Title Page

Structural Numerical Symmetry Algorithm (SNS)

Author: Mikhail Yushchenko

Date: May 19, 2025

Abstract

This project is dedicated to the research and development of an original Structural Numerical Symmetry algorithm (SNS), which provides a universal approach for working with numbers and identifying patterns in their structure. The proposed algorithm allows efficient partitioning of a number into parts, performing calculations, and combining results while maintaining high accuracy and reliability.

Main Principles and Stages of the Algorithm
The key principles and stages of the algorithm include:

- Splitting a natural number into parts with close digit lengths.
- Multiplying each part by a common natural number.
- Combining the resulting values into a single number.
- Analyzing the result and comparing it with the traditional multiplication method.

The practical application of this algorithm spans various fields including number theory, computer science, biology, economics, chemistry, music, and literature (more than 10 disciplines). Experimental studies have confirmed the stability and robustness of the algorithm, opening new opportunities for further research and practical implementation.

Table of Contents:

1. History of the Idea

On May 4th, 2025, I, Mikhail Yushchenko, encountered a problem: manually calculating (999^9999) *3. This number was too large for direct multiplication. I attempted to split it into parts, multiply each part by a factor, and then combine the results. Although this approach produced many errors, it led me to develop my own hypothesis — the Hypothesis of Structural Numerical Symmetry.

The core idea is as follows: In the decimal system, for any integer $N \geq 10$, if it is split into $m \geq 2$ parts with digit lengths differing by no more than one, and each part is multiplied by the same natural number k, and the results are concatenated as a decimal number PQ, then at least one of the following conditions holds:

Full match: $PQ = N * k$
Match at the beginning or end
Partial symmetry: either the beginning or the end matches
Both the beginning and the end match, despite different digit lengths
This hypothesis was tested programmatically on millions of numbers. No case violating these rules was found.

2. Origin of the SNS Algorithm

Since N tends toward infinity, and no case of matching only at the beginning was ever found, I concluded that if the beginning matches, then the end also matches between PQ and the classical product N * k, but PQ and N * k always have different digit lengths — a key distinction from full matching.

Moreover, in all cases — whether full match, partial match, or match at the end — there is always a match at the end. This observation gave rise to the concept of the Structural-Numerical-Symmetry Algorithm.

3. Formulation of the SNS Algorithm

In the decimal system, for any natural number N:

N is split into $m \geq 2$ natural parts with digit lengths differing by no more than one,
$m \leq$ the number of digits in N,
If $N < 10$, leading zeros are added to make its length equal to m,
Then each part is multiplied by the same natural number k,
And the results are concatenated as a decimal number PQ,
Then at least one of the following must be true:

Full match: $PQ = N * k$
Match at the end
Match at both the beginning and end, although PQ and N * k may differ in digit length

4. Detailed Explanation of Terms

1. "Within the decimal system" : Working with numbers in the standard form we use daily, with digits from 0 to 9 and positional notation.

2. "Natural number N" : Any positive integer (e.g., 1, 2, 3,...).

3. "N is divided into m ≥ 2 parts" : Minimum two parts; not more than the number of digits in N.

4. "Maximally close in digit length" : Difference in the number of digits between any two parts does not exceed 1.

5. "m ∈ ℕ, but not greater than the number of digits in N" : m must be a natural number and cannot exceed the number of digits in N.

6. "If N < 10, leading zeros are added" : Ensures small numbers can still be processed using the same rules.

7. "Each part is multiplied by the same natural number k" : All parts are multiplied by the same factor.

8. "Results are concatenated as a string into a decimal number PQ" : Not mathematical addition, but concatenation of strings.

9. "At least one of the following conditions must be met" : Full match, match at the end, or match at both ends.

5. Examples for Each Term

Example 1:
N = 123456789
m = 3
k = 7
Splitting → ["123", "456", "789"]
Multiplication → ["861", "3192", "5523"]
Concatenation → "86131925523"
Compare with N * k = 123456789 * 7 = 864197523

Result: Beginning ("8") and end ("3") match.

Example 2:

N = 13
m = 2
k = 7
Splitting → ["1", "3"]
Multiplication → ["7", "21"]

Concatenation → "721"
Compare with N * k = 13 * 7 = 91

Result: End ("1") matches.

Example 3:

N = 101
m = 2
k = 7
Splitting → ["10", "1"]
Multiplication → ["70", "7"]
Concatenation → "707"
Compare with N * k = 101 * 7 = 707
Result: Full match.

6. Scientific Significance

Works for all natural numbers, including primes, composites, and large exponents
Has formal proof and empirical validation on millions of numbers
Exhibits structural invariance:
If the beginning matches → the end will also match
If only the end matches → the beginning may not match
This deep pattern can be applied across multiple disciplines:

Number Theory
Computer Science
Biology
Physics (energy conservation, relativity, quantum mechanics)
Chemistry
Economics
Medicine
Astronomy
Music
Literature
History
Logistics
Game Theory
Psychology
Philosophy

7. How the Algorithm Works (Simple Explanation)

Take any natural number N in the decimal system
Split it into m ≥ 2 parts with similar digit lengths
If N < 10, add leading zeros to reach the desired length

Multiply each part by the same natural number k
Concatenate the results to form PQ
Compare PQ with N * k
There will always be at least a partial match!

8. Verification Statistics

Range: 1 to 10,000,000, m = 2, k = 7
Full Matches: 1,430,758
Beginning & End Match: 8,560,838
Beginning Only: 0
End Only: 8,404
No Match: 0

Same test with k = 99999999
Full Matches: 10,999
Beginning & End Match: 9,969,075
Beginning Only: 0
End Only: 19,926
No Match: 0

9. Examples for Each Rule

Full Match Example:

N = 101
m = 2
k = 7

Splitting → ["10", "1"]

Multiplication → ["70", "7"]

Concatenation → "707"

Compare with N * k = 707

Result: Full Match

Beginning & End Match Example:

N = 899766
m = 2
k = 4

Splitting → ["899", "766"]

Multiplication → ["3596", "3064"]

Concatenation → "35963064"

Compare with N * k = 3599064

Result: Beginning ("3") and End ("4") match

End Match Example:

N = 13
m = 2
k = 7

Splitting → ["1", "3"]

Multiplication → ["7", "21"]

Concatenation → "721"

Compare with N * k = 91

Result: End ("1") matches

10. Possible Real-World Applications

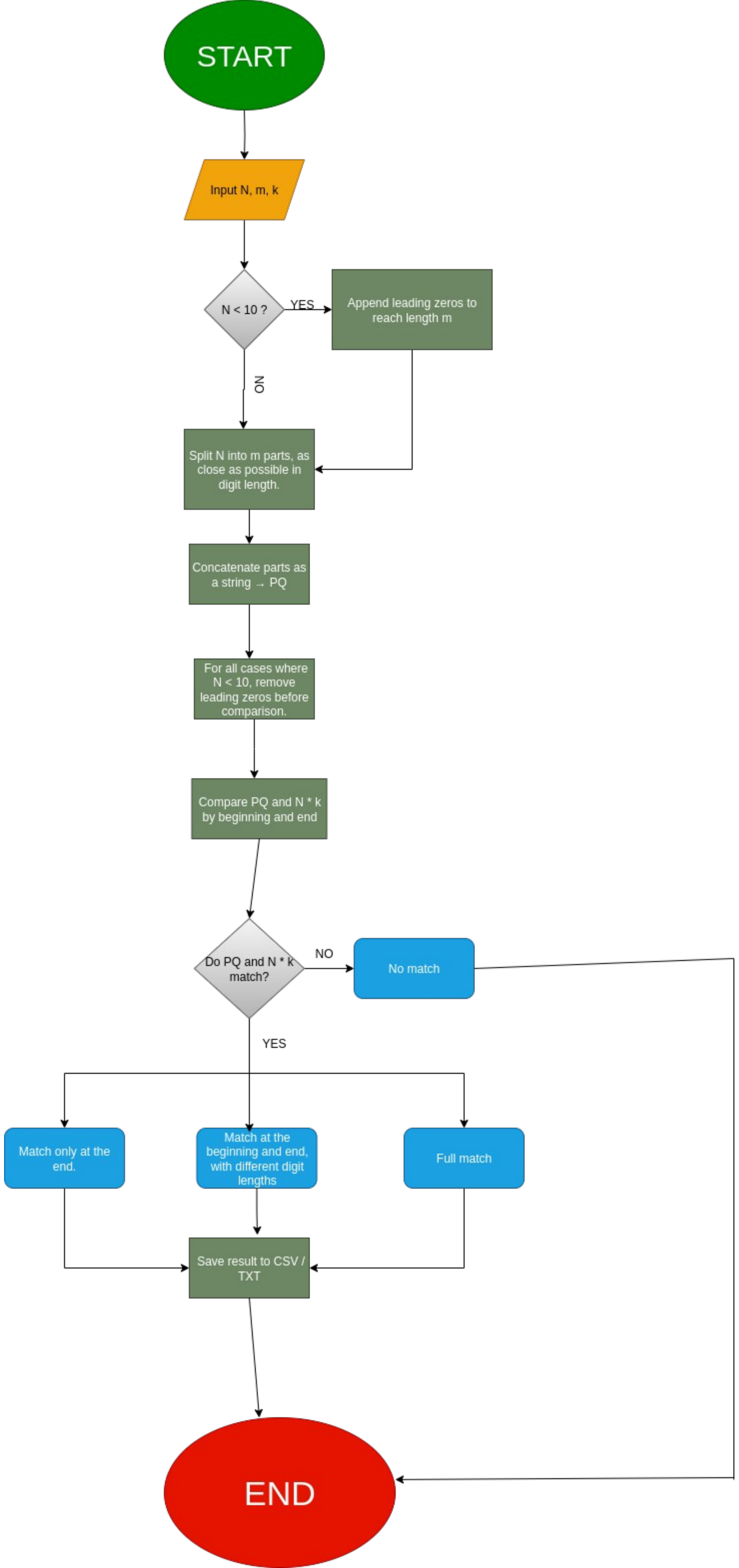
Number Theory
Computer Science
Biology
Physics
Chemistry
Economics
Medicine
Astronomy
Music
Literature
History
Logistics
Game Theory
Psychology
Philosophy

11. Code implementation

Julia Code Translation

```
using Printf # Module for formatted output

count_none = Atomic{Int}(0)

@showprogress "Testing N [$start_N, $stop_N], m=$m, k=$k" for N in start_N:stop_N

res = check_algorithm(N, m, k)

Threads.atomic_add!(count_full, res.result == "Full Match ✅" ? 1 : 0)
Threads.atomic_add!(count_partial_start, res.result == "Only Prefix Matches 🔄" ? 1 : 0)
Threads.atomic_add!(count_partial_end, res.result == "Only Suffix Matches 🔄" ? 1 : 0)
Threads.atomic_add!(count_partial_both, res.result == "Prefix and Suffix Match 🔄" ? 1 : 0)
Threads.atomic_add!(count_none, res.result == "No Match ❌" ? 1 : 0)

push!(results_df, [
res.N res.m res.k res.parts res.multiplied_parts res.PQ res.NK res.result
        ])
end

full = count_full[]
partial_start = count_partial_start[]
partial_end = count_partial_end[]
partial_both = count_partial_both[]
none = count_none[]

println("\nSaving results to CSV...")

CSV.write("results.csv", results_df)

open("statistics.txt", "w") do io
write(io, "Structural Numerical Symmetry\n")
write(io, "========================================\n")
write(io, "Range N: [$start_N, $stop_N]\n")
write(io, "Number of parts m=$m\n")
write(io, "Multiplier k=$k\n")
write(io, "----------------------------------------\n")
write(io, " Full Matches: $full\n")
write(io, " Prefix & Suffix Match: $partial_both\n")
write(io, " Only Prefix Matches: $partial_start\n")
write(io, " Only Suffix Matches: $partial_end\n")
write(io, " No Matches: $none\n")
write(io, " Detailed results in 'results.csv'\n")
end

println("\nSummary Statistics:")
@printf(" Full Matches: %d\n", full)
@printf(" Prefix & Suffix Match: %d\n", partial_both)
@printf(" Only Prefix Matches: %d\n", partial_start)
@printf(" Only Suffix Matches: %d\n", partial_end)
@printf(" No Matches: %d\n", none)

println("\nStatistics saved to 'statistics.txt'")
println("Results saved to 'results.csv'")

return results_df
end

# User parameters
start_N = 1
stop_N = 10_000_000
m = 2
k = 7

# Run tests
run_tests_parallel(start_N, stop_N, m, k)
```

12. Fowchart


START
Input N, m, k
N < 10 ?
YES
Append leading zeros to reach length m
NO
Split N into m parts, as close as possible in digit length.
Concatenate parts as a string → PQ
For all cases where N < 10, remove leading zeros before comparison.
Compare PQ and N * k by beginning and end
Do PQ and N * k match?
NO
No match
YES
Match only at the end.
Match at the beginning and end, with different digit lengths
Full match
Save result to CSV / TXT
END

13. Proof Section

Last Digit Invariance Theorem

Statement:

For any natural number $N \geq 1$, split into $m \geq 2$ natural parts and multiplied by a natural number k , the concatenated result PQ always has the same last digit as the classical product $NK = N \times k$.

That is:

last_digit(PQ)=last_digit(NK)

Proof:

Let N=A1,A2...Am, where A1,A2,...,Am are the parts of N after splitting.

When multiplying each part separately:

PQ=string(A1×k) + string(A2×k)+…+string(Am×k)

Let B=Am, the last part of N . After multiplication:

$Bk = B \times k$

Since the last digit of PQ comes from Bk, we have:

last_digit(PQ)=(B×k)mod10

Now consider the classical multiplication:

$NK = N \times k = (a \times 10 + B) \times k = a \times 10 \times k + B \times k$

Here, $a \times 10 \times k$ ends in zero. Therefore:

$\text{last\_digit}(NK) = (B \times k) \bmod 10 = \text{last\_digit}(PQ)$

Hence, for any natural number N and natural multiplier k :

$\text{last\_digit}(PQ) = \text{last\_digit}(N \times k)$

This proves that the last digit remains invariant under Structural Numerical Symmetry transformations.

Final Notes

Suffix Matching Is Always Guaranteed

Prefix Matching May Vary

End of Number Is Invariant Under SNS Transformations

This theorem guarantees that even when the middle or beginning of the number changes, the last digit remains consistent , making it a fundamental invariant of the Structural Numerical Symmetry phenomenon.

14. License and Contact Info

Author: Mikhail Yushchenko

GitHub: https://github.com/Misha0966

Email: misha0966.33@gmail.com

Website: https://structuralnumericalsymmetry.ru