Федеральное государственное автономное образовательное учреждение высшего профессионального образования

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ МАТЕМАТИКИ И МЕХАНИКИ

КАФЕДРА АЛГЕБРЫ И МАТ. ЛОГИКИ

Направление: 01.03.01 — Математика, бакалавр математики.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(Бакалаврская работа)

Вычисление идемпотентов конечных групповых алгебр в СКА “Sage”

Работа завершена:

«___»_____2015 г. _____Ш.Ш. Ишмухамедов

Работа проверена:

Научный руководитель
кандидат физико-математических наук,
доцент кафедры алгебры и мат. логики
«___»_____2015 г. _____М.Ф. Насрутдинов

Заведующий кафедрой алгебры и мат. логики
доктор физико-математических наук, профессор
«___»_____2015 г. _____М. М. Арсланов

Казань — 2015 г.

Содержание

Введение

В последние годы большое внимание уделяется изучению строения групповых алгебр, в частности конечных групповых алгебр. Это обусловлено повышенным интересом к групповым кодам, которые в свою очередь очень тесно связаны с теорией конечных групповых алгебр, поскольку большое количество групповых кодов являются идеалами в конечных групповых алгебрах.

В то же время активно развиваются мощные компьютерные инструменты для работы с математическими объектами, называемые «Системами компьютерной алгебры (СКА)». СКА Sage одна из таких систем, это некоммерческое ПО с открытым исходным кодом. Sage стремительно растет и имеет большое сообщество пользователей и разработчиков, которые готовы помочь в решении возникающих вопросов в процессе работы с системой.

В дипломной работе демонстрируются возможности системы компьютерной алгебры Sage для работы с конечными группами и конечными групповыми алгебрами. Основная часть работы будет посвящена разработке алгоритма вычисления центральных ортогональных и примитивных центральных идемпотентов конечных групповых алгебр.

Цель данной работы - познакомить читателя с инструментами СКА Sage используемые в теории групп и групповых алгебр, а также продемонстрировать создание на основе существующих возможностей собственного алгоритма вычисления идемпотентов конечных групповых алгебр.

1 Предварительные сведения

Группы

Определение 1. Для элемента g группы G, *левый смежный класс* по подгруппе H — это множество $gH = \{gh|h \in H\}$, *правый смежный класс* по подгруппе H — множество $Hg = \{hg|h \in H\}$.

Определение 2. Для элемента g группы G *классом сопряженности* называется множество всех его сопряжённых элементов: $\{hgh^{-1}|h \in G\}$.

Определение 3. Пусть G — группа, H — её нормальная подгруппа и $a \in G$ — произвольный элемент. Тогда на классах смежности H в G $aH = \{\, ah \mid h \in H\}$ можно ввести умножение:

$$(aH)(bH) = abH$$

Легко проверить, что это умножение не зависит от выбора элементов в классах смежности, то есть, если $aH = a'H$ и $bH = b'H$, то $abH = a'b'H$. Это умножение определяет структуру группы на множестве классов смежности, а полученная группа G/H называется *факторгруппой* G по H.

Определение 4. Для некоторого множества X группа всех его перестановок (то есть биекций $X \to X$) относительно операции композиции называется *симметрической группой*.

Симметрическая группа множества X обычно обозначается $S(X)$. Если $X = \{1, 2, \ldots, n\}$, то $S(X)$ также обозначается через S_n.

Элементы этой группы называют *перестановками(подстановками)* и записываются в виде таблицы из 2 строк, где в верхней строке записаны элементы из множества аргументов функции, а в нижней элеметы из множества значений функции. Для произвольного $\sigma \in S_n$ это выглядит так:

$$\sigma = \begin{pmatrix} 1 & 2 & \dots & i & \dots & n \\ \sigma(1) & \sigma(2) & \dots & \sigma(i) & \dots & \sigma(n) \end{pmatrix}$$

Теорема 1 (Теорема Кэли)**.** Любая конечная группа $(G, \circ)$ изоморфна некоторой подгруппе группы перестановок множества элементов этой группы. При этом каждый элемент $a \in G$ сопоставляется с перестановкой π_a, задаваемой тождеством $\pi_a(g) = a \circ g$, где g — произвольный элемент группы G.

Групповые алгебры

Определение 5. Пусть G - группа, K - произвольное ассоциативное кольцо. Групповое кольцо KG группы G над кольцом K состоит из всевозможных формальных сумм вида $\sum_{g\in G} \alpha_g g$, в которых лишь конечное число коэффициентов $\alpha_g \in K$ отличных от нуля, причем такие суммы равны тогда итолько тогда, когда у них совпадают коэффициенты $\alpha_g \in K$ для всех $g \in G$.

Операции в KG определяются так: если

$$x = \sum\nolimits_{g\in G} \alpha_g g, \quad y = \sum\nolimits_{g\in G} \beta_g g$$

то

$$x + y = \sum\nolimits_{g\in G} (\alpha_g + \beta_g) g, \quad xy = \sum\nolimits_{g,h\in G} \alpha_g \beta_h (gh),$$

Если кольцо K является полем, то KG называется групповой алгеброй.

Определение 6. Пусть KG - групповая алгебра. *Центром групповой алгебры* называется множество $Z(KG) = \{a \in KG| \quad \forall x \in KG : ax = xa\}$

Утверждение 1. Пусть KG - групповая алгебра конечной группы G над полем K. Тогда $x \in Z(KG) \Leftrightarrow x_g = x_{h^{-1}gh} \quad \forall g, h \in G$.

Доказательство. Имеем $x \in Z(KG) \Leftrightarrow xh = hx \quad \forall h \in G$. Таким образом,

$$x \in Z(KG) \Leftrightarrow \sum_{g \in G} x_g g = \sum_{g \in G} x_g hgh^{-1} \qquad (*)$$

Зафиксируем $h \in G$, имеем $\{hgh^{-1} : \ g \in G\} = G$. Пусть $hgh^{-1} = u$, тогда $g = h^{-1}uh$. Из (*) следует:

$$\sum_{g \in G} x_g g = \sum_{g \in G} x_{h^{-1}gh} g.$$

□

Теорема 2. Пусть KG - групповая алгебра конечной группы G над полем K и $C_1, \ldots, C_m$ - классы сопряженных элементов группы G. Тогда $Z(KG) =< \widehat{C_1}, \ldots, \widehat{C_n} >$, где $\widehat{C_k}$ есть сумма всех элементов класса C_k, т.е. $\widehat{C_k} = \sum_{g \in C_k} g \in KG$.

Доказательство. Если $1 \leq i \leq m$, то по определению, $g \in C_i h^{-1}gh \in C_i \quad \forall h \in G$. Следовательно для $g, h \in G$, коэффициенты g и $h^{-1}gh$ в $\widehat{C}_i$ оба равны 1 либо 0. Таким образом согласно утверждению 1 имеем: $\widehat{C}_i \in Z(KG) 1 \leq i \leq m$. Таким образом $< \widehat{C_1}, \ldots, \widehat{C_n} > \subseteq Z(KG)$.

Обратно, если $x = \sum_{g \in G} x_g g \in Z(KG)$, тогда, по утверждению 1, $x_g =$

$x_{g'}$ для любых g и g' находящихся в одном классе сопряженности. Таким образом, если для каждого i мы выберем $g_i \in C_i$, то

$$x = \sum_{i=1}^{m} x_{g_i} \widehat{C}_i$$

Следовательно $Z(KG) \subseteq < \widehat{C_1}, \ldots, \widehat{C_m} >$. □

Идемпотенты

Определение 7. Элемент e из алгебры A называется идемпотентом, если $e^2 = e$.

Определение 8. Семейство идемпотентов $e_1, \ldots, e_n$ из алгебры A называется ортогональным, если выполняется тождество: $e_i e_j = \delta_{ij} e_i$, где δ_{ij} — символ Кронекера.

Определение 9. Идеспотент e алгебры A называется:

1. центральным, если $ex = xe \quad \forall x \in A$.

2. примитивным, если e не может быть представлен в виде суммы двух центральных ортогонольных идемпотентов.

2 СКА Sage

Sage (анг. 'Мудрец') — система компьютерной алгебры покрывающая много областей математики, включая алгебру, комбинаторику, вычислительную математику и матанализ.

Первая версия Sage была выпущена 24 февраля 2005 года в виде свободного программного обеспечения с лицензией GNU GPL. Первоначальной целью проекта было "создание открытого программного обеспечения альтернативного системам Magma, Maple, Mathematica, и MATLAB". Разработчиком системы является Уильям Стейн — математик Университета Вашингтона. Официальный сайт: `sagemath.org`.

Sage написан на интерпретируемом языке программирования Python. Возможности Sage можно расширять собственным кодом написаным на Python.

Принципиальное отличие от других систем компьютерной алгебры - наличие программных интерфейсов для работы с системами Mathematica, Magma, и Maple, другими словами из Sage можно использовать системы Mathematica, Magma, и Maple.

Также для Sage функционируют облачные сервисы, которые позволяют работать с Sage удаленно. Другими словами для работы вам не нужно будет ничего устанавливать себе на компьютер, не нужно иметь большие вычислительные мощности и т.д. Нужен лишь браузер. Вот список некоторых из облачных сервисов:

- SMC
- Sagenb
- Sagecell

Мы будем использовать SMC. Также этот сервис включает в себя полноценный редактор .tex файлов. Над проектами в SMC возможна совместная работа нескольких человек. Причем все это доступно бесплатно.

Создание групп и групповых алгебр

В Sage для полноценной работы с конечными группами необходимо представлять их в виде подгруппы группы перестановок. Согласно теореме Кэли любая конечная группа изоморфна некоторой подгруппе группы перестановок, поэтому здесь и далее мы будем рассматривать не сами группы, а некоторые изоморфные им подгруппы групп перестановок.

Перестановки в Sage представляются в виде произведения циклов (как нам известно любая перестановка $\pi \neq \epsilon$ может быть разложена в произведение (композицию) непересекающихся циклов длины $l \geq 2$).
Используют 2 варианта записи перестановки: в виде строки и в виде кортежа. Например перестановка $\sigma = (13)(254)$ будет выглядеть следующим образом:

1. В виде строки: "(1,3) (2,5,4)"
2. В виде "кортежа": [(1,3), (2,5,4)]

Список групп перестановок известных в Sage:

- SymmetricGroup(n) - симметрическая группа степени n.
- DihedralGroup(n) - группа симметрий правльного n-угольника (n поворотов и n вращений)

- CyclicPermutationGroup(n) - группа поворотов правильного n-угольника.
- AlternatingGroup(n) - знакопеременная группа степени n.
- KleinFourGroup() - четвертая группа Клейна.

Создаются группы перестановок одинаково. Рассмотрим создание на примере симметрической группы.
Для создания необходимо воспользоваться конструкцией:
$SymmetricGroup(n)$, где n - степень симметрической группы. Например создадим группу S_3:

```
g = SymmetricGroup(3)
g
# Symmetric group of order 3! as a permutation group
```

Чтобы создать элемент группы перестановок, необходимо передать перестановку в качестве параметра в эту группу. Например создадим элементы $\sigma = (1,3)(2,5,4), \rho = (2,4)(1,5)$ группы S_5 и перемножим их между собой:

```
sage: G = SymmetricGroup(5)
sage: sigma = G([(1,3), (2,5,4)])
sage: rho = G([(2,4), (1,5)])
sage: sigma*rho
# (1,3,5,2)
```

Конечные группы заданные через определяющие соотношения создаются в виде фактор-группы свободной группы по некоторой нормальной подгруппе. Чтобы создать конечную группу с порождающими $g_1, \ldots, g_m$

с определяющими соотношениями $\alpha = g_1^{\alpha_1} * \cdots * g_m^{\alpha_m} = 1, \ldots, \phi = g_1^{\phi_1} * \cdots * g_m^{\phi_m} = 1$ необходимо воспользоваться методом *quotient*() свободной группы, передав в качестве параметра список $\alpha, \ldots, \phi$.

Рассмотрим пример создания группы с определяющими соотношениями $x^3y = 1, xy^3 = 1, x^2y^{-2}$.

```
# Создаем свободную группу
sage: f.<x, y> = FreeGroup()
# Создаем конечную группу
sage: g = f.quotient([x^3*y, x*y^3, x^2*y^(-2)])
g
# Finitely presented group < x, y | x^3*y, x*y^3, x^2*y^-2 >
```

Чтобы получить для некоторой группы изоморфную ей подгруппу P некоторой группы перестановок необходимо воспользоваться методом *as_permutation_group()*. Для конечной группы из предыдущего примера это будет выглядеть так:

```
sage: P = g.as_permutation_group()
sage: P
# Permutation Group with generators [(1,2,6,5,8,4,7,3), (1,4,6,3,8,2,7,5)]
```

Мы получили группу перестановок $< (1, 2, 6, 5, 8, 4, 7, 3), (1, 4, 6, 3, 8, 2, 7, 5) >$

Для создания конечного поля из n элементов используется конструкция *GF(n)*. Пример создания поля из 7 элементов:

```
sage: F7 = GF(7)
sage: F7
# Finite Field of size 7
```

Другие примеры листингов работы с группами смотрите в Приложении А.

Для создания групповых алгебр используется конструкция *GroupAlgebra(G, F)*, где F - это поле, G - группа. Группа G должна быть группой перестановок. Рассмотрим пример создания групповой алгебры F_7P, где P - группа перестановок изоморфная конечной группе $< x, y >$ с определяющими соотношениями $x^3 * y = 1, x * y^3 = 1, x^2y^{-2}$.

```
sage: F.<x, y> = FreeGroup()
sage: G = F.quotient([x^3*y, x*y^3, x^2*y^(-2)])
sage: P = G.as_permutation_group()

sage: F7 = GF(7)
sage: FP = GroupAlgebra(P, F7)
sage: FP
# Group algebra of group "Permutation Group with generators [(1,2,6,5,8,4,7,3),
# (1,4,6,3,8,2,7,5)]" over base ring Finite Field of size 7
```

Для возможности произведения элементов поля и элементов группы, необходимо элементы поля перевести в элементы групповой алгебры. Рассмотрим пример элемента $5 * (1, 5)(2, 4)(6, 8)$ групповой алгебры F_7D_8.

```
sage: D8 = DihedralGroup(8)
sage: F7 = GF(7)
sage: ga = GroupAlgebraExt(D8, F7)

sage: ga(5)*D8("(1,5)(2,4)(6,8)")
# 5*(1,5)(2,4)(6,8)
```

Из-за того, что при создании групповой алгебры группа всегда должна быть подгруппой группы перестановок, элементы групповых алгебр в Sage представлены в непривычном для читателя виде, в виде линейной комбинации перестановок с коэффициентами из заданного поля. Чтобы это исправить было решено написать функцию $repr\_from\_gens(elem)$, которая для заданного элемента $elem$ возвращала бы его представление в читаемом виде, а именно в виде линейной комбинации образующих элементов группы над заданным полем. Листинг этой функции представлен в Приложении C.

Применим эту функцию к предыдущему примеру:

```
sage: D8 = DihedralGroup(8)
sage: F7 = GF(7)
sage: ga = GroupAlgebraExt(D8, F7)

sage: a = ga(5)*D8("(1,5)(2,4)(6,8)")
sage: a
# 5*(1,5)(2,4)(6,8)
sage: repr_from_gens(a)
# 5x_2 * x_1^-3
```

Примеры листингов работы с групповыми алгебрами описаны в Приложении B.

3 Вычисление идемпотентов

К сожалению, на момент написания работы в Sage не было реализовано вычисление идемпотентов конечных групповых алгебр. Поэтому было решено разработать собственный алгоритм вычисления центральных ортогональных и примитивных центральных идемпотентов.

Алгоритм

Пусть задана групповая алгебра KG группы G над полем K. Для нахождения центральных ортогональных идемпотентов будем перебирать все центральные элементы $x, x \in Z(KG)$ и выбирать из них только те, для которых выполняется $x^2 = x$, то есть только идемпотенты, а затем из них выбрать только ортогональные.

Для нахождения центра групповой алгебры воспользуемся **Теоремой 2**. Согласно теореме множество линейных комбинаций $< \widehat{C_1}, \ldots, \widehat{C_m} >$ над полем K и является центром групповой алгебры. Для этого нам понадобится сначала найти все классы сопряженных элементов $C_1, \ldots, C_m$ группы G, далее для каждого C_i вычислить $\widehat{C_i} = \sum_{g \in C_i} g$ и затем найти всевозможные линейные комбинации $< \widehat{C_1}, \ldots, \widehat{C_m} >$.

Наш алгоритм вычисления центральных идемпотентов будем выглядеть следующим образом:

1. Находим классы сопряженных элементов C_i группы G
2. Вычисляем $\widehat{C_i} = \sum_{g \in C_i} g$ для каждого класса сопряженных элементов C_i

3. Находим множество всех центральных элементов в виде $Z(KG) =< \widehat{C_1}, \ldots, \widehat{C_m} >$

4. Перебираем все центральные элементы, выбирая только идемпотенты

5. Среди найденных идемпотентов выбираем только ортогональные

Ниже представлена реализация каждого шага алгоритма на Sage для заданной групповой алгебры KG группы G над полем K.

1 шаг: Нахождение классов сопряженных элементов

```
# Классы сопряженных элементов C_i
conj_classes = G.conjugacy_classes()
conj_classes_list = []
for cl in conj_classes:
    conj_classes_list.append(cl.list())
```

2 шаг: вычисление $\widehat{C_i} = \sum_{g \in C_i} g$ для каждого класса сопряженных элементов C_i

```
# Суммы C_i^ = sum_{g in C_i} g
conj_class_sums = []
for cl in conj_classes_list:
    conj_class_sum = KG.zero()
    for g in cl:
        conj_class_sums += KG(g)
    conj_class_sums.append(conj_class_sum)
```

3 шаг: вычисление всевозможных линейных комбинаций $Z(KG) =< \widehat{C_1}, \ldots, \widehat{C_m} >$

```
# находим всевозможные лин. комбинации < C_1, ..., C_m >
term_list = []
for conj_class_sum in conj_class_sums:
    terms = []
    for w in K:
        terms.append(KG(w)*conj_class_sum)
    terms_list.append(terms)

central_elements = terms_list[0]
for i in xrange(1, len(terms_list)):
    temp = set()
    for y in terms_list[i]:
        for x in result:
            temp.add(x+y)
    central_elements = temp
```

4 шаг: находим идемпотенты среди центральных элементов

```
# Находим идемпотенты среди центральных элементов
central_idems = []
for elem in central_elements:
    if elem.is_idempotent() and elem != ga.zero():
        central_idems.append(elem)
```

5 шаг: среди найденных выбираем только ортогональные

```
# Проверяем ортогональность
central_orthogonal_idems = []
for e1 in central_idems:
    for e2 in central_idems:
        if e1 != e2 and e1*e2 == self.zero()
            central_orthogonal_idems.append(e1)
```

Для нахождения центральных примитивных идемпотентов достаточно добавить к нашему алгоритму еще один шаг, на котором выбрать из центральных ортогональных идемпотентов только те, которые не выражаются через сумму двух других.

Ниже представлен код реализующий этот шаг:

```
# Находим примитивные идемпотенты
primitive_central_idems = []
for e in central_orthogonal_idems:
    is_primitive = true
    for e1 in central_orthogonal_idems:
        for e2 in central_orthogonal_idems:
            if e != e1 and e != e2 and e == e1+e2:
                is_primitive = false
                break
        if not is_primitive:
            break
    primitive_central_idems.append(e)
```

В итоге алгоритм вычисления примитивных центральных идемпотентов будет выглядеть так:

1. Находим классы сопряженных элементов C_i группы G
2. Вычисляем $\widehat{C}_i = \sum_{g \in C_i} g$ для каждого класса сопряженных элементов C_i
3. Находим множество всех центральных элементов в виде $Z(KG) = < \widehat{C_1}, \ldots, \widehat{C_m} >$
4. Перебираем все центральные элементы, выбирая только идемпотенты

5. Среди найденных идемпотентов выбираем только ортогональные

6. Выбираем среди центральных ортогональных идемпотентов только те, которые не выражаются через сумму двух других.

Для повторного использования оформим наш код в виде класса GroupAlgebraExt (сократив при этом некоторые методы), который будет расширять стандартный в Sage класс GroupAlgebra.
В итоге получим:

```
from sage.algebras.group_algebra import GroupAlgebra
class GroupAlgebraExt(GroupAlgebra):
  _central_orthogonal_idems = set()
  _print_info = false

  def __init__(self, group, base_ring = IntegerRing()):
    super(GroupAlgebraExt, self).__init__(group, base_ring)

  def _central_elements(self):
    F = self.base_ring()
    G = self.group()

    conj_classes = G.conjugacy_classes()
    conj_classes = [cl.list() for cl in conj_classes]

    conj_class_sums = [sum([self(g) for g in cl])
                       for cl in conj_classes]

    # находим всевозможные лин. комбинации
    central_elements = self._linear_combs(conj_class_sums)
    return central_elements

  def primitive_central_idems(self):
    central_orthogonal_idems = self.central_orthogonal_idems()
    return [e for e in central_orthogonal_idems if self._is_primitive_idempotent(e)]

  # продолжение на след. странице
```

```
  # Продолжение

  def central_orthogonal_idems(self):
    if self._central_orthogonal_idems:
      return self._central_orthogonal_idems

    central_elements = self._central_elements()
    central_idems = [c for c in central_elements if c.is_idempotent()
                                                 and c != self.zero()]

    central_idems_len = len(central_idems)
    for i in xrange(central_idems_len):
      for j in xrange(i+1, central_idems_len):
        if central_idems[i]*central_idems[i+1] != self.zero():
          break
        self._central_orthogonal_idems.add(central_idems[i])

    return self._central_orthogonal_idems

  def _is_primitive_idempotent(self, e):
    central_orthogonal_idems = self.central_orthogonal_idems()
    for e1 in central_orthogonal_idems:
      for e2 in central_orthogonal_idems:
        if e != e1 and e != e2 and e == e1+e2:
          return False
    return True

  def _linear_combs(self, elements):
    F = self.base_ring()

    terms = [[self(f)*elem for f in F] for elem in elements]

    result = terms[0]
    for i in xrange(1, len(terms)):
      result = set([x+y for x in result for y in terms[i]])
    return result
```

Идемпотенты групповой алгебры вида F_pC

Применим наш алгоритм для вычисления идемпотентов групповой алгебры F_{57}, где $F5$ - поле из 5 элементов, а C_7 - циклическая группа $< a >$ из 7 элементов.

```
sage: load("logic.sage") # файл с нашим классом
sage: f.<a> = FreeGroup()
sage: C7 = (f/[a^7]).as_permutation_group()
sage: F5 = GF(5)
sage: CF = GroupAlgebraExt(C7, F5)
sage: cidems = CF.central_orthogonal_idems()
sage: pidems = CF.primitive_central_idems()
```

Теперь в переменных $cidems, pidems$ хранятся центральные ортогональные и центральные примитивные идемпотенты соответственно. Покажем их количество и выведем на экран центральные примитивные идемпотенты:

```
sage: len(cidems), len(pidems)
# 3, 2
sage: for i in range(len(pidems)):
sage:     print 'e{}={}'.format(i+1, pidems[i])
# e_1 = 3*() + 2*(1,2,4,6,7,5,3) + 2*(1,3,5,7,6,4,2) + 2*(1,4,7,3,2,6,5)+
# +2*(1,5,6,2,3,7,4) + 2*(1,6,3,4,5,2,7) + 2*(1,7,2,5,4,3,6)
# e_2 = 3*() + 3*(1,2,4,6,7,5,3) + 3*(1,3,5,7,6,4,2) + 3*(1,4,7,3,2,6,5)+
# 3*(1,5,6,2,3,7,4) + 3*(1,6,3,4,5,2,7) + 3*(1,7,2,5,4,3,6)
# В виде порождающих
sage: for i in range(len(pidems)):
sage:     print 'e{}={}'.format(i+1, repr_from_gens(pidems[i]))
# e_1 = 2x_1 + 2x_1^-1 + 2x_1^-2 + 2x_1^-3 + 2x_1^2 + 2x_1^3 + 3
# e_2 = 3 + 3x_1 + 3x_1^-1 + 3x_1^-2 + 3x_1^-3 + 3x_1^2 + 3x_1^3
```

Из листинга видно, что наш алгоритм получил три центральных ортогональных идемпотента и два центральных примитивных идемпотента. Центральные примитивные идемпотенты имеют вид: $e_1 = 2x_1 + 2x_1^{-1} + 2x_1^{-2} + 2x_1^{-3} + 2x_1^2 + 2x_1^3 + 3, e_2 = 3 + 3x_1 + 3x_1^{-1} + 3x_1^{-2} + 3x_1^{-3} + 3x_1^2 + 3x_1^3$.

Проверим действительно ли элементы e_1, e_2 являются идемпотентами и проверим равна ли их сумма единице групповой алгебры:

```
# Проверим действительно ли это идемпотенты
sage: pidems[0]^2 == pidems[0], pidems[1]^2 == pidems[1]
# True, True
# Проверим сумму примитивных центральных элементов
sage: sum(pidems)
# () - это единица нашей групповой алгебры
```

Из листинга видно, что условия выполняются, а это значит, что алгоритм работает корректно.

Идемпотенты групповой алгебры вида F_pG

Вычислим идемпотенты групповой алгебры F_5G, где $F5$ - поле из 5 элементов, а G - группа заданная определяющими соотношениями $x^3y = 1, xy^3 = 1, x^2 = y^2$.

```
sage: load("logic.sage") # файл с нашим классом
sage: f.<x, y> = FreeGroup()
sage: G = (f/[x**3*y, x*y**3, x**2*y**(-2)]).as_permutation_group()
sage: F5 = GF(5)
sage: GF = GroupAlgebraExt(G, F5)
sage: cidems = GF.central_orthogonal_idems()
sage: pidems = GF.primitive_central_idems()
# Количество центральных ортогональных и центральных
# примитивных идемпотентов
sage: len(cidems), len(pidems)
# 63, 6
# Выведем центральные примитивные ид-ты на экран
sage: for i in range(len(pidems)):
sage:     print 'e{}={}'.format(i+1, repr_from_gens(pidems[i]))
# e_1 = 2 + 2x_1 + 2x_1^{-1} + 2x_1^{-1} * x_2^{-1} + 2x_1^{-2} + 2x_1^{-4} + 2x_2 + 2x_2^{-1}
# e_2 = 2 + 2x_1^{-1} * x_2^{-1} + 2x_1^{-2} + 2x_1^{-4} + 3x_1 + 3x_1^{-1} + 3x_2 + 3x_2^{-1}
# e_3 = 2 + 2x_1^{-4} + 3x_1^{-1} * x_2^{-1} + 3x_1^{-2} + 4x_1^{-1} + 4x_2^{-1} + x_1 + x_2
# e_4 = 2x_1^{-2} + 3x_1^{-1} * x_2^{-1} + 4 + x_1^{-4}
# e_5 = 2x_1^{-1} * x_2^{-1} + 3x_1^{-2} + 4 + x_1^{-4}
# e_6 = 2 + 2x_1^{-4} + 3x_1^{-1} * x_2^{-1} + 3x_1^{-2} + 4x_1 + 4x_2 + x_1^{-1} + x_2^{-1}
# Проверим действительно ли это идемпотенты
sage: [x^2==x for x in pidems]
# [True, True, True, True, True, True]
# Проверим сумму примитивных центральных элементов
sage: sum(pidems)
# () - это единица нашей групповой алгебры
```

Идемпотенты групповой алгебры вида F_pD_n

Вычислим идемпотенты групповой алгебры F_5D_8, где $F5$ - поле из 5 элементов, а $D8$ группа Диэдра порядка 16.

```
sage: load("logic.sage") # файл с нашим классом
sage: D8 = DihedralGroup(8)
sage: F5 = GF(5)
sage: FD = GroupAlgebraExt(D8, F5)
sage: cidems = FD.central_orthogonal_idems()
sage: pidems = FD.primitive_central_idems()
# Количество центральных ортогональных и центральных
# примитивных идемпотентов
len(cidems), len(pidems)
# 63, 6
# Выведем на экран центр. прим. ид-ты
sage: for i in range(len(pidems)):
sage:     print 'e{}={}'.format(i+1, repr_from_gens(pidems[i]))
# e_1 = 4 + 4x_1^{-4} + x_1^{-2} + x_1^2
# e_2 = 1 + x_1 + x_1^{-1} + x_1^{-1} * x_2^{-1} + x_1^{-2} + x_1^{-2} * x_2^{-1} + x_1^{-3} + x_1^{-4} + x_1^2 + x_1^3 + x_2 + x_2 * x_1^{-1} +
# +x_2 * x_1^{-2} + x_2 * x_1^{-3} + x_2 * x_1^{-4} + x_2 * x_1^{-5}
# e_3 = 2x_1^{-4} + 3
# e_4 = 1 + 4x_1 + 4x_1^{-1} + 4x_1^{-2} * x_2^{-1} + 4x_1^{-3} + 4x_1^3 + 4x_2 + 4x_2 * x_1^{-2} + 4x_2 * x_1^{-4} + x_1^{-1} * x_2^{-1} +
# +x_1^{-2} + x_1^{-4} + x_1^2 + x_2 * x_1^{-1} + x_2 * x_1^{-3} + x_2 * x_1^{-5}
# e_5 = 1 + 4x_1^{-1} * x_2^{-1} + 4x_1^{-2} * x_2^{-1} + 4x_2 + 4x_2 * x_1^{-1} + 4x_2 * x_1^{-2} + 4x_2 * x_1^{-3} + 4x_2 * x_1^{-4} +
# +4x_2 * x_1^{-5} + x_1 + x_1^{-1} + x_1^{-2} + x_1^{-3} + x_1^{-4} + x_1^2 + x_1^3
# e_6 = 1 + 4x_1 + 4x_1^{-1} + 4x_1^{-1} * x_2^{-1} + 4x_1^{-3} + 4x_1^3 + 4x_2 * x_1^{-1} + 4x_2 * x_1^{-3} + 4x_2 * x_1^{-5} + x_1^{-2} +
# +x_1^{-2} * x_2^{-1} + x_1^{-4} + x_1^2 + x_2 + x_2 * x_1^{-2} + x_2 * x_1^{-4}
# Проверим действительно ли это идемпотенты
sage: [x^2==x for x in pidems]
# [True, True, True, True, True, True]
# Проверим сумму примитивных центральных элементов
sage: sum(pidems)
# () - это единица нашей групповой алгебры
```

A Примеры работы с группами в Sage

Порядок и знак элемента группы перестановок:

```
sage: G = SymmetricGroup(5)
sage: sigma = G([(1,3), (2,5,4)])
sage: sigma.order()
# 6
sage: sigma.sign()
# -1
```

Функция $G.is\_abelian()$ - определяет является ли группа G абелевой.

```
sage: G = DihedralGroup(6)
sage: G.is_abelian()
# False
```

Функция $G.order()$ - вычисляет порядок группы G.

```
sage: G = DihedralGroup(6)
sage: G.order()
# 12
```

Функция $G.list()$ - выводит список всех элементов группы G.

```
sage: G = DihedralGroup(6)
sage: G.list()
# [(), (2,6)(3,5), (1,2)(3,6)(4,5), (1,2,3,4,5,6), (1,3)(4,6), (1,3,5)(2,4,6),
# (1,4)(2,3)(5,6), (1,4)(2,5)(3,6), (1,5)(2,4), (1,5,3)(2,6,4), (1,6,5,4,3,2),
# (1,6)(2,5)(3,4)]
```

Функция $G.random\_element()$ - получает случайный элемент группы G.

```
sage: G = DihedralGroup(6)
sage: G.random_element()
# (2,6)(3,5)
```

Функция $G.cayley\_table()$ - выводит таблицу умножения элементов группы G.

```
sage: G = DihedralGroup(6)
sage: G.cayley_table()
# *  a b c d e f g h i j k l
#  +------------------------
# a| a b c d e f g h i j k l
# b| b a d c f e h g j i l k
# c| c k a e d g f i h l b j
# d| d l b f c h e j g k a i
# e| e j k g a i d l f b c h
# f| f i l h b j c k e a d g
# g| g h j i k l a b d c e f
# h| h g i j l k b a c d f e
# i| i f h l j b k c a e g d
# j| j e g k i a l d b f h c
# k| k c e a g d i f l h j b
# l| l d f b h c j e k g i a
```

По умолчанию для обозначения элементов используются мельнькие латинские буквы. Сами элементы могут быть найдены с помощью функции $G.cayley\_table().row\_keys()$ или $G.cayley\_table().column\_keys()$. Например, для определения пятого элемента, обозначаемого как e, код будет выглядеть следующим образом:

```
sage: G = DihedralGroup(6)
sage: headings = G.cayley_table().row_keys()
sage: headings[4]
# (1,3)(4,6)
```

Функция $G.center()$ - находит центр группы G.

```
sage: G = DihedralGroup(6)
sage: G.center().list()
# [(), (1,4)(2,5)(3,6)]
```

Функция $G.subgroup(elements)$ - вычисляет циклическую подгруппу группы G, порожденную элементами $elements$ группы G. $elements$ - это кортеж из элементов группы G. В листинге ниже показано взятие подгруппы $\langle a \rangle \subseteq G$, где $a \in G$

```
sage: a = G.random_element()
sage: H = G.subgroup([a])
```

Другой пример.

```
# Создаем симметрическую группу S_5 и присваиваем ее переменной G
sage: G = SymmetricGroup(5)
# Создаем элементы sigma = (1,2,3), rho = (1,2) (3,5) этой группы
sage: sigma = G("(1,2,3)")
sage: rho = G("(1,2) (3,5)")
# Создаем подгруппу H = ⟨ σ, ρ ⟩ ⊆ G = S_5
sage: H = G.subgroup([sigma, rho])
# Выводим список элементов H
sage: H.list()
# [(), (2,3,5), (2,5,3), (1,2)(3,5), (1,2,3), (1,2,5), (1,3,2),
# (1,3,5), (1,3)(2,5), (1,5,2), (1,5,3), (1,5)(2,3)]
```

Функция $H.is\_normal(G)$ - определяет является ли группа H нормальной подгруппой для группы G.

```
# Создаем знакопеременную группу A_4
sage: A4 = AlternatingGroup(4)
# Создаем элементы r1 = (1,2) (3,4), r2 = (1,3) (2,4), r3 = (1,4)
sage: r1 = A4("(1,2) (3,4)")
sage: r2 = A4("(1,3) (2,4)")
sage: r3 = A4("(1,4) (2,3)")
# Вычисляем подгруппу H = < r1, r2, r3 > ⊆ A_4
sage: H = A4.subgroup([r1, r2, r3])
# Проверяем является ли H нормальной подгруппой для A_4
sage: H.is_normal(A4)
# True
```

Функция $G.quotient(H)$ - получает группу изоморфную фактор-группе G/H. Воспользуемся предудыщим примером:

```
sage: A4 = AlternatingGroup(4)
sage: r1 = A4("(1,2) (3,4)")
sage: r2 = A4("(1,3) (2,4)")
sage: r3 = A4("(1,4) (2,3)")
sage: H = A4.subgroup([r1, r2, r3])
sage: A4.quotient(H)
# Permutation Group with generators [(1,2,3)]
```

В результате мы получили группу $\langle(1,2,3)\rangle$

Функция $G.is\_simple()$ - определяет является ли группа простой.

```
sage: AlternatingGroup(5).is_simple()
# True
sage: AlternatingGroup(4).is_simple()
# False
```

Функция $G.subgroups()$ - возвращает все подгруппы группы G.

```
sage: s3 = SymmetricGroup(3)
sage: for i in s3.subgroups():
sage:     print i.list()
[()]
[(), (2,3)]
[(), (1,2)]
[(), (1,3)]
[(), (1,2,3), (1,3,2)]
[(), (2,3), (1,2,3), (1,2), (1,3,2), (1,3)]
```

Функция $G.conjugacy\_classes()$ - возвращает все классы сопряженности группы G.

```
# Получаем классы сопряженности и выводим их элементы
sage: s3 = SymmetricGroup(3)
sage: for i in s3.conjugacy_classes():
sage:     print i.list()
[()]
[(2,3), (1,3), (1,2)]
[(1,2,3), (1,3,2)]
```

Остальные методы и их описание указаны в [7].

B Примеры работы с групповыми алгебрами в Sage

Кольцо и группа групповой алгебры:

```
sage: KG = GroupAlgebra(SymmetricGroup(3), GF(5))
sage: KG.base_ring()
# Finite Field of size 5
sage: KG.group()
# Symmetric group of order 3! as a permutation group
```

Случайный элемент групповой алгебры.

```
sage: KG = GroupAlgebra(SymmetricGroup(3), GF(5))
sage: a = KG.random_element()
sage: a
# 2*() + 3*(1,2)
```

Некоторые методы элементов групповой алгебры

```
sage: KG = GroupAlgebra(SymmetricGroup(3), GF(5))
sage: a = KG.random_element()
sage: a
# 2*(1,2,3) + (1,3)
sage: a.is_central()
# False
sage: a.is_idempotent()
# False
sage: a.powers(3)
# [(), 2*(1,2,3) + (1,3), () + 2*(2,3) + 2*(1,2) + 4*(1,3,2)]
```

Порождающие элементы групповой алгебры:

```
sage: KG = GroupAlgebra(SymmetricGroup(3), GF(5))
sage: KG.gens()
# Finite family {(1,2): (1,2), (1,2,3): (1,2,3)}
```

Проверка на коммутативность:

```
sage: KG = GroupAlgebra(SymmetricGroup(3), GF(5))
sage: KG.is_commutative()
# False
```

Проверка на конечность:

```
sage: KG = GroupAlgebra(SymmetricGroup(3), GF(5))
sage: KG.is_finite()
# True
```

Остальные методы и их описание указаны в [8].

C Листинг функции repr_from_gens(elem)

Функция по заданному элементу групповой алгебры, возвращает его представление в виде линейной комбинации образующих элементов группы с коэффициентами из заданного поля.

```
def repr_from_gens(elem):
  ga = elem.parent()
  monomials = elem.monomials()
  mons = elem.monomials()
  coefs = elem.coefficients()
  group_gens = ga.group().gens()

  monomials_symbolic = []
  for m in monomials:
    if m != ga.one():
      monomials_symbolic.append(m.support_of_term().
      word_problem(group_gens, false)[0])
    else:
      monomials_symbolic.append('1')

  result = []
  for i in range(len(coefs)):
    if coefs[i] == 1:
      symb_term = monomials_symbolic[i]
    elif monomials_symbolic[i] == '1':
      symb_term = repr(coefs[i])
    else:
      symb_term = repr(coefs[i])+monomials_symbolic[i]
    result.append(symb_term)

  result = '+'.join(sorted(result))
  return result
```

Список литературы

[1] Кострикин А.И., Введение в алгебру. Часть I. Основы алгебры, 2-е изд. 2001г, 272с.

[2] Кострикин А.И., Введение в алгебру. Часть III. Основные структуры, 2-е изд. 2001г., 272с.

[3] С.Н. Тронин, Введение в теорию групп, 2006г., 80с.

[4] R.A. Ferraz, C. Polcino Milies, Idempotents in group algebras and minimal abelian codes, Finite Fields and their Applications, 2007, V. 13, P. 382-393.

[5] Donald S. Passman, The Algebraic Structure of Group Rings, 2011, 752

[6] Карпов Е.В., Классически полупростые групповые алгебры, дипломная работа, КФУ, 2014г, 44с.

[7] Group theory and Sage, sagemath.org/documentation/html/en/thematic_tutorials/group_theory.html

[8] Group Algebras in Sage, sporadic.stanford.edu/bump/reference-5.4.beta1/sage/algebras/group_algebra_new.html

[9] Учебное пособие по Sage, freetonik.com/sage/tutorial/index.html