Московский государственный технический университет
имени Н. Э. Баумана

А. И. Зобнин, О. В. Соколова

Компьютерная алгебра в системе Sage

учебное пособие

М о с к в а
Издательство МГТУ им. Баумана
2011

Введение

Это учебное пособие было написано по мотивам практических занятий, проведенных авторами в 2010–2011 гг. на факультете «Информатика и системы управления» в МГТУ им. Н. Э. Баумана и на механико-математическом факультете МГУ им. М. В. Ломоносова на 3-м курсе в течение осеннего семестра.

Компьютерная алгебра — современная дисциплина, изучающая алгоритмы преобразования математических выражений в символьном виде. В этом смысле она противопоставляется численным методам. Мы предполагаем, что читатели знакомы с основами линейной алгебры (матрицы, системы линейных уравнений, определители, линейные операторы) и с основными алгебраическими структурами (группами, кольцами, полями, векторными пространствами) в объеме стандартного университетского курса. Кроме того, мы рассчитываем, что читатель владеет в некотором смысле навыками программирования.

В качестве системы компьютерной алгебры мы выбрали систему Sage. Мы рассматриваем работу с символьными выражениями в этой системе, вычисления в различных алгебраических структурах (группах, кольцах, полях, векторных пространствах), работу с матрицами и многочленами. Особое внимание уделено задаче исследования систем алгебраических уравнений. К сожалению, мы не останавливаемся на задачах символьного дифференцирования и интегрирования и на алгоритмах факторизации многочленов. Пособие не является полным справочником по Sage, а представляет из себя обзор возможностей этой системы символьных вычислений, а также круга алгоритмических задач, возникающих в алгебре. Мы рассчитываем, что читатель при необходимости сможет обратиться к справочной системе.

Пособие состоит из 5 глав, которые содержат краткую теоретическую информацию и примеры использования функций системы компьютерной алгебры. В конце глав приведены упражнения и задачи. В упражнениях требуется решить с помощью компьютера данную конкретную задачу (часто вполне решаемую и вручную), либо самостоятельно разобраться с функционалом системы. В задачах же, в отличие от упражнений, требуется составить общий алгоритм решения и записать его в виде отдельной подпрограммы.

А. И. Зобнин, О. В. Соколова
Москва, 2011

Оглавление

Глава 1

Основы работы с Sage

1.1 Система компьютерной алгебры Sage

Система компьютерной алгебры Sage (http://sagemath.org) сравнительно молодая. Ее первая версия появилась в 2005 году, но к настоящему времени проект Sage успел хорошо себя зарекомендовать. Сейчас свой вклад в развитие Sage вносят около 200 математиков и программистов со всего мира как на добровольной основе, так и за счет специализированных грантов. Координирует усилия по развитию системы проф. Уильям Штейн из университета Вашингтона.

Sage свободно распространяется на условиях лицензии GNU Public License; ее исходные коды открыты. Своей целью разработчики системы выбрали *создание открытой и свободно распространяемой альтернативы для систем Magma, Maple, Mathematica и MATLAB.* Чтобы проделать такую работу с нуля, потребовались бы тысячи человеко-часов. С другой стороны, уже существует большое количество эффективных открытых математических пакетов, которые можно было бы интегрировать в систему. Однако такие пакеты написаны на разных языках (в основном, C, C++, Fortran, Python) и имеют разный интерфейс и назначение. Поэтому с самого начала система Sage задумывалась как единый и общий интерфейс к различным существующим свободным пакетам компьютерной алгебры, таким как GAP, Pari, Singular и Maxima. Сейчас Sage объединяет около 80 как узкоспециализированных пакетов, так и систем компьютерной алгебры общего назначения. Пользователь, работая в Sage, может легко переключаться между ними, а при совершении стандартных операций даже не подозревать, с помощью каких именно пакетов эти операции выполняются.

В качестве исходного языка программирования в Sage выбран популярный язык Python («питон»), хотя критичные по времени выполнения блоки кода реализованы на C/C++. Этот выбор оказался очень удачным по ряду причин:

— Вместо того, чтобы создавать новый язык (как это было сделано в системах Magma, Maple, Mathematica, MATLAB, GP/PARI, GAP, Macaulay 2 и т. д.), разработчики Sage выбрали уже существующий популярный язык программирания.

— Python — чрезвычайно выразительный язык. Он позиционируется как язык для быстрой и удобной разработки и прототипирования. Python является интерпретируемым объектно-ориентированным языком с динамической типизацией, сборкой мусора и элементами функционального программирования.

— К языку прилагается богатая стандартная библиотека. Кроме того, в свободном доступе имеется множество пакетов по численным методам, линейной алгебре, двумерной и трехмерной графике, распределенным вычислениям, поддержке баз данных и т.д., которые могут заинтересовать пользователей Sage.

— Python — кроссплатформенный язык. Он поставляется с исходными кодами и может быть собран на любой платформе.

— Python имеет богатые встроенные средства для документации кода, управления памятью, обработки исключений, отладки программ, тестирования и т. д.

Несмотря на то, что значительная часть Sage написана на Python'е, специализированных знаний этого языка от читателей не требуется. Все необходимые комментарии по синтаксису языка будут даны в этом пособии. Изучить язык Python подробнее можно с помощью ресурсов сайта http://python.org.

1.2 Установка Sage

Основная разработка Sage ведется для Linux-подобных операционных систем.

Существует несколько способов установить Sage:

1) Загрузить и собрать систему из исходных кодов (на платформах Linux, OS X, Solaris).

2) Загрузить скомпилированные (исполняемые) файлы (на платформах Linux, OS X, Solaris).

3) Загрузиться с компакт-диска Live CD, содержащего операционную систему Linux и Sage. Эта возможность позволяет изучить работу с Sage, не устанавливая ее на компьютер. Образ диска можно скачать с сайта Sage.

4) Хотя полноценной сборки Sage для Windows не существует, запустить Sage в Windows можно через специльную «виртуальную машину», эмулирующую Linux (например, VirtualBox).

5) Можно воспользоваться online-версией Sage по адресу http://sagenb.org. В этом случае Sage работает на удаленном сервере, а пользователь взаимодействует с ней через обычный браузер. Такой способ имеет ряд ограничений (например, по объему доступной памяти).

Работать с Sage можно в различных режимах:

— Графический интерфейс (notebook): работа с Sage ведется через браузер. При этом Sage может быть запущена как локально, так и удаленно. Именно в этом режиме работает online-версия Sage.

— Режим командной строки: работа с Sage ведется в консоли в текстовом режиме. Графика и сложные формулы при необходимости могут отображаться в отдельном окне.

— Программы, написанные для Sage, можно запускать отдельно с помощью интерпретатора Sage из консоли.

— Библиотеку Sage можно подключать в скрипты на Python'е.

Для начального знакомства мы рекомендуем воспользоваться графическим интерфейсом и работой с удаленным сервером через браузер на сайте http://sagenb.org, хотя работа с системой через командную строку в терминале тоже достаточно проста и удобна. Для постоянного же использования удобнее полностью установить Sage на компьютер.

1.3 Присваивание. Арифметические операции

Далее мы рассмотрим базовые конструкции системы Sage и языка Python. Мы опустим некоторые детали синтаксиса Python'а, отсылая читателя к справочным руководствам и сайту http://docs.python.org.

Мы будем иллюстрировать работу с Sage в режиме диалога с пользователем (в «режиме калькулятора»). В наших примерах строка, начинающаяся с приглашения `sage:`, обозначает команду, введенную пользователем. Следом за такой строкой будет идти результат выполнения команды.

Каждая команда Sage в режиме диалога пишется с новой строки. Для ее выполнения следует нажимать `Shift + Enter`. В командной строке можно также просто нажимать `Enter`.

Оператор `=` обозначает присваивание, как и в языке C. Однако в отличие от C этот оператор не возвращает никакого значения, а просто связывает имя, стоящее слева, со значением, записанным справа. В дальнейшем это имя можно будет привязать к другому значению (возможно, совершенно другого типа) с помощью нового присваивания, или вообще «отвязать» от значения командой `del`:

```
sage: a = 5
sage: a
5
sage: a = "hello"
sage: a
hello
sage: del a
sage: a
NameError: name 'a' is not defined
```

Последняя команда завершилась ошибкой (исключением) `NameError`, поскольку мы «отвязали» имя `a` от значения `"hello"`. Sage автоматически освобождает память, занимаемую значением, если обнаруживает, что с этим значением больше не связано никаких имен. Эта операция называется *сборкой мусора*.

Обозначения для операторов сравнения `==`, `!=`, `<=`, `>=`, `<` и `>` такие же, как и в C. В отличие от других языков программирования, в Python'е допускаются двойные неравенства:

```
sage: 2 == 2
True
sage: 2 == 3
False
sage: 2 < 3
True
sage: a == 5
True
sage: a != 5
False
sage: 2 <= a < 7
True
```

Обратите внимание, что константы `True` и `False` пишутся с заглавной буквы. Язык Python чувствителен к регистру букв.

Приведем пример использования основных арифметических операций. Кроме стандартных `+`, `-`, `*` и `/` в Sage добавлены операция возведения в степень, которая обозначается `^` или `**` и операция `//` нахождения неполного частного от деления целых чисел. Операция `%` возвращает остаток от деления первого целого числа на второе.

Знак `#` в примерах обозначает начало комментария, который простирается до конца строки.

```
sage: 2+2*2
6
sage: 10/4
```

```
5/2
sage: 2**3    # возведение в степень
8
sage: 2^3     # ^ является синонимом для ** (в отличие от Python'a)
8
sage: 10 % 3  # остаток от деления целых чисел
1
sage: 10//4   # неполное частное от деления целых чисел (в отличие от Python'a)
2
sage: 4 * (10 // 4) + 10 % 4 == 10
True
sage: 3^2*4 + 2%5
38
```

Как видно, приоритет операций совершенно естественный.

Язык Sage является надстройкой над языком Python. Это означает, что интерпретатор Sage, прежде чем выполнить команду, преобразует ее в корректное выражение Python'a, возможно, заменяя некоторые конструкции. Так, операция `^` в Python'е обозначает побитовое *исключающее или*, но специальный препроцессор Sage намеренно ее подменяет и преобразует в операцию `**` возведения в степень, так как для системы компьютерной алгебры такой смысл операции `^` более естественный. Подобные примеры встретятся нам и в дальнейшем (например, при работе с кольцами многочленов).

Обратите внимание, что в конце строк не требуется ставить точки с запятой; обычно каждая команда заканчивается переводом строки. Однако если в одной строке размещено несколько команд, они должны быть разделены:

```
sage: a = 5; b = a + 3; c = b^2; c
64
```

Можно разместить одну команду на нескольких строках с помощью обратной косой черты:

```
sage: 2 + \
...       3
5
```

Язык Python обладает динамической типизацией. Это значит, что значение, присвоенное переменной, обладает связанным с ним типом. Но сама переменная может принимать значения любого типа. При этом нигде не нужно явно указывать этот тип:

```
sage: a = 5    # a - целое число
sage: type(a)
<type 'sage.rings.integer.Integer'>
sage: a = 5/3  # теперь a рациональное
sage: type(a)
<type 'sage.rings.rational.Rational'>
sage: a = 'hello'  # теперь a - строка
sage: type(a)
<type 'str'>
```

Для сравнения отметим, что язык программирования C имеет статическую типизацию и в корне отличается от Python'a: в нем тип переменной оговаривается в самом начале и не может изменяться в течение жизни этой переменной.

В этом примере нам встретилась строковая константа `"hello"`. Заметим, что значения строковых констант могут записываться как в апострофах, так и в кавычках. Это бывает

удобно, если в тексте строки встречается кавычка или апостроф. В любом случае ввести кавычку или апостроф можно через экранирующую обратную косую черту: \".

Строка очень похожа на структуру «список», которую мы рассмотрим ниже.

1.4 Основные структуры данных

Базовой структурой данных в Sage является **список** (list), называемый также динамическим массивом. Список задается перечислением элементов через запятую в квадратных скобках. Элементы списка при этом не обязаны быть однотипными:

```
sage: v = [1, "hello", 2/3, sin(x^3)]
sage: v
[1, 'hello', 2/3, sin(x^3)]
```

Как и в языке C, элементы списка индексируются с нуля:

```
sage: v[0]
1
sage: v[3]
sin(x^3)
```

Функция `range` создает список из элементов полуинтервала:

```
sage: w = range(2,10)
sage: w
[2, 3, 4, 5, 6, 7, 8, 9]
```

Продемонстрируем некоторые стандартные операции со списками:

```
sage: len(v)        # получить размер списка
4
sage: v[1:3]        # получить подсписок элементов с индексами 1 и 2
[2/3, sin(x^3)]
sage: v[-1]         # получить первый с конца элемент
sin(x^3)
sage: 2/3 in v      # проверить наличие элемента в списке
True
sage: 2/3 not in v
False
sage: v.append(1.5) # добавить элемент в конец
sage: v
[1, 'hello', 2/3, sin(x^3), 1.50000000000000]
sage: del v[1]      # удалить второй элемент
sage: v
[1, 2/3, sin(x^3), 1.50000000000000]
sage: w
[2, 3, 4, 5, 6, 7, 8, 9]
sage: max(w), min(w)
9, 2
sage: v+w           # объединить списки
[1, 2/3, sin(x^3), 1.50000000000000, 2, 3, 4, 5, 6, 7, 8, 9]
```

С помощью метода `sort` можно отсортировать список по возрастанию:

```
sage: u = [9, 3, 9, 1, 6, 1, 1]
sage: u.sort()
sage: u
[1, 1, 1, 3, 6, 9, 9]
```

Перечисления (tuples) по сути являются неизменяемыми списками. Они задаются в круглых скобках:

```
sage: t = (10, 20, 30)
sage: t[1]
20
```

Там, где возможно, скобки можно опускать. Кроме того, перечисления из имен могут стоять в левой части присваивания. Например, обмен значений двух переменных на языке Python выглядит особенно изящно:

```
sage: a, b = 1, 2  # присвоили значения сразу двум переменным
sage: a, b = b, a  # поменяли значения переменных местами
sage: a, b
(2, 1)
```

Преимущества перечислений — более эффективные хранение и доступ, а также возможность использовать их в качестве ключей в словарях.

Рассмотрим теперь **словарь** или ассоциативный массив (dictionary). Словарь задается как множество пар вида «ключ: значение», где каждый ключ уникален. Ключами в словаре могут являться любые неизменяемые объекты. Порядок пар в словаре при этом не определен. Для обращения к элементу по его ключу используются все те же квадратные скобки:

```
sage: d = {'hi':-2,  3/8:pi,   e:pi} # словарь из трех пар
sage: d['hi']
-2
sage: d[e]
pi
sage: 123 in d  # проверка на наличие ключа
False
sage: d[123] = 4.5  # добавление элемента с новым ключом
sage: del d[3/8]  # удаление элемента
sage: d
{'hi':-2, e:pi, 123:4.5}
```

Структуру данных «**множество**» (set) читателям предлагается изучить самостоятельно.

1.5 Функции

Рассмотрим несколько примеров встроенных математических функций:

```
sage: sqrt(3.4)
1.84390889145858
sage: sin(5.135)
-0.912021158525540
sage: sin(pi/3)
1/2*sqrt(3)
```

Как мы видим, в специальных случаях математические функции возвращают «точные», а не приближенные значения. Чтобы получить численное приближение, следует воспользоваться функцией `n`, либо ее псевдонимом `numerical_approx`:

```
sage: exp(2)
e^2
sage: n(exp(2))
7.38905609893065
sage: sqrt(pi).numerical_approx()
1.77245385090552
```

По умолчанию используется точность в 53 бита. С помощью необязательных аргументов `digits` и `prec` можно управлять числом десятичных цифр в ответе и числом бит в двоичном представлении числа:

```
sage: n(sin(10), digits=5)
-0.54402
sage: n(sin(10), digits=10)
-0.5440211109
sage: numerical_approx(pi, prec=200)
3.1415926535897932384626433832795028841971693993751058209749
```

Создать новую функцию в Sage можно с помощью ключевого слова `def`:

```
sage: def is_even(n):   # является ли число четным?
...       return n%2 == 0
...
sage: is_even(2)
True
sage: is_even(3)
False
```

Обратите внимание, что нигде не нужно указывать типы аргументов и возвращаемого значения. Тело функции должно быть оформлено с отступом, который в Python'е является частью языка (вместо операторных скобок). Эта замечательная особенность Python'а сразу приучает к хорошему стилю программирования: программа, оформленная без отступов, просто не будет работать. В роли отступа может выступать табуляция или фиксированное число пробелов. В командной строке интерпретатора Python'а при вводе двоеточия отступ появляется автоматически.

Для параметров можно указывать значения по умолчанию:

```
sage: def is_divisible_by(number, divisor = 2):
...       return number%divisor == 0
sage: is_divisible_by(6, 2)
True
sage: is_divisible_by(6)
True
sage: is_divisible_by(6, 5)
False
```

Имена параметров можно также явно указывать при вызове:

```
sage: is_divisible_by(6, divisor=5)
False
sage: is_divisible_by(divisor=2, number=6)
True
```

В Python'е допускается объявлять функцию от произвольного числа аргументов. В этом случае в саму функцию будет передан список этих аргументов:

```
sage: def f(x, y, *args):
...       print args

sage: f(1, 2, 3, 4, 5)
[3, 4, 5]
```

Функции в Python'е являются полноправными объектами: их можно присваивать, передавать в качестве параметров и т. д.:

```
sage: s = sin
sage: s(pi/2)
1
```

Пусть требуется отсортировать список чисел по убыванию. Создадим для этого пользовательскую функцию, которая будет сравнивать элементы. В зависимости от результата сравнения, функция должна возвращать отрицательное число, 0, или положительное число.

```
sage: def compare (x, y):
...       return int(y - x)

sage: u = [1, 2, 6, 7, 8, 3, 2]
sage: u.sort(compare)
sage: u
[8, 7, 6, 4, 2, 2, 1]
```

Заметим, что приведение типа в функции `compare` требуется потому, что в Sage целые числа не имеют тип `int`, а автоматически преобразуются в объекты класса `Integer`. Поэтому разность `y-x` будет также иметь тип `Integer`, в то время как функция сортировки ожидает от `compare` результат именно типа `int`.

Тот же пример можно переписать с помощью так называемой лямбда-функции, определяя функцию прямо в месте ее использования с помощью ключевого слова `lambda`:

```
sage: u = [1, 9, 2, 4, 3, 3, 7]
sage: u.sort(lambda x, y: int(y - x))
sage: u
[9, 7, 4, 3, 3, 2, 1]
```

Лямбда-выражения — важный элемент функционального стиля программирования. Другими элементами являются классические функции `map` и `reduce`. Функция `map` позволяет применить свой первый аргумент (функцию) к каждому элементу второго аргумента (списку):

```
sage: seq = [x, y, x^2, y^2, 1]
sage: map(sin, seq)
[sin(x), sin(y), sin(x^2), sin(y^2), sin(1)]
```

Функция `reduce` «сворачивает» список в некоторую величину, применяя свой первый аргумент (функцию) к частичному результату свертки и очередному элементу списка. Отдельно указывается начальное значение «свертки».

```
sage: reduce(lambda x,y: x+y, [1,2,3,4], 0)  # сумма элементов
10
sage: reduce(lambda x,y: x*y, [1,2,3,4], 1)  # произведение
24
sage: reduce(lambda x,y: x+y,['h','e','l','l','o'],'') # конкатенация
'hello'
```

Впрочем, приведенные примеры можно переписать проще:

```
sage: product([1,2,3,4])
10
sage: sum([1,2,3,4])
24
sage: "-".join(['h','e','l','l','o'])
'hello'
```

1.6 Условный оператор

Условный оператор `if` является составным. Тело такого оператора (как и других составных операторов и функций в Python'е) должно быть оформлено с отступом:

```
sage: if x < 0:
...       y = -x
...   else:
...       y = x
```

В языке Python нет оператора множественного выбора (аналога `switch` в C). Вместо этого используются секции `elif` условного оператора:

```
sage: if z == 1:
...       print "One"
...   elif z == 2:
...       print "Two"
...   elif z == 3:
...       print "Three"
...   else:
...       print "Many"
```

Логические операции И, ИЛИ и НЕ записываются с помощью ключевых слов `and`, `or` и `not`:

```
sage: x = 61
sage: if x > 1 and not is_prime(x):
...      print "x is composite"
```

Чтобы явно вычислить логическое выражение, надо воспользоваться приведением к типу `bool`:

```
sage: bool (e^pi < pi^e)
False
```

В этом примере мы воспользовались тем, что константы `e` и `pi` уже определены в Sage. Выражение `e^pi < pi^e` имеет тип `sage.symbolic.expression.Expression` и представляет из себя символьное выражение.

1.7 Циклы

Простейший цикл в Python'е — цикл с предусловием `while`. Тело цикла также должно быть оформлено с отступом:

```
sage: i = 0
sage: while i < 3:
...       print i
...       i += 1
0
1
2
```

В этом примере мы используем простейший счетчик. Заметьте, что в Python'е нет операции `++`, так что увеличивать счетчик следует с помощью `+=`.

Перепишем этот пример с помощью цикла `for`. Цикл с параметром `for` всегда выполняется по некоторому диапазону (списку, последовательности, строке, файлу). Например, конструкция, которая в C++ или в Java была бы записана как `for (int i = 0; i < 3; ++i) ...`, на Python'е будет выглядеть так:

```
sage: for i in range(3):  # итерации по элементам списка [0, 1, 2]
...       print i
0
1
2
```

Можно также указать шаг при построении списка:

```
sage: for i in range(1,6,2):
...       print i
1
3
5
```

Текстовая строка в Sage напоминает тип `list`: с ней можно работать так же, как и со списком символов.

```
sage: for c in 'abcde':
...       print c
a
b
c
d
e
```

При работе с циклами можно использовать ключевые слова `break` и `continue` для выхода из цикла и перехода на следующую итерацию.

1.8 Справочная информация

Sage обладает встроенной системой документации. Чтобы получить справку о какой-либо функции, наберите ее имя и знак вопроса:

```
sage: tan?
Type:        <class 'sage.calculus.calculus.Function_tan'>
Definition:  tan( [noargspec] )
Docstring:   The tangent function
...
```

Точно также можно проверить с каким типом связана переменная:

```
sage: a = 'hello'
sage: a?
<type 'str'>
```

Для посмотра кода той или иной функции следует набрать имя функции с двумя вопросительными знаками.

В оболочку Sage также встроена система автодополнения (tab completion): наберите несколько первых букв имени функции и нажмите Tab. Например, если набрать `ta` + Tab, то Sage предложит варианты `tachyon`, `tan`, `tanh`, `taylor`. Получить список всех полей и функций данного объекта можно с помощью команды `dir`.

Таким образом в Sage можно легко найти нужную функцию, если вы забыли ее имя.

Глава 2

Символьные выражения и алгебраические структуры

2.1 Символьные выражения

В этом разделе мы познакомимся с символьными вычислениями в Sage, то есть, с формальными (а не численными) преобразованиями выражений. Заметим для начала, что сами по себе переменные в Sage не являются по умолчанию «символами»:

```
sage: f = z^2 + 2*z + 1
Traceback (click to the left of this block for traceback)
...
NameError: name 'z' is not defined
```

Если же переменная `z` до этого была связана с каким-либо значением (например, числовым), то в этом случае будет вычисляться значение такого выражения. Для того, чтобы выражение `z^2 + 2*z + 1` стало именно «символьным», необходимо, чтобы переменная `z` была объектом специального класса `Expression`. Проще всего это сделать с помощью специальной функции `var`:

```
sage: var('z')
z
sage: f = z^2 + 2*z + 1; f
z^2 + 2*z + 1
sage: var('n xx yy zz')
(n, xx, yy, zz)
sage: f = xx^n + yy^n + zz^n; f
xx^n + yy^n + zz^n
sage: type(f)
< type 'sage.symbolic.expression.Expression' >
sage: parent(f)
Symbolic Ring
```

Символьное выражение является элементом специальной «алгебраической структуры» — кольца символьных выражений. Стандартные функции, такие как `sin`, `ln` и т. д., примененные к символьному выражению, снова дают символьное выражение.

По умолчанию в Sage определена символьная переменная `x`, поэтому ей можно пользоваться без предварительного объявления с помощью `var`:

```
sage: diff (cos(x^3))  # продифференцировать
-3*x^2*sin(x^3)
```

```
sage: diff (cos(x^3), x, 2)  # вторая производная
-9*x^4*cos(x^3) - 6*x*sin(x^3)
sage: integral(sin(x), x)  # неопределенный интеграл
-cos(x)
sage: integral(sin(x), x, 0, pi)  # определенный интеграл
2
```

Иногда требуется преобразовать символьное выражение: разложить на множители или раскрыть скобки в выражение. Для этих целей используются функции `factor` и `expand`:

```
sage: expand((x-1)*(x-2))   # раскрыть скобки
x^2 - 3*x + 2
sage: factor(x^2 - 2*x + 1) # разложить на множители
(x - 1)^2
```

С помощью функций `operator` и `operands` можно посмотреть на дерево выражения:

```
sage: var('x y')
(x, y)
sage: f=x^2+2*x+1
sage: g=f.operator()
sage: g
<built-in function add>
sage:g(1,3)  # применить функцию сложения к операндам 1 и 3
4
sage: f.operands() # список операндов
[x^2, 2*x, 1]
sage: reduce(f.operator(),f.operands(),0)  # восстановить сумму
x^2 + 2*x + 1
```

С помощью функции `solve` можно решать отдельные уравнения или системы символьных уравнений:

```
sage: x = var('x')
sage: solve(x^2 - 5*x + 6, x)
[x == 3, x == 2]
sage: x, y = var('x, y')
sage: solve([x+y==6, x-y==4], x, y)
[[x == 5, y == 1]]
sage: var('x,y,z,a')
(x, y, z, a)
sage: eqns = [x + z == y, 2*a*x - y == 2*a^2, y - 2*z == 2]
sage: solve(eqns, x, y, z)
[[x == a + 1, y == 2*a, z == a - 1]]
```

По умолчанию функция `var` создает комплекснозначную переменную. Однако можно сузить область значений переменной, и этой информацией Sage сможет пользоваться в дальнейшем:

```
sage: y = var('y', domain='real'); y.conjugate() # комплексное сопряжение
y
sage: y = var('y', domain='positive'); y.abs()   # модуль
y
```

При создании переменной можно также указать стиль ее оформления в выражениях в графическом интерфейсе с помощью соответствующей LaTeX-команды:

```
sage: var('aij', latex_name="a_{i,j}")
```

$$a_{i,j}$$

С помощью функции `restore` можно восстановить старое значение встроенной переменной:

```
sage: var('QQ RR')
(QQ, RR)
sage: QQ
QQ
sage: restore('QQ')
sage: QQ
Rational Field
```

2.2 Алгебраические структуры

При работе с матрицами, векторами, многочленами и т. д. частно бывает полезно (а иногда и необходимо) задать кольцо (пространство, алгебру), в котором происходит работа. Вот примеры основных колец в Sage:

— `ZZ` или `IntegerRing()` (целые числа);

— `Zmod(n)` или `IntegerModRing(n)` (кольцо вычетов);

— `QQ` или `RationalField()` (рациональные числа);

— `AA` или `AlgebraicRealField()` (алгебраические числа);

— `RR` или `RealField()` (действительные числа);

— `CC` или `ComplexField()` (комплексные числа).

Эти конструкции позволяют создать алгебраическую структуру (кольцо, поле и т. д.) как объект соответствующего класса в Sage. Зафиксировав структуру, можно будет создавать объекты, соответствующие элементам этой структуры. Проиллюстрируем это на примерах:

```
sage: RationalField()
Rational Field
sage: QQ
Rational Field
sage: 1/2 in QQ
True
sage: 1.2 in QQ
True
sage: pi in QQ
False
sage: pi in RR
True
sage: sqrt(2) in QQ
False
sage: sqrt(2) in CC
True
```

Рассмотрим другие кольца:

```
sage: GF(3)  # конечное поле
Finite Field of size 3
# если это не простое поле, то нужно указать имя образующего элемента
sage: GF(27, 'a')
Finite Field in a of size 3^3
sage: Zp(5)  # кольцо 5-адических чисел
5-adic Ring with capped relative precision 20
sage: sqrt(3) in QQbar # поле алгебраических чисел
True
```

Построим кольцо матриц над другим кольцом:

```
sage: R = IntegerModRing(51)  # кольцо вычетов
sage: M = MatrixSpace(R,3,3)  # алгебра матриц 3x3
```

Чтобы задать пространство матриц 3×4, следует написать `MatrixSpace(R,3,4)`. Для квадратных матриц последний аргумент можно опустить, так что `MatrixSpace(R,3)` эквивалентно `MatrixSpace(R,3,3)`.

Теперь легко можно получить 0 и 1 в алгебре матриц:

```
sage: M(0)
[0 0 0]
[0 0 0]
[0 0 0]
sage: M(1)
[1 0 0]
[0 1 0]
[0 0 1]
sage: 5*M(1)
[5 0 0]
[0 5 0]
[0 0 5]
```

Создадим два кольца многочленов: `PRQ` — над полем рациональных чисел и `PRR` — над полем действительных чисел.

```
sage: PRQ.<t> = PolynomialRing(QQ)
sage: PRR.<z> = PolynomialRing(RR)
```

Теперь разложим один и тот же многочлен, но принадлежащий разным кольцам, на множители:

```
sage: factor(t^2-2)
t^2 - 2
sage: factor(z^2-2)
(z - 1.41421356237310) * (z + 1.41421356237310)
```

Рассмотрим кольцо многочленов над полем из 97 элементов:

```
sage: R = PolynomialRing(GF(97),'x')
sage: x = R.gen()
sage: f = x^2 + 7
sage: f in R
True
```

Построим теперь факторкольцо этого кольца:

```
sage: R = PolynomialRing(GF(97),'x')
sage: x = R.gen()
sage: S = R.quotient(x^3 + 7, 'a')
sage: a = S.gen()
sage: S
Univariate Quotient Polynomial Ring in a over Finite Field of size 97 with
modulus x^3 + 7
sage: S.is_field()
True
sage: a in S
True
sage: x in S  # имеется естественное отображение x в S
True
sage: S.polynomial_ring()
Univariate Polynomial Ring in x over Finite Field of size 97
sage: S.modulus()
x^3 + 7
sage: S.degree()
3
```

Вычисления в факторкольце можно производить и непосредственно, без его построения:

```
sage: R = PolynomialRing(GF(97),'x')
sage: x = R.gen()
sage: f = x^7+1
sage: (f^3).quo_rem(x^7-1)
(x^14 + 4*x^7 + 7, 8)
```

В заключение рассмотрим для примера конструкцию свободной ассоциативной алгебры.

```
sage: R.<x,y,z> = FreeAlgebra(QQ,3)
sage: R
Free Algebra on 3 generators (x, y, z) over Rational Field
sage: (x+y+z)^3  # произведение в свободной алгебре некоммутативно
x^3 + x^2*y + x^2*z + x*y*x + x*y^2 + x*y*z + x*z*x + x*z*y + x*z^2 +
y*x^2 + y*x*y + y*x*z + y^2*x + y^3 + y^2*z + y*z*x + y*z*y + y*z^2 +
z*x^2 + z*x*y + z*x*z + z*y*x + z*y^2 + z*y*z + z^2*x + z^2*y + z^3
```

2.3 Упражнения

1. Изучите самостоятельно функцию `simplify`, упрощающую символьное выражение. Рассмотрите также ее спецефические варианты `simplify_exp`, `simplify_factorial`, `simplify_log`, `simplify_radical`, `simplify_rational` и `simplify_trig`.

2. Изучите самостоятельно функции `sum` и `product` и продемонстрируйте примеры их использования.

3. Изучите самостоятельно функцию `plot`.

4. Что будет, если в Sage выполнить команду `f(x=g)`, где `f` и `g` — символьные выражения, а x — символьная переменная? Какой тип имеет выражение `x=g`?

5. Рассмотрите самостоятельно функции алгебраических структур, начинающиеся с префикса `is_` (например, функцию `is_field` у колец). Задайте в Sage какие-либо алгебраические структуры, для которых эти функции будут возвращать разные значения.

2.4 Задачи

1. Напишите функцию, определяющую количество операций умножения, необходимых для вычисления символьного выражения по его дереву.

2. Напишите функцию, строящую по заданному выражению его обратную польскую запись, то есть строку, в которой знак операции записывается после операндов.

3. Напишите функцию `mgcd`, принимающую на вход произвольное число аргументов и вычисляющую НОД всех этих аргументов. (Указание: можно передать стандартному алгоритму `gcd` список аргументов в виде одного параметра, либо воспользоваться стандартной функцией `reduce`).

4. Не используя функций `gcd` и `xgcd`, напишите свою версию расширенного алгоритма Евклида.

5. Напишите функции, получающие на вход многочлен от нескольких переменных и вычисляющие его формальную производную и неопределенный интеграл по заданной переменной. Встроенные функции `diff` и `integrate` использовать при этом запрещается.

6. Реализуйте работу с целыми гауссовыми числами (элементами кольца $\mathbb{Z}[i]$) как с вычетами кольца многочленов по модулю многочлена x^2+1. Напишите собственную функцию, которая делила бы одно такое число на другое с остатком и возвращала бы какие-нибудь неполное частное и остаток. Напишите с ее помощью функцию, вычисляющую НОД двух целых гауссовых чисел. Для проверки разделите $a = 40 + i$ на $b = 3 - i$ с остатком и найдите НОД чисел чисел $20 + 9i$ и $11 + 2i$. Продемонстрируйте, что при присоединении к $\mathbb{Z}$ чисел вида $\sqrt{-n}$ при некоторых n кольцо может уже не быть евклидовым и даже не быть факториальным.

7. Проверьте при $n \leqslant 1000$, что $\mathbb{Z}[i]/\langle n\rangle$ является полем тогда и только тогда, когда n — простое число, не равное сумме квадратов двух целых чисел.

8. Напишите свою функцию, строящую поле разложения заданного многочлена над заданным полем с помощью последовательного присоединения корней этого многочлена.

9. Не используя функций `is_field`, `is_maximal`, `is_radical` и т. д., напишите свои функции `factor_is_field(f)`, `factor_is_domain(f)` и `factor_has_nilpotents(f)`, которые по заданному целому числу (многочлену) `f` выясняли бы, является ли соответствующее факторкольцо кольца целых чисел (многочленов от одной переменной над полем) по идеалу $\langle f\rangle$ полем, является ли оно областью целостности и имеет ли оно нильпотентные элементы.

Глава 3

Линейная алгебра

3.1 Матрицы

Создать матрицу можно с помощью функции `matrix` (или ее псевдонима `Matrix`). Эта функция должна получить в том или ином виде элементы матрицы и (при необходимости) ее размеры и кольцо коэффициентов. Например:

```
sage: A = matrix(3)   # создаем нулевую квадратную матрицу размера 3x3
sage: B = matrix(3,4) # создаем нулевую матрицу размера 3x4
sage: C = matrix([[1,2,3],[3,2,1],[1,1,1]]) # конкретная матрица
```

В последнем случае матрица просто задается списком строк. Каждая строку можно задать по отдельности как вектор:

```
sage: v1 = vector([1,1,-4])
sage: v2 = vector([2,3,4])
sage: A = matrix([v1, v2])
sage: A
[1, 1,-4]
[2, 3, 4]
```

При задании матрицы иногда бывает нужно указать, в каком кольце находятся ее коэффициенты:

```
sage: AZ = matrix(ZZ, [[2,0], [0,1]])
sage: AQ = matrix(QQ, [[2,0], [0,1]])
sage: AR = matrix(RR, [[2,0], [0,1]])
```

Вместо списка элементов матрицы можно указать функцию, которая по номеру строки и столбца генерирует эти элементы. Для этого удобно воспользоваться лямбда-выражением:

```
sage: m = matrix(3, 3, lambda i,j: 1/(i+j+1))
sage: m
[  1 1/2 1/3]
[1/2 1/3 1/4]
[1/3 1/4 1/5]
```

Задать матрицу с символьными элементами можно следующим образом:

```
sage: R = PolynomialRing(QQ, 9, 'x')
sage: A = matrix(R, 3, 3, R.gens()); A
[x0 x1 x2]
```

```
[x3 x4 x5]
[x6 x7 x8]
sage: det(A)
-x2*x4*x6 + x1*x5*x6 + x2*x3*x7 - x0*x5*x7 - x1*x3*x8 + x0*x4*x8
```

Рассмотрим элементарные вычисления с матрицами.

```
sage: A = matrix([[1,2,3],[3,2,1],[1,1,1]])
sage: w = vector([1,1,-4])
sage: w*A  # умножить вектор-строку на матрицу
(0, 0, 0)
sage: A*w  # умножить матрицу на вектор-столбец
(-9, 1, -2)
```

Сложение и умножение матриц можно выполнять с помощью обычных операций `+` и `*`. Чтобы обратить матрицу, достаточно возвести ее в степень -1. Определитель, след, ранг матрицы могут быть получены с помощью функций `determinant` (или `det`), `trace`, `rank` и т. д.

Вычислим, например, степени матриц над кольцом вычетов по модулю 51:

```
sage: R = IntegerModRing(51)
sage: M = MatrixSpace(R, 3, 3)
sage: A = M([1,2,3, 4,5,6, 7,8,9])
sage: A^1000*A^1007
[ 3  3  3]
[18  0 33]
[33 48 12]
sage: A^2007  # проверка
[ 3  3  3]
[18  0 33]
[33 48 12]
```

Можно получить список строк или столбцов матрицы с помощью функций `rows()` и `columns()`:

```
sage: M = MatrixSpace(GF(2),4,8)
sage: A = M([1,1,0,0, 1,1,1,1, 0,1,0,0, 1,0,1,1,
...          0,0,1,0, 1,1,0,1, 0,0,1,1, 1,1,1,0])
sage: A
[1 1 0 0 1 1 1 1]
[0 1 0 0 1 0 1 1]
[0 0 1 0 1 1 0 1]
[0 0 1 1 1 1 1 0]
sage: A.rows()
[(1, 1, 0, 0, 1, 1, 1, 1), (0, 1, 0, 0, 1, 0, 1, 1),
 (0, 0, 1, 0, 1, 1, 0, 1), (0, 0, 1, 1, 1, 1, 1, 0)]
sage: A.columns()
[(1, 0, 0, 0), (1, 1, 0, 0), (0, 0, 1, 1), (0, 0, 0, 1),
 (1, 1, 1, 1), (1, 0, 1, 1), (1, 1, 0, 1), (1, 1, 1, 0)]
```

К элементам и к строкам матрицы можно обращаться следующим образом:

```
sage: A = matrix(3,3,range(1, 10))
sage: A[2,0] = 0  # изменяем элемент матрицы
```

```
sage: A[1] = vector([0,0,0]) # заменяем 2-ю строку
sage: A
[1 2 3]
[0 0 0]
[0 8 9]
```

В Sage по умолчанию функция `kernel` вычисляет «левое ядро», то есть, пространство векторов $\{w \,|\, wA = 0\}$. Для вычисления «правого ядра» следует использовать функцию `right_kernel`. Другими словами, `right_kernel` возвращает фундаментальную систему решений соответствующей однородной системы линейных уравнений.

```
sage: kernel(A)
Free module of degree 3 and rank 1 over Integer Ring
Echelon basis matrix:
[ 1  1 -4]
```

Решим систему линейных уравнений вида $AX = Y$:

```
sage: Y = vector([0, -4, -1])
sage: X = A.solve_right(Y)
sage: X
(-2, 1, 0)
sage: A * X   # проверка
(0, -4, -1)
```

Вместо функции `solve_right` можно использовать обратную косую черту `\`. Препроцессор Sage переводит запись `A \ Y` в `A.solve_right(Y)`.

```
sage: A \ Y
(-2, 1, 0)
```

Заметим, что эта функция возвращает только одно частное решение, если оно существует. Если же система несовместна, Sage генерирует исключение:

```
sage: A.solve_right(w)
Traceback (most recent call last):
...
ValueError: matrix equation has no solutions
```

Аналогично, можно использовать функцию `A.solve_left(Y)` для решения матричного уравнения $XA = Y$.

Приведем теперь матрицу к ступенчатому виду:

```
sage: M = MatrixSpace(RationalField(), 2, 3)
sage: A = M([1,2,3, 4,5,6])
sage: A
[1 2 3]
[4 5 6]
sage: A.echelon_form()  # приводим к ступенчатому виду
[ 1  0 -1]
[ 0  1  2]
```

Вычислим теперь собственные векторы и собственные значения матрицы:

```
sage: A = matrix([[0, 4], [-1, 0]])
sage: A.eigenvalues()
[-2*I, 2*I]
```

```
sage: B = matrix([[1, 3], [3, 1]])
sage: B.right_eigenvectors()
[(4, [(1, 1)], 1),
(-2, [(1, -1)], 1)]
```

Последняя функция возвращает список троек вида (собственное значение, собственный вектор, геометрическая кратность).

Найдем теперь характеристический многочлен матрицы, зависящей от параметра:

```
sage: R.<a> = ZZ[]
sage: M = MatrixSpace(R,2)([[a,1], [a,a+1]])
sage: M
[    a     1]
[    a a + 1]
sage: f = M.charpoly()
sage: f
x^2 + (-2*a - 1)*x + a^2
sage: f.parent()
Univariate Polynomial Ring in x over Univariate Polynomial Ring
in a over Integer Ring
sage: M.trace()  # след матрицы
2*a + 1
sage: M.det()  # определитель
a^2
```

В функции `charpoly` можно указать имя переменной, от которой будет зависеть характеристический многочлен (по умолчанию это `x`):

```
sage: R.<u,v> = PolynomialRing(ZZ,2)
sage: A = MatrixSpace(R,2)([u,v,u^2,v^2])
sage: f = A.charpoly('Z'); f
Z^2 + (-v^2 - u)*Z - u^2*v + u*v^2
```

В Sage встроена поддержка разреженных матриц с помощью параметра `sparse`. Такие матрицы более компактны (в памяти хранятся только ненулевые элементы), но в то же время некоторые операции над ними выполняются дольше. Попробуйте самостоятельно провести следующие вычисления:

```
sage: M = MatrixSpace(QQ, 50, 100, sparse=True)
sage: # случайная матрица с данной плотностью заполнения:
sage: A = M.random_element(density=0.05)
sage: E = A.echelon_form()
```

3.2 Векторные пространства

Создадим векторное пространство над полем из двух элементов. Обратите внимание, что базис подпространства в Sage всегда является ступенчатым.

```
sage: V = VectorSpace(GF(2),8)
sage: S1 = V.subspace([V([1,1,0,0,0,0,0,0]),V([1,0,0,0,0,1,1,0])])
sage: # задали подпространство, натянутое на два заданных вектора
```

```
sage: s1
Vector space of degree 8 and dimension 2 over Finite Field of size 2
Basis matrix:
[1 0 0 0 0 1 1 0]
[0 1 0 0 0 1 1 0]
sage: S1.basis()
[
(1, 0, 0, 0, 0, 1, 1, 0),
(0, 1, 0, 0, 0, 1, 1, 0)
]
sage: S1.dimension()
2
```

Построим сумму и пересечение подпространств:

```
sage: S2 = V.subspace([V([1,1,1,1,0,0,0,0]),V([1,1,0,0,0,1,0,1])])
sage: S1+S2
Vector space of degree 8 and dimension 4 over Finite Field of size 2
Basis matrix:
[1 0 0 0 0 0 1 1]
[0 1 0 0 0 0 1 1]
[0 0 1 1 0 0 0 0]
[0 0 0 0 0 1 0 1]
sage: S1.intersection(S2)
Vector space of degree 8 and dimension 0 over Finite Field of size 2
Basis matrix:
[]
```

Важным частным случаем конечномерного векторного пространства является пространство матриц. Создадим пространство квадратных матриц 3×3:

```
sage: M = MatrixSpace(QQ,3)
sage: M
Full MatrixSpace of 3 by 3 dense matrices over Rational Field
```

Базис пространства матриц по умолчанию состоит из матричных единиц:

```
sage: B = M.basis()
sage: len(B)
9
sage: B[1] # элементы базиса нумеруются с нуля!
[0 1 0]
[0 0 0]
[0 0 0]
```

Как обычно, можно явно создавать новые матрицы как элементы указанного пространства:

```
sage: A = M(range(9)); A
[0 1 2]
[3 4 5]
[6 7 8]
```

Натянем на строки матрицы подпространство:

```
sage: V = VectorSpace(GF(2),8)
sage: A = M([1,1,0,0, 1,1,1,1, 0,1,0,0, 1,0,1,1,
...          0,0,1,0, 1,1,0,1, 0,0,1,1, 1,1,1,0])
sage: rows=A.rows()
sage: S = V.subspace(rows)
sage: S
Vector space of degree 8 and dimension 4 over Finite Field of size 2
Basis matrix:
[1 0 0 0 0 1 0 0]
[0 1 0 0 1 0 1 1]
[0 0 1 0 1 1 0 1]
[0 0 0 1 0 0 1 1]
sage: A.echelon_form()
[1 0 0 0 0 1 0 0]
[0 1 0 0 1 0 1 1]
[0 0 1 0 1 1 0 1]
[0 0 0 1 0 0 1 1]
```

Как видно, базис подпространства `S` в Sage был получен из ненулевых строк ступенчатого вида.

3.3 Упражнения

1. Решите систему линейных уравнений

$$\begin{cases} (1+\lambda)x_1 + x_2 + x_3 = 1, \\ x_1 + (1+\lambda)x_2 + x_3 = \lambda, \\ x_1 + x_2 + (1+\lambda)x_3 = \lambda^2; \end{cases}$$

в зависимости от значения параметра λ.

2. Найдите базисы суммы и пересечения линейных оболочек систем векторов

$$\langle(-1,6,4,7,-2),(-2,3,0,5,-2),(-3,6,5,6,-5)\rangle$$

и

$$\langle(1,1,2,1,-1),(0,-2,0,-1,-5),(2,0,2,1,-3)\rangle.$$

3. Найдите систему линейных уравнений, задающую систему векторов

$$\langle(1,-1,1,-1,1),(1,1,0,0,3),(3,1,1,-1,7)\rangle.$$

4. Пусть в пространстве $\mathbb{R}^4$

$$U = \langle(1,1,1,1),(-1,-2,0,1)\rangle, \quad V = \langle(-1,-1,1,-1),(2,2,0,1)\rangle.$$

Докажите, что $\mathbb{R}^4 = U \oplus V$, и найти проекцию вектора $(4,2,4,4)$ на подпространство U параллельно V.

5. При каких значениях λ квадратичная функция $5x_1^2 + x_2^2 + \lambda x_3^2 + 4x_1x_2 - 2x_1x_3 - 2x_2x_3$ является положительно определенной? Когда она является отрицательно определенной?

6. Найдите жорданову форму матрицы линейного оператора $\mathcal{A}$ и базис $(f_1,\dots,f_n)$, в котором $\mathcal{A}$ имеет эту матрицу, если в базисе $(e_1,\dots,e_n)$ оператор $\mathcal{A}$ задается матрицей $\begin{pmatrix} 0 & 1 & -1 & 1 \\ -1 & 2 & -1 & 1 \\ -1 & 1 & 1 & 0 \\ -1 & 1 & 0 & 1 \end{pmatrix}$.

7. Постройте полярное разложение матрицы $\begin{pmatrix} 4 & -2 & 2 \\ 4 & 4 & -1 \\ -2 & 4 & 2 \end{pmatrix}$.

3.4 Задачи

1. Напишите функции, генерирующие следующие матрицы заданного размера:

а) $\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 2 & \dots & 2 \\ 1 & 2 & 3 & \dots & 3 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2 & 3 & \dots & n \end{pmatrix}$, б) $\begin{pmatrix} 0 & \dots & 0 & x_1 \\ \vdots & & \vdots & x_2 \\ 0 & \dots & 0 & \vdots \\ x_1 & x_2 & \dots & x_n \end{pmatrix}$, в) $\begin{pmatrix} x & 1 & 0 & \dots & 0 \\ \frac{x^2}{2!} & x & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & 1 \\ \frac{x^n}{n!} & \dots & \dots & \frac{x^2}{2!} & x \end{pmatrix}$.

Вычислите с помощью этих процедур определители матриц а) и в) и характеристический многочлен матрицы б). Установите закономерность и докажите ее.

2. Напишите функции, которые по заданной последовательности $x_1, \dots, x_{2n-1}$ генерируют матрицы

$$A = \begin{pmatrix} x_n & x_{n+1} & \dots & x_{2n-2} & x_{2n-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ x_2 & & \ddots & \ddots & x_{n+1} \\ x_1 & x_2 & \dots & x_{n-1} & x_n \end{pmatrix} \quad \text{и} \quad B = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ x_2 & x_3 & \dots & x_{n+1} \\ \vdots & \vdots & & \vdots \\ x_n & x_{n+1} & \dots & x_{2n-1} \end{pmatrix}.$$

3. Пусть $A = (a_{ij})$ — матрица размера $m \times n$, а $B = (b_{ij})$ — матрица размера $r \times s$. Тензорным произведением матриц A и B называется матрица

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{pmatrix}$$

размера $mr \times ns$. Напишите функцию, генерирующую тензорное произведение двух матриц. Чему равен определитель тензорного произведения?

4. *(Метод вычисления определителя Льюиса Кэролла.)* Проверьте с помощью компьютерных вычислений справедливость следующего тождества:

$$\begin{vmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{vmatrix} = \frac{1}{x_{12} \dots x_{1,n-1}} \begin{vmatrix} \begin{vmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{vmatrix} & \begin{vmatrix} x_{12} & x_{13} \\ x_{22} & x_{23} \end{vmatrix} & \dots & \begin{vmatrix} x_{1,n-1} & x_{1n} \\ x_{2,n-1} & x_{2n} \end{vmatrix} \\ \vdots & \vdots & \ddots & \vdots \\ \begin{vmatrix} x_{11} & x_{12} \\ x_{n1} & x_{n2} \end{vmatrix} & \begin{vmatrix} x_{12} & x_{13} \\ x_{n2} & x_{n3} \end{vmatrix} & \dots & \begin{vmatrix} x_{1,n-1} & x_{1n} \\ x_{n,n-1} & x_{nn} \end{vmatrix} \end{vmatrix},$$

где $x_{i,j}$ являются формальными переменными. Докажите это тождество.

5. *(Формула Якоби.)* Пусть M — обратимая матрица порядка n. Пусть

$$P = \{p_1 < \dots < p_k\} \text{ и } Q = \{q_1 < \dots < q_k\}$$

— наборы индексов от 1 до n. Пусть

$$\overline{P} = \{p_{k+1} < \dots < p_n\} \text{ и } \overline{Q} = \{q_{k+1} < \dots < q_n\}$$

— оставшиеся индексы. Обозначим через $M_{P,Q}$ подматрицу матрицы M, образованную строками с индексами из P и столбцами с индексами из Q. Проверьте экспериментально следующую формулу, выражающую минор обратной матрицы:

$$\det\left((M^{-1})_{P,Q}\right) = \operatorname{sgn}(\sigma\tau)\left(\det M_{\overline{Q},\overline{P}}\right)(\det M)^{-1},$$

где σ и τ — подстановки из $\mathbf{S}_n$, причем $\sigma(i) = p_i$ и $\tau(j) = q_j$. Докажите эту формулу. Во что превращается эта формула при $k = 1$ и при $k = n$?

6. Пусть $M = (x_{ij})$ и $N = (y_{ij})$ — квадратные матрицы порядка n. Рассмотрим матрицу B того же порядка с элементами

$$B_{ij} = \begin{pmatrix} x_{1i} & y_{11} & \cdots & y_{1,j-1} & y_{1,j+1} & \cdots & y_{1n} \\ x_{2i} & y_{21} & \cdots & y_{2,j-1} & y_{2,j+1} & \cdots & y_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{ni} & y_{n1} & \cdots & y_{n,j-1} & y_{n,j+1} & \cdots & y_{nn} \end{pmatrix}.$$

Проверьте с помощью компьютерных вычислений, что

$$\det B = (\det M)(\det N)^{n-1}.$$

7. Пусть $A = (a_{ij})$ — кососимметрическая матрица четного[1] порядка.

а) Проверьте с помощью компьютерных вычислений, что существует многочлен $\operatorname{pf} A$ от элементов a_{ij}, называемый *пфаффианом* матрицы A, такой, что $\det A = (\operatorname{pf} A)^2$. Вычислите экспериментально $\operatorname{pf} A$ для небольших порядков и предложите общую формулу. Докажите ее.

б) Выведите правило для разложения пфаффиана по строке или столбцу, аналогичное правилу для определителя.

8. Придумайте какое-нибудь вложение тела кватернионов $\mathbb{H}$ в алгебру матриц $\mathbf{M}_4(\mathbb{R})$. С помощью такого вложения реализуйте свой класс «кватернион», определив операции над кватернионами через соответствующие операции с матрицами. Докажите с помощью матричных вычислений тождество Эйлера:

$$\begin{gathered}(x_1^2+x_2^2+x_3^2+x_4^2)(y_1^2+y_2^2+y_3^2+y_4^2) = \\ = (x_1y_1+x_2y_2+x_3y_3+x_4y_4)^2+(x_1y_2-x_2y_1-x_3y_4+x_4y_3)^2+ \\ +(x_1y_3+x_2y_4-x_3y_1-x_4y_2)^2+(x_1y_4-x_2y_3+x_3y_2-x_4y_1)^2.\end{gathered}$$

9. Не используя функции построения жордановой нормальной формы и не вычисляя собственных значений, напишите свою функцию, определяющую, является ли матрица диагонализируемой над указанным полем. (*Указание:* освободите характеристический многочлен от кратных корней.)

10. Не вычисляя собственных значений, напишите свою функцию, вычисляющую минимальный многочлен матрицы. (*Указание:* рассмотрите линейную зависимость между степенями матрицы.)

11. Создайте функцию, возвращающую *все* решения данной системы линейных уравнений над указанным конечным полем.

12. Напишите функцию, которая по заданной системе векторов $v_1, \ldots, v_n$ строит (в символьном виде) однородную систему линейных уравнений от переменных $x_1, \ldots, x_n$, решением которой является подпространство $\langle v_1, \ldots, v_n \rangle$.

[1] Чему равен определитель A, если порядок нечетен?

13. *Континуантой* называется определитель

$$(a_1a_2\dots a_n)=\begin{vmatrix} a_1 & 1 & 0 & 0 & \dots & 0 & 0\\ -1 & a_2 & 1 & 0 & \dots & 0 & 0\\ 0 & -1 & a_3 & 1 & \dots & 0 & 0\\ \dots & \dots & \dots & \dots & \dots & \dots & \dots\\ 0 & 0 & 0 & 0 & \dots & -1 & a_n \end{vmatrix}.$$

Проверьте при небольших n, что

$$\frac{(a_1a_2\dots a_n)}{(a_2a_3\dots a_n)}=a_1+\cfrac{1}{a_2+\cfrac{1}{a_3+\dots+\cfrac{1}{\ddots\,\cfrac{1}{a_{n-1}+\cfrac{1}{a_n}}}}}.$$

14. Заданы две системы векторов. Напишите функцию, возвращающую матрицу перехода от первой системы ко второй (или генерирующей исключение, если какая-то из систем не является базисом).

15. Напишите функцию, представляющую заданную невырожденную матрицу в виде произведения элементарных матриц.

16. Напишите функцию, дополняющую заданную систему векторов до базиса всего пространства.

17. Реализуйте общий алгоритм решения квадратной системы линейных уравнений с параметром λ. (*Указание:* воспользуйтесь теоремой Крамера.)

18. Напишите собственную функцию, строящую для заданной невырожденной квадратной матрицы ее полярное разложение.

19. Напишите свою функцию, применяющую процесс ортогонализации Грама-Шмидта к заданной системе векторов.

20. Напишите функцию, определяющую по заданному вектору v и системе векторов L, принадлежит ли v линейной оболочке векторов из L.

21. Проверьте для $n\leqslant 100$ утверждение, что определитель матрицы $A=(\text{НОД}\,(i,j))$ размера $n\times n$ равен $\varphi(2)\varphi(3)\dots\varphi(n)$, где φ — функция Эйлера. Что можно сказать об определителе матрицы $A=(\text{НОК}\,(i,j))$?

22. Напишите функцию, которая по двум данным квадратным матрицам определяла бы, могут ли они задавать один и тот же линейный оператор.

23. Даны две квадратных матрицы, являющихся жордановыми формами. Напишите функцию, которая проверяет, задают ли эти матрицы один и тот же линейный оператор. Если какая-либо матрица не является жордановой, функция должна генерировать исключение.

24. Напишите функцию, которая получала бы на вход квадратичную форму над $\mathbb{R}$ и выясняла бы, является ли она положительно (отрицательно) определенной.

25. Напишите функцию, которая по двум данным симметрическим матрицам определяла бы, могут ли они задавать одну и ту же квадратичную форму над $\mathbb{R}$.

26. Напишите функцию, которая по заданной квадратной матрице определяла бы, может ли она являться матрицей

а) самосопряженного оператора

б) ортогонального оператора

в каком-либо базисе.

27. Напишите функцию, которая по двум данным ортогональным матрицам определяла бы, является ли первая матрица канонической формой второй матрицы.

28. Напишите функцию, получающую на вход квадратную матрицу $A \in \mathbf{M}_n(\mathbb{C})$ и возвращающую общий вид матрицы A^k, где k — произвольная символьная переменная, принимающая натуральные значения. (*Указание:* приведите матрицу с помощью функции `jordan_form` к жордановой форме и сведите эту задачу к отдельным жордановым клеткам.)

Глава 4

Многочлены и системы алгебраических уравнений

4.1 Основные операции с многочленами

В Sage для работы с многочленами (как и с другими алгебраическими объектами) нужно сначала определить алгебраическую структуру, к которой эти объекты будут относиться. В нашем случае это кольцо многочленов. Для его задания достаточно указать имена переменных и основное поле (или кольцо) коэффициентов. Например, в качестве кольца коэффициентов могут выступать `ZZ`, `QQ`, `RR`, `CC`, `Zmod(n)`, `GF(p)`, `GF(p^n, 'x')`.

Имеется несколько способов объявить кольцо многочленов:

```
sage: R = PolynomialRing(QQ, 't')
sage: R
Univariate Polynomial Ring in t over Rational Field
sage: S = QQ['t']
sage: S == R
True
sage: R3 = PolynomialRing(GF(5),3,'z') # здесь 3 - число переменных
sage: R3
Multivariate Polynomial Ring in z0, z1, z2 over Finite Field of size 5
sage: S3 = GF(5)['z0, z1, z2']
Multivariate Polynomial Ring in z0, z1, z2 over Finite Field of size 5
sage: R3 == S3
True
```

Эти команды создают кольца многочленов $\mathbb{Q}[t]$ и $\mathbb{F}_5[z_0, z_1, z_2]$ и сообщают Sage, что t и z_i — имена переменных кольца, которые надо использовать при выводе на экран. Однако этими именами пока нельзя свободно оперировать в Sage (например, определяя с их помощью новый многочлен, скажем, $t^2 + 1$).

```
sage: t^2 + 1
Traceback (click to the left of this block for traceback)
...
NameError: name 't' is not defined
sage: t = R.0  # взять первую образующую
sage: t^2 + 1
t^2 + 1
sage: w = R3.gens()  # взять список образующих
sage: w[0]*w[1] + w[2]
z[0]*z[1] + z[2]
```

Обратите внимание, что имена тех переменных Sage, которые используются для задания многочленов, не обязаны совпадать с именами указанными при создании кольца и используемыми при выводе на экран.

Напомним, что в Sage встроен так называемый препроцессор, который перед выполнением команды переводит ее при необходимости к синтаксису языка Python. Использование препроцессора позволяет расширить язык Sage, включить в него новые конструкции. Такие конструкции имеются и для задания кольца многочленов одновременно с введением символьных переменных этого кольца. Вот несколько эквивалентных примеров:

```
sage: R3.<x,y,z> = PolynomialRing(QQ)
sage: R3.<x,y,z> = QQ['x,y,z']
sage: R3.<x,y,z> = QQ[]
```

Посмотреть, как подобные конструкции переводится на Python, можно с помощью функции `preparse`:

```
sage: preparse('R3.<x,y,z> = QQ[]')
R3 = QQ['x,y,z']; (x,y,z,) = R3._first_ngens(3)
sage: preparse('x = R3.0')
x = R3.gen(0)
```

Можно также сразу получить кольцо и образующую следующим способом:

```
sage: R, t = QQ['t'].objgen()
sage: t    = QQ['t'].gen()
sage: R, t = objgen(QQ['t'])
sage: t    = gen(QQ['t'])
```

Выполним теперь элементарные символьные вычисления с многочленами:

```
sage: poly = (t+1) * (t+2); poly
t^2 + 3*t + 2
sage: poly in R
True
sage: f = 2*t^7 + 3*t^2 - 15/19
sage: f^2
4*t^14 + 12*t^9 - 60/19*t^7 + 9*t^4 - 90/19*t^2 + 225/361
sage: # многочлен деления круга
sage: cyclo = R.cyclotomic_polynomial(7); cyclo
t^6 + t^5 + t^4 + t^3 + t^2 + t + 1
sage: g = 7 * cyclo * t^5 * (t^5 + 10*t + 2)
sage: g
7*t^16 + 7*t^15 + 7*t^14 + 7*t^13 + 77*t^12 + 91*t^11 + 91*t^10
       + 84*t^9 + 84*t^8 + 84*t^7 + 84*t^6 + 14*t^5
sage: F = factor(g); F
(7) * t^5 * (t^5 + 10*t + 2) * (t^6 + t^5 + t^4 + t^3 + t^2 + t + 1)
sage: F.unit()  # коэффициент в разложении на множители
7
sage: list(F)  # список неприводимых множителей и их степеней
[(t, 5), (t^5 + 10*t + 2, 1), (t^6 + t^5 + t^4 + t^3 + t^2 + t + 1, 1)]
```

Приведем еще примеры разложения многочленов на множители:

```
sage: x = PolynomialRing(RationalField(), 'x').gen()
sage: f = (x^3 - 1)^2 - (x^2-1)^2
sage: f.factor()
(x - 1)^2 * x^2 * (x^2 + 2*x + 2)
sage: x, y = PolynomialRing(RationalField(), 2, ['x','y']).gens()
sage: f =  9*y^6 - 9*x^2*y^5 - 18*x^3*y^4 - 9*x^5*y^4\
...   + 9*x^6*y^2 + 9*x^7*y^3 + 18*x^8*y^2 - 9*x^11
sage: f.factor()
(-9) * (x^5 - y^2) * (x^6 - 2*x^3*y^2 - x^2*y^3 + y^4)
```

Для работы с многочленами Sage использует различные пакеты. Так, функции `cyclotomic_polynomial` и `factor`, приведенные выше, на самом деле вызывают аналогичные функции из пакета PARI. Код функции `factor` для многочленов от нескольких переменных использует пакет Singular.

Продолжим наши вычисления в факторкольце $\mathbb{F}_{97}[x]/(x^3+7)$. Вычислим, например, $x^{2011} \pmod{x^3+7}$. Дадим образу переменной x при каноническом гомоморфизме имя a:

```
sage: R = PolynomialRing(GF(97),'x')
sage: x = R.gen()
sage: S = R.quotient(x^3 + 7, 'a')  # определяем факторкольцо
sage: a = S.gen()
sage: S
Univariate Quotient Polynomial Ring in a over
Finite Field of size 97 with modulus x^3 + 7
sage: a^2011
2*a
```

Вот три примера вычисления наибольшего общего делителя двух многочленов:

```
sage: x = PolynomialRing(QQ, 'x').gen()
sage: gcd (x^3-1, x^2-1)
x - 1
sage: R3.<x,y,z> = PolynomialRing(QQ, 3)
sage: f = 3*x^2*(x+y)
sage: g = 9*x*(y^2 - x^2)
sage: f.gcd(g)
x^2 + x*y
sage: x = PolynomialRing(GF(2), 'x').gen()
sage: f = (x^3 - x + 1)*(x + x^2); f
x^5 + x^4 + x^3 + x
sage: g = (x^3 - x + 1)*(x + 1)
sage: f.gcd(g)
x^4 + x^3 + x^2 + 1
```

Частное двух многочленов дает элемент поля рациональных функций, которое Sage создает автоматически:

```
sage: x = QQ['x'].0
sage: f = x^3 + 1; g = x^2 - 17
sage: h = f/g;  h
(x^3 + 1)/(x^2 - 17)
sage: h.parent()  # алгебраическая структура, которой принадлежит h
Fraction Field of Univariate Polynomial Ring in x over Rational Field
```

Аналогично кольцам многочленов можно строить степенные ряды и ряды Лорана:

```
sage: R.<T> = PowerSeriesRing(GF(7)); R  # кольцо степенных рядов
Power Series Ring in T over Finite Field of size 7
sage: f = T  + 3*T^2 + T^3 + O(T^4)
sage: f^3
T^3 + 2*T^4 + 2*T^5 + O(T^6)
sage: 1/f
T^-1 + 4 + T + O(T^2)
sage: parent(1/f)
Laurent Series Ring in T over Finite Field of size 7
sage: R.<x> = LaurentSeriesRing(QQ); R  # кольцо рядов Лорана
Laurent Series Ring in x over Rational Field
sage: 1/(1-x) + O(x^10)
1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + O(x^10)
```

Кольцо степенных рядов можно также создать с помощью двойных квадратных скобок:

```
sage: GF(7)[['T']]
Power Series Ring in T over Finite Field of size 7
```

Заметим, что кольца многочленов в Sage считаются разными, если переменные имеют различные имена (или если на них заданы разные упорядочения мономов):

```
sage: R.<x> = PolynomialRing(QQ)
sage: S.<y> = PolynomialRing(QQ)
sage: x == y
False
sage: R == S
False
sage: R(y)  # принудительно приводим многочлен от y к многочлену от x
x
sage: R(y^2 - 17)
x^2 - 17
```

```
sage: R = PolynomialRing(QQ, "x")
sage: T = PolynomialRing(QQ, "x")
sage: R == T
True
sage: R is T
True
sage: R.0 == T.0
True
```

Напомним, что *результантом* двух многочленов $f(x) = \sum_{k=0}^{m} a_k x^k$ и $g(x) = \sum_{k=0}^{n} b_k x^k$ называется определитель матрицы Сильвестра размера $(m+n) \times (m+n)$:

$$\mathrm{Res}(f(x), g(x)) := \begin{vmatrix} a_m & a_{m-1} & \cdots & \cdots & \cdots & a_0 & 0 & \cdots & 0 \\ 0 & a_m & \cdots & \cdots & \cdots & a_1 & a_0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_0 \\ b_n & b_{n-1} & \cdots & b_0 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & b_n & \cdots & b_1 & b_0 & \cdots & \cdots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & b_0 \end{vmatrix}$$

Результант двух многочленов равен нулю тогда и только тогда, когда эти многочлены имеют общий корень. Более того,

$$\mathrm{Res}(f,g) = \prod_{f(x)=0, g(y)=0} (x-y),$$

где произведение берется по всем корням многочленов f и g в алгебраическом замыкании поля с учетом кратности. В системе Sage результант вычисляется функцией `resultant`:

```
sage: R.<x> = QQ[]
sage: f = x^3 - 1
sage: g = x^2 - 1
sage: f.resultant(g)
0
```

Другие доступные функции работы с многочленами предлагается изучить самостоятельно.

Рассмотрим теперь работу с многочленами от нескольких переменных. Для работы с такими многочленами (например, для вычисления НОДа или базиса Гребнера) Sage по умолчанию использует пакет Singular.

```
sage: R1 = PolynomialRing(GF(5),3,'z'); R1
Multivariate Polynomial Ring in z0, z1, z2 over Finite Field of size 5
sage: R2.<x,y> = QQ[]
sage: f = (x^3 + 2*y^2*x)^2
sage: g = x^2*y^2
sage: f.gcd(g)
x^2
```

По умолчанию многочлены от нескольких переменных хранятся в Sage в разреженном виде. Получить набор коэффициентов таких многочленов можно, преобразуя многочлен к типу `dict`. В результате получится словарь, сопоставляющий каждой мультистепени ненулевой коэффициент при соответствующем мономе. Обратное преобразование получается приведением словаря коэффициентов к типу кольца:

```
sage: R.<x,y> = QQ[]
sage: f = x^2*y + y^3
sage: f.dict()
{(0, 3): 1, (2, 1): 1}
sage: list(f)
[(1, x^2*y), (1, y^3)]
sage: d = {(1,2):3, (4,5):6}
sage: R(d)
6*x^4*y^5 + 3*x*y^2
```

4.2 Системы алгебраических уравнений и базисы Грёбнера

Мы не претендуем на полноту изложения теории базисов Гребнера, отсылая читателя к книгам [1, 3, 12, 13, 11, 10].

Системе алгебраических (то есть, полиномиальных) уравнений соответствует идеал в кольце многочленов, образованный левыми частями уравнений. Построим, например, идеал (f,g) в $\mathbb{Q}[x,y]$. Для этого можно просто умножить `(f,g)` на `R` (можно также записать `ideal([f,g])` или `ideal(f,g)`):

```
sage: R.<x,y> = QQ[]
sage: I = (f, g)*R; I
Ideal (x^6 + 4*x^4*y^2 + 4*x^2*y^4, x^2*y^2) of Multivariate Polynomial
Ring in x, y over Rational Field
```

По *теореме Гильберта о базисе* всякий идеал в кольце многочленов $F[x_1, \ldots, x_n]$ над полем F является конечно порожденным. Базис Гребнера идеала — это система образующих особого вида, с помощью которой можно конструктивно (алгоритмически) отвечать на различные вопросы о свойствах идеала и соответствующей системы полиномиальных уравнений. Алгоритм построения базиса Гребнера с одной стороны обобщает метод Гаусса приведения системы линейных уравнений к ступенчатому виду, а с другой — алгоритм Евклида нахождения наибольшего общего делителя двух многочленов от одной переменной над полем. Для определения базиса Гребнера требуется задать *мономиальное упорядочение*, то есть, линейный порядок на всех мономах кольца, удовлетворяющий свойствам

1. $m_1 \prec m_2 \implies mm_1 \prec mm_2$ для любых мономов m, m_1, m_2;
2. $1 \preccurlyeq m$ для любого монома m.

Примерами мономиальных упорядочений могут служить *лексикографическое* упорядочение (lex), *степенное лексикографическое* (deglex) и *степенное обратное лексикографическое* (degrevlex):

$$x_1^{a_1} \ldots x_n^{a_n} \prec_{degrevlex} x_1^{b_1} \ldots x_n^{b_n} \quad \iff \quad \left(\sum a_i, b_n, \ldots, b_1\right) \prec_{lex} \left(\sum b_i, a_n, \ldots, a_1\right).$$

В кольце многочленов $F[x]$ существует только одно упорядочение (по степени). В качестве упражнения предлагаем читателям доказать, что в кольце многочленов от нескольких переменных мощность множества всех упорядочений — континуум.

Как только упорядочение зафиксировано, у каждого ненулевого многочлена f можно определить *старший моном*, который обозначается $\operatorname{lm} f$, а также *старший коэффициент* $\operatorname{lc} f$.

Определение 1. Конечное подмножество G идеала $I \lhd F[x_1, \ldots, x_n]$ называется *базисом Гребнера* относительно $\prec$, если $I = (G)$ и для любого ненулевого многочлена $f \in I$ найдется $g \in G$, такой, что $\operatorname{lm} g \mid \operatorname{lm} f$. Другими словами, G — базис Гребнера, если множество $\operatorname{lm} G$ порождает идеал $(\operatorname{lm} I)$.

Если G — базис Гребнера идеала I, то можно определить *процесс редукции* многочлена f относительно G: если в f присутствует слагаемое, которое кратно старшему моному $\operatorname{lm} g$ для некоторого $g \in G$, то из f вычитается g с подходящим множителем, чтобы это старшее слагаемое сократилось, а затем процесс повторяется. Процесс редукции приводит многочлен к однозначно определенной *нормальной форме* по модулю идеала. Эта нормальная форма позволяет задать канонического представителя в смежных классах факторалгебры $F[x_1, \ldots, x_n]/I$. Соответственно, $f \in I$ тогда и только тогда, когда нормальная форма f относительно G равна нулю. Это свойство позволяет проверять принадлежность многочлена идеалу (другими словами — проверять, верно ли, что заданное уравнение является алгебраическим следствием данной системы). На его основе легко построить алгоритм проверки включения двух идеалов: пусть $I_1 = (F)$ и $I_2 = (G)$, причем G — базис Гребнера I_2. Тогда $I_1 \subset I_2$ тогда и только тогда, когда каждый многочлен из F принадлежит I_2, то есть, редуцируется относительно G к нулю.

Среди всех базисов Гребнера данного идеала можно выделить единственный *редуцированный базис*, который определяется однозначно по идеалу и мономиальному упорядочению. Редуцированный базис — минимальный базис Гребнера, в котором старший коэффициент каждого многочлена равен единице и ни одно слагаемое многочленов не делится ни на один старший моном других многочленов.

Проверку совпадения двух идеалов можно выполнить проще, чем проверить оба включения $I_1 \subset I_2$ и $I_2 \subset I_1$: два идеала совпадают тогда и только тогда, когда совпадают их редуцированные базисы Гребнера.

Пусть f и g — многочлены. S-полиномом этих многочленов называется многочлен

$$S(f,g) = \operatorname{lc} g \frac{\operatorname{lm} g}{\text{НОД}\,(\operatorname{lm} f, \operatorname{lm} g)} f - \operatorname{lc} f \frac{\operatorname{lm} f}{\text{НОД}\,(\operatorname{lm} f, \operatorname{lm} g)} g.$$

Другими словами, многочлены f и g домножаются минимальным образом на одночлены так, чтобы у них оказались одинаковые старшие мономы и старшие коэффициенты, а затем они вычитаются друг из друга, чтобы эти старшие мономы сократились.

Имеет место следующее утверждение: *множество G является базисом Гребнера идеала (G) тогда и только тогда, когда все S-полиномы многочленов из G редуцируются относительно G к нулю.* На основе этого утверждения строится простейший алгоритм вычисления базиса Гребнера — алгоритм Бухбергера: последовательно вычисляются результаты редукции S-полиномов пар элементов базиса, и если результат ненулевой, то он добавляется к базису.

Найдем в Sage редуцированный базис Гребнера описанного выше идеала (по умолчанию порядок предполагается лексикографическим, переменные упорядочены по возрастанию):

```
sage: B = I.groebner_basis(); B
[x^6, x^2*y^2]
```

Справедливо следующее утверждение: если G — базис Гребнера идеала $I \triangleleft F[x_1, \ldots, x_n]$ относительно лексикографического упорядочения с $x_1 \prec x_2 \prec \ldots \prec x_n$, то $G \cap F[x_1, \ldots, x_k]$ — базис Гребнера идеала $I \cap F[x_1, \ldots, x_k]$. Поэтому лексикографический базис Гребнера обладает исключающими свойствами: с его помощью можно получать все следствия системы уравнений, зависящие только от определенных переменных. Это позволяет последовательно исключать неизвестные из системы и тем самым сводить решение систем полиномиальных уравнений к решению отдельных полиномиальных уравнений от одной переменной. Так, в приведенном примере есть многочлен x^6, не зависящий от y. В нашем случае оказалось, что базис Гребнера состоит из мономов (то есть, I – мономиальный идеал). Убедимся, что меньшие мономы в идеал не попадают:

```
sage: x^2 in I
False
```

Рассмотрим еще несколько примеров полиномиальных систем.

Пример 1. Система

$$\begin{cases} x^4 + x^3 - x - 1 = 0, \\ x^3 + 2x^2 + 2x + 1 = 0, \\ x^6 + x^5 + x^2 + 2x + 1 = 0 \end{cases}$$

зависит только от одного неизвестного x. Она эквивалентна одному уравнению, левая часть которого — НОД левых частей уравнений системы. В самом деле, в кольце $F[x]$ НОД любых многочленов выражается через сами эти многочлены в виде линейной комбинации с коэффициентами из $F[x]$. На другом языке это означает, что $F[x]$ — кольцо главных идеалов. Соответственно, редуцированным базисом Гребнера этой системы будет НОД — образующий элемент соответствующего идеала:

```
sage: R.<x> = PolynomialRing(QQ, 'x')
sage: I = (x^4+x^3-x-1, x^3+2*x^2+2*x+1, x^6+x^5+x^2+2*x+1)*R; I
Principal ideal (x^3 + 2*x^2 + 2*x + 1) of Univariate Polynomial Ring
in x over Rational Field
```

Главные идеалы автоматически задаются в Sage своим образующим элементом, поэтому для них даже бессмысленно вызывать функцию `groebner_basis`.

Пример 2. Система

$$\begin{cases} x^2 - 3xy - y^2 + 5x - 7y + 1 = 0, \\ 3x^2 + xy + 7x + 3y = 0, \\ x^3 + 5x^2 + x + 3y^2 - 1 = 0 \end{cases}$$

несовместна. В самом деле, комбинация этих уравнений с коэффициентами x, y и -1 приводит к уравнению $1 = 0$. В общем случае *система полиномиальных уравнений над алгебраически замкнутым полем несовместна тогда и только тогда, когда ее редуцированный базис Гребнера содержит единицу*. Проверим это:

```
sage: R.<x,y> = PolynomialRing(QQ, 2, 'x y')
sage: I = (x^2 - 3*x*y - y^2 + 5*x - 7*y + 1,
...        3*x^2 + x*y + 7*x + 3*y,
...        x^3 + 5*x^2 + x + 3*y^2 - 1) * R
sage: I.groebner_basis()
[1]
```

Это — прямое следствие *теоремы Гильберта о нулях*. Впрочем, если поле не является алгебраически замкнутым, то базис Гребнера несовместной системы может не содержать единицу: достаточно рассмотреть уравнение $x^2 + 1 = 0$ над $\mathbb{R}$.

Пример 3. Построим лексикографический базис Гребнера идеала $I = (x^2 + y^2 + z^2, x + y - z, y + z^2)$ при $x > y > z$. Вычислим S-полином первых двух многочленов и отредуцируем его. Получим $z^4 + z^3 + z^2$. Теперь видно, что базис Гребнера образован этим остатком и двумя последними многочленами.

Пример 4. Решим систему

$$\begin{cases} ab = c^2 + c, \\ a^2 = a + bc, \\ ac = b^2 + b. \end{cases}$$

Для этого найдем базис Гребнера соответствующего идеала при лексикографическом порядке ($a > b > c$):

```
sage: R.<a,b,c> = QQ[]
sage: I = (a*b - c^2 - c, a^2 - a - b*c, a*c - b^2 - b) * R
sage: I.groebner_basis()
[c^3 + c^2, a^2 - a, a*b - c^2 - c, b^2 - c^2 + b - c,
a*c - c^2 - c, b*c]
```

Мы видим, что базис Гребнера содержит многочлен, зависящий только от младшей переменной c. Это в точности многочлен, порождающий идеал $I \cap \mathbb{Q}[c]$ в кольце $\mathbb{Q}[c]$. Таким образом мы свели задачу о решении системы к решению уравнения от одной переменной. Находим, что возможные значения c — это 0 и -1. Подставляя их в другие уравнения, получаем уравнения на b и c. Отсюда получаем четыре решения: $(0, 0, 0); (1, 0, 0); (0, -1, 0); (0, 0, -1)$.

Вычисление базиса Гребнера в общем случае — очень трудоемкая задача. Существуют оценки, показывающие, что в некоторых случаях сложность вычислений может быть двойной экспоненциальной. В настоящее время кроме классического алгоритма Бухбергера с различными модификациями существуют и другие алгоритмы (например, F4 и F5). Для сравнения эффективности различных алгоритмов обычно используется стандартный набор тестовых полиномиальных систем. Такова, например, знаменитая серия примеров cyclicN. Вычислим базис Гребнера системы cyclic4:

```
sage: R.<a,b,c,d> = PolynomialRing(QQ, 4, 'abcd', order='lex')
sage: I = (a+b+c+d, a*b+a*d+b*c+c*d, a*b*c+a*b*d+a*c*d+b*c*d, \
           a*b*c*d-1)*R
sage: I
Ideal (a + b + c + d, a*b + a*d + b*c + c*d,
a*b*c + a*b*d + a*c*d + b*c*d, a*b*c*d - 1) of Multivariate Polynomial
Ring in a, b, c, d over Rational Field
```

Видно, что система cyclicN состоит из N уравнений от N неизвестных степеней от 1 до N, причем каждое уравнение (кроме последнего) является однородным и состоит из N слагаемых.

```
sage: B = I.groebner_basis(); B
[a + b + c + d,
 b^2 + 2*b*d + d^2,
 b*c - b*d + c^2*d^4 + c*d - 2*d^2,
 b*d^4 - b + d^5 - d,
 c^3*d^2 + c^2*d^3 - c - d,
 c^2*d^6 - c^2*d^2 - d^4 + 1]
```

Заметим, что в Sage базис Гребнера не является списком, а представляет собой неизменяемую последовательность:

```
sage: B.parent()
Category of sequences in Multivariate Polynomial Ring in x, y
over Rational Field
sage: B.universe()
Multivariate Polynomial Ring in x, y over Rational Field
sage: B[1] = x
Traceback (most recent call last):
...
ValueError: object is immutable; please change a copy instead.
```

Все вычисления базиса Гребнера кешируются, чтобы при повторном обращении к нему не повторять вычисления заново.

Sage включает в себя так называемую «обучающую версию» алгоритма Бухбергера вычисления базиса Гребнера. Эта версия намеренно не содержит никаких оптимизаций и предназначена для образовательных целей. Кроме того, в Sage уже включены стандартные серии примеров (cyclic, katsura и т.д.), на которых обычно тестируют алгоритмы вычисления базисов Гребнера. Попробуем вычислить базис системы Katsura-6 относительно упорядочения degrevlex:

```
sage: from sage.rings.polynomial.toy_buchberger import *
sage: # ведем вычисления по большому простому модулю
sage: P.<a,b,c,e,f,g,h,i,j,k> = \
      PolynomialRing(GF(32003), 10, order="degrevlex")
sage: I = sage.rings.ideal.Katsura(P,6)
sage: g1 = buchberger(I)
sage: g2 = buchberger_improved(I)
sage: g3 = I.groebner_basis()
```

Все эти алгоритмы вычисляют базис Гребнера:

```
sage: Ideal(g1).basis_is_groebner()
True
sage: Ideal(g2).basis_is_groebner()
True
sage: Ideal(g3).basis_is_groebner()
True
```

Результаты, разумеется, совпадают:

```
sage: Ideal(g1) == Ideal(g2) == Ideal(g3)
True
```

Если `get_verbose() >= 1`, то Sage покажет протокол вычислений:

```
sage: set_verbose(1)
sage: P.<a,b,c> = PolynomialRing(GF(127),3)
sage: I = sage.rings.ideal.Katsura(P)
sage: I
Ideal (a + 2*b + 2*c - 1, a^2 + 2*b^2 + 2*c^2 - a,
2*a*b + 2*b*c - b) of Multivariate Polynomial Ring
in a, b, c over Finite Field of size 127
sage: G = I.groebner_basis()
```

В этом примере оригинальный алгоритм Бухбергера выполняет 15 лишних редукций к нулю. Для сравнения отметим, что «улучшенная» версия алгоритма Бухбергера совершает только 3 редукции к нулю. Попробуйте выполнить эти алгоритмы и проанализировать протоколы вычислений.

В заключение продемонстрируем, как построить примарное разложение идеала и найти ассоциированные простые идеалы:

```
sage: R.<x,y> = QQ[]
sage: f = (x^3 + 2*y^2*x)^2
sage: g = x^2*y^2
sage: I = (f, g)*R
sage: I.primary_decomposition()
[Ideal (x^2) of Multivariate Polynomial Ring in x, y over
 Rational Field,
 Ideal (y^2, x^6) of Multivariate Polynomial Ring in x, y over
 Rational Field]
sage: I.associated_primes()
[Ideal (x) of Multivariate Polynomial Ring in x, y over Rational
 Field,
 Ideal (y, x) of Multivariate Polynomial Ring in x, y over Rational
 Field]
sage: a,b,c = QQ['a,b,c'].gens()
sage: X,Y = GF(7)['X,Y'].gens()
sage: I = ideal(a, b^2, b^3+c^3)
sage: J = ideal(X^10 + Y^10)
sage: I.minimal_associated_primes ()
[Ideal (c, b, a) of Multivariate Polynomial Ring in a, b, c
over Rational Field]
sage: J.minimal_associated_primes ()
[Ideal (Y^4 + 3*X*Y^3 + 4*X^2*Y^2 + 4*X^3*Y + X^4) of Multivariate
```

```
Polynomial Ring in X, Y over Finite Field of size 7,
 Ideal (Y^4 + 4*X*Y^3 + 4*X^2*Y^2 + 3*X^3*Y + X^4) of Multivariate
Polynomial Ring in X, Y over Finite Field of size 7,
 Ideal (Y^2 + X^2) of Multivariate Polynomial Ring in X, Y over
Finite Field of size 7]
```

4.3 Упражнения

1. Вычислите НОД многочленов $t^4 + t^2 + 1$, $t^4 - t^2 - 2t - 1$ и $t^3 - 1$.

2. Вычислите НОД многочленов $f(t) = t^3 + 2t^2 - t - 2$, $g(t) = t^3 - 2t^2 - t + 2$ и $h(t) = t^3 - t^2 - 4t + 4$ и найдите такие многочлены $u(t)$, $v(t)$ и $w(t)$, что такие, что $\text{НОД}\,(f, g, h) = uf + vg + wh$.

3. Освободите многочлен $t^{11} - t^{10} + 2t^8 - 4t^7 + 3t^5 - 3t^4 + t^3 + 3t^2 - t - 1$ от квадратов (кратных корней).

4. Разложите многочлен $t^4 + t^3 + t + 2$ на неприводимые множители в $\mathbb{F}_3[x]$.

5. Разложите многочлен $t^3 + 2t^2 + 4t + 1$ на неприводимые множители в $\mathbb{F}_5[x]$.

6. Найдите все целые числа a, для которых все корни многочлена $t^4 - 14t^3 + 61t^2 + 84t + a$ являются целыми.

7. Найдите все значения λ, при которых многочлены $t^3 - \lambda t + 2$ и $t^2 + \lambda t + 2$ имеют общий корень.

8. Найдите все значения λ, при которых многочлен $t^4 - 4t^3 + (2 - \lambda)t^2 + 2t - 2$ имеет кратный корень.

9. Постройте над $\mathbb{Z}_7$ многочлен $f(x)$ наименьшей степени с условием $f(0) = 1$, $f(1) = 0$ и $f(k) = k$ для остальных k.

29. Решите систему уравнений

$$\begin{cases} yz + x^2 + z = 0, \\ xyz + xz - y^3 = 0, \\ xz + y^2 = 0. \end{cases}$$

30. Решите систему уравнений

$$\begin{cases} x^2 + z^2y + yz = 0, \\ y^2 - zx + x = 0, \\ xy + z^2 - 1 = 0. \end{cases}$$

4.4 Задачи

1. Напишите свою функцию, освобождающую заданный многочлен от квадратов. Постарайтесь написать алгоритм, который будет работать над полем любой характеристики.

2. Напишите свою функцию, проверяющую неприводимость многочлена над $\mathbb{Q}$ по признаку Эйзенштейна.

3. Напишите свою функцию, возвращающую k-й элементарный симметрический многочлен от n переменных.

4. Напишите функцию, проверяющую, является ли заданный многочлен симметрическим.

5. Реализуйте функцию разложения данного симметрического многочлена f по элементарным симметрическим многочленам $\sigma_1, \ldots, \sigma_n$. Функция должна возвращать многочлен h, такой, что $h(\sigma_1, \ldots, \sigma_n) = f$.

6. Пусть $s_k = x_1^k + \ldots + x_n^k$ и σ_k — элементарный симметрический многочлен. Проверьте при небольших n, что

$$s_k = \begin{vmatrix} \sigma_1 & 1 & 0 & \ldots & 0 & 0 \\ 2\sigma_2 & \sigma_1 & 1 & \ddots & 0 & 0 \\ 3\sigma_2 & \sigma_2 & \sigma_1 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ (k-1)\sigma_{k-1} & \sigma_{k-2} & \sigma_{k-3} & \ddots & \sigma_1 & 1 \\ k\sigma_k & \sigma_{k-1} & \sigma_{k-2} & \ldots & \sigma_2 & \sigma_1 \end{vmatrix}$$

и

$$\sigma_k = \frac{1}{k!} \begin{vmatrix} s_1 & 1 & 0 & \ldots & 0 & 0 \\ s_2 & s_1 & 2 & \ddots & 0 & 0 \\ s_3 & s_2 & s_1 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ s_{k-1} & s_{k-2} & s_{k-3} & \ddots & s_1 & k-1 \\ s_k & s_{k-1} & s_{k-2} & \ldots & s_2 & s_1 \end{vmatrix}.$$

7. Напишите функцию, которая по двум многочленам $f, g \in \mathbb{Q}[x]$ определяла бы

а) изоморфны ли векторные пространства $\mathbb{Q}[x]/(f)$ и $\mathbb{Q}[x]/(g)$.

б) изоморфны ли алгебры $\mathbb{Q}[x]/(f)$ и $\mathbb{Q}[x]/(g)$.

8. Напишите функцию, которая получала бы на вход квадратичную форму над $\mathbb{R}$, заданную как многочлен второй степени от переменных $x_1, \ldots, x_n$, и выясняла бы, является ли она положительно (отрицательно) определенной.

9. Не используя функции `resultant`, напишите свой (наивный) алгоритм вычисления результанта двух многочленов от одной переменной с помощью вычисления определителя матрицы Сильвестра.

10. Пусть $f \in \mathbb{R}[x]$ и $f = (x - a_1) \ldots (x - a_n)$ над $\mathbb{C}$. По теореме Виета коэффициенты многочлена f с точностью до знака равны элементарным симметрическим многочленам от корней a_i. Положим $s_k = \sum_j a_j^k$ и рассмотрим симметрическую матрицу $B = (s_{i+j-2})$ размера $n \times n$. Так как все s_k являются симметрическими многочленами от a_j, то они однозначно выражаются через коэффициенты f, что позволяет явно построить матрицу B по многочлену f, не находя его корней. Теорема Сильвестра гласит, что ранг матрицы B равен числу различных комплексных корней f, а сигнатура B (разность между положительным и отрицательным индексами инерции) — числу различных вещественных корней f.

Напишите функцию, которая по заданному многочлену с помощью этой теоремы находит число его различных комплексных и действительных корней.

11. Найдите элементарными способами базис идеала $\mathbf{I}(\mathbf{V}(x^5-2x^4+2x^2-x, x^5-x^4-2x^3+2x^2+x-1))$. Убедитесь, что функция `radical` возвращает тот же ответ (эту функцию следует использовать в кольце многочленов от нескольких переменных). В какой минимальной степени полученная образующая попадает в идеал?

12. Напишите функцию, возвращающую S-полином двух многочленов.

13. Напишите функцию, редуцирующую заданный многочлен по множеству других многочленов.

14. Напишите функцию, проверяющую, является ли заданное множество базисом Гребнера при выбранном упорядочении.

15. Напишите функцию, проверяющую, является ли заданное множество авторедуцированным (и, в частности, является ли базис Гребнера минимальным).

16. Известно, что даже если образующие полиномиального идеала имеют степень d, то вычисление базиса Гребнера может приводить к многочленам степени 2^{2^d}. Проиллюстрируйте это на примере идеала $(x^{n+1} - yz^{n-1}w, xy^{n-1} - z^n, x^nz - y^nw)$ при degrevlex-упорядочении с $x > y > z > w$. Покажите с помощью вычислений при небольших n, что в базисе Гребнера содержится многочлен степени $n^2 + 2$.

17. Найдите базис Гребнера идеала $I = (x^5 + y^4 + z^3 - 1, x^3 + y^2 + z^2 - 1)$, используя упорядочения lex и degrevlex с $x > y > z$. Убедитесь, что в degrevlex-случае базис Гребнера выглядит проще. Теперь найдите базис Гребнера идеала $I = (x^5+y^4+z^3-1, x^3+y^3+z^2-1)$ при тех же порядках (здесь вторая образующая отличается только в одном слагаемом). Сколько элементов в lex-базисе? Чему равна максимальная полная степень? Найдите наибольший коэффициент, встречающийся в многочленах базиса.

18. Покажите на примере идеала $I = (x^4 - yz^2w, xy^2 - z^3, x^3z - y^3w)$, что базисы Гребнера при lex- и degrevlex-порядках могут совпадать, так что degrevlex-упорядочение не всегда лучше, чем lex.

19. Рассмотрим систему

$$\begin{cases} t^2 + x^2 + y^2 + z^2 = 0, \\ t^2 + 2x^2 - xy - z^2 = 0, \\ t + y^3 - z^3 = 0. \end{cases}$$

Исключите t из этой системы с помощью лексикографического упорядочения.

20. Поверхность Эннепера задается следующей параметрической системой:

$$\begin{cases} x = 3u + 3uv^2 - u^3, \\ y = 3v + 3u^2v - v^3, \\ z = 3u^2 - 3v^2. \end{cases}$$

Найдите уравнение наименьшего многообразия V, содержащего эту поверхность. Постройте график этой поверхности.

21. Запишите с помощью полиномиальных уравений от координат условие и формулировку теоремы Эйлера об окружности 9 точек: середины сторон, основания высот и середины отрезков, соединяющих ортоцентр с вершинами, лежат на одной окружности. Верно ли, что уравнение, отвечающее за формулировку, принадлежит идеалу, порожденному уравнениями условия? Как учесть в этой задаче вырожденный случай?

22. Докажите с помощью полиномиальных вычислений теорему Морлея: точки пересечения смежных трисектрисс произвольного треугольника образуют равносторонний треугольник.

23. Известно, что задача о построении треугольника по трем бисектриссам циркулем и линейкой неразрешима. Пусть a, b, c — произвольные стороны треугольника. Постройте систему уравнений, из которых можно было бы найти эти стороны по биссектрисам l_a, l_b и l_c, и найдите число ее решений. С помощью исключения переменных постройте в этом идеале многочлен $g(a, l_a, l_b, l_c)$ с целыми коэффициентами от четырех переменных. Найдите явный вид этого многочлена при $l_a = 2$, $l_b = 3$ и $l_c = 6$. Что можно сказать о группе Галуа этого многочлена?

24. Постройте многочлен f с рациональными коэффициентами, такой, что в развернутой записи f^2 слагаемых меньше, чем в самом многочлене f. Найдите такой многочлен минимально возможной степени. Попробуйте записать алгоритм, который для заданного $\varepsilon > 0$ находит многочлен f минимальной степени, такой, что $|\mathrm{Supp}(f^2)| < \varepsilon|\mathrm{Supp}(f)|$, где $|\mathrm{Supp}(f)|$ – число слагаемых в развернутой записи многочлена.

Глава 5

Группы

5.1 Группы перестановок

Sage умеет работать с группами перестановок, классическими конечными группами (такими, как $\mathbf{SU}_n(\mathbb{F}_q)$), конечными матричными группами, а также абелевыми группами (в том числе, бесконечными). Значительная часть системы Sage, предназначенная для работы с группами, основывается на процедурах из пакета GAP (Groups, Algorithms, and Programming, см. http://www.gap-system.org). Группы можно задавать различными способами (например, с помощью образующих и соотношений, или с помощью вложения в группу обратимых матриц и т. д.). Мы будем задавать конечные группы как подгруппы в группе перестановок $\mathbf{S}_n$. По теореме Кэли всякая конечная группа допускает такое вложение. В Sage включено множество функций для работы с группами такого типа.

В Sage перестановки записываются в виде разложения на независимые циклы. Умножение перестановок по умолчанию производится слева направо. Записать перестановку $\sigma = (1\,3)(2\,5\,4)$ можно двумя способами:

— в виде текстовой строки: `"(1,3)(2,5,4)"`;

— в виде списка наборов: `[(1,3), (2,5,4)]`.

Вся симметрическая группа $\mathbf{S}_n$ может быть построена командой `SymmetricGroup(n)`. Используя объект, задающий группу, можно конструировать ее элементы:

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

Вычислим порядок и знак перестановки:

```
sage: sigma = G("(1,3)(2,5,4)")
sage: sigma.order()
6
sage: sigma.sign()
-1
```

Приведем список некоторых известных групп, которые можно задать в Sage (полный список можно найти в файле /sage/devel/sage/sage/groups/perm_gps/permgroup_named.py):

— `SymmetricGroup(n)`: симметрическая группа $\mathbf{S}_n$;

— `CyclicPermutationGroup(n)`: циклическая группа $\mathbf{Z}_n$;

— `AlternatingGroup(n)`: знакопеременная группа $\mathbf{A}_n$ (группа четных перестановок);

— `DihedralGroup(n)`: группа диэдра $\mathbf{D}_n$;

— `KleinFourGroup()`: четверная группа Клейна $\mathbf{V}_4$ (нециклическая группа порядка 4).

Произвольную подгруппу в группе $\mathbf{S}_n$ можно задать с помощью функции `PermutationGroup`, которая принимает на вход список образующих элементов:

```
sage: G = PermutationGroup(['(3,4)', '(1,2,3)(4,5)'])
sage: G
Permutation Group with generators [(3,4), (1,2,3)(4,5)]
sage: g = G.gen(1); g  # второй образующий элемент
(1,2,3)(4,5)
sage: g*g
(1,3,2)
sage: G.order()
120
sage: G.is_abelian()
False
sage: G.center()
Permutation Group with generators [()]
sage: G.random_element()  # случайный элемент группы
(1,5,3)(2,4)
sage: print latex(G)  # записать в формате LaTeX
\langle (3,4), (1,2,3)(4,5) \rangle
```

Вот еще один пример:

```
sage: H = DihedralGroup(6)
sage: H.is_abelian()
False
sage: H.order()
12
sage: H.list()  # список всех элементов группы
[(), (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)]
sage: H.cayley_table()  # таблица умножения группы
sage: H.center().list()
```

Попробуйте также выполнить команду `show(H.cayley_graph())` для построения графа Кэли этой группы.

5.2 Подгруппы

Породим циклическую подгруппу в группе $\mathbf{S}_5$:

```
sage: G = SymmetricGroup(5)
sage: sigma = G("(1,2,3)(4,5)")
sage: H = G.subgroup([sigma])
sage: H.list()
[(), (4,5), (1,2,3), (1,2,3)(4,5), (1,3,2), (1,3,2)(4,5)]
```

Создадим циклическую группу из 20 элементов и выпишем порядки этих элементов:

```
sage: n = 20
sage: CN = CyclicPermutationGroup(n)
sage: for g in CN:
...       print g.order(), g
```

Функция `is_normal()` позволяет проверить, является ли группа нормальной:

```
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: V4 = A4.subgroup([r1,r2,r3])
sage: V4.is_normal(A4)
True
```

Построим факторгруппу $\mathbf{A}_4/\mathbf{V}_4$:

```
sage: A4.quotient(V4)
```

Проверим простоту групп $\mathbf{A}_4$ и $\mathbf{A}_5$:

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

Найдем все нормальные подгруппы в $\mathbf{S}_4$:

```
sage: S4 = SymmetricGroup(4)
sage: S4.normal_subgroups()
[Permutation Group with generators [()],
 Permutation Group with generators [(1,3)(2,4), (1,4)(2,3)],
 Permutation Group with generators [(2,4,3), (1,3)(2,4), (1,4)(2,3)],
 Permutation Group with generators [(1,2), (1,2,3,4)]]
```

По второй теореме Силова все силовские подгруппы сопряжены. Поэтому, например, найти все силовские 2-подгруппы в $\mathbf{S}_8$ можно было бы, выбрав единственную подгруппу порядка $2^7 = 128$ в списке сопряженных подгрупп:

```
sage: G = SymmetricGroup(8)
sage: subgroups = G.conjugacy_classes_subgroups()
sage: S = [H for H in subgroups if order(H) == 2^7][0]
```

Можно поступить гораздо проще:

```
sage: S = G.sylow_subgroup(2)
```

5.3 Классы сопряженности

Найдем представителей классов сопряженности группы:

```
sage: G = PermutationGroup(['(1,2,3)', '(1,2)(3,4)', '(1,7)'])
sage: CG = G.conjugacy_classes_representatives()
sage: gamma = CG[2]
sage: CG; gamma
[(), (1,2), (1,2)(3,4), (1,2,3), (1,2,3)(4,7), (1,2,3,4), (1,2,3,4,7)]
(1,2)(3,4)
```

Найдем число элементов в каждом классе сопряженности группы $\mathbf{S}_4$:

```
sage: G = SymmetricGroup(5)
sage: group_order = G.order()
sage: reps = G.conjugacy_classes_representatives()
sage: class_sizes = [group_order/G.centralizer(g).order() for g in reps]
sage: class_sizes
[1, 10, 15, 20, 20, 30, 24]
sage: sum(class_sizes)
120
```

Если H – подгруппа в G и $g \in G$, то gHg^{-1} — тоже подгруппа, сопряженная H. Команда `G.conjugacy_classes_subgroups()` возвращает список всех подгрупп `G` с точностью до сопряжения (то есть, всякая подгруппа может быть построена из подгруппы из этого списка как gHg^{-1} для подходящего g).

```
sage: K = DihedralGroup(12)
sage: sg = K.conjugacy_classes_subgroups()
sage: sg
sage: sg[1].list()
```

5.4 Примеры

Проиллюстрируем работу с матричными группами в Sage:

```
sage: MS = MatrixSpace(GF(7), 2)  # задаем пространство матриц 2x2
sage: gens = [MS([[1,0],[-1,1]]),MS([[1,1],[0,1]])]  # список образующих
sage: G = MatrixGroup(gens)  # порождаем подгруппу
sage: G.conjugacy_class_representatives()
    [
    [1 0]
    [0 1],
    [0 1]
    [6 1],
    ...
    [6 0]
    [0 6]
    ]
sage: G = Sp(4,GF(7))  # симплектическая группа
sage: G
Symplectic Group of rank 2 over Finite Field of size 7
sage: G.random_element()
[5 5 5 1]
```

```
[0 2 6 3]
[5 0 1 0]
[4 6 3 4]
sage: G.order()
276595200
sage: G = SL(2, GF(5))
sage: G.center()
Matrix group over Finite Field of size 5 with 1 generators:
 [[[4, 0], [0, 4]]]
```

Обратите внимание, что Sage представляет классические конечные группы как группы перестановок:

```
sage: G = PSL(2, 5); G # Проективная специальная линейная группа
Permutation Group with generators [(3,5)(4,6), (1,2,5)(3,4,6)]
sage: G.center()
Permutation Group with generators [()]
```

Теперь прокомментируем работу с абелевыми группами:

```
sage: F = AbelianGroup(5, [5,5,7,8,9], names='abcde')
Multiplicative Abelian Group isomorphic to C5 x C5 x C7 x C8 x C9
sage: (a, b, c, d, e) = F.gens()
sage: d * b^2 * c^3
b^2*c^3*d
sage: F = AbelianGroup(3,[2]*3); F
Multiplicative Abelian Group isomorphic to C2 x C2 x C2
sage: H = AbelianGroup([2,3], names="xy"); H
Multiplicative Abelian Group isomorphic to C2 x C3
sage: AbelianGroup(5)  # свободная абелева группа ранга 5
Multiplicative Abelian Group isomorphic to Z x Z x Z x Z x Z
sage: AbelianGroup(5).order()
+Infinity
```

5.5 Упражнения

1. Постройте в соответствующих группах матриц $\mathbf{GL}_n(\mathbb{C})$ подгруппы, изоморфные группам $\mathbf{Q}_8$ и $\mathbf{D}_n$.

2. Проверьте для некоторых n с помощью Sage следующую теорему: группа обратимых элементов кольца $\mathbb{Z}_n$ является циклической тогда и только тогда, когда либо $n = 2$, либо $n = 4$, либо $n = p^k$, либо $n = 2p^k$, где p – нечетное простое число.

3. Найдите все нормальные подгруппы в группах $\mathbf{A}_4$, $\mathbf{S}_4$ и $\mathbf{A}_5$. Для нетривиальных нормальных подгрупп укажите, чему изоморфны соответствующие факторгруппы.

4. Найдите число элементов порядка 7 в простой группе порядка 168.

5. Найдите все нормальные подгруппы в группе $\mathbf{SL}_2(\mathbb{Z}_3)$.

6. Найдите классы сопряженности групп $\mathbf{A}_4$, $\mathbf{A}_5$, $\mathbf{S}_5$, $\mathbf{S}_6$. Есть ли в группах $\mathbf{S}_5$ и $\mathbf{S}_6$ несопряженные элементы одинаковых порядков?

7. Проверьте, что силовские 2-подгруппы группы $\mathbf{S}_4$ изоморфны группе $\mathbf{D}_4$.

8. Проверьте, что силовская 2-подгруппа группы $\mathbf{SL}_2(\mathbb{Z}_3)$ является нормальной и изоморфна группе кватернионов.

9. Проверьте, что $\mathbf{A}_5$ порождается двумя подстановками, (254) и (12345).

10. Найдите факторгруппу группы $\mathbf{SL}_2(\mathbb{Z}_5)$ по ее центру. Является ли она простой?

5.6 Задачи

1. Задайте в Sage группы симметрий и вращений правильных многогранников как группы перестановок их вершин. Задайте также группу симметрий куба как группу перестановок его диагоналей. Каким классическим группам изоморфны эти группы симметрий?

2. Найдите коммутанты и факторгруппы по коммутантам для групп $\mathbf{A}_4$, $\mathbf{S}_4$, $\mathbf{Q}_8$ и $\mathbf{D}_5$.

3. Напишите функцию, которая по заданной группе и простому p находит число силовских p-подгрупп.

4. Напишите функцию, проверяющую, изоморфны ли группы $\mathbb{Z}_{m_1} \oplus \ldots \oplus \mathbb{Z}_{m_s}$ и $\mathbb{Z}_{n_1} \oplus \ldots \oplus \mathbb{Z}_{n_t}$.

5. Напишите функцию, строящую все абелевы группы заданного порядка.

6. Напишите функцию, которая по списку целых чисел $m_1, \ldots, m_s$ строит разложение группы $\mathbb{Z}_{m_1} \oplus \ldots \oplus \mathbb{Z}_{m_s}$ в прямую сумму примарных циклических групп.

7. Напишите функцию, которая по списку целых чисел $m_1, \ldots, m_s$ строит разложение группы $\mathbb{Z}_{m_1} \oplus \ldots \oplus \mathbb{Z}_{m_s}$ в прямую сумму вида $\mathbb{Z}_{d_1} \oplus \ldots \oplus \mathbb{Z}_{d_r}$, где $d_1 \mid d_2 \mid \ldots \mid d_r$.

8. Пусть A — свободная абелева группа с базисом $x_1, \ldots, x_n$, а B — ее подгруппа, порожденная $y_1, \ldots, y_k$, причем

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = M \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix}$$

для некоторой целочисленной матрицы M. Напишите свою функцию, которая получает матрицу M и раскладывает факторгруппу A/B в прямую сумму циклических групп.

9. В факторгруппе свободной абелевой группы A с базисом x_1, x_2, x_3 по подгруппе B, порожденной $x_1 + x_2 + 4x_3$ и $2x_1 - x_2 + 2x_3$, найти порядок смежного класса $(x_1 + 2x_3) + B$.

10. В факторгруппе свободной абелевой группы A с базисом x_1, x_2, x_3 по подгруппе B, порожденной $2x_1 + x_2 - 50x_3$ и $4x_1 + 5x_2 + 60x_3$, найти порядок элемента $32x_1 + 31x_2 + B$.

11. Является ли разрешимой подгруппа в $\mathbf{S}_7$, порожденная перестановками (123) и (14567)?

12. Напишите функцию, строящую все порождающие элементы мультипликативной группы конечного поля заданного порядка.

13. Задайте в Sage все попарно неизоморфные группы из 12 элементов.

14. Создайте таблицу всех групп, порядок которых не превосходит 18.

15. Найдите все группы, порядок которых не превосходит 100, у которых пересечение всех нетривиальных нормальных подгрупп нетривиально.

16. Рассмотрим все группы, порядок которых не превосходит 2000. Групп какого порядка существует больше всего?

17. Задайте группу кубика Рубика $2 \times 2 \times 2$ как подгруппу в группе перестановок. Найдите ее порядок. Вычислите максимальное количество операций, которое может потребоваться для сборки такого кубика при условии, что он собирается самым оптимальным образом. Укажите последовательность действий для сборки заданной комбинации кубика. Решите эту же задачу для кубика Рубика $n \times n \times n$.

18. Напишите серию функций, которые по заданной квадратной таблице размера $n \times n$ с элементами $1, 2, \ldots, n$ определяют, задает ли эта таблица

— коммутативную бинарную операцию;

— ассоциативную бинарную операцию;

— моноид (если да, то функция должна вернуть нейтральный элемент);

— группу.

19. Напишите функцию, которая по двум таблицам, задающим бинарную операцию, проверяет, изоморфны ли множества с этой бинарной операцией.

20. В условиях предыдущей задачи найдите все полугруппы и моноиды из двух и трех элементов.

Литература

[1] И. В. Аржанцев. Базисы Гребнера и системы алгебраических уравнений. М.: МЦНМО, 2003.

[2] Э. Б. Винберг. Курс алгебры // М.: Факториал пресс, 2002.

[3] Д. Кокс, Д. Литтл, Д. О'Ши. Идеалы, многообразия и алгоритмы. Введение в вычислительные аспекты алгебраической геометрии и коммутативной алгебры // М.: Мир, 2000.

[4] Сборник задач по алгебре // Под ред. А. И. Кострикина. М.: МЦНМО, 2009.

[5] А. И. Кострикин. Введение в алгебру (в 3-х частях) // М.: Физматлит, 2001.

[6] В. Н. Латышев. Комбинаторная теория колец. Стандартные базисы // М., Издательство МГУ, 1988.

[7] А. В. Михалев, А. А. Михалев. Начала алгебры // ИНТУИТ, 2005.

[8] П. Ноден, К. Китте. Алгебраическая алгоритмика // М.: Мир, 1999.

[9] Е. В. Панкратьев. Элементы компьютерной алгебры // М., БИНОМ – Лаборатория зна-ний, 2007.

[10] В. В. Прасолов. Многочлены // М.: МЦНМО, 2001.

[11] Th. Becker, V. Weispfenning. Groebner Bases — A Computational Approach To Commutative Algebra. Springer, New York, 1993.

[12] D. Cox, J. Little, D. O'Shea. Using Algebraic Geometry // Springer-Verlag, 2004.

[13] M. Creuzer, L. Robbiano. Computational Commutative Algebra (in two parts) // Springer-Verlag, 2001, 2005.

[14] G.-M. Greuel, G. Pfister, and H. Schoenemann. Singular 3.0. A Computer Algebra System for Polynomial Computations. Center for Computer Algebra, University of Kaiserslautern (2005). http://www.singular.uni-kl.de.

[15] D. Krob, S. Legros. Algèbre générale et linéaire. Introduction au calcul symbolique et aux mathématiques expérimentales. Vuibert, Paris, 1999.

[16] D. Joyner, W. Stein. Sage Installation Guide.
http://www.sagemath.org/doc/inst/inst.html

[17] D. Joyner. Constructions in Sage. http://www.sagemath.org/doc/const/const.html

[18] W. Stein. The Sage Tutorial. http://www.sagemath.org/doc/tut/tut.html

[19] W. Stein. Sage Reference Manual. http://www.sagemath.org/doc/ref/ref.html

[20] W. Stein. Sage Developer's Guide. http://www.sagemath.org/doc/prog/prog.html

[21] W. Stein et al., Sage Mathematics Software (Version 4.3). The Sage Development Team, 2009, http://www.sagemath.org.