О линейной оценке сложности проверки простоты числа однородными структурами*

А.М. Степаненков

В статье показывается, что при Тьюринговой кодировке натуральных чисел их простота проверяется однородными структурами за время, асимптотически совпадающее с половиной длины кода.

Введение

Проверка свойств натуральных чисел имеет большое значение в приложениях теории чисел. Важным при этом является оценка сложности соответствующих алгоритмов, а также построение алгоритмов, оптимальных по сложности.

Одной из перспективных моделей алгоритмов являются однородные структуры автоматов, поскольку наряду с наглядностью представления данных и простотой задания самой однородной структуры в них реализуется процесс параллельной обработки информации. Это даёт возможность проверить свойство простоты за линейное время.

1. Основные понятия и результаты

В работе будет использоваться понятие однородной структуры. Содержательно, однородная структура (кратко ОС) представляет собой бесконечную схему, построенную из копий одного и того же ко-

*Работа выполнена при поддержке гранта РФФИ 02–01–00162а.

нечного автомата и такую, что правила соединения входов произвольного автомата с выходами других автоматов в ней везде одинаковы. В дальнейшем эти автоматы будем называть ячейками ОС. Дадим формальное определение описанного только что объекта. Однородной структурой размерности k называется четвёрка (Z^k, Q, V, φ), где Z^k есть множество k-мерных векторов с целыми координатами, Q — конечное множество состояний, $V = \{\alpha_1, \ldots, \alpha_{n-1}\}$ — упорядоченный набор различных ненулевых векторов из Z^k, называемый шаблоном соседства ОС и определяющий для каждой ячейки α её окрестность $V(\alpha) = \{\alpha, \alpha+\alpha_1, \ldots, \alpha+\alpha_{n-1}\}$, $\varphi : Q^n \to Q$ — локальная функция переходов. Если в момент времени t состояния ячеек $\alpha, \alpha+\alpha_1, \ldots, \alpha+\alpha_{n-1}$ были равны соответственно $x_0, x_1, \ldots, x_{n-1}$, то состояние α в момент $t+1$ полагается равным $\varphi(x_0, x_1, \ldots, x_{n-1})$. Состоянием ОС будем называть функцию, сопоставляющую каждой ячейке её состояние из множества Q. Если состояние ОС в момент времени t есть f, то её состояние в момент $t+1$ есть функция g, определяемая равенством $g(\alpha) = \varphi(f(\alpha), f(\alpha+\alpha_1), \ldots, f(\alpha+\alpha_{n-1}))$. Это равенство задаёт основную функцию переходов Φ : $g = \Phi(f)$. Поведением ОС назовём последовательность $\{f_i\}$ её состояний такую, что $f_{i+1} = \Phi(f_i)$. В дальнейшем поведение однородной структуры часто будем называть функционированием ОС.

Существует один выделенный элемент из множества Q — символ пустоты. Будем обозначать его через «П». Далее рассматриваем лишь такие ОС, у которых $\varphi(\text{«П»}, \ldots, \text{«П»}) = \text{«П»}$. Состояние «П» является состоянием покоя. Состояние ОС, у которого лишь конечное число ячеек находится в отличном от «П» состоянии, назовём конфигурацией.

Перейдём к формальной постановке задачи, которая и будет рассматриваться в дальнейшем, а именно, задачи проверки простоты натурального числа с помощью однородных структур.

Под кодом натурального числа N будем понимать цепочку, состоящую из N одинаковых символов.

Требуется построить плоскую однородную структуру (ОС) такую, что, записав код любого натурального числа $N > 1$ в её ячейки (то есть, приведя цепочку из N автоматов в состояние, соответствующее символу кода; остальные автоматы должны при этом находиться в выделенном (пустом) состоянии), через конечное число шагов

в результате функционирования ОС возникнет следующая конфигурация: все ячейки, кроме одной, пусты, а последняя пребывает в одном из состояний «Y» или «N». Причём во все последующие моменты времени состояние ОС не изменяется. Это состояние ОС будем интерпретировать как ответ на вопрос: N — простое? Если ОС даёт правильный ответ для любого натурального числа, скажем, что данная ОС решает задачу проверки простоты натурального числа.

Введём некоторые ограничения на ОС. Число состояний ячейки должно быть ограничено некоторой константой m: $|Q| \leqslant m$. В качестве шаблона соседства может выступать любой упорядоченный набор, состоящий из элементов множества $V = \{(-1,-1),(0,-1),(1,-1),(-1,0),(1,0),(-1,1),(0,1),(1,1)\}$.

Естественно стараться оптимизировать время работы ОС. Это время будем выражать как функцию от длины задачи (длины кода) — N. Используем для времени работы ОС σ обозначение $T_\sigma(N)$.

Далее будем рассматривать только те ОС, которые удовлетворяют нашим ограничениям. Рассмотрим все ОС, которые решают задачу проверки простоты натурального числа (то есть, дают ответ «Y», если число простое, и «N», если составное).

Введём функцию сложности $T(N) = \min\{T_\sigma(N)\colon \sigma$ решает задачу о проверке числа на простоту$\}$.

Основным результатом работы является следующее утверждение.

Теорема 1. $T(N) \sim \frac{N}{2}$.

2. Нижняя оценка для $T(N)$

Лемма 1. *Для любого* $N > 1$ *выполнено* $T(N) \geqslant [\frac{N}{2}] - 1$.

Доказательство. Предположим, что существует N такое, что $T(N) < [\frac{N}{2}] - 1$. Тогда из определения функции T существует ОС σ, которая проверяет простоту N за время, меньшее $[\frac{N}{2}] - 1$. Обозначим её время работы числом K.

Под окрестностью ячейки α размера k будем понимать множество ячеек, таких что сигнал от них может достигнуть α за k или менее тактов. Скажем, что окрестность ячейки α_1 размера k в момент времени t_1 совпадает с окрестностью ячейки α_2 размера k в момент

времени t_2, если конфигурация окрестности ячейки α_1 в момент времени t_1 совпадает с конфигурацией ячейки α_2 в момент времени t_2. Заметим, что эти понятия зависят от шаблона соседства.

Поскольку время работы ОС σ на числе N строго меньше $[\frac{N}{2}]-1$, то существуют символы кода, которые не попадают в окрестность ответа размера K. Однако в окрестность ответа обязательно должен входить либо крайний левый, либо крайний правый символ кода. Действительно, если это не так, то в силу $K < [\frac{N}{2}]-1$, существует ячейка, окрестность размера K которой в начальный момент времени совпадает с окрестностью ответа того же размера в тот же момент времени. Следовательно, через K тактов она перейдёт не в пустое состояние, а либо в «Y», либо в «N», что не допустимо. Покажем, что на всех числах M, больших N, ОС σ будет выдавать тот же ответ, что и на N.

Конфигурация для числа M получается из конфигурации для числа N приписыванием некоторого числа символов кода. Будем считать, что все символы кода приписываются так, что они не попадают в окрестность ячейки размера K, в которой был зафиксирован ответ. Рассмотрим ячейку, в которой был сформирован ответ при обработке числа N. Очевидно, её окрестность размера K в начальный момент времени не изменилась. Следовательно, через K тактов эта ячейка перейдёт в то же самое состояние, то есть возникнет тот же ответ.

Теперь рассмотрим любую другую ячейку. Покажем, что через K тактов она перейдёт в пустое состояние. Тем самым мы докажем, что на любом числе M большем N, однородная структура σ выдаёт тот же ответ, что и на N. Справедливость сформулированного только что утверждения следует из того, что для любой подобной ячейки α существует ячейка β такая, что окрестность ячейки α размера K в новой задаче в начальный момент времени совпадает с окрестностью ячейки β размера K в старой задаче в начальный момент времени. Действительно, ячейки α можно разделить на два типа: в окрестность размера K которых попадает хотя бы один крайний символ кода, и остальные (они существуют в силу $K < [\frac{N}{2}] - 1$ и того, что ячейка с ответом не обладает этим свойством). Ясно, что окрестности размера K ячеек второго типа с одинаковыми ординатами в начальный момент времени равны. Приписывание же новых символов кода на самом деле порождает ячейки второго типа, так как $K < [\frac{N}{2}] - 1$.

Таким образом, если N было простым, то возьмём $M = N + 1$ — составное. (Ясно, что N не могло быть двойкой, так как тогда $K < 0$). Если же N было составным, то будем прибавлять к нему по единичке, пока не получим простое число M (это нам гарантирует теорема о бесконечности множества простых чисел). В обоих случаях ОС σ даёт одинаковый ответ на числах N и M, что противоречит определению ОС σ как решающей задачу о проверке простоты натурального числа. Лемма доказана.

3. Верхняя оценка для $T(N)$. Асимптотически наилучший алгоритм проверки простоты числа

Сначала дадим общее описание алгоритма.

Теорема о нижней оценке по существу даёт подсказку, как надо действовать для того, чтобы получить асимптотически наилучший алгоритм — надо собирать информацию в центре конфигурации. И сразу же после того, как последний сигнал добрался до этого центра, или спустя некоторое время ($\mathrm{o}(N)$) на основе собранных данных делать вывод. Так мы и будем действовать. Со стороны, функционирование ОС будет выглядеть так: конфигурация, состоящая из палочек, будет стремительно уменьшаться в размерах с обоих концов кода. Однако, во-первых, информация не должна теряться, во-вторых, информация (палочная кодировка числа) должна воплотиться в нечто более компактное, анализ чего не составил бы труда, например, в двоичное представление числа. Поэтому, сворачиваясь, конфигурация будет преобразовывать себя в двоичный код натурального числа N. Когда мы его получим, все преобразования, которые будут вестись, — будут преобразования над данными длины $\log N$, а это значит, что нам позволяется применять практически любые алгоритмы, в том числе переборные, и всё равно остаёмся в условиях ограничения функцией вида $\mathrm{o}(N)$.

Вначале рассмотрим вспомогательную задачу проверки простоты натурального числа в плоских однородных структурах для случая, когда кодировка представляет собой его двоичное представление. Для этого случая справедлива следующая лемма.

Лемма 2. *Существует плоская однородная структура, решающая задачу определения простоты натурального числа N при двоичной кодировке за время* $o(N)$.

Доказательство. Утверждение леммы 2 вытекает из того, что можно предложить простой алгоритм (построить машину Тьюринга), которая решала бы поставленную задачу.

Предположим, что алфавит МТ есть множества пар (a, b), где a и b могут быть «П», «0», «1», «N», «Y» и некоторыми другими вспомогательными символами.

Начальная конфигурация на ленте: двоичный код числа, записанный в первых компонентах вектора, во вторых компонентах — символы пустоты. Головка стоит над первым символом кода. Финальная конфигурация: на ленте существует лишь один символ (a, b), отличный от («П», «П»). Он имеет вид («Y», «П») или («N», «П»), и над ним стоит головка машины Тьюринга.

В дальнейшем, мы будем часто использовать в качестве алфавита состояний — конечное множество векторов некоторой размерности. Ясно, что ничего криминального в этом нет — всегда можно построить взаимно однозначное соответствие между алфавитом векторов-состояний и классическим алфавитом состояний. Просто использование векторов-состояний автору кажется более удобным средством для описания алгоритмов. Координаты вектора-ячейки однородной структуры или вектора-ячейки ленты машины Тьюринга будем называть слоями.

Алгоритм выглядит так:

Шаг 1. Формирование во втором слое двоичного кода числа 1, начало которого совпадает с началом двоичного кода числа N. Переход на начало кода числа N.

Это можно сделать за константное время.

Шаг 2. Пусть длина кода числа, записанного в первом слое, есть n, во втором — k. Проверка, что $n > 2k - 1$. Переход на начало кода числа в первом слое, но в различных состояниях, в зависимости от результата сравнения.

Это можно сделать за квадратичное время (нанести во втором слое справа от кода числа его копию — квадрат, проверить $n > 2k-1$ — константа, стереть копию — линейное время).

Шаг 3. Если $n > 2k - 1$, то число N — простое. Надо стереть его

код и код числа во втором слое. Написать («Y», «П»), остановиться и перейти в заключительное состояние.

Иначе, прибавить 1 к числу, записанному во втором слое. Установить головку на начало кода числа в первом слое.

Обе операции выполняются за линейное время.

Шаг 4. Скопировать коды чисел первого и второго слоя вправо и остановиться на их начале.

Квадратичное время.

Шаг 5. Проверка, что число в первом слое делится на число во втором слое на основе алгоритма деления в столбик. Стирание копий. Установление головки на начало первоначального кода числа N и переход в состояние, соответствующее результату проверки.

Квадратичное время.

Шаг 6. Если результат проверки положительный, то N — составное. Надо стереть его код и код числа во втором слое. Написать («N», «П»), остановиться и перейти в заключительное состояние.

Иначе, переход к шагу 2.

Легко видеть, что число итераций в шагах $2-6$ есть $\mathrm{O}(2^{\frac{n}{2}})$. Время, затрачиваемое на каждой итерации, есть $\mathrm{O}(n^2)$. Следовательно, общее время есть $\mathrm{O}(n^2 2^{\frac{n}{2}}) = \mathrm{o}(N)$.

В свою очередь, любую машину Тьюринга легко смоделировать с помощью однородной структуры размерности k с шаблоном соседства $\{(-1,0,\ldots,0),(1,0,\ldots,0)\}$ без замедления следующим образом.

Рассмотрим МТ с входным алфавитом A, алфавитом состояний Q, тогда моделирующая ОС может выглядеть так: алфавит состояний — $A \times \{0,1\} \times Q$. Состояние ячейки ОС складывается из символа на ленте МТ — алфавит A, отсутствия или присутствия головки — алфавит $\{0,1\}$, и (в случае присутствия головки) состояния МТ — алфавит Q. Функция переходов выглядит следующим образом. Это тождественная функция, если в окрестности ячейки нет ячейки в состоянии $(a,1,c)$, где a,c — любые. В остальных случаях она строится по программе МТ.

Из описанного выше алгоритма и возможности моделировать МТ в однородных структурах без замедления вытекает справедливость утверждения леммы 2.

Автором была построена плоская однородная структура σ_1, которая проверяет простоту любого натурального числа, большего трой-

ки, со временем работы меньшим $2^{]\frac{n}{2}[} + \frac{n^2}{2} + 5n + 7$. ОС σ_1 имеет шаблон соседства $V_2 = \{(-1,-1),(0,-1),(1,-1),(-1,0),(1,0),(0,1)\}$. Сама ячейка представляет собой шестимерный вектор. Координаты (слои) могут принимать следующие состояния:

1-й: «П», «1», «0»;

2-й: «П», «1», «0», «<», «#»;

3-й: «П», «Y», «N», «U», «D», «B», «C», «X», «x», «y», «m», «u», «v», «l», «r», «o», «<», «>», «#», «/», «|», «!», «?», «9», «8», «7», «6», «5», «4», «1», «0»;

4-й: «П», «1», «p»;

5-й: «П», «1», «0», «s»;

6-й: «П», «1», «0»;

Как видно, число состояний, которые может принимать ячейка ОС σ_1, не превышает $3 \cdot 5 \cdot 31 \cdot 3 \cdot 4 \cdot 3 = 16740$. Однако при тестировании ОС σ_1 на компьютере удалось обнаружить лишь 213 различных состояний.

Начальной конфигурацией для ОС σ_1 будет цепочка двоичного кода числа N, записанная в первом слое. Также в третий слой второй (от начала кода) ячейки наносится символ «>», а в тот же самый слой последней — символ «<».

Функционирование ОС σ_1 можно условно разбить на четыре процесса.

Первый процесс производит проверку числа на чётность и (в случае отрицательного результата) производит подготовку к размножению вниз кода числа, записанного в первом слое и кодов нечётных потенциальных делителей, записанных во втором слое. Поскольку при проверке на простоту число N достаточно делить только на простые, меньшие $]\sqrt{N}[+1$, то, отступая на одну ячейку от середины кода числа в третьем слое, формируется сигнал x, который размножается вместе с потенциальными делителями. Таким образом, можно остановить процесс размножения после того, как появится делитель равный $]\sqrt{N}[$ (точнее $2^{]\frac{n}{2}[} - 1$). После этого запускается процесс остановки и уничтожения размножающейся вниз части конфигурации. Уничтожающие сигналы могут возникнуть и при появлении в третьем слое сигнала «N». Параллельно с размножением, в каждой строке, где нанесён код числа N в первом слое и код его потенциального делителя K во втором, осуществляется процесс проверки делимости N на K (на основе алгоритма деления в столбик). Результат проверки в виде

сигналов «Y» или «N» записывается в третьем слое, лишняя информация стирается, а вышеупомянутые сигналы передаются вниз, где на их основе формируется окончательный ответ.

Опишем подробнее каждый блок.

Блок 1. Подготовка к размножению

Итак, первое, что делается, это проверка числа N на чётность (если в последней непустой ячейке в первом слое стоит символ 0, то N — чётно, и мы даём ответ «N»).

В случае отрицательного результата проверки, сигналы «>» и «<» двигаются навстречу друг другу для того, чтобы обозначить то место на числе N, до которого позволительно расти потенциальным делителям K.

Далее синхронизируется рост части конфигурации, связанной с формированием делителей K, и движение впоследствии останавливающего этот рост сигнала «x».

На этапе подготовке затрачивается время, которое не превышает $3[\frac{n+1}{2}] + 1$.

Блок 2. Размножение

Размножение кода числа N и формирование кодов его потенциальных делителей K происходит с задержкой в один такт. Сделано это для того, чтобы, в случае возникновения сигнала «N», уничтожающие сигналы сумели догнать стремительно разрастающуюся вниз конфигурацию. Передача вниз кода числа N происходит довольно просто: он с задержкой в один такт копируется вниз в первый слой. Формирование кодов потенциальных делителей K происходит несколько сложнее. За это отвечают третий, пятый и шестой слои. Новый потенциальный делитель формируется во втором слое на основе прибавления двойки к потенциальному делителю $K - 2$, записанному во втором слое верхних ячеек. (Заметим, что на этапе подготовки формируется код первого потенциального делителя — тройки).

Блок 3. Проверка на кратность одного натурального числа другому

Здесь решается следующая задача: делится ли число N, записанное в первом слое ячеек, на число K, записанное во втором слое ячеек непосредственно под числом N.

В первую очередь проверяется корректность записи чисел N и K. А именно, если число K больше соответствующей части числа N (то есть начала числа N, под которым записано число K), то его необходимо сдвинуть на одну ячейку вправо. Затем осуществляем вычитание $(N-K)$ с последующим сдвигом числа K вправо и повторяем всё заново. Выполнение цикла осуществляется до тех пор, пока правая часть кода K не совмещается с правой частью кода N. В случае тождественного совпадения кода K и кода остатка N мы делаем вывод, что K делит N, иначе — не делит. В первом случае в третьем слое ячейки ставим символ «N» (число N не простое), во втором случае ставим символ «Y» (число N после деления на K сохраняет возможность быть простым). В заключении, очищаем все ячейки (кроме той, которая содержит ответ).

Время проверки на кратность числа N числу K, если считать размеры их кодов равными n и k соответственно, не больше $2k(n-k)+n+1$ тактов.

Блок 4. Остановка роста конфигурации. Удаление лишней информации

Критерий остановки роста конфигурации следующий: либо в процессе проверки на кратность выяснилось, что N — составное, либо закончились потенциальные делители, то есть N — простое. В первом случае формируется сигнал «N», который поглощает сигналы «Y», во втором случае имеют место только сигналы «Y», которые сольются в один. В обоих случаях возникает уничтожающий сигнал со скоростью движения в два раза превышающей скорость роста конфигурации, который обеспечивает уничтожение всех сигналов, отличных от «N» и «Y».

Как видно, худший случай времени работы ОС σ_1 достигается на простых числах. В случае составного N время работы ОС σ_1 асимптотически равно наименьшему делителю числа N. В приведённой

однородной структуре на каждом такте задействуется не так много автоматов-вычислителей — размер максимальной конфигурации не больше $\frac{n^3}{4} + 5\frac{n^2}{4} + 2n$. Путём вовлечения в работу большего числа вычислителей можно существенно улучшить время работы в худшем случае. Сделать это можно следующим образом.

Будем проверять делимость числа N на блоках, состоящих из $O(2^{\frac{n}{4}})$ потенциальных делителей. Проверка на делимость в разных блоках будет осуществляться параллельно.

Более подробно.

Сначала осуществляем проверку на чётность.

Затем сформируем начальную конфигурацию первого блока. Она будет состоять из кода числа N, кода первого потенциального делителя — тройки и кода чётного числа-ограничителя порядка $2^{\frac{n}{4}}$, и дополнительных символов, необходимых алгоритму.

На каждом блоке применяется уже описанный алгоритм, у которого изменится лишь критерий возникновения уничтожающего сигнала. Появляться он будет тогда и только тогда, когда очередной потенциальный делитель превысит число-ограничитель. Следовательно, время работы нового алгоритма на таком блоке есть $O(2^{\frac{n}{4}})$.

Начальная конфигурация каждого блока один раз копируется вправо следующим образом: копируется код числа N, формируется код первого потенциального делителя, равного увеличенному на единицу числу-ограничителю соседнего слева блока, формируется код числа-ограничителя, равного сумме числа-ограничителя первого блока и числа-ограничителя соседнего слева блока. Все эти действия можно осуществить за $O(n)$. Критерий остановки роста конфигурации вправо — первый потенциальный делитель превысил $2^{]\frac{n}{2}[}$.

В результате проверки на делимость в каждом блоке формируется свой промежуточный результат. Эти результаты необходимо объединить. Ответ каждого блока, кроме первого, будет направляться влево, где он объединяется с ответом соседнего блока.

Нетрудно оценить время работы этого алгоритма. Оно складывается из времени на формирование начальной конфигурации последнего блока — $O(n2^{\frac{n}{4}})$, проверки на делимость в последнем блоке — $O(2^{\frac{n}{4}})$ и передачи промежуточного результата в первый блок — $O(n2^{\frac{n}{4}})$. В сумме получится $O(n2^{\frac{n}{4}})$. Несложно посчитать максимальное число автоматов-вычислителей, задействованных в работе.

Максимальное число непустых ячеек в каждом блоке есть $O(n^3)$. Поскольку новые блоки формируются с задержкой в $O(n)$, то число блоков, в которых ведутся вычисления, есть $O(\frac{1}{n}2^{\frac{n}{4}})$. В итоге получаем, что размер максимальной конфигурации есть $O(n^2 2^{\frac{n}{4}})$. Таким образом, добавив $O(\frac{1}{n}2^{\frac{n}{4}})$ новых вычислителей, мы во столько же уменьшили верхнюю оценку времени работы ОС.

Теперь переходим к заключительному шагу.

Лемма 3. *Существует плоская однородная структура, решающая задачу простоты натурального числа N, заданного в палочной кодировке за время $\frac{N}{2} + o(N)$.*

Доказательство. В качестве базовой возьмём ОС σ_1, которая решает задачу простоты любого натурального числа больше тройки, представленного в двоичном коде. Добавив новые состояния ячейки и правила перехода, получим новую однородную структуру σ_2, решающую задачу простоты натурального числа N, палочный код которой записан во втором слое. Если кратко, функционирование ОС σ_2 будет состоять из двух этапов: на первом конфигурация, состоящая из палочного представления, преобразуется в начальную конфигурацию для ОС σ_1, и обрабатываются случаи для $N = 2, 3$, на втором этапе ОС σ_2 работает в соответствии с алгоритмом для ОС σ_1.

Далее перейдём к подробному описанию процесса перекодировки натурального числа из палочного представления в двоичное. Состояния слоёв ячейки будут обозначаться латинскими буквами a, b, c, d, e, f. Координаты ячейки будут записываться в скобках. Например, состояние второго слоя ячейки ОС с координатами $(-1, 1)$ будет обозначаться $b(-1, 1)$. Литералы состояний будут заключены в кавычки («»). Состояние слоя ячейки в следующий (относительно некоторого текущего) момент времени будет обозначаться латинской буквой со штрихом. Литералом «П» обозначается пустой символ.

Перекодировка

1. Образование стартовой конфигурации

В начальный момент времени мы из палочного кода числа N, записанного во втором слое, образуем стартовую конфигурацию: все

палочки заменяются единичками, а в первом слое крайних слева и справа ячеек генерируются сигналы «>» и «<» соответственно.
Если $\mathrm{b}(x-1,y) =$ «П», $\mathrm{b}(x,y) =$ «|», *то* $\mathrm{a}'(x,y) =$ «>».
Если $\mathrm{b}(x,y) =$ «|», $\mathrm{b}(x+1,y) =$ «П», *то* $\mathrm{a}'(x,y) =$ «<».
Если $\mathrm{b}(x,y) =$ «|», *то* $\mathrm{b}'(x,y) =$ «1».

2. Правила свёртки кода слева

Сигнал «>» движется вправо навстречу сигналу «<», уничтожая по пути своего следования единички.
Если $\mathrm{a}(x-1,y) =$ «>», $\mathrm{a}(x,y) \neq$ «<», $\mathrm{a}(x+1,y) \neq$ «<», *то* $\mathrm{a}'(x,y) =$ «>».
Если $\mathrm{a}(x,y) =$ «>», $\mathrm{b}(x,y) =$ «1», *то* $\mathrm{a}'(x,y) =$ «П», $\mathrm{b}'(x,y) =$ «П».

3. Правила свёртки кода справа

Сигнал «<» движется влево до встречи с «>», реализуя в процессе движения формирование двоичного кода числа, равного сумме поглощаемых им единиц.
Если $\mathrm{a}(x-1,y) \neq$ «>», $\mathrm{a}(x,y) \neq$ «>», $\mathrm{a}(x+1,y) =$ «<», *то* $\mathrm{a}'(x,y) =$ «<».
Если $\mathrm{a}(x,y) =$ «<», *то* $\mathrm{a}'(x,y) =$ «П».

Двоичный код будет записываться во втором слое «справа налево», то есть левее всех будет младший разряд. Формирование младшего разряда описано ниже.
Если $\mathrm{a}(x+1,y) =$ «<», $\mathrm{b}(x+1,y) =$ «0», *то* $\mathrm{b}'(x,y) =$ «1».
Если $\mathrm{a}(x+1,y) =$ «<», $\mathrm{b}(x+1,y) =$ «1», *то* $\mathrm{b}'(x,y) =$ «0».

Двоичный код в процессе формирования движется вслед за своим создателем — сигналом «<», изменяясь в пути. Это движение обеспечивают сигналы шестого слоя «/». Их формированием также занимается сигнал «<». В процессе прибавления единиц необходимо иногда держать единичку «в уме» для прибавления её к старшему разряду. Это обеспечивает символ «1», который стоит в пятом слое и меняет пробегающие мимо во втором слое «1» на «0», «0» на «1» до тех пор, пока не произойдёт прибавка к старшему разряду (то есть, пока во втором слое не появится «0»). Ниже приведены правила формирования и движения сигнала «/», «0» или «1» второго слоя и «1» пятого.

Если $\mathrm{a}(x-1,y) \neq$ «>», $\mathrm{a}(x,y) =$ «<», $\mathrm{b}(x,y) =$ «1», $\mathrm{b}(x+1,y) \neq$ «П», *то* $\mathrm{b}'(x,y) = \mathrm{b}(x+1,y)$, $\mathrm{e}'(x,y) =$ «1», $\mathrm{f}'(x,y) =$ «/».

Если $\mathrm{a}(x-1,y) \neq$ «>», $\mathrm{a}(x,y) =$ «<», $\mathrm{b}(x,y) =$ «1», $\mathrm{b}(x+1,y) =$ «П», *то* $\mathrm{b}'(x,y) = \mathrm{b}(x+1,y)$, $\mathrm{e}'(x,y) =$ «1».

Если $\mathrm{a}(x-1,y) \neq$ «>», $\mathrm{a}(x,y) =$ «<», $\mathrm{b}(x,y) =$ «0», *то* $\mathrm{b}'(x,y) = \mathrm{b}(x+1,y)$, $\mathrm{f}'(x,y) =$ «/».

Если $\mathrm{b}(x,y) =$ «1» *или* $\mathrm{b}(x,y) =$ «0», $\mathrm{f}(x,y) =$ «/», *то* $\mathrm{b}'(x,y) = \mathrm{b}(x+1,y)$, $\mathrm{f}'(x,y) = \mathrm{f}(x+1,y)$.

Далее даны условия пропадания символа «1» пятого слоя.

Если $\mathrm{f}(x-1,y) =$ «/», $\mathrm{b}(x,y) =$ «0» или $\mathrm{b}(x,y) =$ «П», $\mathrm{e}(x,y) =$ «1», *то* $\mathrm{e}'(x,y) =$ «П».

Если $\mathrm{a}(x-1,y) =$ «<», $\mathrm{b}(x,y) =$ «0» или $\mathrm{b}(x,y) =$ «П», $\mathrm{e}(x,y) =$ «1», *то* $\mathrm{e}'(x,y) =$ «П».

Описание функционирования символа «1» пятого слоя.

Если $\mathrm{a}(x,y) =$ «<», $\mathrm{b}(x+1,y) =$ «0» или $\mathrm{b}(x+1,y) =$ «П», $\mathrm{e}(x+1,y) =$ «1», *то* $\mathrm{b}'(x,y) =$ «1», $\mathrm{f}'(x,y) =$ «/».

Если $\mathrm{a}(x,y) =$ «<», $\mathrm{b}(x+1,y) =$ «1», $\mathrm{e}(x+1,y) =$ «1», *то* $\mathrm{b}'(x,y) =$ «0», $\mathrm{f}'(x,y) =$ «/».

Если $\mathrm{e}(x,y) \neq$ «#», $\mathrm{f}(x,y) =$ «/», $\mathrm{b}(x+1,y) =$ «0» *или* $\mathrm{b}(x+1,y) =$ «П», $\mathrm{e}(x+1,y) =$ «1», *то* $\mathrm{b}'(x,y) =$ «1», $\mathrm{f}'(x,y) =$ «/».

Если $\mathrm{e}(x,y) \neq$ «#», $\mathrm{f}(x,y) =$ «/», $\mathrm{b}(x+1,y) =$ «1», $\mathrm{e}(x+1,y) =$ «1», *то* $\mathrm{b}'(x,y) =$ «0», $\mathrm{f}'(x,y) =$ «/».

Здесь появился новый символ «#». Он возникает при столкновении сигналов «>» и «<». Условие неравенства ему объясняется так: в случае нечётного N мы не хотим прибавлять последнюю единичку (так как в двоичной записи проще умножить число на 2 и прибавить единицу, что эквивалентно приписыванию к правой части его кода символа 1, чем умножить на 2 и вычесть 1). Следующая строчка из той же серии — вопреки общему правилу, младший разряд кода передаётся без изменений.

Если $\mathrm{a}(x-1,y) =$ «>», $\mathrm{a}(x,y) =$ «П», $\mathrm{a}(x+1,y) =$ «<», *то* $\mathrm{b}'(x,y) = \mathrm{b}(x+1,y)$, $\mathrm{e}'(x,y) =$ «#», $\mathrm{f}'(x,y) =$ «/».

Также необходимо уничтожить сформировавшуюся, возможно, на последнем шаге прибавления единичку в пятом слое.

Если $\mathrm{e}(x-1,y) =$ «#», $\mathrm{e}(x,y) =$ «1», *то* $\mathrm{e}'(x,y) =$ «П».

4. Обработка случая чётного N

Сразу скажем о чётном случае. Если сигналы «>» и «<» вплотную подошли друг к другу, то можно сделать вывод, что N — чётное число. В этом случае генерируется сигнал ответа «N». Далее он передаётся вправо до тех пор, пока не выйдет на край.
Если $\mathrm{a}(x-1,y) = $ «>», $\mathrm{a}(x,y) = $ «<», *то* $\mathrm{a}'(x,y) = $ «П», $\mathrm{b}'(x,y) = $ «П», $\mathrm{c}'(x,y) = $ «N», $\mathrm{d}'(x,y) = $ «П», $\mathrm{e}'(x,y) = $ «П», $\mathrm{f}'(x,y) = $ «П».
Если $\mathrm{c}(x-1,y) = $ «N», $\mathrm{a}(x,y) \neq $ «П», *или* $\mathrm{b}(x,y) \neq $ «П», *или* $\mathrm{e}(x,y) \neq$ «П», *или* $\mathrm{f}(x,y) \neq $ «П», *то* $\mathrm{a}'(x,y) = $ «П», $\mathrm{b}'(x,y) = $ «П», $\mathrm{c}'(x,y) = \mathrm{c}(x-1,y)$, $\mathrm{e}'(x,y) = $ «П», $\mathrm{f}'(x,y) = $ «П».

5. Разворот кода

В нечётном же случае в пятом слое (как мы уже видели) рождается сигнал «#», и процесс формирования кода вступает в новую фазу. Далее нам надо перекопировать бегущие по второму слою символы кода в первый слой, причём, записывая их в обратном порядке, чтобы в полученном коде крайним левым был старший разряд. Процессом копирования кода в первый слой будет заниматься символ «#». Далее нам надо разрешить символам кода забегать в ячейку с сигналом «#».
Если $\mathrm{e}(x,y) = $ «#», $\mathrm{c}(x+1,y) \neq $ «>», *то* $\mathrm{b}'(x,y) = \mathrm{b}(x+1,y)$, $\mathrm{f}'(x,y) = \mathrm{f}(x+1,y)$.

Сигнал «#» копирует содержимое второго слоя в первый, после чего передаётся влево для очередного копирования.
Если $\mathrm{a}(x,y) = $ «П», $\mathrm{b}(x,y) = $ «1» или $\mathrm{b}(x,y) = $ «0», $\mathrm{e}(x,y) = $ «#», *то* $\mathrm{a}'(x,y) = \mathrm{b}(x,y)$.
Если $\mathrm{b}(x,y) = $ «1» *или* $\mathrm{b}(x,y) = $ «0», $\mathrm{e}(x,y) = $ «#», *то* $\mathrm{e}'(x,y) = $ «П».
Если $\mathrm{b}(x+1,y) \neq $ «П», $\mathrm{e}(x+1,y) = $ «#», *то* $\mathrm{e}'(x,y) = $ «#».

6. Формирование начальной конфигурации для ОС σ_1

Когда перестанут поступать сигналы из второго слоя, наступит новый этап в формировании двоичного кода числа N. Нам надо дописать справа единичку, а также в третьем слое крайней правой ячейки сгенерировать сигнал «<», в третьем слое второй слева ячейки сгенерировать сигнал «>», причём сделать это надо синхронно. Тогда

мы получим в точности начальную конфигурацию для ОС σ_1 (к чему мы, собственно, и стремимся). Для реализации этого мы запустим с левого края на правый два сигнала: в третьем слое — «>» и в пятом слое — «r», который будет двигаться с задержкой в один такт. Задержка будет организована с помощью сигнала «R».

Если $\mathrm{a}(x,y) =$ «П», $\mathrm{b}(x,y) =$ «1» *или* $\mathrm{b}(x,y) =$ «0», $\mathrm{e}(x,y) =$ «#», $\mathrm{b}(x+1,y) =$ «П», *то* $\mathrm{c}'(x,y) =$ «>», $\mathrm{e}'(x,y) =$ «r».

Выполнивший свою часть работы сигнал «#» стирается.

Если $\mathrm{e}(x,y) =$ «#», $\mathrm{c}(x+1,y) =$ «>», *то* $\mathrm{e}'(x,y) =$ «П».

Подойдя к правому краю кода, сигнал «>» добавляет к нему единичку.

Если $\mathrm{c}(x-1,y) =$ «>», $\mathrm{a}(x,y) =$ «П», *то* $\mathrm{a}'(x,y) =$ «1».

Остаётся пустить какой-то сигнал влево, который поставит на левом краю символ «>», причём, сделает это в тот момент, когда на правом краю появится сигнал «r». В следующий момент времени сигнал «r» поставит в третий слой ячейки символ «<», за это время сигнал «>» передастся вправо и окажется во второй ячейке, таким образом будет обеспечена синхронизация. В качестве сигнала, бегущего влево, используем уже имеющееся образование из «1» во втором слое и «/» в шестом слое.

Если $\mathrm{c}(x,y) =$ «>», $\mathrm{a}(x+1,y) =$ «П», *то* $\mathrm{b}'(x,y) =$ «1», $\mathrm{c}'(x,y) =$ «П», $\mathrm{f}'(x,y) =$ «/».

Далее им нужно разрешить двигаться по массиву из нулей и единиц первого слоя.

Если $\mathrm{a}(x,y) =$ «1» *или* $\mathrm{a}(x,y) =$ «0», $\mathrm{b}(x+1,y) =$ «1», $\mathrm{f}(x+1,y) =$ «/», *то* $\mathrm{b}'(x,y) =$ «1», $\mathrm{f}'(x,y) =$ «/».

При подходе к левому краю кода тандем формирует сигнал «>».

Если $\mathrm{a}(x-1,y) =$ «П», $\mathrm{a}(x,y) \neq$ «П», $\mathrm{b}(x,y) =$ «1», $\mathrm{f}(x,y) =$ «/», *то* $\mathrm{c}'(x,y) =$ «>».

Опишем правила движения сигналов пятого слоя «r» и «R».

Если $\mathrm{e}(x,y) =$ «r», *то* $\mathrm{e}'(x,y) =$ «П».

Если $\mathrm{e}(x-1,y) =$ «r», $\mathrm{a}(x,y) \neq$ «П», *то* $\mathrm{e}'(x,y) =$ «R».

Если $\mathrm{e}(x,y) =$ «R», *то* $\mathrm{e}'(x,y) =$ «r».

При подходе к правому краю «r» генерирует сигнал «<».

Если $\mathrm{e}(x,y) =$ «r», $\mathrm{a}(x+1,y) =$ «П», *то* $\mathrm{c}'(x,y) =$ «<», $\mathrm{e}'(x,y) =$ «П».

7. Обработка случаев $N = 2$ и $N = 3$.

И, наконец, ещё одно соображение. Дело в том, что в область применимости ОС σ_1 не входили числа 2 и 3. Здесь можно посвятить несколько строк этим двум случаям.

Случай $N = 2$.

Если $\mathrm{a}(x-1,y) = $ «>», $\mathrm{a}(x,y) = $ «<», $\mathrm{b}(x+1,y) = $ «П», $\mathrm{e}(x+1,y) = $ «П», $\mathrm{f}(x+1,y) = $ «П», *то* $\mathrm{a}'(x,y) = $ «П», $\mathrm{b}'(x,y) = $ «П», $\mathrm{c}'(x,y) = $ «Y», $\mathrm{e}'(x,y) = $ «П», $\mathrm{f}'(x,y) = $ «П».

Случай $N = 3$.

Если $\mathrm{a}(x-1,y) = $ «П», $\mathrm{b}(x-1,y) = $ «П», $\mathrm{c}(x-1,y) = $ «П», $\mathrm{e}(x-1,y) = $ «#», $\mathrm{f}(x-1,y) = $ «П», $\mathrm{a}(x,y) = $ «1», $\mathrm{b}(x,y) = $ «П», $\mathrm{c}(x,y) = $ «>», $\mathrm{d}(x,y) = $ «П», $\mathrm{e}(x,y) = $ «r», $\mathrm{f}(x,y) = $ «П», $\mathrm{a}(x+1,y) = $ «П», $\mathrm{b}(x+1,y) = $ «П», $\mathrm{c}(x+1,y) = $ «П», $\mathrm{e}(x+1,y) = $ «П», $\mathrm{f}(x+1,y) = $ «П», *то* $\mathrm{a}'(x,y) = $ «П», $\mathrm{b}'(x,y) = $ «П», $\mathrm{c}'(x,y) = $ «Y», $\mathrm{e}'(x,y) = $ «П», $\mathrm{f}'(x,y) = $ «П».

Далее символ «Y» двигается вправо, пока не выйдет на границу.

Если $\mathrm{c}(x-1,y) = $ «Y», $\mathrm{a}(x,y) \neq $ «П», *то* $\mathrm{a}'(x,y) = $ «П», $\mathrm{c}'(x,y) = \mathrm{c}(x-1,y)$.

Описание алгоритма закончено.

Замечание. Отметим, что функция переходов ОС была представлена в виде цепочки условий и следствий. Эта цепочка получена автоматически по программе ОС, написанной на некотором языке. Программа, а следовательно, и цепочка условий, имеет некоторые особенности. Именно, она выполняется последовательно сверху вниз. То есть, одновременно могут быть выполнены несколько условий, причём, они могут как дополнять, так и противоречить друг другу. В случае противоречия, в действительности, выполняется самое последнее условие.

Приведём основные характеристики ОС σ_2. Начнём с числа состояний ячейки. 213 состояний ОС σ_1, естественно, сохраняются, и к ним ещё добавляются 27 новых состояний, возникших при трансформировании кода. Окончательно, получим 240 различных состояний.

Далее найдём размер максимальной конфигурации. Здесь мы воспользуемся уже имеющимся результатом для ОС σ_1. Тогда для ОС σ_2 будет иметь место оценка $\max\{\frac{n^3}{4} + 5\frac{n^2}{4} + 2n, N\}$, где, понятно, $n = [\log(N)] + 1$. Легко видеть, что почти для всех N максимальной конфигурацией будет начальная.

Оценим время работы ОС σ_2. Оно, несомненно, в случае нечётного N, будет складываться из времени, затраченного на преобразование палочного кода в двоичный, и времени функционирования ОС σ_1. Посчитаем, чему равно первое время.

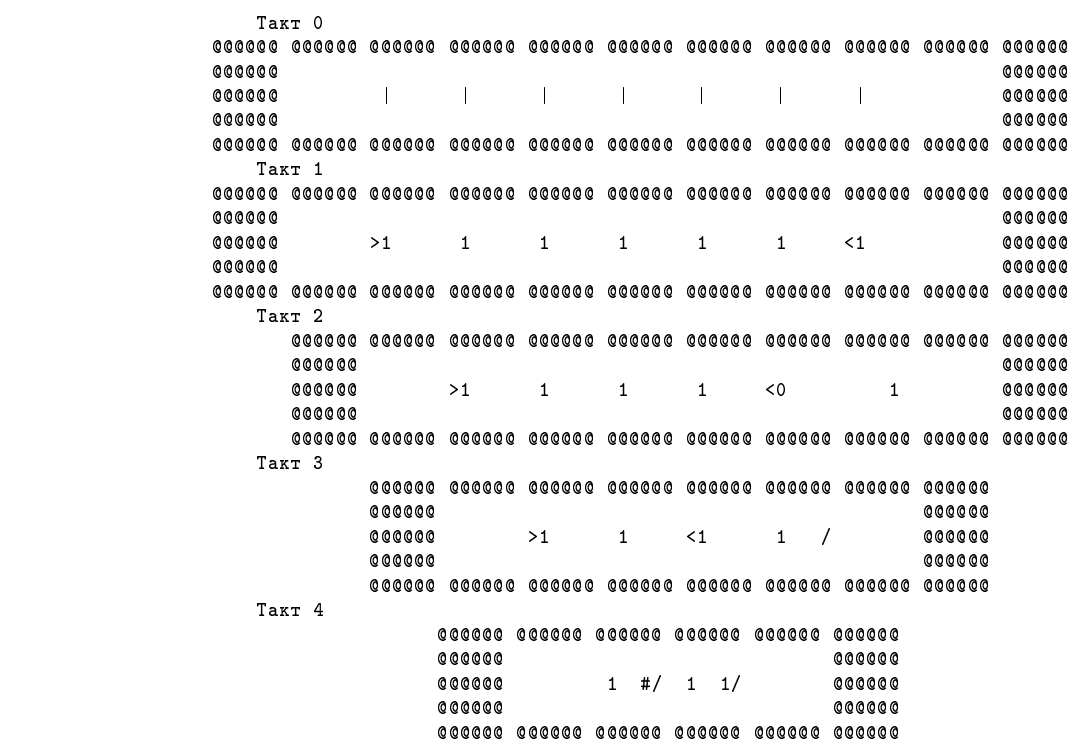
Формирование сигнала «#» происходит на $[\frac{N}{2}]+1$ такте. Для копирования кода из второго слоя в первый надо $2[\log(N)]-1$ тактов. На достижение сигналом «r» правой части кода и постановку им символа «<» необходимо $2[\log(N)]+1$ тактов. Всего на перекодирование надо потратить $[\frac{N}{2}]+4[\log(N)]+1$ тактов времени (или $[\frac{N}{2}]+4n-3$).

Таким образом, для нечётного N время работы ОС σ_2 будет меньше $\frac{N}{2}+2\sqrt{N}+\frac{n^2}{2}+9n+4$.

Для чётных N время работы ещё меньше, оно ограничено сверху числом $\frac{N}{2}+n$.

Итак, ОС σ_2 полностью описана, тем самым, лемма 3 доказана.

Приведём наглядный пример функционирования ОС σ_2 для $N=7$.



```
Такт 5
              @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
              @@@@@@                               @@@@@@
              @@@@@@             #  11   /         @@@@@@
              @@@@@@                               @@@@@@
              @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 6
              @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
              @@@@@@                               @@@@@@
              @@@@@@          1  #/ 1              @@@@@@
              @@@@@@                               @@@@@@
              @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 7
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
         @@@@@@                               @@@@@@
         @@@@@@             #  1 > r  1          @@@@@@
         @@@@@@                               @@@@@@
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 8
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
         @@@@@@                          @@@@@@
         @@@@@@        1      1 > R      @@@@@@
         @@@@@@                          @@@@@@
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 9
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
         @@@@@@                               @@@@@@
         @@@@@@        1      11  r/ 1          @@@@@@
         @@@@@@                               @@@@@@
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 10
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
         @@@@@@                               @@@@@@
         @@@@@@        11   / 1      1   R      @@@@@@
         @@@@@@                               @@@@@@
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 11
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
         @@@@@@                               @@@@@@
         @@@@@@        1 >    1      1   r      @@@@@@
         @@@@@@                               @@@@@@
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 12
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
         @@@@@@                               @@@@@@
         @@@@@@        1      1 >    1 <        @@@@@@
         @@@@@@                               @@@@@@
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
```

```
Такт 13
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@        1      1      1 #           @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 14
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@        1      1<<    1 x           @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 15
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@        1 <    1#     1 x           @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 16
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@        1<X    1      1 x           @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 17
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@        116    1      1 x           @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 18
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@        1191   1 6    1 x           @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 19
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@        1181   10o11  1 5           @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 20
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@        118    10 1   1  p          @@@@@@
        @@@@@@        1 |p   1 ops1   y          @@@@@@
        @@@@@@                                     @@@@@@
        @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
```

```
Такт 21
       @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
       @@@@@@                                     @@@@@@
       @@@@@@        11?    10     1  1          @@@@@@
       @@@@@@        1 |1 0 11o1s    x          @@@@@@
       @@@@@@                                     @@@@@@
       @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 22
       @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
       @@@@@@                                     @@@@@@
       @@@@@@        11     10?    1            @@@@@@
       @@@@@@        1181   11 1   1  p         @@@@@@
       @@@@@@        1 |p   1 mp     u          @@@@@@
       @@@@@@                                     @@@@@@
       @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 23
       @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
       @@@@@@                                     @@@@@@
       @@@@@@        11     100    10           @@@@@@ @@@@@@
       @@@@@@        118    11 1   1  1                @@@@@@
       @@@@@@        1 |1   1 m1     l      r          @@@@@@
       @@@@@@                 v                      @@@@@@
       @@@@@@ @@@@@@                        @@@@@@ @@@@@@ @@@@@@
       @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 24
       @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
       @@@@@@                                     @@@@@@
       @@@@@@        110    10     10           @@@@@@ @@@@@@
       @@@@@@        11?    11     1  1                @@@@@@
       @@@@@@        1 |1     l      l      r          @@@@@@
       @@@@@@        1 |p     u                        @@@@@@
       @@@@@@                                @@@@@@ @@@@@@ @@@@@@
       @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 25
       @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
       @@@@@@                                     @@@@@@
       @@@@@@        0 !    11     10           @@@@@@ @@@@@@
       @@@@@@        11     11?    1                  @@@@@@
       @@@@@@          l      l      l      r          @@@@@@
       @@@@@@        1 |1     l                       @@@@@@
       @@@@@@                                @@@@@@ @@@@@@ @@@@@@
       @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 26
              @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
       @@@@@@ @@@@@@                              @@@@@@
       @@@@@@               11?    10           @@@@@@ @@@@@@
       @@@@@@        11     110    11                 @@@@@@
       @@@@@@          l      l      l      r          @@@@@@
       @@@@@@          l                              @@@@@@
       @@@@@@                        @@@@@@ @@@@@@ @@@@@@ @@@@@@
       @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
```

```
Такт 27
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
         @@@@@@ @@@@@@                             @@@@@@
         @@@@@@               11     10?           @@@@@@ @@@@@@
         @@@@@@        110    01     11                  @@@@@@
         @@@@@@          l      l      l      r           @@@@@@
         @@@@@@                                           @@@@@@
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 28
                @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
         @@@@@@ @@@@@@                             @@@@@@
         @@@@@@               11       Y           @@@@@@ @@@@@@
         @@@@@@        0 !    01     11                  @@@@@@
         @@@@@@          l      l      l      r           @@@@@@
         @@@@@@                                           @@@@@@
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 29
         @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
         @@@@@@                                 @@@@@@
   @@@@@@ @@@@@@          U      Y            @@@@@@ @@@@@@
   @@@@@@              01?    11                    @@@@@@
   @@@@@@          l      l      l      r           @@@@@@
   @@@@@@                                           @@@@@@
   @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 30
                @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
          @@@@@@ @@@@@@                         @@@@@@
   @@@@@@ @@@@@@                   Y            @@@@@@ @@@@@@
   @@@@@@               1     11B                   @@@@@@
   @@@@@@          l      l      l      r           @@@@@@
   @@@@@@                                           @@@@@@
   @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 31
                @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
          @@@@@@ @@@@@@                         @@@@@@
   @@@@@@ @@@@@@                   Y            @@@@@@ @@@@@@
   @@@@@@               1       Y                   @@@@@@
   @@@@@@          l      l      l      r           @@@@@@
   @@@@@@                                           @@@@@@
   @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 32
          @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
   @@@@@@ @@@@@@                               @@@@@@ @@@@@@
   @@@@@@               U      Y                   @@@@@@
   @@@@@@          l      l      l      r           @@@@@@
   @@@@@@                                           @@@@@@
   @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
```

```
Такт 33
                @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
   @@@@@@ @@@@@@ @@@@@@                     @@@@@@ @@@@@@
   @@@@@@                       Y                  @@@@@@
   @@@@@@          l      l     l      r           @@@@@@
   @@@@@@                                         @@@@@@
   @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 34
                @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
   @@@@@@ @@@@@@ @@@@@@                     @@@@@@ @@@@@@
   @@@@@@                       Y                  @@@@@@
   @@@@@@          l      l     D      r           @@@@@@
   @@@@@@                                         @@@@@@
   @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 35
                @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
   @@@@@@ @@@@@@ @@@@@@                     @@@@@@ @@@@@@
   @@@@@@                       Y                  @@@@@@
   @@@@@@          l      D             D           @@@@@@
   @@@@@@                                         @@@@@@
   @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 36
                @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
   @@@@@@ @@@@@@ @@@@@@                     @@@@@@
   @@@@@@                       Y           @@@@@@
   @@@@@@          D                        @@@@@@
   @@@@@@                     @@@@@@ @@@@@@ @@@@@@
   @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
Такт 37
                @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
                @@@@@@                     @@@@@@
                @@@@@@          Y          @@@@@@
                @@@@@@                     @@@@@@
                @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@
```

4. Основная теорема. Следствия

Из лемм 1, 2, 3 получаем основную теорему.

Теорема 1. $T(N) \sim \frac{N}{2}$.

Если внимательно проследить за построением ОС σ_2, можно заметить, что при перекодировке натурального числа из палочного в двоичное представление нигде не используется тот факт, что ОС σ_2 — плоская однородная структура, то есть линейная однородная структура с шаблоном соседства $\{-1, 0, 1\}$ и с функцией перехода, полученной из уже описанной функции переходов для ОС σ_2 выбрасыванием

несущественных переменных, будет также осуществлять перекодировку. Далее, лемма 2 верна и для линейных ОС. Теперь можно по аналогии рассмотреть задачу проверки простоты натурального числа в классе линейных однородных структур. Обозначим функцию временной сложности для этой задачи $L(N)$.

Следствие 1. $L(N) \sim \frac{N}{2}$.

Пусть $\mathbf{M}$ — массовая задача, которая состоит в получении ответа «Да» или «Нет» на вопрос о наличии какого-либо свойства натурального числа. Пусть $I \in \mathbf{M}$ — индивидуальная задача, то есть проверка свойства уже конкретного натурального числа N. Пусть длина задачи — длина двоичной записи числа N. Предположим, что массовая задача решается за время $\mathrm{o}(N)$, то есть существует машина Тьюринга T, решающая задачу $\mathbf{M}$, и такая, что время её работы $T(N) = \mathrm{o}(N)$. Рассмотрим следующую постановку задачи $\mathbf{M}$. Берём линейные однородные структуры с шаблоном соседства $\{-1, 0, 1\}$. Скажем, что линейная ОС решает задачу $\mathbf{M}$, если она решает все её индивидуальные задачи, то есть, после задания начальной конфигурации — палочного представления натурального числа, ОС за конечное число шагов перейдёт в состояние — конфигурацию из единственного символа «Y» или «N», в зависимости от того, обладает или нет натуральное число свойством, в проверке которого заключается задача $\mathbf{M}$. Тогда справедливо следующее утверждение.

Следствие 2. Для любой такой задачи $\mathbf{M}$ существует линейная однородная структура σ с шаблоном соседства $\{-1, 0, 1\}$, решающая её за время $L_\sigma(N) = \frac{N}{2} + \mathrm{o}(N)$.

Автор выражает благодарность академику В.Б. Кудрявцеву за постановку задачи и ценные советы по её решению.

Список литературы

[1] Кудрявцев В.Б., Алёшин С.В., Подколзин А.С. Введение в теорию автоматов. М.: Наука, 1985.

[2] Кудрявцев В.Б., Подколзин А.С., Болотов А.А. Основы теории однородных структур. М.: Наука, 1990.

[3] Василенко О.Н. Современные способы проверки простоты чисел. Обзор // Кибернетический сборник. Нов. сер. Вып. 25. С. 162–188.