(x,s)$ for $s=2$, we get

$$\begin{aligned} G_2(x) &= \mathcal{G}.\mathcal{F}.\ \prod_{k\geq 0}^{\infty}\sum_{i=0}^{2-1}\omega_2^i\cdot x^{i\cdot 2^k} \\ &= \mathcal{G}.\mathcal{F}.\ \prod_{k\geq 0}^{\infty}\omega_2^0\cdot x^{0\cdot 2^k}+\omega_2^i\cdot x^{i\cdot 2^k} \\ &= \mathcal{G}.\mathcal{F}.\ \prod_{k\geq 0}^{\infty}1+(-1)\cdot x^{2^k} \\ &= \mathcal{G}.\mathcal{F}.\ \prod_{k\geq 0}^{\infty}1-x^{2^k} \end{aligned} \tag{56}$$

This is identical to the product found in $T_{2,17}$.

Conclusion: $T_{n,9}(x,s)=T_{2,17}(x)$ □

J. Summary

Of the 9 we discussed above:

- 1 was found on the OEIS ($T_{n,1}$)
- 1 was derived concurrently with another paper ($T_{n,3}$)
- 2 were found in another paper ($T_{n,6}, T_{n,8}$)
- 5 are original to this paper ($T_{n,2}, T_{n,4\ldots5}, T_{n,7}, T_{n,9}$)

- 8 utilize recursion ($T_{n,1\ldots8}$)
- 4 utilize floor-division ($T_{n,1\ldots4}$)
- 4 use operations on strings, not integers ($T_{2,5\ldots8}$)
- 0 have closed form solutions

V. Proving Equivalence Between Extended Definitions

A. Correlating Definition 1 and Definition 2

Proof.

Observation 1: For all integers n, $\omega_s^n=\omega_s^{n \bmod s}$

Inference 1: $\dfrac{\log(\omega_s^n)}{\log(\omega_s)}=n \bmod s$

Conclusion:

$$\begin{aligned} T_{n,2}(x,s) &= \frac{\log(\omega_s^{p_s(x)})}{\log(\omega_s)} \\ &= \frac{(p_s(x) \bmod s)\cdot\log(\omega_s)}{\log(\omega_s)} \\ &= \frac{(p_s(x) \bmod s)\cdot 2i\pi s^{-1}}{2i\pi s^{-1}} \\ &= p_s(x) \bmod s \\ &= T_{n,1}(x,s) \end{aligned} \tag{57}$$

□

B. Correlating Definition 4 and Definition 8

Proof.

Observation 1: For any given row of $L(N)$, it will start with the index of the row

Observation 2: For any given row of $L(N)$, it will end with 1 less than the index of the row $\pmod N$

Inference 1: $L(N)_{x,*}=r(b(N),x)$

Observation 3: $T_{n,4}$ is defined as substituting $r(b(N),x)$ for each element x in the previous iteration

Conclusion: $T_{n,4}=T_{n,8}$ □

C. Summary

VI. Proving Persistence (Or Lack Thereof) of Original Properties

A. Use as a Fair-Share Sequence

Goal: show that for a variety of value functions, greedy algorithms given this turn order will always minimize inequality. They should at least do so more than the standard turn order. This is going to look like setting up an equation to show

$$\begin{aligned} eq(x,y) &= \begin{cases} 1 & \text{if } x=y \\ 0 & \text{if } x\neq y \end{cases} \\ f(v,p,s) &= \lim_{n\to\infty}\sum_{i=0}^{n} v(i)\cdot eq(T_n(i,s),p) \\ \forall p &: p\in\{1,\ldots,s-1\} \\ f(v,0,s)+\epsilon &< f(v,p,s) < f(v,0,s)-\epsilon \end{aligned} \tag{58}$$

This is for $v(i)$ being a value function and ϵ being an arbitrarily small number. $f()$ is therefore the sum of total value that they will be receiving. For example, one could model a board game as $v(i)=\frac{1}{2^i}$, where $v()$ models the amount each turn contributes to your probability of victory. Note that this may be very hard to show for versions of $v()$ which shrink too quickly, such as $v(i)=\frac{1}{i!}$, so for those cases we must show that it's better than the standard turn order

$v(0)$ should always return the highest value, and $v(n)$ the lowest value, where $n+1$ is the number of items

1) On the value function of 1: It is well known [53] for the Standard Prouhet-Thue-Morse Sequence that

$$\lim_{n\to\infty}\sum_{i=0}^{n}\frac{T_2(i)}{n+1}=\frac{1}{2} \tag{59}$$

Does this generalize to: ?

$$\lim_{n\to\infty}\sum_{i=0}^{n}\frac{T_n(i,s)}{n+1}=\sum_{i=0}^{s-1}\frac{i}{s}=\frac{n-1}{2} \tag{60}$$

and

$$\begin{aligned} &\forall x \mid 0\leq x<s \text{ and } x\in\mathbf{Z} \\ &\lim_{n\to\infty}\sum_{i=0}^{n}\frac{eq\left(T_n(i,s),x\right)}{n+1}=\frac{1}{s} \end{aligned} \tag{61}$$

2) Galois Duels & Related Decreasing Functions: The OEIS Sequence A287150 [20, 58] gives an extension of $T_{2,21}$ for 3 players. It should be noted that it is different from the extensions in this paper, meaning that this property is not preserved.

$$\begin{aligned} T_3 &= \langle 0,1,2,1,2,0,2,0,1,\ldots\rangle \\ A287150 &= \langle 0,1,2,2,1,0,2,1,0,\ldots\rangle \end{aligned} \tag{62}$$

3) Uniform Distributions: Let us have a value function that goes over a uniform distribution:

$$v_u(x) = a + (b-a)(1 - \frac{x}{n}) \tag{63}$$

Where

- a is the lower bound of the distribution
- b is the upper bound of the distribution, and
- n is the total number of items

Conjecture 1. *For $n \geq 2$ players and n^k items, greedily distributing items in the order T_n will minimize the difference between players' received values, when values are distributed according to $v_u(x)$.*

4) Normal Distributions: Let us have a value function that goes over a normal distribution:

$$v_n(x) = \mu - \sigma \cdot \Phi^{-1}\left(\frac{x}{n}\right) \tag{64}$$

Where

- μ is the mean of the distribution
- σ is the standard deviation, and
- Φ^{-1} is inverse CDF of the normal distribution

Conjecture 2. *For $n \geq 2$ players and n^k items, greedily distributing items in the order T_n will minimize the difference between players' received values, when values are distributed according to $v_n(x)$.*

5) Exponential Distributions: Let us have a value function that goes over an exponential distribution:

$$v_e(x) = -\frac{1}{\lambda} \cdot \ln\left(1 - \frac{x}{n}\right) \tag{65}$$

Where λ is the rate parameter of the exponential distribution

Conjecture 3. *For $n \geq 2$ players and n^k items, greedily distributing items in the order T_n will minimize the difference between players' received values, when values are distributed according to $v_e(x)$.*

6) Discrete Value Distributions: It is trivial to show that for non-continuous value functions, neither the Standard nor Extended Prouhet-Thue-Morse Sequence distributed fairly. For example, consider a set of items with the value $\{2^5, 2^3, 2^3, 1, 1, 1\}$. When split among two players, they will end up with a total value of $\langle 34, 17\rangle$. A more equitable distribution is achieved by giving player 0 the first item, and the remainder to player 1, resulting in $\langle 32, 19\rangle$. Similar observations can be made for larger numbers of players. This is left to the reader.

B. Palindrome

In base 2:

$$\forall x, y : (x > 1) \wedge (0 \leq y < 2^x), \;\; t_2(x)_y = t_2(x)_{2^x - y - 1} \tag{66}$$

(Find a citation for this)

Proof.

Assumption 1: $n > 2$
Assumption 2: $t_n(x)$ is palindromic, for $x > 1$
Observation 1:

$$t_n(x)_{n^x-1} = p_n(n^x - 1) \equiv (n-1)x \pmod n$$

Inference 1: By Assumption 2, $t_n(x)_0 = 0 = t_n(x)_{n^x-1}$
Observation 2:

$$t_n(x)_{n^x-2} = p_n(n^x - 2) \equiv (n-1)(x-1) - 2 \pmod n$$

Inference 2: By Assumption 2, $t_n(x)_1 = 1 = t_n(x)_{n^x-2}$
Inference 3: By Inferences 1 & 2,

$$\begin{aligned} t_n(x)_{n^x-1} &\equiv t_n(x)_{n^x-2} - 1 \\ (n-1)x &\equiv (n-1)(x-1) - 2 - 1 \\ &\equiv (n-1)(x-1) + (n-1) - 2 \\ &\equiv (n-1)x - 2 \\ 0 &\equiv -2 \end{aligned} \tag{67}$$

Contradiction! While $n > 2$, these terms cannot be equal. This means that if the first and last terms match, the second and penultimate will not, and vice versa. □

C. Uniform Recurrence

The Prouhet-Thue-Morse Sequence is a uniformly recurrent word: given any finite string X in the sequence, there is some length nX (often much longer than the length of X) such that X appears in every block of length nX. Tackle this by using $T_{n,6}$

D. Deriving Square Free Sequences

Given a word $X : |X| > 1, X \in T_n$, $(X \parallel X) \notin T_n$. Square Free implies Overlap Free. The original Prouhet-Thue-Morse Sequence is *not* square-free, as it is not possible to construct a square-free sequence with an alphabet smaller than 3 (cite to Thue himself). However it can produce a square-free sequence easily by taking the difference of subsequent terms (A029883).

Let T_{dn} be the sequence generated by taking the difference of terms in the sequence T_n, such that $T_{d2} = \langle -1, 0, 1, -1, \ldots\rangle$. The resulting sequence is square-free.

Theorem 1. *This method does not generalize to $n \in \mathbb{Z}_{\geq 2}$*

Proof.

Proof by counterexample:

- $T_{d3}(19 \ldots 25) = \langle -1, 1, -1, -1, 1, -1\rangle$
- $T_{d4}(37 \ldots 45) = \langle -1, -1, 2, -1, -1, -1, 2, -1\rangle$
- $\forall n : n > 4$, $T_{dn}(0 \ldots 3) = \langle -1, -1, -1, -1\rangle$

□

Another method to make a square-free sequence from T_2 is to construct a different alphabet [59] using pairs of terms. This generates a new sequence $T_{p2} = \langle 01, 11, 10, 01, 10, 00, \ldots\rangle$

Conjecture 4. *This method generalizes to T_n by iterating pairwise, making T_{pn}, where $T_{pn}(x) = \langle T_n(x), T_n(x+1)\rangle$.*

Proof. Not sure how to do this rigorously, but computer testing hasn't found a counterexample □

Conjecture 5. *This method generalizes to T_n by iterating over a sliding window of size n, making $T_{n\times n}$, where $T_{n\times n}(x) = \langle T_n(x), T_n(x+1), \ldots, T_n(x+n-1)\rangle$.*

Proof. Not sure how to do this rigorously, but computer testing hasn't found a counterexample □

E. Overlap Free

Extensions of the Prouhet-Thue-Morse Sequence that comply with $T_{n,8}$ are overlap-free [56]. Is that true? I thought they used invertible matrices in this paper, and I don't think the general solution in $T_{n,8}$ are invertible. Needs followup.

Given a word $X : |X| > 1, X \in T_n, (X \parallel X \parallel X_0) \notin T_n$. For T_2, this is shown in [4]. Note additionally that this apparently is equivalent to 7/3-power-free [60]. Overlap Free implies Cube Free

F. Cube Free

Because extensions of the Prouhet-Thue-Morse Sequence that comply with $T_{n,8}$ are overlap-free [56], they are also cube-free.

Given a word $X : |X| > 1, X \in T_n, (X \parallel X \parallel X) \notin T_n$.

G. Aperiodicity

Not totally sure how this should go, but I think a good approximation would be to show that the distance between the nearest two appearances of a substring grows to infinity much faster than the length of those substrings grows. For words of size greater than n, this seems feasible. For words in size 2 thru n, I think I need to show that the distance between repetitions is aperiodic. That *should* be equivalent to distance between repetitions of integers, since one of the definitions expands digits to words

H. Generalization of Prouhet-Terry-Escott Problem

The standard Prouhet-Terry-Escott problem [61, 62] is

> Given $m > 1$ find $r > m$ and disjoint sets of distinct integers $\{a_1 \ldots a_r\}$ and $\{b_1 \ldots b_r\}$ such that:
>
> $$\sum_{i=1}^{r} a_i^k = \sum_{i=1}^{r} b_i^k \tag{68}$$
>
> holds for every $k = 1 \ldots m$

Does this extension allow us to generalize to

> Given $m > 1$ find $r > m$ and n disjoint sets of distinct integers $\{s_{1,1} \ldots s_{1,r}\}$, $\{s_{2,1} \ldots s_{2,r}\}$, $\ldots$, $\{s_{n,1} \ldots s_{n,r}\}$ such that:
>
> $$\forall x : 2 \le x \le n \left(\sum_{i=1}^{r} s_{1,i}^k = \sum_{i=1}^{r} s_{x,i}^k \right) \tag{69}$$
>
> holds for every $k = 1 \ldots m$

I *think* [8] says yes, but I'm not certain I understand it yet

I. Maresh Multifractions

It is a well known result [3] that the infinite nested fraction defined by:

$$\frac{1}{2} \to \frac{\left(\frac{1}{2}\right)}{\left(\frac{3}{4}\right)} \to \frac{\left(\frac{\left(\frac{1}{2}\right)}{\left(\frac{3}{4}\right)}\right)}{\left(\frac{\left(\frac{5}{6}\right)}{\left(\frac{7}{8}\right)}\right)} \to \cdots \to \prod_{k=1}^{\infty} k^{(-1)^{T_2(k-1)}} \tag{70}$$

and that this converges to $\frac{1}{\sqrt{2}}$. Can we generalize fractions in a way that this result holds for T_n?

Let a Maresh multifraction [51] be

$$M_n(a_0, \ldots, a_{n-1}) = \prod_{k=0}^{n-1} a_k^{e^{2ki\pi/n}} \tag{71}$$

Note that in the case of $n = 2$, this reduces to a standard fraction: $M_2(a, b) = \frac{a}{b}$. For all other cases, it rotates the input values such that they are equidistant around the unit circle, then takes their product.

Conjecture 6. *Suppose for some multifraction of size n, we begin chunking the counting numbers into groups of n. Each of these groups then gets put into a multifraction, and these multifractions subsequently get put into a multifraction, and so on. We will call this sequence F_n*

$$\begin{aligned} F_n &= \lim_{x\to\infty} F_n(x) \\ F_n(1) &= M_n(1, \ldots, n-1) \\ F_n(2) &= M_n \begin{pmatrix} M_n(1, \ldots, n), \\ M_n(n+1, \ldots, 2n), \\ \ldots, \\ M_n((n-1)n+1, \ldots, n^2) \end{pmatrix} \\ \forall n : n \ge 2, |F_n| &= \left| \sum_{k=0}^{\infty} (k+1)(\omega_n^{T_n(k)}) \right| = \frac{1}{\sqrt{n}} \end{aligned} \tag{72}$$

J. Fractal Turtle Geometry

T_2 generates the von Koch snowflake. Do other integer bases also generate fractals in a way that can be generalized? [51, 52]

VII. Acknowledgment

We thank Dan Rowe for helping with the initial work on this paper, as well as Randy Appleton and Lydia Crocker who helped greatly with proofreading and editing.

VIII. Future Work

While this paper is intended to be as comprehensive as possible, there are still many things left to analyze. Fill in with anything I didn't get to from notes

A. Open Conjectures

The following conjectures were not able to be definitively proven or disproven in this paper. They have been computationally tested to the best our resources can handle.

- **Conjecture 1** — T_n distributes values equitably when values are determined by $v_u(x)$
- **Conjecture 2** — T_n distributes values equitably when values are determined by $v_n(x)$
- **Conjecture 3** — T_n distributes values equitably when values are determined by $v_e(x)$
- **Conjecture 4** — T_{pn} is square free
- **Conjecture 5** — $T_{n\times n}$ is square free
- **Conjecture 6** — The Magnitude of the Maresh Multi-fraction F_n converges to $\frac{1}{\sqrt{n}}$

B. Negative Bases

There are existing extensions that deal with negative bases [40]. To what extent can those be harmonized with the extensions in this paper?

C. Rational Bases

There are existing extensions that deal with rational bases [21]. To what extent can those be harmonized with the extensions in this paper?

D. Claims about the Standard Prouhet-Thue-Morse Sequence

These claims are found in [1].

Conjecture 7.

$$\forall x : x \geq 0, T_2(\textit{A004760}(x+1)) = 1 - T_2(x) \tag{73}$$

Conjecture 8.

$$\forall x : x \geq 0, T_2(\textit{A160217}(x)) = 1 - T_2(x) \tag{74}$$

Conjecture 9.

$$\mathcal{G}.\mathcal{F}.\ A(x)\ \textit{satisfies} : A(x) = \frac{x}{1-x^2} + (1-x)\cdot A(x^2) \tag{75}$$

Conjecture 10.

$$T_2((2^m-1)^2) = \frac{1-(-1)^m}{2} \tag{76}$$

E. Claims about the Inverted Prouhet-Thue-Morse Sequence $\langle 1, 0, 0, 1, \ldots\rangle$

These claims are found in [5].

Conjecture 11.

$$\begin{gathered}\textit{If } A(n) = (a(0), a(2), \ldots, a(2^n-1)),\\ \textit{then } A(n+1) = (A(n), 1-A(n)),\\ \textit{where } T_2 = 1 - [x^n]A(x)\end{gathered} \tag{77}$$

F. Claims about the Signed Prouhet-Thue-Morse Sequence $\langle 1, -1, -1, 1, \ldots\rangle$

These claims are found in [18].

Conjecture 12.

$$\begin{gathered}\mathcal{G}.\mathcal{F}.\ A(x)\ \textit{makes}\ 0 = f(A(x), A(x^2), A(x^4))\\ \textit{where } f(u,v,w) = v^3 - 2uvw + u^2w\\ \textit{and } T_2(n) = \frac{1-[x^n]A(x)}{2}\end{gathered} \tag{78}$$

Conjecture 13.

$$\begin{gathered}\mathcal{G}.\mathcal{F}.\ A(x)\ \textit{makes}\ 0 = f(A(x), A(x^2), A(x^3), A(x^6))\\ \textit{where } f(a,b,c,d) = a^3d - 3a^2bd + 3ab^2d - b^3c\\ \textit{and } T_2(n) = \frac{1-[x^n]A(x)}{2}\end{gathered} \tag{79}$$

Conjecture 14.

$$(-1)^{T_2(n)} = \frac{B_n(\textit{–A038712}(1)\cdot 0!, \ldots, \textit{–A038712}(n)\cdot(n–1)!)}{n!} \tag{80}$$

Where

- $B_n(x_1, \ldots, x_n)$ *is the* n*-th complete Bell polynomial*
- *A038712 [63] is* $a(n) = 2^{A007814(n)+1} - 1$
- *A007814 [64] is the exponent of highest power of 2 dividing* n

Conjecture 15.

$$\forall x : x \geq 0, (-1)^{T_2(x)} = \textit{A008836}(\textit{A005940}(1+x)) \tag{81}$$

Where:

- *A008836 [65] is Liouville's function* $\lambda(n) = (-1)^k$*, where* k *is number of primes dividing* n *(counted with multiplicity)*
- *A005940 [66] is the Doudna sequence*

G. Claims about the Standard Extended Prouhet-Thue-Morse Sequence For $n = 3$

These claims are found in [44].

Conjecture 16.

$$\forall x : x \geq 0, T_3(x) = \textit{A026600}(x+1) - 1 \tag{82}$$

where

- *A026600 [67]* $a(n)$ *is the* n*-th letter of the infinite word generated from* $w(1) = 1$ *inductively by* $w(n) =$ *JUXTAPOSITION*$\{w(n-1), w'(n-1), w"(n-1)\}$*, where* $w(k)$ *becomes* $w'(k)$ *by the cyclic permutation* $1 \to 2 \to 3 \to 1$ *and* $w"(k) = (w')'(k)$.

IX. Appendix 1 — Other Considered Extensions

There are a number of extensions that don't meet the strict standard of mutual equivalence laid out in this paper, but are nonetheless worthy of mention.

A. Honorable Mention — Square Free Ternary Sequences

A family of sequences commonly derived from the Prouhet-Thue-Morse Sequence is the morphism $a \to abc, b \to ac, c \to b$, with various substitutions of the letters [4, 23–35]. They have the wonderful property of begin among the smallest square-free sequences, as measured by alphabet size. This generates sequences like:

- $\langle 2,1,0,2,0,1,2,1,0,\ldots\rangle$ [4, 23] $(2\to 210, 1\to 20, 0\to 1)$
- $\langle 1,2,3,1,3,2,1,2,3,\ldots\rangle$ [24] $(1\to 123, 2\to 13, 3\to 2)$
- $\langle 3,2,1,3,1,2,3,2,1,\ldots\rangle$ [25] $(3\to 321, 2\to 31, 1\to 2)$
- $\langle 0,1,2,0,2,1,0,1,2,\ldots\rangle$ [26] $(0\to 012, 1\to 02, 2\to 1)$
- $\langle 2,1,3,2,3,1,2,1,3,\ldots\rangle$ [27] $(2\to 213, 1\to 23, 3\to 1)$
- $\langle 0,2,1,0,1,2,0,2,1,\ldots\rangle$ [28] $(0\to 021, 2\to 01, 1\to 2)$
- $\langle 2,3,1,2,1,3,2,3,1,\ldots\rangle$ [29] $(2\to 231, 3\to 21, 1\to 3)$
- $\langle 3,1,2,3,2,1,3,1,2,\ldots\rangle$ [30] $(3\to 312, 1\to 32, 2\to 1)$
- $\langle 1,2,0,1,0,2,1,2,0,\ldots\rangle$ [31] $(1\to 120, 2\to 10, 0\to 2)$
- $\langle 1,3,2,1,2,3,1,3,2,\ldots\rangle$ [32] $(1\to 132, 3\to 12, 2\to 3)$
- $\langle 2,0,1,2,1,0,2,0,1,\ldots\rangle$ [33] $(2\to 201, 0\to 21, 1\to 0)$
- $\langle 1,0,2,1,2,0,1,0,2,\ldots\rangle$ [34] $(1\to 102, 0\to 12, 2\to 0)$
- $\langle 1,0,-1,1,-1,0,1,0,\ldots\rangle$ [35] $(1\to 10\text{-}1, 0\to \text{-}1 1, -1\to 0)$

B. A071858, A051329

These sequences [68, 69] are of the form

$$a(n) = p_s(n) \pmod{s+1} \tag{83}$$

In the notation of some other papers, they are $t_{2,3}$ and $t_{3,4}$, respectively.

C. A005681

D. A239110

E. A317189

F. A089215

G. A029884

X. Appendix 2 — General Performance Results

Fig. 2. Benchmark results up to seconds.

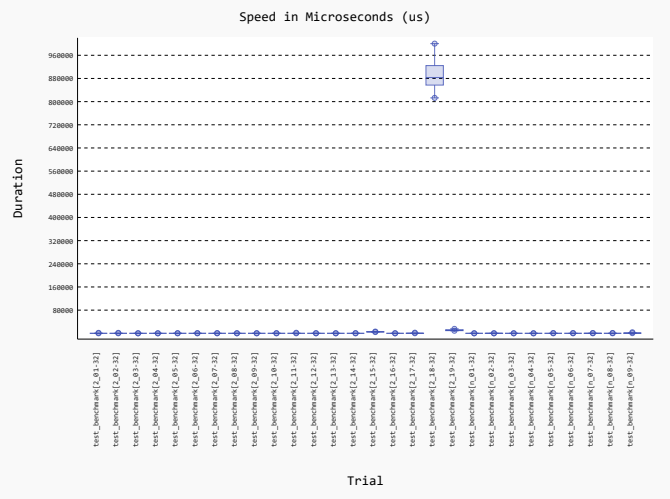


Fig. 3. Benchmark results up to milliseconds.

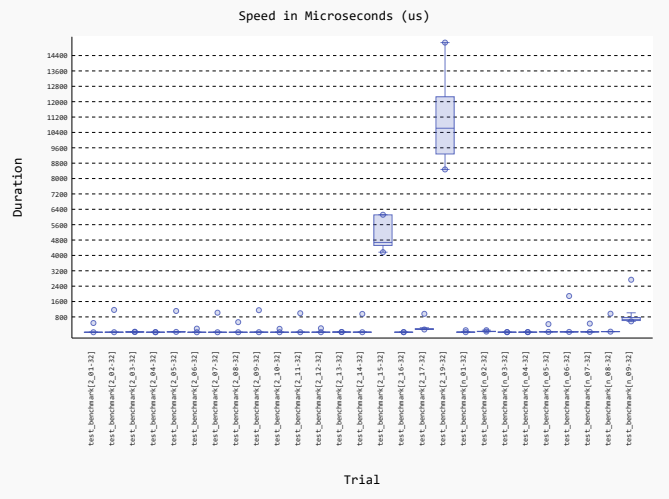


Fig. 4. Benchmark results up to microseconds.

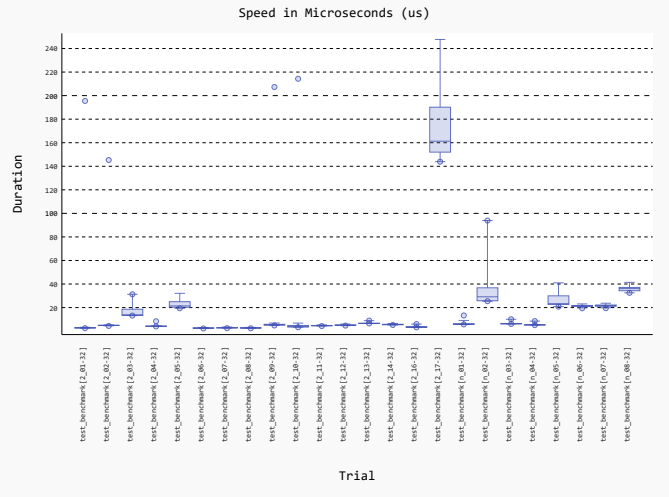


Fig. 5. Compared Complexity for Fixed Size Integers and One Element.

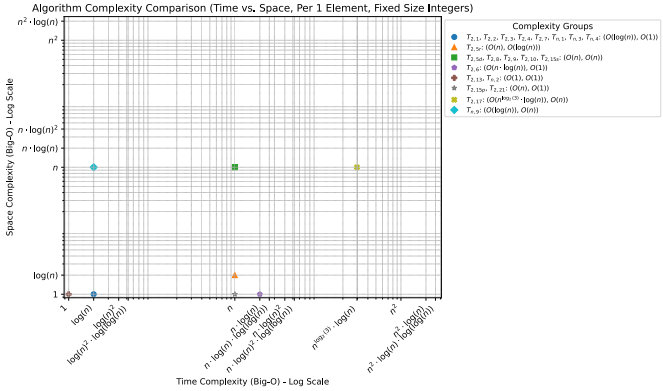


Fig. 6. Compared Complexity for Fixed Size Integers and n Elements.

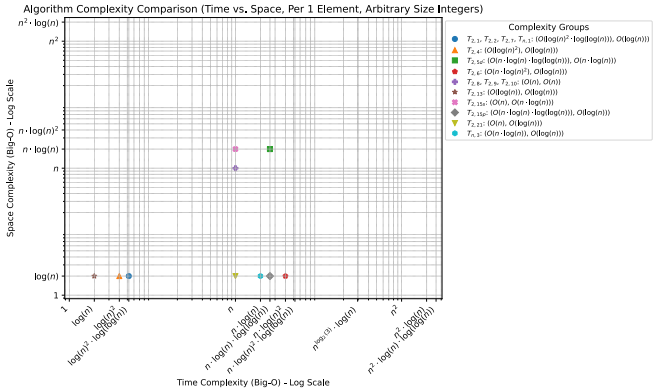


Fig. 7. Compared Complexity for Arbitrary Size Integers and One Element.

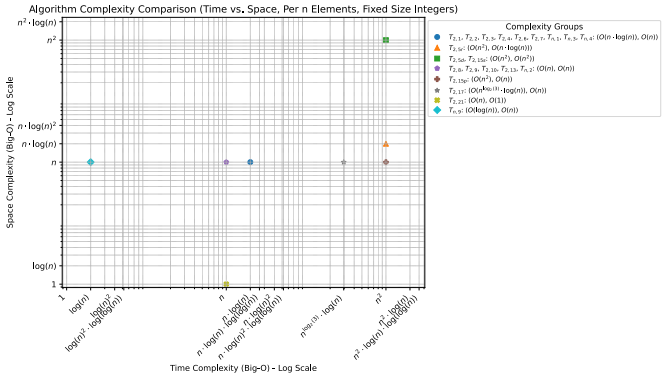
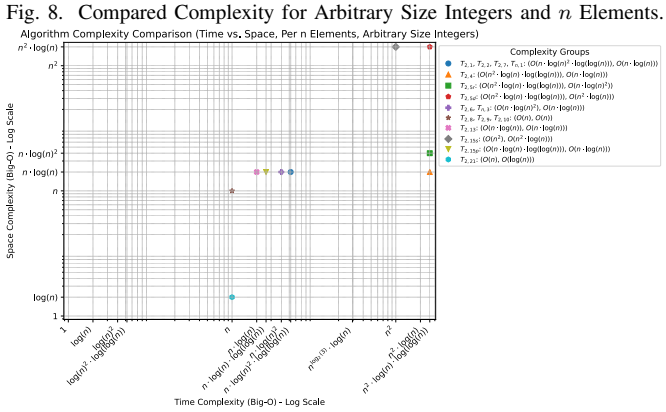


Fig. 8. Compared Complexity for Arbitrary Size Integers and n Elements.



These figures are temporary and will be replaced with something prettier when things are finalized. Complexity figures assume that the base is constant and therefore factored out.

XI. Appendix 3 — Complexity of Orig. Definitions

Need to clarify: space complexity for 'in total' assumes you are storing all n numbers. This may not be the case. If you are merely iterating over them, remove that factor.

A. Complexity of Original Def. 1 — Parity of Hamming Weight

1) Time Complexity: In an idealized case, this definition will simplify to:

$$T_{2,1}(n) = \left(\sum_{i=0}^{\lceil \log_2(n+1) \rceil} \left\lfloor \frac{n}{2^i} \bmod 2 \right\rfloor \right) \bmod 2 \qquad (84)$$

This is pretty explicitly $O(\log(n))$ operations. This means that generating the first n entries will take $O(n \log(n))$ operations.

In languages with dynamically sized integers, this can be slightly more complicated. In the above, we perform $\log(n)$ bit shifts, multiplications, moduli, and additions. Since a bit shift is constant time, calculation will be dominated by multiplication, division, and moduli. Each of these take $O(\log(n) \cdot \log(\log(n)))$, where n is the largest number involved. This means that in such languages, we can expect it to take $O(\log(n)^2 \cdot \log(\log(n)))$ operations per element, for $O(n \cdot \log(n)^2 \cdot \log(\log(n)))$ in total.

TABLE II
Time Complexity Summary of Standard Definition 1

	Fixed Size	Arbitrary Size
Per Element	$O(\log(n))$	$O(\log(n)^2 \cdot \log(\log(n)))$
In Total	$O(n \cdot \log(n))$	$O(n \cdot \log(n)^2 \cdot \log(\log(n)))$

2) Space Complexity: This is one of the more space-efficient implementations. Each element takes at most the same size as the passed integer. In languages that use Fixed Size integers, that means it will take $O(1)$ space. In languages like Python that use Arbitrary Size integers, it would take $O(\log(n))$ space, where n is the largest element you intend to calculate. If you intend to store all n elements, it will therefore take $O(n)$ or $O(n \cdot \log(n))$ space.

TABLE III
Space Complexity Summary of Standard Definition 1

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

B. Complexity of Original Def. 2 — Powers of Negative One

1) Time Complexity: This algorithm is dominated by the time it takes to calculate $p()$, as raising (-1) to a power just requires a number of sign flips. This means we can duplicate the analysis from above.

TABLE IV
Time Complexity Summary of Standard Definition 2

	Fixed Size	Arbitrary Size
Per Element	$O(\log(n))$	$O(\log(n)^2 \cdot \log(\log(n)))$
In Total	$O(n \cdot \log(n))$	$O(n \cdot \log(n)^2 \cdot \log(\log(n)))$

2) Space Complexity:

TABLE V
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 2

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

C. Complexity of Original Definition 3 — Root of Unity

1) Time Complexity:

TABLE VI
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 3

	Fixed Size	Arbitrary Size
Per Element	$O(\log(n))$	
In Total	$O(n \cdot \log(n))$	

2) Space Complexity:

TABLE VII
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 3

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

D. Complexity of Original Definition 4 — Recursion

1) Time Complexity: At each step in calculation, the value of n passed to the next recursion is halved. This means that it will take $O(\log_2(n))$ recursive steps. Each recursion involves at maximum 2 subtractions and a bit shift. In most languages with Fixed Size integers, this will take constant time. However, in languages with Arbitrary Size integers these subtractions will typically take $O(\log(n))$, where n is the largest integer in the operation. This means we can expect it to take $O(\log(n)^2$ operations.

TABLE VIII
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 4

	Fixed Size	Arbitrary Size
Per Element	$O(\log(n))$	$O(\log(n)^2)$
In Total	$O(n \cdot \log(n))$	$O(n \cdot \log(n)^2)$

2) Space Complexity: This is one of the more space-efficient implementations. Each element takes at most the same size as the passed integer. In languages that use Fixed Size integers, that means it will take $O(1)$ space. In languages like Python that use Arbitrary Size integers, it would take $O(\log(n))$ space, where n is the largest element you intend to calculate. If you intend to store all n elements, it will therefore take $O(n)$ or $O(n \cdot \log(n))$ space.

TABLE IX
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 4

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

E. Complexity of Original Def. 5 — Floor-Ceiling Difference

1) Time Complexity: memoization doesn't seem to help in the worst case of $1\ldots1_2$, so you should still end up calculating the value of $b()$ for every positive number less than n

TABLE X
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 5

	Fixed Size	Arbitrary Size
Per Element	$O(n)$	$O(n \cdot \log(n) \cdot \log(\log(n)))$
In Total	$O\left(n^2\right)$	$O\left(n^2 \cdot \log(n) \cdot \log(\log(n))\right)$

2) Space Complexity: There are two ways to implement this algorithm in terms of space complexity. They both have equal worst-case time complexity. The first is to take the recursive approach, and the second is to use dynamic programming.

In a recursive approach, you will end up descending $O(\log(n))$ stack frames, each of which will contain at minimum 1 integer. In the dynamic approach, you will keep a table of all the values of $b()$ from 0 through n. The biggest difference between these approaches is that in the recursive approach you may need to repeat calculations.

TABLE XI
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 5

		Fixed Size	Arbitrary Size
Recursive	**Per Element**	$O(\log(n))$	$O(n \cdot \log(n))$
	In Total	$O(n \cdot \log(n))$	$O\left(n^2 \cdot \log(n)\right)$
Dynamic	**Per Element**	$O(n)$	$O(n \cdot \log(n))$
	In Total	$O\left(n^2\right)$	$O\left(n^2 \cdot \log(n)\right)$

F. Complexity of Original Def. 6 — Highest Bit Difference

1) Time Complexity: Since this algorithm works sequentially, and cannot perform computation of an arbitrary element without recursing to the base case, the time is equal on a per-element and in-total basis

TABLE XII
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 6

	Fixed Size	Arbitrary Size
Per Element	$O(n \cdot \log(n))$	$O\left(n \cdot \log(n)^2\right)$
In Total	$O(n \cdot \log(n))$	$O\left(n \cdot \log(n)^2\right)$

2) Space Complexity:

TABLE XIII
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 6

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

G. Complexity of Orig. Def. 7 — Hamming Weight Compl.

1) Time Complexity: Like in $T_{2,2}$, this definition is dominated by the calculation of $p()$ inside of $\overline{p}()$. This means that we can duplicate the analysis there.

TABLE XIV
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 7

	Fixed Size	Arbitrary Size
Per Element	$O(\log(n))$	$O(\log(n)^2 \cdot \log(\log(n)))$
In Total	$O(n \cdot \log(n))$	$O(n \cdot \log(n)^2 \cdot \log(\log(n)))$

2) Space Complexity:

TABLE XV
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 7

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

H. Complexity of Original Definition 8 — Invert and Extend

1) Time Complexity: At each step, we copy the previous sequence, invert it, and extend it. This means that at each step we are doubling the length of the sequence (($O(2^k)$). Since we only double when we've reached the next power of two, this cancels out, leading to $O(2^{\log_2(n)}) = O(n)$. Because this method cannot produce an individual entry, fixed size and per-element analyses are identical.

We assume that languages with arbitrary sized integers will still provide an equivalent to booleans or single bit/byte numbers. If they do not, add a factor of $\log(n)$

TABLE XVI
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 8

	Fixed Size	Arbitrary Size
Per Element	$O(n)$	$O(n)$
In Total	$O(n)$	$O(n)$

2) Space Complexity:

TABLE XVII
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 8

	Fixed Size	Arbitrary Size
Per Element	$O(n)$	$O(n)$
In Total	$O(n)$	$O(n)$

I. Complexity of Original Def. 9 — Substitute and Flatten

1) Time Complexity: The time and space complexity of this method are as above.

TABLE XVIII
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 9

	Fixed Size	Arbitrary Size
Per Element	$O(n)$	$O(n)$
In Total	$O(n)$	$O(n)$

2) Space Complexity:

TABLE XIX
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 9

	Fixed Size	Arbitrary Size
Per Element	$O(n)$	$O(n)$
In Total	$O(n)$	$O(n)$

J. Complexity of Original Definition 10 — Recursive Rotation

1) Time Complexity: The time and space complexity of this method are as above.

TABLE XX
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 10

	Fixed Size	Arbitrary Size
Per Element	$O(n)$	$O(n)$
In Total	$O(n)$	$O(n)$

2) Space Complexity:

TABLE XXI
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 10

	Fixed Size	Arbitrary Size
Per Element	$O(n)$	$O(n)$
In Total	$O(n)$	$O(n)$

K. Complexity of Orig. Def. 11 — Odious Number Derivation

1) Time Complexity: The problem I'm running into is I don't know the time complexity of generating odious numbers

TABLE XXII
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 11

	Fixed Size	Arbitrary Size
Per Element		
In Total		

2) Space Complexity:

TABLE XXIII
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 11

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

L. Complexity of Orig. Def. 12 — Evil Numbers Derivation 1

1) Time Complexity: The problem I'm running into is I don't know the time complexity of generating odious numbers

TABLE XXIV
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 12

	Fixed Size	Arbitrary Size
Per Element		
In Total		

2) Space Complexity:

TABLE XXV
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 12

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

M. Complexity of Orig. Def. 13 — Evil Nums. Derivation 2

1) Time Complexity: Fortunately, this algorithm can be implemented entirely using bitwise operations. This helps the efficiency metrics quite a lot, as with fixed words this is $O(1)$ and arbitrarily sized integers it is $O(\log(n))$ time.

TABLE XXVI
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 13

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

2) Space Complexity:

TABLE XXVII
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 13

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

N. Complexity of Orig. Def. 14 — Odious & Evil Derivation

1) Time Complexity:

TABLE XXVIII
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 14

	Fixed Size	Arbitrary Size
Per Element		
In Total		

2) Space Complexity:

TABLE XXIX
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 14

	Fixed Size	Arbitrary Size
Per Element		
In Total		

O. Complexity of Orig. Def. 15 — Gould's Seq. Derivation

1) Time Complexity: There are two ways one could reasonably calculate this. The first is by building each row of Pascal's Triangle iteratively. This allows you to avoid multiplication whenever possible, and lets you apply a bitmask or modulus operation to take the parity of each entry. The downside is that this version is not parallelizable. Using the bit mask approach, this means that each entry will take $O(n)$ time.

The other is to take advantage of the relation $\binom{n}{k} = \binom{n}{k-1} \cdot \frac{n-(k-1)}{k}$. This allows you to calculate each row independently, using $\frac{n}{2}$ moduli, multiplications, and divisions. This means that each entry will take $O(n)$ operations, each of which take $O(\log(n) \cdot \log(\log(n)))$ if with arbitrary sized integers, totaling $O(n)$ or $O(n \cdot \log(n) \cdot \log(\log(n)))$.

TABLE XXX
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 15

		Fixed Size	Arbitrary Size
Serial	**Per Element**	$O(n)$	$O(n)$
	In Total	$O\left(n^2\right)$	$O\left(n^2\right)$
Parallel	**Per Element**	$O(n)$	$O(n \cdot \log(n) \cdot \log(\log(n)))$
	In Total	$O\left(n^2\right)$	$O\left(n^2 \cdot \log(n) \cdot \log(\log(n))\right)$

2) Space Complexity:

TABLE XXXI
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 15

		Fixed Size	Arbitrary Size
Serial	**Per Element**	$O(n)$	$O\left(n \cdot \log(n)\right)$
	In Total	$O\left(n^2\right)$	$O\left(n^2 \cdot \log(n)\right)$
Parallel	**Per Element**	$O(1)$	$O\left(\log(n)\right)$
	In Total	$O(n)$	$O\left(n \cdot \log(n)^2\right)$

P. Complexity of Orig. Def. 16 — Derivation from Blue Code

1) Time Complexity:

TABLE XXXII
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 16

	Fixed Size	Arbitrary Size
Per Element		
In Total		

2) Space Complexity:

TABLE XXXIII
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 16

	Fixed Size	Arbitrary Size
Per Element		
In Total		

Q. Complexity of Original Def. 17 — Generating Function 1

1) Time Complexity: Time complexity of multiplying polynomials via Karatsuba method is $O(n^{\log_2(3)})$. I think this means that, since we need to apply it $\log_2(n)$ times to get n terms, the overall complexity is $O(n^{\log_2(3)} \cdot \log(n))$

TABLE XXXIV
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 17

	Fixed Size	Arbitrary Size
Per Element	$O\left(n^{\log_2(3)} \cdot \log(n)\right)$?	
In Total	$O\left(n^{\log_2(3)} \cdot \log(n)\right)$	

2) Space Complexity:

TABLE XXXV
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 17

	Fixed Size	Arbitrary Size
Per Element	$O(n)$?	$O(n \log(n))$?
In Total	$O(n)$?	$O(n \log(n))$?

R. Complexity of Original Def. 18 — Generating Function 2

1) Time Complexity:

TABLE XXXVI
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 18

	Fixed Size	Arbitrary Size
Per Element		
In Total		

2) Space Complexity:

TABLE XXXVII
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 18

	Fixed Size	Arbitrary Size
Per Element		
In Total		

S. Complexity of Original Def. 19 — Hypergeometry

1) Time Complexity:

TABLE XXXVIII
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 19

	Fixed Size	Arbitrary Size
Per Element		
In Total		

2) Space Complexity:

TABLE XXXIX
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 19

	Fixed Size	Arbitrary Size
Per Element		
In Total		

T. Complexity of Original Def. 20 — Cellular Automaton

1) Time Complexity:

TABLE XL
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 20

	Fixed Size	Arbitrary Size
Per Element		
In Total		

2) Space Complexity:

TABLE XLI
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 20

	Fixed Size	Arbitrary Size
Per Element		
In Total		

U. Complexity of Original Def. 21 — Galois Duels

1) Time Complexity: Like several of the other definitions, this one cannot generate individual elements, and most generate the entire sequence up to that point.

TABLE XLII
TIME COMPLEXITY SUMMARY OF STANDARD DEFINITION 21

	Fixed Size	Arbitrary Size
Per Element	$O(n)$	$O(n \cdot \log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

2) Space Complexity:

TABLE XLIII
SPACE COMPLEXITY SUMMARY OF STANDARD DEFINITION 21

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(1)$	$O(\log(n))$

XII. APPENDIX 4 — COMPLEXITY OF EXT. DEFINITIONS

A. Complexity of Extension Def. 1 — Modular Digit Sums

In an idealized case, this definition will simplify to:

$$T_{n,1}(x,s) = \left(\sum_{i=0}^{\lceil \log_s(x+1) \rceil} \left\lfloor \frac{x}{s^i} \bmod s \right\rfloor \right) \bmod s \tag{85}$$

This is pretty explicitly $O(\log(n))$ operations. This means that generating the first n entries will take $O(n \log(n))$ operations.

In languages with dynamically sized integers, this can be slightly more complicated. In the above, we perform $\log(n)$ multiplications, moduli, and additions. Since additions are simpler, calculation will be dominated by multiplication, division, and moduli. Each of these take $O(\log(n) \cdot \log(\log(n)))$, where n is the largest number involved. This means that in such languages, we can expect it to take $O(\log(n)^2 \cdot \log(\log(n)))$ operations per element, for $O(n \cdot \log(n)^2 \cdot \log(\log(n)))$ in total.

TABLE XLIV
TIME COMPLEXITY SUMMARY OF EXTENDED DEFINITION 1

	Fixed Size	Arbitrary Size
Per Element	$O(\log(n))$	$O(\log(n)^2 \cdot \log(\log(n)))$
In Total	$O(n \cdot \log(n))$	$O(n \cdot \log(n)^2 \cdot \log(\log(n)))$

1) Space Complexity: This is one of the more space-efficient implementation. Each element takes at most the same size as the passed integer. In languages that use Fixed Size integers, that means it will take $O(1)$ space. In languages like Python that use Arbitrary Size integers, it would take $O(\log(n))$ space, where n is the largest element you intend to calculate. If you intend to store all n elements, it will therefore take $O(n)$ or $O(n \cdot \log(n))$ space.

TABLE XLV
SPACE COMPLEXITY SUMMARY OF EXTENDED DEFINITION 1

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

B. Complexity of Extension Definition 2 — Roots of Unity

1) Time Complexity:

TABLE XLVI
TIME COMPLEXITY SUMMARY OF EXTENDED DEFINITION 2

	Fixed Size	Arbitrary Size
Per Element	$O(b)$	
In Total	$O(bn)$	

2) Space Complexity:

TABLE XLVII
SPACE COMPLEXITY SUMMARY OF EXTENDED DEFINITION 2

	Fixed Size	Arbitrary Size
Per Element	$O(b)$	$O(b \cdot \log(n))$
In Total	$O(bn)$	$O(bn \cdot \log(n))$

C. Complexity of Extension Definition 3 — Recursion

1) Time Complexity: At each step in calculation, the value of n passed to the next recursion is divided by s (the selected base). This means that it will take $O(\log_s(n))$ recursive steps. Each recursion involves at maximum 2 subtractions and a bit shift. In most languages with Fixed Size integers, this will take constant time. However, in languages with Arbitrary Size integers these subtractions will typically take $O(\log(n))$, where n is the largest integer in the operation. This means we can expect it to take $O(\log(n)^2$ operations.

TABLE XLVIII
TIME COMPLEXITY SUMMARY OF EXTENDED DEFINITION 3

	Fixed Size	Arbitrary Size
Per Element	$O(\log(n))$	$O(\log(n)^2)$
In Total	$O(n \cdot \log(n))$	$O(n \cdot \log(n)^2)$

2) Space Complexity: This is one of the more space-efficient implementations. Each element takes at most the same size as the passed integer. In languages that use Fixed Size integers, that means it will take $O(1)$ space. In languages like Python that use Arbitrary Size integers, it would take $O(\log(n))$ space, where n is the largest element you intend to calculate. If you intend to store all n elements, it will therefore take $O(n)$ or $O(n \cdot \log(n))$ space.

TABLE XLIX
SPACE COMPLEXITY SUMMARY OF EXTENDED DEFINITION 3

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

D. Complexity of Extension Def. 4 — Highest Digit Difference

1) Time Complexity:

TABLE L
TIME COMPLEXITY SUMMARY OF EXTENDED DEFINITION 4

	Fixed Size	Arbitrary Size
Per Element	$O(\log(n))$	
In Total	$O(n \cdot \log(n))$	

2) Space Complexity:

TABLE LI
SPACE COMPLEXITY SUMMARY OF EXTENDED DEFINITION 4

	Fixed Size	Arbitrary Size
Per Element	$O(1)$	$O(\log(n))$
In Total	$O(n)$	$O(n \cdot \log(n))$

E. Complexity of Extension Def. 5 — Increment and Extend

1) Time Complexity:

TABLE LII
TIME COMPLEXITY SUMMARY OF EXTENDED DEFINITION 5

	Fixed Size	Arbitrary Size
Per Element		
In Total		

2) Space Complexity:

TABLE LIII
SPACE COMPLEXITY SUMMARY OF EXTENDED DEFINITION 5

	Fixed Size	Arbitrary Size
Per Element		
In Total		

F. Complexity of Extension Def. 6 — Substitute and Flatten

1) Time Complexity:

TABLE LIV
TIME COMPLEXITY SUMMARY OF EXTENDED DEFINITION 6

	Fixed Size	Arbitrary Size
Per Element		
In Total		

2) Space Complexity:

TABLE LV
SPACE COMPLEXITY SUMMARY OF EXTENDED DEFINITION 6

	Fixed Size	Arbitrary Size
Per Element		
In Total		

G. Complexity of Extension Definition 7 — Recursive Rotation

1) Time Complexity:

TABLE LVI
TIME COMPLEXITY SUMMARY OF EXTENDED DEFINITION 7

	Fixed Size	Arbitrary Size
Per Element		
In Total		

2) Space Complexity:

TABLE LVII
SPACE COMPLEXITY SUMMARY OF EXTENDED DEFINITION 7

	Fixed Size	Arbitrary Size
Per Element		
In Total		

H. Complexity of Ext. Def. 8 — Latin Square Constructions

1) Time Complexity:

TABLE LVIII
TIME COMPLEXITY SUMMARY OF EXTENDED DEFINITION 8

	Fixed Size	Arbitrary Size
Per Element		
In Total		

2) Space Complexity:

TABLE LIX
SPACE COMPLEXITY SUMMARY OF EXTENDED DEFINITION 8

	Fixed Size	Arbitrary Size
Per Element		
In Total		

I. Complexity of Extension Def. 9 — Generating Functions

1) Time Complexity: Time complexity of multiplying polynomials via Karatsuba method is $O(n^{\log_2(3)})$. I think this means that, since we need to apply it $\log_s(n)$ times to get n terms, the overall complexity is $O(b^{\log_2(3)} \cdot \log(n))$. I'm a *little* uncertain how the base scales things.

TABLE LX
TIME COMPLEXITY SUMMARY OF EXTENDED DEFINITION 9

	Fixed Size	Arbitrary Size
Per Element	$O(b^{\log_2(3)} \cdot \log(n))$?	
In Total	$O(b^{\log_2(3)} \cdot \log(n))$?	

2) Space Complexity:

TABLE LXI
SPACE COMPLEXITY SUMMARY OF EXTENDED DEFINITION 9

	Fixed Size	Arbitrary Size
Per Element	$O(n)$?	$O(n \cdot \log(n))$?
In Total	$O(n)$?	$O(n \cdot \log(n))$?

REFERENCES

[1] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A010060. The Prouhet-Thue-Morse Sequence.

[2] Lukas Spiegelhofer. The level of distribution of the thue–morse sequence. *Compositio Mathematica*, 156(12):2560–2587, 2020.

[3] Jean-Paul Allouche and Jeffrey Shallit. The ubiquitous prouhet-thue-morse sequence. In C. Ding, T. Helleseth, and H. Niederreiter, editors, *Sequences and their Applications*, pages 1–16, London, 1999. Springer London.

[4] Diyath Pannipitiya. To symbolic dynamics through the thue-morse sequence, 2024.

[5] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A010059. The inversion of the Prouhet-Thue-Morse Sequence.

[6] M. Kolář, M. K. Ali, and Franco Nori. Generalized thue-morse chains and their physical properties. *Phys. Rev. B*, 43:1034–1047, Jan 1991.

[7] Jorg Arndt. *Matters computational*. Springer, Berlin, Germany, October 2010.

[8] Ethan D. Bolker, Carl Offner, Robert Richman, and Catalin Zara. The prouhet–tarry–escott problem and generalized thue–morse sequences. *Journal of Combinatorics*, 7(1):117–133, 2016.

[9] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A000069. The Odious Numbers.

[10] Olivia Appleton-Crocker. Extending the thue-morse sequence source code. Available at: https://github.com/LivInTheLookingGlass/Thue-Morse, https://thue.oliviappleton.com, 2024. This repository contains the source code and data for the research presented in this paper.

[11] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A001969. The Evil Numbers.

[12] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A159481. Number of evil numbers $\leq n$.

[13] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A001316. The Gould Sequence.

[14] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A193231. Blue code for n.

[15] Jialin Ma, Isuru Fernando, and Xin Chen. *symengine: Interface to the 'SymEngine' Library*, 2022. Python package version 0.13.0.

[16] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore,

Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.

[17] Eric W. Weisstein. Thue-morse sequence. https://mathworld.wolfram.com/Thue-MorseSequence.html, 2024. Accessed: 2024-12-02.

[18] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A106400. The Prouhet-Thue-Morse Sequence, substituting $1 \to -1$ and $0 \to 1$.

[19] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A309303. Expansion of g.f. $(\sqrt{x+1} - \sqrt{1-3x})/(2(x+1)^{3/2})$.

[20] Joshua N. Cooper and Aaron M. Dutle. Greedy galois games, 2011.

[21] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A357448. The Extended Prouhet-Thue-Morse Sequence in Base 3/2.

[22] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A001285. The inversion of the Prouhet-Thue-Morse Sequence plus one.

[23] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A036577. Ternary Thue-Morse sequence: closed under $a \to abc, b \to ac, c \to b$.

[24] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A007413. A squarefree (or Thue-Morse) ternary sequence: closed under $1 \to 123, 2 \to 13, 3 \to 2$. Start with 1.

[25] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A036585. Ternary Thue-Morse sequence: closed under $a \to abc, b \to ac, c \to b$.

[26] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A036580. Ternary Thue-Morse sequence: closed under $a \to abc, b \to ac, c \to b$.

[27] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A005679. Ternary Thue-Morse sequence: closed under $a \to abc, b \to ac, c \to b$.

[28] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A036581. Ternary Thue-Morse sequence: closed under $a \to abc, b \to ac, c \to b$.

[29] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A036582. Ternary Thue-Morse sequence: closed under $a \to abc, b \to ac, c \to b$.

[30] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A036584. Ternary Thue-Morse sequence: closed under $a \to abc, b \to ac, c \to b$.

[31] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A036579. Ternary Thue-Morse sequence: closed under $a \to abc, b \to ac, c \to b$.

[32] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A036583. Ternary Thue-Morse sequence: closed under $a \to abc, b \to ac, c \to b$.

[33] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A036586. Ternary Thue-Morse sequence: closed under $a \to abc, b \to ac, c \to b$.

[34] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A036578. Ternary Thue-Morse sequence: closed under $a \to abc, b \to ac, c \to b$.

[35] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A029883. First differences of the Thue-Morse sequence (A001285).

[36] Ricardo Astudillo. On a class of thue-morse type sequences. *Journal of Integer Sequences*, 6, 2003.

[37] F. M. Dekking. The thue-morse sequence in base 3/2. *Journal of Integer Sequences*, 26, 2023.

[38] Jeffrey Shallit, Sonja Linghui Shan, and Kai Hsiang Yang. Automatic sequences in negative bases and proofs of some conjectures of shevelev, 2022.

[39] Vladimir Shevelev. Two analogs of thue-morse sequence, 2017.

[40] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A269027. The Extended Prouhet-Thue-Morse Sequence in base -2.

[41] Štěpán Starosta. Generalized thue-morse words and palindromic richness, 2011.

[42] Olga G. Parshina. *On Arithmetic Index in the Generalized Thue-Morse Word*, page 121–131. Springer International Publishing, 2017.

[43] Gerardo González Robert. Generalized thue-morse continued fractions, 2013.

[44] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A053838. The Extended Prouhet-Thue-Morse Sequence in Base 3, A.K.A sum of digits of n written in base 3 modulo 3.

[45] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A053839. The Extended Prouhet-Thue-Morse Sequence in Base 4, A.K.A sum of digits of n written in base 4 modulo 4.

[46] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A053840. The Extended Prouhet-Thue-Morse Sequence in Base 5, A.K.A sum of digits of n written in base 5 modulo 5.

[47] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A053841. The Extended Prouhet-Thue-Morse Sequence in Base 6, A.K.A sum of digits of n written in base 6 modulo 6.

[48] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A053842. The Extended Prouhet-Thue-Morse Sequence in Base 7, A.K.A sum of digits of n written in base 7 modulo 7.

[49] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A053843. The Extended Prouhet-Thue-Morse Sequence in Base 8, A.K.A sum of digits of n written in base 8 modulo 8.

[50] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A053844. The Extended Prouhet-Thue-Morse Sequence in Base 9, A.K.A sum of digits of n written in base 9 modulo 9.

[51] Matt McIrvin. Fractions, fractals and thue-morse sequences. Accessed: 2024-10-24, 2019. A 3 part series of blog posts: 1) https://mmcirvin.dreamwidth.org/499645.html, 2) https://mmcirvin.dreamwidth.org/499830.html, 3) https://mmcirvin.dreamwidth.org/500104.html.

[52] Leif Schaumann. Generalized results on the convergence of thue-morse turtle curves, 2024.

[53] Yi Cai and Vilmos Komornik. Difference of cantor sets and frequencies in thue–morse type sequences, 2020.

[54] Jin Chen and Zhi-Xiong Wen. On the abelian complexity of generalized thue-morse sequences. *Theoretical Computer Science*, 780:66–73, August 2019.

[55] Srećko Brlek. Enumeration of factors in the thue-morse word. *Discrete Applied Mathematics*, 24(1):83–96, 1989.

[56] C. Robinson Tompkins. Latin square thue-morse sequences are overlap-free, 2007.

[57] Dzmitry Badziahin and Evgeny Zorin. On generalized thue-morse functions and their values, 2015.

[58] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A287150. Fairest turn sequence for 3 players where the probability of a win for a player on his turn approaches 0.

[59] Carl D Offner. Repetitions of words and the thue-morse sequence.

[60] Narad Rampersad. Words avoiding 7/3-powers and the thue-morse morphism, 2003.

[61] Allan Adler and Shuo-Yen Robert Li. Magic cubes and prouhet sequences. *The American Mathematical Monthly*, 84(8):618–627, 1977.

[62] Eugene Prouhet. Mémoire sur quelques relations entre les puissances des nombres. *CR Acad. Sci. Paris*, 33(225):1851, 1851.

[63] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A038712.

[64] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A007814.

[65] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A008836.

[66] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A005940.

[67] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A026600.

[68] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A071858.

[69] OEIS Foundation Inc. The on-line encyclopedia of integer sequences (oeis). Accessed: 2024-10-24, 2024. OEIS Sequence ID: A051329.

XIII. Task Tracker

TABLE LXII
Comparison Matrix of the Standard Definitions to Their Extensions

4 Done, 5 Remaining

1	3	4	6	8	9	10	11	17
X	X	X						X
1	2	3	4	5	6	7	8	9

TABLE LXIII
Comparison Matrix of the Extended Definitions

X = done, S = started, O = target

2 Done, 6 Remaining

1								
X	2							
		3						
			4					
				5				
					6			
						7		
			X				8	
								9

TABLE LXIV
COMPARISON MATRIX OF THE STANDARD DEFINITIONS
X = done, S = started, O = target
4 Done, 2 Started, 14 Remaining

1																				
X	2																			
	X	3																		
X			4																	
	S			5																
					6															
						7														
X							8													
								9												
								S	10											
										11										
											12									
												13								
													14							
														15						
															16					
																17				
																	18			
																		19		
																			20	
																				21

TABLE LXV
LIST OF POSSIBLY PRESERVED PROPERTIES
X = Preserved, N = Not Preserved, L = Likely Modifier
4 Done, 4 Started, 8 Remaining

1	Equitable Distribution ($v(x) = 1$)	LX
2	Equitable Distribution (Galois)	N
3	Equitable Distribution (uniform)	
4	Equitable Distribution (normal)	
5	Equitable Distribution (exponential)	
6	Equitable Distribution (discrete)	N
7	Palindrome	N
8	Uniform Recurrence	
9	Cube Free	
10	Overlap Free	
11	Derived Square Free Sequence T_{dn}	N
12	Derived Square Free Sequence T_{pn}	LX
13	Derived Square Free Sequence $T_{n \times n}$	LX
14	Aperiodic	
15	Multifractions	LX
16	Fractal Turtle Graphics	

TABLE LXVI
ANALYZE COMPLEXITY
X = Done, S = started, N = Not Possible, L = Likely Modifier
40 Done, 13 Started, 28 Remaining

Def.	**Extended?**	**Standard**		**Extended**	
		Time	Space	Time	Space
1	X	X	X	X	X
2	LN	X	X	X	X
3	X	S	X	X	X
4	X	X	X	S	X
5		S	X		
6	X	S	S		
7		X	X		
8	X	X	X		
9	X	X	X	S	S
10	X	X	X		
11		S	X		
12		S	X		
13		X	X		
14					
15	LN	X	X		
16					
17	X	S	S		
18					
19					
20					
21	N	X	X		