An Algorithmic and Theoretical Analysis of Integer Classification via Recursive Binary Splitting

I. Executive Summary

This report provides an in-depth analysis of the "Binary Split Game" (BSG) and its application to integer classification. The BSG is a bespoke recursive algorithm that operates on the binary representation of integers, splitting strings and matching digits at corresponding positions to generate a sequence of transformed strings. The core novelty of the presented work lies not in the algorithm's fundamental operations, but in its use as a feature generator for a new taxonomy of integers. This classification framework is built upon a set of original metrics, most notably "Structural Complexity" (SC), and a multi-dimensional "Classification Vector," which quantify the behavior of an integer's binary representation under the BSG.

The analysis indicates that the BSG algorithm itself, while composed of simple steps, possesses a unique combination of rules for splitting and matching that distinguish it from standard string processing algorithms or arithmetic procedures. The integer classification system derived from it is novel in its approach, categorizing numbers based on their structural decay under this specific algorithmic probe, rather than traditional arithmetic properties. This offers a genuinely new perspective on integer structure.

The mathematical interest in this system is potentially significant. Strong parallels are identified with information theory, particularly in how the BSG acts as a filter for a specific type of low Kolmogorov complexity characterized by recursive bilateral symmetry. Furthermore, the generative processes for strings that are highly structured under the BSG show connections to formal language theory, specifically L-systems. However, the most compelling avenue for establishing deep mathematical interest lies in the hypothesized relationship between sequences derived from BSG metrics (like SC(n) or Depth(n)) and the theory of k-regular sequences. If these sequences are proven to be 2-regular, it would embed the BSG system within a rich,

established mathematical framework.

Suitability for peer-reviewed publication is contingent upon further rigorous research. The current framework provides a fascinating set of empirical observations and a novel algorithmic tool. To meet the standards of high-impact mathematical or theoretical computer science journals, formal proofs of key properties are necessary. The primary recommendation is to vigorously pursue the proof (or disproof) of the 2-regularity of the SC(n) and Depth(n) sequences. Concurrently, a formal comparative analysis of SC against established string complexity measures and a deeper investigation into the mathematical properties of the "structural families" generated by clustering BSG metrics would significantly bolster the work. With such developments, particularly the establishment of 2-regularity, the concepts presented would be well-positioned for impactful publication.

II. The Binary Split Game: Algorithmic Foundations and Classification Framework

The foundation of the proposed integer classification system is a deterministic algorithm termed the "Binary Split Game" (BSG). This algorithm operates on binary strings, typically derived from the binary representation of integers, and employs a recursive process of splitting, comparing, and matching to transform an initial string into a sequence of progressively (usually) shorter strings.

A. Mechanics of the Recursive Binary Split Algorithm

The BSG algorithm is defined by a precise set of rules that govern its execution.[1] The process begins with an input binary string,

s, which for integer classification is bin(n), the canonical base-2 representation of an integer $n \geq 1$ without leading zeros.[1] The steps are as follows:

1. **Input:** A binary string s. If $|s| \leq 1$, the algorithm conventionally outputs an empty string, ϵ.[1]
2. **Split:** The string s is divided into two halves. The midpoint m is calculated as

$m=\lfloor |s|/2 \rfloor$ (integer division). The left half, L, consists of characters $s_{0 \ldots m-1}$, and the right half, R, consists of characters $s_{m \ldots |s|-1}$.[1] For strings of odd length, this definition implies the right half will be one character longer than the left half.[1] This specific splitting rule is a fundamental component of the algorithm's behavior.

3. **Compare:** The digits at corresponding positions in L and R are compared. This comparison proceeds up to the length of the shorter half, $\tilde{l}=\min(|L|,|R|)$.[1]
4. **Match:** If the digits at the same position i (where $0 \le i < \tilde{l}$) in L and R are identical (i.e., $L_i=R_i$), it is considered a "match".[1]
5. **Recurse:** A new binary string, s', is formed by concatenating the matched digits in the order they appeared. This new string $s'=(L_i \mid 0 \le i < \tilde{l}, L_i=R_i)$ serves as the input for the next level of the algorithm, i.e., $s_{k+1}=f(s_k)$ where f is the binary-split operator.[1]
6. **End Conditions:** The recursive process terminates when either:
    - No matches are found in a comparison step, resulting in an empty string ($s'=\epsilon$) for the next level. This is often referred to as the "Null" state.[1]
    - The string is reduced to a single digit or becomes empty ($|s'| \le 1$), at which point it cannot be split further according to the definition of f(s) for $|s| \le 1$.[1]

The entire recursive process can be visualized as a three-dimensional cone-like structure. Each level of recursion corresponds to a ring of spheres, where spheres represent binary digits (e.g., blue for '0', red for '1'). Gray spheres indicate positions where matches occurred, contributing to the string at the next, smaller ring. Lines connect spheres between rings, illustrating the propagation of matched values, forming a cascade towards the cone's apex.[1]

The choice of integer division for the midpoint and comparison limited by the shorter half ensures a deterministic process. However, it also introduces an inherent asymmetry when processing strings of odd length. The final character of an odd-length string (which falls into the longer right half) does not participate in the comparison process at that level, effectively being "lost" for that stage of symmetry detection.[1] This consistent behavior for odd-length strings might subtly influence their classification pathways compared to even-length strings, potentially leading to distinct structural characteristics based on the parity of string lengths at various stages.

The termination conditions imply that single-bit strings ('0' or '1') and the empty string ('Null') act as absorbing states or fixed points for the transformation f. Consequently, the game's dynamics are inherently convergent, always leading to one of these terminal states after a finite number of steps for any finite input string.[1]

The distinctiveness of the BSG algorithm arises from this particular recursive halving and matching mechanism. It does not directly map to standard operations in formal language theory like concatenation or Kleene star, nor does it conform to typical update rules of cellular automata, which usually involve local neighborhoods rather than a global split-and-match operation.[1] This unique string transformation forms the basis of the proposed classification system.

B. The Proposed Integer Taxonomy: Metrics and Structural Families

The central innovation presented is the use of the BSG algorithm not as an end in itself, but as a means to generate a novel classification system, or taxonomy, for integers. This classification is predicated on the structural properties revealed by the algorithm's behavior when applied to the binary representation of numbers.[1]

1. Core Metrics: "Structural Complexity" and the "Initial Classification Vector"

To quantify the algorithm's output and the "shape" of the reduction process, a set of core metrics was developed. These form an initial multi-dimensional feature vector for each integer.[1]

The primary components are summarized in Table 1.

Table 1: Core Metrics of the Binary Split Game

Metric Name	Formal Definition [1]	Interpretation/Significance [1]
Depth ($d(n)$)	K, the number of non-empty applications of f until s_K is terminal. $s_0=bin(n), s_{k+1}=f(s_k)$.	Number of recursive steps; height of the cone visualization; measures persistence of the detected structural pattern.
Final State ($FS(n)$)	s_K, the terminal string: "0", "1", or "Null" (if $s_K=\epsilon$).	The ultimate reduction product of the algorithm; a categorical feature.

Max Width (Wmax)	$\max_{0 \le k \le K}	s_k
Color Ratio (CR)	(Total number of '1's in all sk, 0≤k≤K) / SC(n).	Ratio of '1's to '0's across all digits in the entire reduction path; indicates if the structure is predominantly composed of '1's or '0's.
Structural Complexity (SC(n))	$NC(n) = \sum_{k=0}^{K}	s_k

A key single-figure metric, **Structural Complexity (SC(n))**, is defined as the Total Node Count – the sum of the lengths of all binary strings appearing in an integer's reduction path.[1] For instance, the integer 990 (binary

1111011110) reduces as: 1111011110→11110→11→1. Its Structural Complexity is 10+5+2+1=18.[1] This metric inherently values numbers that not only undergo many reduction steps (high Depth) but also maintain longer intermediate strings during these steps.[1] This captures a notion of "information persistence" or "structural inertia" throughout the recursive decay, distinguishing SC from metrics that might only consider the final state or the number of steps. A number with a very deep reduction path where strings shrink rapidly might have a lower SC than a number with a shallower path but longer intermediate strings.

Empirical analysis of integers from 1 to 1000 reveals a non-uniform distribution of Structural Complexity values.[1] Certain SC scores (e.g., 10, 11, 12, 13) are far more common, while very low or very high scores are rare (e.g., SC=18 for 990 is unique in this range).[1] This non-uniformity is significant; if the BSG's rules were arbitrary or did not resonate with underlying structural preferences in binary representations, a more uniform or simple monotonic distribution might be expected. The observed peaks and troughs suggest that the BSG acts as a non-trivial filter, highlighting preferred structural pathways or stabilities within the integers.[1]

2. Advanced Classification: Expanded Feature Vectors and High-Dimensional Clustering

To achieve a more nuanced classification, the initial four-component vector was expanded with additional metrics designed to capture more subtle aspects of the structural decay process, as detailed in Table 2.[1]

Table 2: Selected Expanded Feature Vector Components

Metric Category	Metric Name	Formula [1]	Interpretation [1]
Geometric & Shape	Taper Factor (τ)	(Wmax−wK)/d, where $w_K =	s_K
	Aspect Ratio (AR)	d/Wmax	Ratio of cone's height to base width; categorizes structure as "tall/slender" or "short/wide."
	Node Density (ρ)	$NC / \sum_{k=0}^{K}	s_k
Compositional & Color	Color Ratio per Level (CRk)	Vector of color ratios for each sk	Tracks evolution of '1'-dominance or '0'-dominance through recursion.
	Color Volatility (σC)	Standard deviation of (CRk)	Measures stability of string's binary composition during reduction.
	Match Ratio (μ)	Total matches / Total comparisons (all levels)	Efficiency of the matching process at each step or overall.
Topological & Connectivity	Connection Length Variance (σL)	Standard deviation of lengths of lines connecting spheres	Potential indicator of orderliness of information flow (conceptual).
	Convergence Factor (κ)	Mean in-degree of match → next-node edges	How many "match" nodes converge into a single node at the next level (conceptual).
	Level of Max Collapse (λ)	Ring number where most significant drop in node count occurs	Identifies critical transition point in structural decay (conceptual).

This expanded feature vector allows for a richer, multi-faceted "signature" of each number's behavior under the BSG. The proposed methodology leverages these vectors through high-dimensional clustering techniques.[1] Each number's normalized feature vector is treated as a point in a high-dimensional space. Clustering algorithms such as K-Means or DBSCAN are then applied to identify "structural families"—groups of numbers whose feature vectors are close in this space, implying shared structural characteristics as perceived by the BSG.[1] Visualization of these clusters, for instance, via Principal Component Analysis (PCA) to reduce dimensionality to 2D or 3D, has been demonstrated for numbers 1-1000 using a 6-dimensional vector (Depth, Node Count, Taper Factor, Match Ratio, Color Ratio, Color Volatility).[1]

The success of such high-dimensional clustering critically depends on the chosen features capturing relatively independent (orthogonal) aspects of the structure to provide maximum discriminatory power.[1] If many features are highly correlated, the effective dimensionality of the feature space is reduced, potentially diminishing the ability of clustering algorithms to find meaningful and interpretable groupings. Further empirical validation, such as correlation analysis between the proposed expanded features, would be necessary to optimize the feature set for robust clustering. The stability and interpretability of the discovered clusters are key areas for ongoing investigation: how sensitive are they to the range of numbers or the specific metrics included, and do these clusters reveal mathematically meaningful groupings?.[1]

III. Analysis of Novelty

A critical aspect of evaluating any new mathematical concept is determining its originality in relation to existing knowledge. This section assesses the novelty of the Binary Split Game algorithm itself, the integer classification system derived from it, and its primary metric, Structural Complexity.

A. Originality of the Binary Split Game Algorithm Itself

The Binary Split Game (BSG) is a deterministic recursive algorithm operating on binary strings.[1] While the general concept of recursive splitting is a well-established algorithmic paradigm, often termed "Divide & Conquer" [2], and is used in various contexts such as decision tree construction (where data is recursively partitioned based on feature values to optimize a criterion like the Gini impurity) [3], the BSG's specific rules differentiate it. The method of splitting (midpoint

$m=\lfloor |s|/2 \rfloor$), the comparison logic (digit-wise up to the length of the shorter half), and particularly the generation of the subsequent string purely from matched symbols based on positional symmetry, are bespoke to this algorithm.[1]

Its novelty is not in its individual components (binary representation, splitting, comparison) but in their *specific combination and iterative application for the purpose of generating structural descriptors of the string's form*. This contrasts with other uses of "binary splitting," such as the algorithm used by y-cruncher for high-precision evaluation of mathematical series, which is arithmetically focused rather than on string patterns.[5] Similarly, while Lempel-Ziv (LZ) complexity algorithms involve finding previous occurrences of substrings to build a dictionary of phrases for compression, their mechanism does not involve recursive halving or the BSG's specific symmetric matching rule.[6]

Searches for similar "recursive binary splitting string" algorithms in existing literature point to applications in data structure manipulation (like decision trees [8]) or subword segmentation in natural language processing (e.g., Morfessor [9]), neither of which employ the BSG's exact matching and recursion rules for integer classification. Investigations into "recursive binary splitting string symmetry" on platforms like arXiv reveal uses in domains such as Lie algebra decomposition or quantum physics encodings, which are structurally and contextually distinct from the BSG's application to integers.[10]

The BSG algorithm is not presented as a tool to solve a pre-existing problem (like primality testing or data compression, for which it is not effective [1]) but rather as a

generator of features intended to form the basis of a new classification system.[1] This purpose-driven design, focusing on extracting descriptors of structural decay, further underscores its distinctiveness.

B. Novelty of the Integer Classification System

The primary claim to originality for the concepts surrounding the Binary Split Game lies in its application as a *classifier of integers*.[1] The system proposes a taxonomy based on the observed behavior of integers' binary representations when subjected to the BSG algorithm, quantified by the bespoke metrics and feature vectors previously discussed.

Traditional number theory classifies integers based on their arithmetic properties (e.g., divisors, prime factorization, behavior under modular arithmetic) or simpler characteristics of their digit expansions (e.g., palindromic numbers, sum of digits).[1] Computer science commonly uses binary representations for integers, but typically for arithmetic operations or efficient storage, not for this type of iterated structural analysis.[12] The BSG classification, in contrast, is founded on the iterated application of the split-and-match function

f to the binary form of an integer: f(bin(n)), f(f(bin(n))), and so on.[1] This iterative, structural decay approach to defining numerical classes appears to be a genuinely new perspective.

The "structural families" that emerge from this classification method, whether through direct comparison of metric vectors or via high-dimensional clustering, are novel constructs.[1] While some of these families may partially overlap with known sets of numbers (e.g., the "Mersenne-like" family corresponding to numbers of the form

2k–1, whose binary representations are all '1's), the *criteria for inclusion* in a BSG family are new: shared values in the BSG-derived classification vector.[1] A significant aspect of this novelty is the system's potential to group arithmetically disparate numbers—for example, a palindromic prime and a non-palindromic composite—if their algorithmic reduction paths and resulting metrics are sufficiently similar.[1] This prioritization of algorithmic structural similarity over traditional arithmetic properties is a defining and original feature.

The 'Null' class, comprising numbers whose binary strings exhibit no detectable symmetry by the BSG in the first step, is also a unique construct, defined entirely by this specific algorithm's "failure" to process them beyond the initial input.[1]

This classification system is fundamentally *representation-dependent* (binary) and *algorithm-dependent*. This characteristic is both a source of its novelty (as it may reveal patterns tied specifically to this representation and algorithmic probe) and a point requiring careful consideration. The universality of the insights derived, or their connection to deeper, representation-independent mathematical truths, largely hinges on establishing formal properties, such as the potential k-regularity of its derived sequences, which will be discussed later.

C. Relationship to Established String Complexity Measures

A key metric in the BSG classification system is "Structural Complexity" (SC), defined as the sum of the lengths of all binary strings that appear in an integer's reduction path.[1] This metric is bespoke to the BSG framework. Its novelty and utility must be assessed by comparison with established string complexity measures.[1]

- **Kolmogorov Complexity (K-complexity):** This measures the length of the shortest computer program that can produce a given string, representing its algorithmic randomness or simplicity.[14] The analysis in [1] suggests that strings highly structured according to the BSG (e.g., high Depth or high SC) tend to have low K-complexity. This is because their recursive, symmetrical nature often allows for very concise generative programs (e.g., strings formed by Sk=A+A).[1] However, the 'Null' class can contain both strings of high K-complexity (truly random strings) and strings of low K-complexity whose simple patterns are not detected by the BSG's specific symmetry test (e.g., 11110000).[1] Thus, the BSG acts as a specific test for a particular type of low K-complexity—one characterized by recursive bilateral symmetry—rather than a general measure of K-complexity.
- **Lempel-Ziv (LZ) Complexity:** Related to data compression algorithms like LZ77 and LZ78, LZ complexity typically measures the number of distinct phrases or patterns encountered when parsing a sequence sequentially, often by building a dictionary of previously seen substrings.[6] This is fundamentally different from SC, which is derived from a recursive halving and symmetric matching process. For example, a highly repetitive string like ababab... would have low LZ complexity but could have high SC if the repetitions align with the BSG's splitting mechanism. Conversely, a string like 00...011...1 might have low LZ complexity (two long runs) but could be classified as 'Null' by the BSG if its halves do not match, resulting in low SC.[1]
- **Subword Complexity (pw(n) or Π-complexity):** This refers to the number of distinct substrings (or factors) of a given length n that appear in a word.[1] It is a general measure of a string's "richness" in terms of its unique constituent parts. SC, by contrast, is not a direct count of all subwords; it is a cumulative measure derived from the lengths of strings generated through a specific transformation process. A string with high SC might, for instance, have many repeated elements

within its reduction path, which could imply that the *set* of unique strings in the path (and thus their individual subword complexities) might not directly correlate with the SC value in a simple way. The relationship between SC and the subword complexities of the initial, intermediate, and final strings in the BSG path is an area for further exploration.

The "Structural Complexity" metric is therefore not a direct measure of compressibility in the standard LZ sense, nor is it a measure of global K-complexity. It quantifies "decomposability" or "structural persistence" under the BSG's specific recursive symmetric filter. Its novelty as a complexity measure hinges on demonstrating that it captures structural aspects of strings not equivalently captured by existing metrics or simple combinations thereof. This requires further research into its mathematical properties, such as its behavior on well-known families of strings (e.g., Thue-Morse sequences [20], Sturmian words) and whether it satisfies interesting bounds or inequalities, analogous to those studied for other complexity measures.[1]

Table 3: Comparative Overview: Structural Complexity vs. Established String Complexity Measures

Feature	Structural Complexity (SC)	Lempel-Ziv (LZ) Complexity	Kolmogorov (K) Complexity	Subword Complexity (pw(n))
Basis of Complexity	Recursive bilateral symmetry detection and persistence	Dictionary of distinct phrases encountered sequentially	Length of shortest program to generate the string	Number of distinct substrings of a given length n
Primary Mechanism	Recursive halving, positional matching, sum of path lengths	Building a vocabulary of new patterns during a single pass	Algorithmic incompressibility, theoretical limit of compression	Counting unique factors within the string
Sensitivity To	Specific recursive symmetries aligned with midpoint splitting	Repetitiveness of substrings, new pattern discovery	Ultimate algorithmic simplicity/randomness	Variety of local patterns (substrings) of a fixed length

Example Behavior	High for Sk=A+A type strings; 'Null' for 11110000	Low for highly repetitive strings (e.g., aN)	Low for Sk=A+A; high for random strings	Low for aN; high for strings rich in unique n-length factors
Typical Application Area	Proposed for integer classification based on binary structure	Lossless data compression	Theoretical measure of randomness, information theory	Analysis of linguistic complexity, combinatorial properties of words

This comparative overview underscores that SC targets a different facet of string structure than these established measures. Its value will be determined by the unique insights it offers into this specific type of structure and its potential connections to other mathematical properties.

IV. Assessment of Mathematical Interest

The mathematical interest in the Binary Split Game and its derived classification system stems from its potential to offer new perspectives on integer structures and its connections to various established mathematical fields.

A. Potential Contributions to Number Theory

The BSG system, as established, is not a primality test.[1] Analysis shows that prime and composite numbers can exhibit similar structural behaviors under the BSG; for example, the composite number 15 (binary

1111) and the prime number 31 (binary 11111) both reduce to the final state '1' via similar paths involving strings of '1's.[1] Furthermore, a significant proportion of prime numbers (41 out of the first 100 analyzed) fall into the 'Null' class, meaning their binary representations show no detectable symmetry by the BSG in the initial step.[1] The distribution of prime numbers across the BSG-defined structural classes also bears no resemblance to the asymptotic distribution predicted by the Prime Number

Theorem.[1]

Despite this independence from primality, the classification system does interact with known sets of numbers in number theory. For instance:

- The "Mersenne-like" family, characterized by binary strings of all '1's (e.g., for integers 3,7,15,31), directly corresponds to numbers of the form $2k-1$.[1]
- Palindromic numbers (integers whose binary representations are palindromes, like $17=10001_2$) are often identified and grouped by the algorithm due to their inherent symmetries.[1]

The novelty in these cases is not the discovery of these number types, but the specific BSG-derived metric criteria used for grouping them. This can lead to numbers with different arithmetic properties being placed in the same structural family if their BSG reduction paths are similar.[1]

The 'Null' family itself represents a novel number-theoretic construct: a set of integers defined by their failure to exhibit the specific recursive bilateral symmetry detected by the BSG.[1] The observation that many primes fall into this 'Null' class might suggest that this particular BSG-symmetry is a relatively uncommon feature in the binary structure of prime numbers. This could prompt number-theoretic inquiries into whether 'Null' class numbers, particularly primes, possess other distinguishing statistical characteristics in their binary representations that are

not this specific symmetry.

Moreover, the BSG framework allows for new questions about the "density" of its detected structural properties within the integers or within specific subsets like primes or composites.[1] For example, one could investigate the asymptotic behavior of the proportion of integers

$n \le N$ such that $\mathrm{Depth}(n)>0$, or $FS(n)='1'$. Such questions are analogous to classical problems concerning the density of primes or perfect squares, but applied to new, algorithmically-defined properties.

B. Connections to Information Theory, Formal Languages, and Combinatorics

The BSG framework exhibits interesting parallels with concepts from information theory, formal language theory, and combinatorics on words.

- **Information Theory:**
  - The Shannon entropy of an input binary string is generally a poor predictor of the BSG's behavior. The algorithm is primarily sensitive to positional order and recursive symmetry, not the statistical distribution of bits that Shannon entropy measures.[1] For instance,
    101010102 (high entropy) and 111100002 (also high entropy) can have vastly different BSG reduction paths, with the former potentially reducing deeply if its length aligns with the symmetry detection (e.g., (10)22k) and the latter yielding 'Null' immediately.[1]
  - A more fruitful connection exists with Kolmogorov Complexity (K-complexity). Strings that the BSG finds to be highly structured (e.g., achieving high Depth or SC) are often those with low K-complexity, as their recursive, symmetrical nature allows for very concise algorithmic descriptions (e.g., via rules like Sk=A+A).[1] However, the 'Null' class can contain both truly random strings (high K-complexity) and strings that are algorithmically simple but possess patterns the BSG does not detect (e.g.,
    111100002).[1] Thus, the BSG acts as an "Information Filter," selectively identifying a
    specific kind of low K-complexity—one characterized by recursive bilateral symmetry.[1]
- Formal Languages and Generative Processes:
  The generative process Sk=A+A (or more generally, Sk=Sk−1Sk−1), used to construct strings that exhibit maximal depth or high SC under the BSG, is directly analogous to simple substitution rules in formal language theory or the production rules found in L-systems (Lindenmayer systems).1 L-systems are known for their ability to generate self-similar and fractal structures through parallel rewriting rules.22 The BSG effectively "recognizes" and assigns high structural scores to strings that are compactly describable by such simple, recursive generative processes. This suggests that the "most structured" integers, according to the BSG, are those whose binary forms are constructible by these elementary, parallel rewriting mechanisms. This provides a concrete link to an established area of theoretical computer science.
- Combinatorics on Words:
  The BSG algorithm is fundamentally concerned with identifying specific patterns—recursive bilateral symmetries—within binary words. This naturally connects to the field of combinatorics on words, which studies properties like periodicity, pattern avoidance, and special classes of words (e.g., Sturmian words,

square-free words).1

The algorithm demonstrably favors certain repetitive structures. For example, strings composed entirely of '1's (Mersenne-like family) reduce to a single '1'.1 The behavior with alternating bit strings (e.g.,
1010102) is more nuanced: strings of the form (10)22k (like 10102 or 101010102) are processed deeply, as the recursive halving aligns with their structure. However, strings like 1010102 (length 6) result in a 'Null' state because the initial split (1012 vs 0102) yields no matches. This highlights that the BSG's symmetry detection is highly specific to the halving process; it is not merely about general repetitiveness but about repetitiveness that aligns precisely with the recursive bisections. This specificity is a key characteristic of the type of structure it identifies. Further research could explore the BSG's sensitivity to other known combinatorial properties of words.[1]

C. Potential for Uncovering New Mathematical Structures or Relationships

The ultimate mathematical interest of the BSG classification system hinges on its ability to uncover previously unobserved patterns or relationships. The primary potential lies in whether the "structural families," identified through clustering based on the expanded BSG feature vectors, exhibit interesting collective properties not explicitly used in their formation.[1] For example, if a cluster of numbers defined by a specific, complex BSG feature vector were found to consistently share other, non-obvious mathematical properties—such as a peculiar distribution of their prime factors (which could be investigated using tools for prime factorization [26]), or a connection to the coefficients of a specific type of generating function, or particular behaviors in other mathematical contexts—then the BSG classification method would transcend being a mere algorithmic curiosity and become a tool for mathematical discovery.[1]

The framework offers a new lens through which to view integers; the significance of what is seen through this lens will be determined by empirical investigation and the subsequent discovery of such unexpected correlations between BSG-defined families and established mathematical properties or constants.

Table 4: Illustrative Classification of Selected Integers by the Binary Split Game

Integer	Binary Rep.	Path (s0⟶s1⟶…⟶FS)	Depth (d)	SC	Final State (FS)	Color Ratio (CR)	Traditional Properties	BSG Class (Conceptual)
2	10	10⟶Null	0	2	Null	0.50	Prime	Null Class
3	11	11⟶1	1	3	1	1.00	Prime, 22–1	Mersenne-like (Simple)
5	101	101⟶Null	0	3	Null	0.67	Prime, Palindrome	Null Class
6	110	110⟶Null	0	3	Null	0.67	Composite	Null Class
7	111	111⟶1	1	4	1	1.00	Prime, 23–1	Mersenne-like (Simple)
10	1010	1010⟶10⟶Null	1	6	Null	0.50	Composite, (10)22	Alternating (Shallow)
15	1111	1111⟶11⟶1	2	7	1	1.00	Composite, 24–1	Mersenne-like (Deeper)
17	10001	10001⟶0⟶Null	1	6	Null	0.40 (approx, s2 is 0)	Prime, Palindrome	Simple Palindromic to '0'
31	11111	11111⟶11⟶1	2	8	1	1.00	Prime, 25–1	Mersenne-like (Deeper)
85	101010	101010	0	7	Null	0.57	Compo	Null

	1	1⟶Null					site, Palindrome, (101)2×(101)2	Class
170	10101010	10101010⟶1010⟶10⟶Null	2	14	Null	0.50	Composite, (10)24	Alternating (Deeper)
255	11111111	11111111⟶1111⟶11⟶1	3	15	1	1.00	Composite, 28−1	Mersenne-like (Very Deep)
990	1111011110	1111011110⟶11110⟶11⟶1	3	18	1	0.78	Composite	High SC, Complex Path

This table illustrates how the BSG classifies numbers based on their binary structural decay, often yielding groupings distinct from traditional arithmetic classifications. For example, both the prime 31 and the composite 15 fall into a "Mersenne-like (Deeper)" category due to similar reduction paths of their all-'1' binary strings. The alternating bit strings like 10 (10102) and 170 (101010102) show increasing depth and SC as their length, of the form (10)22k, increases.

V. Suitability for Peer-Reviewed Publication and Future Research Directions

The assessment of the Binary Split Game and its associated integer classification system for peer-reviewed publication involves evaluating its current strengths and weaknesses, identifying promising avenues for formal investigation that would enhance its academic standing, and suggesting appropriate publication venues.

A. Strengths and Weaknesses for Academic Scrutiny

The BSG framework possesses several strengths that make it intriguing:

- **Novelty of the Classification Approach:** The use of an iterated structural decay process, based on the bespoke BSG algorithm, to classify integers by their binary representations is a distinct and original perspective compared to traditional methods.[1]
- **Originality of Metrics:** Metrics such as "Structural Complexity" (SC) and the components of the expanded feature vector (e.g., Taper Factor, Color Volatility) are new constructs within this specific framework.[1]
- **Deterministic Algorithm:** The BSG algorithm itself is clearly defined and deterministic, allowing for reproducible results.[1]
- **Potential Theoretical Connections:** The system shows promising, albeit currently hypothesized, connections to established mathematical theories, notably k-regular sequences, L-systems, and aspects of information theory.[1]
- **Intuitive Visualization:** The cone-like visualization of the reduction process provides an intuitive way to understand the algorithm's behavior.[1]

However, from the perspective of rigorous academic publication, particularly in mathematics or theoretical computer science journals, there are areas that require further development:

- **Lack of Rigorous Comparison for SC:** The "Structural Complexity" metric has not yet been formally and rigorously compared against the spectrum of existing string complexity measures to establish its unique contributions and mathematical properties.[1]
- **Unexplored Properties of Structural Families:** The mathematical properties of the "structural families" derived from clustering BSG metrics are largely unexplored. It is not yet known if these algorithmically defined groups share other significant, non-obvious mathematical characteristics.[1]
- **Hypothesized Connections:** The most significant theoretical underpinning, the potential 2-regularity of sequences derived from BSG metrics, remains a hypothesis requiring formal proof.[1]
- **Limited Direct Applicability:** The system does not currently offer solutions to well-known, pre-existing mathematical problems, which can sometimes be a factor in assessing impact.[1]

The primary challenge for securing publication in high-tier journals will be to transition the work from a collection of interesting empirical observations and a novel algorithm to a framework supported by formally proven properties and demonstrated connections to established mathematical theory. The pursuit of k-regularity proofs is central to this endeavor.

B. High-Potential Avenues for Formal Investigation

Several avenues for formal mathematical research emerge from the BSG concepts, which, if successfully explored, would significantly strengthen its academic merit.

1. Investigation of k-Regularity of Derived Sequences (Primary Focus):
This is the most critical and promising direction. A sequence s(n) is k-regular if its k-kernel—the set of subsequences Kk(s)=s(ken+r):e≥0,0≤r<ke—generates a finitely-generated module (or vector space over a field like Q).28 Since the BSG operates on binary representations, the relevant value is
k=2.
It is hypothesized that sequences such as s1(n)=StructuralComplexity(n), s2(n)=Depth(n), and potentially others derived from the classification vector components, are 2-regular.1

- **Significance:** A proof of 2-regularity would connect the BSG system to the rich, well-established mathematical theory of k-regular and automatic sequences (prominently developed by Allouche and Shallit).[1] This would imply that these BSG metrics are not arbitrary but possess a deep, inherent "regularity" intrinsically tied to the binary representation of
n. Such sequences can often be computed by generalized automata (linear representations over a ring) and typically exhibit at most polynomial growth [1], a property that can be checked against empirical data for SC. Moreover, k-regular sequences often possess matrix representations for their terms (e.g., s(n)=uM(n0)M(n1)⋯M(nl)v, where nl…n0 is the base-k representation of n).[29] If SC(n) or Depth(n) were found to be 2-regular, such a matrix representation could provide powerful analytical tools for studying their asymptotic behavior or for more efficient computation for large
n.
- **Methodology:** The standard approach to proving (or disproving) k-regularity involves computing terms of the k-kernel of the sequence in question (e.g., for SC(n), the subsequences SC(2n), SC(2n+1), SC(4n), SC(4n+1), etc.) and attempting to find a finite set of basis sequences such that all other kernel sequences can be expressed as linear combinations (over Q or Z) of these basis elements.[1]
- **Tools:** The IntegerSequences Mathematica package, developed by Eric Rowland,

is specifically designed for working with, guessing, and analyzing k-regular sequences.[29] This package implements algorithms, such as Shallit's method for guessing k-regular relations from initial terms, by maintaining a set of generator sequences and finding linear dependencies among their k-kernel elements.[29] It can be an invaluable tool for exploring the 2-kernels of BSG-derived sequences and formulating conjectures about their basis.

2. Rigorous Analysis of "Structural Complexity" as a String Complexity Measure:
A formal study is needed to compare SC with established string complexity measures like Lempel-Ziv complexity, Kolmogorov complexity, and various subword complexities (e.g., Π-complexity, which counts distinct substrings).1 This involves:

- Investigating correlations and, more importantly, distinctions in what aspects of string structure they capture.
- Determining mathematical properties of SC: e.g., bounds, growth rates, behavior on known structured sequences like the Thue-Morse sequence [20] or de Bruijn sequences, and its computational complexity for a single string.[1]

3. Characterization of "Structural Families" and Their Mathematical Properties:
The "structural families" identified by clustering numbers based on their BSG feature vectors require deeper analysis.1

- Beyond confirming their statistical stability, research should investigate whether numbers within these algorithmically-defined families share other, non-obvious, mathematical properties (e.g., related to divisibility, distribution of prime factors [26], connections to coefficients of specific generating functions, or behavior in other mathematical systems).[1] This could lead to a feedback loop: if a family is found to correlate with a known property X, the BSG metrics could potentially be refined to enhance discrimination based on X.

4. Formalization of Generative Rules (e.g., via L-systems, Automata Theory):
The methods for constructing strings that exhibit maximal depth or complexity under the BSG (typically through recursive concatenation like Sk=A+A) can be formalized.1

- This involves characterizing the set of all strings generated by such rules, or the set of all strings achieving maximal depth for their length, using concepts from formal language theory such as L-systems [22] or other generative grammars. This would provide a formal understanding of the "most structured" strings as defined by this algorithm.

5. Exploration from a Dynamical Systems Perspective:
The iterative application of the BSG algorithm can be viewed as a discrete dynamical system operating on the space of binary strings.1

- Fixed points appear to be simple strings like "0", "1", or the empty string

(representing the 'Null' state). The "structural families" can be seen as related to the basins of attraction for these fixed points.[1]
- Analyzing properties such as convergence rates, the structure of pre-images, or the overall landscape of this dynamical system could offer another formal lens. While the system is convergent for finite strings, the paths to termination could be complex, and exploring if "chaotic" transient behavior or limit cycles exist for certain classes of (potentially infinite) input strings could be an interesting extension, connecting to concepts in symbolic dynamics.[34]

C. Recommendations for Manuscript Preparation

Should the pursued research directions yield significant results, a manuscript for peer-reviewed publication should:

- Begin with a clear and concise definition of the Binary Split Game algorithm and the novel metrics derived from it, particularly Structural Complexity and the components of the classification vector.
- Present the strongest proven results as the central theme (e.g., if 2-regularity of SC(n) is established, this should be a cornerstone of the paper).
- Include compelling empirical evidence, such as the distribution of SC values, PCA plots illustrating cluster formations, and illustrative examples of how different classes of integers are classified.
- Thoroughly discuss the connections to, and critically, the distinctions from, existing work in number classification, string complexity, and related areas of mathematics and computer science.
- Acknowledge any limitations of the current work and clearly state open questions or conjectures that arise, paving the way for future research.
- Ensure all claims are well-supported by data or formal arguments, and that the methodology is transparent and reproducible.

D. Suggested Publication Venues

The choice of publication venue will heavily depend on the nature and strength of the results obtained from further research. Based on the current understanding and potential developments:

Table 5: Potential Peer-Reviewed Publication Venues

Journal Title	Publisher	Relevant Scope [24]	Typical Content Themes	Potential Fit for Binary Split Game Research
Theoretical Computer Science	Elsevier	Algorithms, automata theory, formal languages, logic in computer science, computational complexity. [37]	Foundational aspects of computation, new algorithms, complexity analysis, formal methods.	Strong fit if 2-regularity of key sequences is proven, or if strong connections to automata theory or formal language theory (e.g., L-systems) are formalized.
Discrete Mathematics	Elsevier (North-Holland)	Broad discrete mathematics, combinatorics, graph theory, and their applications. [39]	Combinatorial structures, enumerative combinatorics, graph algorithms, properties of sequences.	Good fit if significant combinatorial properties of the BSG-derived sequences or structural families are established, or if the analysis of SC(n) reveals interesting combinatorial characteristics.
Journal of Combinatorics on Words (or similar specialized journals)	Various (e.g., via societies or publishers like Cambridge Univ. Press [24])	Study of words (finite sequences), formal languages, patterns in strings, automata theory. [25]	Properties of specific sequences (e.g., Thue-Morse), subword complexity, pattern avoidance, algorithmic aspects of words.	Ideal if the analysis of SC as a novel string complexity measure is well-developed, or if deep connections to known combinatorial word properties

				are proven.
Experimental Mathematics	Taylor & Francis	Formal results inspired by experimentation, conjectures suggested by experiments, data supporting significant hypotheses. [40]	Computational experiments leading to new mathematical insights, well-supported conjectures, exploration of mathematical structures using computational tools.	Very suitable for the current empirical aspects (clustering, SC distributions), especially if strong conjectures (like 2-regularity) are presented with substantial computational evidence but formal proofs are still developing. Also a good venue for showcasing the novel classification system and its visual outputs.
Recreational Mathematics Journals (e.g., Journal of Recreational Mathematics, or sections in broader math magazines)	Various	Puzzles, games, novel mathematical explorations accessible to a wider audience. [1]	Interesting mathematical patterns, new algorithmic games, accessible explorations of number properties.	Possible if the work is presented with an emphasis on the "game" aspect and its intriguing patterns, aimed at a less specialized audience, perhaps focusing on the visualization and empirical findings.

If the research successfully establishes the 2-regularity of key BSG-derived sequences, journals like *Theoretical Computer Science* or those focused on discrete mathematics and combinatorics on words would be primary targets. If the work remains more empirically focused, highlighting the novel classification and strong computational evidence for conjectures, *Experimental Mathematics* would be an excellent venue.

VI. Conclusion and Strategic Recommendations

The exploration of the "Binary Split Game" (BSG) and its application to integer classification has unveiled a conceptually novel and mathematically intriguing framework. This system offers a unique lens through which to examine the structural properties of integers based on the behavior of their binary representations under a bespoke recursive algorithm.

A. Consolidated Assessment: Novelty, Interest, and Publishability

- **Novelty:** The core algorithm, while employing simple operations, is combined in a unique manner for the purpose of feature generation. The primary novelty resides in the classification system itself: the use of an iterated structural decay process, quantified by original metrics like "Structural Complexity" (SC) and an expanded feature vector, to define "structural families" of integers. This approach is distinct from traditional arithmetic or simpler digit-based classifications and offers a new way to categorize numbers based on algorithmic behavior.
- **Mathematical Interest:** The potential mathematical interest is substantial. The characterization of the BSG as an "Information Filter" sensitive to a specific type of low Kolmogorov complexity (recursive bilateral symmetry) provides a strong information-theoretic underpinning. Connections to generative processes like L-systems are also evident. The most compelling prospect for deep mathematical interest lies in the hypothesized 2-regularity of sequences derived from BSG metrics (e.g., SC(n), Depth(n)). Proving this would link the BSG system to the well-established and rich theory of k-regular and automatic sequences, significantly elevating its theoretical standing.
- **Publishability:** In its current state as presented in the source documents, the work constitutes a fascinating set of empirical observations and a novel algorithmic proposal. For publication in high-impact mathematics or theoretical computer science journals, further rigorous development is essential. Specifically, formal proofs of the hypothesized k-regularity, a thorough comparative analysis of SC against existing complexity measures, and deeper investigation into the mathematical properties of the discovered "structural families" are needed. With

such advancements, the concepts would be well-suited for peer-reviewed publication.

B. Prioritized Actionable Steps for the Developer

To advance this research towards formal validation and publication, the following steps are recommended, in order of priority:

1. **Vigorously Pursue Proofs of 2-Regularity:** This is the most critical next step.
   - Focus on the sequences s1(n)=StructuralComplexity(n) and s2(n)=Depth(n).
   - Utilize tools like the IntegerSequences Mathematica package by Eric Rowland to compute terms of their 2-kernels (s(2en+r)) and to aid in guessing linear relations among kernel subsequences, which is the foundation for proving 2-regularity.[29]
   - Study the algebraic properties of these 2-kernels to determine if they are finitely generated over Q or Z.[28] Success here would be a landmark result for the project.
2. **Conduct Systematic Comparative Analysis of "Structural Complexity":**
   - Rigorously compare SC, both theoretically and empirically, with a range of established string complexity measures, including Lempel-Ziv complexity, Kolmogorov complexity, and various subword complexity definitions (e.g., pw(n)).[1]
   - Analyze the behavior of SC on benchmark sets of strings (e.g., Thue-Morse sequence, de Bruijn sequences, random strings, periodic strings) to delineate its unique contributions and mathematical properties (bounds, growth rates, etc.).[1]
3. **Investigate Properties of Generated "Structural Families":**
   - For the distinct "structural families" identified via clustering on the expanded feature vector, select representative families.
   - Empirically investigate whether their members share other non-obvious number-theoretic properties (e.g., related to divisibility, distribution of prime factors [26]) or combinatorial characteristics not explicitly captured by the classification vector.[1]
4. **Formalize and Extend the Generative Framework:**
   - Develop a formal grammatical or automata-theoretic description (e.g., using L-systems [22]) for the strings that are "maximally structured" according to the BSG algorithm (i.e., those achieving maximal depth or SC, often via Sk=A+A type rules).[1] Explore variations in seed strings and concatenation rules.

C. Concluding Remarks on the Concept's Potential

The Binary Split Game and its derived classification system, born from a simple recursive idea, have evolved into a creative and non-obvious framework for analyzing integers. While the system may not offer immediate solutions to long-standing, famous problems in number theory like primality testing, its genuine strength lies in providing a new, structurally-based perspective on numbers. This perspective has the clear potential to uncover previously unobserved patterns and relationships within the integers. The connection to k-regular sequences, if formally established, stands out as the most concrete and promising avenue for embedding these ideas within established mathematical theory, thereby validating their depth and opening doors for further fruitful research and publication. The journey from an intuitive game to a potentially robust mathematical construct is a compelling one, and the next steps of formal investigation will be crucial in realizing its full potential.

VII. References

- [1] An Algorithmic and Theoretical Analysis of Number Classification via Recursive Binary Splitting (Binary Split Game Analysis_.pdf)
- [1] Formalisation of the Binary Split Game and Derived Metrics (Binary Split Game Formalization.pdf)
- [1] Binary Split Game Integer Classification NotebookLM Output (binary-split-game-integer-classification-notebooklm-output.pdf)
- [2] Recursive binary string splitting algorithm (patterns.eecs.berkeley.edu)
- [5] Recursive binary string splitting algorithm (numberworld.org)
- [43] Binary string decomposition mathematics (angular.love)
- [44] Binary string decomposition mathematics (en.wikipedia.org)
- [12] Integer classification binary representation (en.wikipedia.org)
- [13] Integer classification binary representation (unacademy.com)
- [31] Structural complexity of binary strings (cs.purdue.edu)
- [14] Structural complexity of binary strings (en.wikipedia.org)
- [3] Recursive binary splitting string matching (researchgate.net)
- [4] Recursive binary splitting string matching (read.learnyard.com)
- [45] Iterated function on binary representation of integers (en.wikipedia.org)
- [46] Iterated function on binary representation of integers (preprints.org)

- [28] k-regular sequences tutorial (en.wikipedia.org)
- [47] k-regular sequences tutorial (stacks.math.columbia.edu)
- [48] Testing for 2-regularity of a sequence (facultyweb.kennesaw.edu)
- [49] Testing for 2-regularity of a sequence (projecteuclid.org)
- [30] Software for k-regular sequences (icms-conference.org)
- [50] Software for k-regular sequences (ccb.jhu.edu)
- [6] Lempel-Ziv complexity vs subword complexity (en.wikipedia.org)
- [7] Lempel-Ziv complexity vs subword complexity (biorxiv.org)
- [20] Properties of Thue-Morse sequence (cs.arizona.edu)
- [21] Properties of Thue-Morse sequence (cs.umb.edu)
- [22] L-systems and formal grammar in string generation (en.wikipedia.org)
- [23] L-systems and formal grammar in string generation (cs.unm.edu)
- [34] Dynamical systems on string spaces (scholarpedia.org)
- [51] Dynamical systems on string spaces (en.wikipedia.org)
- [24] Journal of Combinatorics on Words (cambridge.org)
- [25] Journal of Combinatorics on Words (en.wikipedia.org)
- [37] Theoretical Computer Science journal scope (slogix.in)
- [52] Theoretical Computer Science journal scope (frontiersin.org)
- [40] Experimental Mathematics journal scope (tandfonline.com)
- [41] Experimental Mathematics journal scope (en.wikipedia.org)
- [9] site:arxiv.org "recursive binary splitting" string (researchgate.net)
- [8] site:arxiv.org "recursive binary splitting" string (arxiv.org)
- [53] Eric Rowland IntegerSequences package documentation (ericrowland.github.io)
- [54] Eric Rowland IntegerSequences package documentation (iris.uniroma1.it)
- [17] Subword complexity of binary strings (numdam.org)
- [18] Subword complexity of binary strings (arxiv.org)
- [15] Kolmogorov complexity tutorial (faculty.cc.gatech.edu)
- [16] Kolmogorov complexity tutorial (users.cs.duke.edu)
- [32] L-systems formal grammar tutorial (cs.arizona.edu)
- [33] L-systems formal grammar tutorial (esolangs.org)
- [35] Symbolic dynamics tutorial (cirm-math.fr)
- [36] Symbolic dynamics tutorial (numberanalytics.com)
- [55] "IntegerSequences" package tutorial k-regular (semanticscholar.org)
- [56] "IntegerSequences" package tutorial k-regular (scribd.com)
- [57] Computational methods for testing k-regularity of sequences (pmc.ncbi.nlm.nih.gov)
- [58] Computational methods for testing k-regularity of sequences (journals.plos.org)
- [59] Subword complexity tutorial (d2l.ai)

- [19] Subword complexity tutorial (irif.fr)
- [60] site:arxiv.org "subword complexity" (arxiv.org)
- [61] site:arxiv.org "subword complexity" (arxiv.org)
- [29] "IntegerSequences" mathematica package examples k-regularity (ericrowland.github.io)
- [62] "IntegerSequences" mathematica package examples k-regularity (researchgate.net)
- [29] site:ericrowland.github.io "IntegerSequences" package (ericrowland.github.io)
- [63] site:ericrowland.github.io "IntegerSequences" package (ericrowland.github.io)
- [64] Mathematica code for testing k-regular sequences (wolfram.com)
- [28] Mathematica code for testing k-regular sequences (en.wikipedia.org)
- [65] Subword complexity of strings lecture notes (mdpi.com)
- [66] Subword complexity of strings lecture notes (cs.ucr.edu)
- [10] site:arxiv.org "binary string" "recursive decomposition" symmetry (arxiv.org)
- [11] site:arxiv.org "binary string" "recursive decomposition" symmetry (arxiv.org)
- [26] Online tool for prime factorization of a list of numbers (byjus.com)
- [27] Online tool for prime factorization of a list of numbers (lcm-calculator.com)
- [67] Integer sequence from binary representation of n (en.wikipedia.org)
- [68] Integer sequence from binary representation of n (geeksforgeeks.org)
- [69] Properties of numbers with high structural complexity in binary representation (wpmedia.wolfram.com)
- [70] Properties of numbers with high structural complexity in binary representation (stackoverflow.com)
- [55] Eric Rowland "IntegerSequences" package tutorial (semanticscholar.org)
- [71] Eric Rowland "IntegerSequences" package tutorial (researchgate.net)
- [24] Combinatorics on Words journal (cambridge.org)
- [25] Combinatorics on Words journal (en.wikipedia.org)
- [39] Discrete Mathematics journal (en.wikipedia.org)
- [39] Discrete Mathematics journal (en.wikipedia.org)
- [38] Theoretical Computer Science journal (en.wikipedia.org)
- [38] Theoretical Computer Science journal (en.wikipedia.org)
- [41] Experimental Mathematics journal (en.wikipedia.org)
- [42] Experimental Mathematics journal (amathr.org)
- [28] What is the formal definition of a k-regular sequence and what are its key properties? What is the k-kernel of a sequence? (en.wikipedia.org)
- [6] What is Lempel-Ziv complexity and how is it calculated? What is its relationship to other complexity measures? (en.wikipedia.org)
- [22] What are L-systems and how are they used to generate strings and fractal structures? What are the components of an L-system grammar?

(en.wikipedia.org)

- [34] What is symbolic dynamics and how is it used to study dynamical systems? What is a shift space? (scholarpedia.org)
- [53] Can you provide a tutorial or examples of how to use the 'IntegerSequences' Mathematica package to test if a sequence is k-regular? (ericrowland.github.io)
- [18] Are there any papers on arXiv that provide a good overview or tutorial on subword complexity of binary strings? (arxiv.org)
- [8] Are there any papers on arXiv that discuss recursive binary splitting of strings in a similar manner to the 'Binary Split Game'? (arxiv.org)
- [9] Are there any papers on arXiv that discuss Morfessor EM+Prune for subword segmentation? (researchgate.net)
- [55] Are there any examples or documentation on how to use the 'IntegerSequences' Mathematica package for k-regular sequences? (semanticscholar.org)
- [30] What are the main functionalities of the 'IntegerSequences' Mathematica package, especially concerning k-regular sequences? (icms-conference.org)
- [59] What are the different types of subword complexity and how are they calculated? (d2l.ai)
- [29] Can you provide more details on the practical steps of using the 'IntegerSequences' package to guess a k-regular recurrence for a sequence, based on the paper by Eric Rowland? (ericrowland.github.io)

Works cited

1. Binary Split Game Analysis_.pdf
2. Recursive Splitting - Our Pattern Language, accessed June 13, 2025, https://patterns.eecs.berkeley.edu/?page_id=217
3. Binary recursive partitioning: Background, methods, and application to psychology, accessed June 13, 2025, https://www.researchgate.net/publication/51064456_Binary_recursive_partitioning_Background_methods_and_application_to_psychology
4. Binary Search Algorithm Solution: Iterative & Recursive Ways - LearnYard, accessed June 13, 2025, https://read.learnyard.com/dsa/binary-search-algorithm-solution/
5. Binary Splitting - Numberworld.org, accessed June 13, 2025, https://www.numberworld.org/y-cruncher/internals/binary-splitting.html
6. Lempel–Ziv complexity - Wikipedia, accessed June 13, 2025, https://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv_complexity
7. The Lempel-Ziv complexity, represents the smallest number of distinct words W, needed to fully reconstruct the information in th - bioRxiv, accessed June 13, 2025, https://www.biorxiv.org/content/biorxiv/early/2025/01/07/2024.01.23.576807/DC1/e

mbed/media-1.pdf?download=true
8. Surveying the reach and maturity of machine learning and artificial ..., accessed June 13, 2025, https://arxiv.org/pdf/1912.02934
9. (PDF) Morfessor EM+Prune: Improved Subword Segmentation with ..., accessed June 13, 2025, https://www.researchgate.net/publication/339786861_Morfessor_EMPrune_Improved_Subword_Segmentation_with_Expectation_Maximization_and_Pruning
10. A Scheme of Cartan Decomposition for su(N) - arXiv, accessed June 13, 2025, https://arxiv.org/pdf/quant-ph/0603190
11. arXiv:2103.08056v4 [quant-ph] 14 Jun 2022, accessed June 13, 2025, https://arxiv.org/pdf/2103.08056
12. en.wikipedia.org, accessed June 13, 2025, https://en.wikipedia.org/wiki/Integer_(computer_science)#:~:text=Integers%20are%20commonly%20represented%20in,memory%20address%20as%20an%20integer.
13. All About Types of Integer Representation - Unacademy, accessed June 13, 2025, https://unacademy.com/content/nta-ugc/study-material/computer-science/types-of-integer-representation/
14. Kolmogorov complexity - Wikipedia, accessed June 13, 2025, https://en.wikipedia.org/wiki/Kolmogorov_complexity
15. Lecture 17: Kolmogorov Complexity 1 Introduction 2 Definition, accessed June 13, 2025, https://faculty.cc.gatech.edu/~ladha/S25/4510/L17.pdf
16. Kolmogorov complexity and its applications - Duke CS, accessed June 13, 2025, https://users.cs.duke.edu/~reif/courses/complectures/Li/KC-Lecture1.pdf
17. A note on constructing infinite binary words with polynomial subword complexity∗ - Numdam, accessed June 13, 2025, https://www.numdam.org/article/ITA_2013__47_2_195_0.pdf
18. The subword complexity of a class of infinite binary words - arXiv, accessed June 13, 2025, https://arxiv.org/pdf/math/0512256
19. Subword complexity and finite characteristic numbers - IRIF, accessed June 13, 2025, https://www.irif.fr/~steiner/num09/firicel.pdf
20. The Morse-Thue Sequence, accessed June 13, 2025, https://www2.cs.arizona.edu/patterns/weaving/webdocs/mo/H/MorseThue.pdf
21. Repetitions of Words and the Thue-Morse sequence, accessed June 13, 2025, https://www.cs.umb.edu/~offner/files/thue.pdf
22. L-system - Wikipedia, accessed June 13, 2025, https://en.wikipedia.org/wiki/L-system
23. L-System Rules - UNM CS, accessed June 13, 2025, https://www.cs.unm.edu/~joel/PaperFoldingFractal/L-system-rules.html
24. Combinatorics on Words 2nd Edition | Cambridge University Press & Assessment, accessed June 13, 2025, https://www.cambridge.org/py/universitypress/subjects/mathematics/discrete-mathematics-information-theory-and-coding/combinatorics-words-2nd-edition?site_view=desktop
25. Combinatorics on words - Wikipedia, accessed June 13, 2025,

https://en.wikipedia.org/wiki/Combinatorics_on_words
26. Prime Factors Calculator - Free Online Calculator - BYJU'S, accessed June 13, 2025, https://byjus.com/prime-factors-calculator/
27. Prime Factorization Calculator, accessed June 13, 2025, https://www.lcm-calculator.com/prime-factorization-calculator
28. k-regular sequence - Wikipedia, accessed June 13, 2025, https://en.wikipedia.org/wiki/K-regular_sequence
29. Eric Rowland - IntegerSequences: a package for computing with k ..., accessed June 13, 2025, https://ericrowland.github.io/papers/IntegerSequences--_a_package_for_computing_with_k-regular_sequences.pdf
30. ICMS 2018 - Session 13: Symbolic Combinatorics · ICMS ..., accessed June 13, 2025, https://icms-conference.org/2018/sessions/session13/
31. Structural Complexity of Random Binary Trees - Computer Science Purdue, accessed June 13, 2025, https://www.cs.purdue.edu/homes/spa/papers/isit09-tree.pdf
32. L-Systems 1, accessed June 13, 2025, https://www.cs.arizona.edu/patterns/weaving/webdocs/mo/J/L-Systems1.pdf
33. L-system - Esolang, accessed June 13, 2025, https://esolangs.org/wiki/L-system
34. Symbolic dynamics - Scholarpedia, accessed June 13, 2025, http://www.scholarpedia.org/article/Symbolic_dynamics
35. Symbolic dynamics and representations - CIRM, accessed June 13, 2025, https://www.cirm-math.fr/ProgWeebly/Renc1584/NotesBerthe.pdf
36. Unlocking Symbolic Dynamics - Number Analytics, accessed June 13, 2025, https://www.numberanalytics.com/blog/ultimate-guide-symbolic-dynamics
37. Theoretical Computer Science - Elsevier - Impact Factor | S-Logix, accessed June 13, 2025, https://slogix.in/research/journals/theoretical-computer-science/
38. en.wikipedia.org, accessed June 13, 2025, https://en.wikipedia.org/wiki/Theoretical_Computer_Science_(journal)
39. en.wikipedia.org, accessed June 13, 2025, https://en.wikipedia.org/wiki/Discrete_Mathematics_(journal)
40. Learn about Experimental Mathematics - Taylor & Francis Online: Peer-reviewed Journals, accessed June 13, 2025, https://www.tandfonline.com/journals/uexm20/about-this-journal
41. en.wikipedia.org, accessed June 13, 2025, https://en.wikipedia.org/wiki/Experimental_Mathematics_(journal)
42. Journals – AMR - Association for Mathematical Research, accessed June 13, 2025, https://amathr.org/publications/journals/
43. The simple math behind decimal-binary conversion algorithms - Angular.love, accessed June 13, 2025, https://angular.love/the-simple-math-behind-decimal-binary-conversion-algorithms/
44. Binary number - Wikipedia, accessed June 13, 2025, https://en.wikipedia.org/wiki/Binary_number
45. Iterated binary operation - Wikipedia, accessed June 13, 2025,

https://en.wikipedia.org/wiki/Iterated_binary_operation
46. Binary Representation of Natural Numbers and Collatz Conjecture - Preprints.org, accessed June 13, 2025, https://www.preprints.org/manuscript/202502.1743/v1
47. Section 10.68 (0AUH): Regular sequences—The Stacks project, accessed June 13, 2025, https://stacks.math.columbia.edu/tag/0AUH
48. Lecture 7: Regular graphs 1 Degree sequences and the graphic sequence problem 2 Regular graphs - Faculty Web Pages - Kennesaw State University, accessed June 13, 2025, https://facultyweb.kennesaw.edu/mlavrov/courses/graph-theory/lecture7.pdf
49. SUPER-REGULAR SEQUENCES - Project Euclid, accessed June 13, 2025, https://projecteuclid.org/journals/pacific-journal-of-mathematics/volume-84/issue-2/Superregular-sequences/pjm/1102784222.pdf
50. Kraken - CCB at JHU - Johns Hopkins University, accessed June 13, 2025, https://ccb.jhu.edu/software/kraken/
51. String theory - Wikipedia, accessed June 13, 2025, https://en.wikipedia.org/wiki/String_theory
52. Theoretical Computer Science - Frontiers, accessed June 13, 2025, https://www.frontiersin.org/journals/computer-science/sections/theoretical-computer-science/about
53. The Hierarchy of Integer Sequences Eric Rowland, accessed June 13, 2025, https://ericrowland.github.io/The_Hierarchy_of_Integer_Sequences.pdf
54. Integer Sequences in Cryptography: A New Generalized Family and its Application - IRIS, accessed June 13, 2025, https://iris.uniroma1.it/retrieve/0b2e5a82-d995-46aa-b20a-defe01f13654/Tesi_dottorato_Raso.pdf
55. Algebraic power series and their automatic complexity I: finite fields - Semantic Scholar, accessed June 13, 2025, https://www.semanticscholar.org/paper/48f7b5185ee37cd414febd10df74cb73868a0787
56. Ic_1 | PDF | Dynamic Programming | Algorithms And Data Structures - Scribd, accessed June 13, 2025, https://www.scribd.com/document/852486656/Ic-1
57. Alignment-Free Sequence Analysis and Applications - PMC, accessed June 13, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC6905628/
58. Cross-species regulatory sequence activity prediction | PLOS Computational Biology, accessed June 13, 2025, https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008050
59. 15.6. Subword Embedding — Dive into Deep Learning 1.0.3 ..., accessed June 13, 2025, https://d2l.ai/chapter_natural-language-processing-pretraining/subword-embedding.html
60. [1801.05376] Subword complexity and power avoidance - arXiv, accessed June 13, 2025, https://arxiv.org/abs/1801.05376
61. [1406.3974] Subword complexity and decomposition of the set of factors - arXiv, accessed June 13, 2025, https://arxiv.org/abs/1406.3974
62. Eric ROWLAND | Hofstra University, Hempstead | Department of Mathematics |

Research profile - ResearchGate, accessed June 13, 2025, https://www.researchgate.net/profile/Eric-Rowland
63. Packages - Eric Rowland, accessed June 13, 2025, https://ericrowland.github.io/packages.html
64. Sequences, Sums, Series | Mathematica & Wolfram Language for Math Students—Fast Intro, accessed June 13, 2025, https://www.wolfram.com/language/fast-introduction-for-math-students/en/sequences-sums-and-series/
65. Asymptotic Analysis of the kth Subword Complexity - MDPI, accessed June 13, 2025, https://www.mdpi.com/1099-4300/22/2/207
66. On average sequence complexity - Computer Science and Engineering, accessed June 13, 2025, https://www.cs.ucr.edu/~stelo/papers/tcs04.pdf
67. List of integer sequences - Wikipedia, accessed June 13, 2025, https://en.wikipedia.org/wiki/List_of_integer_sequences
68. Generate all the binary strings of N bits - GeeksforGeeks, accessed June 13, 2025, https://www.geeksforgeeks.org/generate-all-the-binary-strings-of-n-bits/
69. Regularity versus Complexity in the Binary Representation of 3^n - Wolfram, accessed June 13, 2025, https://wpmedia.wolfram.com/sites/13/2018/02/18-3-6.pdf
70. Space Complexity of Storing a Binary Representation of an Integer - Stack Overflow, accessed June 13, 2025, https://stackoverflow.com/questions/79178998/space-complexity-of-storing-a-binary-representation-of-an-integer
71. Review of the article “On the 2-adic valuation of differences of harmonic numbers” by Tamás Lengyel - ResearchGate, accessed June 13, 2025, https://www.researchgate.net/publication/387067818_Review_of_the_article_On_the_2-adic_valuation_of_differences_of_harmonic_numbers_by_Tamas_Lengyel