Федеральное государственное бюджетное образовательное учреждение
высшего образования «Сибирский государственный университет
науки и технологий имени академика М.Ф. Решетнёва»

На правах рукописи

Кишкан Владимир Владимирович

РАЗРАБОТКА И ОЦЕНКА СЛОЖНОСТИ АЛГОРИТМОВ, НАХОДЯЩИХ ПРИМЕНЕНИЕ В АППАРАТНОМ И ПРОГРАММНОМ ОБЕСПЕЧЕНИИ МНОГОПРОЦЕССОРНЫХ СИСТЕМ

Специальность 05.13.17 – Теоретические основы информатики

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
доктор физико-математических наук, профессор
Кузнецов Александр Алексеевич

Красноярск 2020

Оглавление

Введение

0.1 Проблематика исследования

В настоящее время постоянно увеличивающийся спрос на облачные вычисления приводит к росту крупномасштабных центров обработки данных (ЦОД). Современные ЦОД содержат сотни тысяч узлов, соединенных между собой сетью. Топология такой сети, т.е. способ соединения узлов, является ключевым звеном, от которого зависит быстродействие, отказоустойчивость, надежность и другие характеристики ЦОД. По этой причине проектирование сети является очень важной задачей, включающей в себя поиск моделей графов, которые имеют хорошие топологические свойства и позволяют использовать эффективные алгоритмы маршрутизации. Этими качествами обладают графы Кэли, имеющие такие привлекательные топологические свойства как высокая симметрия, иерархическая структура, рекурсивная конструкция, высокая связность и отказоустойчивость [60]. Определение графа Кэли подразумевает, что вершины графа являются элементами некоторой алгебраической группы. Выбор группы и ее порождающих элементов позволяет получить граф, отвечающий необходимым требованиям по диаметру, степени вершин, количеству узлов и т. д. [65].

Пусть $G := \langle X \rangle$ – конечная группа, порожденная упорядоченным множеством $X := \{x_1 \prec x_2 \prec \ldots \prec x_m\}$, которое также называют алфавитом. Множество всех слов (строк) над алфавитом X будем обозначать X^*. Пусть $w := x_1 x_2 \ldots x_l$ – слово над X и $|w| := l$ – его длина. На множестве X^* также определим отношение порядка. Пусть v и w – два произвольных слова в алфавите X. Тогда $v \prec w$, если $|v| < |w|$, а в случае равенства длин слов,

меньшее слово будет определяться согласно введенному лексикографическому порядку на порождающих. Если необходимо подчеркнуть, что строка $v \in X^*$ соответствует элементу $g \in G$, то мы будем писать v_g. Строку v будем называть минимальным словом элемента g, если для всех других $w \in X^*$, таких что $v_g = w_g$, будет выполняться $v \prec w$. Очевидно, что каждому $g \in G$ соответствует уникальное минимальное слово. Длиной элемента группы g будем называть длину его минимального слова v, т. е. $|g| := \min\{|v_g| \ : \ v_g \in X^*\}$. Заметим, что в общем случае задача по определению минимального слова является NP-сложной [56].

Графом Кэли $\Gamma := Cay(G, X)$ группы G относительно X называют ориентированный невзвешенный помеченный граф с множеством вершин $V(\Gamma) := \{g \mid g \in G\}$ и множеством ребер $E(\Gamma) := \{(g, gx) \mid x \in X, g \in G\}$. Пусть $(g, gx) \in E(\Gamma)$, тогда генератор x называют меткой данного ребра. Если $X = X \cup X^{-1}$, то граф Γ будет неориентированным. Будем считать, что единичный элемент $e \notin X$, т. е. в Γ нет петель. Как известно [61], кратчайшее расстояние между двумя произвольными вершинами графа g и h, которое мы обозначим $d(g, h)$, равно длине минимального слова элемента $g^{-1}h$, т. е. $d(g, h) := |g^{-1}h|$.

Остановимся на известных к настоящему времени алгоритмах маршрутизации на графах Кэли. Традиционные методы, такие как алгоритмы Дейкстры или Беллмана-Форда, могут использоваться на графах любого вида, но требуют значительных пространственных и временных ресурсов [78]. Для некоторых семейств графов Кэли существуют особые алгоритмы маршрутизации, которые в отличие от традиционных методов, используют топологические характеристики графа, уменьшая при этом временную и/или пространственную сложность. Сюда можно отнести такие семейства графов как гиперкуб [83], бабочка [84] и звездный граф [62], которые являются графами Кэли. В работе [51] представлен алгоритм маршрутизации для pancake и звездных графов, основанный на сортировке перестановок. Однако этот подход не обеспечива-

ет маршрутизацию по кратчайшему пути. К. Тэнг и Б. Арден доказывают [85], что все конечные графы Кэли могут быть представлены обобщенными хордальными кольцами, а затем предлагают итерационный алгоритм маршрутизации, основанный на таблице поиска. Пространственная сложность такого алгоритма составляет $O(|G|^2)$, а временная – $O(D)$, где $|G|$ и D – размер и диаметр сети соответственно. Л. Вэнг и К. Тэнг в [88] для поиска кратчайших путей на графе Бореля предлагают алгоритм, который сначала вычисляет в автономном режиме таблицу маршрутизации от одного узла ко всем остальным; затем, используя свойство транзитивности графа Кэли, создает таблицу маршрутизации для всех узлов. Вычислительная сложность данного алгоритма ограничена $O(\log_4 |G|)$, а пространственная – $O(|G|^2 D)$. В [71] представлен распределенный отказоустойчивый алгоритм маршрутизации на графе Бореля. Этот двухфазный алгоритм использует два типа таблиц маршрутизации: статическую и динамическую. В статье [80] представлен алгоритм маршрутизации для специального класса графов Кэли, используемых в качестве топологии сети беспроводного ЦОД. Данный алгоритм является двухуровневым: для отправки сообщений между серверами в одной стойке и серверами в разных стойках. В работах [63, 26, 27, 28, 64, 25] исследуются графы Кэли конечных групп Бернсайда периода 3, 4, 5 и 7.

Монография [57] представляет собой фундаментальную работу о взаимосвязи алгебраических групп и конечных автоматов. В этом случае на группе $G = \langle X \rangle$ определяется автоматная структура специального вида, используя которую, можно вычислить минимальное слово для любого элемента группы. Согласно [57] конечный автомат группы $\mathcal{A}$ считывает произвольное слово $w_g \in X^*$, обрабатывает его и выдает минимальное слово элемента g. При этом время T_0 обработки слова w будет пропорционально квадрату его длины, т. е. $T_0 = O(|w|^2)$. Используя данный результат, в работе [53] был предложен алгоритм поиска кратчайшего пути на графе Кэли (обозначим его A–0), при этом его вычислительная сложность ограничена $O(D^2)$, а пространственная –

$M_0 = O(|X| \cdot |G| + |\mathcal{A}|)$, где $|\mathcal{A}|$ – число состояний автомата группы.

Заметим также, что при построении больших сетей будет выполняться $|X| \ll |G|$, отсюда следует, что $O(|X| \cdot |G|) = O(|G|)$.

Отметим, что все представленные выше алгоритмы маршрутизации могут быть отнесены в одну из следующих категорий: а) те, которые предназначены для конкретных графов Кэли; б) универсальные с высокой пространственно-временной сложностью и в) с низкой сложностью, которые не обеспечивают кратчайших путей.

Для многопроцессорных вычислительных систем (МВС) актуален ряд направлений исследований. Одним из них являются исследования, связанные с перспективными языками программирования, которые могут разрабатываться в дальнейшем для обеспечения работы с МВС.

Практически все известные в настоящее время языки программирования являются контекстно–свободными языками (кс–языками), порождёнными контекстно–свободными грамматиками (кс–грамматиками). Рассмотрим необходимые определения.

Исходным объектом в теории программирования, а также теории кс–языков и грамматик, заложенной в работах выдающегося лингвиста Н. Хомского и других исследователей [48, 49, 50], является алфавит, который удобно разделять на две группы символов:

$$z_1, \ldots, z_n, x_1, \ldots, x_m.$$

Обычно символы $x_1, \ldots, x_m$ из второй группы называются терминальными символами и образуют словарь контекстно–свободного языка [13, 18, 19]. Применительно к языкам программирования к терминальным символам относятся цифры, буквы, вспомогательные знаки, а также состоящие из них «блоки», обозначающие, например, операторы языка программирования. Такие операторы могут обозначаться в виде некоторой последовательности букв и других символов, но сами рассматриваются как неделимые символы (например, в некоторых языках программирования операторы GOTO, RETURN и др., которые

сами рассматриваются как неделимые символы алфавита) [30, 33, 32, 34, 38, 47].

Символы другой группы $z_1, \ldots, z_n$ называются нетерминальными. Они играют вспомогательную роль, не присутствуя явно в тексте программ, и нужны для задания грамматики — совокупности грамматических правил, порождающих кс–язык.

По правилам грамматики формируются мономы от терминальных символов $x_1, \ldots, x_m$, которые интерпретируются как правильные предложения языка [10, 11, 12, 72]. Такие мономы рассматриваются как корректные, в отличие от произвольных мономов, которые могут не соответствовать правилам грамматики и, таким образом, являются некорректными.

Над символами алфавита определены три операции: 1) некоммутативная операция формального умножения — операция конкатенации; 2) коммутативная операция формальной суммы (вместе с этими операциями алфавит образует такую алгебраическую структуру как полукольцо); 3) дополнительная коммутативная операция умножения элементов алфавита, а значит, и мономов на числа (обычно достаточно рассматривать действительные числа, но в некоторых исследованиях нужны комплексные числа) [12, 52, 58, 72].

Далее, над полукольцом можно рассматривать символьные многочлены и формальные степенные ряды (ФСР) с числовыми коэффициентами. Кс–языком называется такой ФСР, членами которого являются все корректные мономы, которые порождены данной грамматикой [12, 72]. Таким образом, язык представляет собой совокупность всех возможных (корректных мономов) правильных предложений, порождённых его грамматикой.

Важно отметить, что в теории кс–языков имеется эффективный инструмент исследования — система уравнений с многочленами специального вида от символов алфавита, называемая (собственной) системой уравнений Хомского — Шютценберже. А именно, система уравнений Хомского — Шютценберже имеет вид

$$z_j = Q_j(z, x), \;\; j = 1, \ldots, n, \tag{1}$$

при этом многочлены в правых частях удовлетворяют естественно объяснимым требованиям: во-первых, $Q_j(0,0)=0$, во-вторых, многочлены $Q_j(z,0)$ не содержат линейных членов, что означает отсутствие в правилах грамматики простого переобозначения нетерминальных символов [9, 8, 11, 12, 72].

Предполагается, что указанная система решается относительно нетерминальных символов $(z_1,\ldots,z_n)=z$, а решение ищется в виде ФСР от терминальных символов $(x_1,\ldots,x_m)=x$:

$$z=z(x)=(z_1(x),\ldots,z_n(x)),$$

эти ФСР, будучи подставленными в систему уравнений Хомского — Шютценберже, дадут верные равенства.

Символ z_1 играет особую роль как начальный символ в данном языке: стартовый для написания программы, либо обозначающий начало предложения в естественном языке. По этой причине его выражение в виде ФСР $z_1(x)$ и есть кс–язык, который порождён грамматикой, записанной в виде системы уравнений Хомского — Шютценберже [11, 12, 72].

Вообще заметим, что именно из начального символа z_1 при помощи правил подстановки (продукций) выводятся все корректные мономы языка, интерпретируемые применительно к естественным языкам как правильные предложения, а к языкам программирования — как правильные программы [1, 3, 4, 5, 6, 7, 31].

Для того, чтобы сформулировать расширенную проблему синтаксического анализа (разбора), нужно конкретизировать структуру правых частей системы уравнений Хомского — Шютценберже с точки зрения правил грамматики.

Для этого рассмотрим грамматику кс–языка, которая является совокупностью правил подстановки (продукций):

$$z_j \to\ q_{j1}(z,x),\ \ldots\ , z_j \to\ q_{jp_j}(z,x),\ j=1,\ \ldots\ ,n, \tag{2}$$

где $q_{jk}(z,x)$ — мономом от некоммутативных символов алфавита с числовым коэффициентом, который равен 1.

Вывод корректных мономов осуществляется так. Продукции сначала следует применить к начальному символу z_1, а затем к другим получающимся мономам неограниченное число раз и в произвольном порядке, что позволяют продуцировать новые мономы от терминальных символов и нетерминальных символов — корректный моном языка. Все корректные мономы образуют соответствующий кс–язык.

Теперь можно отметить, что многочлены $Q_j(z,x),\ j=1,\ \ldots\ ,n$, стоящие в правой части системы уравнений Хомского — Шютценберже, имеют следующую структуру:

$$Q_j(z,x) = \ q_{j1}(z,x) + \ \cdots \ + q_{jp_j}(z,x),$$

$$j=1,\ \ldots\ ,n.$$

Итак, перейдём к проблеме синтаксического анализа мономов кс–языка.

В монографии, написанной академиком В. М. Глушковым в соавторстве с Г. Е. Цейтлиным и Е. Л. Ющенко сказано: «Одной из важных проблем в разработке современных систем программирования является проблема синтаксического анализа программ. Процесс синтаксического анализа программы состоит в распознавании правильности данной программы, т. е. её принадлежности к рассматриваемому алгоритмическому языку. Этот этап называется этапом синтаксического контроля программы. Одновременно с контролем осуществляется описание синтаксической структуры правильных программ, подобно тому как производится грамматический разбор предложений в естественных языках» [12, с. 234].

Таким образом, выделяют две составляющие проблемы синтаксического анализа, первая часть, называемая проблемой принадлежности или этапом синтаксического контроля, состоит в том, чтобы определить, принадлежит ли моном данному кс–языку, т. е. может ли быть получен из начального символа z_1 при помощи продукций. Одновременно с решением первой части проблемы синтаксического анализа решается и вторая часть проблемы — описание синтаксической структуры монома.

Такое описание понимается различными авторами по-разному. Например, в упомянутой монографии В. М. Глушкова, Г. Е. Цейтлина и Е. Л. Ющенко отмечено: «В математической лиигвистике широко распространён способ представления синтаксической структуры языковых объектов в виде деревьев грамматического разбора» [12, с. 303].

Например, проблема синтаксического анализа ставится следующим образом — требуется разработать алгоритм, для того чтобы:

— установить, какие правила подстановки и сколько раз использовались при выводе данного монома, при этом порядок использования правил подстановки не имеет значения [42, 43];

— установить, какие правила подстановки, сколько раз и в каком порядке использовались при выводе этого монома, т. е. построить хотя бы один из возможных выводов монома [3].

Как видно, для полного решения проблемы синтаксического анализа возможен также подход, при котором необходимо построить сразу все возможные выводы монома, если таких несколько.

Кроме того, отметим следующее важное обстоятельство, которому исследователи уделяют большое внимание — разработать беступиковый (бесперебойный, безостановочный, беспереборный) алгоритм синтаксического анализа мономов кс–языка. Так, в [12, с. 248] сказано: «С точки зрения практических приложений значительный интерес представляют формализмы для описания языков, допускающие беступиковый (беспереборный) синтаксический анализ».

Если алгоритм таков, что может приводить к тупикам, то он должен предусматривать возвраты с анализом определённой предыстории алгоритма, что значительно усложняет алгоритм. Однако, для произвольной кс–грамматики беступикового алгоритма синтаксического анализа на основе свёртки или развёртки не существует, отмечается лишь, что «важным классом однозначных кс–грамматик, допускающих беступиковый анализ разверткой, являются $LL(k)$–грамматики» [12, с. 259].

Таким образом, во–первых, имеющиеся алгоритмы достаточно сложные, во–вторых, могут решать ограниченные задачи, например, нахождения одного из возможных выводов монома.

Далее, будем называть расширенной проблемой синтаксического анализа мономов кс–языка проблему разработки беступикового алгоритма, который позволяет установить, может ли быть выведен моном при помощи системы продукций кс–языка (решить проблему принадлежности), а также найти сразу все выводы этого монома.

Вывод монома можно представить следующим образом: определить, какие продукции, сколько раз и в каком порядке применяются для вывода этого монома — именно в таком виде будем искать описание возможных выводов. Нахождение вывода в таком виде, очевидно, равносильно построению дерева вывода.

Подчеркнём, что такие алгоритмы в настоящее время не известны, поскольку для произвольных кс–грамматик разработанные известные алгоритмы (свёрткой–развёрткой и др.) могут приводить к тупикам [12].

Естественно, при разработке алгоритмов синтаксического анализа произвольного кс–языка значительный интерес представляет вопрос об их сложности [2, 41, 66, 79, 86, 70].

Для произвольной кс–грамматики одним из эффективных алгоритмов является алгоритм Кока — Янгера — Касами (CYK – алгоритм или CKY – алгоритм), позволяющий установить, можно ли в заданной кс–грамматике вывести заданную строку, и если да, то предоставить один из её выводов.

Пусть дан моном (программа) w степени (длины) N. Обычно сложность алгоритма синтаксического анализа выражают через число длину монома N в виде $O(f(N))$, где $f(N)$ — некоторая функция от N.

Для многих алгоритмов в теоретической информатике функция $f(N)$ является мономом (в этом случае говорят, что сложность полиномиальная, она считается небольшой) либо экпонента (тогда сложность считается значитель-

ной) [2, 66, 86]. Некоторые алгоритмы, естественно, имеют сложность, которая больше, чем экспоненциальная.

В процессе разработки языка программирования может понадобиться проводить синтаксический анализ тестовых программ, которые удобно рассматривать как мономы кс–языка, порождаемого системой продукций.

Алгоритм Кока — Янгера — Касами является универсальным в том смысле, что он применим к кс–грамматике в нормальной форме Хомского, к которой можно привести произвольную кс–грамматику. Сложность алгоритма Кока — Янгера — Касами — полиномиальная и равна N^3 [86].

Однако, с точки зрения разработки перспективных языков программирования, в том числе для МВС, известные алгоритмы синтаксического анализа не применимы. Как правило, известные алгоритмы реализованы в виде специальных программ (парсеров), предназначенных для анализа выражений, написанных на определённом языке программирования.

Однако, в ситуации, когда разрабатывается новый язык программирования, никаких парсеров, естественно нет. В случае, когда необходимо провести синтаксический анализ (разбор) некоторого выражения относительно совокупности грамматических правил, находящихся в стадии разработки, могут быть полезными различные алгоритмы, в том числе имеющие высокую сложность. Как правило, тестируются выражения ограниченной длины, и потому высокая сложность алгоритма в такой ситуации не играет роли — важно, чтобы алгоритм был простым в программной реализации и вполне конструктивным. Сложность такого алгоритма может быть даже выше экспоненциальной, что вполне допустимо для случаев, когда длина программы N не слишком велика.

Таким образом, разработка беступиковых конструктивных алгоритмов для решения расширенной проблемы синтаксического анализа является достаточно актуальной задачей [23, 55, 69, 75, 87].

0.2 О содержании диссертации

В данном разделе перед тем, как кратко изложить основное содержание диссертации, дадим общую характеристику работы.

Объект исследования. Графы Кэли, порождённые конечными группами подстановок, а также контекстно–свободные языки и грамматики (связанные с аппаратно–программным обеспечением МВС).

Предмет исследования. Алгоритмы маршрутизации на графах Кэли, а также алгоритмы решения расширенной проблемы синтаксического анализа мономов контекстно–свободных языков программирования.

Целью диссертационной работы является создание универсального алгоритма маршрутизации на графах Кэли групп подстановок, имеющего лучшие пространственно–временные характеристики по сравнению с известными алгоритмами (аппаратный аспект обеспеченич МВС), а также разработка беступикового алгоритма решения расширенной проблемы синтаксического анализа мономов контекстно–свободных языков (программный аспект обеспечения МВС).

Поставленная цель достигается путем решения следующих **задач** диссертационного исследования:

1. разработать эффективный алгоритм маршрутизации на графах Кэли, заданных конечными группами подстановок;
2. исследовать ранее неизвестные характеристики для некоторых графов Кэли;
3. разработать беступиковый алгоритм решения расширенной проблемы синтаксического анализа мономов контекстно–свободных языков;
4. оценить сложность беступикового алгоритма решения расширенной проблемы синтаксического анализа, позволяющего установить все возможные выводы монома контекстно–свободного языка.

Соответствие диссертации паспорту специальности. Диссертационная работа соответствует области исследований специальности 05.13.17 — Теоретические основы информатики по п. 10 *«Разработка основ математической теории языков и грамматик, теории конечных автоматов и теории графов»*.

Методы исследования диссертационной работы. Основные результаты получены на основе методов алгебры, в том числе алгебры многочленов и теории групп, дискретной математики, теории графов, теории формальных языков и грамматик.

Научная новизна:

1. Разработан новый эффективный алгоритм, позволяющие вычислять кратчайшие маршруты между вершинами графа Кэли $Cay(G, X)$, заданного произвольной конечной группой подстановок $G = \langle X \rangle$. Обоснована корректность представленного алгоритма, даны оценки его сложности. В этом случае, для произвольного слова $w \in X^*$, проблема поиска минимального слова разрешается за время $T = O(|w|)$.

2. При помощи МВС вычислены ранее неизвестные характеристики графов Кэли модифицированной пузырьковой сортировки $MBS(n)$ для случаев $n = 14$ и 15. Получены новые результаты о некоторых графах Кэли, заданных конечными двупорождёнными группами периода 7.

3. Впервые разработан беступиковый алгоритм синтаксического анализа, использующий иерархию маркированных скобок и позволяющий решать уточнённую и расширенную проблему синтаксического анализа, включая нахождение всех выводов монома (программы), имеющий применения в условиях разработки языков программирования для МВС.

4. Получена оценка сложности алгоритма синтаксического анализа, использующего иерархию маркированных скобок, особенно эффективного при разработке новых языков программирования.

Теоретическая и практическая значимость работы. Работа носит теоретический характер. Новый эффективный алгоритм для вычисления кратчайших маршрутов между вершинами графа Кэли $Cay(G, X)$, порожденного произвольной конечной группой подстановок $G = \langle X \rangle$ может быть использован при разработке аппаратно–программного обеспечения МВС. Предложенный алгоритм решения расширенной проблемы синтаксического анализа может быть эффективно использован при разработке языков программирования, в том числе для обеспечения МВС, в условиях отсутствия парсеров.

Положения, выносимые на защиту диссертационной работы. На защиту выносятся следующие основные результаты:

1. Новый эффективный алгоритм для вычисления кратчайших маршрутов между вершинами графа Кэли $Cay(G, X)$, порожденного произвольной конечной группой подстановок $G = \langle X \rangle$.
2. Ранее неизвестные характеристики некоторых графов Кэли, полученных при помощи МВС.
3. Новый беступиковый алгоритм синтаксического анализа решения уточнённой и расширенной проблемы синтаксического анализа на основе иерархии вложенных скобок.
4. Оценка сложности разработанного беступикового алгоритма синтаксического анализа мономов контекстно–свободных языков на основе иерархии вложенных скобок.

Достоверность результатов работы подтверждается математическими доказательствами основных положений.

Апробация результатов работы. Результаты диссертационной работы были доложены автором на следующих всероссийских и международных конференциях:

— Международной научно-практической конференции «Решетнёвские чтения» (Красноярск, 2017–2019 гг.);

— Всероссийской конференции «Сибирская научная школа–семинар с международным участием «Компьютерная безопасность и криптография» — Sibecrypt (Томск, 2019 г.);

— Всероссийской с международным участием научно-методической конференции, посвященной 80-летию профессора Ларина С. В. (Красноярск, 2019 г.).

Результаты работы обсуждались на научно-исследовательских семинарах в Сибирском государственном университете науки и технологий имени академика М. Ф. Решетнёва;

Публикации по теме диссертационной работы. По результатам диссертационного исследования опубликовано 16 работ, из которых:

— 4 в журналах, рекомендованных ВАК;

— 2 свидетельства на программное обеспечение;

— 2 статьи в изданиях, индексируемых Web of Science и Scopus.

Из статей, написанных в соавторстве, в диссертацию включены результаты, принадлежащие лично автору.

Структура и объём работы. Диссертационная работа изложена на 101 странице и состоит из введения, двух глав, заключения, списка литературы из 104 источников и приложения.

Рассмотрим краткое содержание Главы 1 **«Алгоритмы на графах Кэли групп подстановок»**.

В разделе 1.1 «Алгоритмы на группах подстановок» представлены основные определения, а также новые вспомогательные алгоритмы.

После чего приведён алгоритм А–1, который получив на входе слово, состоящее из произведения элементов $g_1 g_2 \dots g_l$ группы G, ее базу B, полное семейство представителей смежных классов U и его инверсию $\hat{U}$, а также вспомогательный массив A, возвращает каноническое представление $u_m u_{m-1} \dots u_1$ данного произведения.

Пусть $T_i = O(f)$ и $M_i = O(f)$ — верхние асимптотические оценки вы-

числительной и пространственной сложности i–го алгоритма соответственно. Доказана следующая

Лемма 1.2. *Алгоритм А–1 корректен и* $T_1 = O(l \cdot m + m^2)$.

Разложение элементов в каноническом виде дает возможность эффективно нумеровать все $g \in G$, используя метод перечисления элементов кортежа в смешанной системе счисления [21]. Пусть $c := (c_m, c_{m-1}, \ldots, c_1)$ – основание смешанной системы счисления, в которой $c_1 := 1$ и $c_i = c_{i-1} \cdot |U^{(i-1)}|$ для $i \geq 2$.

Пусть $g := u_m u_{m-1} \ldots u_1$. Определим биективное отображение

$$\mathcal{N}\colon u_m u_{m-1} \ldots u_1 \longrightarrow (a_m, a_{m-1}, \ldots, a_1),$$

где $a_i \in [0, |U^{(i)}| - 1]$ – номер элемента u_i в $U^{(i)}$.

Заметим, что вектор $(a_m, \ldots, a_1)$ представляет собой число $\mathcal{N}(g) \in [0, |G| - 1]$ в системе счисления со смешанным основанием $(c_m, \ldots, c_1)$. Пусть $k := \mathcal{N}(g)$ – номер элемента g. Далее представлена лемма в которой даны оценки сложности указанных операций.

Лемма 1.3. *Вычислительная сложность* $\mathcal{N}(g)$ *и* $\mathcal{N}^{-1}(k)$ *не превышает* $O(m)$.

Отметим, что для нумерации элементов группы нам необходимо знать ее базу B и полное семейство представителей смежных классов U. Для их вычисления мы будем использовать известный алгоритм Шрайера-Симса, предложенный Ч. Симсом в 1970 году [81]. В настоящее время существует множество его модификаций [76]. Наиболее эффективные версии алгоритма имеют низкую вычислительную сложность и реализованы в таких системах компьютерной алгебры, как GAP, Magma и Mathematica, а также в библиотеке SymPy для языка Python.

Далее рассмотрим краткое содержание раздела 1.2 «Алгоритмы маршрутизации на графах Кэли».

В начале раздела дано определение таблицы маршрутизации $P_{2\times|G|}$, которую также называют родительским деревом [45].

Затем представлен новый алгоритм А–2, который, получив на входе порождающее множество группы X, ее базу B, полное семейство представителей смежных классов U и его инверсию $\hat{U}$, а также вспомогательный массив A, возвращает указанную таблицу маршрутизации P. Доказана следующая

Теорема 1.1. *Алгоритм А–2 корректен и* $T_2 = O(m^2 \cdot |X| \cdot |G|)$.

Продолжает изложение данного раздела алгоритм А–3, который при помощи таблицы маршрутизации P вычисляет кратчайший путь $w := x_1 x_2 \dots x_s$ между вершинами $a, b \in Cay(G, X)$, где $x_i \in X$.

Пусть D – диаметр графа. В сформулированной ниже теореме обоснована корректность алгоритма А–3, а также получена оценка его вычислительной сложности.

Теорема 1.2. *Алгоритм А–3 корректен,* $T_3 = O(m^2 + D)$ *и* $M_3 = O(m \cdot n + |G|)$.

Заметим, что во многих прикладных задачах при исследовании графа Кэли $\Gamma := Cay(G, X)$ порядок порождающей группы G значительно превышает ее степень, т. е. $m \leq n \ll |G|$. В этом случае $M_3 = O(|G|)$.

В разделе 1.3 «Проблема минимального слова» представлен алгоритм А–4, который вычисляет минимальное слово w из произвольной строки $v := x_1 x_2 \dots x_r$ в алфавите порождающих X. Данный алгоритм является модифицированной версией алгоритма А–3. Доказана

Теорема 1.3. $T_4 = O(|v|)$ *и* $M_4 = O(m \cdot n + |G|)$.

В разделе 1.4 «Сравнительный анализ алгоритмов» показано, что алгоритм А–4 имеет значительное преимущество перед алгоритмом из [53], основанном на использовании конечных автоматов.

Представленный в данной главе алгоритм А–3 послужит отправной точкой для создания новых ресурсно-эффективных алгоритмов маршрутизации. Здесь можно выделить два направления. Во-первых, создание алгоритмов, учитывающих топологию сети. В этом случае алгоритмы будут проектироваться для конкретных классов графов Кэли. Во-вторых, разработка гибридных ал-

горитмов, включающих в себя как статическую, так и динамическую таблицы маршрутизации. Это позволит рассчитывать оптимальные маршруты в зависимости от текущего состояния сети.

Раздел 1.5 «Исследования графов Кэли некоторых групп» содержит результаты вычислительных экспериментов по определению ранее неизвестных характеристик некоторых больших графов Кэли.

Рассмотрим семейство графов $MBS(n)$, каждого представителя которого называют «Modified bubble-sort graph» или в переводе на русский «Модифицированный граф пузырьковой сортировки» [60]. Указанные графы, задаются симметрическими группами S_n, порождаемыми транспозициями следующего вида:

$$S_n = \langle X_n \rangle, \text{ где } X_n = \{(i\ i+1) \mid i = 1, 2, \ldots, n-1\} \cup \{(1\ n)\}\}.$$

Используя способ нумерации элементов группы (1.2), при помощи высокопроизводительной многопроцессорной системы, удалось вычислить ранее неизвестные функции роста групп S_{14} и S_{15}, а также диаметр D и средний диаметр $\overline{D}$ соответствующих графов Кэли: MBS(14) и MBS(15).

Теорема 1.4. *Пусть $S_n = \langle X_n \rangle$. В этом случае, при $n = 14, 15$, верны следующие утверждения:*

(*i*) $D_{X_n}(S_n)$ *и* $\overline{D}_{X_n}(S_n)$ *такие как в таблице* 1.1,

(*ii*) *Функции роста S_n заданы таблицами* 1.2–1.3.

Полученные результаты подтверждают гипотезу из [24], в которой утверждается, что $D = \left\lfloor \frac{n^2}{4} \right\rfloor$ и $\overline{D} = \frac{n^2-n+1}{6}$.

Пусть $B_k = \langle a_1, a_2 \rangle$ – некоторая конечная двупорожденная группа периода 7, где a_1 и a_2 – порождающие элементы группы и $|B_k| = 7^k$. Для каждой из B_k при помощи системы компьютерной алгебры GAP легко получить pc-представление элементов группы (power commutator presentation [17]). В этом случае:

$$\forall g \in g \Longrightarrow g = a_1^{x_1} \ldots a_k^{x_k},\ x_i \in \mathbb{Z}_7.$$

Пусть $a_1^{x_1} \dots a_n^{x_n}$ и $a_1^{y_1} \dots a_n^{y_n}$ – два произвольных элемента в группе B_k, записанные в коммутаторном виде. Тогда их произведение равно

$$a_1^{x_1} \dots a_n^{x_n} \cdot a_1^{y_1} \dots a_n^{y_n} = a_1^{z_1} \dots a_n^{z_n}.$$

Обозначим $X := \{a_1, a_2\}$ – минимальное порождающее множество B_k и $Y := X \cup X^{-1} = \{a_1, a_2, a_1^{-1}, a_2^{-1}\}$ – симметричное порождающее множество.

Для исследования графов Кэли групп B_k относительно порождающих множеств X и Y в алгоритм А–4 был добавлен указанный выше быстрый способ нумерации элементов. В итоге, используя вычисления на суперкомпьютере, удалось получить функции роста групп B_k при $k \leq 14$, а также характеристики соответствующих графов Кэли. Результаты вычислительных экспериментов представлены в виде теорем 1.5 и 1.6:

Теорема 1.5. *Пусть $B_k = \langle X \rangle$ при $k \leq 14$. В этом случае значения диаметров $D_X(B_k)$ и средних диаметров $\overline{D}_X(B_k)$ такие как в таблице* 1.4*, а соответствующие функции роста заданы в таблицах* 1.6–1.18*.*

Теорема 1.6. *Пусть $B_k = \langle Y \rangle$ при $k \leq 14$. В этом случае значения диаметров $D_Y(B_k)$ и средних диаметров $\overline{D}_Y(B_k)$ такие как в таблице* 1.5*, а соответствующие функции роста заданы в таблицах* 1.19–1.31*.*

Перейдём к изложению основного содержания Главы 2 **«Алгоритм расширенного синтаксического анализа языков программирования методом иерархии маркированных скобок».**

В разделе 2.1 «Постановка расширенной проблемы синтаксического анализа с учётом порядка применения продукций» уточняется формулировка проблемы синтаксического анализа, как результат, формулируется расширенная проблема синтаксического анализа мономов кс–языков: *разработать беступиковый алгоритм, который позволяет установить, может ли быть выведен данный моном при помощи системы продукций заданного кс–языка, а также найти сразу все выводы этого монома, указав, какие продукции, сколько раз и в каком порядке применяются для вывода этого монома.*

В разделах 2.2 «Алгоритм решения расширенной проблемы синтаксического анализа с использованием маркированных скобок» и 2.3 «Оценка сложности алгоритма синтаксического анализа на основе иерархии маркированных скобок» рассматривается два способа маркировки скобок, которые можно использовать в алгоритме синтаксического анализа, основанного на иерархии маркированных таким образом скобок. Рассмотрим подробнее один из способов маркировки.

Рассматривается грамматика кс–языка, заданная в виде совокупности правил подстановки (продукций):

$$z_j \to \ q_{j1}(z,x),\ \ldots\ , z_j \to \ q_{jp_j}(z,x),\ j=1,\ \ldots\ , n,$$

где $q_{jk}(z,x)$ — мономом от некоммутативных символов алфавита с числовым коэффициентом, который равен 1.

Кс–язык, порождённый данной грамматикой, является решением системы уравнений Хомского — Шютценберже:

$$z_j = Q_j(z,x) = \ q_{j1}(z,x) + \ \cdots \ + q_{jp_j}(z,x),\ j=1,\ldots,n,$$

и представляется в виде ФСР, который выражает первую компоненту решения $(z_1(x),\ldots,z_n(x))$ этой системы уравнений:

$$z_1 = z_1(x) = \sum_{i=0}^{\infty} \langle z_1, w_i \rangle \ w_i,$$

где w_i — мономы от терминальных символов $x_1,\ldots,x_m$.

Для решения расширенной проблемы синтаксического анализа рассматривается соответствующая расширенная грамматика

$$z_j \to t_{jk}[\ q_{jk}(z,x)],\ j=1,\ldots,n,\ k=1,\ldots,p_j$$

где метка t_{jk} — символ из расширенного алфавита, причём она «привязана» к стоящей справа скобке, помечая правило вывода $z_j \to t_{jk} q_{jk}(z,x)$. Такая расширенная грамматика позволяет определить порядок применения продукций.

Итак, можно маркировать левую (открывающуюся) скобку мономиальной меткой, привязанной к ней слева, считая, что сразу за мономиальной меткой

продукции t_{jk} должна следовать привязанная к ней открывающаяся скобка. Показано (лемма 2.2), что для открывающейся скобки всегда можно найти соответствующую закрывающуюся скобку, а значит, по метке, маркирующей стоящую справа от неё скобку, можно устанавливать иерархию (вложение) скобок.

Иерархию скобок позволяет определить соответствующая расширенная система уравнений Хомского — Шютценберже, которая имеет вид:

$$z_j = Q_j^*(z,x,t) \stackrel{\text{def}}{=} t_{j1}[\ q_{j1}(z,x)\ ] + \ \cdots \ + t_{jp_j}[\ q_{jp_j}(z,x)\ ], \qquad (3)$$

$$j = 1,\ \ldots\ , n.$$

Решение этой системы можно получать методом последовательных приближений. Его итерации, генерирующие начальные члены ФСР решения системы Хомского — Шютценберже, осуществляются по формулам

$$z_j^{(k+1)}(x,t) = Q_j^*(z^{(k)}(x,t),x,t);\ k = 0,1,\ldots;\ z^{(0)}(x,t) = 0,$$

$$j = 1,\ldots,n.$$

Далее начальные члены этого ряда анализируются в соответствии с алгоритмом, описанным в конце раздела 2.2. При этом, «внутренние» скобки соответствуют продукциям, которые использованы позже.

Таким образом, алгоритм синтаксического анализа методом иерархии маркированных скобок можно использовать применительно к расширенной системе уравнений Хомского — Шютценберже.

Для применения метода последовательных приближений необходимо иметь оценку числа итераций, обеспечивающих стабилизацию начальных членов ФСР, которые представляют решение системы уравнений Хомского — Шютценберже.

Такую оценку даёт следующая лемма.

Лемма 2.1. *Пусть $k_2 > k_1$, тогда мономы степени не выше k_1 по переменным x многочленов*

$$z_j^{(k_2)}(x,t),\ j = 1,\ldots,n,$$

не зависят от k_2.

Оценка числа итераций, таким образом, состоит в том, что мономы многочленов $z_j^{(k)}(x,t),\ j=1,\ldots,n$, стабилизированы до степени k включительно и уже не изменятся при дальнейшем росте числа итераций.

Для решения расширенной проблемы синтаксического анализа предложен алгоритм метода иерархии маркированных скобок.

Алгоритм синтаксического анализа, основанный на методе иерархии маркированных скобок, состоит в следующем.

1. Пусть дан моном (программа) w степени (длины) N.

Проводим N итераций метода последовательных приближений для решения соотвествующей расширенной системы уравнений Хомского — Шютценберже.

2. Перебираем все полученные в п. 1 мономы, степень которых равна N, определяя те мономы, которые, с точностью до множителей t_{jk}, совпадают с мономом w, множители t_{jk} при этом пропускаются.

Если таких мономов нет, то моном w вывести невозможно, если такие мономы есть, то они дают решение расширенной проблемы синтаксического анализа в соответствии со следующим п. 3.

3. Считываем все найденные в п. 2 мономы слева направо, устанавливая иерархию маркированных скобок (по признаку сравнения скобок — внутренняя или внешняя скобка) и определяя тем самым порядок применения продукций при выводе монома w, т. е. все возможные способы вывода монома.

На вопрос о работоспособности предложенного алгоритма отвечает следующая теорема.

Теорема 2.1. *Алгоритм синтаксического анализа методом иерархии маркированных скобок, основанный на расширенной системе уравнений Хомского — Шютценберже, позволяет за конечное число шагов осуществить беступиковый синтаксический анализ, с учётом порядка применения продукций, любого монома кс–языка, порождённого кс–грамматикой.*

Далее в разделе 2.3 рассмотрен вопрос о сложности разработанного конструктивного беступикового алгоритма синтаксического анализа, позволяющего провести синтаксический анализ монома w, который имеет степень (длину) N.

Этот вопрос решается леммами 2.3 — 2.6, а также следующей теоремой.

Теорема 2.2. *Сложность алгоритма, основанного на методе иерархии маркированных скобок, использующем расширенную систему уравнений Хомского — Шютценберже, равна*

$$O(Ng^{d^N}).$$

Как видно, сложность данного алгоритма достаточно высока. Следует отметить, что эта сложность равна сложности метода мономиальных меток, предложенного К.В. Сафоновым для решения более ограниченной задачи синтаксического анализа [43].

С другой стороны, алгоритм, основанный на иерархии маркированных скобок, является конструктивным алгоритмом, который весьма прост.

Кроме того, данный алгоритм является беступиковым (безвозвратным, безостановочным) алгоритмом, в отличие от известных ранее алгоритмов.

Наконец, отметим, что предложенный алгоритм особенно эффективен в ситуации, когда степень (длина) исследуемого монома (программы) не слишком большая, а других возможностей провести синтаксический анализ в виде парсеров, например, нет. Такая ситуация, как уже отмечалось, может возникать при разработке языков программирования.

В заключении работы представлены основные результаты и выводы диссертационного исследования.

Далее приведен приведен список используемых литературных источников.

В приложении содержатся копии свидетельств о государственной регистрации программ для ЭВМ.

Глава 1

Алгоритмы на графах Кэли групп подстановок

1.1 Алгоритмы на группах подстановок

Пусть G – конечная группа подстановок на множестве точек $\Omega = \{1, 2, \ldots, n\}$.

Обозначим $\alpha^g := g[\alpha]$ образ элемента $\alpha \in \Omega$ под действием $g \in G$.

Орбитой точки $\alpha \in \Omega$ называется множество $\alpha^G := \{\alpha^g \mid g \in G\}$.

Стабилизатором точки $\alpha \in \Omega$ будем называть множество $G_\alpha := \{g \in G \mid \alpha^g = \alpha\}$. Для заданных $\beta_1, \beta_2, \ldots, \beta_i \in \Omega$ индуктивно определим

$$G_{\beta_1,\beta_2,\ldots,\beta_i} := (G_{\beta_1,\beta_2,\ldots,\beta_{i-1}})_{\beta_i} = \{g \in G \mid \beta_j^g = \beta_j,\ j = 0, 1, \ldots, i\}.$$

Последовательность различных элементов $B := (\beta_1, \beta_2, \ldots, \beta_m)$ будем называть базой группы G, если $G_{\beta_1,\beta_2,\ldots,\beta_m} = e$. Таким, образом только единичный элемент группы оставляет неподвижными все точки базы.

Пусть $G^{(i)} := G_{\beta_1,\beta_2,\ldots,\beta_{i-1}}$. Далее определим цепь стабилизаторов

$$G := G^{(1)} \geq G^{(2)} \geq \cdots \geq G^{(m)} \geq G^{(m+1)} = e.$$

Если $G^{(i+1)}$ является собственной подгруппой $G^{(i)}$ для $i \in [1, m]$, то базу B называют несократимой.

Смежные классы группы $G^{(i)}$ по подгруппе $G^{(i+1)}$ имеют взаимно однозначное соответствие с элементами орбиты $\Delta^{(i)} := \beta_i^{G^{(i)}}$. Действительно, если

a, $b \in G^{(i)}$ и $G^{(i+1)}a = G^{(i+1)}b$, то для некоторого $h \in G^{(i+1)}$ будет выполняться $a = hb$. Поэтому, $\beta_i^a = \beta_i^{hb} = \beta_i^b$.

Указанный выше факт дает возможность вычислить семейство представителей смежных классов (трансверсаль) $U^{(i)}$ группы $G^{(i)} \mod G^{(i+1)}$. Для каждого $\gamma \in \Delta^{(i)}$ определим $u_i(\gamma) \in G^{(i)}$, который отображает β_i в γ, т. е. $\beta_i^{u_i(\gamma)} = \gamma$. В частном случае, если $\beta_i \to \beta_i$, то $u_i(\beta_i) := e$.

Согласно введенным обозначениям мы получим упорядоченную последовательность $U^{(i)} := (u_i(\gamma) \mid \gamma \in \Delta^{(i)})$. Очевидно, что $|U^{(i)}| = |\Delta^{(i)}|$.

Объединив все $U^{(i)}$, мы получим полное семейство представителей смежных классов группы:

$$U = \bigcup_{i=1}^{m} U^{(i)}.$$

По теореме Лагранжа $|G| = |G^{(1)} : G^{(2)}| \cdot |G^{(2)}| = |U^{(1)}| \cdot |G^{(2)}|$. Аналогично, $|G^{(2)}| = |G^{(2)} : G^{(3)}| \cdot |G^{(3)}| = |U^{(2)}| \cdot |G^{(3)}|$. Продолжив данный процесс, мы получим

$$|G| = |U^{(1)}| \cdot |U^{(2)}| \cdots |U^{(m)}|.$$

Пусть $B = (\beta_1, \beta_2, \ldots, \beta_m)$ – база G, тогда для любого элемента группы мы можем определить его базовый образ $B^g := (\beta_1^g, \beta_2^g, \ldots, \beta_m^g)$.

Лемма 1.1 *Для $\forall g \in G$ базовый образ B^g имеет уникальное представление.*

Доказательство. Пусть $B^g = B^h$, тогда $B^{gh^{-1}} = B$. Поэтому, согласно определению базы, $gh^{-1} = e$. Следовательно, $g = h$. ■

Лемма 1.2 *Любой элемент $g \in G$ может однозначно быть записан в каноническом виде*

$$g := (u_m, u_{m-1}, \ldots, u_1) = u_m u_{m-1} \ldots u_1, \text{ где } u_i \in U^{(i)} \ u \ i \in [1, m]. \qquad (1.1)$$

Доказательство. Так как $g \in G$, то он содержится в некотором смежном классе группы G по подгруппе $G^{(2)}$, поэтому запишем его в виде $g = h_2 u_1$,

где $h_2 \in G^{(2)}$ и $u_1 \in U^{(1)}$. Аналогично, $h_2 = h_3 u_2$, где $h_2 \in G^{(2)}$ и $u_2 \in U^{(2)}$. Продолжая данный процесс, получим искомую формулу. ■

В дальнейшем нам также понадобится множество обратных элементов полного семейства представителей смежных классов

$$\hat{U} = \bigcup_{i=1}^{m} \hat{U}^{(i)}, \text{ где } \hat{U}^{(i)} := (u_i^{-1}(\gamma) \mid \gamma \in \Delta^{(i)}).$$

Для быстрого поиска элементов в $U^{(i)}$ и $\hat{U}^{(i)}$ нам понадобится вспомогательный массив индексов $A_{m \times n}$, элементы которого определяются следующим образом:

$$A_{ij} := \begin{cases} a_{ij} \in [0, |U^{(i)}| - 1] \text{ – номер элемента } u_i(j) \text{ в } U^{(i)}, \text{ если } j \in \Delta^{(i)}; \\ -1 \text{ в противном случае.} \end{cases}$$

Ниже представлен алгоритм, который получив на входе слово, состоящее из произведения элементов $g_1 g_2 \dots g_l$ группы G, ее базу B, полное семейство представителей смежных классов U и его инверсию $\hat{U}$, а также вспомогательный массив A, возвращает каноническое представление $u_m u_{m-1} \dots u_1$ данного произведения.

Алгоритм А–1. $u_m u_{m-1} \dots u_1 =$ `Factor`$(g_1 g_2 \dots g_l, U, \hat{U}, A, B)$

Вход: $g_1 g_2 \dots g_l$, $\hat{U}$, A и B.

Выход: $u_m u_{m-1} \dots u_1$

1: $h := B$
2: **Для всех** $i = 1, 2, \dots, l$
3: $\quad h := (h_1^{g_i}, h_2^{g_i}, \dots, h_m^{g_i})$
4: **Для всех** $i = 1, 2, \dots, m$
5: $\quad j := h[\,i\,]$
6: $\quad u_i := U^{(i)}[a_{ij}]$
7: $\quad u_i^{-1} := \hat{U}^{(i)}[a_{ij}]$
8: $\quad h := (h_1^{u_i^{-1}}, h_2^{u_i^{-1}}, \dots, h_m^{u_i^{-1}})$
9: **Вернуть** $u_m u_{m-1} \dots u_1$

Пусть $T_i = O(f)$ и $M_i = O(f)$ — верхние асимптотические оценки вычислительной и пространственной сложности i–го алгоритма соответсвенно.

Лемма 1.3 *Алгоритм А–1 корректен и* $T_1 = O(l \cdot m + m^2)$.

Доказательство. Корректность алгоритма следует из лемм 1.1–1.2. Сначала мы получаем базовый образ элемента $g := g_1 g_2 \dots g_l$, а затем при каждом $i \in [1, m]$ стабилизируем точку $\beta_i \in B$. В итоге мы получим $B^{g u_1^{-1} \dots u_m^{-1}} = B$. Следовательно, $g = u_m u_{m-1} \dots u_1$.

Для оценки вычислительной сложности обратим внимание на то, что число операций в цикле 2–3 ограничено $O(l \cdot m)$, а цикла 4–8 – $O(m^2)$. Поэтому, $T_1 = O(l \cdot m + m^2)$. ■

Разложение (1.1) дает возможность эффективно нумеровать все $g \in G$, используя метод перечисления элементов кортежа в смешанной системе счисления [21]. Пусть $c := (c_m, c_{m-1}, \dots, c_1)$ – основание смешанной системы счисления, в которой $c_1 := 1$ и $c_i = c_{i-1} \cdot |U^{(i-1)}|$ для $i \geq 2$.

Пусть $g := u_m u_{m-1} \dots u_1$. Определим биективное отображение

$$\mathcal{N}\colon u_m u_{m-1} \dots u_1 \longrightarrow (a_m, a_{m-1}, \dots, a_1), \tag{1.2}$$

где $a_i \in [0, |U^{(i)}| - 1]$ – номер элемента u_i в $U^{(i)}$.

Заметим, что вектор $(a_m, \dots, a_1)$ представляет собой число $\mathcal{N}(g) \in [0, |G| - 1]$ в системе счисления со смешанным основанием $(c_m, \dots, c_1)$.

Пусть $k := \mathcal{N}(g)$ – номер элемента g.

Лемма 1.4 *Вычислительная сложность* $\mathcal{N}(g)$ *и* $\mathcal{N}^{-1}(k)$ *не превышает* $O(m)$.

Доказательство. Пусть $g = u_m u_{m-1} \dots u_1$ и a_i – номер u_i в $U^{(i)}$. Тогда, $k = \mathcal{N}(g) = a_m c_m + \dots + a_1 c_1$.

В обратном случае $g = \mathcal{N}^{-1}(k) = u_m u_{m-1} \dots u_1$, где факторы u_i вычисляются следующим образом:

Для всех $i = 1, 2, \dots, m$

$a_i := k \mod |U^{(i)}|$

$k := \lfloor k \,/\, |U^{(i)}| \rfloor$

$u_i := U^{(i)}[a_i]$.

Очевидно, что число операций в данных процедурах не превышает $O(m)$.

■

Замечание 1.1 *Для определенности, будем полагать, что все последовательности* $U^{(i)}$ *начинаются с единичного элемента* e. *В этом случае* $\mathcal{N}(e) := 0$.

Отметим, что для нумерации элементов группы нам необходимо знать ее базу B и полное семейство представителей смежных классов U. Для их вычисления мы будем использовать известный алгоритм Шрайера-Симса, предложенный Ч. Симсом в 1970 году [81]. В настоящее время существует множество его модификаций [76]. Наиболее эффективные версии алгоритма имеют низкую вычислительную сложность и реализованы в таких системах компьютерной алгебры, как GAP, Magma и Mathematica, а также в библиотеке SymPy для языка Python.

1.2 Алгоритмы маршрутизации на графах Кэли

Для поиска кратчайших путей на графе Кэли $Cay(G, X)$ нам понадобится таблица маршрутизации $P_{2\times|G|}$, которую также называют родительским деревом [45]. Далее представлен алгоритм, который, получив на входе порождающее множество группы X, ее базу B, полное семейство представителей смежных классов U и его инверсию $\hat{U}$, а также вспомогательный массив A, возвращает указанную таблицу.

Алгоритм А–2. $P =$ `BFS`$(X, U, \hat{U}A, B)$

Вход: X, U, $\hat{U}$, A и B

Выход: Таблица маршрутизации P на графе Кэли $Cay(G, X)$

1: $P_{2\times|G|} := \begin{bmatrix}\infty & \dots & \infty\end{bmatrix}$
2: $k := \mathcal{N}(e) := 0$
3: $P[\,1\,][\,k\,] := -1$
4: $P[\,2\,][\,k\,] := -1$
5: $Q := \{k\}$
6: **Пока** $Q \neq \emptyset$
7: извлечь $q \in Q$ из очереди
8: **Для всех** $x \in X$
9: $s :=$ `Factor` $(\mathcal{N}^{-1}(q)\, x, U, \hat{U}, A, B)$
10: $k := \mathcal{N}(s)$
11: **Если** $P[\,1\,][\,k\,] = \infty$, **то**
12: добавить k в очередь Q
13: $P[\,1\,][\,k\,] := q$
14: $P[\,2\,][\,k\,] := x$
15: **Вернуть** P

Теорема 1.1 *Алгоритм А–2 корректен и* $T_2 = O(m^2 \cdot |X| \cdot |G|)$.

Доказательство. Данный алгоритм представляет собой классический метод поиска в ширину на графе [45]. В этом случае вершина с номером единичного элемента e (согласно замечанию 1.1 имеем $\mathcal{N}(e) := 0$) будет являться корнем родительского дерева P. Пусть $gx = h$, где $\mathcal{N}(g) = k$ и $\mathcal{N}(h) = l$, тогда

$P[1][l] := k$, $P[2][l] = x$. Это означает, что вершина k является родителем вершины l и x – метка ребра (k, l).

Нам необходимо проверить всего $|X| \cdot |G|$ элементов. Время выполнения проверки каждого элемента согласно леммам 1.3 и 1.4 ограничено $O(m^2)$. Следовательно, $T_2 = O(m^2 \cdot |X| \cdot |G|)$. ■

Следующий алгоритм при помощи таблицы маршрутизации P вычисляет кратчайший путь $w := x_1 x_2 \dots x_s$ между вершинами $a, b \in Cay(G, X)$, где $x_i \in X$. Очевидно, что $s \leq D$, где D – диаметр графа.

Алгоритм А–3. $w =$ `Route`$(a, b, U, \hat{U}A, B, P)$

Вход: a, b, U, $\hat{U}$, A, B и P

Выход: $w := x_1 x_2 \dots x_s$ – кратчайший маршрут из вершины a в вершину b

1: $w :=$ [ ] – пустое слово
2: $g_1 := \mathcal{N}^{-1}(a)$
3: $g_2 := \mathcal{N}^{-1}(b)$
4: $g :=$ `Factor` $(g_1^{-1} g_2, U, \hat{U}, A, B)$
5: $k := \mathcal{N}(g)$
6: $l := k$
7: **Пока** $P[1][k] \neq -1$
8: $\quad k := P[1][l]$
9: $\quad w := P[2][l] \oplus w$ – конкатенация строк
10: $\quad l := k$
11: **Вернуть** w

Теорема 1.2 *Алгоритм А–3 корректен,* $T_3 = O(m^2 + D)$ *и* $M_3 = O(m \cdot n + |G|)$.

Доказательство. В графе Кэли кратчайший путь от вершины a к вершине b будет представлять собой минимальное слово $w := x_1 x_2 \dots x_s$ элемента $g := g_1^{-1} g_2$, где $g_1 := \mathcal{N}^{-1}(a)$, $g_2 := \mathcal{N}^{-1}(b)$ и $x_i \in X$ [61]. Пусть k – номер вершины элемента g. Двигаясь по родительскому дереву P от вершины k к его корню, мы получим искомый путь.

Время выполнения пунктов 1–6 ограничено $O(m^2)$, а цикла 7–10 – $O(D)$. В итоге получим $T_3 = O(m^2 + D)$.

Пространственная сложность переменных U, $\hat{U}$, A и B ограничена $O(m \cdot n)$, а таблицы P – $O(|G|)$. Поэтому, $M_3 = O(m \cdot n + |G|)$. ■

Замечание 1.2 *Во многих прикладных задачах при исследовании графа Кэли* $\Gamma := Cay(G, X)$ *порядок порождающей группы* G *значительно превышает ее степень [60, 78, 83, 84, 62, 51, 54], т. е.* $m \leq n \ll |G|$. *В этом случае* $M_3 = O(|G|)$.

Пример 1

Рассмотрим группу $G = \langle X \rangle$, порожденную двумя циклами $x = (1, 5, 4)$ и $y = (3, 4)$. Вычислим в GAP базу G и полное семейство представителей смежных классов:

$$B = (1, 3, 4),$$

$$U^{(1)} = (e, (1, 4)(3, 5), (1, 5)(3, 4), (1, 3)(4, 5)),\ \Delta^{(1)} = (1, 4, 5, 3),$$

$$U^{(2)} = (e, (3, 4, 5), (3, 5, 4)),\ \Delta^{(2)} = (3, 4, 5),$$

$$U^{(3)} = (e, (4, 5)),\ \Delta^{(3)} = (4, 5).$$

При помощи Алгоритма 2 получим граф Кэли $\Gamma = Cay(G, X)$ и его родительское дерево P (см. рис. 1).

Для иллюстрации Алгоритма 3 найдем расстояние от вершины $a := 15$ до $b := 22$.

$$g_1 := \aleph^{-1}(a) = U^{(3)}[\,1\,]\,U^{(2)}[\,0\,]\,U^{(1)}[\,3\,],$$

$$g_2 := \aleph^{-1}(b) = U^{(3)}[\,1\,]\,U^{(2)}[\,2\,]\,U^{(1)}[\,2\,],$$

$$g := \texttt{Factor}(g_1^{-1} g_2) = U^{(3)}[\,0\,]\,U^{(2)}[\,2\,]\,U^{(1)}[\,0\,],$$

$$k := \aleph(g) = 8.$$

Двигаясь по родительскому дереву P от вершины с номером 8 к его корню, получим

$$8 \xrightarrow{y} 20 \xrightarrow{x} 17 \xrightarrow{x} 14 \xrightarrow{y} 6 \xrightarrow{x} 0.$$

Следовательно, $w := xyxxy$. На графе Γ маршрут будет иметь следующий вид:

$$15 \xrightarrow{x} 19 \xrightarrow{y} 1 \xrightarrow{x} 4 \xrightarrow{x} 10 \xrightarrow{y} 22.$$

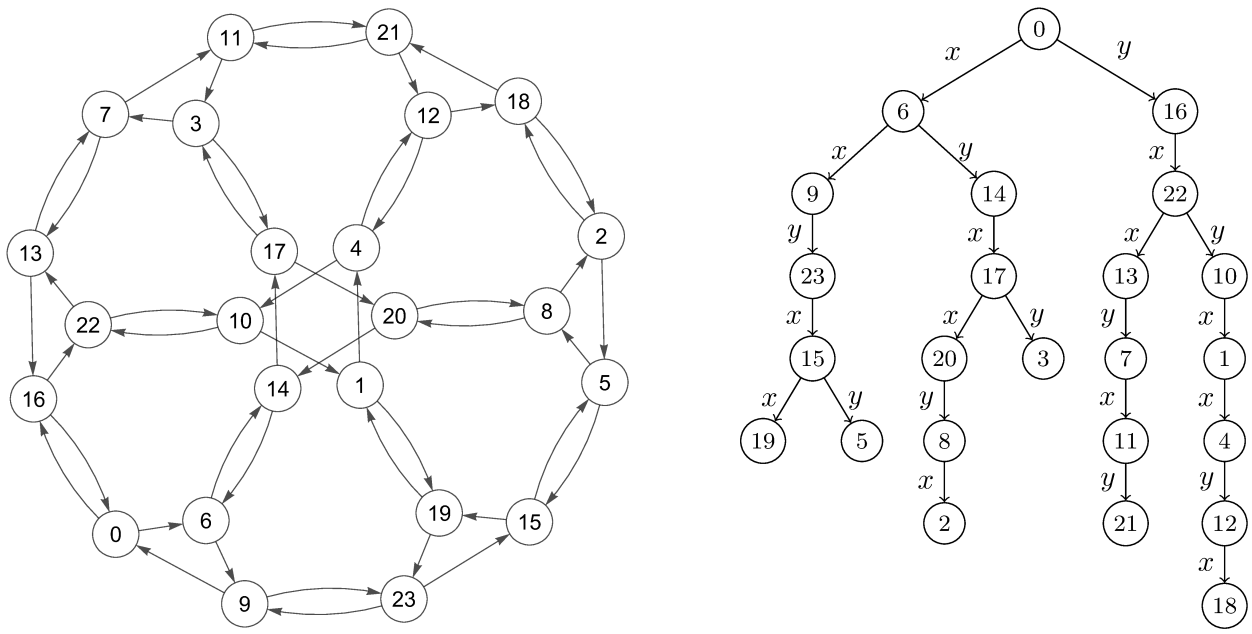


Рис. 1.1: Граф Кэли $\Gamma = Cay(G, X)$ и его родительское дерево P. Ребра графа Γ с меткой x представлены в виде прямой линии, а y в форме дуги

1.3 Проблема минимального слова

Небольшая модификация алгоритма A–3 позволяет вычислить минимальное слово w из произвольной строки $v := x_1 x_2 \dots x_r$ в алфавите порождающих X.

Алгоритм A–4. $w =$ `MinimalWord`$(v, U, \hat{U}A, B, P)$

Вход: v, U, $\hat{U}$, A, B и P

Выход: $w := x_1 x_2 \dots x_s$ – минимальное слово

1: $w :=$ [ ] – пустое слово
2: $g :=$ `Factor` $(v, U, \hat{U}, A, B)$
3: $k := \mathcal{N}(g)$
4: $l := k$
5: **Пока** $P[1][k] \neq -1$
6: $\quad k := P[1][l]$
7: $\quad w := P[2][l] \oplus w$ – конкатенация строк
8: $\quad l := k$
9: **Вернуть** w

Теорема 1.3 *$T_4 = O(|v|)$ и $M_4 = O(m \cdot n + |G|)$.*

Согласно леммам 1.3 и 1.4 время выполнения пунктов 1–4 ограничено $O(|v| \cdot m + m^2)$. Пункты 5–9 алгоритма представляют собой восходящее движение

по родительскому дереву P, поэтому сложность этого участка не превышает $O(D)$. Таким образом, мы получим, $T_4 = O(|v| \cdot m + m^2 + D)$. Если длина слова велика, то $T_4 = O(|v|)$.

Пространственная сложность переменных U, $\hat{U}$, A и B ограничена $O(m \cdot n)$, а таблицы P – $O(|G|)$. Поэтому, $M_4 = O(m \cdot n + |G|)$. ■

1.4 Сравнительный анализ алгоритмов

В следующей таблице приведены оценки временной и пространственной сложности алгоритмов A–0 и A–4.

Алгоритм	T_i	M_i
A–0	$O(\|v\|^2)$	$O(\|X\| \cdot \|G\| + \|\mathcal{A}\|)$
A–4	$O(\|v\|)$	$O(m \cdot n + \|G\|)$

Из таблицы видно, что алгоритм A–4 имеет более низкую вычислительную сложность в сравнении с алгоритмом A–0.

Что касается пространственной сложности, то согласно замечанию 1.2, для многих интересных топологий будет выполняться $M_0 \sim M_4 \sim |G|$.

A–4 был реализован автором на языке C++, в свою очередь A–0 написан на языке C и является частью свободно распространяемого пакета KBMAG . Для того, чтобы обеспечить чистоту эксперимента, данные алгоритмы транслировались компилятором gcc с одинаковыми параметрами. На начальном этапе для тестирования были взяты две группы.

а) Симметрическая группа $S_9 = \langle Y \rangle$, где $Y = \{(i, i+1) \mid i \in [1, 8]\}$. Граф Кэли $Cay(S_9, Y)$ называют также графом пузырьковой сортировки [60]. Хорошо известно, что $S_n = n!$, $m = n - 1$ и $D_Y(S_n) = \frac{n(n-1)}{2}$.

б) Спорадическая простая группа Мэтьё $M_{22} = \langle X \rangle$, где $X = \{x_1, x_2, x_2^{-1}\}$ и

$x_1 = (1, 13)(2, 8)(3, 16)(4, 12)(6, 22)(7, 17)(9, 10)(11, 14)$;

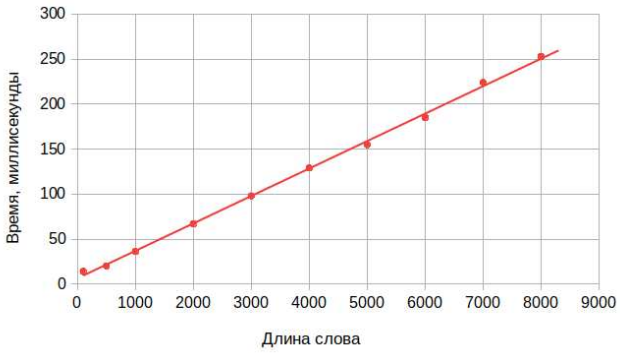
$x_2 = (1, 22, 3, 21)(2, 18, 4, 13)(5, 12)(6, 11, 7, 15)(8, 14, 20, 10)(17, 19)$.

При этом $|M_{22}| = 443520$, $m = 5$ и $D_X(M_{22}) = 34$.

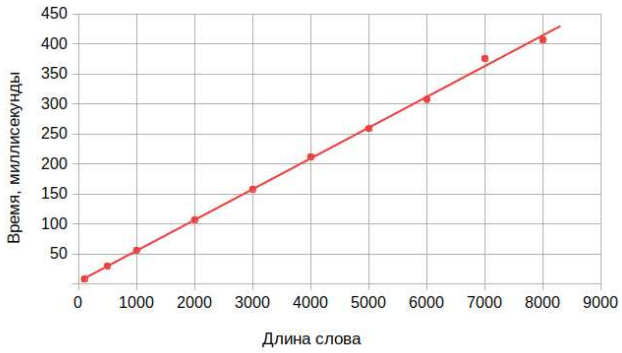
На рис. 1.2 приведены графики зависимости времени $T_4(l)$ выполнения алгоритма A–4 для групп $S_9 = \langle Y \rangle$ и $M_{22} = \langle X \rangle$ в зависимости от длины входящего слова.

На рис. 1.3 приведены графики зависимости времени $T_0(l)$ и $T_4(l)$ от длины входящего слова для указанных групп.

Эти графики наглядно показывают, что алгоритм A–4 значительно быстрее, чем A–0.

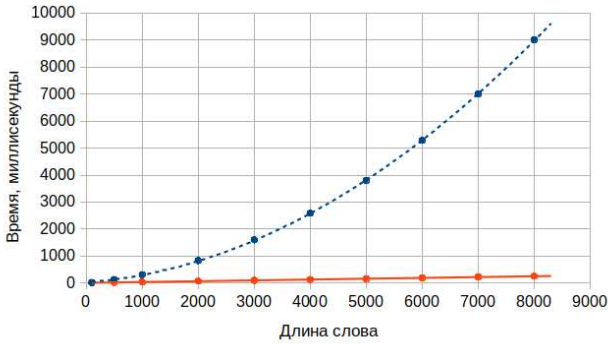


S_9

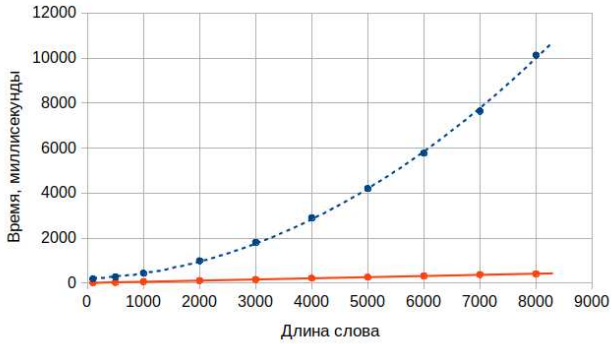


M_{22}

Рис. 1.2: Графики $T_4(l)$ для групп $S_9 = \langle Y \rangle$ и $M_{22} = \langle X \rangle$.



S_9



M_{22}

Рис. 1.3: Графики $T_0(l)$ (пунктир) и $T_4(l)$ для групп $S_9 = \langle Y \rangle$ и $M_{22} = \langle X \rangle$.

Представленные в данной главе алгоритмы послужит отправной точкой для создания новых ресурсно-эффективных алгоритмов маршрутизации. Здесь можно выделить два направления. Во-первых, создание алгоритмов, учитывающих топологию сети. В этом случае алгоритмы будут проектироваться для конкретных классов графов Кэли. Во-вторых, разработка гибридных алгоритмов, включающих в себя как статическую, так и динамическую таблицы маршрутизации. Это позволит рассчитывать оптимальные маршруты в зависимости от текущего состояния сети.

1.5 Исследования графов Кэли некоторых групп

Напомним основные определения [28]. Пусть $G = \langle X \rangle$. *Шаром* K_s радиуса s группы G будем называть множество всех её элементов, которые могут быть представлены в алфавите X в виде несократимых групповых слов длины не больше s. Все элементы одинаковой длины i образуют *сферу* P_i радиуса i. Единица группы e является пустым словом, длина которого равна нулю. Согласно данным определениям, $K_s = \bigcup_{i=0}^{s} P_i$.

Для каждого целого неотрицательного i можно определить (сферическую) функцию роста группы $F(G)$, которую мы будем записывать в виде вектора: $F(G) = (F_0, F_1, \ldots, F_i, \ldots)$, где $F_i = |P_i|$. Пусть $F_{s_0} > 0$, но $F_{s_0+1} = 0$, тогда s_0 является диаметром графа Кэли группы G в алфавите порождающих X, который будем обозначать $D_X(G)$. Средний диаметр $\overline{D}_X(G)$ равен $\frac{1}{|G|} \sum_{s=0}^{s_0} s \cdot F_s$.

Алгоритм А–5. $K_s =$ `Ball`(G, X, s)

Вход: X — порождающее множество группы G, радиус s

Выход: шар K_s группы G радиуса s

1: $K_s := \bigcup_{i=0}^{s} P_i$, где $P_i := \emptyset$ — сферы радиуса i
2: $P_0 := \{e\}$
3: **Для всех** $i = 1, 2, \ldots, s$
4: **Для всех** $x \in X$ и $p \in P_{i-1}$
5: $g := x \cdot p$
6: **Если** $g \notin K_s$, **то**
7: добавить g в $P_i \subset K_s$
8: **Если** $|P_i| = 0$, **то**
9: переход в п. 10
10: **Вернуть** K_s

По построению алгоритм А–5 выражает каждый элемент группы G в виде группового слова наименьшей длины в алфавите X. После каждого прохода от п. 3 до п. 7 множество K_s будет представлять собой шар радиуса i группы G относительно X. Конечность G гарантирует остановку алгоритма при некотором $i \leq s$. Согласно [28, 25] имеем $T_4 = \Theta(|K_s|^2)$ и $M_4 = \Theta(|K_s|)$, где T_4 и M_4

– вычислительная и пространственная сложность алгоритма A–5.

Если мы располагаем быстрым способом нумерации элементов K_s (например, как в [27], или [95]), то тогда K_s можно представить в виде булева вектора размерности $|G|$, в котором на i-ом месте будет стоять единица, если элемент с указанным номером лежит в K_s, и ноль в противном случае. Теперь для хранения элемента достаточно одного бита. Кроме того, по номеру элемента легко осуществить проверку: встречался ли ранее данный элемент группы или нет? Сложность данной операции равна $O(1)$. В этом случае получим $T_4 = \Theta(|K_s|)$.

1.5.1 Графы MBS(n)

Рассмотрим семейство графов MBS(n), каждого представителя которого называют «Modified bubble-sort graph» или в переводе на русский «Модифицированный граф пузырьковой сортировки» [60]. Указанные графы, задаются симметрическими группами S_n, порождаемыми транспозициями следующего вида:

$$S_n = \langle X_n \rangle, \text{ где } X_n = \{(i\ i+1) \mid i = 1, 2, \ldots, n-1\} \cup \{(1\ n)\}\}.$$

Используя способ нумерации элементов группы (1.2), при помощи высокопроизводительной многопроцессорной системы, удалось вычислить ранее неизвестные функции роста групп S_{14} и S_{15}, а также диаметр и средний диаметр соответствующих графов Кэли: MBS(14) и MBS(15).

Теорема 1.4 *Пусть $S_n = \langle X_n \rangle$. В этом случае, при $n = 14, 15$, верны следующие утверждения:*

(*i*) *$D_{X_n}(S_n)$ и $\overline{D}_{X_n}(S_n)$ такие как в таблице* 1.1,

(*ii*) *Функции роста S_n заданы таблицами* 1.2–1.3.

Доказательство. Непосредственно вычисления по алгоритму A–5. ■

Полученные результаты подтверждают гипотезу из [24], в которой утверждается, что $= \left\lfloor \frac{n^2}{4} \right\rfloor$ и $\overline{D} = \frac{n^2-n+1}{6}$.

Таблица 1.1: Характеристики графов Кэли групп $S_n = \langle X_n \rangle$.

n	14	15
D	49	56
$\overline{D}$	$61/2$	$211/6$

Таблица 1.2: Функция роста $S_{14} = \langle X_{14} \rangle$.

s	$F(s)$	s	$F(s)$	s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	10	1144066	20	523576508	30	7875671024	40	439630193
1	14	11	2496144	21	807898884	31	8184285564	41	177894717
2	105	12	5200300	22	1206399194	32	8059876006	42	59073300
3	560	13	10399573	23	1741769068	33	7486458696	43	15497794
4	2380	14	20044817	24	2428791549	34	6521589932	44	3095274
5	8568	15	37346764	25	3266926299	35	5291298298	45	457459
6	27132	16	67385500	26	4232447674	36	3964966720	46	46501
7	77520	17	117857285	27	5272211438	37	2715665810	47	2912
8	203490	18	199872075	28	6301765938	38	1678335828	48	93
9	497420	19	328616470	29	7210522410	39	920955932	49	1
$\lvert S_{14} \rvert = 14! = 87178291200$									

1.5.2 Графы Кэли конечных групп периода 7

Пусть $B_k = \langle a_1, a_2 \rangle$ – некоторая конечная двупорожденная бернсайдова группа периода 7, где a_1 и a_2 – порождающие элементы группы и $|B_k| = 7^k$. Для каждой из B_k при помощи системы компьютерной алгебры GAP легко получить pc-представление элементов группы (power commutator presentation [17]). В этом случае:

$$\forall g \in g \Longrightarrow g = a_1^{x_1} \dots a_k^{x_k},\ x_i \in \mathbb{Z}_7.$$

Пусть $a_1^{x_1} \dots a_n^{x_n}$ и $a_1^{y_1} \dots a_n^{y_n}$ – два произвольных элемента в группе B_k, записанные в коммутаторном виде. Тогда их произведение равно

$$a_1^{x_1} \dots a_n^{x_n} \cdot a_1^{y_1} \dots a_n^{y_n} = a_1^{z_1} \dots a_n^{z_n}.$$

Основой для нахождения коэффициентов z_i является собирательный процесс (см. [82, 61]), который реализован в системах компьютерной алгебры GAP и MAGMA. Кроме того, существует альтернативный способ для вычисления

Таблица 1.3: Функция роста $S_{15} = \langle X_{15} \rangle$.

s	$F(s)$	s	$F(s)$	s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	12	9657700	24	8330511890	36	111514773216	48	723446997
1	15	13	20058300	25	12267072742	37	108665304904	49	217500207
2	120	14	40115312	26	17570631028	38	100844738930	50	52359036
3	680	15	77540544	27	24463138336	39	88757487936	51	9748102
4	3060	16	145284325	28	33079985068	40	73721997284	52	1343342
5	11628	17	264439921	29	43405123242	41	57447140400	53	126700
6	38760	18	468283120	30	55204143800	42	41700857614	54	7282
7	116280	19	807550010	31	67970322498	43	27958028594	55	210
8	319770	20	1356808058	32	80902850743	44	17132134736	56	2
9	817190	21	2221351570	33	92936928305	45	9472716396		
10	1961256	22	3543440716	34	102839910030	46	4651580804		
11	4457400	23	5505931806	35	109374995290	47	1989274794		
$\lvert S_{15} \rvert = 15! = 1307674368000$									

произведений элементов группы, предложенный Ф. Холлом (см. [59]). Холл показал, что z_i представляют собой полиномиальные функции (в нашем случае над полем $\mathbb{Z}_7$), зависящие от переменных $x_1, \ldots, x_i, y_1, \ldots, y_i$, которые принято сейчас называть *полиномами Холла*. Согласно [59]:

$$z_i = x_i + y_i + p_i(x_1, \ldots, x_{i-1}, y_1, \ldots, y_{i-1}).$$

В работе [63] были вычислены полиномы Холла групп B_k, которые позволяют осуществлять произведение элементов группы значительно быстрее собирательного процесса:

$z_1 = x_1 + y_1,$

$z_2 = x_2 + y_2,$

$z_3 = x_3 + y_3 + x_2y_1,$

$z_4 = x_4 + y_4 + 3x_2y_1 + x_3y_1 + 4x_2y_1^2,$

$z_5 = x_5 + y_5 + 3x_2y_1 + x_3y_2 + 4x_2^2y_1 + x_2y_1y_2,$

$z_6 = x_6 + y_6 + 5x_2y_1 + 3x_3y_1 + x_4y_1 + 3x_2y_1^2 + 6x_2y_1^3 + 4x_3y_1^2,$

$z_7 = x_7 + y_7 + 2x_2^2y_1^2 + 2x_2y_1 + x_4y_2 + x_5y_1 + 5x_2y_1^2 + 5x_2^2y_1 + 4x_2y_1^2y_2 + 3x_2y_1y_2 + x_3y_1y_2,$

$z_8 = x_8 + y_8 + 5x_2y_1 + 3x_3y_2 + x_5y_2 + 3x_2^2y_1 + 6x_2^3y_1 + 4x_3y_2^2 + 4x_2y_1y_2^2 +$

$4x_2^2y_1y_2 + 6x_2y_1y_2$,

$z_9 = x_9 + y_9 + 5x_2y_1 + 5x_3y_1 + 3x_4y_1 + x_6y_1 + 6x_2y_1^2 + 5x_2y_1^3 + 3x_3y_1^2 + 5x_2y_1^4 + 6x_3y_1^3 + 4x_4y_1^2$,

$z_{10} = x_{10} + y_{10} + x_2^2y_1^3 + 3x_2y_1 + 4x_3y_1 + 6x_5y_1 + 6x_4y_3 + x_6y_2 + 4x_2y_1^2 + 6x_2^2y_1 + 3x_3^2y_1 + 3x_2x_3y_1^2 + 3x_2y_1^2y_2 + 3x_2y_1^2y_3 + 6x_2y_1^3y_2 + 4x_3y_1^2y_2 + 4x_2x_3y_1 + 5x_2y_1y_2 + 4x_2y_1y_3 + 3x_3y_1y_2 + 6x_3y_1y_3 + x_4y_1y_2$,

$z_{11} = x_{11} + y_{11} + 5x_2^2y_1^2 + 2x_2^2y_1^3 + 5x_2y_1 + 3x_3y_1 + 4x_5y_1 + x_4y_3 + x_7y_1 + 5x_2y_1^2 + 4x_2y_1^3 + 4x_3^2y_1 + 4x_5y_1^2 + 4x_2x_3y_1^2 + 4x_2y_1^2y_3 + 3x_2x_3y_1 + 3x_2y_1y_3 + x_3y_1y_3$,

$z_{12} = x_{12} + y_{12} + 5x_2^3y_1^2 + 6x_2y_1 + 4x_3y_2 + 3x_4y_2 + 6x_5y_1 + 6x_5y_3 + x_7y_2 + 2x_2y_1^2 + 4x_2^2y_1 + 4x_2^3y_1 + 3x_3^2y_2 + 4x_4y_2^2 + 3x_2^2x_3y_1 + 5x_2y_1y_2^2 + 3x_2y_1^2y_2 + 5x_2^2y_1y_2 + 4x_3y_1y_2^2 + 3x_2^2y_1y_3 + 2x_2y_1^2y_2^2 + 5x_2^2y_1^2y_2 + 4x_2x_3y_1 + x_2y_1y_2 + 4x_2y_1y_3 + 3x_3y_1y_2 + 6x_3y_2y_3 + x_5y_1y_2 + 6x_2x_3y_1y_2 + 6x_2y_1y_2y_3$,

$z_{13} = x_{13} + y_{13} + 5x_2^2y_1^2 + 5x_2^3y_1^2 + 2x_2y_1 + 3x_3y_2 + x_5y_1 + x_5y_3 + x_8y_1 + 4x_2y_1^2 + 5x_2^2y_1 + 4x_3^2y_2 + 4x_2^2x_3y_1 + 4x_2^2y_1y_3 + 4x_2^2y_1^2y_2 + 3x_2x_3y_1 + 3x_2y_1y_2 + 3x_2y_1y_3 + x_3y_2y_3 + x_2x_3y_1y_2 + x_2y_1y_2y_3$,

$z_{14} = x_{14} + y_{14} + 5x_2y_1 + 5x_3y_2 + 3x_5y_2 + x_8y_2 + 6x_2^2y_1 + 5x_2^3y_1 + 3x_3y_2^2 + 5x_2^4y_1 + 6x_3y_2^3 + 4x_5y_2^2 + x_2y_1y_2^2 + x_2^2y_1y_2 + 6x_2y_1y_2^3 + 6x_2^3y_1y_2 + 2x_2^2y_1y_2^2 + 5x_2y_1y_2$,

На их основе вычислим важные частные случаи полиномов, необходимые для дальнейшего построения графов Кэли группы B_{14} и ее факторов.

1) $a_1 \cdot a_1^{y_1}a_2^{y_2}\dots a_{14}^{y_{14}} = a_1^{y_1+1}a_2^{y_2}\dots a_{14}^{y_{14}}$.

2) $a_1^{-1} \cdot a_1^{y_1}a_2^{y_2}\dots a_{14}^{y_{14}} = a_1^{y_1+6}a_2^{y_2}\dots a_{14}^{y_{14}}$.

3) $a_2 \cdot a_1^{y_1}a_2^{y_2}\dots a_n^{y_n} = a_1^{z_1}a_2^{z_2}\dots a_n^{z_n}$, где:

$z_1 = y_1$,

$z_2 = y_2 + 1$,

$z_3 = y_1 + y_3$,

$z_4 = 3y_1 + y_4 + 4y_1^2$,

$z_5 = y_5 + y_1y_2$,

$z_6 = 5y_1 + y_6 + 3y_1^2 + 6y_1^3$,

$z_7 = y_7 + 3y_1y_2 + 4y_1^2y_2$,

$z_8 = y_8 + 3y_1y_2 + 4y_1y_2^2,$

$z_9 = 5y_1 + y_9 + 6y_1^2 + 5y_1^3 + 5y_1^4,$

$z_{10} = 2y_1 + y_{10} + 5y_1y_2 + 4y_1y_3 + 3y_1^2y_2 + 3y_1^2y_3 + 6y_1^3y_2 + 4y_1^2 + y_1^3,$

$z_{11} = 5y_1 + y_{11} + 3y_1y_3 + 4y_1^2y_3 + 3y_1^2 + 6y_1^3,$

$z_{12} = y_{12} + 2y_1^2y_2^2 + 6y_1y_2 + 5y_1y_2^2 + y_1^2y_2 + 6y_1y_2y_3,$

$z_{13} = y_{13} + 3y_1y_2 + 4y_1^2y_2 + y_1y_2y_3,$

$z_{14} = y_{14} + 5y_1y_2 + 3y_1y_2^2 + 6y_1y_2^3.$

4) $a_2^{-1} \cdot a_1^{y_1}a_2^{y_2} \dots a_n^{y_n} = a_1^{z_1}a_2^{z_2} \dots a_n^{z_n}$, где:

$z_1 = y_1,$

$z_2 = y_2 + 6,$

$z_3 = 6y_1 + y_3,$

$z_4 = 4y_1 + y_4 + 3y_1^2,$

$z_5 = y_1 + y_5 + 6y_1y_2,$

$z_6 = 2y_1 + y_6 + 4y_1^2 + y_1^3,$

$z_7 = 3y_1 + y_7 + 4y_1y_2 + 3y_1^2y_2 + 4y_1^2,$

$z_8 = 6y_1 + y_8 + 5y_1y_2 + 3y_1y_2^2,$

$z_9 = 2y_1 + y_9 + y_1^2 + 2y_1^3 + 2y_1^4,$

$z_{10} = 3y_1 + y_{10} + 2y_1y_2 + 3y_1y_3 + 4y_1^2y_2 + 4y_1^2y_3 + y_1^3y_2 + 3y_1^2 + y_1^3,$

$z_{11} = 2y_1 + y_{11} + 4y_1y_3 + 3y_1^2y_3 + 5y_1^3,$

$z_{12} = y_1 + y_{12} + 5y_1^2y_2^2 + 4y_1y_2 + 6y_1y_3 + 2y_1y_2^2 + 2y_1^2y_2 + y_1y_2y_3,$

$z_{13} = 3y_1 + y_{13} + 4y_1y_2 + y_1y_3 + 4y_1^2y_2 + 3y_1^2 + 6y_1y_2y_3,$

$z_{14} = y_1 + y_{14} + 4y_1y_2 + y_1y_2^2 + y_1y_2^3.$

Для снижения вычислительной сложности алгоритма А–5 нам требуется быстрый способ для нумерации элементов. Пусть $g = a_1^{x_1} \dots a_k^{x_k}$ – произвольный элемент из B_k. Определим биективное отображение φ следующего вида:

$$\varphi(g) := (x_k x_{k-1} \dots x_1)_7,$$

$$\varphi^{-1}\Big((x_k x_{k-1} \dots x_1)_7\Big) := a_1^{x_1}a_2^{x_2} \dots a_k^{x_k}.$$

Здесь $\varphi(g)$ представляет собой целое неотрицательное число, записанное

в семеричной системе счисления, которое мы возьмем в качестве порядкового номера g. Легко увидеть, что $\varphi(g)$ пробегает все значения от 0 до $(7^k - 1)$.

Обозначим $X := \{a_1, a_2\}$ – минимальное порождающее множество B_k и $Y := X \cup X^{-1} = \{a_1, a_2, a_1^{-1}, a_2^{-1}\}$ – симметричное порождающее множество.

Для исследования графов Кэли групп B_k относительно порождающих множеств X и Y в алгоритм А–4 был добавлен указанный выше быстрый способ нумерации элементов. В итоге, используя вычисления на суперкомпьютере, удалось получить функции роста групп B_k при $k \leq 14$, а также характеристики соответствующих графов Кэли.

В таблицах 1.4 и 1.5 приведены диаметры и средние диаметры графов Кэли групп B_k для $k \leq 14$ относительно порождающих множеств X и Y соответственно.

В таблицах 1.6–1.18 представлены функции роста групп B_k относительно порождающего множества X, а в таблицах 1.19–1.31 – относительно Y.

На рисунках 2.3–1.5.2 приведены графики функций роста групп B_k относительно порождающего множества X, а на рисунках 1.17–1.5.2 – относительно Y.

Теорема 1.5 *Пусть $B_k = \langle X \rangle$ при $k \leq 14$. В этом случае значения диаметров $D_X(B_k)$ и средних диаметров $\overline{D}_X(B_k)$ такие как в таблице* 1.4*, а соответствующие функции роста заданы в таблицах* 1.6–1.18*.*

Таблица 1.4: Характеристики графов Кэли групп $B_k = \langle X \rangle$.

k	2	3	4	5	6	7	8	9	10	11	12	13	14
D	12	14	18	23	28	28	35	36	39	42	43	49	56
$\overline{D}_X$	6	8	11	15	17	20	23	26	28	31	34	37	41

Доказательство. Значения диаметров $D_X(B_k)$ и средних диаметров $\overline{D}_X(B_k)$ при $k \leq 14$ вычислены по алгоритму А–5. Поскольку алгоритм А–5 является корректным, то определяемые значения $D_X(B_k)$ и $\overline{D}_X(B_k)$ также являются корректными. ■

Теорема 1.6 *Пусть* $B_k = \langle Y \rangle$ *при* $k \leq 14$. *В этом случае значения диаметров* $D_Y(B_k)$ *и средних диаметров* $\overline{D}_Y(B_k)$ *такие как в таблице* 1.5, *а соответствующие функции роста заданы в таблицах* 1.19–1.31.

Пусть $S_n = \langle X_n \rangle$. В этом случае, при $n = 14, 15$, верны следующие утверждения:

Таблица 1.5: Характеристики графов Кэли групп $B_k = \langle Y \rangle$.

k	2	3	4	5	6	7	8	9	10	11	12	13	14
D	6	8	11	12	15	17	21	22	24	26	27	31	34
$\overline{D}_Y$	3	5	7	8	10	12	14	16	18	20	21	23	25

Доказательство. Значения диаметров $D_Y(B_k)$ и средних диаметров $\overline{D}_Y(B_k)$ при $k \leq 14$ вычислены по алгоритму А–5. Поскольку алгоритм А–5 является корректным, то определяемые значения $D_Y(B_k)$ и $\overline{D}_Y(B_k)$ также являются корректными. ■

Таблица 1.6: Функция роста $B_2 = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	5	6	10	3
1	2	6	7	11	2
2	3	7	6	12	1
3	4	8	5		
4	5	9	4		

Таблица 1.7: Функция роста $B_3 = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	5	26	10	41
1	2	6	35	11	34
2	4	7	42	12	23
3	8	8	47	13	14
4	15	9	45	14	6

Таблица 1.8: Функция роста $B_4 = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	7	116	14	226
1	2	8	192	15	149
2	4	9	264	16	75
3	8	10	308	17	27
4	16	11	325	18	2
5	32	12	311		
6	63	13	280		

Таблица 1.9: Функция роста $B_5 = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	8	223	16	1993
1	2	9	398	17	1727
2	4	10	666	18	1343
3	8	11	1042	19	883
4	16	12	1486	20	460
5	32	13	1878	21	175
6	63	14	2105	22	33
7	120	15	2143	23	6

Таблица 1.10: Функция роста $B_6 = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	10	957	20	10810
1	2	11	1833	21	7314
2	4	12	3399	22	3674
3	8	13	5931	23	1178
4	16	14	9365	24	233
5	32	15	12881	25	26
6	64	16	15133	26	6
7	126	17	15609	27	2
8	250	18	14932	28	1
9	492	19	13370		

Таблица 1.11: Функция роста $B_7 = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	10	984	20	109911
1	2	11	1948	21	103335
2	4	12	3815	22	91288
3	8	13	7410	23	72061
4	16	14	14146	24	45965
5	32	15	26109	25	19503
6	64	16	45088	26	3802
7	126	17	70692	27	264
8	250	18	96182	28	41
9	496	19	110000		

Таблица 1.12: Функция роста $B_8 = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	12	3832	24	723856
1	2	13	7506	25	656391
2	4	14	14594	26	547416
3	8	15	27992	27	400554
4	16	16	52198	28	238734
5	32	17	94559	29	104018
6	64	18	164762	30	28700
7	126	19	272183	31	4136
8	250	20	415379	32	424
9	496	21	570068	33	80
10	984	22	691281	34	40
11	1950	23	742145	35	20

Таблица 1.13: Функция роста $B_9 = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	13	7666	26	5285744
1	2	14	15188	27	5028234
2	4	15	30010	28	4476825
3	8	16	58242	29	3619050
4	16	17	111675	30	2491979
5	32	18	211668	31	1314091
6	64	19	395499	32	450331
7	126	20	723285	33	79658
8	250	21	1272333	34	5552
9	496	22	2115487	35	328
10	984	23	3214080	36	2
11	1952	24	4338043		
12	3868	25	5100834		

Таблица 1.14: Функция роста $B_{10} = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	14	15194	28	37654869
1	2	15	30094	29	37016705
2	4	16	59447	30	34254885
3	8	17	117184	31	29466298
4	16	18	230473	32	22202310
5	32	19	451806	33	13167863
6	64	20	881091	34	5127397
7	126	21	1703457	35	979960
8	250	22	3247079	36	59477
9	496	23	6038042	37	733
10	984	24	10769113	38	20
11	1952	25	17918854	39	2
12	3868	26	26730038		
13	7666	27	34337389		

Таблица 1.15: Функция роста $B_{11} = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	15	30096	30	245328766
1	2	16	59469	31	265017401
2	4	17	117335	32	258048480
3	8	18	231210	33	237602927
4	16	19	454897	34	202759568
5	32	20	893137	35	150184264
6	64	21	1748508	36	85695268
7	126	22	3409987	37	30478439
8	250	23	6608075	38	4648917
9	496	24	12690604	39	166336
10	984	25	24027255	40	3067
11	1952	26	44410458	41	1254
12	3868	27	78813560	42	623
13	7666	28	130526533		
14	15194	29	193339646		

Таблица 1.16: Функция роста $B_{12} = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	15	30106	30	637535150
1	2	16	59620	31	1040687484
2	4	17	118024	32	1492693778
3	8	18	233540	33	1803070327
4	16	19	461898	34	1864547940
5	32	20	912817	35	1781416672
6	64	21	1802493	36	1613641687
7	126	22	3555635	37	1332728438
8	250	23	7003035	38	922755091
9	496	24	13765152	39	457422311
10	984	25	26975146	40	121184353
11	1952	26	52606265	41	10105826
12	3868	27	101752430	42	119769
13	7666	28	193972488	43	426
14	15194	29	360098637		

Таблица 1.17: Функция роста $B_{13} = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	17	118024	34	7275336600
1	2	18	233544	35	10348342166
2	4	19	461922	36	12504419490
3	8	20	912953	37	12986907222
4	16	21	1803081	38	12408626965
5	32	22	3558286	39	11221127101
6	64	23	7014036	40	9267624275
7	126	24	13808840	41	6460835309
8	250	25	27144164	42	3288077264
9	496	26	53251742	43	940187997
10	984	27	104189907	44	94515106
11	1952	28	203047018	45	1555763
12	3868	29	393091070	46	21986
13	7666	30	752790717	47	9425
14	15194	31	1417176617	48	4702
15	30106	32	2592758438	49	2351
16	59620	33	4519935937		

Таблица 1.18: Функция роста $B_{14} = \langle X \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	19	461930	38	57882148948
1	2	20	913002	39	73362302255
2	4	21	1803266	40	83192272509
3	8	22	3558901	41	85169026095
4	16	23	7016029	42	80133871146
5	32	24	13815188	43	69743986789
6	64	25	27164864	44	55001283615
7	126	26	53321773	45	37527749947
8	250	27	104437179	46	20730704342
9	496	28	203955597	47	8505351368
10	984	29	396478656	48	2325932994
11	1952	30	765560449	49	371226046
12	3868	31	1465086007	50	30269266
13	7666	32	2768568784	51	1203772
14	15194	33	5137581783	52	91301
15	30106	34	9288684837	53	34650
16	59620	35	16178134746	54	17144
17	118024	36	26714686437	55	8570
18	233544	37	41113886422	56	4285

Таблица 1.19: Функция роста $B_2 = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	3	12	6	4
1	4	4	12		
2	8	5	8		

Таблица 1.20: Функция роста $B_3 = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	3	36	6	58
1	4	4	78	7	8
2	12	5	144	8	2

Таблица 1.21: Функция роста $B_4 = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	4	96	8	448
1	4	5	248	9	128
2	12	6	584	10	24
3	36	7	818	11	2

Таблица 1.22: Функция роста $B_5 = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	5	304	10	2386
1	4	6	864	11	202
2	12	7	2276	12	12
3	36	8	4870		
4	104	9	5736		

Таблица 1.23: Функция роста $B_6 = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	6	888	12	17162
1	4	7	2518	13	1658
2	12	8	6868	14	66
3	36	9	16736	15	6
4	104	10	32178		
5	304	11	39108		

Таблица 1.24: Функция роста $B_7 = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	6	888	12	254626
1	4	7	2546	13	272304
2	12	8	7240	14	80008
3	36	9	19860	15	2222
4	104	10	52786	16	36
5	304	11	130562	17	4

Таблица 1.25: Функция роста $B_8 = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	8	7344	16	817808
1	4	9	20740	17	97694
2	12	10	57314	18	2000
3	36	11	154416	19	258
4	104	12	392056	20	52
5	304	13	887826	21	4
6	888	14	1578296		
7	2560	15	1745084		

Таблица 1.26: Функция роста $B_9 = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	8	7344	16	11800900
1	4	9	20904	17	11978996
2	12	10	59006	18	4435628
3	36	11	164734	19	284302
4	104	12	453434	20	1994
5	304	13	1214782	21	126
6	888	14	3071636	22	4
7	2560	15	6855908		

Таблица 1.27: Функция роста $B_{10} = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	9	21584	18	94754842
1	4	10	62288	19	68335662
2	12	11	179122	20	11467010
3	36	12	513400	21	132938
4	104	13	1462186	22	1724
5	304	14	4106244	23	204
6	888	15	11176234	24	8
7	2574	16	28404404		
8	7460	17	61846016		

Таблица 1.28: Функция роста $B_{11} = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	9	21584	18	222798588
1	4	10	62288	19	478782628
2	12	11	179254	20	692516568
3	36	12	514812	21	405458620
4	104	13	1473004	22	38184268
5	304	14	4192588	23	101898
6	888	15	11827954	24	2130
7	2574	16	32820802	25	304
8	7460	17	88378058	26	12

Таблица 1.29: Функция роста $B_{12} = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	10	64064	20	2026136294
1	4	11	186476	21	4052920330
2	12	12	542128	22	4746223024
3	36	13	1573516	23	1679746016
4	104	14	4556024	24	52622844
5	304	15	13150238	25	45912
6	888	16	37783394	26	7102
7	2588	17	107656354	27	614
8	7544	18	301801872		
9	21992	19	816237526		

Таблица 1.30: Функция роста $B_{13} = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	11	186476	22	15270545778
1	4	12	542128	23	28900625456
2	12	13	1573696	24	31648007232
3	36	14	4558744	25	10581633652
4	104	15	13176940	26	328715474
5	304	16	37988020	27	193092
6	888	17	109047018	28	18342
7	2588	18	310945494	29	3304
8	7544	19	876497018	30	270
9	21992	20	2418535902	31	4
10	64064	21	6386118830		

Таблица 1.31: Функция роста $B_{14} = \langle Y \rangle$

s	$F(s)$	s	$F(s)$	s	$F(s)$
0	1	12	543604	24	99025768312
1	4	13	1580284	25	170950704030
2	12	14	4586236	26	196493554258
3	36	15	13284300	27	114203733630
4	104	16	38384232	28	22108855406
5	304	17	110498248	29	741260730
6	888	18	316466942	30	2333160
7	2588	19	899270392	31	141320
8	7544	20	2523973960	32	28984
9	21992	21	6938126426	33	3744
10	64104	22	18379922058	34	88
11	186760	23	45469768168		

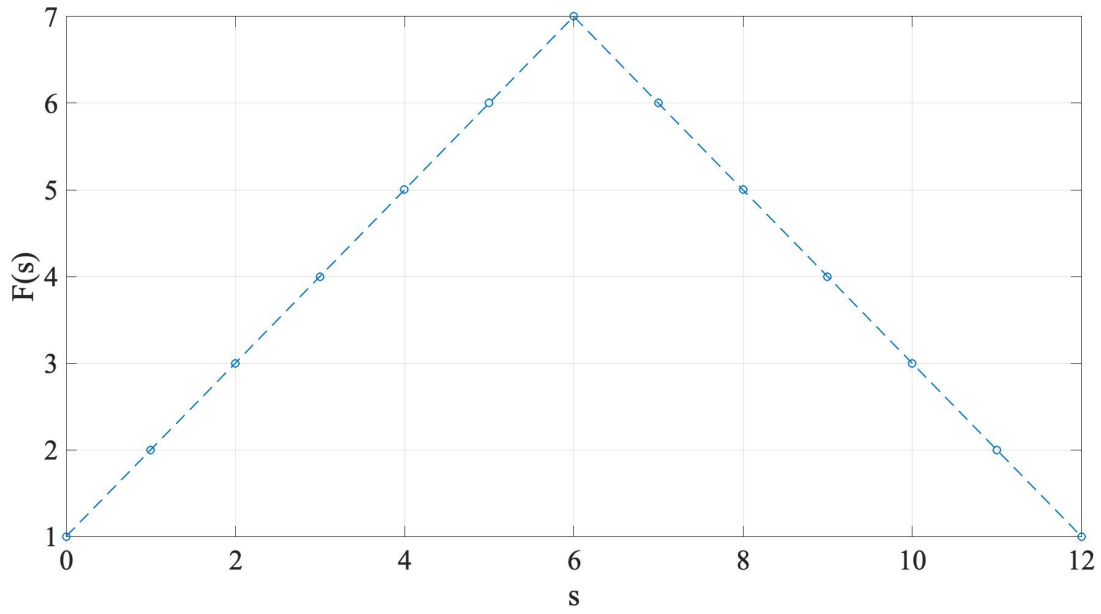


Рис. 1.4: График функции роста $B_2 = \langle X \rangle$.

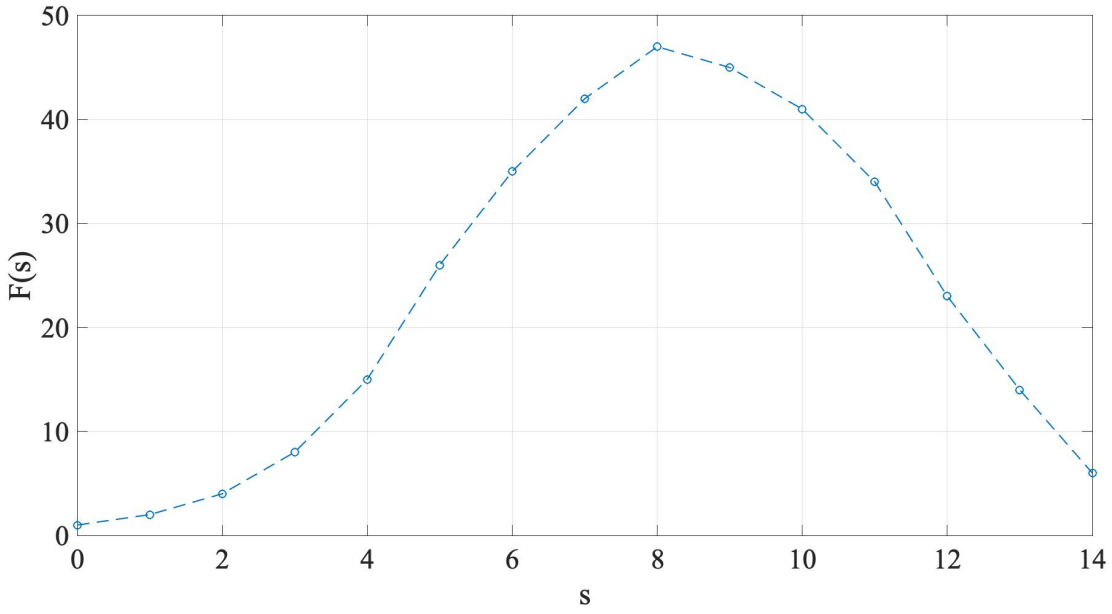


Рис. 1.5: График функции роста $B_3 = \langle X \rangle$.

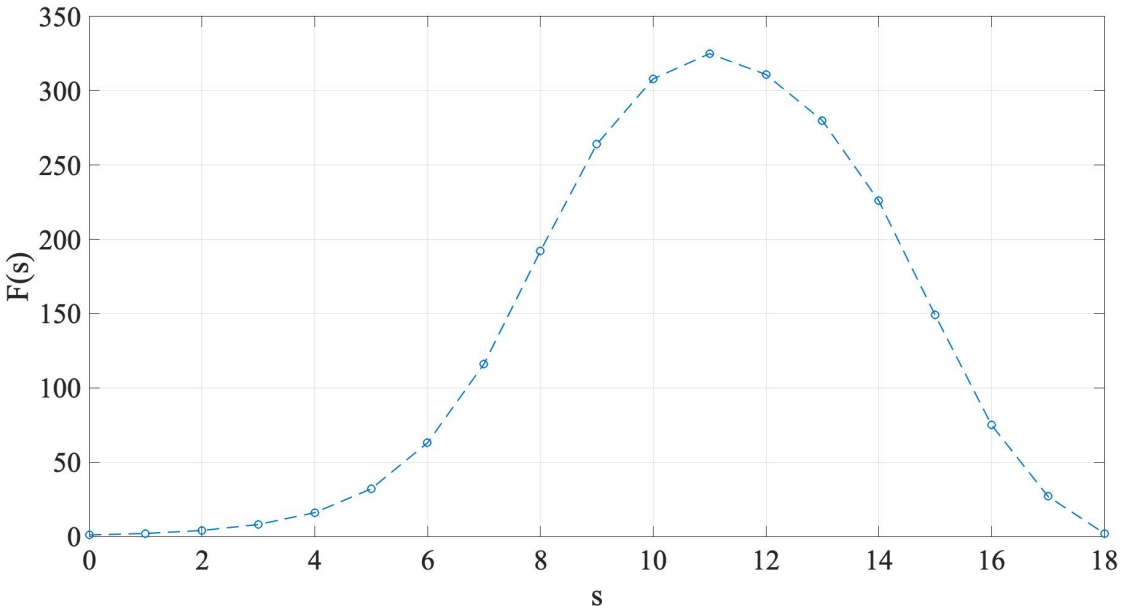


Рис. 1.6: График функции роста $B_4 = \langle X \rangle$.

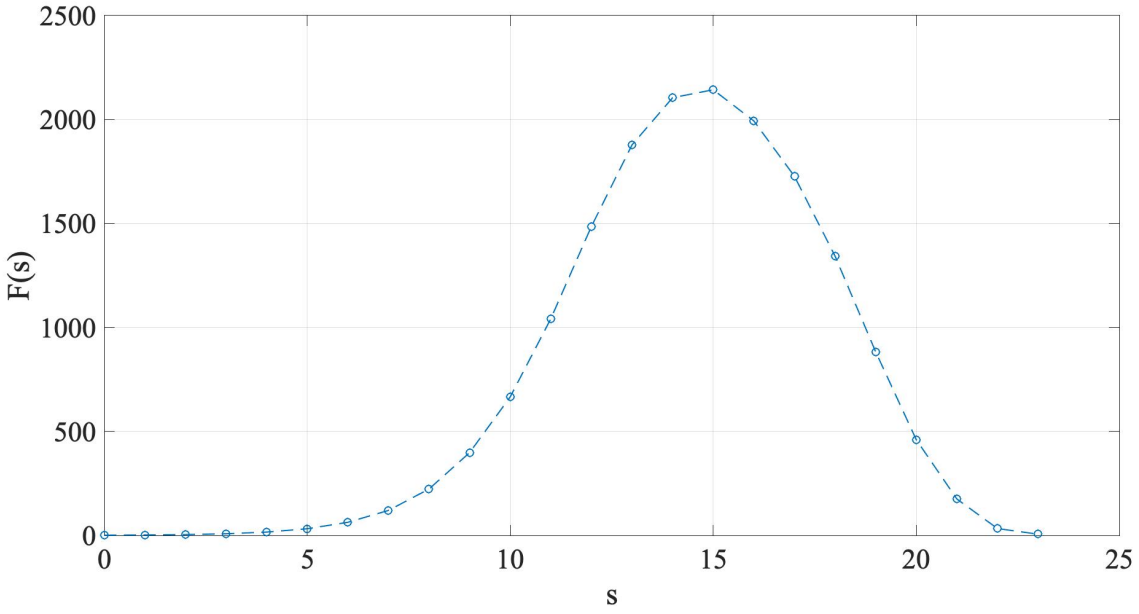


Рис. 1.7: График функции роста $B_5 = \langle X \rangle$.

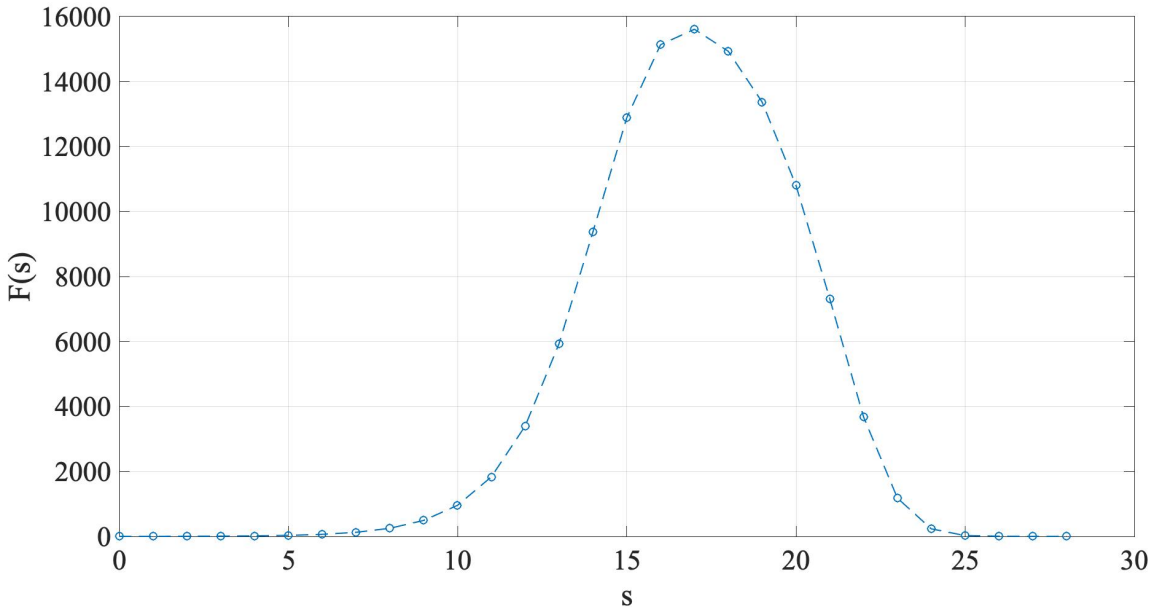


Рис. 1.8: График функции роста $B_6 = \langle X \rangle$.

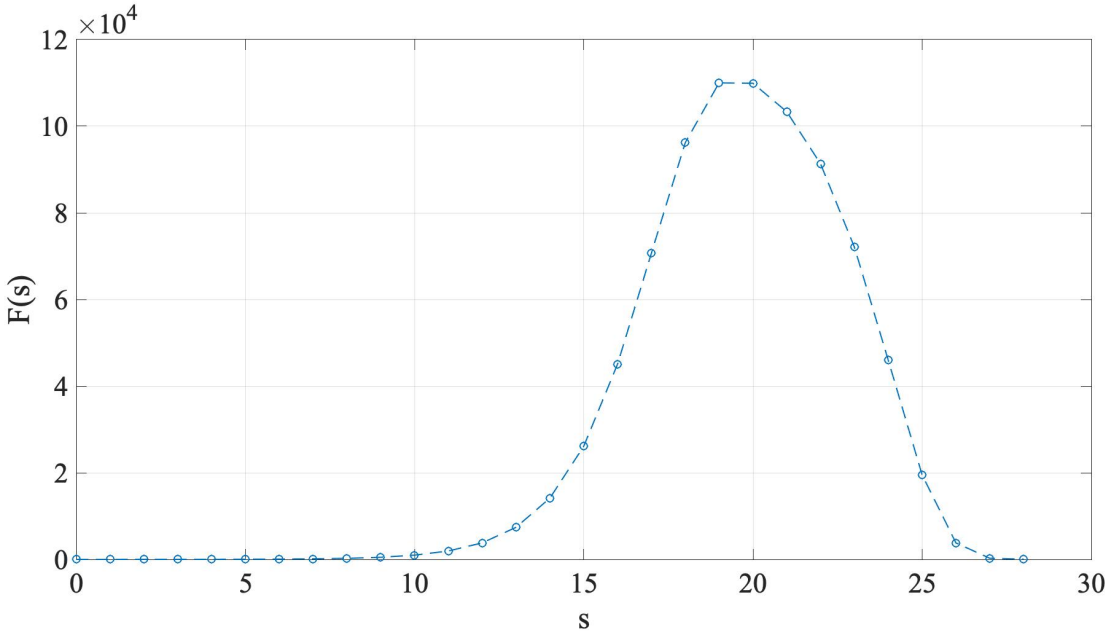


Рис. 1.9: График функции роста $B_7 = \langle X \rangle$.

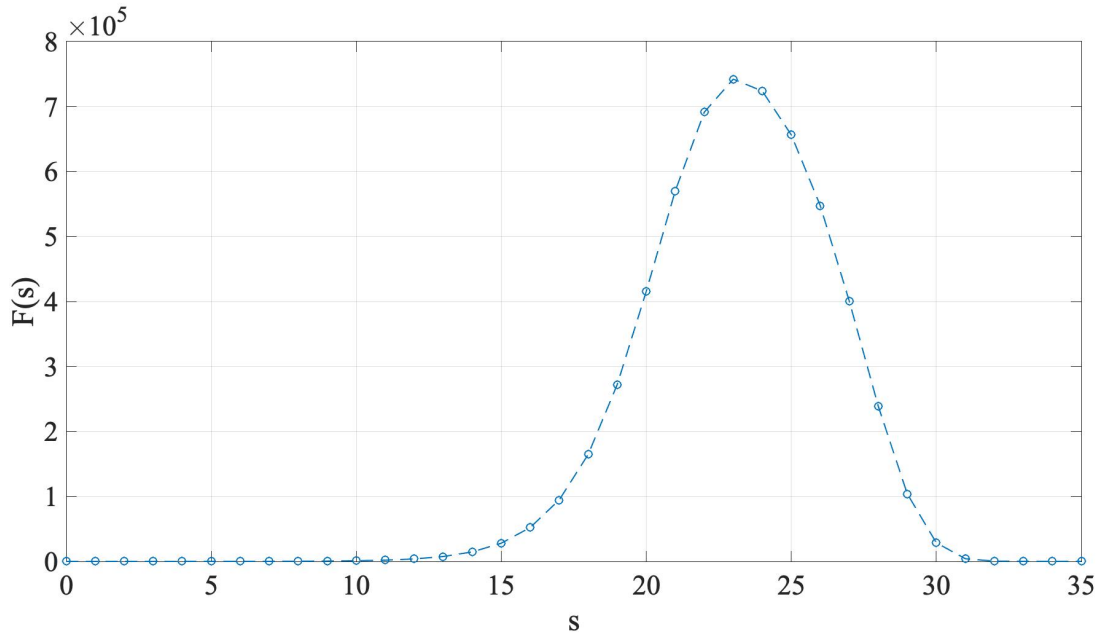


Рис. 1.10: График функции роста $B_8 = \langle X \rangle$.

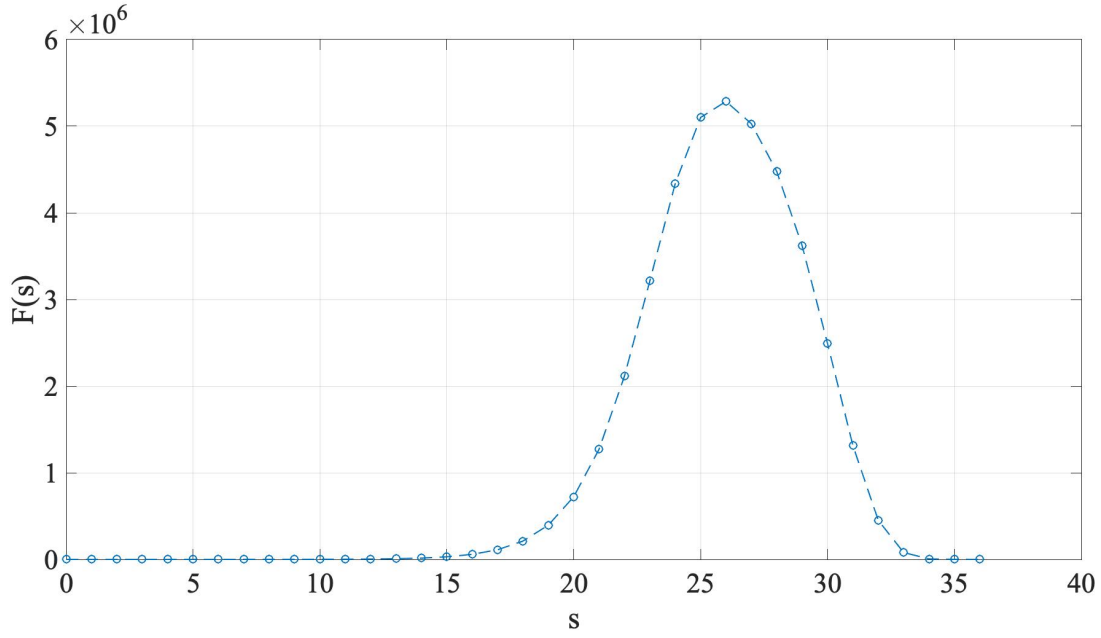


Рис. 1.11: График функции роста $B_9 = \langle X \rangle$.

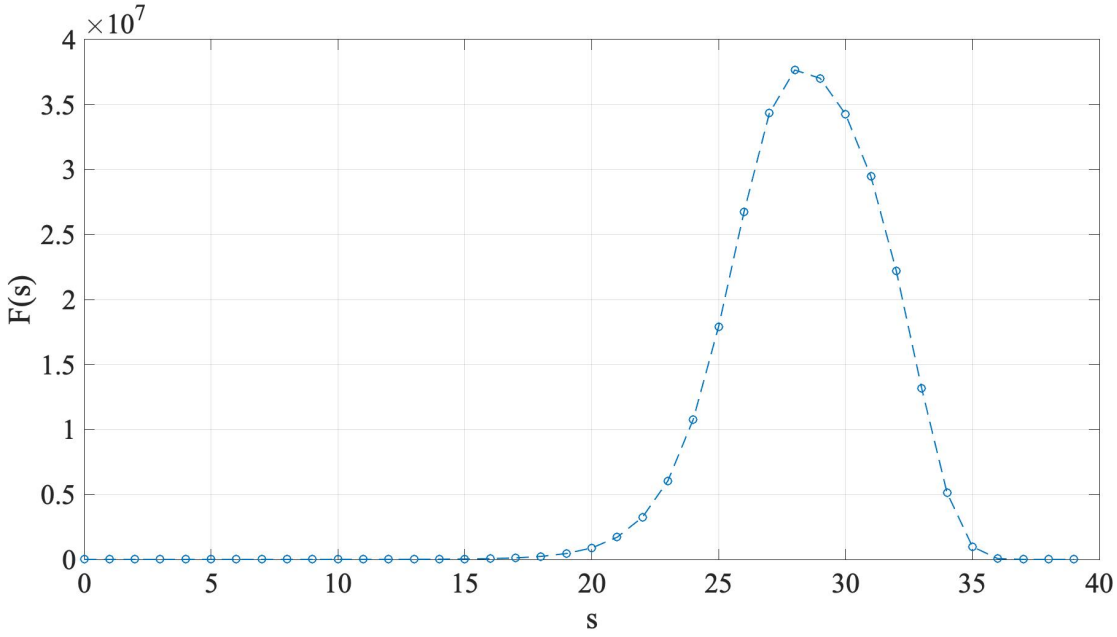


Рис. 1.12: График функции роста $B_{10} = \langle X \rangle$.

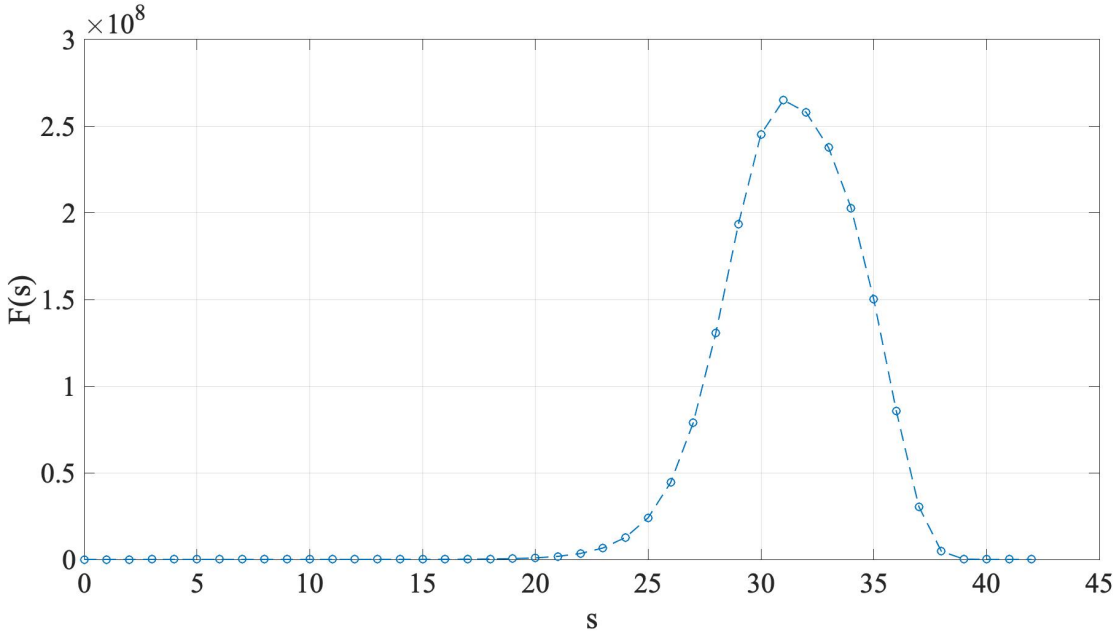


Рис. 1.13: График функции роста $B_{11} = \langle X \rangle$.

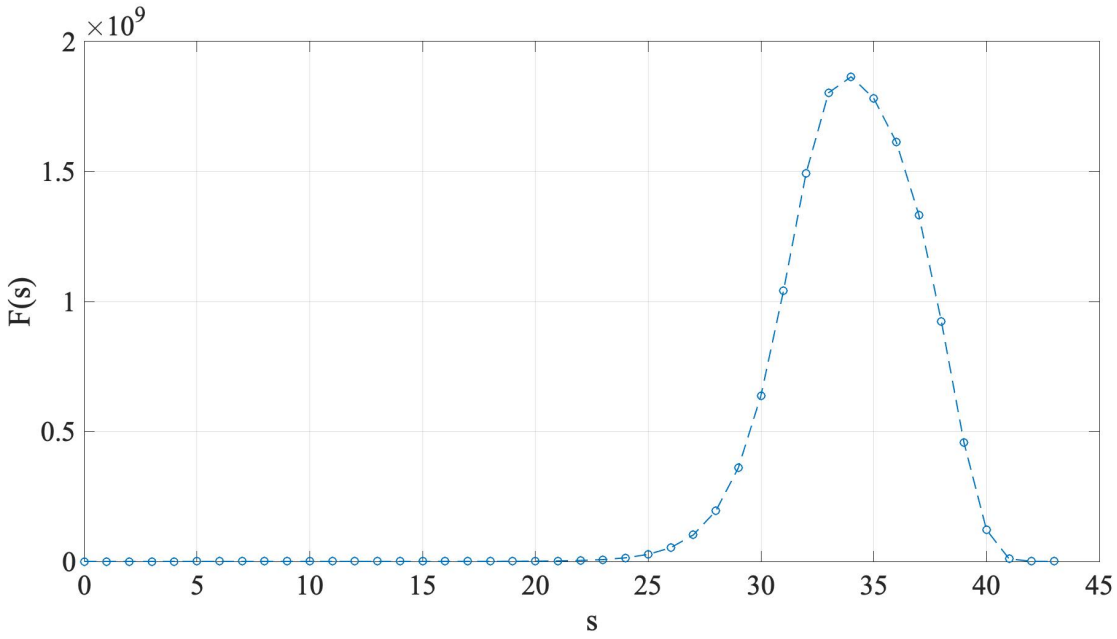


Рис. 1.14: График функции роста $B_{12} = \langle X \rangle$.

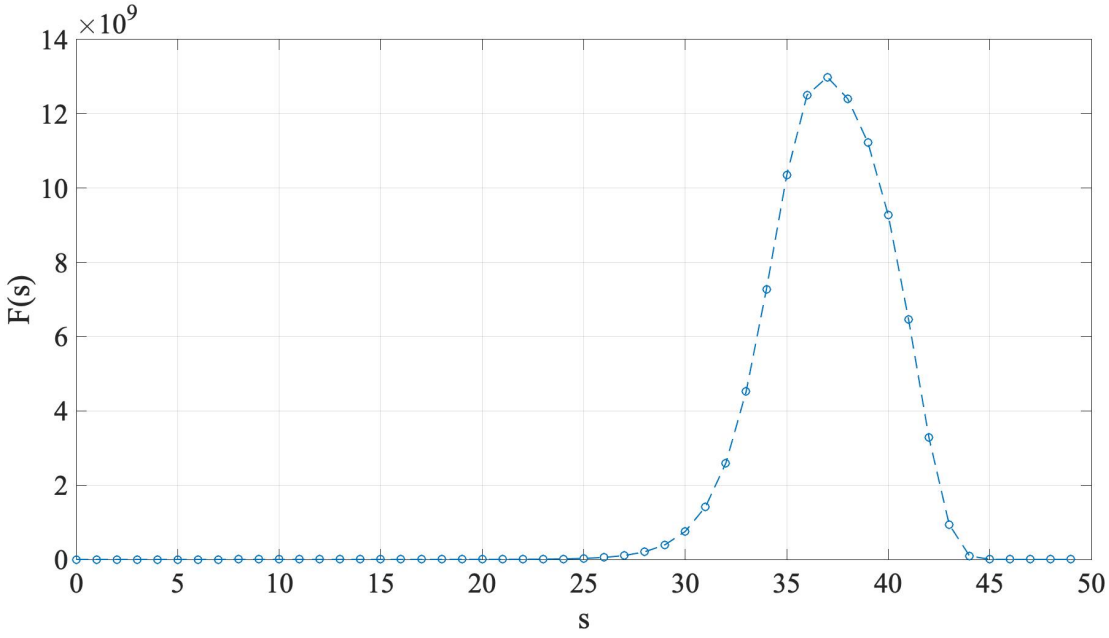


Рис. 1.15: График функции роста $B_{13} = \langle X \rangle$.

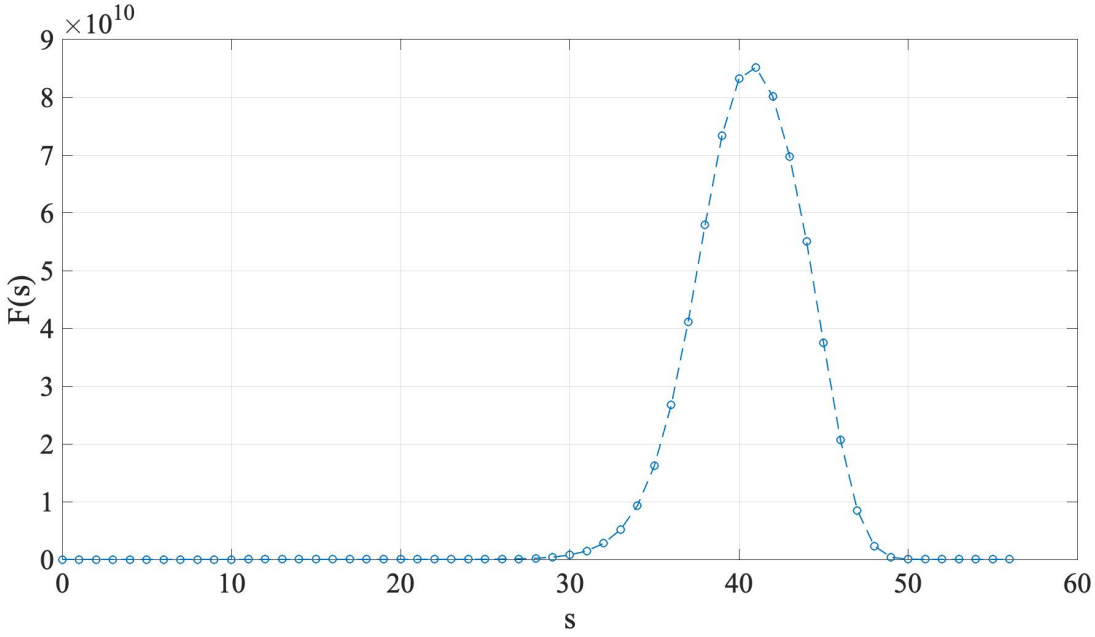


Рис. 1.16: График функции роста $B_{14} = \langle X \rangle$.

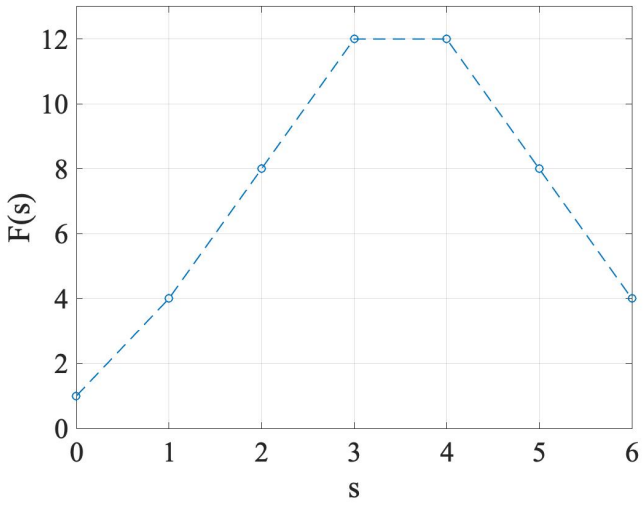


Рис. 1.17: График функции роста $B_2 = \langle Y \rangle$.

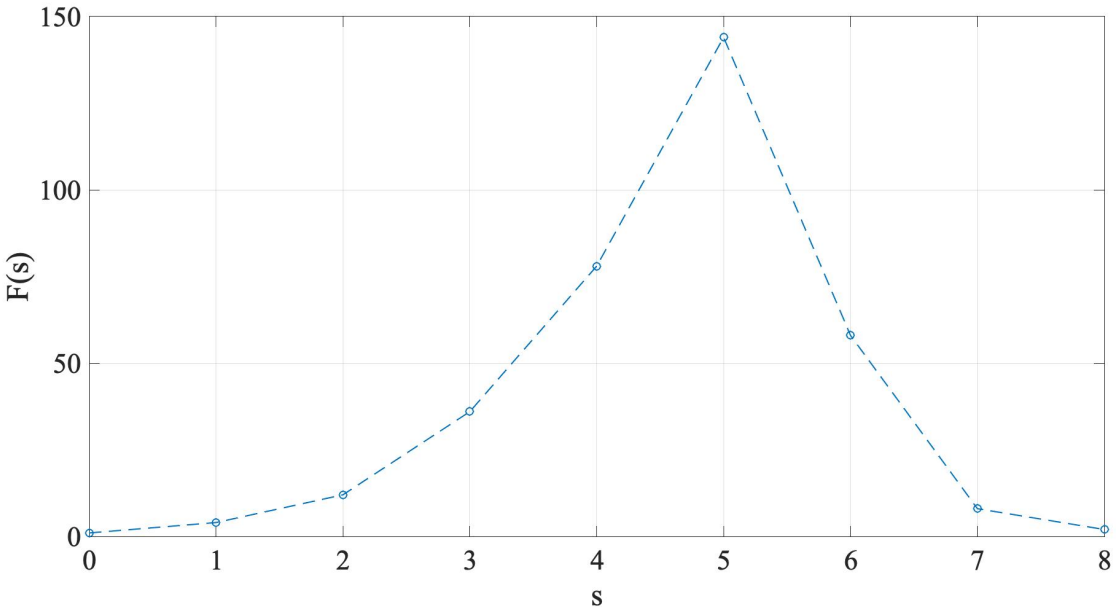


Рис. 1.18: График функции роста $B_3 = \langle Y \rangle$.

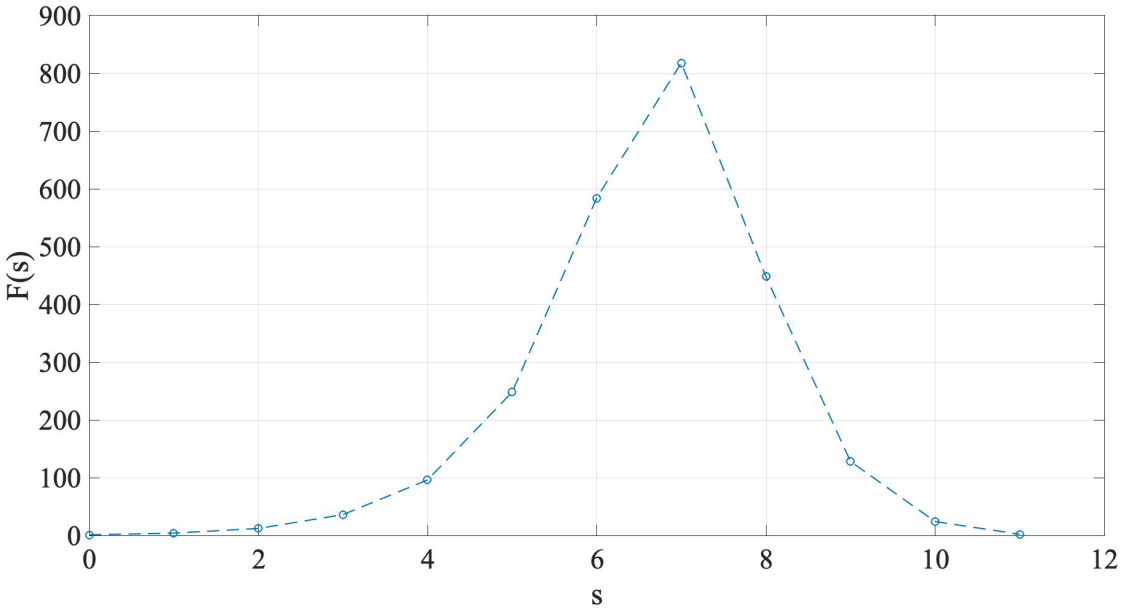


Рис. 1.19: График функции роста $B_4 = \langle Y \rangle$.

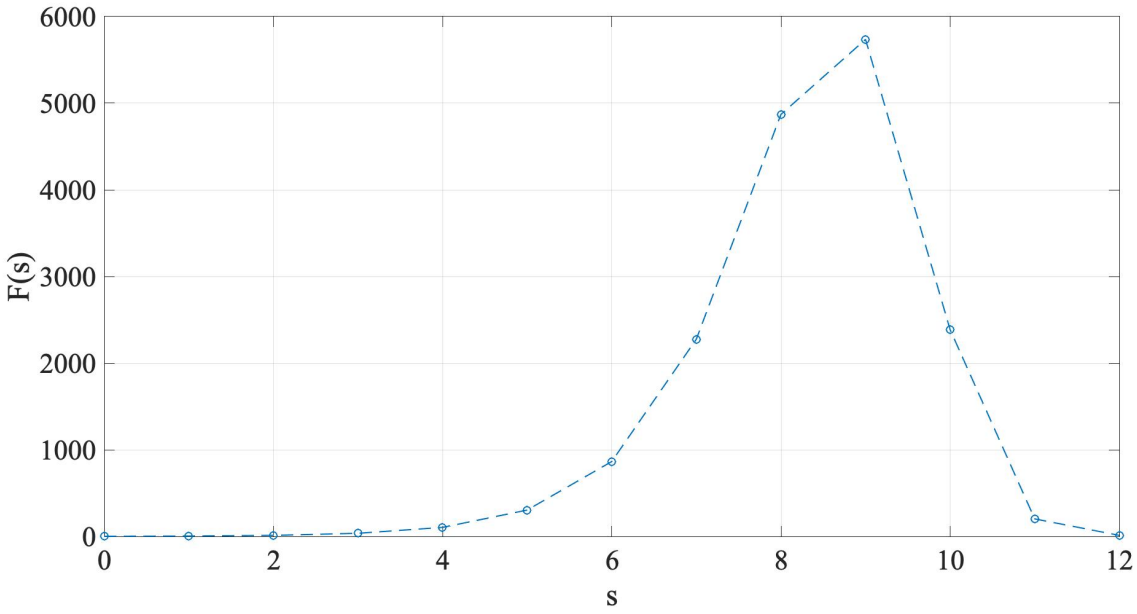


Рис. 1.20: График функции роста $B_5 = \langle Y \rangle$.

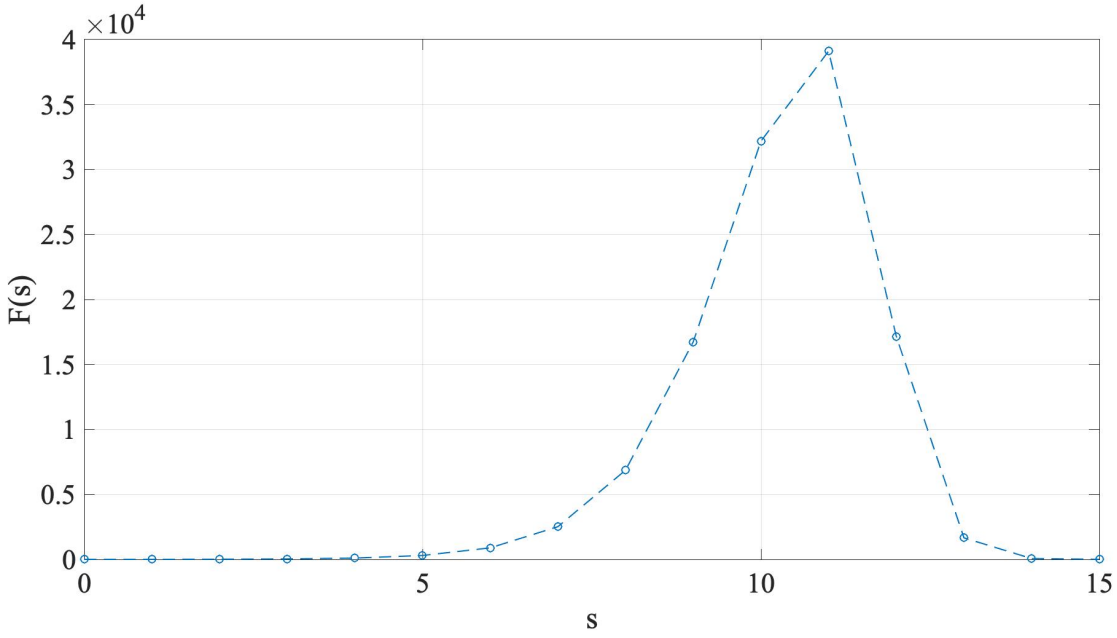


Рис. 1.21: График функции роста $B_6 = \langle Y \rangle$.

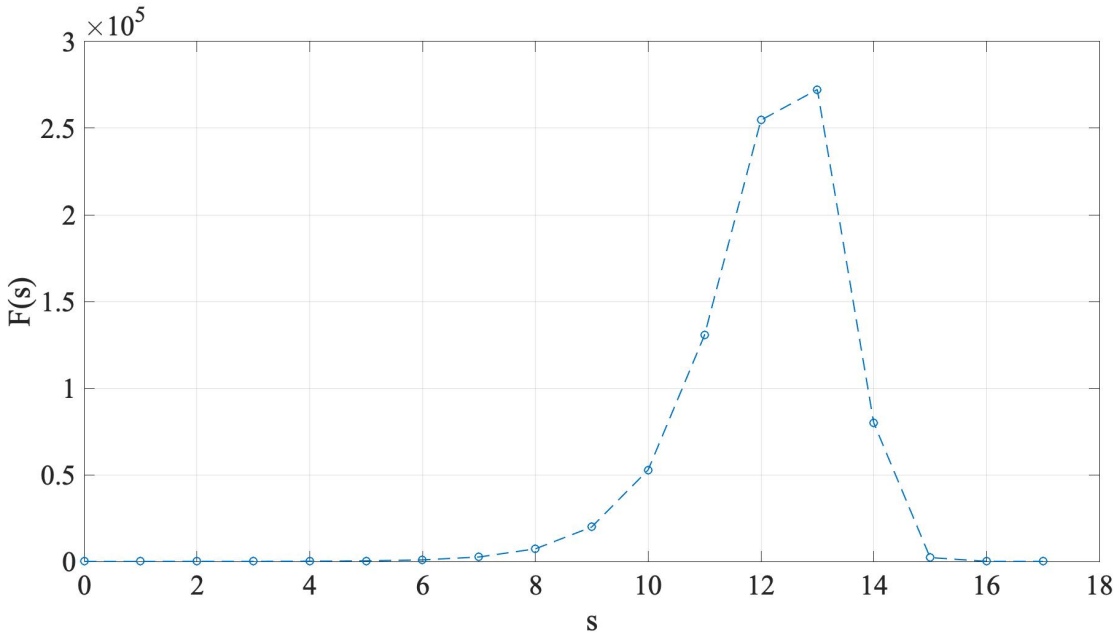


Рис. 1.22: График функции роста $B_7 = \langle Y \rangle$.

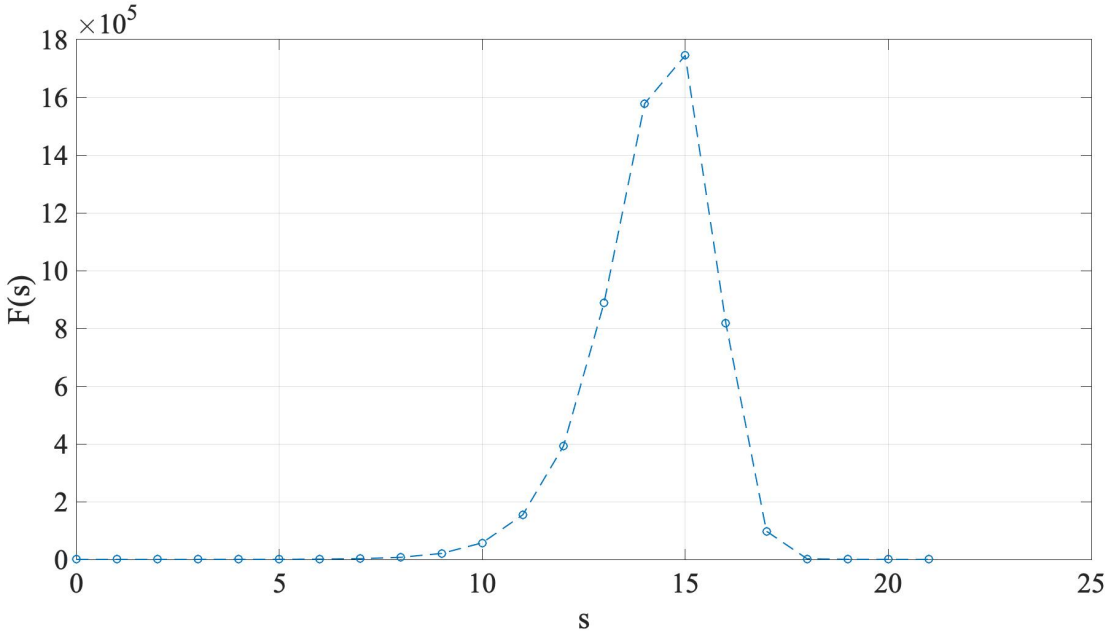


Рис. 1.23: График функции роста $B_8 = \langle Y \rangle$.

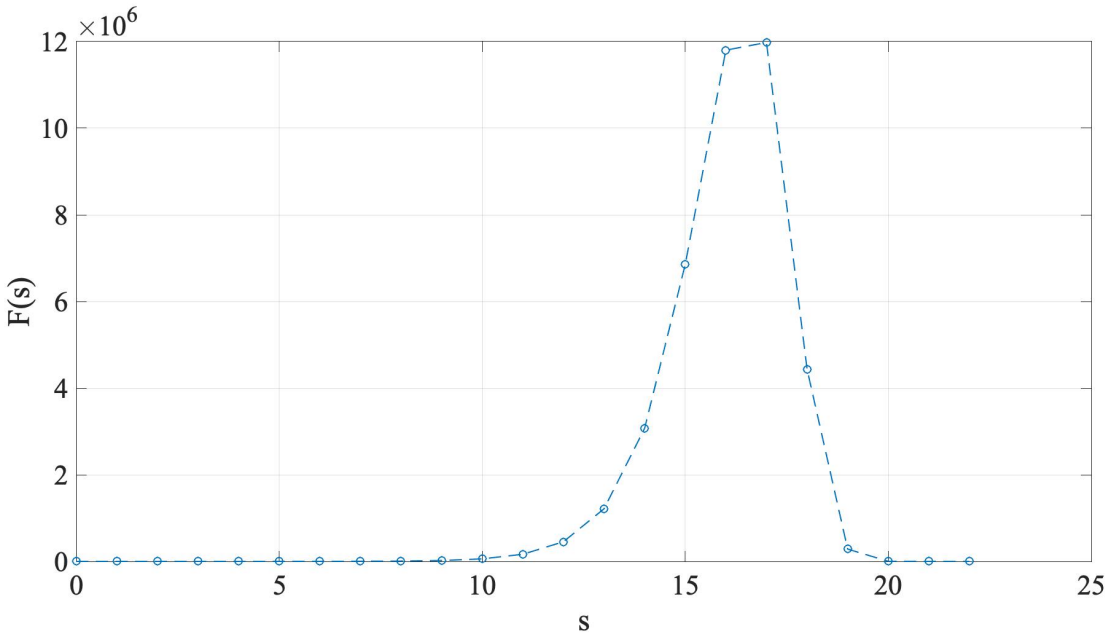


Рис. 1.24: График функции роста $B_9 = \langle Y \rangle$.

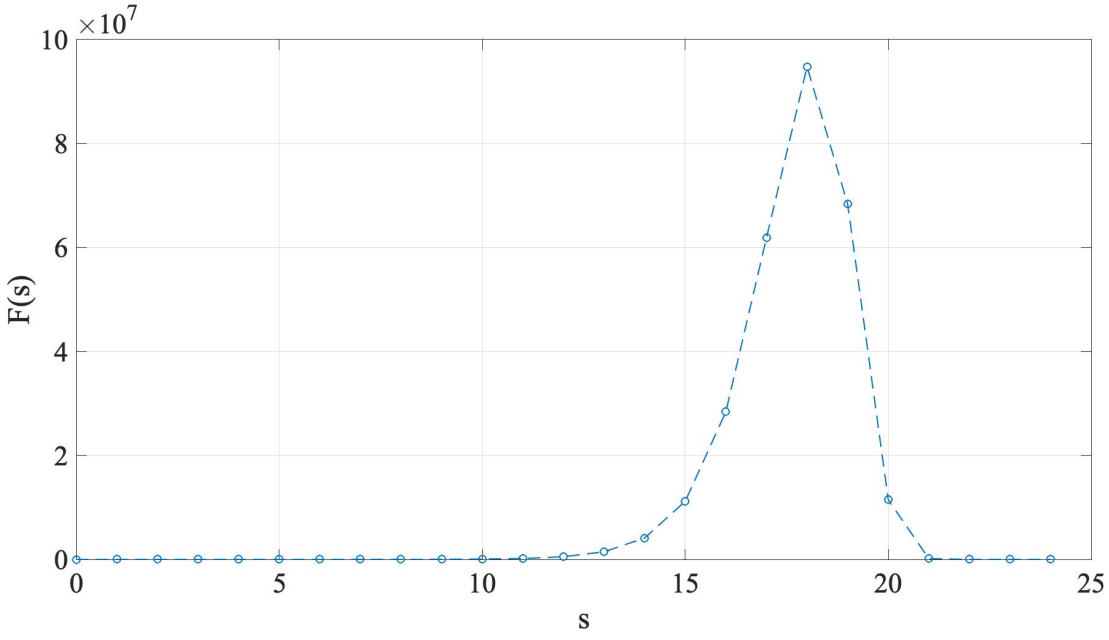


Рис. 1.25: График функции роста $B_{10} = \langle Y \rangle$.

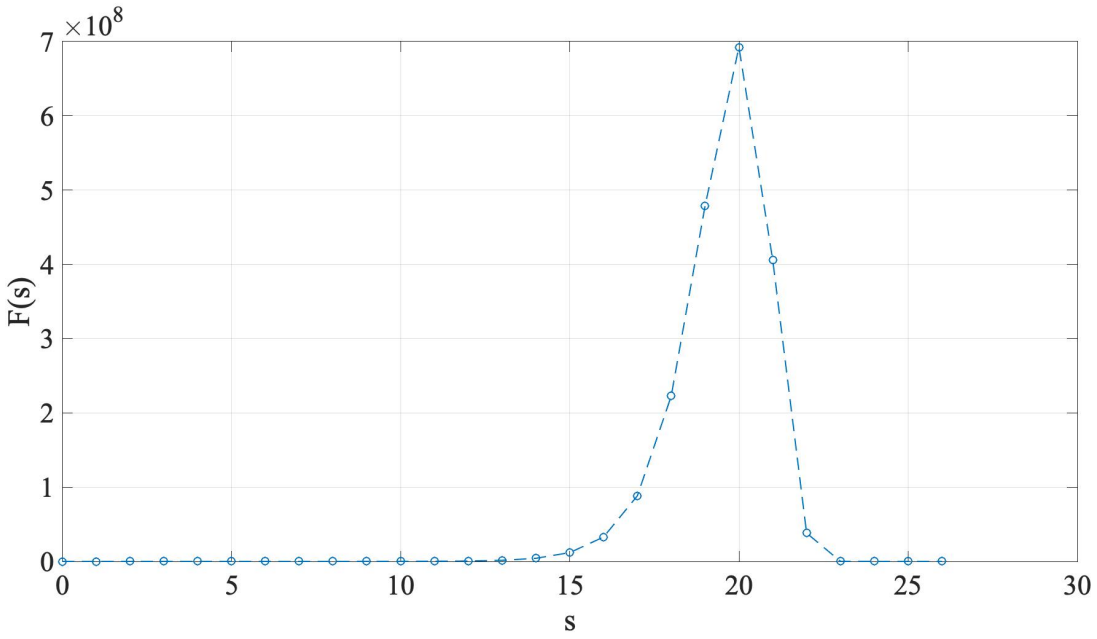


Рис. 1.26: График функции роста $B_{11} = \langle Y \rangle$.

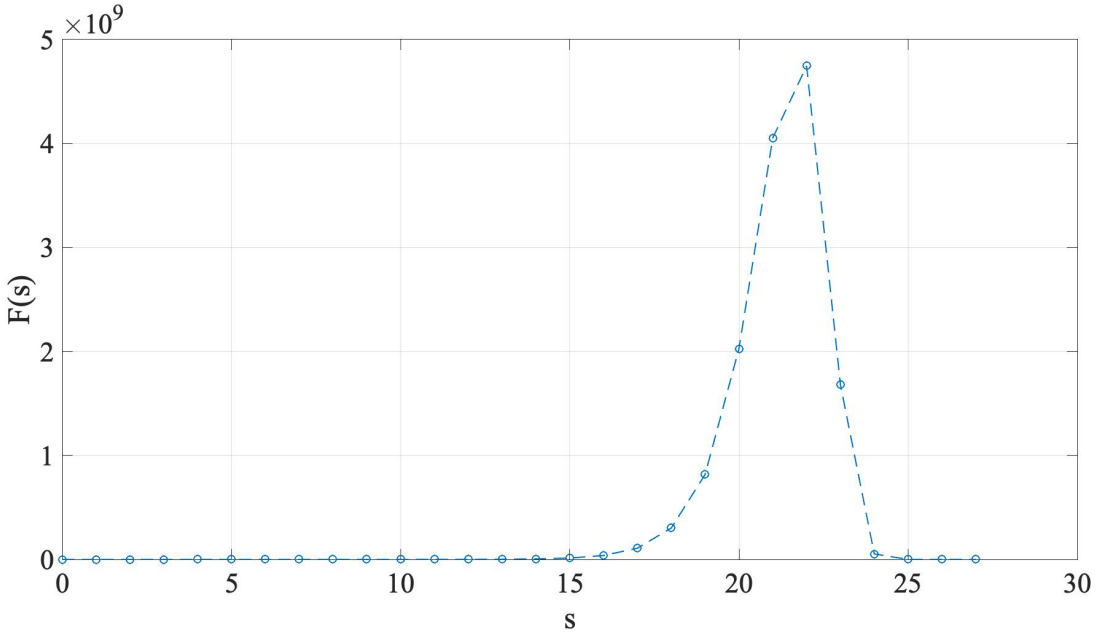


Рис. 1.27: График функции роста $B_{12} = \langle Y \rangle$.

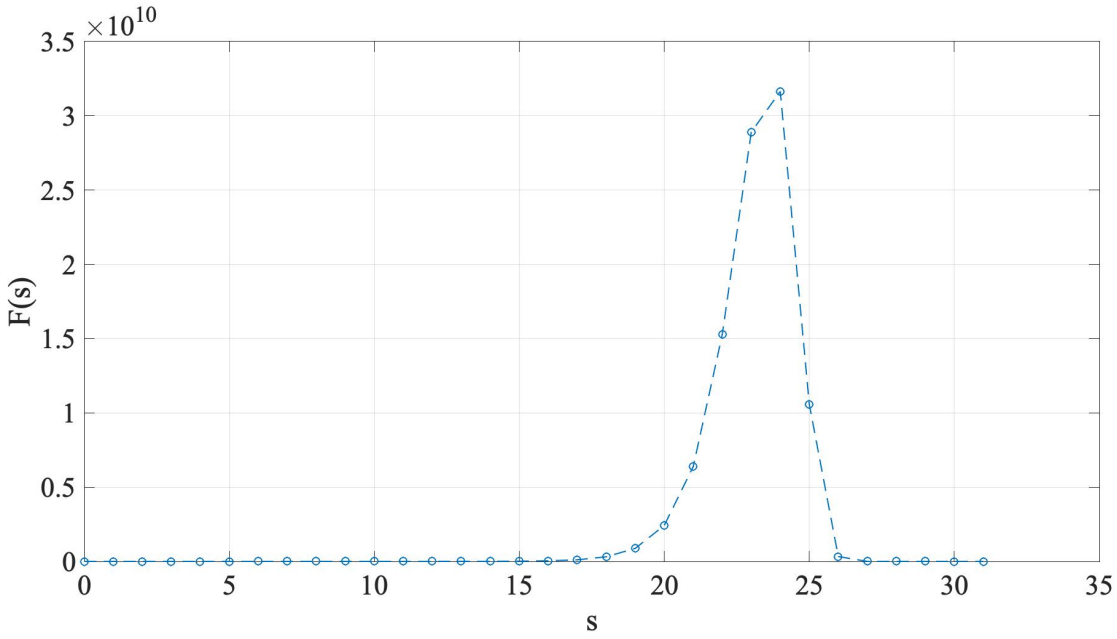


Рис. 1.28: График функции роста $B_{13} = \langle Y \rangle$.

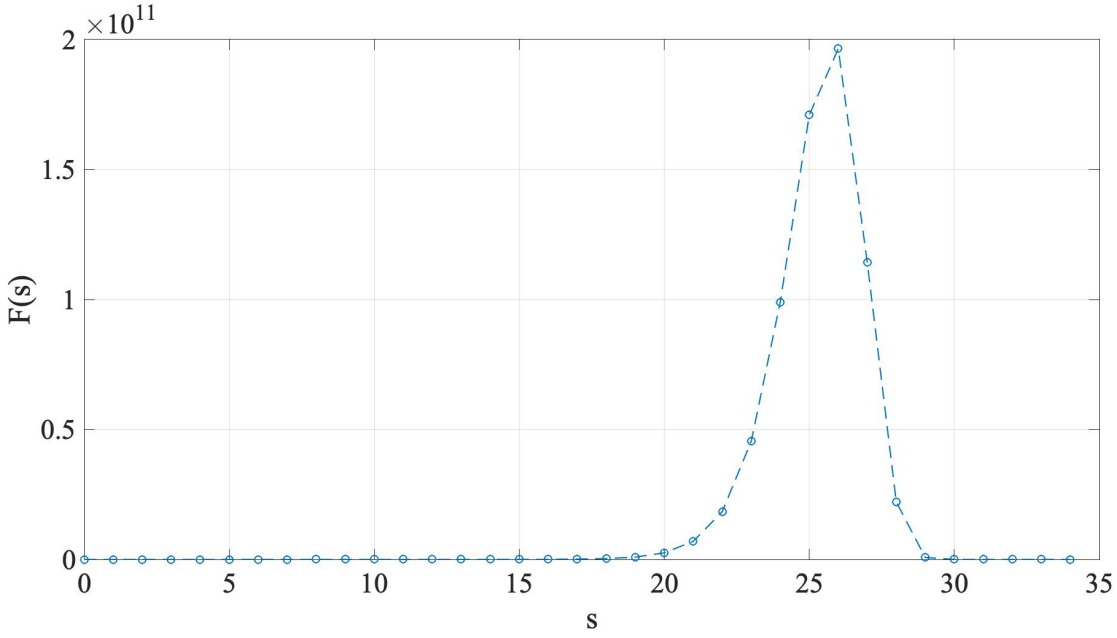


Рис. 1.29: График функции роста $B_{14} = \langle Y \rangle$.

Глава 2

Алгоритм расширенного синтаксического анализа языков программирования методом иерархии маркированных скобок

2.1 Постановка расширенной проблемы синтаксического анализа с учётом порядка применения продукций

Для того, чтобы сформулировать расширенную проблему синтаксического анализа мономов (программ), написанных на кс–языке (программирования), будем использовать следующие общепринятые определения.

Базовым объектом в теории контекстно–свободных языков и грамматик является алфавит $z_1, \ldots, z_n, x_1, \ldots, x_m$, который удобно разделять на две группы символов.

А именно, символы $x_1, \ldots, x_m$ первой группы называются терминальными символами; они образуют словарь контекстно–свободного языка. Символы другой группы $z_1, \ldots, z_n$ называются нетерминальными; они играют вспомогательную роль, поскольку нужны для задания грамматики — совокупности грамматических правил, порождающих контекстно–свободный язык.

По правилам грамматики формируются корректные мономы от терми-

нальных символов $x_1, \ldots, x_m$, которые интерпретируются как правильные предложения языка [11, 12, 72].

Над символами алфавита определены три операции:

– некоммутативная операция конкатенации (умножения);

– коммутативная операция формальной суммы (вместе с этими операциями алфавит образует полукольцо);

– коммутативная операция умножения мономов на числа (обычно достаточно рассматривать действительные числа, но в некоторых ситуациях нужны комплексные числа) [12, 52, 58, 72].

Теперь можно рассматривать символьные многочлены и формальные степенные ряды (ФСР) с числовыми коэффициентами. Кс–языком называется такой ФСР, членами которого являются все корректные мономы, которые порождены данной грамматикой [12, 72]. Таким образом, язык представляет собой совокупность всех возможных правильных предложений, порождённых его грамматикой.

Следует отметить, что интерес представляют и другие классы формальных языков, некоторые из которых связаны с классом кс–языков [20, 22, 29, 35, 36, 37, 67, 68], например, языки Ламбека [73, 74]. Большой интерес представляют не только вопросы синтаксиса, но и семантики различных классов языков [39, 40].

Возвращаясь к кс–языкам, отметим, что в имеется эффективный инструмент их исследования — система полиномиальных уравнений специального вида от символов алфавита, называемая (собственной) системой уравнений Хомского — Шютценберже. А именно, система уравнений Хомского — Шютценберже имеет вид

$$z_j = Q_j(z, x),\ j = 1, \ldots, n, \tag{2.1}$$

где правые части удовлетворяют естественным требованиям: $Q_j(0, 0) = = 0$, и многочлены $Q_j(z, 0)$ не содержат линейных членов. Последнее условие означает, что правила грамматики не предусматривают простого переобозначения

нетерминальных символов) [9, 8, 10, 11, 12, 72].

Предполагается, что указанная система решается относительно нетерминальных символов $(z_1, \ldots, z_n) = z$ в виде ФСР от терминальных символов $(x_1, \ldots, x_m) = x$:

$$z = z(x) = (z_1(x), \ldots, z_n(x)).$$

Поскольку символ z_1 играет особую роль как начальный символ в данном языке, первый ФСР $z_1(x)$ и есть кс–язык, который порождён грамматикой, записанной в виде системы уравнений Хомского — Шютценберже [10, 11, 12, 72].

Из начального символа z_1 при помощи правил подстановки (продукций) выводятся все корректные мономы языка, интерпретируемые применительно к естественным языкам как правильные предложения.

Как уже отмечалось, классу кс–языков фактически принадлежат все языки программирования, поэтому мономы с точки зрения языков программирования являются правильно написанными программами [1, 3, 4, 5, 6, 7, 31].

Для того, чтобы сформулировать расширенную проблему синтаксического анализа, нужно уточнить структуру правых частей системы уравнений Хомского — Шютценберже.

Итак, рассмотрим грамматику кс–языка, которая является совокупностью правил подстановки (продукций):

$$z_j \to \ q_{j1}(z, x), \ \ldots \ , z_j \to \ q_{jp_j}(z, x), \ j = 1, \ \ldots \ , n, \tag{2.2}$$

где $q_{jk}(z, x)$ — мономом от некоммутативных символов алфавита с числовым коэффициентом, который равен 1.

Продукции сначала следует применять к начальному символу z_1, а затем к другим мономам неограниченное число раз и в произвольном порядке, что позволяют продуцировать новые корректные мономы от терминальных символов (при их получении дальнейший вывод заканчивается); полученные мономы образуют соответствующий кс–язык.

Теперь можно отметить, что многочлены $Q_j(z, x), \ j = 1, \ \ldots \ , n$, стоящие

в правой части системы уравнений Хомского — Шютценберже, имеют следующую структуру:

$$Q_j(z,x) = \ q_{j1}(z,x) + \ \cdots \ + q_{jp_j}(z,x),$$

$$j = 1, \ \ldots \ , n.$$

Итак, проблема синтаксического анализа мономов состоит в следующем. Выделяют две её составляющие, первая часть, называемая проблемой принадлежности или этапом синтаксического контроля, состоит в том, чтобы определить, принадлежит ли моном данному кс-языку, т. е. может ли быть получен из начального символа z_1 при помощи правил подстановки. Одновременно с решением первой части проблемы синтаксического анализа решается и вторая часть проблемы.

Она состоит в «описании синтаксической структуры правильных программ, подобно тому как производится грамматический разбор предложений в естественных языках» [12].

Такое описание понимается различными авторами по-разному, например, требуется разработать алгоритм, для того чтобы:

— установить, какие правила подстановки и сколько раз использовались при выводе этого монома, при этом порядок использования правил подстановки не имеет значения [12];

— установить, какие правила подстановки, сколько раз и в каком порядке использовались при выводе этого монома, т. е. построить хотя бы один из возможных выводов монома [3].

Кроме того, возможен подход, при котором необходимо построить сразу все возможные выводы монома, если таких несколько.

Следующее важное обстоятельство, которому исследователи уделяют большое внимание, состоит в том, чтобы разработать беступиковый (бесперебойный, безостановочный) алгоритм синтаксического анализа мономов контекстно–свободного языка.

Если алгоритм таков, что может приводить к тупикам, то он должен предусматривать возвраты с анализом определённой предыстории алгоритма, что значительно усложняет алгоритм. Однако, для произвольной кс–грамматики беступикового алгоритма синтаксического анализа не существует [12]. Таким образом, во–первых, имеющиеся алгоритмы достаточно сложные, во–вторых, могут решать ограниченные задачи, например, нахождения одного из возможных выводов монома.

Далее, будем называть расширенной проблемой синтаксического анализа мономов кс–языка проблему разработки беступикового алгоритма, который позволяет установить, может ли быть выведен моном при помощи системы продукций кс–языка (решить проблему принадлежности), а также найти сразу все выводы этого монома.

Подчеркнём, что такие алгоритмы в настоящее время не известны, поскольку для произвольных кс–грамматик разработанные ранее алгоритмы могут приводить к тупикам.

Замечание 2.1. Вывод монома можно представить следующим образом: определить, какие продукции, сколько раз и в каком порядке применяются для вывода этого монома — именно в таком виде будем искать возможные выводы. Нахождение выводов в таком виде, очевидно, равносильно построению деревьев выводов.

Замечание 2.2. Очевидно, без знания порядка применения продукций невозможно получить желаемый моном, т. к. применение двух продукций к моному в разном порядке может привести к разным мономам.

Действительно, имеет место следующий пример.

Пример 2.1. Пусть даны продукции

$$z_1 \to z_1 z_2,$$

$$z_2 \to z_2 z_1$$

и моном $z_2 z_1$.

Применяя к этому моному первую продукцию, а затем вторую, получим моном

$$z_1 z_2 z_1 z_2 z_1.$$

Если сначала применить вторую продукцию, а затем первую, получится другой моном:

$$z_1 z_2 z_2 z_1 z_2.$$

Таким образом, значительное внимание следует обратить на установление порядка применения продукций при выводе монома.

Как итог, окончательно сформулируем расширенную проблему синтаксического анализа мономов кс–языка: *разработать беступиковый алгоритм, который позволяет установить, может ли быть выведен данный моном при помощи системы продукций заданного кс–языка (решить проблему принадлежности), а также найти сразу все выводы этого монома, указав, какие продукции, сколько раз и в каком порядке применяются для вывода этого монома.*

Заметим, что любые новые конструктивные алгоритмы синтаксического анализа представляют интерес для практического либо теоретического программирования.

Решение расширенной проблемы синтаксического анализа будет рассмотрено ниже.

2.2 Алгоритм решения расширенной проблемы синтаксического анализа с использованием маркированных скобок

Будем опираться на метод мономиальных меток, предложенный в работе [42]. Достоинством этого метода является то, что он является беступиковым и позволяет провести синтаксический анализ монома от терминальных символов $x_1, \ldots, x_m$ для произвольного кс–языка. Однако, к недостаткам этого метода относится то, что он, позволяя установить, какие продукции и сколько раз следует использовать для вывода монома, не даёт информации о том, в каком порядке применяются эти продукции.

В дальнейшем метод мономиальных меток был использован в работе для привлечения в теорию программирования комплексно — аналитических методов [15]. Впервые идея применения аналитических методов в теории кс–языков была выдвинута в работе [44]. Впоследствии аналитические методы использовались в том числе при решении проблемы синтаксического анализа кс–языков [14, 16, 17].

Суть метода мономиальных меток такова. Во–первых, для каждой продукции из кс–грамматики в моном из левой части вводится метка, которая несёт информацию об использовании этой продукции и, во–вторых, организуется алгоритмически простой процесс вывода всех мономов кс–языка вплоть до нужной степени с помощью метода последовательных приближений, который решает систему уравнений Хомского — Шютценберже, соответствующую данной кс–грамматике.

Сначала более подробно запишем, в чём состоит метод мономиальных меток, а затем модифицируем его для решения расширенной проблемы синтаксического анализа.

В каждое правило подстановки

$$z_j \to q_{jk}(z, x)$$

вводится мономиальная метка, которая является символом из расширенного алфавита, и получается правило

$$z_j \to t_{jk} q_{jk}(z, x),$$

совокупность которых образует новую кс–грамматику.

Далее, для новых правил подстановки (продукций) рассматривается соответствующая ей система уравнений Хомского — Шютценберже:

$$z_j = Q_j^*(z, x, t) \stackrel{\text{def}}{=} t_{j1} q_{j1}(z, x) + \ \cdots \ + t_{jp_j} q_{jp_j}(z, x), \tag{2.3}$$

$$j = 1, \ \ldots \ , n.$$

Систему уравнений (2.3) можно решать методом последовательных приближений:

$$z^{(k+1)}(x, t) = Q_j^*(z^{(k)}(x, t), x, t);$$

$$k = 0, 1, \ldots;$$

$$z^{(0)}(x, t) = 0;$$

здесь понимается, что

$$z^{(k)}(x, t) = (z_1^{(k)}(x, t), \ldots, z_n^{(k)}(x, t)).$$

В этом процессе решение получается в виде ФСР

$$z_j = z_j^*(x, t) = \sum_{i=0}^{\infty} \langle z_j^*, w_i \rangle \ w_i; \ j = 1, \ldots, n, \tag{2.4}$$

где w_i — мономы от терминальных символов и мономиальных меток (символов расширенного алфавита):

$$x_1, \ldots, x_m, t_{11}, t_{12}, \ldots, t_{np_n}$$

.

Теперь очевидно, что заменив каждый символ t_{jk} пустой цепочкой e, получим равенство решений исходной системы уравнений (2.2) и расширенной системы уравнений (2.3):

$$z_j^*(x, e) = z_j(x); \ j = 1, \ldots, n. \tag{2.5}$$

Понятно, что итерации метода последовательных приближений для расширенной системы уравнений (2.3) будут порождать многочлены относительно всех символов

$$x_1, \ldots, x_m, t_{11}, t_{12}, \ldots, t_{np_n},$$

причём степени этих многочленов относительно терминальных символов

$$x_1, \ldots, x_m$$

будут возрастать.

Важно отметить, что при возрастании числа итераций k начальные мономы получаемых многочленов

$$z_j^{(k)}(x,t), j = 1, \ldots, n,$$

стабилизируются, не меняясь при росте числа k, начиная с некоторого номера, зависящего от выбранной степени начальных членов.

Таким образом, при возрастании числа итераций мономы постепенно стабилизируются до всё более высоких степеней, образуя начальный отрезок членов ФСР (2.4), который решает систему уравнений (2.3). Это значит, что можно получить начальные члены решения системы уравнений вплоть до любой, сколь угодно большой степени, в том числе начальный отрезок ФСР, который представляет первую компоненту этого решения, т. е. непосредственно кс–язык:

$$z_1 = z_1^*(x,t) = \sum_{i=0}^{\infty} \langle z_j^*, w_i \rangle \; w_i \; . \tag{2.6}$$

Теперь синтаксический анализ монома v кс–языка $z_1(x)$ можно провести таким образом. Считывав полученные мономы степени, равной степени данного монома $\deg_x(v)$ относительно символов $x_1, \ \ldots \ , x_m$ и пропустив при этом символы мономиальных меток t_{jk}, можно выяснить, присутствует ли среди них моном v, значит, можно ли осуществить его вывод при помощи системы продукций (2.1).

Легко видеть, что каждая мономиальная метка t_{jk}, содержащаяся в таком мономе, покажет, что при выводе его была использована продукция (правило

вывода)

$$z_j \to t_{jk}\ q_{jk}(z, x)$$

. Действительно, как видно из системы уравнений (2.3) и метода последовательных приближений, применение этой продукции к некоторому моному приводит к умножению его слева на символ t_{jk}.

Таким образом, метод мономиальных меток решает более узкую, ограниченную проблему синтаксического анализа мономов кс–языка, выясняя, какие правила вывода кс–грамматики и сколько раз были использованы при получении этого монома, с точностью до порядка их применения [42].

А именно, метод мономиальных меток за конечное число шагов позволяет реализовать беступиковый синтаксический анализ любого монома (интерпретируемого как программа) кс–языка, заданного порождающей кс–грамматикой (2.1), при этом порядок применения продукций не выясняется, а значит, информация о дереве вывода монома v восстанавливается не полностью.

В связи с этим ниже предлагается модифицировать метод мономиальных меток так, чтобы он позволил решать расширенную проблему синтаксического анализа.

Замечание 2.3. Весьма полезно было бы иметь оценку числа итераций, обеспечивающих стабилизацию начальных членов многочленов, которые получаются в результате этих итераций и совпадают с начальными членами ФСР, которые представляют решение системы уравнений Хомского — Шютценберже, ассоциированной с кс–грамматикой.

Имеет место следующая лемма.

Лемма 2.1. *Пусть $k_2 > k_1$, тогда мономы степени не выше k_1 по переменным x многочленов*

$$z_j^{(k_2)}(x, t),\ j = 1, \ldots, n,$$

не зависят от k_2.

Таким образом, оценка состоит в том, что мономы многочленов

$z_j^{(k)}(x,t),\ j=1,\ldots,n$, стабилизировались вплоть до степени k включительно, уже не меняясь при дальнейшем росте числа итераций.

Доказательство. Лемма 2.1 следует из равенства $z_1^*(x,e)=z_1(x)$, в котором параметры t заменены на пустую цепочку, а также оценки, данной в работе [Хомс–Шют, с. 205].

Теперь модифицируем метод мономиальных меток, расширяя его возможности так, чтобы можно было установить порядок применения продукций.

А именно, «расширим» грамматику с мономиальными метками и рассмотрим следующую грамматику:

$$z_j \to [\ t_{jk}\ q_{jk}(z,x)\ ]_{jk},\ j=1,\ldots,n,\ k=1,\ldots,p_j, \tag{2.7}$$

где $]_{jk}$ — закрывающаяся маркированная скобка, и $[$ — открывающаяся скобка, парная к скобке $]_{jk}$. Открывающуюся скобку можно не маркировать, поскольку для каждой закрывающейся скобки всегда можно однозначно найти открывающуюся скобку, парную к ней.

Можно написать простые алгоритмы, которые позволяют для каждой закрывающейся скобки найти парную к ней открывающуюся скобку (и наоборот), при условии, конечно, что выражение содержащее скобки, составлено корректно.

Лемма 2.2. *Если выражение, содержащее скобки, составлено корректно, то алгоритм, основанный на вычёркивании ближайших к друг другу парных открывающихся и закрывающихся скобок, позволяет для каждой закрывающейся скобки найти единственную парную открывающуюся скобку, а в случае некорректности выражения, установить это.*

Доказательство. Сначала заметим, что можно проверить корректность расстановки скобок следующим образом.

Найдём самую левую (стоящую левее других) закрывающуюся скобку, затем найдём ближайшую к ней слева открывающуюся скобку. Очевидно, что эти скобки — парные и расставлены корректно; вычеркнем их.

Теперь снова найдём самую левую закрывающуюся скобку и для неё вновь ближайшую слева открывающуюся парную скобку. Если после вычёркивания всех парных скобок в выражении больше не останется скобок, они были расставлены корректно.

При этом легко видеть, как в корректном выражении для каждой закрывающейся скобки единственным образом находится открывающая скобка.

Теперь на примере продемонстрируем идею, как маркированные скобки позволяют установить порядок применения продукций при выводе монома.

Пример 2.2. Рассмотрим продукции

$$z_1 \to z_1 z_2^3,$$

$$z_1 \to z_1 z_2$$

и запишем их в виде расширенной грамматики:

$$z_1 \to [t_{11} z_1 z_2^3]_{11},$$

$$z_1 \to [t_{12} z_1 z_2]_{12}.$$

Применяя к начальному символу первую продукцию, а затем вторую, получим моном

$$[t_{11}[t_{12} z_1 z_2]_{12} z_2^3]_{11}.$$

Теперь можно видеть, что определяет порядок применения продукций. А именно, внешние скобки указывают на то, что продукция с мономиальной меткой t_{11} применена первой, и внутренние скобки указывают на то, что продукция с мономиальной меткой t_{12} применяется после того.

Таким образом, синтаксический анализ монома v кс–языка, устанавливающий порядок применения продукций, проводится расширенным методом мономиальных меток. Рассмотрим расширенную систему расширенную систему уравнений Хомского — Шютценберже:

$$z_j = Q_j^{**}(z, x, t) \stackrel{\text{def}}{=} [t_{j1} q_{j1}(z, x)]_{j1} + \ \cdots \ + [t_{jp_j} q_{jp_j}(z, x)]_{jp_j}, \qquad (2.8)$$

$$j = 1, \ \dots \ , n.$$

Далее, как и в методе мономиальных меток, решение этой системы может быть получено методом последовательных приближений:

$$z^{(k+1)}(x,t) = Q_j^{**}(z^{(k)}(x,t), x, t); \ k = 0, 1, \dots; \ z^{(0)}(x,t) = 0. \qquad (2.9)$$

Теперь итерации метода последовательных приближений для расширенной системы уравнений Хомского — Шютценберже дают стабилизирующие полиномы возрастающей степени относительно символов x, t.

Считывая мономы степени $\deg_x(v)$ ряда $z_1(x)$ относительно символов $x_1, \dots, x_m$ и пропуская символы t_{jk}, можно определить, существует ли среди них моном v, и какие продукции его генерируют, а иерархия маркированных скобок указывает порядок их применения.

Наконец, сформулируем основной результат в виде следующей теоремы.

Теорема 2.1. *Расширенный метод мономиальных меток, основанный на расширенной системе уравнений Хомского — Шютценберже (2.8), позволяет за конечное число шагов осуществить беступиковый синтаксический анализ, с учётом порядка применения продукций, любого монома кс–языка, порождённого кс–грамматикой (2.2).*

Фактически, переход от грамматики (2.2) к расширенной грамматике (2.7) позволяет включать в процесс вывода мономов информацию как о продукциях, так и порядке их применения.

Замечание 2.4. Однако, порядок применения продукций не всегда может быть определён однозначно. Рассмотрим иерархию маркированных скобок в следующей ситуации.

Пример 2.3. Пусть даны продукции

$$z_1 \to z_1 z_2,$$

$$z_1 \to x_1,$$

$$z_2 \to x_2$$

и моном x_1x_2.

Запишем продукции в виде расширенной грамматики, содержащей мономиальные метки и маркированные скобки:

$$z_1 \to [t_{11}z_1z_2]_{11},$$

$$z_1 \to [t_{12}x_1]_{12},$$

$$z_2 \to [t_{21}x_2]_{21},$$

и заметим, что моном x_1x_2 получается путем последовательного применения продукций с мономиальными метками t_{11}, t_{12}, t_{21} или продукций с мономиальными метками t_{11}, t_{21}, t_{12}.

В обоих случаях имеется вывод в терминах расширенной грамматики:

$$z_1 \to [\ t_{11}z_1z_2\ ]_{11} \to [\ t_{11}[t_{12}x_1\ ]_{12}[\ t_{21}x_2\ ]_{21}\ ]_{11}.$$

Последний моном показывает, что продукция, которая соответствует внешним скобкам, используется первой, а продукции, которые соответствуют внутренним скобкам, используются позже, однако, между ними нет иерархии, они не подчинены друг другу и поэтому могут использоваться в любом порядке.

Как итог данного раздела, опишем алгоритм расширенного синтаксического анализа монома на основе изложенного метода, который назовём *методом иерархии маркированных скобок.*

Далее рассмотрим задачу оценки сложности (количества операций) для разбора монома с учётом порядка применения продукций.

2.3 Оценка сложности алгоритма синтаксического анализа на основе иерархии маркированных скобок

Сначала заметим, что для считывания мономов удобнее маркировать при скобки иначе.

А именно, можно маркировать левую (открывающуюся) скобку мономиальной меткой, «привязанной» к ней слева. Таким образом, считаем, что сразу

Алгоритм А–6. Метод иерархии маркированных скобок

Вход: моном (программа) w степени (длины) N.

Выход: преобразованный моном.

1: Проводим N итераций метода последовательных приближений (2.9) для решения соответствующей расширенной системы уравнений Хомского–Шютценберже.

2: Перебираем все полученные в п. 1 мономы степени N, определяя те мономы, которые, с точностью до множителей t_{jk}, совпадают с мономом w (множители t_{jk}, как отмечено выше, пропускаются).

3: Считываем все найденные в п. 2 мономы слева направо, устанавливая иерархию маркированных скобок (по признаку сравнения скобок — внутренняя или внешняя скобка) и определяя тем самым порядок применения продукций при выводе монома w, т. е. все возможные способы вывода монома.

за мономиальной меткой продукции t_{jk} должна следовать привязанная к ней открывающаяся скобка. По лемме 2.2, для открывающейся скобки всегда можно найти соответствующую закрывающуюся скобку. Таким образом, по метке, маркирующей стоящую справа от неё скобку, можно устанавливать иерархию скобок.

Для этого рассмотрим соответствующую расширенную грамматику

$$z_j \to t_{jk}[\ q_{jk}(z,x)],\ j=1,\dots,n,\ k=1,\dots,p_j$$

с учётом того, что метка t_{jk} привязана к стоящей справа скобке и помечает правило вывода $z_j \to t_{jk}q_{jk}(z,x)$. Такая расширенная грамматика позволяет определить порядок применения продукций.

В самом деле, соответствующая расширенная система уравнений Хомского — Шютценберже имеет вид

$$z_j = Q_j^{***}(z,x,t) \stackrel{\text{def}}{=} t_{j1}[\ q_{j1}(z,x)\ ] + \ \cdots\ + t_{jp_j}[\ q_{jp_j}(z,x)\ ], \qquad (2.10)$$

$$j = 1,\ \dots\ , n$$

Решение этой системы можно получить методом последовательных приближений, который предусматривает, что итерации, генерирующие начальные члены ФСР решения системы, осуществляются по формуле

$$z_j^{(k+1)}(x,t) = Q_j^{***}(z^{(k)}(x,t),x,t);\ k=0,1,\dots;\ z^{(0)}(x,t)=0, \qquad (2.11)$$

$$j = 1, \ldots, n.$$

Далее начальные члены этого ряда анализируются в соответствии с алгоритмом, описанным в конце раздела 2.2. При этом, «внутренние» скобки соответствуют продукциям, которые использованы позже.

Таким образом, алгоритм методом иерархии маркированных скобок можно использовать как применительно к системе уравнений (2.8), так и системе (2.10) на основе метода последовательных приближений по формулам (2.9) или (2.10) соотвественно.

Сложность (число операций) данного алгоритма обозначим K. Очевидно, что

$$K = K_1 + K_2 + K_3,$$

где K_1, K_2, K_3 — число операций, предусмотренных пунктами $1, 2, 3$ этого алгоритма соответственно.

Рассмотрим число K. По формуле (2.11), одна итерация метода последовательных приближений состоит в подстановке многочлена в многочлен.

Имеет место следующая лемма.

Лемма 2.3. *Число операций при подстановке многочлена степени r в фиксированный многочлен равно $O(S_r)$, где S_r — число мономов в подставляемом многочлене степени r.*

Доказательство. Достаточно оценить число операций при подстановке многочлена степени r в моном; легко видеть, что это число равно S_r, что и доказывает лемму 2.3.

Теперь имеет место следующая лемма.

Лемма 2.4. *Имеет место оценка*

$$K_1 = O(NS_r).$$

Доказательство. Для получения оценки достаточно умножить оценку из леммы 2.3 на число итераций N.

Оценим далее числа K_2 и K_3. Понятно, что оценка для этих чисел одинаковая; она определяется необходимостью прочитать каждый полученный моном слева направо. Такая оценка даётся следующей леммой.

Лемма 2.5. *Имеют место оценки*

$$K_2 = O(NS_r),\ K_3 = O(NS_r).$$

Доказательство. Очевидно, что каждая оценка равна произведению числа мономов степени N на длину монома N.

Таким образом, сложность алгоритма синтаксического анализа методом иерархии маркированных скобок описывается следующей леммой.

Лемма 2.6. *Сложность алгоритма, реализующего метод синтаксического анализа из теоремы 2.1, равна* $O(NS_r)$.

Рассмотрим теперь оценку для числа S_r применительно к многочленам, участвующим в методе последовательных приближений (2.11), а именно, рассмотрим многочлен

$$z_1^{(N)}(x,t) = Q_1^{***}(z^{(N-1)}(x,t), x, t),$$

определяющий начальные мономы порождаемого кс–языка.

Понятно, что в качестве многочленов Q_j достаточно рассмотреть многочлены вида

$$z^d + h(x),$$

а значит, достаточно оценить число мономов многочлена

$$(z^{(N-1)}(x,t))^d.$$

Очевидно, что при возведении суммы g слагаемых в квадрат получается не более g^2 слагаемых (каждое слагаемое умножается на каждое), а при возведении в степень d получается не более g^d слагаемых. Далее, при возведении полученной суммы снова в степень d получится $g^d \cdot g = g^d = g^{d^2}$ слагаемых.

Наконец, при выполнении N итераций число слагаемых может возрастать до g^{d^N}.

Это значит, что число S_r из лемм 2.4 — 2.6 фактически имеет найденный порядок:

$$S_r = O(g^{d^N}).$$

Как результат, сформулируем полученную теорему.

Теорема 2.2. *Сложность алгоритма, основанного на расширенном методе мономиальных меток, основанном на расширенной системе уравнений Хомского — Шютценберже (2.8), равна*

$$O(Ng^{d^N}).$$

Таким образом, сложность данного метода достаточно высока. Следует отметить, что эта же сложность равна сложности метода мономиальных меток, предложенного К.В. Сафоновым для решения более ограниченной задачи синтаксического анализа [43].

С одной стороны, большое число операций актуализирует поиск других методов синтаксического анализа, например, метод интегрального представления синтаксического многочлена и др.

С другой стороны, алгоритм, основанный на иерархии маркированных скобок, является конструктивным алгоритмом, который весьма прост.

Кроме того, данный алгоритм является беступиковым (безвозвратным, безостановочным) алгоритмом, в отличие от известных ранее алгоритмов.

Наконец, отметим, что предложенный алгоритм особенно эффективен в ситуации, когда длина исследуемого монома (программы) не слишком большая.

Заключение

В результате проделанной работы получены следующие результаты:

1. Разработан новый эффективный алгоритм, позволяющий вычислять кратчайшие маршруты между вершинами графа Кэли $Cay(G, X)$, заданного произвольной конечной группой подстановок $G = \langle X \rangle$. Обоснована корректность представленного алгоритма, даны оценки его сложности. В этом случае, для произвольного слова $w \in X^*$, проблема поиска минимального слова разрешается за время $T = O(|w|)$.

2. При помощи МВС вычислены ранее неизвестные характеристики графов Кэли модифицированной пузырьковой сортировки S_n для случаев $n = 14$ и 15. Получены новые результаты о некоторых графах Кэли, заданных конечными двупорождёнными группами периода 7.

3. Впервые разработан беступиковый алгоритм синтаксического анализа, использующий иерархию маркированных скобок и позволяющий решать расширенную проблему синтаксического анализа, включая нахождение всех выводов монома (программы), имеющий применения в условиях разработки языков программирования для МВС.

4. Получена оценка сложности алгоритма синтаксического анализа, использующего иерархию маркированных скобок, особенно эффективного при разработке новых языков программирования.

Таким образом, решены все поставленные в диссертации задачи, цель работы достигнута.

Литература

[1] Адаменко А. Н., Кучуков А. М. Логическое управление программирование и Visual Prolog. — СПб.: БХВ–Петербург. 2003. 992 с.

[2] Алексеева А. Г. Применение итераций конечных языков в алгоритмических задачах теории формальных языков: дисс. ... канд. физ. – матем. наук. — 2012. Тольятти. 123 с.

[3] Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. — М.: Мир. 1978. 352 с.

[4] Ахо А., Сети Р., Ульман Дж. Компиляторы. Принципы, технологии, инструменты. — М.: Вильямс. 2008. 135 с.

[5] Бауэр Ф. Л., Гооз Г. Информатика. Т. 1. — М: Мир. 1990. 324 с.

[6] Бауэр Ф. Л., Гооз Г. Информатика. Т. 2. — М: Мир. 1990. 410 с.

[7] Вирт Н. Алгоритм + структуры данных = программы. — М.: Мир. 1985. 342 с.

[8] Гинзбург С. Математическая теория контекстно-свободных языков. — М. Мир. 1970. 325 с.

[9] Гинзбург С., Райс Х. Два класса языков типа АЛГОЛ // Кибернетический сборник. Нов. серия. Вып. 6. 1969. С. 114—145.

[10] Гладкий А. В. Некоторые алгоритмические проблемы для КС-грамматик // Алгебра и логика. — 1965. Т. 4. Вып. 1. С. 3—13.

[11] Гладкий А. В. Формальные грамматики и языки. — М.: Наука. 1973. 337 с.

[12] Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л. Алгебра, языки, программирование. — Киев: Наукова думка. 1974. 328 с.

[13] Демьянков В. З. Универсальная грамматика. Краткий словарь когнитивных терминов / Кубрякова Е.С., Демьянков В.З., Панкрац Ю.Г., Лузина Л. Г. / Под общ. ред. Е.С. Кубряковой. — М.: МГУ. 1997. 245 с.

[14] Егорушкин О.И., Колбасина И.В., Сафонов К.В. Синтаксический анализ программ методом интегральных представлений / Материалы 17 Всероссийской конференции «Сибирская научная школа-семинар с международным участием "Компьютерная безопасность и криптография"» — SIBECRYPT'18. Абакан. — 2018 // Прикладная дискретная математика. Приложение. — 2018. № 11. С. 128—130.

[15] Егорушкин О. И., Колбасина И. В., Сафонов К. В. О применении многомерного комплексного анализа в теории формальных языков и грамматик // Прикладная дискретная математика. — 2017. Т. 36. № 2. С. 76—89.

[16] Егорушкин О. И., Колбасина И. В., Сафонов К. В. Интегральное представление синтаксического многочлена // Материалы XXI Международной научно–практической конференции «Решетнёвские чтения». Т. 2. Красноярск. — 2017. С. 75—76.

[17] Егорушкин О. И., Колбасина И. В., Сафонов К. В. On solvability of systems of symbolic polynomial equations // Журнал Сибирского федерального университета. Математика и физика. — 2016. Т. 9. № 2. С. 166—172.

[18] Искусственный интеллект. Книга 2. Модели и методы: Справочник / Под ред. Д. А. Поспелова. М.: Радио и связь. 1990. 304 с.

[19] Карпов Ю. Г. Теория и технология программирования. Основы построения трансляторов. — СПб.: БХВ-Петербург. 2005. 272 с.

[20] Кибрик А. Е. Проблема синтаксических отношений в универсальной грамматике. Вып. XI. М.: Прогресс. 1982. С. 5–36.

[21] Кнут Д. Искусство программирования. Том 4, А. Комбинаторные алгоритмы. Часть 1. М.: Вильямс, 2013. 960 с.

[22] Компаниец Р. И., Маньков Е. В., Филетов И. Е. Системное программирование. Основы построения трансляторов. — СПб: Коронапринт. 2000. 256 с.

[23] Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. – 2-е изд. – М.: Вильямс. – 2006.

[24] Кузнецов А. А., Кузнецова А. С. О взаимосвязи функций роста в симметрических группах с задачами комбинаторной оптимизации // Вестник СибГАУ. 2012. № 6(46). С. 93 –97.

[25] Кузнецов А. А., Кузнецова А. С. Ресурсно-эффективный алгоритм для исследования роста в конечных двупорождённых группах периода 5 // Прикладная дискретная математика. 2018. № 42. С. 94–103.

[26] Кузнецов А. А. Графы Кэли бернсайдовых групп периода 3 // Сибирские электронные математические известия. 2015. Т. 12. С. 248–254.

[27] Кузнецов А. А., Кузнецова А. С. Перспективные топологии многопроцессорных вычислительных систем, основанные на графах Кэли, заданных группами периода 4 // Вестник СибГАУ. 2016. № 3. С. 575–578.

[28] Кузнецов А. А. Об одном алгоритме вычисления функций роста в конечных двупорождённых группах периода 5 // Прикладная дискретная математика. 2016. № 3(33). С. 116–125.

[29] Кузнецов С. Л. О преобразовании контекстно–свободных грамматик в грамматики Ламбека // Труды МИАН. — 2015. Т. 290. С. 72—79.

[30] Летичевский А. А. Синтаксис и семантика формальных языков / Кибернетика. Вып. 4. — 1968. С. 71—79.

[31] Льюис Ф., Розенкранц Д., Стринз Р. Теоретические основы проектирования компиляторов. — М.: Мир. 1979. 288 с.

[32] Мартыненко Б.И. Языки трансляции. — СПб.: СПбГУ. 2004. 234 с.

[33] Наур Н. Алгоритмический язык АЛГОЛ-60 / Пересмотренное сообщение. — М.: Мир. 1965. 245 с.

[34] Опалева Э. Языки программирования и методы трансляции. — СПб.: BHV. 2005. 480 с.

[35] Пентус А. Е., Пентус М. Р. Математическая теория формальных языков: Учебное пособие. — М.: МГУ. 2004. 80 с.

[36] Пентус М. Р. Полнота синтаксического исчисления Ламбека // Фунд. и прикл. математика. — 1999. Т. 5. № 1. С. 193—219.

[37] Пентус М. Р. Исчисление Ламбека и формальные грамматики // Фунд. и прикл. математика. — 1995. Т. 1. № 3. С. 729—751.

[38] Поляков В. Н. Синтез формальных моделей языка и смысла как проблема семантической обработки естественного языка // Новости искусственного интеллекта. — 1997. № 1. С. 6—63.

[39] Поспелов Д. А. Прикладная семантика и искусственный интеллект // Программные продукты и системы. — 1996. № 3. С. 24—28.

[40] Поспелов Д. А., Осипов Г.С. Введение в прикладную семиотику // Новости искусственного интеллекта. — 2002. № 6. С. 28–35.

[41] Саватеев Ю. В. Алгоритмическая сложность фрагментов исчисления Ламбека: дисс. ... канд. физ.-мат. наук. М.: МГУ, 2009. 75 с.

[42] Сафонов К. В. О возможности вычислительного распознавания контекстно-свободных языков // Вычислительные технологии. — 2005. Т. 10. № 4. С. 91—98.

[43] Сафонов К. В. О синтаксическом анализе и свойствах формальных языков программирования // Вычислительные технологии. — 2005. Т. 10. Специальный выпуск. С. 9—15.

[44] Семёнов А. Л. Алгоритмические проблемы для степенных рядов и контекстно-свободных грамматик // Доклады АН СССР. — 1973. Т. 212. С. 50—52.

[45] Скиена С. Алгоритмы. Руководство по разработке. СПб.: БХВ–Петербург, 2014. 720 с.

[46] Тюгашев А.А. Основы программирования. Часть I. – СПб: Ун-т ИТМО. – 2016.

[47] Хантер Р. Основные концепции компиляторов. – М.: Вильямс. – 2002.

[48] Хомский Н., Шютценберже М.Н. Алгебраическая теория контекстно-свободных языков // Кибернетический сборник. Нов. серия. Вып. 3. — 1966. С. 195—242.

[49] Хомский Н., Щютценберже М. П. Алгебраическая теория контекстно-свободных языков // Кибернетический сборник. Нов. серия. Вып. 2. — 1966. С. 121—230.

[50] Хомский Н. Логические основы лингвистической теории. — Биробиджан: ИП Тривиум. 2000. 146 с.

[51] Akers S., Krishnamurthy B. A group-theoretic model for symmetric interconnection networks // IEEE Transactions on Computers. 1989. Vol. 38(4). P. 555–566.

[52] Belov A. Y., Chernyat'ev A. L. Description of normal bases of boundary algebras and factor languages of slow growth // Mathematical Notes. 2017. V. 101. № 1-2. P. 203-207.

[53] Camelo M., Papadimitriou D., Fàbrega L., Vilà P. Efficient Routing in Data Center with Underlying Cayley Graph // Proceedings of the 5th Workshop on Complex Networks CompleNet. 2014. P. 189–197.

[54] Camelo M., Papadimitriou D., Fàbrega L., Vilà P. Cayley-Graph-based Data Centers and Space Requirements of a Routing Scheme using Automata // Proceedings of the 34th International Conference on Distributed Computing Systems Workshops. 2014. P. 63–69.

[55] Doh K.G., Kim H., Schmidt D.A. Abstract LR–Parsing. – Berlin – Heidelberg: Springer. – 2011. P. 90–109.

[56] Even S. and Goldreich O. The Minimum Length Generator Sequence is NP-Hard. J. Algorithms, 1981, no. 2, pp. 311–313.

[57] Epstein D., Paterson M., Cannon J., Holt D., Levy S. and Thurston W. Word Processing in Groups. Boston: Jones and Barlett Publishers, 1992. 330 p.

[58] Give'on Y. On some properties of the free monoids with applications to automata theory // J. of Computer and System Sciences. — 1977. № 2. P. 65—70.

[59] Hall Ph. Nilpotent groups, Notes of lectures given at the Canadian Mathematical Congress 1957 Summer Seminar, in The collected works of Philip Hall, *Oxford: Clarendon Press*, (1988), 415—462.

[60] Heydemann M. Cayley graphs and interconnection networks, in Graph symmetry: algebraic methods and applications (Editors: Hahnand Sabidussi). Dordrecht: Kluwer Academic Publishers. 1997. P. 167–226.

[61] Holt D., Eick B., and O'Brien E. Handbook of Computational Group Theory. Boca Raton: Chapman & Hall/CRC Press, 2005. 514 p.

[62] Kiasari A., Sarbazi-Azad H. Analytic performance comparison of hypercubes and star graphs with implementation constraints // Journal of Computer and System Sciences. 2008. no. 6. P. 1000–1012.

[63] Kuznetsov A. A., Safonov K. V. Hall's polynomials of finite two-generator groups of exponent seven // Journal of Siberian Federal University. Mathematics and Physics. 2014. № 2. P. 186–190.

[64] Kuznetsov A. A., Safonov K. V. On applications of the Cayley graphs of some finite groups of exponent five // Journal of Siberian Federal University. Mathematics and Physics. 2018. № 11(1). P. 70–78.

[65] Loz E. New record graphs in the degree-diameter problem // Australasian Journal of Combinatorics 2008. Vol. 41. P. 63–80.

[66] Morrill G. Categorial grammar. Logical Syntax, Semantics, and Processing. — Oxford: Oxford Univ. Press. 2011. 236 p.

[67] Pentus M. A polynomial–time algorithm for Lambek grammars of bounded order // Linguistic Analysis. — 2010. V. 36. № 1—4. P. 441—471.

[68] Pentus M. Lambek calculus is NP-complete // Theor. Comput. Sci. — 2006. V. 357. P. 186—201.

[69] Pingali K., Bilardi G. A graphical model for context-free grammar parsing. – Heidelberg – Berlin: Springer. – 2015.

[70] Podolskii V. V. Circuit complexity meets ontology-based data access // Proc. 10th Intl. Comput. Sci. Symp. in Russia. — 2015. Berlin: Springer. P. 7—26. (Lect. Notes Comput. Sci. V. 9139).

[71] Ryu J., Noel E., and Tang K. Fault-tolerant Routing on Borel Cayley Graph // IEEE ICC Next Generation Networking Symposium. 2012. P. 2872–2877.

[72] Salomaa A., Soittola M. Automata-Theoretic Aspects of Formal Power Series. — N.Y.: Springer-Verlag. 1978. 288 p.

[73] Savateev Yu. Lambek grammars with one division are decidable in polynomial time // Computer Science Theory and Applications. Berlin: Springer. — 2008. P. 273—282. (Lect. Notes Comput. Sci. V. 5010).

[74] Savateev Yu. Product-free Lambek calculus is NP-complete // Ann. Pure Appl. Log. — 2012. V. 163. Iss. 7. P. 775—788.

[75] Scott E., Johnstone A. GLL parsing // Electronic Notes in Theoretical Computer Science. – V. 253 (7). – 2009.

[76] Seress A. Permutation Group Algorithms. Cambridge: Cambridge University Press, 2003. 274 p.

[77] Shamir E. A remark on discovery algorithms for grammars // Information and Control. — 1962. V. 5. P. 246—251.

[78] Schibell S., Stafford R. Processor interconnection networks and Cayley graphs // Discrete Applied Mathematics. 1992. Vol. 40. P. 337–357.

[79] Soitola M. Positive rational sequences // Theoretical Computer Science. — 1976. V. 2. P. 313—321.

[80] Shin J., Sirer E., Weatherspoon H., and Kirovski D. On the feasibility of completely wireless datacenters // EEE/ACM Transaction On Networking. 2013. Vol. 21(5). P. 1666–1679.

[81] Sims C. Computational methods in the study of permutation groups // Computational problems in abstract algebra (Pergamon Press, Oxford). 1970. P. 169–183.

[82] Sims C. Computation with finitely presented groups. Cambridge: Cambridge University Press, 1994. 628 p.

[83] Stamoulis G., Tsitsiklis J. Efficient routing Scheme for Multiple Broadcasts in Hypercubes // IEEE Trans. on Parallel and Distributed Systems. 1993. Vol. 4(7). P. 725–739.

[84] Stamoulis G., Tsitsiklis J. The Efficiency of Greedy Routing in Hypercubes and Butterflies // IEEE Transaction on Communication. 1994. Vol. 42(11). P. 3051–3061.

[85] Tang K., Arden B. Vertex-transitivity and routing for Cayley graphs in GCR representations // Proceedings of ACM Symposium on Applied Computing – SAC. 1992. P. 1180–1187.

[86] Valiant L. G. General context-free recognition in less than cubic time // J. Comp. Syst. Sci. — 1975. V. 10. P. 308—315.

[87] Verbitskaia E., Grigorev S., Avdyukhin D. Relaxed parsing of regular approximations of string-embedded languages. – Cham: Springer International Publishing. – 2016. P. 291–302.

[88] Wang L., Tang K. Topology-Based Routing for Xmesh in Wireless Sensor Networks // Lecture Notes in Electrical Engineering. 2009. Vol. 44. P. 229–239.

Работы автора по теме диссертации

Свидетельства о государственной регистрации программ для ЭВМ

[89] Кишкан В.В., Кузнецов А.А., Кузнецова А.С. Произведение элементов в двупорожденных группах периода 7, основанное на полиномах Холла // Свидетельство о регистрации программы для ЭВМ RU 2019611975, 07.02.2019. Заявка № 2018661656 от 16.10.2018.

[90] Кишкан В.В., Кузнецов А.А., Кузнецова А.С. Произведение элементов в двупорожденных группах периода 5, основанное на полиномах Холла // Свидетельство о регистрации программы для ЭВМ RU 2015663307, 15.12.2015. Заявка № 2015619899 от 20.10.2015.

Публикации, входящие в базы данных Web of Science и Scopus

[91] Kuznetsov A.A., Kuznetsova A.S., Kishkan V.V. The Cayley graphs of a centralizer of the Burnside group $B_0(2,5)$ // IOP Conference Series: Materials Science and Engineering. — 2020. DOI: 10.1088/1757-899X/822/1/012043.

[92] Kishkan V.V., Safonov K.V., Tsarev R.Yu. Syntactical analysis of context-free languages taking into account order of application of productions // Journal of Physics: Conference Series. — 2019. Vol. 1333, no. 3, 032072 (1–6). DOI:10.1088/1742-6596/1333/3/032072.

Публикации в журналах из перечня ВАК

[93] Кишкан В.В., Сафонов К.В. Беступиковый алгоритм расширенного синтаксического анализа и его приложение к языкам программирования для квантовых компьютеров // Computational Nanotechnology. 2020 Т. 20. № 2. С. 42–48.

[94] Кишкан В. В. , Кузнецов А. А. Об одном алгоритме маршрутизации на графах Кэли, порожденных группами подстановок // Сибирский журнал науки и технологий. 2020 Т. 21. № 2. С. 187–194.

[95] Kuznetsov A.A., Kishkan V.V. The Cayley graphs of finite two-generator burnside groups of exponent 7 // Сибирский журнал науки и технологий. 2018 Т. 19. № 2. С. 217–222.

[96] Кишкан В. В. , Кузнецов А. А. Исследование графов модифицированной пузырьковой сортировки на основе высокопроизводительных вычислений // Сибирский журнал науки и технологий. 2017 Т. 18. № 3. С. 525–529.

Публикации в материалах конференций

[97] Кишкан В.В., Сафонов К.В. Синтаксический анализ мономов контекстно-свободных языков с учетом порядка применения продукций // Материалы XXIII Международной научно–практической конференции «Решетнёвские чтения». Красноярск. — 2019. С. 19—20.

[98] Кузнецов А.А., Кишкан В.В. Алгоритм эффективной нумерации элементов произвольной группы подстановок // Материалы XXIII Международной научно–практической конференции «Решетнёвские чтения». Красноярск. — 2019. С. 23—24.

[99] Кузнецов А.А., Кишкан В.В. Об одном алгоритме исследования роста в конечной группе подстановок // Материалы VIII Всероссийской с международным участием научно–методической конференции, посвященной 80-летию профессора Ларина С.В. — 2019. С. 43—47.

[100] Кишкан В.В., Сафонов К.В. О новом подходе в синтаксическом анализе мономов контекстно-свободных языков // Материалы VIII Всероссийской с международным участием научно–методической конференции, посвященной 80-летию профессора Ларина С.В. — 2019. С. 54—56.

[101] Кишкан В.В., Сафонов К.В. Синтаксический анализ программ методом интегральных представлений / Материалы 18 Всероссийской конференции «Сибирская научная школа-семинар с международным участием «Компьютерная безопасность и криптография» — SIBECRYPT'19. // Прикладная дискретная математика. Приложение. — 2019. № 12. С. 194—196.

[102] Кишкан В.В., Колбасина И.В., Сафонов К.В. Синтаксический анализ контекстно-свободных языков с учетом порядка применения продукции // Материалы XXII Международной научно–практической конференции «Решетнёвские чтения». Красноярск. — 2018. Т. 2. С. 14—15.

[103] Кишкан В.В., Колбасина И.В., Сафонов К.В. О синтаксическом анализе контекстно-свободных языков // Материалы XXI Международной научно–практической конференции «Решетнёвские чтения». Красноярск. — 2017. Т. 2. С. 77—78.

[104] Кузнецов А.А., Кишкан В.В. Исследование графов модифицированной пузырьковой сортировки на основе высокопроизводительных вычислений // Материалы XXI Международной научно–практической конференции «Решетнёвские чтения». Красноярск. — 2017. С. 82—83.

Приложение

В данном дополнительном разделе приведены копии свидетельств о государственной регистрации программ для ЭВМ:

1. Кишкан В.В., Кузнецов А.А., Кузнецова А.С. Произведение элементов в двупорожденных группах периода 5, основанное на полиномах Холла // Свидетельство о регистрации программы для ЭВМ RU 2015663307, 15.12.2015. Заявка № 2015619899 от 20.10.2015.

2. Кишкан В.В., Кузнецов А.А., Кузнецова А.С. Произведение элементов в двупорожденных группах периода 7, основанное на полиномах Холла // Свидетельство о регистрации программы для ЭВМ RU 2019611975, 07.02.2019. Заявка № 2018661656 от 16.10.2018.

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015663307

Произведение элементов в двупорожденных группах периода 5, основанное на полиномах Холла

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный аэрокосмический университет имени академика М.Ф. Решетнева» (СибГАУ) (RU)***

Авторы: ***Кузнецов Александр Александрович (RU), Кузнецова Александра Сергеевна (RU), Кишкан Владимир Владимирович (RU)***

Заявка № **2015619899**
Дата поступления **20 октября 2015 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***15 декабря 2015 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

ВЕРНА
ОБЕСПЕЧЕНИЯ
ОТДЕЛ ДОКУМЕНТАЦИОННОГО ОБЕСПЕЧЕНИЯ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2019611975

ПРОИЗВЕДЕНИЕ ЭЛЕМЕНТОВ В ДВУПОРОЖДЕННЫХ ГРУППАХ ПЕРИОДА 7, ОСНОВАННОЕ НА ПОЛИНОМАХ ХОЛЛА

Правообладатель: ***Кишкан Владимир Владимирович (RU)***

Авторы: ***Кишкан Владимир Владимирович (RU), Кузнецов Александр Алексеевич (RU), Кузнецова Александра Сергеевна (RU)***

Заявка № **2018661656**

Дата поступления **16 октября 2018 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***07 февраля 2019 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев