ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА»

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА МАТЕМАТИЧЕСКОЙ ТЕОРИИ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(ДИПЛОМНАЯ РАБОТА)
специалиста

«ОБ ЭФФЕКТИВНОСТИ ПОРОЖДЕНИЯ СЛУЧАЙНЫХ КОНЕЧНЫХ КВАЗИГРУПП»

Выполнил студент
группы 611
Жигляев Родион Алексеевич

_____
подпись студента

Научный руководитель:
к.ф.-м.н., с.н.с.
Галатенко Алексей Владимирович

_____
подпись научного
руководителя

Москва
2023

Аннотация

В данной работе представляется программная реализация нескольких алгоритмов порождения случайных конечных квазигрупп. А именно, метод Джейкобсона-Мэтьюза и методы генерации функционально заданных квазигрупп, основанные на правильных семействах функций, регистрах сдвига и сетях Фейстеля. Приводится сравнение времени работы этих методов. Порождаемые этими методами квазигруппы проверяются на наличие подквазигрупп и полиномиальной полноты. Приводится программная реализация алгоритмов проверки простоты и аффинности. Для алгоритма проверки аффинности приводится оптимизированная версия. В работе также предложен способ порождения n-квазигрупп с помощью полных перестановок. Найдены условия, при которых квазигруппы на основе обобщенных регистров сдвига и обобщенных сетей Фейстеля имеют тривиальную подквазигруппу.

This paper presents a software implementation of several algorithms for generating random finite quasigroups. Namely, the Jacobson-Matthews method and methods for generating functionally defined quasigroups based on proper families of functions, shift registers and Feistel networks. A comparison of the running time of these methods is given. The quasigroups generated by these methods are checked for the presence of subquasigroups and for polynomial completeness. A software implementation of algorithms for checking simplicity and affinity is given. An optimized version is given for the affinity check algorithm. The paper also proposes a method for generating n-quasigroups using complete mappings. Conditions are found under which quasigroups based on generalized shift registers and generalized Feistel networks have a trivial subquasigroup.

Содержание

1 Введение **2**

2 Основные определения **4**

3 Порождение квазигруппы методом Джейкобсона-Мэтьюза **7**
3.1 Описание алгоритма 7
3.2 Описание программной реализации 8

4 Порождение квазигруппы правильными семействами **8**
4.1 Описание алгоритма 9
4.2 Описание программной реализации 10

5 Порождение квазигруппы перестановками с полным дифференциалом **11**
5.1 Описание алгоритма 11
5.2 О неподвижных точках 13
5.2.1 Задание квазигруппы формулой $\sigma(x+y)-y$ 13
5.2.2 Задание квазигруппы формулой $\sigma(x-y)+y$ 16
5.3 Обобщение для задания n-квазигрупп 18

6 Проверка аффинности **18**
6.1 Описание алгоритма 18
6.2 Описание программной реализации 19

7 Проверка простоты **20**
7.1 Описание алгоритма 21
7.2 Описание программной реализации 21

8 Алгоритм нахождения подквазигрупп **22**

9 Эксперименты **23**
9.1 Сравнение времени работы алгоритмов генерации 23
9.2 Скорость проверки аффинности 24
9.3 Скорость проверки простоты 26
9.4 Проверка равномерности порождения на множестве правильных семейств 27
9.5 Проверка полиномиальной полноты квазигрупп на основе правильных семейств 28
9.6 Проверка наличия подквазигрупп у квазигрупп на основе правильных семейств 28
9.7 Проверка полиномиальной полноты квазигрупп на основе перестановок с полным дифференциалом 29
9.8 Проверка наличия подквазигрупп у квазигрупп на основе перестановок с полным дифференциалом 30
9.9 Проверка наличия подквазигрупп в общем случае 32

10 Заключение **33**

Список литературы **34**

1 Введение

Для построения криптографических алгоритмов активно используются некоммутативные и неассоциативные алгебраические структуры [1]. Отдельный интерес среди таких структур представляют конечные квазигруппы. Например, квазигруппы используются в алгоритмах GAGE и InGAGE [2], которые были участниками конкурса Lightweight Cryptography от NIST. Среди заявленных в конкурсе SHA-3 также были квазигрупповые кандидаты Edon-R' [3] и NaSHA [4]. На «многомерных квадратичных квазигруппах» основана сверхбыстрая схема цифровой подписи MQQ-SIG [5], которая на три порядка быстрее, чем RSA и ECDSA. Табличное гаммирование, основанное на квазигруппах, обладает свойством совершенной секретности, что было доказано Шенноном [6]. В работе [7] приводится обзор применения квазигрупп в построении однонаправленных функций, A-кодов, и методов шифрования. Один из таких методов, например, получается путем комбинирования поточного квазигруппового шифра с шифром Эль-Гамаля.

Известно, что почти все квазигруппы обладают свойством полиномиальной полноты [8]. Это значит, что на практике разумно выбирать случайные квазигруппы. С высоким шансом случайная квазигруппа окажется полиномиально полной, что гарантирует NP-полноту задачи проверки разрешимости уравнений и систем уравнений [9], [10].

Ряд криптографических примитивов, построенных на конечных квазигруппах, требуют квазигруппы больших размеров. К ним относится, например, ранее упомянутое семейство алгоритмов NaSHA. Это требование делает применение таких примитивов недопустимым в тех системах, размеры ключей которых должны быть небольшими. Более того, слишком большая квазигруппа может попросту не поместиться в память компьютера. В связи с этим актуальным остается вопрос рассмотрения отдельных классов квазигрупп, которые можно задавать без вычисления всего латинского квадрата, а, например, функционально.

При построении новых классов, однако, вообще говоря, нельзя воспользоваться утверждением, что почти все квазигруппы полиномиально полны. Поэтому функциональные классы квазигрупп требуют дополнительных исследований на предмет наличия полезных свойств.

Еще один важный вопрос — есть ли у квазигруппы подквазигруппа. Наличие подквазигруппы может упростить для потенциального злоумышленника взлом криптоалгоритма.

В задачи данной работы входит разработка программы, которая способна порождать случайные конечные квазигруппы с помощью некоторых известных методов, а также проверять порожденные объекты на полиномиальную полноту и наличие подквазигрупп. Среди методов порождения рассматривается метод Джейкобсона-Мэтьюза, а также методы порождения на основе правильных семейств функций, регистров сдвига и обобщенных сетей Фейстеля. Приводится сравнительная характеристика этих методов на основе практически значимых факторов, таких как время работы алгоритма, мощность получаемого множества, наличие полиномиальной полноты и подквазигрупп. Полиномиальная полнота, в свою очередь, проверяется как наличие в отдельности свойств простоты и неаффинности. При поиске подквазигрупп отдельно учитываются наличие тривиальных и нетривиальных подквазигрупп. В работе представлен способ задания n-квазигрупп с помощью полных перестановок. Найдены условия, при которых квазигруппы на основе регистров сдвига и обобщенных сетей Фейстеля содержат тривиальную подквазигруппу. Для алгоритма проверки аффинности был разработан модифицированный вариант, уменьшающий сложность алгоритма.

Программа написана на языке C++. Все эксперименты проводились на компьютере с процессором Intel(R) Core(TM) i5-11400H @ 2.70GHz и 8ГБ оперативной памяти.

Работа устроена следующим образом: в разделе 2 даны основные определения; в разделе 3 описывается порождение случайных конечных квазигрупп алгоритмом Джейкобсона-Мэтьюза; в разделе 4 описано порождение квазигрупп правильными семействами функций; в разделе 5 описано порождение квазигрупп с помощью перестановок с полным дифференциалом; в разделе 6

описан алгоритм проверки аффинности; в разделе 7 описан алгоритм проверки простоты; в разделе 8 описывается алгоритм для нахождения подквазигрупп; в разделе 9 показаны эксперименты, которые проводились с программой; в разделе 10 подводятся итоги работы.

2 Основные определения

Определение 1. *Конечное множество Q, на котором задана бинарная операция умножения f_Q: $Q \times Q \to Q$, такая, что для любых элементов $a, b \in Q$ уравнения $f_Q(a, x) = b$ и $f_Q(y, a) = b$ однозначно разрешимы в Q, называется конечной квазигруппой. Операцию f_Q будем называть квазигрупповой.*

Операцию f_Q также можно задать с помощью таблицы умножения, которая представляет собой латинский квадрат.

Квазигруппы можно обобщить до понятия n-квазигрупп.

Определение 2. *Для натурального $n \geq 3$ под n-квазигруппой понимается множество Q, на котором определена n-арная операция f_Q, такая что для любых элементов $a_1, a_2, ..., a_n, b \in Q$ все уравнения*

$$\begin{gathered} f_Q(x, a_2, a_3, ..., a_n) = b, \\ f_Q(a_1, x, a_3, ..., a_n) = b, \\ . \\ . \\ . \\ f_Q(a_1, a_2, ..., a_{n-1}, x) = b \end{gathered}$$

однозначно разрешимы в Q.

Далее говоря о квазигруппах будем подразумевать 2-квазигруппы, за исключением раздела 5.3.

Определение 3. *Квазигруппа (Q, f_Q) называется аффинной, если на множестве Q можно ввести структуру абелевой группы $(Q, +)$, такую, что существуют автоморфизмы α, β группы $(Q, +)$ и элемент $c \in Q$, для которых выполнено тождество*

$$f_Q(x, y) = \alpha(x) + \beta(y) + c.$$

Рассмотрим множество элементов квазигруппы $Q = \{q_1, ..., q_N\}$, $N \geq 2$ и некоторое разбиение α множества Q в объединение непересекающихся подмножеств $Q = A_1 \sqcup ... \sqcup A_m$. Будем называть разбиение α нетривиальным, если $m > 1$, $A_i \neq \emptyset$, $i = 1, ..., m$, и существует индекс j, $1 \leq j \leq m$, такой, что $|A_j| > 1$. В случае если $|A_1| = ... = |A_m|$, такое нетривиальное разбиение будем называть равномерным. Элементы a, b, которые принадлежат одному множеству A_i, далее назовем эквивалентными и будем использовать запись $a \sim b$.

Будем говорить, что f_Q сохраняет разбиение α, если для любой пары наборов (a_1, b_1), $(a_2, b_2) \in Q^2$, таких, что $a_i \sim b_i$, $i = 1, 2$, выполнено $f_Q(a_1, a_2) \sim f_Q(b_1, b_2)$. Как можно заметить, квазигрупповые операции могут сохранять только равномерные разбиения.

Определение 4. *Квазигруппа (Q, f_Q) называется простой, если операция f_Q не сохраняет никакое нетривиальное разбиение Q.*

Для фиксированного (конечного) множества A обозначим через $\mathcal{O}_n(A)$ совокупность всех n-арных операций на A ($n \geq 0$). Под 0-арными операциями будем подразумевать константы. Пусть $\mathcal{O}(A) = \bigcup_{n=0}^{\infty} \mathcal{O}_n(A)$. Далее под множеством A понимается множество элементов квазигруппы, поэтому будем использовать упрощённую запись: $\mathcal{O}_n$ и $\mathcal{O}$.

На произвольном подмножестве $F \subseteq \mathcal{O}(A)$ можно стандартным образом ввести операции суперпозиции и замыкания [11]. Обозначим замыкание множества F через $[F]$.

Определение 5. *Квазигруппа Q называется полиномиально полной, если $[f_Q \cup \mathcal{O}_0] = \mathcal{O}$.*

Известно, что полиномиальная полнота эквивалентна одновременной простоте и неаффинности [12].

Определение 6. *Пусть задана квазигруппа (Q, f_Q) и $Q' \subset Q$, $1 \leq |Q'| < |Q|$. Если для любых элементов $a, b \in Q'$ верно, что $f_Q(a, b) \in Q'$, то будем говорить, что квазигруппа (Q, f_Q) содержит собственную подквазигруппу $(Q', f_{Q'})$, где $f_{Q'}$ — ограничение операции f_Q на $Q' \times Q'$. Подквазигруппы порядка 1 будем называть тривиальными. Подквазигруппы порядка 2 и больше будем называть нетривиальными.*

Рассмотрим квазигруппу с носителем Q, $|Q| = k^n$, где k, n натуральные числа, $k \geq 2$. Пронумеруем все элементы квазигруппы числами от 0 до $k^n - 1$, а каждому номеру сопоставим его кодировку в k-ичной системе счисления. Далее будем отождествлять элемент x с его k-ичной кодировкой $(x_1, ..., x_n)$. Операцию умножения в таких квазигруппах можно задать функционально. Рассмотрим несколько таких способов задания.

Первый из них — задание с помощью правильного семейства $(g_1, ..., g_n)$ n-местных функций k-значной логики. Правильные семейства и способ получениях из них квазигрупп были введены в работах [13]—[15].

Определение 7. *Семейство $(g_1, ..., g_n)$ n-местных функций k-значной логики будем называть правильным, если для любой пары различных наборов $s = (s_1, ..., s_n)$, $t = (t_1, ..., t_n)$ из E_k^n найдется номер $1 \leq i \leq n$, такой, что $s_i \neq t_i$, но $g_i(s) = g_i(t)$.*

Для любых двух элементов квазигруппы $x = (x_1, ..., x_n)$, $y = (y_1, ..., y_n)$ произведение $f_Q(x, y) = (f_1, ..., f_n)$ этих элементов можно получить по формулам:

$$f_i = x_i + y_i + g_i(\pi_1(x_1, y_1), ..., \pi_n(x_n, y_n)),\ 1 \leq i \leq n.$$

Здесь сложение ведется по модулю k, а функции $\pi_1, ..., \pi_n$ — произвольные двухместные функции k-значной логики.

Еще один способ функционального задания квазигрупп — полные перестановки.

Определение 8. *Пусть задана конечная абелева группа $(G;+)$, а σ некоторая перестановка на множестве G. Перестановку σ будем называть полной над группой $(G;+)$, если функция $\sigma(x) - x$ является биекцией.*

Полные перестановки также будем называть перестановками с полным дифференциалом. Связь между квазигрупповыми операциями и полными перестановками задается следующим образом:

$$f_Q(x,y) = \sigma(x \pm y) \mp y.$$

Эта связь была установлена в [16]. Отметим также конструкции, которые будем использовать в дальнейшем. Будем рассматривать сценарий, когда группа задана прямым произведением $n > 1$ одинаковых сомножителей $(G_0,+)$, $|G_0| = k$. Следующее соотношение будем называть обобщенным регистром сдвига с обратной связью:

$$\begin{cases} f_1 = x_2 + c_1 \\ f_2 = x_3 + c_2 \\ ... \\ f_{n-1} = x_n + c_{n-1} \\ f_n = x_1 + g(x_2, ..., x_n) + c_n \end{cases}$$

Здесь $c_1, ..., c_n$ — произвольные константы. Квазигруппы на основе регистров сдвига можно задать следующим образом:

$$\begin{cases} f_1 = x_2 \pm y_2 \mp y_1 + c_1 \\ f_2 = x_3 \pm y_3 \mp y_2 + c_2 \\ ... \\ f_{n-1} = x_n \pm y_n \mp y_{n-1} + c_{n-1} \\ f_n = x_1 \pm y_1 + g(x_2 \pm y_2, ..., x_n \pm y_n) \mp y_n + c_n \end{cases}$$

Для случая, когда $n = 2$, рассмотрим также конструкцию, называемую обобщенной сетью Фейстеля:

$$\begin{cases} f_1 = s(x_2) \\ f_2 = x_1 + p(x_2) \end{cases}$$

Квазигруппу из такой конструкции можно получить по формулам:

$$\begin{cases} f_1 = s(x_2 \pm y_2) \mp y_1 \\ f_2 = x_1 \pm y_1 + p(x_2 \pm y_2) \mp y_2 \end{cases}$$

Обе конструкции, однако, не всегда позволяют получить квазигруппы. Условия на функции g, s, p, при которых эти конструкции задают полные перестановки, были даны в [17]. Отметим, что под термином «обобщенные/расширенные сети Фейстеля (generalized/extended Feistel networks)» могут подразумеваться разные конструкции. Например, разные варианты обобщения рассматривались в [18] и в [19]. Говоря об обобщенной сети Фейстеля, будем считать, что это конструкция из работы [17].

При оценке сложности алгоритмов будем считать элементарными арифметические операции и обращение к ячейкам памяти компьютера.

3 Порождение квазигруппы методом Джейкобсона-Мэтьюза

Квазигруппу можно задать не только латинским квадратом, но и правильным кубом инцидентности — трёхмерной матрицей, у которой в каждой строке каждого измерения имеется ровно одна единица, а все остальные элементы равны нулю.

Один из способов генерации конечных квазигрупп был предложен в [20]. Он основан на цепочке преобразований кубов инцидентности.

3.1 Описание алгоритма

Пусть $N \in \mathbb{N}$ порядок квазигруппы, которую необходимо получить. Выберем на старте алгоритма куб инцидентности M размера N^3 и зададим его следующим образом: $M_{ijk} = 1$ при $k \equiv i + j \pmod N$, $1 \leq i, j, k \leq N$ и $M_{ijk} = 0$ иначе. Такому кубу соответствует квазигруппа (Q, f_Q), такая, что $Q = \{q_1, \ldots, q_N\}$ — множество элементов квазигруппы и $f_Q(q_i, q_j) = q_k$. Назовем куб инцидентности неправильным, если:

1) он состоит только из 0, 1 и -1;

2) элемент со значением -1 ровно один;

3) в каждой строке каждого измерения сумма всех элементов равна 1.

Каждый шаг алгоритма является преобразованием куба инцидентности. При этом на каждом шаге могут получаться только правильные или неправильные кубы. Шаги отличаются в зависимости от того, каким является текущий куб.

1) Если на текущем шаге куб инцидентности является правильным, то выберем в нём случайную клетку, значение в которой равно нулю. Пусть координаты этой клетки (x, y, z). В строках $M_{iyz}, M_{xjz}, M_{xyk}$ $1 \leq i, j, k \leq N$ выберем значения i_1, j_1, k_1 таким образом, что $M_{i_1yz} = M_{xj_1z} = M_{xyk_1} = 1$.

2) Если на текущем шаге куб инцидентности неправильный, то выберем в нём клетку со значением -1 и обозначим координаты этой клетки (x, y, z). В строках $M_{iyz}, M_{xjz}, M_{xyk}$ $1 \leq i, j, k \leq N$ по определению присутствуют по две клетки со значением 1. Для каждой из этих строк выбираем по одной случайной единице из двух. Обозначим их $M_{i_1yz}, M_{xj_1z}, M_{xyk_1}$.

После выбора необходимых значений вычтем единицы из $M_{i_1yz}, M_{xj_1z}, M_{xyk_1}$, $M_{i_1j_1k_1}$ и добавим единицы к $M_{xyz}, M_{xj_1k_1}, M_{i_1yk_1}$, $M_{i_1j_1z}$. Если в результате получилось, что $M_{ijk} = -1$, то объявим куб инцидентности неправильным. В противном случае объявим куб правильным. Далее переходим к следующему шагу.

Теорема 1 ([20]). *Пусть $N \geq 2$. Для любых двух (правильных или неправильных) кубов размера N^3 существует последовательность шагов, которая*

преобразует один куб в другой. Верхняя граница длины кратчайшей такой последовательности равна $2(N-1)^3$.

3.2 Описание программной реализации

Из теоремы выше непосредственно следует, что от стартовой квазигруппы до любой другой можно добраться за $2(N-1)^3$ шагов алгоритма. Возьмем $2(N-1)^3$ как число шагов алгоритма. Рассмотрим программную реализацию.

На каждом шаге выполняется обращение к ячейкам памяти, а также добавление, вычитание и поиск единицы. Обращение к ячейкам памяти и арифметические операции выполняются с константной сложностью. Для того чтобы поиск единиц также выполнялся с константной сложностью, куб инцидентности хранится в памяти в виде трёх двумерных массивов $unitXY$, $unitXZ$, $unitYZ$, элементы которых соответствуют строкам куба и численно равны координатам единиц в этих строках. Например, при необходимости найти единицу в строке M_{ijk}, где i, j фиксированы, искомая координата $k = unitXY[i][j]$. Таким образом, задача поиска единицы также сводится к обращению к ячейке памяти и выполняется с константной сложностью. Элементы массива $unitXY$ с индексами i, j по факту являются элементами генерируемой квазигруппы, находящимися в i-й строке и j-м столбце соответствующего латинского квадрата. Добавление и удаление единиц сводится к переопределению значений в массивах.

Такой подход, однако, не позволяет учитывать наличие в строке двух единиц и минус единицы. Чтобы это учесть, в случае перехода куба M в неправильный вид, координаты ячеек, содержащих вторые единицы и минус единицу, сохраняются в отдельных переменных. В тот момент, когда происходит случайный выбор между единицами, одну из которых нужно обнулить, возможны два варианта.

1) Обнулить необходимо ту единицу, которая хранится в отдельной переменной. Тогда в кубе хранятся координаты той единицы, которая не будет обнулена после выполнения шага, и эти координаты изменять не нужно.

2) Обнулить необходимо единицу, которая хранится в кубе. Тогда нужно поменять местами эти две единицы. Ту, что была в кубе, сохранить в отдельной переменной, а ту, что была в отдельной переменной, сохранить в кубе. Тем самым, задача свелась к первому пункту.

Утверждение 1. *Временная сложность одного шага алгоритма Джейкобсона-Мэтьюза* $O(1)$. *Общая временная сложность алгоритма* $O(N^3)$, *где* N — *порядок квазигруппы.*

4 Порождение квазигруппы правильными семействами

В данном разделе описывается алгоритм порождения случайных правильных семейств, полученный из результатов работ [21] и [22].

4.1 Описание алгоритма

На старте работы алгоритма возьмем произвольное правильное семейство $(g_1, ..., g_n)$ n-местных функций k-значной логики. Сгодится, например, такое семейство, что $g_i \equiv \text{const}$, $1 \leq i \leq n$. Здесь можно взять любые случайные константы от 0 до $k-1$, причем для каждой функции свою. Несложно видеть, что семейство таких функций является правильным.

Пусть на текущем шаге имеется некоторое правильное семейство $(g_1, ..., g_n)$. Получим из него новое правильное семейство по следующему алгоритму.

1) Сгенерируем случайное число $1 \leq i \leq n$.

2) Поменяем местами функции g_i с g_n и переменные x_i с x_n. Далее во всех записях будем подразумевать, что перестановка уже проведена.

3) Сформируем k семейств $G_0 = (g_1^0, ..., g_{n-1}^0), ..., G_{k-1} = (g_1^{k-1}, ..., g_{n-1}^{k-1})$. Здесь $g_j^m(x_1, ..., x_{n-1}) = g_j(x_1, ..., x_{n-1}, m)$. Иными словами, мы удаляем последнюю переменную и последнюю функцию и каждое семейство G_m составляем из оставшихся функций, которым на место последней переменной поставили число m, $0 \leq m \leq k-1$.

4) Зададим первые $n-1$ функцию нового семейства по формуле:

$$g_j^{'}(x_1, ..., x_n) = \bigvee_{m=0}^{k-1}(I_{x_n}(m) \wedge g_j^m(x_1, ..., x_{n-1})),\ 1 \leq j \leq n-1.$$

Здесь под $\vee$ будем подразумевать операцию взятия максимума, под $\wedge$ взятие минимума, а функции $I_{x_n}(m)$ имеют вид:

$$I_{x_n}(m) = \begin{cases} k-1, & x_n = m \\ 0, & \text{иначе.} \end{cases}$$

5) Построим граф, в котором вершинами будут всевозможные наборы длины $n-1$ с элементами из E_k. Две различные вершины $s = (s_1, ..., s_{n-1})$ и $t = (t_1, ..., t_{n-1})$ соединим ребром, если можно найти такие номера $0 \leq j < m \leq k-1$, что $G_j(s)$ отличается от $G_m(t)$ во всех позициях, в которых s отличается от t.

6) Вычислим компоненты связности построенного графа.

7) На каждой из компонент связности будем задавать $g_n^{'}$ случайным значением от 0 до $k-1$ независимо друг от друга.

8) Снова поменяем местами функции g_i с g_n и переменные x_i с x_n.

После выполнения всех пунктов мы получим новое правильное семейство функций $(g_1^{'}, ..., g_n^{'})$. С новым семейством переходим к следующему шагу алгоритма.

После выполнения необходимого числа шагов мы получим случайное правильное семейство функций. В качестве необходимого числа шагов в дальнейшем будем брать порядок квазигруппы, возведенный в квадрат. Как будет видно из экспериментов, такого числа шагов будет достаточно.

Описанный алгоритм в пределе порождает равномерное распределение на множестве правильных семейств n-местных функций k-значной логики. Доказательство этого факта вытекает из результатов работ [21] и [22].

4.2 Описание программной реализации

Семейство функций будем хранить в памяти компьютера как двумерный массив. По одной размерности будем индексировать сами функции семейства, а по другой размерности будем хранить значения конкретной функции на всевозможных наборах из n чисел от 0 до $k-1$. При этом индексировать значения функции будем не самими наборами, а их десятичным представлением. В связи с этим, потребуется реализовать два метода — для кодирования чисел из десятичной системы в k-ичную и обратно. Оба метода несложно реализовать за $O(n)$.

Выделим память под новое семейство, которое получим из исходного путём перестановки i-й функции с последней и i-й переменной с последней. Чтобы сформировать новое семейство, необходимо для каждой из n функций перебрать все k^n значений, каждое из значений преобразовать в k-ичный набор длины n, поменять местами i-ые разряды и преобразовать k-ичные наборы обратно в десятичные. Итого, для этого потребуется $2n^2k^n$ операций. Далее необходимо поменять местами каждое из k^n значений i-й функции и последней. После этого нужно присвоить старому семейству значения нового (еще nk^n действий). Итого, на перестановку i-й функции с последней и i-й переменной с последней уйдет $2n^2k^n + k^n + nk^n$ действий.

Набор семейств функций G_m, $0 \leq m \leq k-1$ представляет собой трехмерный массив из $k(n-1)k^{n-1}$ элементов. По одной размерности индексируются сами семейства, а две других — есть двумерный массив, хранящий семейство функций, аналогично исходному, но уже на наборах из $n-1$ числа.

Сформировать такой набор семейств можно не более чем за $k(n-1)k^{n-1}(3n-2)$ действий. Действительно: k штук семейств, каждое из которых состоит из $n-1$ функции, на каждой функции каждого семейства необходимо задать k^{n-1} значений. При этом, задавая значения функции, нам нужно получить из десятичного представления значения его k-ичную кодировку из $n-1$ символа, образовать на основе этой кодировки новую, состоящую уже из n элементов, поставив на последнюю позицию значение m, и уже новую k-ичную кодировку перевести обратно в десятичную. На это потребуется не более $3n-2$ действий.

На основании построенного набора G_m начинаем строить $(g_1^{'}, ..., g_{n-1}^{'})$. Это можно осуществить не более чем за $(n-1)k^n(3n-2+k)$ действий: для каждой из $n-1$ функций необходимо вычислить значения на k^n входных наборах. При этом, во время вычисления набора, в функцию $I_{x_n}(m)$ подставляется значение переменной x_n, для вычисления которой нужно получить k-ичную кодировку текущего значения, что делается не более чем за n шагов. Далее из этого набора нужно взять первые $n-1$ элемент и новый набор из $n-1$ элемента перевести в десятичное представление. Также, при вычислении каждого значения, потребуется k вычислений максимума из минимумов.

Далее, необходимо построить граф. Для этого нужно перебрать все пары наборов из E_k^{n-1} и для каждого из наборов вычислить k-ичные кодировки этой пары, а затем перебрать все пары семейств из набора G_m, $0 \leq m \leq k-1$ и на каждой паре сравнивать наборы из $n-1$ элемента. Итого, на это потребуется $\frac{k^{n-1}(k^{n-1}-1)}{2}(n-1)(\frac{k(k-1)}{2}+2)$ действий. Также, в процессе построения

графа, будем задавать список смежностей, для дальнейшего применения алгоритма обхода графа в глубину и формирования компонент связности. Список смежностей будет содержать столько же элементов, сколько существует рёбер в построенном графе. Это не более чем $\frac{k^{n-1}(k^{n-1}-1)}{2}$ элементов.

Для вычисления компонент связности проинициализируем массив из k^{n-1} булевых переменных значением false. Каждое из значений будет обозначать посетили ли мы ту или иную вершину графа. После этого обойдем граф и зададим значения функции ${g_n}'$ на каждой из компонент случайной константой. Учитывая, что сложность обхода графа в глубину линейно зависит от суммы числа вершин и числа ребер графа, а инициализировать функцию придется на k^n наборах, приходим к выводу, что на всю процедуру уйдет не более $k^{n-1} + \frac{k^n(k^n-1)}{2} + k^n$ действий.

Можно заметить, что наибольшую сложность в алгоритме представляет обход и хранение графа. Отсюда получаем, что временная сложность алгоритма $O(nk^{2n})$, а сложность по памяти составляет $O(k^{2n})$. Обозначив порядок квазигруппы за $N = k^n$, получаем следующее утверждение.

Утверждение 2. *Один шаг алгоритма имеет временную сложность $O(N^2 \log_k N)$, общая временная сложность высчитывается как произведение числа шагов и сложности одного шага. Сложность по памяти как одного шага алгоритма, так и всего алгоритма в целом, составляет $O(N^2)$.*

5 Порождение квазигруппы перестановками с полным дифференциалом

Далее будут рассмотрены алгоритмы порождения случайных конечных квазигрупп на основе регистров сдвига с обратной связью и обобщенных сетей Фейстеля. Будут описаны условия, при которых данные квазигруппы обладают тривиальными подквазигруппами. Также будет предложен способ задания n-квазигрупп с помощью полных перестановок.

5.1 Описание алгоритма

Рассмотрим для начала обобщение сетей Фейстеля.

Теорема 2 ([17]). *Обобщенная сеть Фейстеля задает полную перестановку тогда и только тогда, когда $s(x)$ и $s'(x) = s(x) + p(x) - x$ являются биекциями.*

Таким образом, для порождения квазигруппы необходимо лишь сгенерировать две случайные перестановки s и s'. Сделать это можно, например, используя алгоритм Фишера-Йетса [23]. После генерации обеих перестановок можно вычислить функцию $p(x) = s'(x) - s(x) + x$. Сложность алгоритма Фишера-Йетса для генерации перестановки из k элементов $O(k)$. Сложность вычисления функции $p(x)$ также $O(k)$. Таким образом, верно следующее утверждение о сложности.

Утверждение 3. *Общая сложность всех преобразований составляет* $O(k)$ *или* $O(\sqrt{N})$. *Сложность алгоритма по памяти также* $O(k)$ *или* $O(\sqrt{N})$. *Здесь* N *— порядок квазигруппы.*

Если потребуется вычисление всего латинского квадрата, то для вычисления каждого конкретного значения $f_Q(x, y)$ потребуется преобразовать x и y в k-ичные векторы (x_1, x_2) и (y_1, y_2) соответственно. После этого необходимо вычислить вектор (f_1, f_2) и преобразовать этот k-ичный вектор в $f_Q(x, y)$. Все эти операции выполняются с константной сложностью, а значит, сложность преобразования обобщенных сетей Фейстеля в латинский квадрат составляет $O(N^2)$.

Рассмотрим теперь алгоритм генерации регистров сдвига.

Теорема 3 ([17]). *Регистр сдвига с обратной связью задает полную перестановку если и только если любой нетривиальный сдвиг меняет значение* g. *То есть для любого набора* $(x_2, ..., x_n)$ *и любого* α, $\alpha \neq 0$, *значения* $g(x_2, ..., x_n)$ *и* $g(x_2 + \alpha, ..., x_n + \alpha)$ *различны.*

Таким образом, весь алгоритм заключается в генерации функции g, из которой однозначно можно получить некоторую квазигруппу. Будем перебирать все наборы вида $(0, x_3, ..., x_n)$. Для каждого такого набора сгенерируем случайную перестановку $(p_0, ..., p_{k-1})$. После этого зададим значение функции g на наборе $(i, x_3 + i, ..., x_n + i)$ равным p_i. Таким образом получится функция g, такая, что любой нетривиальный сдвиг будет менять её значение. При такой функции g регистр сдвига с обратной связью будет являться перестановкой с полным дифференциалом и задавать квазигруппу. Поскольку изменение константы c_n эквивалентно тому, чтобы выбрать другую функцию g, сдвинутую на константу c_n, без ограничения общности далее считаем, что $c_n = 0$.

Программно алгоритм выглядит следующим образом.

1) Перебираем числа от 0 до $k^{n-2} - 1$.

2) Каждое из них преобразуем в k-ичный вектор $(x_2, ..., x_n)$.

3) После этого формируем новый k-ичный вектор $(0, x_2, ..., x_n)$.

4) Генерируем перестановку алгоритмом Фишера-Йетса.

5) Перебирая числа от 0 до $k - 1$ формируем набор, на котором хотим задать значение функции g.

6) Преобразуем этот набор в десятичное представление и задаем на нём значение функции g.

7) Генерируем случайные значения для $c_1, ..., c_{n-1}$.

Сложность данного алгоритма по времени составляет $O(k^{n-2}(2n - 4 + k(2n - 1)) + n - 1)$. Сложность алгоритма по памяти $O(k^{n-1} + n - 1)$, так как потребуется хранить k^{n-1} значений функции g и $n - 1$ констант. С учетом того, что порядок квазигруппы $N = k^n$, получаем следующее утверждение.

Утверждение 4. *Общая временная сложность алгоритма порождения квазигруппы на основе регистра сдвига с обратной связью* $O(\frac{N}{k} \log_k N)$. *Сложность алгоритма по памяти составляет* $O(\frac{N}{k})$.

Если потребуется преобразовать регистры сдвига в латинский квадрат, то для этого необходимо для каждого значения функции $f_Q(x,y)$ преобразовать оба аргумента в k-ичные векторы длины n, вычислить k-ичный вектор $(f_1,...,f_n)$, используя значения функции g и преобразовать этот вектор в $f_Q(x,y)$. Общая временная сложность такого преобразования составит $O(nk^{2n})$ или $O(N^2 \log_k N)$.

5.2 О неподвижных точках

Далее будут рассматриваться условия, при которых квазигруппы на основе обобщенных сетей Фейстеля и регистров сдвига содержат тривиальные подквазигруппы. Отдельно будут рассматриваться два сценария: когда квазигрупповая операция задана формулой $\sigma(x+y)-y$ и когда она задана формулой $\sigma(x-y)+y$. Будем считать, что все арифметические операции выполняются по модулю k.

5.2.1 Задание квазигруппы формулой $\sigma(x+y)-y$

Утверждение 5. *Пусть $g(x_2,...,x_n)$ — функция, задающая регистр сдвига с обратной связью, на основе которого построена квазигруппа (f_Q,Q). Обозначим $c'_j = \sum_{i=j}^{n-1} c_i$. Тогда (f_Q,Q) содержит тривиальную подквазигруппу тогда и только тогда, когда существует элемент $x=(x_1,x_2,...,x_n) \in Q$ такой, что $g(2x_n+c'_2,...,2x_n+c'_{n-1},2x_n) = -c'_1$, а каждое из уравнений вида $2x_i = 2x_{i+1}+c_i$, $1 \le i \le n-1$ имеет хотя бы одно решение.*

Доказательство. Множество $Q'=\{q_i\}$ (с ограничением операции f_Q на него) будет тривиальной подквазигруппой тогда и только тогда, когда $f_Q(q_i,q_i)=q_i$. Рассмотрим k-ичный вектор $(x_1,...,x_n)$, который соответствует q_i, и запишем тождество $f_Q(q_i,q_i)=q_i$ в виде системы, соответствующей регистру сдвига с обратной связью:

$$\begin{cases} x_1 = x_2 + x_2 - x_1 + c_1 \\ x_2 = x_3 + x_3 - x_2 + c_2 \\ ... \\ x_{n-1} = x_n + x_n - x_{n-1} + c_{n-1} \\ x_n = x_1 + x_1 + g(x_2+x_2,...,x_n+x_n) - x_n \end{cases}$$

Эта система равносильна следующей:

$$\begin{cases} 2x_1 = 2x_2 + c_1 \\ 2x_2 = 2x_3 + c_2 \\ ... \\ 2x_{n-1} = 2x_n + c_{n-1} \\ 2x_n = 2x_1 + g(2x_2,...,2x_n) \end{cases}$$

Последовательно выражая все $2x_i, 1 \le i \le n-1$ через $2x_n$ и подставляя в последнее уравнение получаем требуемое утверждение. □

Следствие 5.1. *Если k нечётно, то все квазигруппы на основе регистров сдвига с обратной связью, получаемые по формуле $\sigma(x+y)-y$, содержат тривиальную подквазигруппу, причем только одну.*

Доказательство. При нечётном k элемент $2x_n$ пробегает все элементы множества $\{0,1,...,k-1\}$, когда их же пробегает x_n. Следовательно, условие утверждения равносильно тому, что хотя бы один из элементов $g(c'_2,...,c'_{n-1},0), g(c'_2+1,...,c'_{n-1}+1,1),...,g(c'_2+k-1,...,c'_{n-1}+k-1,k-1)$ принимает значение $-c'_1$. Поскольку функция g задает квазигруппу, она должна менять свои значения при любом нетривиальном сдвиге. Это значит, что $\{g(c'_2,...,c'_{n-1},0), g(c'_2+1,...,c'_{n-1}+1,1),...,g(c'_2+k-1,...,c'_{n-1}+k-1,k-1)\}$ — перестановка элементов $\{0,1,...,k-1\}$, т.е. одно из них принимает значение $-c'_1$. Следовательно, это условие выполнено для любой функции g, задающей квазигруппу с помощью регистра сдвига.

Пусть теперь c — такая константа, что $g(c+c'_2,...,c+c'_{n-1},c)=-c'_1$. Уравнение $2x_n=c$ всегда будет иметь ровно одно решение. Найдя x_n можно последовательно решить уравнения $2x_i=2x_{i+1}+c_i$, $1\le i\le n-1$, которые также будут иметь по одному решению. Таким образом для любой квазигруппы можно установить единственный элемент $x=(x_1,...,x_n)$, который будет являться тривиальной подквазигруппой. □

Следствие 5.2. *Пусть k чётно. Тогда:*

1) *Доля квазигрупп на основе регистров сдвига с обратной связью, заданных формулой $\sigma(x+y)-y$ и содержащих тривиальную подквазигруппу, равна $\frac{1}{2^n}$.*

2) *Квазигруппа на основе регистра сдвига с обратной связью, полученная по формуле $\sigma(x+y)-y$, либо не содержит тривиальных подквазигрупп, либо содержит сразу 2^n тривиальных подквазигрупп.*

Доказательство. При чётном k элемент $2x_n$ пробегает только чётные элементы множества $\{0,1,...,k-1\}$, когда x_n пробегает все значения из $\{0,1,...,k-1\}$. Сделаем два замечания:

1) Если хотя бы один из c_i при $1\le i\le n-1$ является нечётным числом, то квазигруппа не содержит тривиальной подквазигруппы. В противном случае нашелся бы i, такой, что $2x_i=2x_{i+1}+c_i$, где $2x_i$ — чётное, $2x_{i+1}$ — чётное, а c_i — нечётное. Т.е., чётное число было бы равно нечётному. Таким образом, для каждого c_i существует $\frac{k}{2}$ возможных значений. А всего подходящих наборов $c_1,...,c_{n-1}$ существует $\frac{k^{n-1}}{2^{n-1}}$ штук.

2) Пусть зафиксирован подходящий набор $c_1,...,c_{n-1}$. Тогда выражение $g(2x_n+c'_2,...,2x_n+c'_{n-1},2x_n)=-c'_1$ эквивалентно тому, что хотя бы одно из значений $g(c'_2,...,c'_{n-1},0), g(c'_2+2,...,c'_{n-1}+2,2),...,g(c'_2+k-2,...,c'_{n-1}+k-2,k-2)$ совпадает с $-c'_1$. Но это возможно ровно в половине случаев. Таким образом, для фиксированного набора $c_1,...,c_{n-1}$ существует $\frac{(k!)^{k^{n-2}}}{2}$ квазигрупп с тривиальной подквазигруппой.

Исходя из этих замечаний число квазигрупп с тривиальной подквазигруппой равно количеству подходящих наборов $c_1,...,c_{n-1}$ умноженному на число

квазигрупп с тривиальной подквазигруппой для этого набора. Т.е., всего их $\frac{k^{n-1}*(k!)^{k^{n-2}}}{2^n}$. Значит их доля от общего числа составляет $\frac{1}{2^n}$.

Пусть c, как и ранее, такая константа, что $g(c+c'_2,...,c+c'_{n-1},c) = -c'_1$, т.е. тривиальная подквазигруппа существует. Тогда уравнение $2x_n = c$ будет иметь два решения. Для каждого из x_i каждое из уравнений $2x_i = 2x_{i+1} + c_i$, $1 \leq i \leq n-1$ так же будет иметь по два решения. Таким образом, можно найти 2^n различных элементов $x = (x_1,...,x_n)$, каждый из которых будет являться тривиальной подквазигруппой.

□

Утверждение 6. *Пусть $s(x)$ и $p(x) = s'(x) - s(x) + x$ — функции из обобщенной сети Фейстеля, которые задают квазигруппу (f_Q, Q) по формуле $\sigma(x+y)-y$. Тогда (f_Q, Q) имеет тривиальную подквазигруппу тогда и только тогда, когда существует пара (x_1, x_2), где $x_1, x_2 \in \{0,1,...,k-1\}$, такая, что $s'(2x_2) = 0$ и $s(2x_2) = 2x_1$. Этой подквазигруппой будет являться элемент q_i, который соответствует этой паре.*

Доказательство. Аналогично прошлому утверждению достаточно расписать систему, которой должно удовлетворять произведение в тривиальной подквазигруппе:

$$\begin{cases} x_1 = s(x_2 + x_2) - x_1 \\ x_2 = x_1 + x_1 + p(x_2 + x_2) - x_2 \end{cases}$$

Подставив вместо p его выражение через s' и s, а также подставив первое уравнение во второе, получим систему:

$$\begin{cases} 2x_1 = s(2x_2) \\ s'(2x_2) = 0 \end{cases}$$

Из этой системы и следует утверждение. □

Следствие 6.1. *Если k нечётно, то все квазигруппы на основе обобщенных сетей Фейстеля содержат тривиальную подквазигруппу, причем только одну.*

Доказательство. Как и ранее, когда x_2 пробегает множество $\{0,1,...,k-1\}$, $2x_2$ также пробегает его (аналогично и $2x_1$). Следовательно, условие существования такого x_2, что $s'(2x_2) = 0$, верно всегда, т.к. s' — перестановка. Найдя такой x_2 нужно найти такую перестановку s, для которой $s(2x_2) = 2x_1$, но такой x_1 тоже всегда можно подобрать для любой перестановки. Следовательно, для любых перестановок s и s' утверждение верно. А следовательно, все квазигруппы имеют тривиальную подквазигруппу.

Пусть c — такая константа, что $s'(c) = 0$. Тогда уравнение $2x_2 = c$ имеет ровно одно решение. Уравнение $2x_1 = s(c)$ также будет иметь ровно одно решение. Таким образом, можно найти единственный элемент $x = (x_1, x_2)$, являющийся тривиальной подквазигруппой. □

Следствие 6.2. *Пусть k чётно. Тогда:*

1) *Ровно четверть квазигрупп на основе обобщенных сетей Фейстеля содержат тривиальную подквазигруппу.*

2) *Квазигруппа на основе обобщенной сети Фейстеля либо не содержит тривиальных подквазигрупп, либо содержит сразу 4 тривиальных подквазигруппы.*

Доказательство. В этом случае $2x_2$ и $2x_1$ — это чётные числа, а условие $s'(2x_2) = 0$ значит, что прообразом нуля у перестановки s' должно быть чётное число. Это верно ровно в половине случаев. Теперь, если это верно, найдя x_2 и зафиксировав s', нужно найти x_1 из условия, что $2x_1 = s(2x_2)$. Т.е. s — это такая перестановка, которая зафиксированное нами число отображает в произвольное чётное. Существует $\frac{k}{2}$ вариантов в какое чётное число отобразить каждое фиксированное $2x_2$ и еще $(k-1)!$ вариантов как расставить остальные значения. Т.е. для каждой перестановки s' подходит $\frac{\frac{k}{2}(k-1)!}{k!} = \frac{1}{2}$ — ровно половина всех перестановок из k элементов в качестве s. Итого, ровно четверть всех пар (s', s) задают квазигруппу, которая подходит под утверждение. Следовательно, ровно четверть квазигрупп имеют тривиальную подквазигруппу.

Пусть тривиальная подквазигруппа существует и c — такая константа, что $s'(c) = 0$. Тогда уравнение $2x_2 = c$ имеет два решения — c_1 и c_2. Каждое из уравнений $2x_1 = s(2c_1)$, $2x_1 = s(2c_2)$ также имеет по два решения. Таким образом, существует 4 различных элемента $x = (x_1, x_2)$, которые могут являться тривиальной подквазигруппой. □

5.2.2 Задание квазигруппы формулой $\sigma(x - y) + y$

Утверждение 7. *Пусть регистр сдвига с обратной связью задает квазигруппу по формуле $\sigma(x - y) + y$. Тогда эта квазигруппа содержит тривиальную подквазигруппу тогда и только тогда, когда $c_1 = c_2 = ... = c_{n-1} = g(0, ..., 0) = 0$.*

Доказательство. Как и ранее, распишем явно условие существования тривиальной подквазигруппы:

$$\begin{cases} x_1 = x_2 - x_2 + x_1 + c_1 \\ x_2 = x_3 - x_3 + x_2 + c_2 \\ ... \\ x_{n-1} = x_n - x_n + x_{n-1} + c_{n-1} \\ x_n = x_1 - x_1 + g(x_2 - x_2, ..., x_n - x_n) + x_n \end{cases}$$

Эта система равносильная следующей, которая и доказывает требуемое утверждение.

$$\begin{cases} c_1 = 0 \\ c_2 = 0 \\ ... \\ c_{n-1} = 0 \\ g(0, ..., 0) = 0 \end{cases}$$

□

Следствие 7.1. *Доля квазигрупп на основе регистров сдвига с обратной связью, заданных формулой $\sigma(x-y)+y$ и содержащих тривиальную подквазигруппу, равна $\frac{1}{k^n}$. Причем, в таком случае, каждый элемент квазигруппы будет являться тривиальной подквазигруппой.*

Доказательство. Существует $(k-1)!$ задать значения в классе эквивалентности, содержащем 0, чтобы было выполнено условие $g(0,...,0)=0$. Остальные $k^{n-2}-1$ классов эквивалентности можно задать произвольным образом, для них существует по $k!$ вариантов. Таким образом, итоговая доля составляет:

$$\frac{(k!)^{k^{n-2}-1}*(k-1)!}{(k!)^{k^{n-2}}*k^{n-1}} = \frac{(k!)^{k^{n-2}-1}*(k-1)!*k}{(k!)^{k^{n-2}}*k^{n-1}*k} = \frac{1}{k^n}.$$

Поскольку условие утверждения никак не зависит от элемента квазигруппы, каждый элемент квазигруппы будет являться тривиальной подквазигруппой. □

Утверждение 8. *Пусть обобщенная сеть Фейстеля задает квазигруппу по формуле $\sigma(x-y)+y$. Тогда эта квазигруппа содержит тривиальную подквазигруппу тогда и только тогда, когда $s(0)=p(0)=0$.*

Доказательство. Выпишем условие существования тривиальной подквазигруппы:

$$\begin{cases} x_1 = s(x_2-x_2)+x_1 \\ x_2 = x_1 - x_1 + p(x_2-x_2)+x_2 \end{cases}$$

Система равносильна следующей, из которой следует требуемое утверждение.

$$\begin{cases} s(0)=0 \\ p(0)=0 \end{cases}$$

□

Следствие 8.1. *Доля квазигрупп на основе обобщенной сети Фейстеля, заданных формулой $\sigma(x-y)+y$ и содержащих тривиальную подквазигруппу, равна $\frac{1}{k^2}$. Причем, в таком случае, каждый элемент квазигруппы будет являться тривиальной подквазигруппой.*

Доказательство. Поскольку s является произвольной перестановкой, а p получается из произвольной перестановки, существует по $(k-1)!$ вариантов задать эти перестановки таким образом, чтобы 0 отображался в 0. Таким образом, доля квазигрупп, содержащих тривиальную подквазигруппу, составляет $\frac{((k-1)!)^2}{(k!)^2} = \frac{1}{k^2}$.

При этом, аналогично предыдущему утверждению, условие никак не зависит от элемента квазигруппы. Значит все элементы квазигруппы будут являться тривиальными подквазигруппами. □

5.3 Обобщение для задания n-квазигрупп

Утверждение 9. *Пусть задана конечная абелева группа* $(G;+)$, *а* σ — *полная перестановка над* $(G;+)$. *Тогда можно задать* n*-квазигрупповую операцию* $f(x_1,...,x_n)$ *следующим образом:*

$$f(x_1,...,x_n) = \sigma(x_1 + x_2 + ... + x_n) - x_2 - ... - x_n.$$

Доказательство. Достаточно показать, что для любых $a_1,...,a_n \in Q$ следующие отображения $f(x,a_2,...,a_n), f(a_1,x,a_3,...,a_n),...,f(a_1,...,a_{n-1},x)$ являются биекциями. Распишем первое из них: $f(x,a_2,...,a_n) = \sigma(x+a_2+...+a_n) - a_2 - ... - a_n$. Это биекция, т.к. σ биекция, а операция групповая. Теперь рассмотрим произвольное из оставшихся выражений: $f(a_1,...,a_{i-1},x,a_{i+1},...,a_n) = \sigma(a_1+...+a_{i-1}+x+a_{i+1}+...+a_n) - (a_1 - a_2 - ... - a_{i-1} - x - a_{i+1} - ... - a_n) + a_1$. Это биекция, т.к. σ полная перестановка, а операция групповая. □

Замечание 1. *Несложно видеть, что если в утверждении для произвольного* $2 \leq i \leq n$ *изменить знаки перед всеми вхождениями элемента* x_i *на противоположные, то утверждение останется верным. Таким образом, сменой знаков можно получить* 2^{n-1} *вариантов задать* n*-квазигрупповую операцию через полную перестановку.*

6 Проверка аффинности

Далее рассматривается процедура проверки квазигрупп на аффинность, подробное описание которой можно найти в [24], а также приводится описание программной реализации алгоритма.

6.1 Описание алгоритма

Пусть L — латинский квадрат, являющийся таблицей умножения квазигруппы (Q, f_Q), $Q = \{q_1,...q_n\}$. Будем обозначать σ_i перестановку, соответствующую i-й строке L.

Построим матрицу L' таким образом, что её i-я строка соответствует перестановке $\sigma_i * \sigma_1^{-1}$.

Переставим строки L' так, чтобы первый столбец совпадал с первой строкой. Полученную матрицу обозначим L'', а порожденную ей бинарную операцию обозначим $f_{L''}$.

В случае если матрица L'' несимметрична объявим квазигруппу неаффинной.

Проверим, что равенство

$$f_{L''}(f_{L''}(q_r,q_s),q_t) = f_{L''}(q_r,f_{L''}(q_s,q_t));$$

выполнено для любой тройки $1 \leq r,s,t \leq n$. В случае, если хотя бы для одной тройки равенство не выполнено, объявим квазигруппу неаффинной.

Найдем строку и столбец матрицы L, первые элементы которых совпадают с левым верхним элементом матрицы L''. Обозначим соответствующие им перестановки за α и β соответственно.

Проверим, что равенства

$$\alpha(f_{L''}(q_i, q_j)) = f_{L''}(\alpha(q_i), \alpha(q_j));$$
$$\beta(f_{L''}(q_i, q_j)) = f_{L''}(\beta(q_i), \beta(q_j));$$

выполнены для любой пары $1 \leq i, j \leq n$. Т.е., что α и β сохраняют операцию $f_{L''}$. Если хотя бы для одной пары будет получено неравенство, объявим квазигруппу не аффиной.

Проверим, что для любой пары $1 \leq i, j \leq n$ выполнено равенство

$$f_Q(q_i, q_j) = f_{L''}(f_{L''}(\alpha(q_i), \beta(q_j)), c);$$

где c — левый верхний элемент матрицы L. Если хотя бы для одной пары равенство не выполнено, объявим квазигруппу неаффинной. В противном случае объявим квазигруппу аффинной.

Утверждение 10 ([24]). *Сложность процедуры проверки аффинности $O(N^3)$, где N — порядок квазигруппы.*

В квазигрупповом случае для проверки ассоциативности известна процедура, называемая тест Лайта.

Утверждение 11 ([25]). *Пусть G — множество с заданной операцией умножения $*$, и у G есть порождающее множество S. Тогда для проверки ассоциативности операции $*$ достаточно проверить тождества $x * (g * y)$ и $(x * g) * y$ для всех $x, y \in G$ и $g \in S$.*

Утверждение 12 ([26]). *Пусть G — квазигруппа порядка N. Тогда в G можно выделить порождающее множество S размером не больше $\lfloor \log_2 N \rfloor + 1$.*

Поскольку основная сложность в алгоритме уходит на проверку ассоциативности $f_{L''}$ перебором всех троек элементов, можно применить тест Лайта для оптимизации алгоритма. Вместо перебора всех троек можно найти порождающее множество для L''. А далее, проверить тождество ассоциативности в соответствии с утверждением 11.

6.2 Описание программной реализации

Будем считать, что на старте работы алгоритма мы уже храним в памяти матрицу L в виде двумерного массива. Матрица L' тоже будет являться двумерным массивом. Из определения σ_i следует, что $L_{ij} = \sigma_i(j)$. Поскольку строки L' являются произведениями перестановок σ_i и σ_1^{-1}, то элемент $L'_{ij} = \sigma_i(\sigma_1^{-1}(j)) = L_{i\sigma_1^{-1}(j)}$, $1 \leq i, j \leq n$. Здесь σ_1^{-1} хранится в виде одномерного массива. В результате построения L' первая строка матрицы будет соответствовать тождественной перестановке.

Таким образом, для построения матрицы L'' достаточно переставить строки $2, ..., n$ матрицы L' так, чтобы первый столбец также представлял собой тождественную перестановку. Это осуществляется путем сортировки строк по первому элементу столбца. После осуществления процедуры на месте матрицы L' в памяти будет храниться матрица L''.

Проверка L'' на симметричность осуществляется предельно просто. Достаточно пройти в цикле по левому верхнему треугольнику матрицы и проверить на равенство элементы L''_{ij} и L''_{ji}, $1 \leq j < i \leq n$.

Далее необходимо проверить ассоциативность операции, заданной матрицей L''. Для этого необходимо построить порождающее множество. Воспользуемся следующей процедурой.

1) Зададим на старте множества $R = S = \emptyset$, $T = \{q_0, ..., q_{N-1}\}$.

2) Берем произвольный элемент q из множества T, удаляем его из T и добавляем его во множества R и S. Если элементов в T не было — процедура завершается.

3) Последовательно умножаем элемент q на все элементы множества R слева и справа, в соответствии с таблицей умножения L''. Если в результате умножения появляются элементы, которые присутствуют в T, удаляем их из T и добавляем в R. Если в результате текущего шага новых элементов в R не появилось, переходим к шагу 2. В противном случае повторяем шаг 3.

По окончании данной процедуры множество R станет равным $\{q_0, ..., q_{N-1}\}$, а множество T будет пустым. Множество S на каждом шаге расширялось только в том случае, если в T находился элемент, который еще нельзя представить в виде произведения уже имеющихся в S элементов. Поскольку T изначально содержало все элементы квазигруппы, а в результате оказалось пустым, каждый элемент квазигруппы можно представить в виде произведения элементов из S. Значит S является порождающим множеством. Поскольку шаг 3 для каждого элемента из T будет выполнен не более чем $2(N-1)$ раз, приходим к тому, что порождающее множество можно построить со сложностью $O(N^2)$.

После построения порождающего множества переберем все тройки x, y, g, такие что $x, y \in Q$, а $g \in S$ и проверим выполнение тождества $f_{L''}(f_{L''}(x, g), y) = f_{L''}(x, f_{L''}(g, y))$.

Поиск перестановок α и β осуществляется путем проходов по первым столбцу и строке матрицы L и поиску в них элемента q_1, который и является левым верхним элементом L''.

Проверка выполнения всех равенств в алгоритме выполняется путем перебора всех требуемых комбинаций и сравнения нужных элементов, что сводится к обращению к ячейкам памяти.

Из утверждений 11 и 12 следует, что ассоциативность квазигруппы можно проверить со сложностью $O(N^2 \log_2 N)$. Таким образом, получаем новую сложности алгоритма.

Утверждение 13. *Сложность процедуры проверки аффинности* $O(N^2 \log_2 N)$, *где* N — *порядок квазигруппы.*

7 Проверка простоты

Ниже описан алгоритм проверки простоты, предложенный в [24], и его последующая оптимизация из [27]. Помимо алгоритма также описана его программная реализация.

7.1 Описание алгоритма

Пусть (Q, f_Q) — квазигруппа, $Q = \{q_1, ..., q_n\}$, L — латинский квадрат задающий f_Q. Зафиксируем индекс i_0 и для всех пар вида $\{q_{i_0}, q_j\}$, $1 \leq j \leq n$, $j \neq i_0$ рассмотрим следующую процедуру.

1) Каждой неупорядоченной паре $1 \leq t_1, t_2 \leq n$, $t_1 \neq t_2$, сопоставим две метки — рассмотренность и эквивалентность. Изначально все пары считаем нерассмотренными и неэквивалентными. Отмечаем пару $\{i_0, j\}$ как эквивалентную.

2) Возьмем произвольную нерассмотренную эквивалентную пару $\{a, b\}$. Для всех перестановок σ, соответствующих строкам и столбцам L, будем рассматривать элементы $\sigma(a)$ и $\sigma(b)$. Если их эквивалентность еще не была установлена, объединим соответствующие им классы эквивалентности и отметим новые эквивалентные пары. После рассмотрения всех перестановок помечаем пару $\{a, b\}$ как рассмотренную.

3) Если множество нерассмотренных эквивалентных пар пусто или все элементы попарно эквивалентны, заканчиваем работу, в противном случае переходим к шагу 2.

Если для всех $\{i_0, j\}$ процедура завершилась после того, как все элементы оказались попарно эквивалентными, объявим квазигруппу простой. В противном случае квазигруппа простой не является.

В оптимизированном варианте алгоритма учитывается, что нетривиального разбиения не существует, если есть хотя бы один класс эквивалентности, состоящий более чем из $\frac{n}{2}$ элементов. Также утверждается, что достаточно добавлять в очередь на проверку только одну пару из нового класса эквивалентности.

Утверждение 14 ([24], [27]). *Сложность процедуры проверки простоты $O(N^3)$, где N — порядок квазигруппы.*

7.2 Описание программной реализации

Для простоты в качестве q_{i_0} будем брать q_1. Тогда необходимо выполнять указанную процедуру для всех пар вида q_1, q_j, $2 \leq j \leq n$.

Неупорядоченную пару t_1, t_2 будем задавать в виде структуры с двумя полями: два целых числа t_1 и t_2. Будем хранить классы эквивалентности в виде массива из n контейнеров $std :: vector$. Также сохраним массив из n целых чисел, i-й элемент которого будет содержать номер класса эквивалентности, в котором находится элемент i. Это позволит определять, в каком классе находится нужный элемент, за константное время и избавит от нужды помечать новые эквивалентные пары.

Так как существует необходимость обращаться к нерассмотренным эквивалентным парам, было бы удобно хранить их в памяти. В качестве контейнера для них будем использовать структуру данных «очередь». При этом, пары, не находящиеся в очереди на проверку, можно не сохранять. После объединения двух классов будет происходить добавление одной пары в очередь, а после её рассмотрения — удаление её из очереди.

Рассмотрение перестановок выполняется перебором строк и столбцов матрицы, хранящей латинский квадрат L, которая является двумерным массивом. Для того чтобы узнать, является ли пара эквивалентной, достаточно узнать принадлежат ли элементы одному классу, что делается путем обращения к соответствующему массиву. Когда возникает необходимость соединить два класса эквивалентности, элементы вектора, соответствующего меньшему из классов, добавляются в вектор большего класса и удаляются из меньшего, а номера классов эквивалентности переброшенных элементов изменяются на новое значение.

Для того чтобы не выполнять проверку на то, являются ли все элементы попарно эквивалентными, введем переменную-счетчик. При объединении двух классов, состоящих из m и k элементов соответственно, число эквивалентных пар увеличивается на $m \cdot k$. На то же число увеличивается переменная-счетчик.

Аналогично введем переменную, хранящую размер наибольшего из классов. При объединении двух классов, состоящих из m и k элементов соответственно, получится класс размером $m + k$. Если это значение превышает текущий максимальный размер класса, то переменной присваивается новое значение. Если это значение окажется больше чем $\frac{n}{2}$, то, как было сказано в описании алгоритма, нетривиального разбиения не будет существовать.

Таким образом, если число эквивалентных пар не равно числу неупорядоченных пар, то есть $\frac{n \cdot (n-1)}{2}$, и при этом размер максимального класса не превосходит $\frac{n}{2}$, а нерассмотренных эквивалентных пар не осталось, то квазигруппа не является простой.

8 Алгоритм нахождения подквазигрупп

Для проверки существования нетривиальных подквазигрупп использовался алгоритм, предложенный в [28]. Алгоритм состоит из трех этапов.

1) Рассматриваем всевозможные неупорядоченные пары $\{q_i, q_j | q_i, q_j \in Q\}$; вычисляем замыкание каждой пары, пока либо не получится замкнутое подмножество (в этом случае, очевидно, находится нетривиальная подквазигруппа и алгоритм завершает работу), либо размер частичного замыкания не станет равным $[c \cdot \sqrt{|Q|}]$ (положительная константа c является параметром).

2) Каждое частичное замыкание рассматриваем как множество неупорядоченных пар входящих в него элементов и строим систему представителей.

3) Для каждой пары из системы представителей строим замыкание; квазигруппа содержит нетривиальные подквазигруппы, если и только если найдется пара, замыкание которой не равно Q.

Для данного алгоритма верна следующая теорема.

Теорема 4 ([28]). *При фиксированной константе c предложенный алгоритм устанавливает наличие нетривиальных подквазигрупп в квазигруппе порядка N с временной сложностью $O(N^3 \log N)$ и пространственной сложностью $O(c \cdot N^{\frac{5}{2}})$, $N \to \infty$.*

В рамках данной работы использовалась программная реализация алгоритма, предоставленная его авторами.

В тех случаях, когда нужно было найти тривиальную подквазигруппу, перебирались все элементы $q \in Q$ и проверялось равенство $f_Q(q, q) = q$.

9 Эксперименты

Далее представлена часть работы, посвященная экспериментам с программой. В этой части приводится сравнительная характеристика всех описанных ранее алгоритмов порождения с точки зрения скорости работы. Полученные различными методами квазигруппы исследуются на наличие подквазигрупп и полиномиальной полноты. Приводятся скорости работы алгоритмов проверки простоты и аффинности. Дополнительно приводится экспериментальное подтверждение того, что алгоритм порождения правильных семейств функций задает равномерное распределение. В данном разделе будем считать, что все арифметические операции выполняются по модулю k.

9.1 Сравнение времени работы алгоритмов генерации

Обозначим алгоритм Джейкобсона-Мэтьюза как JM. Метод, при котором будет генерироваться случайное правильное семейство, а из него случайная квазигруппа, обозначим PF. Алгоритмы порождения регистров сдвига и обобщенных сетей Фейстеля с последующим преобразованием их в квазигруппу обозначим SR и FN соответственно.

При этом, в случае алгоритмов SR и FN выбор итоговой формулы для получения квазигруппы, очевидно, не влияет на скорость порождения, поскольку формулы отличаются лишь знаками. Без ограничения общности будем считать, что в данном эксперименте квазигруппы получаются из полных перестановок по формуле $\sigma(x + y) - y$.

В экспериментах генерировались от 100 до 1000 квазигрупп заданного порядка и замерялось среднее время порождения одного объекта. В случае, когда квазигруппы задавались функционально, перебирались различные варианты значений k и n, при которых можно получить нужный порядок. Результаты экспериментов представлены в таблице далее.

Порядок	JM	PF	FN	SR
16	0.00058	k = 2, n = 4: 0.00501 k = 4, n = 2: 0.0012	0.00004	k = 2, n = 4, 0.00011 k = 4, n = 2 0.00006
32	0.00528	0.06708	-	0.00035
64	0.04411	k = 2, n = 6: 0.85705 k = 4, n = 3: 0.19669 k = 8, n = 2: 0.0823	0.00023	k = 2, n = 6: 0.0013 k = 4, n = 3: 0.00057 k = 8, n = 2: 0.00033
81	0.09059	k = 3, n = 4: 0.76527 k = 9, n = 2: 0.17597	0.00034	k = 3, n = 4: 0.0012 k = 9, n = 2 0.0005
128	0.36359	10.5716	-	0.005
256	2.96117	k = 2, n = 8: 134.987 k = 4, n = 4: 31.9878 k = 16, n = 2: 8.53718	0.00327	k = 2, n = 8: 0.02245 k = 4, n = 4: 0.01031 k = 16, n = 2: 0.0046

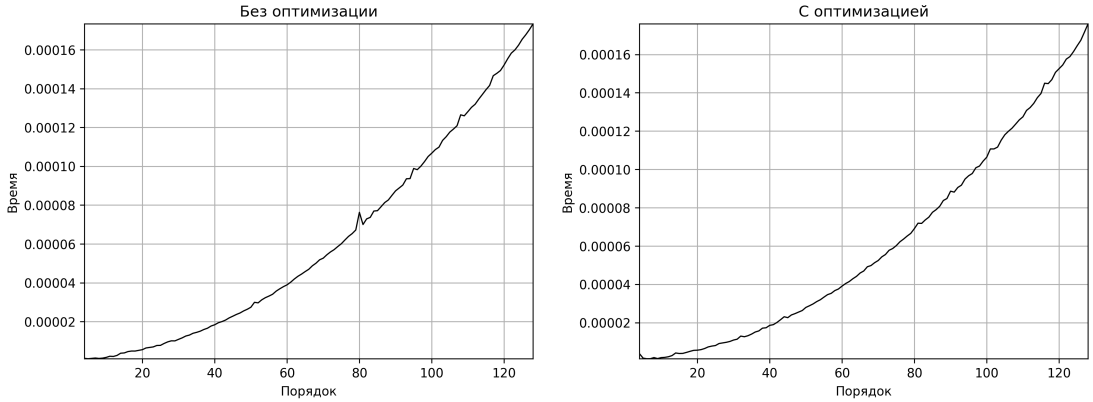
Таблица 1: В первом столбце указан порядок порождаемых квазигрупп, в остальных столбцах — время в секундах для соответствующих методов. Для функциональных квазигрупп время в таблице указано с учетом вычисления всего латинского квадрата.

Можно заметить, что наиболее эффективными по времени генерации оказываются квазигруппы на основе сетей Фейстеля. Время работы этого алгоритма многократно меньше, чем у остальных. В текущей реализации хуже всего себя показывает алгоритм генерации правильного семейства с последующей генерацией квазигруппы. С учетом того, что алгоритм Джейкобсона-Мэтьюза дает все квазигруппы, а также работает быстрее, он является более предпочтительным вариантом для практического использования, чем алгоритм на основе правильных семейств. Стоить отметить, что для одного и того же порядка все рассматриваемые функциональные семейства работают ощутимо быстрее, если выбрать k максимально возможным, а n наименьшим. Однако, этот же подход может уменьшить мощность множества порождаемых объектов.

9.2 Скорость проверки аффинности

В рамках эксперимента проводилась генерация квазигрупп всех порядков от 4 до 128 методом Джейкобсона-Мэтьюза. Каждая генерация производилась

по 250 раз. Измерялось среднее время работы программы при определенных значениях порядка квазигруппы. При этом отдельно измерялось время работы алгоритма до применения оптимизации и после. Результат представлен на графиках далее.



Видно, что графики похожи друг на друга, т.е. оптимизация проверки ассоциативности в среднем не влияет на скорость работы алгоритма.

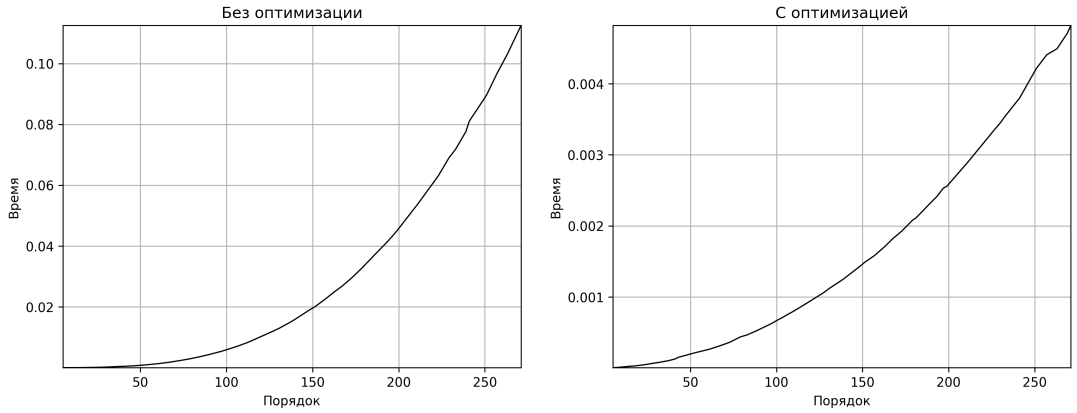
Оба графика имеют рост, близкий к квадратичному. Такое поведение можно объяснить тем, что почти все квазигруппы неаффинны. Часто алгоритм завершает работу еще во время процедуры проверки коммутативности, которая имеет квадратичную сложность. Можно заметить, что проверка аффинности, как правило, занимает меньше времени, чем генерация квазигруппы.

Отдельно было проведено сравнение двух версий алгоритма на аффинных квазигруппах. Аффинные квазигруппы задавались операцией вида $f_Q(x, y) = \alpha(x) + \beta(y) + c$, где сложение выполнялось по модулю порядка квазигруппы, c было случайной константой, а α и β были случайными автоморфизмами Z_N следующего вида:

$$0 \to 0$$
$$1 \to m, \text{ где } m \text{ — случайное число от 1 до } N-1$$
$$i \to i * m,\ 2 \le i \le N-1.$$

Такие отображения, очевидно, являются автоморфизмами, когда N — простое число. Здесь под N подразумевается порядок квазигруппы.

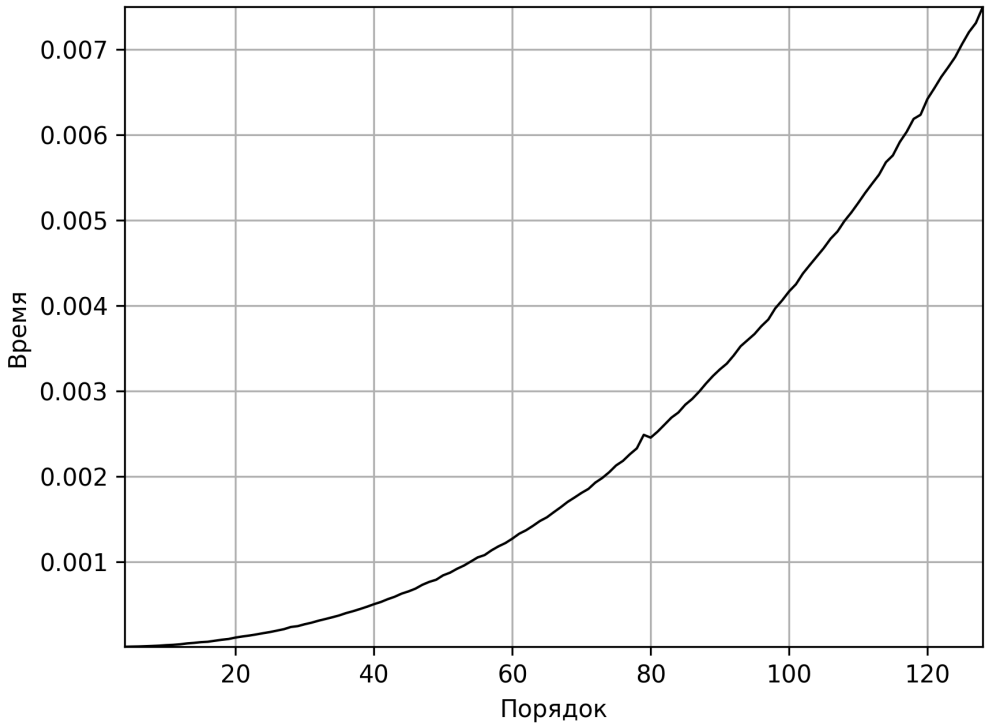
В эксперименте проводилась генерация квазигрупп всех простых порядков от 5 до 271. Каждая генерация производилась по 250 раз. Сравнение версий алгоритма представлено на графиках далее.



Обратим внимание на масштабы графиков по вертикальной оси. Видно, что оптимизированная версия алгоритма работает значительно быстрее и имеет лучшую асимптотику. Так, например, проверка аффинной квазигруппы порядка 271 после оптимизации в среднем работает примерно в 23 раза быстрее.

9.3 Скорость проверки простоты

Эксперимент проводился на тех же квазигруппах, что и в предыдущем пункте. Аналогичным образом замерялось время работы программы в зависимости от порядка сгенерированной квазигруппы. Приведем среднее время работы.

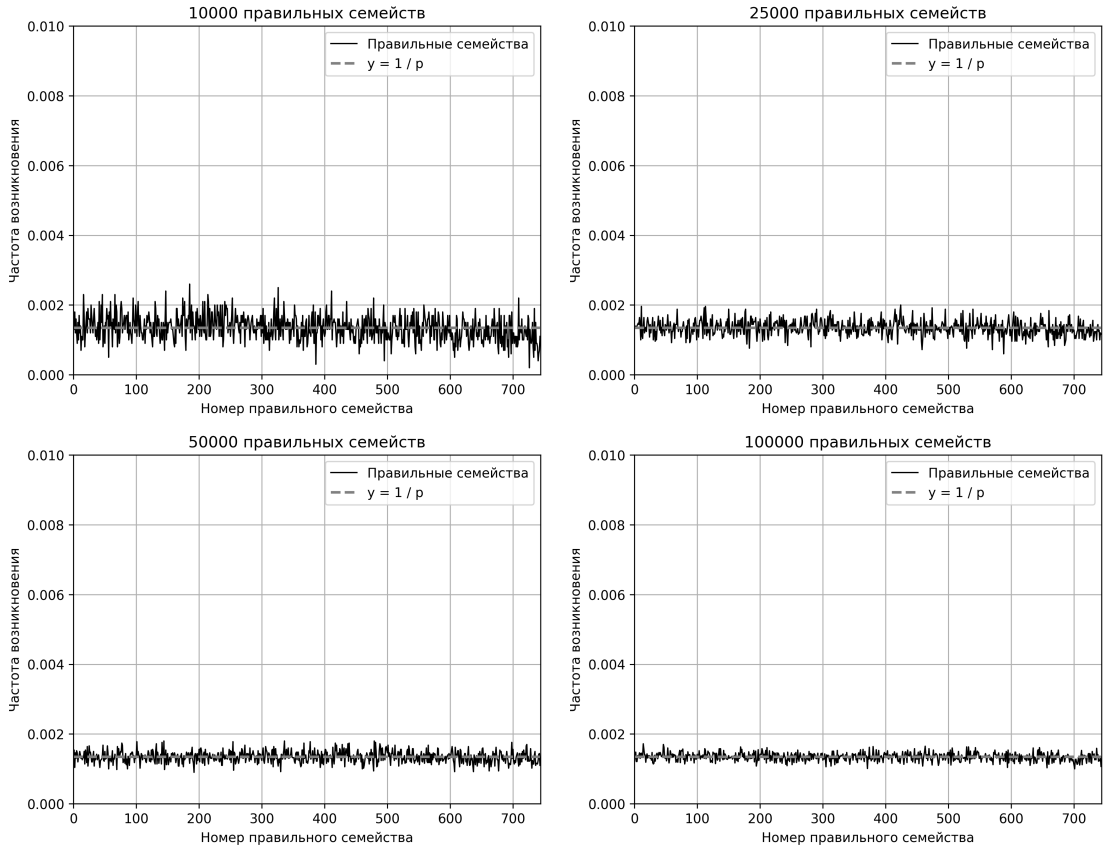


Данный график также близок к квадратичному. Это связано с тем, что зачастую в алгоритме не требуется проверять все пары, находящиеся в очереди. Почти все квазигруппы простые, и, в большинстве случаев, классы, содержащие больше половины элементов, появляются довольно быстро, что позволяет

осуществлять переход к следующей итерации внешнего цикла. Можно также заметить, что проверка простоты, как правило, занимает не больше времени, чем генерация квазигрупп большинством из рассмотренных алгоритмов.

9.4 Проверка равномерности порождения на множестве правильных семейств

Параметрами были взяты $k = 2$, $n = 3$. Всего было 4 различных запуска тестов, на 10000, 25000, 50000, 100000 генераций. Во всех запусках были получены все 744 семейства порядка 3. Далее замерялись отношения числа возникновения каждого конкретного семейства к их общему числу. На графиках далее показаны частоты возникновения семейств и проведена прямая $y = 1/p$, где p — количество всех правильных семейств порядка 3.



Как видно из графиков, с ростом числа сгенерированных семейств, частоты возникновения каждого семейства сходятся к $1/p$. Этот факт наглядно показывает равномерность генерации.

Далее, на выборке, полученной во время эксперимента на 100000 генераций, проверялась гипотеза о том, что выборка взята из равномерного дискретного распределения. Проверка гипотезы осуществлялась с помощью критерия хи-квадрат, реализованного в библиотеке SciPy языка программирования Python. В результате было получено P-значение равное 0.584431, что позволяет, например, принять гипотезу о равномерности распределения с уровнем значимости 0.05.

9.5 Проверка полиномиальной полноты квазигрупп на основе правильных семейств

Параметрами были взяты $k = 4$, $n = 3$. После генерации правильное семейство преобразовывалось в квазигруппу. При этом, функции π_i генерировались случайным образом. Всего было сгенерировано 2500 квазигрупп порядка 64. Каждая полученная квазигруппа проверялась на простоту и афинность. Все полученные квазигруппы были неафинными, однако, лишь 27 из них оказались простыми. Остальные не обладали свойством простоты, а следовательно и не являлись полиномиально полными.

Дополнительно были порождены все квазигруппы для случая, когда $k = 2$, $n = 3$. Таких квазигрупп 108312. Среди них лишь 21952 являются простыми. Аффинными являются 2552 квазигруппы. Такое малое количество простых квазигрупп объяснимо тем, что значительная часть из правильных семейств в этом случае содержат константную функцию.

Утверждение 15. *Пусть в правильном семействе содержится константная функция. Тогда все получаемые из неё квазигруппы не являются простыми.*

Доказательство. Пусть задано правильное семейство $g_1, ..., g_n$, где g_i равна некоторой константе c, $1 \leq i \leq n$. Пусть (f_Q, Q) — квазигруппа, полученная из этого правильного семейства. Два элемента этой квазигруппы $q = (q_1, ..., q_n)$ и $p = (p_1, ..., p_n)$ назовем эквивалентными, если $q_i = p_i$. Такое разбиение множества Q, очевидно, является нетривиальным. Выберем пары наборов (a_1, b_1), $(a_2, b_2) \in Q^2$, таких, что $a_j \sim b_j$, $j = 1, 2$. Обозначим i-ую компоненту a_1, b_1 за x_i, а i-ую компоненту a_2, b_2 за y_i. Тогда i-ая компонента $f_Q(a_1, a_2)$ равна $x_i + y_i + c$. Но этому же равна i-ая компонента $f_Q(b_1, b_2)$. Следовательно, $f_Q(a_1, a_2) \sim f_Q(b_1, b_2)$. Т.е., f_Q сохраняет заданное нетривиальное разбиение, а значит, квазигруппа (f_Q, Q) не является простой. □

9.6 Проверка наличия подквазигрупп у квазигрупп на основе правильных семейств

Были проверены те же квазигруппы, что были получены в предыдущем пункте. Среди них 2499 квазигрупп имели нетривиальную подквазигруппу.

В случае $k = 2$, $n = 3$ нетривиальные подквазигруппы были у 102936 квазигрупп. Объяснение этого факта также кроется в большом количестве правильных семейств, содержащих константную функцию.

Утверждение 16. *Пусть в правильном семействе содержится константная функция. Тогда все получаемые из неё квазигруппы содержат нетривиальную подквазигруппу порядка k^{n-1}.*

Доказательство. Пусть, как и ранее, задано правильное семейство $g_1, ..., g_n$, где g_i равна некоторой константе c, $1 \leq i \leq n$, а (f_Q, Q) — квазигруппа, полученная из этого правильного семейства. Рассмотрим произведение элементов

$x = (x_1, ..., x_n)$ и $y = (y_1, ..., y_n)$, таких, что $x_i = y_i = -c$. Тогда i-ая компонента произведения x и y также равна $x_i + y_i + c = -c - c + c = -c$. Таким образом, множество $\{q = \{q_1, ..., q_n\} | q \in Q, q_i = -c\}$ является нетривиальной подквазигруппой порядка k^{n-1}. □

Касательно тривиальных подквазигрупп, имеет место следующее утверждение.

Утверждение 17 ([29]). *У каждой квазигруппы, полученной из правильного семейства функций, существует единственная тривиальная подквазигруппа.*

9.7 Проверка полиномиальной полноты квазигрупп на основе перестановок с полным дифференциалом

Поскольку число квазигрупп для обоих методов известно и составляет $(k!)^2$ для сетей Фейстеля и $k^{n-1}(k!)^{k^{n-2}}$ для регистров сдвига, можно породить их все для небольших порядков. Для того чтобы получить все квазигруппы нужного порядка, можно сгенерировать достаточно большое количество объектов и хранить их в $std :: unordered\_set$ (неупорядоченное множество). В силу особенностей $std :: unordered\_set$, каждый новый сгенерированный объект не будет добавлен в эту структуру данных, если он в ней уже присутствует. Таким образом, можно генерировать квазигруппы до тех пор, пока число объектов в $std :: unordered\_set$ не станет равно числу квазигрупп заданного порядка. Далее, перебирая все сгенерированные объекты, можно проверить их на аффинность и простоту.

k	n	Всего квазигрупп	Простых	Аффинных
2	2	4	4	4
2	3	16	16	16
2	4	128	112	64
2	5	4096	4032	256
3	2	18	18	18
3	3	1944	1890	162
4	2	96	64	32

Таблица 2: Результаты эксперимента для квазигрупп на основе регистров сдвига с обратной связью при небольших k и n.

Эти значения верны как для задания квазигрупп формулой $\sigma(x + y) - y$, так и для формулы $\sigma(x - y) + y$.

Также проводился эксперимент с генерацией 2500 квазигрупп при $k = 8$, $n = 2$. Все сгенерированные квазигруппы добавлялись в $std :: unordered\_set$. Отдельно рассматривались разные формулы для задания квазигрупп. Не все из 2500 квазигрупп, однако, оказались уникальными. Результаты представлены далее.

Формула	Уникальных квазигрупп	Простых	Аффинных
$\sigma(x+y)-y$	2495	2401	1
$\sigma(x-y)+y$	2491	2388	1

Таблица 3: Результаты эксперимента для 2500 квазигрупп на основе регистров сдвига с обратной связью при $k = 8$, $n = 2$.

Таким образом, кажется, что регистры сдвига с обратной связью порождают преимущественно полиномиально полные квазигруппы. При этом, из результатов для малых порядков возникает гипотеза, что число простых и число аффинных квазигрупп не зависят от выбора формулы.

Аналогичные эксперименты были проведены с сетями Фейстеля.

k	Всего квазигрупп	Простых	Аффинных
2	4	4	4
3	36	27	36
4	576	512	64
5	14400	14250	400

Таблица 4: Результаты эксперимента для квазигрупп на основе сетей Фейстеля при небольших k.

Как и в случае с регистрами сдвига, эта таблица верна для обеих формул, задающих квазигруппу.

Как и в предыдущем случае, проводился эксперимент с генерацией 2500 квазигрупп при $k = 8$.

Формула	Уникальных квазигрупп	Простых	Аффинных
$\sigma(x+y)-y$	2500	2496	0
$\sigma(x-y)+y$	2500	2498	0

Таблица 5: Результаты эксперимента для 2500 квазигрупп на основе сетей Фейстеля при $k = 8$.

Результаты экспериментов показывают, что оба метода генерации порождают множества, содержащие достаточно большое количество полиномиально полных квазигрупп.

9.8 Проверка наличия подквазигрупп у квазигрупп на основе перестановок с полным дифференциалом

Таким же образом, как в предыдущем эксперименте, можно получить все квазигруппы нужного порядка. Над каждым полученным объектом проводилось две операции — поиск тривиальных и поиск нетривиальных подквазигрупп. Результаты для регистров сдвига приведены далее.

k	n	Всего квазигрупп	Квазигрупп с тривиальными подквазигруппами	Квазигрупп с нетривиальными подквазигруппами
2	2	4	1	0
2	3	16	2	0
2	4	128	8	4
2	5	4096	128	78
3	2	18	18	0
3	3	1944	1944	216
4	2	96	8	24

Таблица 6: Случай, когда квазигруппы на основе регистров сдвига заданы формулой $\sigma(x + y) - y$.

k	n	Всего квазигрупп	Квазигрупп с тривиальными подквазигруппами	Квазигрупп с нетривиальными подквазигруппами
2	2	4	1	0
2	3	16	2	0
2	4	128	8	4
2	5	4096	128	78
3	2	18	2	0
3	3	1944	72	40
4	2	96	8	6

Таблица 7: Случай, когда квазигруппы на основе регистров сдвига заданы формулой $\sigma(x - y) + y$.

Дополнительно проверялись наборы из 2500 квазигрупп из прошлого эксперимента на наличие нетривиальных подквазигрупп.

Формула	Уникальных квазигрупп	Квазигрупп с нетривиальными подквазигруппами
$\sigma(x + y) - y$	2495	105
$\sigma(x - y) + y$	2491	44

Таблица 8: Результат эксперимента для 2500 квазигрупп на основе регистров сдвига с обратной связью при $k = 8$, $n = 2$.

Аналогичные таблицы были построены для сетей Фейстеля.

k	Всего квазигрупп	Квазигрупп с тривиальными подквазигруппами	Квазигрупп с нетривиальными подквазигруппами
2	4	1	0
3	36	36	9
4	576	144	16
5	14400	14400	150

Таблица 9: Случай, когда квазигруппы на основе сетей Фейстеля заданы формулой $\sigma(x + y) - y$

k	Всего квазигрупп	Квазигрупп с тривиальными подквазигруппами	Квазигрупп с нетривиальными подквазигруппами
2	4	1	0
3	36	4	3
4	576	36	16
5	14400	576	42

Таблица 10: Случай, когда квазигруппы на основе сетей Фейстеля заданы формулой $\sigma(x - y) + y$

Формула	Уникальных квазигрупп	Квазигрупп с нетривиальными подквазигруппами
$\sigma(x + y) - y$	2500	10
$\sigma(x - y) + y$	2500	7

Таблица 11: Результат эксперимента для 2500 квазигрупп на основе сетей Фейстеля при $k = 8$.

Таким образом, по результатам экспериментов кажется, что лишь небольшая часть квазигрупп имеет нетривиальную подквазигруппу. Результаты для тривиальных подквазигрупп соответствуют доказанным ранее утверждениям.

9.9 Проверка наличия подквазигрупп в общем случае

В данном эксперименте генерировались наборы из 1000 уникальных квазигрупп разных порядков. Для генерации был использован метод Джейкобсона-Мэтьюза, чтобы проверить частоту возникновения подквазигрупп в общем случае. Результаты представлены в таблице далее.

Порядок	Квазигрупп с тривиальными подквазигруппами	Квазигрупп с нетривиальными подквазигруппами
8	673	16
12	648	3
16	643	3
24	632	1
32	628	0
48	664	0
64	616	0

Таблица 12: Проверка 1000 квазигрупп различных порядков на наличие подквазигрупп.

Видно, что в общем случае квазигруппы с нетривиальными подквазигруппами возникают довольно редко. При этом, судя по всему, наличие тривиальной подквазигруппы не редкость. При всех рассмотренных порядках таких квазигрупп было больше половины.

10 Заключение

Были рассмотрены несколько способов задания конечных квазигрупп. Наилучшими с точки зрения скорости генерации являются алгоритмы на основе полных перестановок. Алгоритм генерации случайного правильного семейства с последующим преобразованием в квазигруппу оказался наименее эффективным. Также, квазигруппы, получаемые из правильных семейств функций, оказываются преимущественно не простыми, а следовательно, не полиномиально полными. Квазигруппы на основе перестановок с полным дифференциалом, напротив, оказались в большей степени полиномиально полными и не содержащими нетривиальных подквазигрупп, что делает их потенциально полезными в вопросах криптографии.

Для двух рассматриваемых видов квазигрупп на основе полных перестановок были получены условия существования тривиальных подквазигрупп, а также было найдено количество таких квазигрупп, содержащих тривиальные подквазигруппы. Для полных перестановок также было найдено обобщение на случай n-квазигрупп.

Алгоритм проверки аффинности был модифицирован. Новая версия работает значительно быстрее в случае проверки аффинной квазигруппы.

Среди дальнейших направлений исследований можно выделить следующие: понижение сложности проверки аффинности до квадратичной; получение способа порождения равномерного распределения на множестве квазигрупп, получаемых из одного правильного семейства; поиск новых классов квазигрупп, порождаемых функционально.

Список литературы

[1] V. T. Markov, A. V. Mikhalev и A. A. Nechaev, "Nonassociative algebraic structures in cryptography and coding", *Journal of Mathematical Sciences*, т. 245, № 2, с. 178—196, 2020.

[2] D. Gligoroski, "On the S-box in GAGE and InGAGE", 2019, [Электронный ресурс], дата обращения: 31.03.2023. url: `http://gageingage.org/upload/LWC2019NISTWorkshop.pdf`.

[3] D. Gligoroski, R. S. Ødegå̊rd, M. Mihova, S. J. Knapskog, A. Drapal, V. Kl´ıma, J. Amundsen и M. El-Hadedy, "Cryptographic hash function EDON-R'", *Proceedings of the 1st International Workshop on Security and Communication Networks*, с. 1—9, 2009.

[4] S. Markovski и A. Mileva, "NaSHA — family of cryptographic hash functions", *The First SHA-3 Candidate Conference*, 2009.

[5] D. Gligoroski, R. S. Ødegå̊rd, R. Jensen, L. Perret, J. C. Faug'ere, S. Knapskog и S. Markovski, "MQQ–SIG: an ultra-fast and provably CMA resistant digital signature scheme", *INTRUST'11: Proceedings of the Third international conference on Trusted Systems*, с. 184—203, 2011.

[6] C. Shannon, "Communication theory of secrecy systems", *Bell System Technical Journal*, т. 28, № 4, с. 656—715, 1949.

[7] М. М. Глухов, "О применениях квазигрупп в криптографии", *Прикладная дискретная математика*, № 2, с. 28—32, 2008.

[8] P. J. Cameron, "Almost all quasigroups have rank 2", *Discrete Mathematics*, т. 106–107, с. 111—115, 1992.

[9] G. Horvath, G. L. Nehaniv и C. Szab´o, "An assertion concerning functionally complete algebras and NP-completeness", *Theoretical Computer Science*, т. 407, с. 591—595, 2008.

[10] B. Larose и L. Zadori, "Taylor terms, constraint satisfaction and the complexity of polynomial equations over finite algebras", *International Journal of Algebra and Computation*, т. 16, с. 563—581, 2006.

[11] С. В. Яблонский, "Введение в дискретную математику", *Наука*, 1986.

[12] J. Hagemann и C. Herrmann, "Arithmetical locally equational classes and representation of partial functions", *Universal Algebra*, с. 345—360, 1982.

[13] В. А. Носов, "О построении классов латинских квадратов в булевой базе данных", *Интеллектуальные системы*, т. 4, № 3–4, с. 307—320, 1999.

[14] В. А. Носов, "Построение параметрического семейства латинских квадратов в векторной базе данных", *Интеллектуальные системы*, т. 8, № 1–4, с. 517—528, 2004.

[15] В. А. Носов и А. Е. Панкратьев, "Латинские квадраты над абелевыми группами", *Фундаментальная и прикладная математика*, т. 12, № 3, с. 65—71, 2006.

[16] A. Sade, “Quasigroups automorphes par le groupe cyclique”, *Canadian Journal of Mathematics*, т. 9, с. 321—335, 1957.

[17] S. Chakrabarti, A. V. Galatenko, V. A. Nosov, A. E. Pankratiev и S. K. Tiwari, “Quasigroups generated by shift registers and Feistel networks”, *Quasigroups and Related Systems*, (in press).

[18] S. Markovski и A. Mileva, “Generating huge quasigroups from small non-linear bijections via extended Feistel function”, *Quasigroups and Related Systems*, с. 97—106, 2009.

[19] V. T. Hoang и P. Rogaway, “On Generalized Feistel Networks”, *Advances in Cryptology — CRYPTO 2010*, с. 613—630, 2010.

[20] M. T. Jacobson и P. Matthews, “Generating uniformly distributed random Latin squares”, *Journal of Combinatorial Designs*, с. 405—437, 1996.

[21] A. V. Galatenko, A. E. Pankratiev и V. M. Staroverov, “Generation of Proper Families of Functions”, *Lobachevskii Journal of Mathematics*, т. 53, № 3, с. 571—581, 2022.

[22] I. A. Schurr, “Unique sink orientations of cubes”, *Doctoral Thesis*, 2004.

[23] R. A. Fisher и F. Yates, “Statistical tables for biological, agricultural and medical research”, *London: Oliver & Boyd*, 1948.

[24] А. В. Галатенко и А. Е. Панкратьев, “О сложности проверки полиномиальной полноты конечных квазигрупп”, *Дискретная математика*, с. 3—11, 2018.

[25] A. Clifford и G. Preston, “Light’s associativity test”, *The Algebraic Theory of Semigroups*, т. 1, с. 7—8, 1961.

[26] R. E. Tarjan, “Determining whether a groupoid is a group”, *Information Processing Letters*, т. 1, с. 120—124, 1972.

[27] A. V. Galatenko, A. E. Pankratiev и V. M. Staroverov, “Efficient verifiation of polynomial completeness of quasigroups”, *Lobachevskii Journal of Mathematics*, с. 1444—1453, 2020.

[28] А. В. Галатенко, А. Е. Панкратьев и В. М. Староверов, “Об одном алгоритме проверки существования подквазигрупп”, *Чебышевский сборник*, с. 76—89, 2021.

[29] A. V. Galatenko, V. A. Nosov и A. E. Pankratiev, “Latin squares over quasigroups”, *Lobachevskii Journal of Mathematic*, 2020.