Московский государственный университет имени М.В. Ломоносова

Университет МГУ-ППИ в Шэньчжэне

Факультет вычислительной математики и кибернетики

Направление подготовки 01.03.02 «Прикладная математика и информатика»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА НА ТЕМУ:

«Вычисление скобки Кауфмана и полинома Джонса для граф-зацеплений»

Работу выполнила

Студентка

Дудак Екатерина Сергеевна

Научный руководитель:

к.ф.-м.н. доцент,

Ильютко Виктор Петрович

Шэньчжэнь

2022

Содержание

Введение

В настоящий момент в различных научных сферах используется теория узлов: в генетике в связи с зацеплением нитей молекул ДНК, в гидродинамике в связи с изучением устойчивых вихрей, образующих узлы, в ферромагнетизме, где возникают заузленные потоки магнитных полей [О., Экскурс в теорию узлов, 2004], в химии интерес представляют молекулы, сплетенные узлами, а также в физике и механике. Помимо этого, теория узлов затрагивает разные аспекты математики, такие как: топология, теория дискриминантов, теория групп и алгебры Ли, теория мультипликативного интеграла, квантовые группы и так далее. [В.О., 2005]

Целью данной работы является ознакомление с некоторыми понятиями из теории узлов, а также вычисление таких полиномов как скобка Кауфмана и полином Джонса для граф-зацеплений.

Данная работа является актуальной, так как упрощает вычисление полиномов, а также знаком других с теорией узлов.

Практической значимостью выпускной квалификационной работы является упрощение вычислений полиномов с помощью предложенной программы.

Определены следующие задачи дипломной работы:

1. Познакомить слушателей с теорией узлов.
2. Ознакомиться с методами исследования скобки Кауфмана
3. Ознакомиться с методами исследования полинома Джонса
4. Написание программы, вычисляющей скобку Кауфмана и полином Джонса для граф-зацеплений.
5. Упростить вычисление полиномов для дальнейшей работы.

В данной работе используются общенаучные и математические методы исследования вопроса, а именно анализ поведения узлов, анализ методов вычисления полиномов, а также самостоятельное вычисление и написание программы.

Работа поделена по главам. В первой главе идет знакомство с некоторыми понятиями и проблемами, поставленными в теории узлов такими как: узел, движения Рейдемейстера, виды полиномов, виртуальные диаграммы и узлы, а также граф-зацепление. Также первая глава решает первую задачу дипломной работы.

Во второй главе происходит описание скобки Кауфмана и методы нахождения. Отдельно описана программа, которая находит данный полином

с помощью граф-зацепления и матрицы смежности. Данная глава решает вторую, четвертую и пятую задачи.

В третьей главе идет описание полинома Джонса, а также метод его нахождения с помощью граф-зацепления и числа закрученности. Описаны методы программы, вычисляющей полином Джонса. Эта глава решает задачи под номерами три, четыре и пять.

Глава 1. Теория узлов. Методы исследования узлов

Прежде всего я бы хотела рассмотреть основную теорию, а также затронуть другие методы исследования.

Теория узлов – раздел топологии, отвечающая за изучение математического узла.

Узел – это вложение окружности в трёхмерное евклидово пространство, рассматриваемое с точностью до изотопии, либо же зацепление с одной компонентой. Узлы часто описываются диаграммами и схемами, которые представляют его проекцию на плоскость. Необходимо также упомянуть представление узлов и зацеплений в виде d-диаграмм и атомов, имеющие свои преимущества.

Для решения некоторых вопросов, связанных с развязыванием узла, также используются арк-представление узла, то есть строится диаграмма, в которой используются только горизонтальные и вертикальные линии, причем горизонтальные линии находятся под вертикальными. Причем требуется, чтобы любые два края не были коллинеарными. (Kauffman L. H., 2014)

Часто при исследовании узлов нужно определить, изотопны два узла или нет. Для решения этого вопроса необходимо найти с помощью движений Рейдемейстера инварианты узлов, принимающие разные значения. Движение Рейдемейстера – это три локальных движений на диаграмме зацепления. Две диаграммы, относящиеся к одному и тому же узлу, с точностью до плоской изотопии могут быть преобразованы одна в другую с помощью последовательного применения одного движения Рейдемейстера. Всего рассматривается три вида движения Рейдемейстера:

- Тип I. Скручивание и раскручивание в любом направлении. (см. Приложение А, Тип I)
- Тип II. Перемещение одной дуги через другую. (см. Приложение А, Тип II)
- Тип III. Перемещение нити целиком над или под пересечением. (см. Приложение А, Тип III)

Замечу, что данные локальные движения позволили Курту Рейдемейстеру составить таблицу вплоть до семи перекрестков (пересечений двух веток узла). (см. Приложение Б)

Другим способом проверки изотопии является соотношение типов Конвея (skein relation), с помощью которого строится полином Джона Хортона Конвея. Этот метод основывается на комбинаторных соотношениях, описывающих взаимосвязь значений инварианта и «похожих» плоских

диаграмм. (В.О., 2005) К другим инвариантам, удовлетворяющим соотношению Конвея относятся полиномы Джонса, Кауфмана и HOMFLY. Название последнего является аббревиатурой фамилий ученых, открывших данный многочлен, а именно Jim Hoste, Adrian Ocneanu, Kenneth Millett, Peter J. Freyd, W. B. R. Lickorish и David N. Yetter. Также заслуживает упоминание работа Михаила Хованова, благодаря которому открылся новый вид инварианта узла под названием гомология Хованова.

Отдельного упоминания заслуживает инвариант узла, принимающий значения на Z_2, а именно арф-инвариант, названный в честь турецкого математика Джахита Арфа. Узлы считаются арф-инвариантными, если они считаются пасс-эквивалентными, то есть один из них может быть преобразован в другой последовательным применением изотопии и локальных преобразований. (В.О., 2005) Два локальных преобразования показаны в Приложении В.

Другим важным вопросом, часто появляющимся при рассмотрении узлов, является распознавание тривиального узла, то есть узла, имеющего нулевое зацепление – вложение дизъюнктного объединения окружностей. Для его решения также можно применять движения Рейдеместера. Однако бывают случаи, когда для решения этого вопроса приходится дополнительно запутывать узел, в таком случае узел будет считаться трудно распутываемым. Пример такого узла и метод его распутывания приведен в Приложении Г.

Также тривиальный узел не может быть раскрашен в три цвета, то есть не является раскрашиваемым. Узел считается раскрашиваемым, если каждая арка его диаграммы может быть раскрашена одним из трех цветов удовлетворяя двум требованиям:

1. В арке используется как минимум два цвета
2. На каждом перекрестке встречаются либо все три цвета, либо один и тот же. (Santi, 2002)

Одно из свойств раскраски в три цветя является то, что движения Рейдемейстера не меняет раскрашиваемость.

Замечу, однако, если узел нельзя раскрасить, это не значит, что он является тривиальным. В пример можно привести узел под названием «восмерка», который невозможно раскрасить, но он не является тривиальным. (см. Приложение В)

На данный момент существует нерешенная проблема, связанная с написанием программы, которая сможет определить, является ли узел

тривиальным. Существующие алгоритмы справляются с простыми узлами, но при повышении сложности выдают бесконечное время решения.

Первый алгоритм решения данной проблемы был предложен в 1961 году Вольфгангом Хакином в 1961 году. Идея состоит в том, чтобы разделить эвклидово пространство на множество маленьких тетраэдров так, чтобы узел состоял из их ребер. Затем необходимо посмотреть, образуют ли объединения треугольников диск с узлом в качестве границы. Такой диск существует только тогда, когда узел является тривиальным. (C. M. , 2012)

Помимо алгоритма Хакина существуют алгоритмы Хасса-Лагариаса, Бирмана-Хирша, Дынникова и так далее.

При упоминании зацепления стоит также рассказать о компоненте зацепления, которая является вложением каждой окружности зацепления.

Для построения полинома Джонса и скобки Кауфмана, необходимо определить направление так называемой нити и построить ориентированное зацепление. При рассмотрении неориентированного же зацепления мы сможем определить число закрученности только в Z_2.

Пусть в перекрестке ориентированного зацепления верхняя ветвь проходит слева направо. Такой перекресток мы будем называть положительным. В противном случае перекресток будет отрицательным. Разность количества положительных перекрестков и количества отрицательных перекрестков называется числом закрученности. Для тривиального узла число закрученности будет равняться нулю.

Отдельно стоит рассмотреть виртуальные узлы и диаграммы. Виртуальная диаграмма — это 4-валентный граф, у которого каждая вершина снабжена структурой классического или виртуального перекрестка. (Мантуров В. О., 2009) Пример виртуального перекрестка представлен в Приложении Д рисунок 7.

Две виртуальные диаграммы называются эквивалентными, если существует последовательность обобщенных движений Рейдемейстера, переводящая первую диаграмму во вторую.
Как и в классическом случае, все движения предполагаются локальными: они производятся внутри некоторой ограниченной области. Вне этой области диаграмма не изменяется. (Мантуров В. О. К. Л., 2006)

В список обобщенных движений Рейдемейстера входят:
1. Классические движения Рейдемейстера, относящиеся только к обычным перекресткам.
2. Виртуальные движения Рейдемейстера. (см. Приложение Е, рисунок 8)

3. Полувиртуальная версия третьего движения Рейдемейстера. (см. Приложение Е, рисунок 9)

Виртуальное зацепление — это класс эквивалентности виртуальных диаграмм по виртуальным движениям Рейдемейстера.

Виртуальный узел — это однокомпонентное виртуальное зацепление. (Мантуров В. О. К. Л., 2006)

Также диаграмма может быть чисто виртуальной, если все перекрестки являются виртуальными. Каждая чисто виртуальная диаграмма эквивалентна по виртуальным переходам скоплению непересекающихся кругов на плоскости. (H., Introdiction to virtual knot theory, 2012)

Также для вычисления скобки Кауфмана и полинома Джонса необходимо рассказать про полином Лорана. Полиномом Лорана является выражение типа

$$h(z) = \sum_{k=m}^{n} h_k z^{-k}$$

где m, n — целые числа, $m \leq n$, и h_k — вещественные коэффициенты. (Малозёмов В. Н., 2009)

Помимо полинома Лорана, также следует рассказать про граф-зацепления. Для этого определим несколько понятий:

Хордовая диаграмма – это ориентированную окружность, на которой проведены несколько хорд, все концы которых различны. Хордовые диаграммы рассматриваются как комбинаторный объект, то есть на окружности важно лишь взаимное расположение точек, соединяемых хордами (см. Приложение Ж). (О. М. В., Экскурс в теорию узлов, 2004) Две хорды будут зацепленными, если концы одной хорды лежат в разных связных компонентах множества, получаемых из S^1 путем удаления концов другой хорды. Иначе же хорды будут незацепленными.

Граф пересечений хордовой диаграммы D — это простой граф без петель и кратных ребер, вершины которого находятся в однозначном соответствии с хордами диаграммы D, причем две вершины соединены ребром в том и только в том случае, если соответствующие им хорды зацеплены. Данный граф будет считаться меченым, если каждая его вершина v снабжена меткой (a, α), где $a \in \{0, 1\}$ — это оснащение вершины, а $\alpha \in \{\pm\}$ — является знаком вершины. (Д. П. Ильютко, 2011)

Для определения граф-зацепления необходимо рассмотреть граф-преобразования, то есть преобразования меченых графов. Всего выделяют четыре вида:

- Первое граф-преобразование состоит в добавлении или же удалении изолированной вершины с меткой $(0, \alpha), \alpha \in \{\pm\}$.
- Второе граф-преобразование состоит в добавлении/удалении двух несмежных (соответственно, смежных) вершин с метками $(0, \pm\alpha)$ (соответственно, $(1, \pm\alpha)$), и имеющих одинаковые смежности с остальными вершинами.
- Третье граф-преобразование определяется следующим образом. Пусть u, v, w — три произвольные вершины графа G, все имеющие метки $(0, -)$, причем вершина u смежна только с вершинами v и w в G, а вершины v и w не смежны друг с другом. Тогда мы изменяем только смежности вершины u с вершинами v, w и $t \in (N(v) \setminus N(w)) \cup (N(w) \setminus N(v))$. Кроме того, необходимо поменять знаки вершин v и w на +. Замечу, что обратная операция также называется третьим граф-преобразованием.
- Четвертое граф-преобразование для графа G определяется следующим образом. Мы берем две смежные вершины u и v с метками $(0, \alpha)$ и $(0, \beta)$ соответственно. Заменяем граф G на $piv(G; u, v)$ и, кроме того, меняем знаки вершин u и v на $-\alpha$ и $-\beta$ соответственно. В этом четвертом граф-преобразовании мы берем одну вершину v с меткой $(1, \alpha)$. Заменяем G на $LC(G; v)$ и, кроме того, меняем знак вершины v и оснащение каждой вершины $u \in N(v)$. (Д. П. Ильютко, 2011)

Граф-зацепление – это класс эквивалентности простых меченых графов по четырем граф-преобразованиям. Для его получения возьмем меченые хордовые диаграммы, построенные посредством поворачивающих обходов, и движения, полученные с помощью «настоящих» движений Рейдемейстера виртуальных диаграмм. Эти движения переносим на произвольные меченые графы, используя графы пересечений хордовых диаграмм.

Граф-зацепление называется ориентированным, если на каждом его представителе задана ориентация, и для любых двух представителей G' и G'' существует такая последовательность меченых графов $G_1 = G', G_2, \ldots, G_s = G''$, что графы G_p и G_{p+1} имеют одинаковую ориентацию. (Ильютко Д. П., 2013)

Помимо этого, стоит определить реализуемость графов. Простой граф H называется реализуемым, если существует такая хордовая диаграмма D, что H = G(D). В противном случае граф называется нереализуемым. Граф-

зацепление же будет нереализуем, если каждый его представитель не является реализуемым графом.

Вывод по главе: Рассмотрена основная теория, необходимая для вычисления скобки Кауфмана и полинома Джонса. Помимо этого, данная глава знакомит нас с некоторыми проблемами и методами, требующими решения.

Глава 2. Вычисление скобки Кауфмана

Поговорим о скобке Кауфмана. Данный полином был предложен в 1987 году американским математиком Льюисом Хиршем Кауфманом. Для построения скобки Кауфмана используются полином Лорана и соотношения Конвея.

Строится данный полином следующим образом:

1. Полином тривиального узла равен 1.
2. Разделяем узлы вертикально и горизонтально, при этом у нас получаются две новые более простые проекции. Причем первая упрощенная диаграмма умножается на A, а вторая на A^{-1}.
3. Если при разделении в одной проекции появились тривиальный и не тривиальный узлы, то необходимо вычеркнуть тривиальный узел из проекции и умножить ее на $-\mathrm{A}^2-\mathrm{A}^{-2}$.

Данный полином является инвариантным относительно второго и третьего движений Рейдемейстера, но не инвариантным относительно первого движения, в следствии чего скобка Кауфмана узла будет отличаться от его зеркального отражения (см. Приложение Ж). Решить данную проблему помогает обобщенный полином Кауфмана или же по-другому полином Джонса.

Скобку Кауфмана можно раскрасить с помощью квантовых групп и идемпотентов Джонса-Вензля.

Еще одним способом нахождения скобки Кауфмана является рекурсивный метод. Он основывается на лемме и теореме о рекурсивном соотношении.

Лемма: для тривиального узла с k скручиванием скобка Кауфмана будет определяться как

$$\langle U_k \rangle = (-1)^k a^{3k}$$

Теорема (о рекурсивном соотношении): Для любого $n \geq 2$ будет верно следующее соотношение:

$$\langle \widehat{x_1^n} \rangle = (-1)^{n-1} a^{3n-2} + a^{-1} \langle \widehat{x_1^{n-1}} \rangle$$

В таком случае скобка Кауфмана для $\widehat{x_1^n}, n \geq 2$ будет

$$\langle \widehat{x_1^n} \rangle = -a^{-n-2} + \sum_{k=1}^{n-1} (-1)^{n+k-2} a^{(3n+2)-4k}$$ (Nizami A. R., 2014)

Последний метод вычисления скобки Кауфмана основывается на граф-зацеплениях. Для этого нам необходимо составить матрицу смежности используя данные из граф-зацепления. Строится она следующим образом:

1. Если $i = j$, то a_{ij} будет равно оснащению рассматриваемой вершины графа
2. Если $i \neq j$ и две рассматриваемые вершины смежны, то $a_{ij} = 1$, в противном случае $a_{ij} = 0$.

Скобка Кауфмана для меченного графа вычисляется по следующей формуле:

$$\langle G \rangle(a) = \sum_{s} a^{\alpha(s) - \beta(s)} (-a^2 - a^{-2})^{corank_{Z_2} A(G(s))},$$

где s – это состояние графа, α(s) – это количество вершин с метками (a, –) из s и (b, +) из V(G) \ s. β(s) = V(G) – α(s). Corank – это число окружностей состояния s – 1 и вычисляется как разность размерности матрицы A(G(s)) и ее ранга.

Состояние – это подмножество из V(G) для графа G.

Ниже будет дано описание программы, которая вычисляет скобку Кауфмана с помощью граф-зацепления. Ознакомиться с программой можно в Приложении И.

1. Создан класс Byte, с помощью которого будет вычисляться состояние графа, а также, в последствии, строиться матрица для вычисления Corank. Данный класс переводит число в его двоичное представление.
2. Созданы вспомогательные функции Swap и Fact (от слова факториал). Первая функция меняет местами строки, а вторая вычисляет факториал.
3. Создана функция Sum, которая вычисляет один элемент многочлена с помощью представленной выше формулой. Далее элемент записывается в строку.
4. Функция String выполняет сокращение первоначального получившегося многочлена. Сокращение происходит следующим образом:
   4.1. Для упрощения дальнейших действий, разделяются суммы многочлена и добавляются в конец.
   4.2. Идет поиск пары элементов полинома с противоположными знаком. Если такая пара существует, то она удаляется.

4.3. Элементы полинома, степень которых равняется нулю, приравниваются к единице.

4.4. Суммируются одинаковые элементы многочлена для лучшей читаемости результата.

5. Функции Alpha и Beta находят α(s) и β(s) соответственно.
6. Функция Rank необходима для поиска Corank. Данная функция ищет ранг матрицы состояния s, используя вспомогательную функцию Swap.
7. Функция Corank ищет Corank матрицы состояния s. Нахождение числа окружностей состояния описано выше.
8. Функция KauffmanBracket фактически собирает всю программу и высчитывает скобку Кауфмана. Также в ней описаны случаи, когда Corank равен нулю либо единице. После вызывает функцию JonesPolynomial.
9. В методе main задаются размерность матрицы смежности, сама матрица, а также знаки на перекрестках. Позже вызывается метод KauffmanBracket.

Дополнительные пояснения к программе:

- Для поиска состояний было решено создать дополнительный класс данных, так как приходилось постоянно перебирать перекрестки, которые будут входить в состояние и которые не будут. Данный способ показался наиболее простым для отслеживания и решения поставленных задач.
- Скобку Кауфмана было решено представить в виде строки в силу упрощения записи элементов многочлена, а также ввиду библиотеки для работы со строками, которая упрощает работу с ними.
- Первоначальная скобка Кауфмана часто получалась нечитаемой из-за того, что предыдущие методы только записывают элементы в строку. Функция String сокращает формулу в несколько раз, путем удаления одинаковых элементов. Однако данный метод очень сложно читать из-за постоянных проверок символов. Предполагаю, что функция не является оптимальной в силу работы со строками.

Результаты вычисления программой узлов трилистник и «восьмерка» совпадают с вычислениями, проведенными на бумаге классическим методом, а также с помощью граф-зацеплений. Вычисления трилистника, а также результаты выполнения программы можно увидеть в Приложении К, вычисления узла «восьмерка», а также вывод программы в Приложении Л.

Вывод по главе: вычисление скобки Кауфмана является задачей, требующей достаточно много времени, а также внимательности. Если для вычисления узла с числом пересечений три требуется около десяти-пятнадцати минут, то для узла с числом пересечений шесть может понадобиться около часа, при этом легко перепутать переменные либо упрощенный узел из-за чего полином будет построен неверно. Написанная программа упрощает вычисление даже для более сложных узлов, однако для ее использования необходимо правильно составить граф-зацепление, а также правильно расставить знаки на перекрестках.

Глава 3. Вычисление полинома Джонса

Перейдем к полиному Джонса. Впервые он был предложен в 1984 году американским математиком Воном Фредериком Рэндэлом Джонсом. Также его еще называют обобщенным полиномом Кауфмана либо полиномом Джонса-Кауфмана.

Интересен полином Джонса тем, что в случае узлов он не зависит от ориентации узла, то есть является инвариантным относительно всех трех движений Рейдемейстера, а в случае зацеплений строится для ориентированных зацеплений.

Полином Джонса составного узла также допускает разложение на произведение полиномов. Заметим, что это утверждение справедливо если полином Джонса является многочленом для каждого нетривиального узла в T × I или в S^3. (A., 2015)

До недавнего момента существование нетривиального узла в общем случае, полином Джонса которого является одночленом, было под вопросом. Его пытались найти Р. П. Ансти, Дж. Х. Пжитицкий и Д. Рольфсен (R. P. Anstee, J. H. Przytycki, D. Rolfsen) применяя к диаграмме тривиального узла преобразования, не меняющие полином Джонса, но их постигла неудача. (Dasbach O. T., 1997) На данный момент был найден нетривиальный граф-узел (то есть граф-зацепление с одной переменной), полином Джонса которого тривиален. (Ильютко Д. П., 2013) (см. Приложение М)

Полином Джонса-Кауфмана строится следующим образом:

$$(\mathrm{X})\langle G\rangle = (-a)^{-3w(G)} \sum_{s} a^{\alpha(s)-\beta(s)}(-a^2-a^{-2})^{corank_{Z_2}A(G(s))},$$

где w – это число закрученности ориентированной диаграммы. Если сделать замену переменной $a = q^{-1/2}$, то получим стандартный вид полинома Джонса.

Также существует теорема, говорящая о том, что если количество компонент ориентированного зацепления нечетно, то полином Джонса содержит члены вида $q^k, k \in Z$. Если же количество четно, то члены полинома имеют вид $q^{(2k+1)/2}, k \in Z$. (Б., 1997)

Полином Джонса также можно раскрасить с помощью квантовых групп и идемпотентов Джонса-Вензля. В таком случае можно говорить о «ортогональном соотношении между полиномом Джонса и A-полиномами». Однако при рассмотрении этого соотношения проще использовать раскрашенную скобку Кауфмана, которая отличается от раскрашенного

полинома Джонса заменой переменной $t \to it$. Данное соотношение имеет вид:

$$K_n^s(K_0) \coloneqq \sum_{i=i}^{k} c_i t^{-p_i q_i} \left\{ \left(-t^{2(n+p_i)+2}\right)^{q_i} C_{n+p_i}(K_0) + \left(-t^{2(n-p_i)+2}\right)^{-q_i} C_{n-p_i}(K_0) \right\} = 0$$

где K_0 – это оформленный узел в S^3. (F., 2004)

А-полином – это полином, который исчезает на кривой и не имеет неустранимых коэффициентов. Данный полином является уникальным с точностью до умножения на ненулевую константу. (Cooper D., 1994)

Пример построения раскрашенного полинома Джонса для трилистника:

$$J_K(n) = \frac{q^{n-1}+q^{4-4n}-q^{-n}-q^{-1-2n}}{q^{\frac{1}{2}}(q^{n-1}-q^{2-n})} J_K(n-1) + \frac{q^{4-4n}-q^{3-2n}}{q^{2-n}-q^{n-1}} J_K(n-2),$$

где $J_K(0) = 0,\ J_K(1) = 1.$ (S. Garoufalidis, 2005)

Представленная в Приложении И программа также высчитывает полином Джонса. Предлагаю описание программы:

1. Функция TwistNumber вычисляет число закрученности. Число закрученности вычисляется по данной формуле:

   $\sum_{i=1}^{n}(-1)^{corank_{Z_2}(A(G)+E+E_{ii})} signv_i$, где v_i – это вершина графа

2. Функция JonesPolynomial вычисляет полином Джонса с помощью методов KauffmanBracket и TwistNumber.

Вычисления полинома Джонса для трилистника и узла «восьмерка» также представлены в Приложении К и Приложении Л. Вывод программы совпадает с вычислениями, проведенными на бумаге.

Вывод по главе: полином Джонса достаточно легко вычислить, если знать скобку Кауфмана заранее, при этом данный полином открывает большое пространство для исследования узлов и других смежных сфер математики. Исследования, использующие полином Джонса, до сих пор продолжаются.

Заключение

В данной работы мы рассмотрели основные понятия, а также вычислили скобку Кауфмана и полином Джонса для граф-зацеплений. Работа отличается тем, что больше раскрывает некоторые проблемы, а также предоставляет возможность упростить вычисления тем, кто будет исследовать данный вопрос дальше. Задачи, поставленные в рамках дипломной работы, считаю выполненными.

Важными выводами, к которым я пришла, являются то, что в теории узлов еще достаточно много нерешенных проблем, некоторые из которых не требуют углубленных знаний в этой области. Также некоторые решенные на данный момент методы требуют внимания для написания программ, которые в дальнейшем упрощают вычисления.

Дальнейшее рассмотрение темы представляется углублением в теорию узлов, а также в труды основных деятелей данной области математики. Также есть возможность улучшить и упростить приложенную программу для нахождения скобки Кауфмана. Помимо этого, считаю, что в освещенной теме необходимо искать и улучшать некоторые методы в более узких сферах, таких как: соотношения раскрашенного полинома Джонса и A-полиномом, программы для поиска сложных тривиальных узлов и так далее. В этом может помочь предоставленная программа для более быстрого нахождения скобки Кауфмана и полинома Джонса.

Считаю, что тема раскрыта обширно и с разных сторон, предоставлены различные методы вычисления полиномов для граф-зацеплений и не только.

Библиографический список

1. Акимова А. А. Классификация виртуальных узлов рода 1 малой сложности: диссертация / А. А. Акимова - Институт математики и механики УрО РАН, Екатеринбург, 2015, 77 (А., 2015)
2. Ильютко Д. П. Граф-зацепления: нереализуемость, ориентация и полином Джонса: статья / Д. П. Ильютко, В. С. Сафина – СМФН, том 51, 2013, 33-63 (Ильютко Д. П., 2013)
3. Ильютко Д. П. Четность в теории узлов и граф-зацепления: монография / Д. П. Ильютко, В. О. Мантуров, И. М. Никонов - РУДН, Москва, 2011, 3–163 (Д. П. Ильютко, 2011)
4. Малозёмов В. Н. Полиномы Лорана: доклад / В. Н. Малозёмов, Н. А. Соловьёва, 2009, 6 (Малозёмов В. Н., 2009)
5. Мантуров В. О. Виртуальные узлы и зацепления: книга / В. О. Мантуров, Л. Х. Кауфман – Труды математического института им. В.А. Стеклова, том 252, 2006, 114–133 (Мантуров В. О. К. Л., 2006)
6. Мантуров В. О. Геометрия и комбинаторика виртуальных узлов: автореферат / В. О. Мантуров – Московский государственный областной университет, Москва, 2007, 367 (О. М. В., 2007)
7. Мантуров В. О. Граф-зацепления: статья / В. О. Мантуров, Д. П. Ильютко – Доклады академии наук, том 428, №5, 2009, 591-594 (Мантуров В. О., 2009)
8. Мантуров В. О. Комплекс Хованова для виртуальных узлов / В. О. Мантуров - Центр новых информационных технологий МГУ, Издательский дом «Открытые системы», Москва, 2005, 22 (О., Комплекс Хованова для виртуальных узлов, 2005)
9. Мантуров В. О. Теория узлов: учебник / В. О. Мантуров – Институт Компьютерных Исследований, Москва-Ижевск 2005, 512 (В.О., 2005)
10. Мантуров В. О. ЭКСКУРС В ТЕОРИЮ УЗЛОВ: журнал / В. О. Мантуров – Соросовский образовательный журнал, том 8, No1, 2004, 122-127 (О., Экскурс в теорию узлов, 2004)
11. Прасолов В. В. Наглядная топология: книга / В. В. Прасолов – МЦНМО, Москва, 1995, 111 (В., 1995)
12. Прасолов В. В. Узлы, зацепления, косы и трехмерные многообразия: книга / В. В. Прасолов, А. Б. Сосинский – МЦНМО, Москва, 1997, 352 (Б., 1997)

13. Adams C. C. The knot book: an elementary introduction to the mathematical theory of knots: учебник / C. C. Adams – American Mathematical Society, Rhode Island, 2004, 307 (C., 2004)

14. Cooper D. Plane Curves Associated to Character Varieties of 3-Manifolds: статья / D. Cooper, M. Culler, H. Gillet, D. D. Long, P. B. Shalen - Inventiones mathematica, 1994, 39 (Cooper D., 1994)

15. Dasbach O. T. Does the Jones Polynomial Detect Unknottedness?: статья / O. T. Dasbach, S. Hougardy - Experimental Mathematics, 1997, 51-56 (Dasbach O. T., 1997)

16. Garoufalidis S. The colored Jones function is q–holonomic: статья / S. Garoufalidis, Thang T Q Le - Geometry & Topology, Volume 9, 2005, 1253-1293 (S. Garoufalidis, 2005)

17. G. D. Santi An Introduction to the Theory of Knots / Giovanni De Santi, 2002 (Santi, 2002)

18. Nagasato F. Efficient formula of the colored Kauffman brackets: статья / F. Nagasato - Lobachevskii Journal of Mathematics, Vol. 16, 2004, 71-78 (F., 2004)

19. Kauffman L. H. Introduction to Virtual Knot Theory: статья / L. H. Kauffman – University of Illinois at Chicago, Chicago, 2012, 39 (H., Introdiction to virtual knot theory, 2012)

20. Kauffman L. H. Knots: статья / L. H. Kauffman - University of Illinois at Chicago, Chicago, 82 (H.)

21. Kauffman L. H. On knots: книга / L. H. Kauffman – Princeton university press, New Jersey, 1987, 480 (H., On knots, 1987)

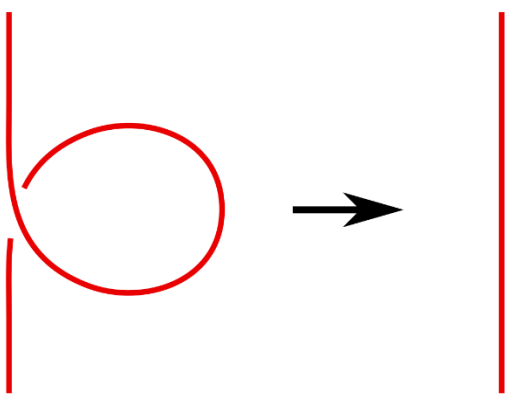
22. Kauffman L. H. Unknotting unknots: статья / L. H. Kauffman, A. Henrich – The Mathematical Association of America Vol. 121, 2014, 379-390 (Kauffman L. H., 2014)

23. Manolescu C. A Journey From Elementary to Advanced Mathematics: The Unknotting Problem: презентация / C. Manolescu – UCLA, Los Angeles, 2012, 31 (C. M. , 2012)

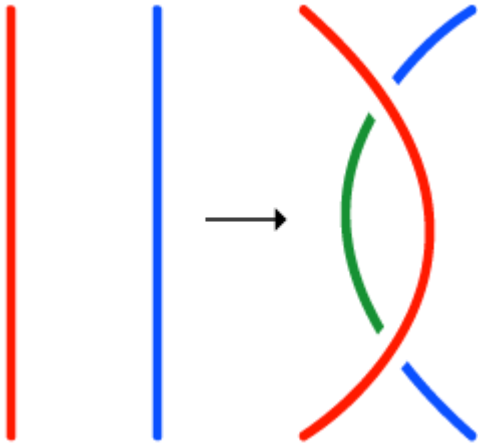
24. Nizami A. R. A Recursive Approach to the Kauffman Bracket: статья / A. R. Nizami, M. Munir, U. Saleem, A. Ramzan - University of Education, Lahore, 2014, 11 (Nizami A. R., 2014)

Приложение А

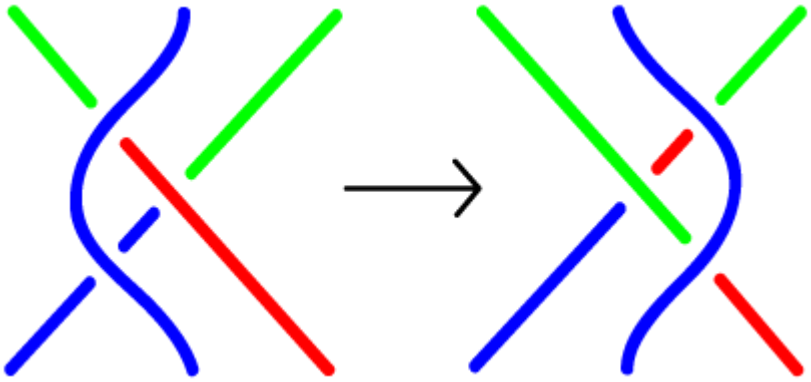
Движения Рейдемейстера и раскраска



Тип I



Тип II



Тип III

Приложение Б

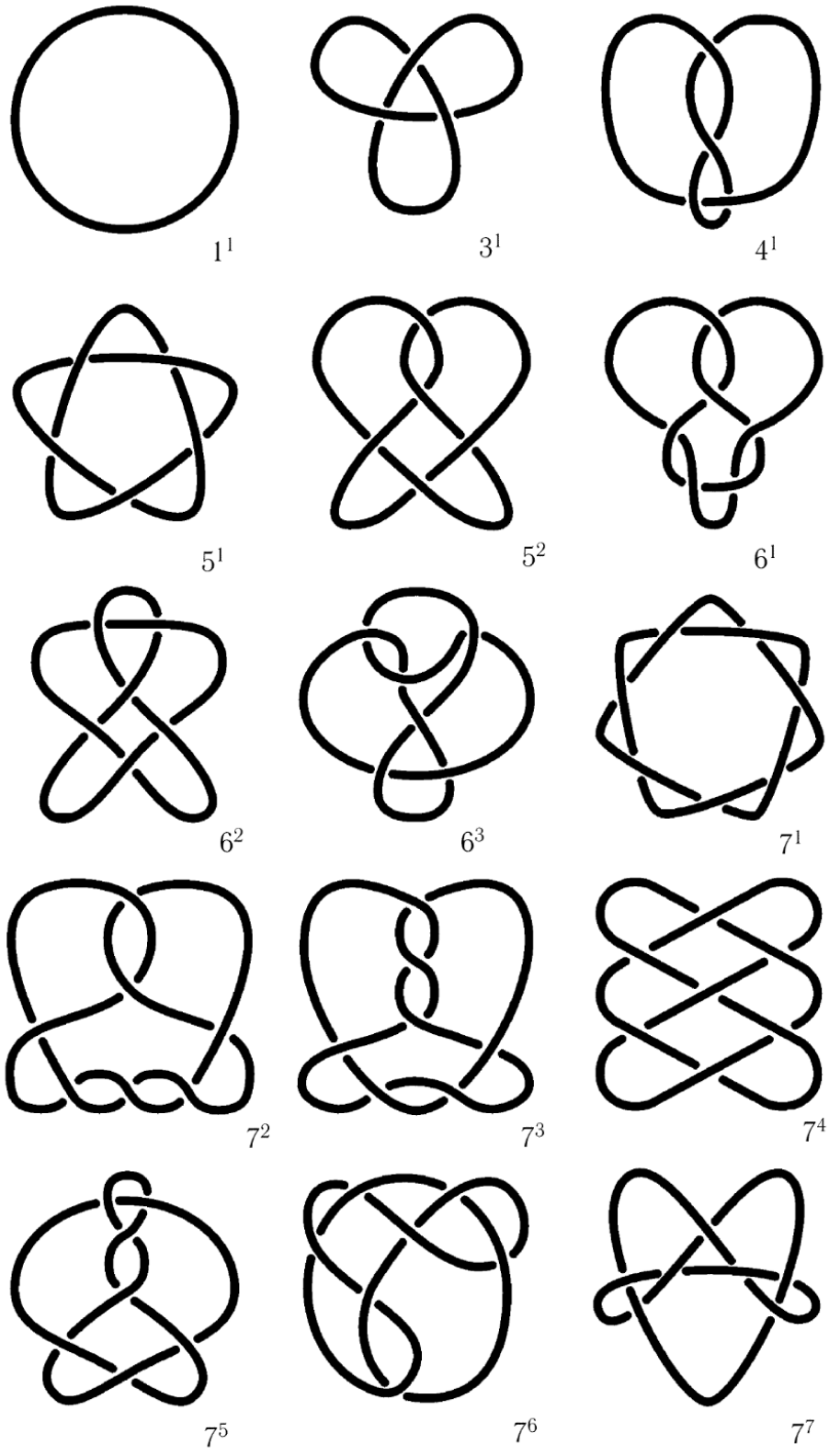


Рисунок 1. Пример таблицы простых узлов с семью или менее перекрестками

Приложение В

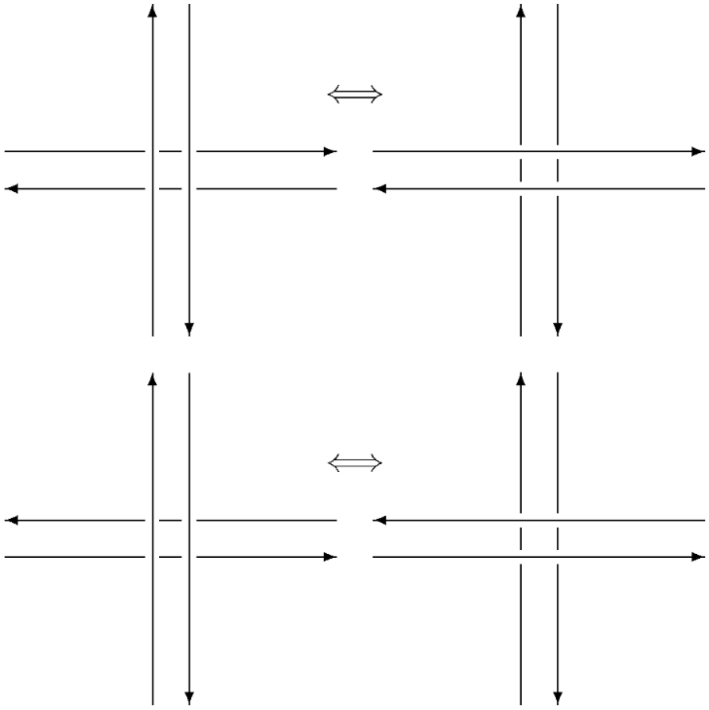


Рисунок 2. Пасс-эквивалентность

Приложение Г

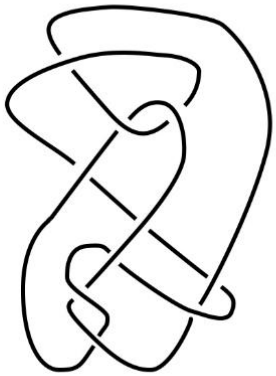


Рисунок 3. Пример трудно распутываемого узла

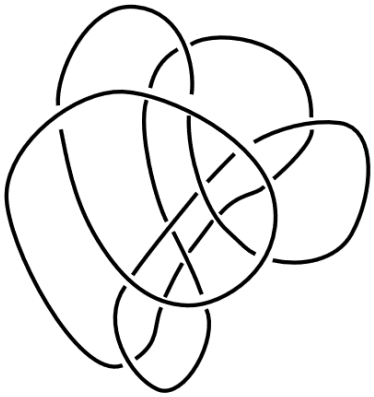


Рисунок 4. Пример простого узла

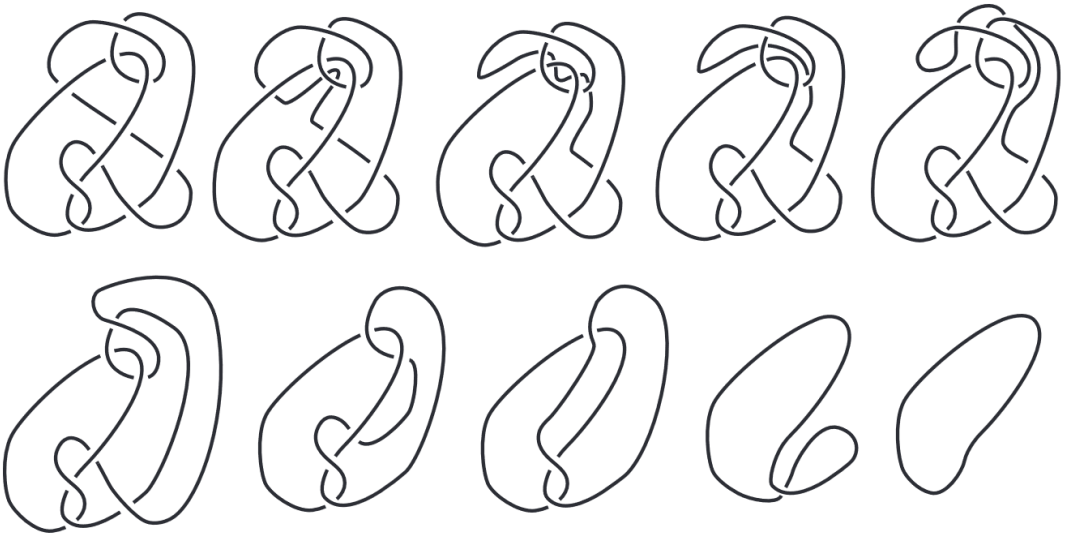


Рисунок 5. Распутывание узла

Приложение Д

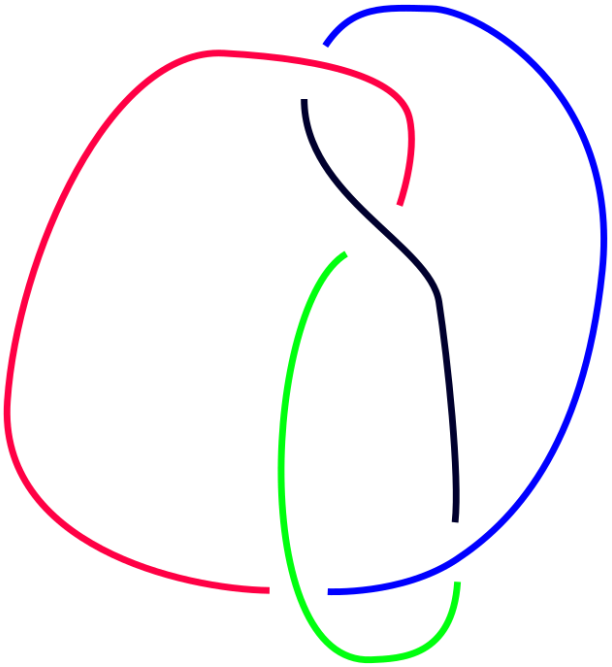


Рисунок 6. Раскраска узла "восьмерка"

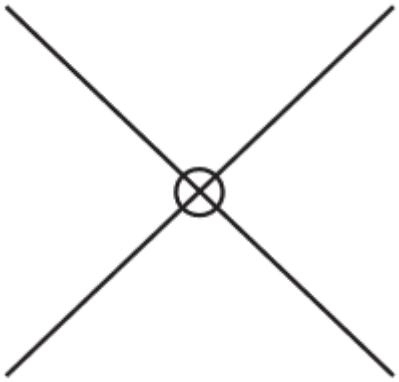


Рисунок 7. Виртуальный перекресток

Приложение Е

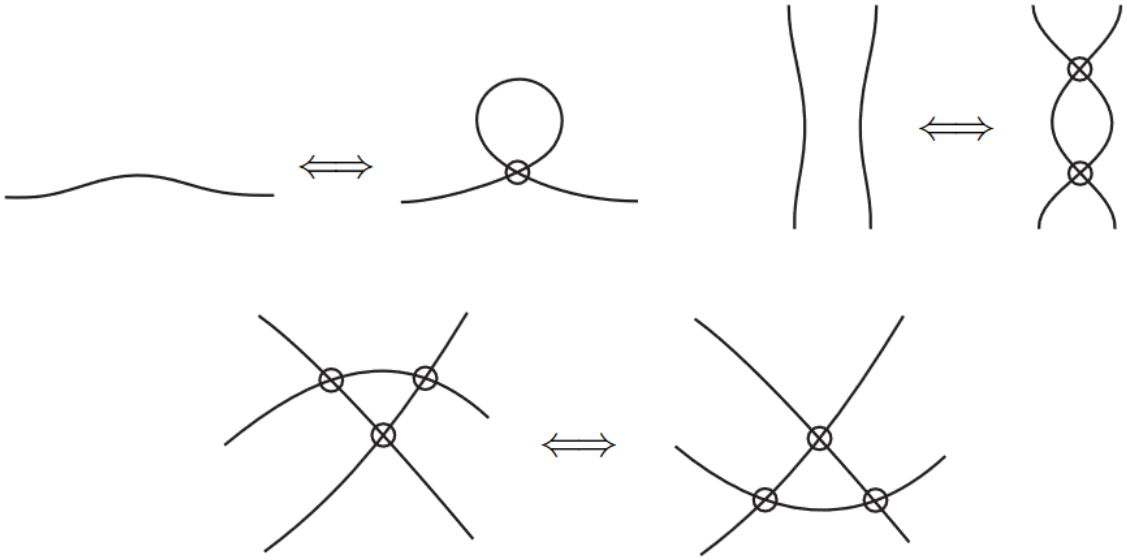


Рисунок 8. Виртуальные движения Рейдемейстера

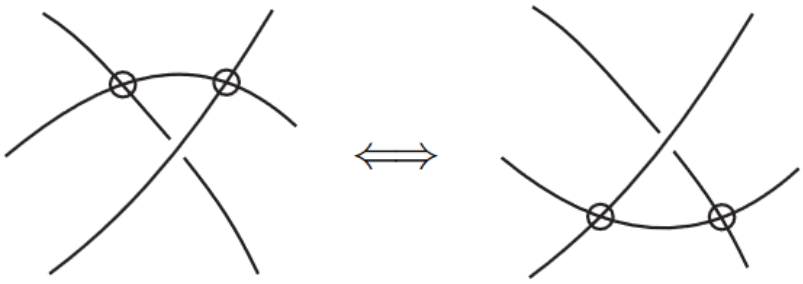


Рисунок 9. Полувиртуальное третье движение Рейдемейстера

Приложение Ж

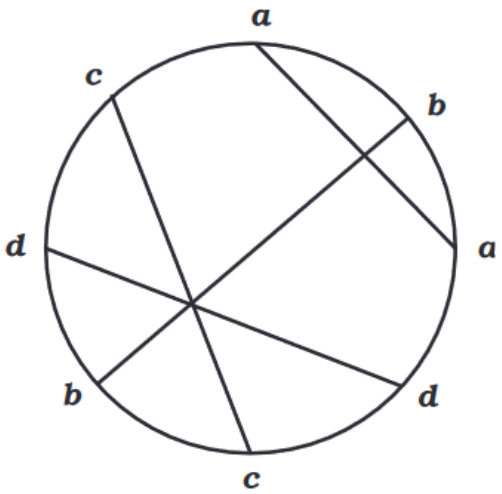


Рисунок 2. Пример хордовой диаграммы

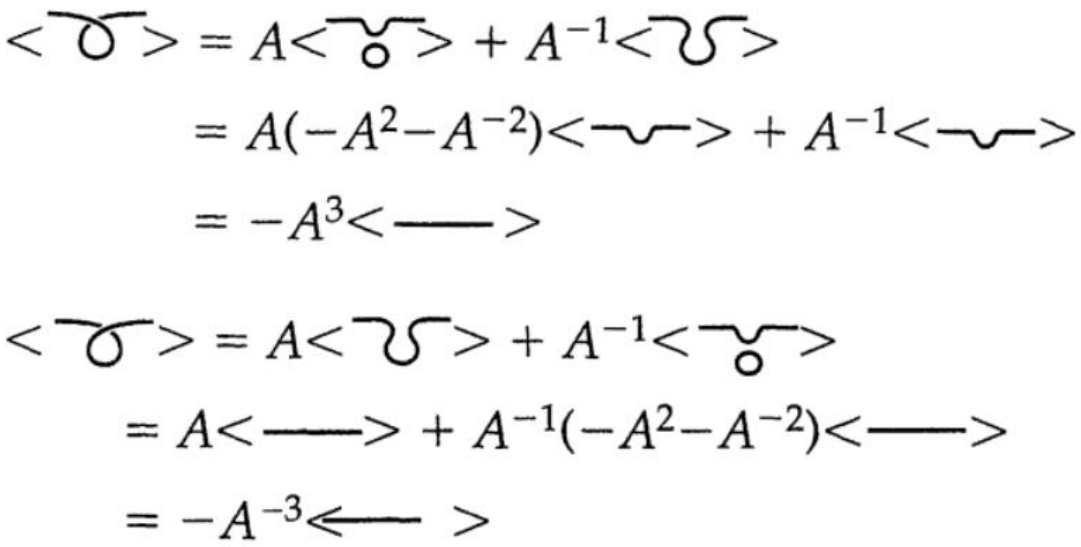


Рисунок 10. Скобка Кауфмана для первого движения Рейдемейстера

Приложение И

```
#include <iostream>
#include <string>
using namespace std;
class Byte{
    int n;
public:
    int* b;
    Byte() {
        n = 1;
        b = new int[n];
        b[0] = 0;
    }
    Byte(int a, int n) {
        this->n = n;
        b = new int[n];
        if (a == 0) {
            for(int i = 0; i<n; i++)
                b[i] = 0;
        }
        else{
            for (int i = n - 1; i >= 0; i--){
                int aa = a;
                a /= 2;
                b[i] = aa - 2 * a;
            }
        }
    }
};
void Swap(int* a, int* b, int n) {
    int tmp;
    for (int i = 0; i < n; i++) {
        tmp = a[i];
        a[i] = b[i];
        b[i] = tmp;
    }
}
long int Fact(int n){
    int f = 1;
    for (int i = 0; i < n; i++)
        f *= i + 1;
    return f;
}
void Sum(string* bracket, int cor, int al) {
    for (int i = 0; i <= cor; i++) {
        int fact = Fact(cor) / (Fact(cor - i) * Fact(i));
        int pw = 2 * (cor - i) - 2 * i;
        if (cor % 2 == 0) *bracket += "+";
        else *bracket += "-";
        if(fact == 1) *bracket += "a(" + to_string(al + pw) + ")";
        else *bracket += to_string(fact) + "a(" + to_string(al + pw) + ")";
    }
}

void String(string* bracket) {
    string str = *bracket, str1;
    long int n = str.length(), a = 0, b = 0;
    bool flag = false, br = false, brSgn = false, boo = false;
```

преображаем строку для более простого поиска

```
for (int i = 0; i < n; i++) {
    if (str[i] == '(') br = true;
    if (str[i] == ')') {
        br = false;
        if (boo == true) {
            str1 += str[i];
            boo = false;
            for (int j = 0; j < a; j++) {
                if (flag == true) str += '-' + str1;
                else str += '+' + str1;
            }
            if (flag == true) str1 = '-' + to_string(a) + str1;
            else str1 = '+' + to_string(a) + str1;
            b = str.find(str1, 0);
            str.erase(b, str1.length());
            i -= str1.length();
            str1.clear();
            a = 0;
        }
        flag = false;
    }
    if (br == false && str[i] == '-') flag = true;
    if (br == false && (str[i] >= '0' && str[i] <= '9')) {
        if (brSgn == false) brSgn = true;
        else a *= 10;
        a += str[i] - '0';
    }
    if (brSgn == true && str[i] == 'a') {
        boo = true;
        brSgn = false;
    }
    if (boo == true) str1 += str[i];
}
```

Удаляем из строки одинаковые элементы с разными знаками

```
br = flag = boo = false;
n = str.length();
for (int i = 0, j; i < n; i++){
    if (str[i] == '(') br = true;
    if (br == false && str[i] == '-') flag = true;
    if (str[i] == '+') boo = true;
    if (str[i] != ')') str1 += str[i];
    else {
        br = false;
        if (flag == true) {
            str1.erase(0, 1);
            str1 = '+' + str1 + ')';
            a = str.find(str1, i);
            if (a > -1) {
                str.erase(a, str1.length());
                str.erase(i - str1.length() + 1, str1.length());
                i -= str1.length();
            }
        }
        else{
            if (boo == true) str1.erase(0, 1);
            str1 = '-' + str1 + ')';
            a = str.find(str1, i + 1);
            if (a > -1) {
                str.erase(a, str1.length());
                if (boo == true)                    {
                    str.erase(i - (str1.length() - 1), str1.length());
                    i -= str1.length();
                    boo = false;
                }
                Else {
```

```
                    str.erase(i - (str1.length() - 2), str1.length() - 1);
                    i -= str1.length() - 1;
                }

            }
        }
        str1.clear();
        flag = false;
        n = str.length();
    }
}
```

Меняем элементы с нулевой степенью на 1

```
flag = false;
for (int i = 0; i < n; i++){
    if (str[i] == ')') {
        flag = false;
        boo = false;
    }
    if (str[i] == '-') boo = true;
    if (i == 0 && str[i] != '-')  = true;
    if (i > 0 && str[i] == '0' && str[i - 1] == '(') {
        if (flag == true) {
            i -= 2;
            str.erase(i, 4);
            str = str + "+1";
        }
        else {
            i -= 3;
            str.erase(i, 5);
            if (boo == true) str = str + "-1";
            else str = str + "+1";
        }
        n = str.length();
    }
}
```

Суммируем одинаковые элементы

```
flag = boo = br = false;
a = 0;
string s;
for (int i = 0; i < n; i++) {
    str1 += str[i];
    if (str1 == s) break;
    if (i == 0 && str[i] != '-' && str[i] != '+') boo = true;
    if (str1 == "-") flag = true;
    if (str[i] == ')') {
        int k = 1;
        if (boo == true)                str1 = '+' + str1;
        do {
            b = str.find(str1, i + 1);
            if (b != -1)                {
                k++;
                str.erase(b, str1.length());
            }
        } while (b != -1);
        if (boo == true) {
            str.erase(i - (str1.length() - 2), str1.length() - 1);
            i -= str1.length() - 1;
        }
        else{
            str.erase(i - (str1.length() - 1), str1.length());
            i -= str1.length();
        }
        if (flag == true) str += '-';
        else str += '+';
```

```
            str1.erase(0, 1);
            if (k > 1) str += to_string(k);
            str += str1;
            if (a == 0) {
                if (flag == true) s += '-';
                else s += '+';
                if (k > 1) s += to_string(k);
                s += str1;
            }
            str1.clear();
            boo = false;
            flag = false;
            n = str.length();
            a++;
        }
```

Суммируем единицы

```
        if (i > 0 && str[i] == '1' && str[i-1] == '+') {
            str1 += '+';
            int k = 1;
            do {
                b = str.find(str1, i + 1);
                if (b != -1) {
                    k++;
                    str.erase(b, str1.length() - 1);
                }
            } while (b != -1);
            str.erase(i - 1, 2);
            str += '+' + to_string(k);
            break;
        }
    }
    if (str[0] == '+') str.erase(0, 1);
    *bracket = str;
}

int Alpha(int* sgn, int n, Byte b){

    int al = 0;
    for (int i = 0; i < n; i++){
        if (sgn[i] * b.b[i] == -1) al++;

    }
    for (int i = 0; i < n; i++){
        if (b.b[i] == 1) continue;
        if (sgn[i] == 1) al++;
    }


    return al;
}
int Beta(int n, int al){
    return n - al;
}

int Rank(int** a, int n){
    int rank = 0;

    for (int i = 0; i < n; i++){

        bool flag = true;
        for (int k = 0; k < n; k++)
            for (int j = 0; j < n; j++)
                if (a[k][j] == 1) flag = false;
        if (flag == true) return rank;
```

```
        flag = true;
        for (int j = 0; j < n; j++)
            if (a[i][j] > 0) flag = false;
        if (flag == true) continue;
        if (i == n - 1) {
            rank++;
            break;
        }
        if (a[i][i] == 0 && i != n - 1) {
            int m = i;
            for (int j = i + 1; j < n; j++)
                if (a[j][i] != 0 && a[j][i] != a[m][i]) m = j;
            Swap(a[i], a[m], n);
        }
        for (int k = i + 1; k < n; k++){
            bool flag = false;
            for (int j = 0; j < n; j++)
                if (a[i][j] != a[k][j]) flag = true;
            if (flag == false) {
                for (int j = 0; j < n; j++)
                    a[k][j] = 0;
                continue;
            }
            int del;
            if (a[i][i] != 0 && a[k][i] != 0) del = a[k][i] / a[i][i];
            else del = 0;
            for (int j = 0; j < n; j++)
                a[k][j] -= del * a[i][j];
        }
        rank++;
    }
    return rank;
}

int Corank(int** a, int n, Byte b){
    int m = 0;
    for (int i = 0; i < n; i++)
        if (b.b[i] == 1) m++;
    if (m == 0) return 0;
    int** aa = new int* [m];
    for (int i = 0; i < m; i++)
        aa[i] = new int[m];
    for (int i = 0, k = -1; i < n; i++)
        if (b.b[i] == 1) {
            k++;
            for (int j = 0, g = -1; j < n; j++)
                if (b.b[j] == 1) {
                    g++;
                    aa[k][g] = a[i][j];
                }
        }
    return m - Rank(aa, m);
}

int TwistNumber(int** a, int* sgn, int n){
    int** e = new int* [n];
    int** c = new int* [n];
    int** aa = new int* [n];
    int cor = 0;
    Byte b(pow(2, n) - 1, n);
    for (int i = 0; i < n; i++){
        e[i] = new int[n];
        c[i] = new int[n];
        aa[i] = new int[n];
        for (int j = 0; j < n; j++)
```

```
            c[i][j] = e[i][j] = 0;
        e[i][i] = 1;
    }
    int sum = 0;
    for (int i = 0; i < n; i++){
        c[i][i] = 1;
        for (int k = 0; k < n; k++){
            for (int j = 0; j < n; j++)
                aa[k][j] = (a[k][j] + e[k][j] + c[k][j]) % 2;
        }
        sum += pow(-1, Corank(aa, n, b)) * sgn[i];
        c[i][i] = 0;
    }
    return sum;
}
void JonesPolynomial(string bracket, int sum){
    string polynomial = "";
    int n = bracket.length(), a = 0;
    bool flag = false, sgn = false, br = false;

    if (sum == 0) polynomial = bracket;
    else{
        if (bracket[0] == 'a') {
            polynomial += '-';
            polynomial += bracket[0];
        }
        for (int i = 1; i < n; i++){
            if (flag == false && bracket[i] == '-') {
                polynomial += '+';
                continue;
            }
            if (bracket[i] == '+') {
                polynomial += '-';
                continue;
            }
            if (flag == false) polynomial += bracket[i];
            if (bracket[i] == '(') flag = true;
            if (flag == true && bracket[i] == '-') sgn = true;
            if (flag == true && (bracket[i] >= '0' && bracket[i] <= '9')) {
                if (br == true) a *= 10;
                else br = true;
                a += bracket[i] - '0';
            }
            if (bracket[i] == ')') {
                if (sgn == true) a *= -1;
                polynomial += to_string(a - 3 * sum) + ')';
                a = 0;
                flag = sgn = br = false;
            }
        }
    }
    cout << "Jones Polynomial: " << polynomial;
}

void KauffmanBracket(int** a, int* sgn, int n){
    string bracket;
    for (int i = 0; i < pow(2, n); i++){
        int alpha, beta, cor;
        Byte b(i, n);
        alpha = Alpha(sgn, n, b);
        beta = Beta(n, alpha);
        cor = Corank(a, n, b);
        int al = alpha - beta;

        switch (cor)
```

```
        {
        case 0:
            if (i == 0)
                bracket += "a(" + to_string(al) + ")";
            else
                bracket += "+a(" + to_string(al) + ")";
            break;
        case 1:
            bracket += "-a(" + to_string(2 + al) + ")-a(" + to_string(-2 + al) + ")";
            break;
        default:
            Sum(&bracket, cor, al);
            break;
        }
    }
    String(&bracket);
    cout << endl << "Kauffman bracket: " << bracket << endl;
}

int main(){
    int n = 3;
    cout << "Enter the number of crossing: ";
    cin >> n;
    cout << "Enter the matrix:" << endl;
    int** a = new int* [n];
    for (int i = 0; i < n; i++){
        a[i] = new int[n];
        for (int j = 0; j < n; j++)
            cin >> a[i][j];
    }
    cout << "Enter signs: ";
    int* sgn = new int[n];
    for (int i = 0; i < n; i++)
        cin >> sgn[i];

    KauffmanBracket(a, sgn, n);
    JonesPolynomial(bracket, TwistNumber(a, sgn, n));
}
```

Приложение К

Знаки на перекрестках узла следующие:

1. +
2. –
3. +

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Рисунок 11. Матрица смежности трилистника

Вычисления:

$$000 = a^{2-1}(-a^2 - a^{-2})^0 = a$$

$$001 = a^{1-2}(-a^2 - a^{-2})^1 = -a - a^{-3}$$

$$010 = a^{3-0}(-a^2 - a^{-2})^1 = -a^5 - a$$

$$011 = a^{2-1}(-a^2 - a^{-2})^0 = a$$

$$100 = a^{1-2}(-a^2 - a^{-2})^1 = -a - a^{-3}$$

$$101 = a^{0-3}(-a^2 - a^{-2})^2 = a + 2a^{-3} + a^{-7}$$

$$110 = a^{2-1}(-a^2 - a^{-2})^0 = a$$

$$111 = a^{1-2}(-a^2 - a^{-2})^1 = -a - a^{-3}$$

Результат вычислений:

$$\langle G \rangle(a) = \sum_s a^{\alpha(s)-\beta(s)}(-a^2-a^{-2})^{corank_{Z_2}A(G(s))} = -a^5 + a^{-7} - a^{-3}$$

```
Enter the number of crossing: 3
Enter the matrix:
0 1 0
1 0 1
0 1 0
Enter signs: 1 -1 1

Kauffman bracket: -a(5)+a(-7)-a(-3)
Jones Polynomial: a(2)-a(-10)+a(-6)
```

Рисунок 12. Вычисление скобки Кауфмана и полинома Джонса для трилистника. Вывод программы

Приложение Л

Знаки на перекрестках узла следующие:

$+ + \ - \ -$

$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Рисунок 13. Матрица смежности узла "восьмерка"

Вычисления:

$0000 = a^{2-2}(-a^2 - a^{-2})^0 = 1$

$0001 = a^{3-1}(-a^2 - a^{-2})^1 = -a^4 - 1$

$0010 = a^{3-1}(-a^2 - a^{-2})^1 = -a^4 - 1$

$0011 = a^{4-0}(-a^2 - a^{-2})^2 = a^8 + 2a^4 + 1$

$0100 = a^{1-3}(-a^2 - a^{-2})^1 = -1 - a^{-4}$

$0101 = a^{2-2}(-a^2 - a^{-2})^0 = 1$

$0110 = a^{2-2}(-a^2 - a^{-2})^0 = 1$

$0111 = a^{3-1}(-a^2 - a^{-2})^1 = -a^4 - 1$

$1000 = a^{1-3}(-a^2 - a^{-2})^1 = -1 - a^{-4}$

$1001 = a^{2-2}(-a^2 - a^{-2})^0 = 1$

$1010 = a^{2-2}(-a^2 - a^{-2})^0 = 1$

$1011 = a^{3-1}(-a^2 - a^{-2})^1 = -a^4 - 1$

$1100 = a^{0-4}(-a^2 - a^{-2})^2 = 1 + 2a^{-4} + a^{-8}$

$1101 = a^{1-3}(-a^2 - a^{-2})^1 = -1 - a^{-4}$

$1110 = a^{1-3}(-a^2 - a^{-2})^1 = -1 - a^{-4}$

$1111 = a^{2-2}(-a^2 - a^{-2})^2 = a^4 + 2 + a^{-4}$

Результат вычислений:

$$\langle G \rangle(a) = \sum_s a^{\alpha(s) - \beta(s)}(-a^2 - a^{-2})^{corank_{Z_2} A(G(s))} = a^8 + a^{-8} - a^{-4} - a^4 + 1$$

```
Enter the number of crossing: 4
Enter the matrix:
0 0 1 1
0 0 1 1
1 1 0 0
1 1 0 0
Enter signs: 1 1 -1 -1

Kauffman bracket: a(8)-a(4)+a(-8)-a(-4)+1
Jones Polynomial: a(8)-a(4)+a(-8)-a(-4)+1
```

Рисунок 14. Вычисление скобки Кауфмана и полинома Джонса для узла "восьмерка". Вывод программы

Приложение М

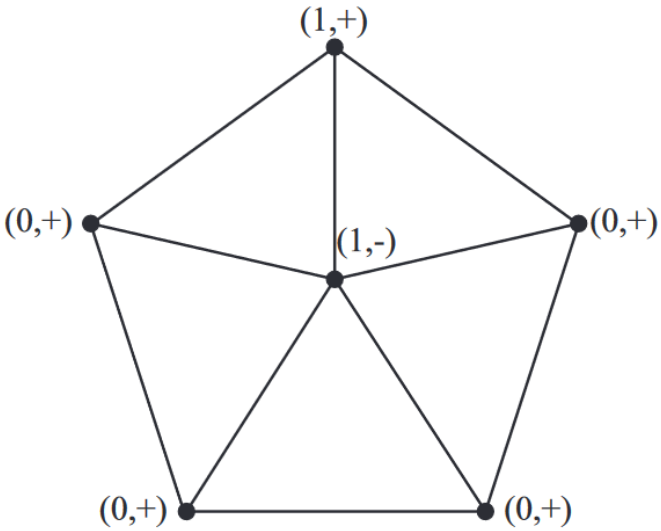


Рисунок 11. Граф-зацепление с тривиальным полиномом Джонса