Structural-Numerical-Symmetry-Algorithm-Mikhail-Yushchenko-2025

Алгоритм структурной числовой симметрии(СЧС)

Титульная страница

Автор: Ющенко Михаил Юрьевич

Дата: 19.05.2025 год

Аннотация

Настоящий проект посвящен исследованию и разработке оригинального алгоритма структурной числовой симметрии (СЧС), который представляет собой универсальный подход к оперированию числами и выявлению закономерностей в их структуре. Предлагаемый алгоритм позволяет проводить эффективное деление числа на блоки, производить расчеты и объединять результаты, обеспечивая высокую степень точности и надежности.

Основные положения и этапы алгоритма включают:

Разбиение натурального числа на блоки с соблюдением близости по разрядности.

Последующее локальное умножение каждого блока на единое натуральное число.

Объединение полученных результатов в единую конструкцию.

Анализ полученной величины и сравнение с традиционным методом умножения.

Практическое применение данного алгоритма возможно в различных областях, включая теорию чисел, информатику, биологию, экономику, химию, музыку и литературу (всего свыше 10 дисциплин). Проведенные экспериментальные исследования подтвердили стабильность и устойчивость алгоритма, что открывает новые перспективы для дальнейших исследований и внедрения в практику.

Содержание:

---

1. История возникновения идеи

Как родилась гипотеза о СЧС (Структурной Числовой Симментрии),которая позже стала **явлением** в связи её доказательством :

В 2025 году, 4 мая, я, Ющенко Михаил Юрьевич столкнулся с задачей:

Вычислить вручную (999 ^ 9999) * 3. Это число оказалось слишком большим для прямого умножения. Тогда я попробовал разбить его на блоки, умножить каждый блок на коэффициент, а потом объединить результаты. Но данный подход выдал колоссальное кол-во ошибок, но я продолжал вытаскивать из этого числа всё новые и новые числа, эти все неудачные попытки, привели меня к выведению собственной гипотезы. Так родилась гипотеза структурной числовой симметрии, суть которой заключается в следующем:

В рамках десятичной системы счислений, для любого целого числа $N \geq 10$, если его разбить на $m \geq 2$ блоков, максимально близких по разрядности, чтобы количество разрядов всех блоков отличались не более чем на единицу, при этом $m \in N$, но не больше разрядности числа N, умножить каждый блок на одно и то же натуральное число k, а затем объединить результаты как десятичное число PQ, то:

Возможно полное совпадение: PQ = N * k

Может совпадать начало или конец

Может наблюдаться частичная симметрия: совпадает начало или конец

Совпадают и начало, и конец, при разных разрядностях

Вопрос: Может ли существовать такое число N, для которого ни одно из правил не выполняется?

Она была проверена программно на миллионах чисел. см. файл: Структурная_числовая_симметрия.jl Ни одного случая без совпадений найдено не было.

Потом я задался вопросом, а что если расширить данную гипотезу для всех натуральных чисел? т.е. для тех, при которых $N<10$, то тогда я получил более обобщённую формулировку:

В рамках десятичной системы счисления, для любого натурального числа N, если: N разбивается на m≥2 натуральных блоков, максимально близких по разрядности (разница в количестве разрядов между любыми двумя блоками не превышает единицу), m∈N, но m при этом не должно превышать количество разрядов числа N, Если N<10, то перед числом дописываются ведущие нули , чтобы его длина стала равна m. Затем каждый блок умножается на одно и то же натуральное число k. Результаты объединяются как конкатенация в десятичное число PQ. Для чисел N<10, при сравнени с классическим умножением, удаляются ведущие нули. Тогда выполняется хотя бы одно из следующих условий: Полное совпадение : PQ = N * k. Совпадает начало или конец. Совпадают и начало, и конец , но при этом PQ и N * k имеют разную разрядность.

Вопрос: Может ли существовать такое число N, для которого ни одно из правил не выполняется?

Ответ: Нет, такого числа нет! [См.доказательство]

Таким образом, гипотеза стала **явлением**.

2. История возникновения алгоритма для Структурной числовой симметрии.

Ну так, поскольку само N стремиться к бесконечности, а помимо всего прочего совпадения только по началу так и не было найдено, но зато я пришёл к выводу, что если совпадает начало то так же совпадает конец между оператором PQ и классическим произведением N * k, при этом PQ и N * k обязательно имеют разные разрядности, это ключевое отличие от полного совпадения. Но при этом обратной зависимости нет. А ещё одно явление - во всех случаях, будь то полное совпадение или совпадение по началу и концу, ну а так же совпадение только по концу, наблюдается следующее явление, а именно, везде и всегда совпадение по концу, так возникла идея о алгоритме для Структурной числовой симметрии.

3. Сам алгоритм:

Алгоритм Структурной Численной Симметрии, который работает, следующем образом: В рамках десятичной системы счисления, для любого натурального числа N, если: N разбивается на m≥2 натуральных блоков, максимально близких по разрядности (разница в количестве разрядов между любыми двумя блоками не превышает единицу), $m \in N$, но не больше разрядности самого числа N. Если $N<10$, то перед числом дописываются ведущие нули , чтобы его длина стала равна m. Затем каждый блок умножается на одно и то же натуральное число k, результаты объединяются как в десятичное число PQ. Для чисел $N<10$, при сравнени с классическим умножением, удаляются ведущие нули. Тогда обязательно выполняется одно из следующих условий: полное совпадение : PQ = N * k, совпадает конец, совпадают и начало и конец, но при этом PQ и N * k имеют разную разрядность.

4. Подробные пояснения к терминам:

Описание терминов:

- Что такое СЧС ?

Это метод представляющий собой оригинальный подход в теории чисел, направленный на поиск закономерностей и универсальное описание структуры чисел через их разбиение на блоки и выявление симметрических свойств.

- Что N ?

N - любое натуральное число

- Что такое k ?

k - абсолютно любой натуральный коофициент умножения

- Что такое m ?

m - кол-во блоков, на которое разбивается число N

- Что такое PQ?

PQ - это цифровая композиция результатов умножения блоков исходного числа, а не арифметическая сумма. Эта конструкция лежит в основе анализа цифровой устойчивости и структурных совпадений в этой работе

- Что такое Nk?

Nk - результат классического умножения N на k (он же N * k)

- Сколько чисел должно совпасть?

Как минимум по 1 числу, или полностью всё число целоком.

- Как именно разбивать?

Разбивать **рекомендуется** слева направо, подобно позиционной записи числа в десятичной системе, НО! Вы можете разбивать как угодно, только **(!)** в **соответствии с правилами СЧС.**

1.«В рамках десятичной системы счисления».

Это значит, что мы работаем с числами так, как они записаны в привычной нам форме : от 0 до 9, слева направо, с учётом позиции цифры.

- 2.«Натуральное число

N». Это означает:

N — любое положительное целое число: 1, 2, 3, ..., 100, ..., 1000000.

Не требуется, чтобы оно было простым, чётным или большим.

- 3.«N разбивается на $m \geq 2$

частей». Это говорит о том, что:

Мы не можем разбить число на 1 блок — минимальное количество блоков: 2.

Можно разбивать на 2, 3, 4... , но не больше, чем количество разрядов в числе N.

- 4.«Максимально близкие по

разрядности». Это значит:

При разбиении все блоки должны быть очень похожи по разрядности.

Разница в количестве разрязрядах между блоками не превышает 1.

- 5.«$m \in \mathbb{N}$, но не больше разрядности самого числа

N». То есть:

- Если N = 1234 (4 разряда), то m может быть: 2, 3, 4.

Нельзя разбивать на 5 блоков, потому что в числе всего 4 разряда.

- 6.«Если N < 10, то перед числом дописываются ведущие нули, чтобы его длина стала равна m».

Это нужно, чтобы малые числа тоже можно было обработать по тем же правилам, что и большие.

- 7.«Каждый блок умножается на одно и то же натуральное число

k». Значит:

Все блоки умножаются на один и тот же множитель.

Множитель может быть любым натуральным числом: 1, 2, 3, ...

- 8.«Результаты объединяются как конкатенация в десятичное число

PQ». Это означает:

Умноженные блоки соединяются как конкатенация , а не как математическая сумма.

Это не просто математическое умножение , а структурное преобразование.

- 9.«Тогда обязательно выполняется одно из следующих

условий». После всех действий всегда:

Совпадает всё число целиком.

Или конец.

Или и начало, и конец , даже если разрядность чисел разная.

Этот принцип пока ни разу не был нарушен за миллионы проверок!

5. Примеры по каждому пояснению из терминов:

1.) Это значит:

Мы работаем с числами так, как мы привыкли — в привычной форме.

Числа записываются цифрами от 0 до 9.

Позиция цифры важна (например, 123 ≠ 321, потому что позиции разные).

Пример:

- N = 123456789
- m = 3
- k = 7

разбиение → ["123", "456", "789"]

Поблоковое умножение: → ["861", "3192", "5523"] **конкатенация** → PQ = "86131925523"

сравнение чисел:

N * K = 864197523 и PQ = 86131925523

Совпадают начало (8) и конец (3)

2.) Не нужно, чтобы число было простым или чётным.

Подходит любое положительное целое число.

Пример:

- N = 13
- m = 2
- k = 7

разбиение → ["1", "3"]

поблоковое умножение: → ["7", "21"] **конкатенация** → PQ = "721"

N * K = 91

сравнение чисел:

- PQ = 721
- N * k = 91

Совпадает только конец (1)

3.) Минимум на 2

Можно больше: 3, 4, 5... но не больше кол-ва разрядов числа

Пример:

- N = 101
- m = 2
- k = 7

Разбиение → ["10", "1"]

поблоковое умножение: → ["70", "7"] **конкатенация:** → PQ = "707"

Сравнием:

N * K = 707 и PQ = 707

Полное совпадение.

4.) Все имеют очень похожую разрядность.

Разница между блоками не больше одной 1 разряда.

Пример:

- N = 12345
- m = 3

конкатенация: числа (N, m) → ["12", "34", "5"]

здесь : 2, 2, 1 → разница не превышает единицу.

Условие выполнено.

5.) **Если число имеет 4 разряда, то нельзя делить на 5 блоков.**

Можно делить на 2, на 3 и 4 .

Пример:

N = 1234

кол-во разрядов(N) = 4

- m = 5 → Недопустимое значение.
- m = 4 → Допустимое значение.
- m = 3 → Допустимое значение
- m = 2 → Допустимое значение.

6.) Так можно использовать малые числа в полной системе проверки.

Пример:

- N = 3
- k = 7
- m = 2
- → разрядность(N) = 1 → дополняем до 2 разрядов → "03"

разбиение → ["0", "3"]

умножение частей → ["0×7=0", "3×7=21"] **поблоковое умножение:** → PQ = "021"

- N * K = 21
- **После очистки от нулей**: PQ = 21 → N * k = 21 → Полное совпадение

7.) **Это важно для сохранения структуры.**

где k -натуральное число , может быть любым: 1, 2, 3, ..., 99999999

Пример:

- N = 1234
- m = 2
- k = 4

разбиение → ["12", "34"]

поблоковое умножение: → ["48", "136"] **конкатенация** → PQ = "48136"

N * K = 1234 × 4 = 4936

PQ = 48136

сравнием PQ и N * k

Совпадают начало ("4") и конец ("6")

8.) **Это не просто математическое умножение.**

Это цифровое преобразование : умножаются, затем соединяются как конкатенация.

- N = 899766
- m = 2
- k = 4

разбиение → ["899", "766"]

поблоковое умножение: → ["3596", "3064"] **конкатенация** → PQ = "35963064"

сравнение PQ и N * k

PQ = 35963064

N * K = 899766 × 4 = 3599064

Совпадают начало ("3") и конец ("4")

9.) Всегда будет совпадать:

Полное совпадение

Только конец

И начало, и конец

Ни одного случая без совпадений не найдено

Пример:

- N = 11
- m = 2
- k = 7

разбиение → ["1", "1"]

поблоковое умножение: → ["7", "7"] **конкатенация:** → PQ = "77"

- PQ = 77
- N * K = 77
- **Сравнием** PQ с N * k
- Полное совпадение!

6. Научная значимость.

- Алгоритм **работает для всех натуральных чисел**, включая простые, составные, большие степени
- Для него есть **формальное доказательство**, и к тому же он **эмпирически проверен на миллионах чисел**
- Он имеет **структурную инвариантность**:

- Если совпадает начало → **обязательно совпадёт и конец**
- Но если совпадает только конец → начало **может не совпадать**
- Доказательство, постольку он был выведен из гипотезы, а в данный момент **явления** СЧС (Структурной числовой семмитрии) , то и доказательство его лежит в рамках СЧС См.доказательство

Это говорит о **глубокой закономерности**, которая может быть использована в:

- Теории чисел
- Информатике
- Биологии
- Физике
- Химии
- Экономике
- Медицине
- Астрономии
- Музыке
- Литературе
- Истории
- Логистики
- Теории игр
- Психологии
- Философии
- и в других областях

7. Принцип работы алгоритма

Как это работает? (Простыми словами)

- 1.В рамках десятичной системы счисления, берём любое натуральное число N

- 2.Разбиваем его на $m \geq 2$ натуральных натуральных блоков, близких по разрядности

- 3.Если число маленькое ($N < 10$), дополняем его нулями до нужной длины

- 4.Каждый блок умножаем на натуральное число k

- 5.Результаты соединяем как конкатенацию: → получаем `PQ`
- 6.При сравнении для чисел N<10 с классическим умножением, удаляем ведущие нули.

- 7.Сравниваем `PQ` с классическим произведением `NK = N × k`
- 8.Всегда будет хотя бы частичное

совпадение!

Примеры работы алгоритма:

N	m	k	Разбиение	PQ	NK	Результат
101	2	7	["10", "1"]	"707"	"707"	Полное совпадение
135	2	7	["13", "5"]	"9135"	"945"	Совпадают начало и конец
13	2	7	["1", "3"]	"721"	"91"	Совпадает только конец
1234	2	4	["12", "34"]	"48136"	"4936"	Совпадают начало и конец
1001	2	7	["10", "01"]	"7007"	"7007"	Полное совпадение

Все эти примеры показывают, что **начало и конец чисел сохраняют связь**, даже если середина меняется.

8. Статистика проверок

Диапазон: от 1 до 10000000, m=2, k=7

✓ Полных совпадений: 1430758

🔄 Совпадают начало и конец: 8560838

🔄 Совпадает только начало: 0

🔄 Совпадает только конец: 8404

❌ Без совпадений: 0 [Скачать данные]

То же самое, но при k=99999999

📊 Сводная статистика:

✅ Полных совпадений: 10999

🔄 Совпадают начало и конец: 9969075

🔄 Совпадает только начало: 0

🔄 Совпадает только конец: 19926

❌ Без совпадений: 0

[Скачать данные]

9. Примеры по каждому правилу.

Полное совпадение:

- N = 101
- m = 2
- k = 7

разбиение → ["10", "1"]

поблоковое умножение: → ["70", "7"] **конкатенация:** → PQ = 707

Сравнение чисел

NK = 101 × 7 = 707 и PQ = 707

Результат: Полное совпадение

Совпадают начало и конец:

- N = 899766
- m = 2

- k = 4

разбиение → ["899", "766"]

поблоковое умножение: → ["3596", "3064"] **конкатенация** → PQ = 35963064

сравнием

NK = 899766 × 4 = 3599064 и PQ = 35963064

- Совпадают: "3" и "4"
- Совпадают начало и конец

Совпадает только конец:

- N = 13
- m = 2
- k = 7

разбиение → ["1", "3"]

поблочное умножение → ["7", "21"] **конкатенация:** → PQ = "721"

сравнием

N * K = 13 × 7 = 91 и PQ = 721

Совпадает только конец ("1")

10. Возможные(но не очевидные) применения в реальных задачах

11. Кодовая реализация на языке программирования Julia

```
using Printf
using CSV
using DataFrames
using Base.Threads
using ProgressMeter
function split_number_str(N::Integer, m::Integer)
s = string(N)
if N < 10
s = lpad(s, m, '0')
end
len = length(s)
base_len = div(len, m)
remainder = len % m
parts = String[]
idx = 1
for i in 1:m
current_len = base_len + (i <= remainder ? 1 : 0)
push!(parts, s[idx:idx+current_len-1])
idx += current_len
end
return parts
end
function multiply_preserve_length(part::String, k::Integer)
num = parse(BigInt, part) * k
result = string(num)
return lpad(result, length(part), '0')
end
function remove_leading_zeros(s::String)
if all(c -> c == '0', s)
return "0" # Return "0"
else
idx = findfirst(c -> c != '0', s)
return s[idx:end]
end
end
function compare_pq_nk(pq::String, nk::String)
if pq == nk
return "Full match"
end
min_len = min(length(pq), length(nk))
prefix_match = 0
for i in 1:min_len
pq[i] == nk[i] ? prefix_match += 1 : break
end
suffix_match = 0
for i in 1:min_len
pq[end - i + 1] == nk[end - i + 1] ? suffix_match += 1 : break
```

```
end
if prefix_match > 0 && suffix_match > 0
return "Prefix and suffix match"
elseif prefix_match > 0
return "Prefix matches only"
elseif suffix_match > 0
return "Suffix matches only"
else
return "No match"
end
end
function check_algoritm(N::Integer, m::Integer, k::Integer)
N_str = string(N)
nk_str = string(N * k)
parts_str = split_number_str(N, m)
multiplied_parts_str = [multiply_preserve_length(p, k) for p in parts_str]
pq_str = join(multiplied_parts_str)
pq_clean = remove_leading_zeros(pq_str)
nk_clean = remove_leading_zeros(nk_str)
result = compare_pq_nk(pq_clean, nk_clean)
return (
N = N,
m = m,
k = k,
parts = string(parts_str),
multiplied_parts = string(multiplied_parts_str),
PQ = pq_clean,
NK = nk_clean,
result = result
)
end
function run_tests_parallel(start_N::Integer, stop_N::Integer, m::Integer, k::Integer)
results1_df = DataFrame(
N = Int[],
m = Int[],
k = Int[],
parts = String[],
multiplied_parts = String[],
PQ = String[],
NK = String[],
result = String[]
)
count_full = Atomic{Int}(0)
count_partial_start = Atomic{Int}(0)
count_partial_end = Atomic{Int}(0)
count_partial_both = Atomic{Int}(0)
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_algoritm(N, m, k)
Threads.atomic_add!(count_full, res.result == "Full match" ? 1 : 0)
```

```julia
Threads.atomic_add!(count_partial_start, res.result == "Prefix matches only" ? 1 : 0)
Threads.atomic_add!(count_partial_end, res.result == "Suffix matches only" ? 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!(results1_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("\n Saving results to CSV...")
CSV.write("results.csv", results1_df)
open("statistics.txt", "w") do io
write(io, "Structural Numerical Symmetry Algorithm")
write(io, "=========================================\n")
write(io, "N range: [$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 and suffix match: $partial_both\n")
write(io, "Prefix matches only: $partial_start\n")
write(io, "Suffix matches only: $partial_end\n")
write(io, "No matches: $none\n")
write(io, "Per-number results in 'results.csv'\n")
end
println("\n Summary statistics:")
@printf("Full matches: %d\n", full)
@printf("Prefix and suffix match: %d\n", partial_both)
@printf("Prefix matches only: %d\n", partial_start)
@printf("Suffix matches only: %d\n", partial_end)
@printf("No matches: %d\n", none)
println("\n Statistics saved to 'statistics.txt'")
println("Results saved to 'results.csv'")
return results1_df
end
start_N = 1
stop_N = 10000000
m = 2
k = 99999999
```

```
run_tests_parallel(start_N, stop_N, m, k)
```

Документация.

Назначение:

Скрипт тестирует поведение числового преобразования:

для каждого целого числа `N` из диапазона `[start_N, stop_N]`:

1. Разбивает десятичную запись `N` на `m` частей.
2. Умножает каждую часть на `k`, сохраняя исходную длину части (с ведущими нулями).
3. Конкатенирует результаты умножения → строка `PQ`.
4. Вычисляет прямое произведение `N * k` → строка `NK`.
5. Удаляет ведущие нули из `PQ` и `NK`.
6. Сравнивает полученные строки по префиксу и суффиксу.
7. Сохраняет результаты в CSV и выводит статистику.

Зависимости:

using Printf
using CSV
using DataFrames
using Base.Threads
using ProgressMeter

Функции:

`split_number_str(N::Integer, m::Integer) → Vector{String}`
Разбивает десятичную запись числа `N` на `m` подстрок приблизительно равной длины:
- Если `N < 10`, дополняет слева нулями до длины `m` перед разбиением.
- Первые `len(N) % m` частей будут длиннее остальных на 1 символ.
- Возвращает вектор строк — части числа слева направо.

`multiply_preserve_length(part::String, k::Integer) → String`

Умножает числовое значение строки `part` на `k` (с использованием `BigInt`) и возвращает строку результата, дополненную **слева нулями** до длины исходной строки `part`.
Если результат длиннее — возвращается без обрезки.

`remove_leading_zeros(s::String) → String`

Удаляет все ведущие нули из строки `s`.
Если строка состоит только из нулей — возвращает `"0"`.

`compare_pq_nk(pq::String, nk::String) → String`

Сравнивает две строки `pq` и `nk`:
- Если строки идентичны → `"Full match"`.
- Иначе определяет длину общего префикса (совпадение с начала до первого различия) и общего суффикса (совпадение с конца до первого различия).
- Возвращает одну из строк:

- "Prefix and suffix match"` — если оба совпадения ненулевые,
- `"Prefix matches only"` — только префикс,
- `"Suffix matches only"` — только суффикс,
- `"No match"` — если ни префикс, ни суффикс не совпадают.

`check_algoritm(N::Integer, m::Integer, k::Integer) → NamedTuple`
Выполняет полный цикл обработки одного числа `N`:
- Вычисляет `NK = string(N * k)`.
- Разбивает `N` на `m` частей.
- Умножает каждую часть на `k` с сохранением длины.
- Конкатенирует → `pq_str`.
- Удаляет ведущие нули из `pq_str` и `nk_str`.
- Сравнивает очищенные строки.
- Возвращает именованный кортеж с полями: `N`, `m`, `k`, `parts`,

`multiplied_parts`, `PQ`, `NK`, `result`.

Примечание: `parts` и `multiplied_parts` сохраняются как строковые представления массивов (через `string(...)`), а не как массивы.

`run_tests_parallel(start_N, stop_N, m, k) → DataFrame`
Запускает обработку всех `N` от `start_N` до `stop_N` (включительно):
- Использует многопоточность (`Base.Threads`).
- Отображает прогресс через `ProgressMeter`.

- Подсчитывает количество каждого типа результата с атомарными счётчиками.
- Сохраняет все результаты в `DataFrame` и записывает его в файл `results.csv`.
- Записывает сводную статистику в `statistics.txt`.
- Выводит статистику в консоль.
- Возвращает `DataFrame` с результатами.

Глобальные параметры (в конце скрипта)
start_N = 1
stop_N = 10000000
m = 2
k = 99999999
Запускает тестирование 10 миллионов чисел, разбивая каждое на 2 части и умножая на 99 999 999.

Выходные файлы
-`results.csv` — CSV-файл с колонками:
`N`, `m`, `k`, `parts`, `multiplied_parts`, `PQ`, `NK`, `result`
-`statistics.txt` — текстовый файл со статистикой:
- Диапазон `N`
- Значения `m` и `k`
- Количество случаев каждого типа совпадения

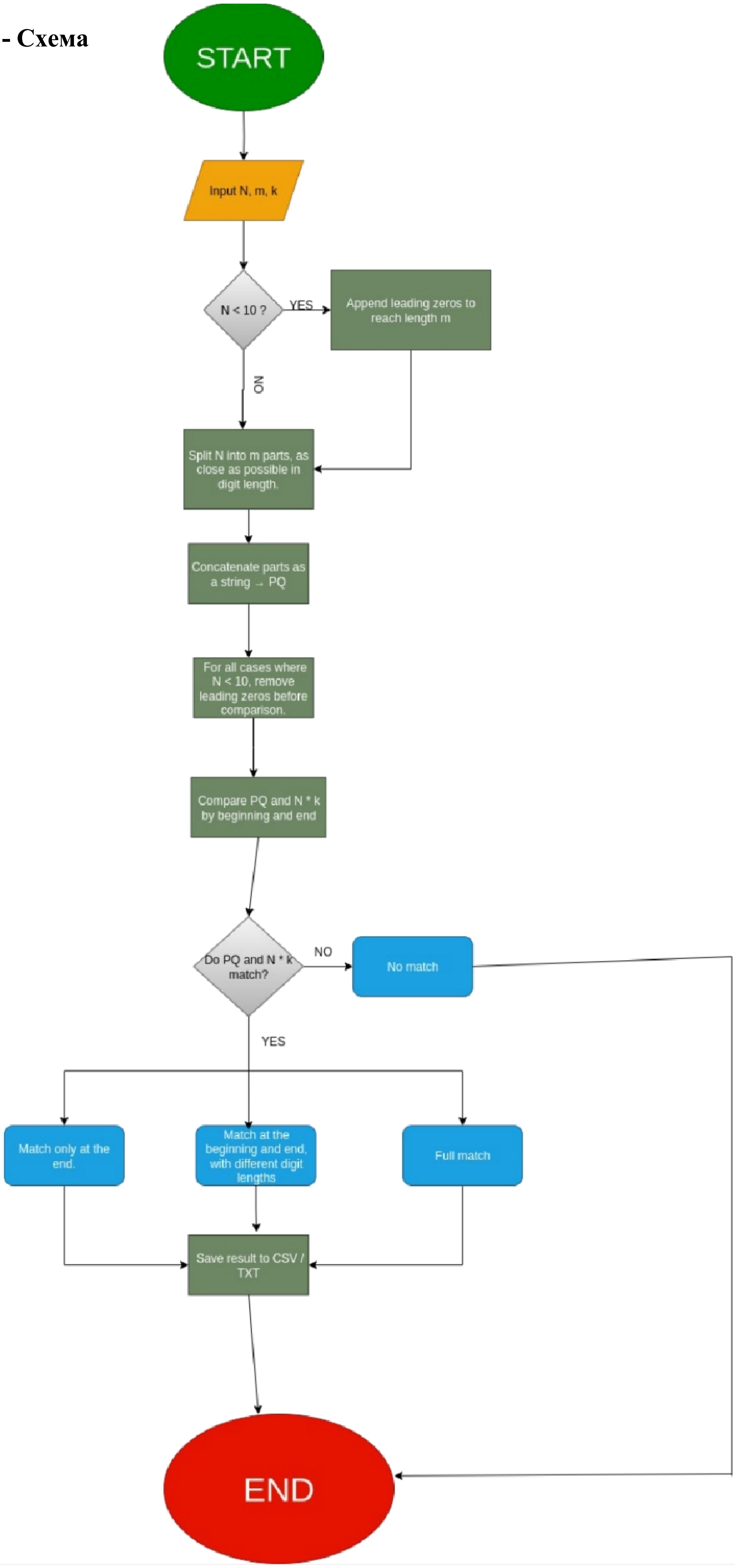
Требования к среде:
- Julia ≥ 1.6
- Пакеты: `Printf`, `CSV`, `DataFrames`, `Base.Threads`, `ProgressMeter`
- Для многопоточности рекомендуется запускать с флагом `--threads=N`

Лицензия и авторство:

Код разработан в контексте исследования Структурной числовой симметрии (SNS).

Автор: Ющенко Михаил Юрьевич

12. Блок - Схема



13. Доказательство:

Нужно классифицировать числа по типу совпадения

Например:

Класс F: числа с полным совпадением (доказано)

Класс B: числа с совпадением начала и конца (не доказано/не доказуемо в теории, но имеет крепкую эмпирическую базу!)

Класс E: числа с совпадением только конца (доказано)

Класс S: числа с совпадением только по началу (опровергнуто)

Класс N: числа без совпадений (доказано)

В рамках алгоритма СЧС любое натуральное число N, разбитое на m≥2 частей и подвергнутое поблоковому умножению на общий множитель k∈N, порождает результат PQ, который всегда относится к одному из пяти классов совпадения с классическим произведением NK=N×k. Ниже приведена строгая классификация с доказательствами или обоснованиями для каждого класса.

Обозначения:

N∈N — исходное число.

m ≥ 2 — количество частей при разбиении.

k∈N — общий множитель.

N = a1, a2… am — разбиение на непересекающиеся десятичные блоки (конкатенация).

PQ = ($a1 * k$) ‖ ($a2 * k$) ‖ … ‖ ($am * k$) — результат конкатенации умноженных блоков (с сохранением длины).

NK = N * k — обычное произведение.

L = длина(am) — количество цифр в последнем блоке (L ≥ 1).

Класс F — Полное совпадение

Условие:

PQ = NK

Доказательство:

Полное совпадение достигается тогда и только тогда, когда при умножении ни одна часть не увеличивается в разрядности:

$\forall i$, длина (ai * k) = длина (ai)

Если это условие выполнено, то позиционные веса всех блоков в PQ совпадают с их вкладом в NK, переносы между блоками отсутствуют, и структура сохраняется полностью.

→Результат:

PQ = NK → Доказано конструктивно.

Класс B — Совпадают начало и конец

Условие: Префикс и суффикс PQ и NK совпадают, но ≠ PQ = NK.

Статус:

Не доказуемо в рамках современной математики.

Эмпирически устойчиво: наблюдается в любых случаев при различных m , k.

Теоретически — открытая проблема.

Значение:

Этот класс составляет ядро открытия СЧС:

несмотря на игнорирование переносов между частями, границы числа остаются согласованными, даже когда середина расходится.

Заключение:

→ Эмпирический факт, не следующий из известных теорем.

Класс E — Совпадает только конец

Условие:

Только младшие (конечные) разряды чисел PQ и NK совпадают, а старшие — различаются.

Доказательство:

Любое натуральное число N можно представить как:

$N = A * 10 ^ L + am$

где:

am — последняя часть числа (последний блок при разбиении),

L — количество цифр в этом блоке ($L \geq 1$),

A — всё, что стоит перед этим блоком (целое неотрицательное число). Тогда

обычное произведение:

$NK = N * k = A * k * 10 ^ L + am * k$

Заметим, что первое слагаемое $A * k * 10 ^ L$ делится на $10 ^ L$, поэтому не влияет на последние L цифр результата.

Следовательно: $NK \equiv am * k(\mod 10 ^ L)$

Теперь рассмотрим PQ — результат поблокового умножения и конкатенации.

Последние L цифр PQ — это результат умножения am * k, записанный с ведущими нулями до длины L.

Это означает, что:

$PQ \equiv am * k(\mod 10 ^ L)$

Отсюда немедленно следует:

$PQ \equiv NK(\mod 10 ^ L)$

→Последние L ≥ 1 цифр всегда совпадают.

Таким образом, совпадение хотя бы в конечных разрядах — не случайность, а строгое следствие десятичной позиционной записи.

Если при этом старшие разряды различаются (что часто бывает при переносах между блоками), то результат относится к классу E.

→ Класс E возможен и доказуем.

Класс S — Совпадает только начало

Условие:

Префикс (начальные разряды) чисел PQ и NK совпадает,

но суффикс (конечные разряды) — различен.

Доказательство от противного:

1. Предположим, что существует число N, для которого результат СЧС PQ и классическое произведение NK = N * k совпадают только в начале, а в конце — не совпадают.

1. Однако, как показано в доказательстве для класса E, для любого разбиения числа N на части, где последняя часть имеет длину L ≥ 1, всегда выполняется: $PQ \equiv NK(\mod 10^L)$ Это означает, что последние L цифр всегда совпадают.

1. Следовательно, совпадение в конце неизбежно.

2. Но это противоречит исходному предположению, что совпадение есть только в начале.
→Предположение ложно.

Вывод: класс S логически невозможен.

Ни одно натуральное число не может принадлежать классу S.

S = ∅.

Класс N — Без совпадений

Условие: Ни один разряд чисел PQ и NK не совпадает — ни в начале, ни в конце, ни в середине.
Доказательство: Как показано в анализе класса E, последние L цифр результата СЧС всегда совпадают с последними L цифрами обычного произведения N * k, где L — количество цифр в последней части числа N. Поскольку L не может быть меньше 1 (последняя часть всегда содержит хотя бы одну цифру), как минимум последняя цифра всегда совпадает. Следовательно, полное отсутствие совпадений невозможно.

Вывод: класс N логически не может существовать. Множество чисел, относящихся к классу N, пусто.

N = ∅.

Итоговая таблица

Класс	Возможен?	Основание
F	✅ Да	Конструктивное условие: длина блоков не растёт
B	✅ Да	Эмпирический феномен, теоретически открыт
E	✅ Да	PQ ≡ NK(mod10 ^ L) — строго доказано
S	❌ Нет	Противоречит совпадению конца
N	❌ Нет	Противоречит совпадению хотя бы в одной цифре

14. Лицензия и контакты:

Автор: ([Михаил])([https://github.com/Misha0966]) почта для
связи: misha0966.33@gmail.com

Сайт: https://structuralnumericalsymmetry.ru

Телеграмм-канал: @structuralnumericalsymmetry

Ссылка на группу в ВК:

https://vk.com/structuralnumericalsymmetry