Министерство образования и науки Российской Федерации

САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Приоритетный национальный проект «Образование»
Национальный исследовательский университет

Н. Н. ВАСИЛЬЕВ Ф. А. НОВИКОВ

КОМПЬЮТЕРНАЯ АЛГЕБРА

ЧАСТЬ I.
ДИСКРЕТНАЯ МАТЕМАТИКА,
ТЕОРИЯ АЛГОРИТМОВ

Учебное пособие

Рекомендовано Учебно-методическим объединением по университетскому политехническому образованию в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки магистров «Прикладная математика и информатика»

Санкт-Петербург
Издательство Политехнического университета
2011

УДК
ББК
Т

Рецензент:
Доктор физико-математических наук,
Ведущий научный сотрудник
Санкт-Петербургского отделения математического института
имени В. А. Стеклова РАН
Николай Витальевич Проскурин

Васильев Н. Н **Компьютерная алгебра. Часть I. Дискретная математика, теория алгоритмов:** учеб. пособие / Васильев Н. Н., Новиков Ф. А. – СПб.: Изд-во Политехн. ун-та, 2011. – 197 с.

ISBN

В пособии рассматриваются основные понятия дискретной математики, которая имеет широкий спектр приложений, прежде всего в областях, связанных с информационными технологиями и компьютерами.

Важнейшими приложениями дискретных структур в программировании являются компьютерная алгебра и вычислительная геометрия. Основу рассмотрений составляет теория алгоритмов. Именно этот круг тем покрывает данное пособие.

Учебное пособие предназначено для студентов вузов, обучающихся по магистерской программе «Математическое и программное обеспечение компьютерных систем» направления подготовки магистров «Прикладная математика и информатика». Оно может быть также использовано при обучении в системах повышения квалификации и в учреждениях дополнительного профессионального образования.

Работа выполнена в рамках реализации программы развития национального исследовательского университета «Модернизация и развитие политехнического университета как университета нового типа, интегрирующего мультидисциплинарные научные исследования и надотраслевые технологии мирового уровня с целью повышения конкурентоспособности национальной экономики»

Печатается по решению редакционно-издательского совета
Санкт-Петербургского государственного политехнического университета.

ISBN

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В этом пособии рассматриваются некоторые элементарные понятия "дискретной", или "конечной", математики, то есть математики, прежде всего изучающей конечные множества и различные структуры на них. Это означает, что понятия бесконечности, предела или непрерывности не являются предметом изучения, хотя могут использоваться как вспомогательные средства. Дискретная математика имеет широкий спектр приложений, прежде всего в областях, связанных с информационными технологиями и компьютерами. В самом первоначальном, ныне редко используемом названии компьютера — "электронная цифровая вычислительная машина" — слово "цифровая" указывает на принципиально дискретный характер работы данного устройства. Современные компьютеры неотделимы от современных программистов, для которых и написана эта книга.

Важнейшими приложениями дискретных структур в программировании являются компьютерная алгебра и вычислительная геометрия. Основу рассмотрений составляет теория алгоритмов. Именно этот круг тем покрывает данное пособие.

Назначение и особенности книги

Данная книга представляет собой учебное пособие по дискретной математике, теории алгоритмов и компьютерной алгебре, включающее в себя описания важнейших алгоритмов над объектами дискретной математики. Учебное пособие ориентировано на студентов специальностей, связанных с компьютерами и программированием, которым по роду их занятий приходится иметь дело с конструированием и анализом нетривиальных алгоритмов.

Данное пособие принадлежит к такому жанру учебной математической литературы, в котором математическое изложение доводится до уровня практически исполнимых программ. Он отличается большей *широтой*, но, пожалуй, меньшей *глубиной* охвата материала.

Учебное пособие основано на ряде лекционных курсов, который авторы в течение многих лет читают студентам кафедры "Прикладная математика" Санкт-Петербургского государственного политехнического университета, что наложило определённый отпечаток на выбор материала. Пособие охватывает почти все основные разделы дискретной математики: теорию множеств, общую алгебру, комбинаторику и теорию графов. Кроме того, представлены и некоторые более специальные разделы, необходимые программистам, такие как теория алгоритмов и булевы функции.

Учебное пособие преследует три основные цели:

1) познакомить читателя с максимально широким кругом понятий дискретной математики. Количество определяемых и упоминаемых понятий и специальных терминов намного превышает количество понятий, обсуждаемых более детально. Тем самым у студента формируется терминологический запас, необходимый для самостоятельного изучения специальной математической и теоретико-программистской литературы;

2) сообщить читателю необходимые конкретные сведения из дискретной математики, предусматриваемые стандартной программой технических высших учебных заведений. Разбор доказательств приведенных утверждений и выполнение упражнений позволят студенту овладеть методами дискретной математики, наиболее употребительными при решении практических задач;

3) пополнить запас примеров нетривиальных алгоритмов. Изучение алгоритмов решения типовых задач дискретной математики и способов представления математических объектов в программах абсолютно необходимо практикующему программисту, поскольку позволяет уменьшить трудозатраты на "изобретение велосипеда" и существенно обогащает навыки конструирования алгоритмов.

Структура книги

Книга, фактически, делится на две части. Первая часть, которая входит в этой пособие, содержит самые общие сведения из основных

разделов дискретной математики, теории алгоритмов и компьютерной алгебры. Во второй части собраны различные специальные разделы, которые можно включать или не включать в учебный курс в зависимости от конкретных обстоятельств.

Главы делятся на разделы, которые, в свою очередь, делятся на параграфы. Каждый раздел посвящён одному конкретному вопросу темы главы и по объёму соответствует экзаменационному вопросу. Параграфы нужны для внутренних ссылок и более детальной структуризации материала. Как правило, в параграфе рассматривается какое-нибудь одно понятие, теорема или алгоритм. Параграфы в пределах раздела упорядочены по сложности: сначала приводятся основные определения, а затем рассматриваются более сложные вопросы, которые, в принципе, при первом чтении можно пропустить без ущерба для понимания остального материала.

Главы книги более или менее независимы и могут изучаться в любом порядке, за исключением первой главы, которая содержит набор базисных определений для дальнейшего изложения

Используемые обозначения

Данная книга предназначена для программистов, то есть предполагается, что студент не испытывает затруднений в понимании текстов программ на языке высокого уровня. Именно поэтому в качестве нотации для записи алгоритмов используется *некоторый* неспецифицированный язык программирования (*псевдокод*), похожий по синтаксису на Паскаль (но, конечно, Паскалем не являющийся).

Ключевые слова псевдокода выделены жирным шрифтом, идентификаторы обьектов записываются курсивом, как это принято для математических объектов, примечания отделяются двумя символами "слеш" //.

В языке используются общеупотребительные структуры управления: ветвления в краткой (**if** ... **then** ... **end if**) и полной (**if** ... **then** ... **else** ... **end if**) формах, циклы со счётчиком (**for** ... **from** ... **to** ... **do** ... **end for**), с постусловием (**repeat** ... **until**), с предусловием (**while** ... **do**

... **end while**) и по множеству (**for** ... **do** ... **end for**), а также переходы (**go to** ...).

Для обозначения присваивания используется традиционный знак :=, вызов процедуры или функции ключевыми словами не выделяется, выход из процедуры вместе с возвратом значения обозначается ключевым словом **return**.

Подразумевается строгая статическая типизация, даже приведения не используются, за исключением разыменования, которое подразумевается везде, где оно нужно по здравому программистскому смыслу. В то же время объявления локальных переменных почти всегда опускаются, если их тип ясен из контекста.

Из структур данных считаются доступными массивы (**array** [...] **of** ...), структуры (**record** ... **end record**) и указатели (↑).

В программах широко используются математические обозначения, которые являются самоочевидными в конкретном контексте. Например, конструкция

for $x \in X$ **do**

$P(x)$

end for

означает применение процедуры P ко всем элементам множества X.

"Лишние" разделители систематически опускаются, функции могут возвращать несколько значений, и вообще, разрешаются любые вольности, которые позволяют сделать тексты программ более краткими и читабельными.

Особого упоминания заслуживают три "естественных" оператора, аналоги которых в явном виде редко встречаются в настоящих языках программирования. Оператор

select $m \in M$

означает выбор *произвольного* элемента m из множества M. Этот оператор часто необходим в "переборных" алгоритмах. Оператор

yield x

означает возврат значения x, но при этом выполнение функции не прекращается, а продолжается со следующего оператора. Этот оператор позволяет очень просто записать "генерирующие" алгоритмы, результатом которых является некоторое заранее неизвестное множество значений. Оператор

next for x

означает прекращение выполнения текущего фрагмента программы и продолжение выполнения со следующего шага цикла по переменной x. Этот оператор должен находиться в теле цикла по x (на любом уровне вложенности). Такого рода "структурные переходы" позволяют описывать обработку исключительных ситуаций в циклах непосредственно, без введения искусственных переменных, отслеживающих состояние вычислений в цикле.

Операторы **select**, **yield** и **next for** не являются чем-то необычным и новым: первый из них легко моделируется использованием датчика псевдослучайных чисел, второй — присоединением элемента к результирующему множеству, а третий — переходом по метке. Однако указанные операторы сокращают запись и повышают её наглядность.

Поскольку эта книга написана на "программно-математическом" языке, в ней не только математические обозначения используются в программах, но и некоторые программистские обозначения используются в математическом тексте.

Прежде всего, обсудим способы введения обозначений объектов и операций. Если обозначение объекта или операции является глобальным (в пределах учебного пособия), то используется знак **:=**, который следует читать "по определению есть". При этом слева от знака **:=** может стоять обозначение объекта, а справа — выражение, определяющее этот объект. Например,

$$\mathbb{R}_+ := \{x \in \mathbb{R} \mid x > 0\}.$$

В случае определения операции в левой части стоит *выражение*. В этом случае левую часть следует неформально понимать как заголовок процедуры выполнения определяемой операции, а правую часть — как тело этой процедуры. Например, формула

$$A = B \mathbf{:=} A \subseteq B \ \& \ B \subseteq A$$

означает следующее: "чтобы вычислить значение выражения $A = B$, нужно вычислить значения выражений $A \subseteq B$ и $B \subseteq A$, а затем вычислить конъюнкцию этих значений".

Отметим широкое использование символа присваивания :=, который в математическом контексте можно читать как "положим". Если в левой части такого присваивания стоит простая переменная, а в правой части — выражение, определяющее конкретный объект, то это имеет очевидный смысл введения локального (в пределах доказательства, определения и т. п.) обозначения. Например, запись

$$Z' := Z \cup Z_k \setminus \{e_k\}$$

означает: "положим, что Z' содержит все элементы Z и Z_k за исключением элемента e_k. Наряду с обычным для математики "статическим" использованием знака := когда выражению в левой части придается постоянное в данном контексте значение (определение константы), знак присваивания используется и в "динамическом", программистском смысле. Например, пусть в множестве X есть элемент x. Тогда формулу $X := X - x$ следует читать и понимать так: "удалим из множества X элемент x ".

Одной из задач книги является выработка у студентов навыка чтения математических текстов. Поэтому, начиная с самой первой страницы, без дополнительных объяснений интенсивно используются язык исчисления предикатов и другие общепринятые математические обозначения. При этом стиль записи формул совершенно свободный и неформальный, но соответствующий общепринятой практике записи формул в математических текстах. Например, вместо формулы

$$\forall k\,((k < n) \Rightarrow P(k))$$

может быть написано

$$\forall k < n\,(P(k))$$

или даже

$$\forall k < n\,P(k)$$

в предположении, что читатель знает или догадается, какие синтаксические упрощения используются.

В первых главах книги основные утверждения (формулировки теорем) дублируются, то есть приводятся на естественном языке и на языке формул. Тем самым на примерах объясняется используемый язык формул. Но в последних частях книги и в доказательствах использование естественного языка сведено к минимуму. Это существенно сокращает объём текста, но требует внимания при чтении.

Для краткости в тексте книги иногда используются обороты, которые подразумевают восстановление читателем опущенных слов по контексту. Например, если символ A означает множество, то вместо аккуратного, но длинного оборота "где объект x является элементом множества A", может быть использована более короткая фраза "где x — элемент A" или даже формула "где $x \in A$".

В целом используемые обозначения строго следуют классическим образцам, принятым в учебной математической и программистской литературе. Непривычным может показаться только совместное использование этих обозначений в одном тексте. Но такое взаимопроникновение обозначений продиктовано основной задачей книги.

Выделения в тексте

В пособии имеются следующие основные виды текстов: определения, теоремы, леммы и следствия, доказательства и обоснования, замечания, алгоритмы и примеры. Фактически, обычный связующий текст сведен к минимуму в целях сокращения объёма книги.

Определения никак специально не выделяются, поскольку составляют львиную долю основного текста книги. Вместо этого курсивом выделяются *определяющие вхождения* терминов, а текст, соседствующий с термином, и есть определение.

Формулировки теорем, лемм и следствий в соответствии с общепринятой в математической литературе практикой выделены *курсивом*. При этом формулировке предшествует соответствующее ключевое слово: "**теорема**", "**лемма**" или "**следствие**". Как правило, утверждения не нумеруются, за исключением случаев вхождения нескольких однородных утверждений в один параграф. Для ссылок на утверждения используются либо номера параграфов, в которых утверждения сформулированы, либо собственные имена утверждений. Дело в том, что в данном учебном пособии теоремами оформлены как простые утверждения, являющиеся непосредственными следствиями определений, так и глубокие факты, действительно заслуживающие статуса теоремы. В последнем случае приводится собственное имя (название теоремы), которое либо выносится в название параграфа (или раздела), либо указывается в скобках после слова "теорема".

Доказательства относятся к предшествующим утверждениям, а *обоснования* — к предшествующим алгоритмам. В пособии практически нет недоказанных утверждений и приведенных без достаточного обоснования алгоритмов. Из этого правила имеется несколько исключений, то есть утверждений, доказательства которых не приводятся, поскольку авторы считают их технически слишком сложными для выбранного уровня изложения. Отсутствие технически сложного доказательства всегда явно отмечается в тексте. Иногда же доказательство опускается, потому что утверждение легко может быть обосновано читателем самостоятельно. Такие случаи не отмечаются в тексте. Доказательства и обоснования начинаются с соответствующего ключевого слова ("**доказательство**" или "**обоснование**") и заканчиваются специальным значком □. Если формулировка теоремы содержит несколько утверждений или если доказательство состоит из нескольких

шагов (частей), то оно делится на абзацы, а в начале каждого абзаца указывается [в скобках], что именно в нем доказывается.

Замечания также начинаются с соответствующего ключевого слова: "замечание" и занимают один абзац. Замечание сообщает некоторые специальные или дополнительные сведения, относящиеся к основному материалу учебного пособия.

Замечание. В очень редких случаях *курсив* используется не для выделения определяющих вхождений терминов и формулировок утверждений, а для того, чтобы сделать эмфатическое ударение на каком-то слове в тексте.

Алгоритмы, как уже было сказано, записаны на псевдокоде, синтаксис которого кратко описан выше. Как правило, перед текстом алгоритма на естественном языке указываются его назначение, а также входные и выходные данные. Ключевые слова в текстах алгоритмов выделяются полужирным шрифтом. Исполняемые операторы, как правило, комментируются, чтобы облегчить их понимание. После алгоритма приводятся его обоснование и иногда пример протокола выполнения.

Примеры, как правило, приводятся непосредственно вслед за определением понятия, поэтому не используется никаких связующих слов, поясняющих, к чему относятся примеры. В самих примерах интенсивно используются факты, которые должны быть известны читателю из курса математики средней школы, и понятия, рассмотренные в предшествующем тексте книги.

1. МНОЖЕСТВА И ОТНОШЕНИЯ

Понятия "множество", "отношение", "функция" и близкие к ним составляют основной словарь дискретной (равно как и "непрерывной") математики. Именно эти базовые понятия рассматриваются в первой главе, тем самым закладывается необходимая основа для дальнейших построений. Особенность изложения состоит в том, что здесь рассматриваются почти исключительно *конечные* множества, а тонкие и сложные вопросы, связанные с рассмотрением бесконечных множеств, излагаются с "программистской" точки зрения. С другой стороны, значительное внимание уделяется "представлению" множеств в программах. Эти вопросы не имеют прямого отношения к собственно теории множеств в её классическом виде, но очень важны для практикующего программиста.

1.1. МНОЖЕСТВА

При построении доступной для рационального анализа картины мира часто используется термин "объект" для обозначения некой сущности, отделимой от остальных. Выделение объектов — это не более чем произвольный акт нашего сознания. В одной и той же ситуации объекты могут быть выделены по-разному, в зависимости от точки зрения, целей анализа и других обстоятельств. Но как бы то ни было, выделение объектов и их совокупностей — естественный (или даже единственно возможный) способ организации нашего мышления, поэтому неудивительно, что он лежит в основе главного инструмента описания точного знания — математики.

1.1.1. Элементы и множества

Понятие множества принадлежит к числу фундаментальных понятий математики. Можно сказать, что *множество* — это любая определённая совокупность объектов. Объекты, из которых составлено множество, называются его *элементами*. Элементы множества раз-

личны и отличимы друг от друга. Как множествам, так и элементам можно давать имена или присваивать символьные обозначения.

Примеры. Множество S страниц в данной книге. Множество $\mathbb{N}$ натуральных чисел {1, 2, 3, ...}. Множество *простых* чисел {2, 3, 5, 7, 11, ...}. Множество $\mathbb{Z}$ целых чисел {…, –2, –1, 0, 1, 2, ...}. Множество $\mathbb{R}$ *вещественных* чисел. Множество A различных символов на этой странице.

Если объект x является элементом множества M, то говорят, что x *принадлежит* множеству M. Обозначение: $x \in M$. В противном случае говорят, что x *не принадлежит* M. Обозначение: $x \notin M$.

Обычно множества обозначают прописными буквами латинского алфавита, а элементы множеств — строчными буквами.

Множество, не содержащее элементов, называется *пустым.* Обозначение: $\varnothing$. Введение в рассмотрение пустого множества является сильным допущением. Например, известно, что синих лошадей в природе не бывает. Тем не менее, мы позволяем себе рассматривать "множество синих лошадей" как полноправный объект, вводить для него обозначения и т. д.

Понятия множества, элемента и принадлежности, которые на первый взгляд представляются интуитивно ясными, при ближайшем рассмотрении такую ясность утрачивают. Во-первых, даже отличимость элементов при более глубоком рассмотрении представляет некоторую проблему. Например, символы "а" и "*а*", которые встречаются на этой странице, — это один элемент множества A или два разных элемента? Или же, например, два вхождения символа "о" в слово "множество" — графически они неотличимы невооружённым глазом, но это символы букв, которые обозначают звуки, а читаются эти две гласные по-разному: первая под ударением, а вторая — безударная. Во-вторых, проблематична возможность (без дополнительных усилий) указать, принадлежит ли данный элемент данному множеству. Например, является ли число 869584769215374850678523 простым?

Множества как объекты могут быть элементами других множеств. Множество, элементами которого являются множества, иногда называют *семейством.*

Замечание. Семейства множеств часто обозначают прописными "рукописными" буквами латинского алфавита.

Совокупность объектов, которая *не* является множеством, называется *классом*. Неформально говоря, называя совокупность элементов классом, а не множеством, мы берём на себя сравнительно меньшую ответственность за определённость, отличимость и бесповторность элементов.

Обычно в конкретных рассуждениях элементы всех рассматриваемых множеств (и семейств) берутся из некоторого одного, достаточно широкого множества U (своего для каждого случая), которое называется *универсальным множеством* (или *универсумом*).

1.1.2. Задание множеств

Чтобы задать множество, нужно указать, какие элементы ему принадлежат. Это указание заключают в пару фигурных скобок, оно может иметь одну из следующих основных форм:

- *перечисление элементов* $M := \{a, b, c, ..., z\}$;
- *характеристический предикат* $M := \{x \mid P(x)\}$;
- *порождающая процедура* $M := \{x \mid x := f\}$.

При задании множеств перечислением обозначения элементов разделяют запятыми. Характеристический предикат — это некоторое условие, выраженное в форме логического утверждения или процедуры, возвращающей логическое значение, и позволяющее проверить, принадлежит ли любой данный элемент множеству. Если для данного элемента условие выполнено, то он принадлежит определяемому множеству, в противном случае — не принадлежит. Порождающая процедура — это процедура, которая в процессе работы порождает объекты, являющиеся элементами определяемого множества.

Примеры.

1) $M_9 := \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

2) $M_9 := \{n \mid n \in \mathbb{N} \ \& \ n < 10\}$.%

3) $M_9 := \{n \mid n := 0;$ **for** i **from** 1 **to** 9 **do** $n := n + 1;$ **yield** n **end for**$\}$.

При задании множеств перечислением обозначения элементов иногда снабжают индексами и указывают множество, из которого берутся индексы. В частности, запись $\{a_i\}_{i=1}^{k}$ означает то же, что $\{a_1, ..., a_k\}$, а запись $\mathcal{M} = \{M_\alpha\}_{\alpha \in A}$ означает, что $\mathcal{M}$ является семейством, элементами которого являются множества M_α, причём индекс α "пробегает" множество A. Знак многоточия (...), который употребляется при задании множеств, является сокращённой формой записи, в которой порождающая процедура для множества индексов считается очевидной.

Пример. $\{a_1, a_2, a_3, ... \} = \{a_i\}_{i=1}^{\infty}$.

Замечание. Множество целых чисел в диапазоне от m до n в этой книге обозначается так: $m..n$. То есть

$m..n$ **:=** $\{k \in \mathbb{Z} \mid m \le k \ \& \ k \le n\} =$

$= \{k \in \mathbb{Z} \mid$ **for** k **from** m **to** n **do yield** k **end for**$\}$.

Перечислением элементов можно задавать только конечные множества. Бесконечные множества задаются характеристическим предикатом или порождающей процедурой.

Пример.

$\mathbb{N}$ **:=** $\{n \mid n := 0;$ **while true do** $n := n + 1;$ **yield** n **end while**$\}$.

1.1.3. Парадокс Рассела

Возможность задания множеств характеристическим предикатом зависит от предиката. Использование некоторых предикатов для этой цели может приводить к противоречиям. Например, все рассмотренные в примерах множества не содержат себя в качестве элемента. Рассмотрим множество Y всех множеств, не содержащих себя в качестве элемента:

$$Y := \{X \mid X \notin X\}.$$

Если множество Y существует, то мы должны иметь возможность ответить на следующий вопрос: $Y \in Y$? Пусть $Y \in Y$, тогда $Y \notin Y$. Пусть $Y \notin Y$, тогда $Y \in Y$. Получается неустранимое логическое противоречие, которое известно как *парадокс Рассела*. Вот три способа избежать этого конкретного парадокса.

1) Ограничить используемые характеристические предикаты видом $P(x) = x \in A \;\&\; Q(x)$, где A — известное, заведомо существующее множество (*универсум*). Обычно при этом используют обозначение $\{x \in A \mid Q(x)\}$. Для Y универсум не указан, а потому Y множеством не является.

2) Теория типов. Объекты имеют тип 0, множества элементов типа 0 имеют тип 1, множества элементов типа 0 и 1 — тип 2 и т. д. Y не имеет типа и множеством не является.

3) Явный запрет принадлежности множества самому себе: $X \in X$ — недопустимый предикат. При аксиоматическом построении теории множеств соответствующая аксиома называется *аксиомой регулярности*.

Замечание. Существование и анализ парадоксов в теории множеств способствовали развитию так называемого *конструктивизма* — направления в математике, в рамках которого рассматриваются только такие объекты, для которых известны процедуры (алгоритмы) их порождения. В конструктивной математике исключаются из рассмотрения те понятия и методы классической математики, которые не заданы алгоритмически.

Парадокса Рассела можно избежать, ограничив рассматриваемые множества. Например, достаточно запретить рассматривать в качестве множеств классы, содержащие самих себя. При этом, однако, нет полной уверенности в том, что не обнаружатся другие противоречия. Полноценным выходом из ситуации являлось бы аксиоматическое построение теории множеств и доказательство непротиворечиво-

сти построенной формальной теории. Однако исследование парадоксов и непротиворечивости систем аксиом является технически трудной задачей и уводит далеко в сторону от программистской практики, для которой важнейшими являются конечные множества. Поэтому формальная аксиоматика теории множеств здесь не приводится. Мы излагаем необходимые сведения полуформально, опираясь везде, где это возможно, на программистскую интуицию читателя.

1.1.4. Мультимножества

В множестве все элементы различны, а значит, входят в множество ровно один раз. В некоторых случаях оказывается полезным рассматривать совокупности элементов, в которые элементы входят по несколько раз.

Пусть $X = \{x_1, \ldots, x_n\}$ — некоторое (конечное) множество и пусть $\{a_1, \ldots, a_n\}$ — неотрицательные целые числа. Тогда *мультимножеством* $\underline{\underline{X}}$ над множеством X называется совокупность элементов множества X, в которую элемент x_i входит a_i раз, $a_i \geq 0$.

Пример.

Пусть $X = \{a, b, c\}$. Тогда $\underline{\underline{X}} = [a^0\ b^3\ c^4] = \langle 0(a), 3(b), 4(c)\rangle$ — мультимножство над X, содержащее элемент a ноль раз, элемент b — три раза, а элемент c — четыре раза.

Замечание. Элементы мультимножества, равно как и элементы множества, считаются неупорядоченными.

Пусть $\underline{\underline{X}} = \langle a_1(x_1), \ldots, a_n(x_n)\rangle$ — мультимножество над множеством $X = \{x_1, \ldots, x_n\}$. Тогда число a_i называется *показателем* элемента x_i, множество X — *носителем* мультимножества $\underline{\underline{X}}$, число $m = a_1 + \ldots + a_n$ — *мощностью* мультимножества $\underline{\underline{X}}$, а множество $\underline{X} = \{x_i \in X \mid a_i > 0\}$ называется *составом* мультимножества $\underline{\underline{X}}$.

Пример. Пусть $\underline{\underline{X}} = [a^0\ b^3\ c^4]$ — мультимножество над множеством $X = \{a, b, c\}$. Тогда $\underline{X} = \{b, c\}$.

1.2. АЛГЕБРА ПОДМНОЖЕСТВ

Самого по себе понятия множества ещё недостаточно — нужно определить способы конструирования новых множеств из уже имеющихся, то есть определить операции над множествами.

1.2.1. Сравнение множеств

Множество A *содержится* в множестве B (множество B *включает* множество A), если каждый элемент множества A есть элемент множества B:

$$A \subseteq B := x \in A \Rightarrow x \in B.$$

В этом случае A называется *подмножеством* B, B — *надмножеством* A. По определению $\forall M\ (\varnothing \subseteq M)$.

Два множества *равны*, если они являются подмножествами друг друга:

$$A = B := A \subseteq B \ \& \ B \subseteq A.$$

Теорема. *Включение множеств обладает следующими свойствами*:

1) $\forall A\ (A \subseteq A)$;

2) $\forall A, B\ (A \subseteq B \ \& \ B \subseteq A \Rightarrow A = B)$;

3) $\forall A, B, C\ (A \subseteq B \ \& \ B \subseteq C \Rightarrow A \subseteq C)$.

Доказательство.

[1**]** Имеем $x \in A \Rightarrow x \in A$, и значит, $A \subseteq A$.

[2**]** По определению.

[3**]** Если $x \in A \Rightarrow x \in B$ и $x \in B \Rightarrow x \in C$, то $x \in A \Rightarrow x \in C$, и значит, $A \subseteq C$.□

Если $A \subseteq B$ и $A \neq B$, то множество A называется *собственным* подмножеством множества B, а B — *собственным* надмножеством множества A.

Замечание. Если требуется различать собственные и несобственные подмножества, то для обозначения включения собственных подмножеств используется знак $\subset$, а для несобственных — знак $\subseteq$.

Следствие. $A \subset B \subseteq C \Rightarrow A \subset C$.

1.2.2. Равномощные множества

Говорят, что между множествами A и B установлено взаимно-однозначное соответствие, если каждому элементу множества A поставлен в соответствие один и только один элемент множества B и для каждого элемента множества B один и только один элемент множества A поставлен в соответствие этому элементу множества B. В этом случае говорят также, что множества A и B *изоморфны*, и используют обозначение $A \sim B$. Если при заданном соответствии элементу $a \in A$ соответствует элемент $b \in B$, то данное обстоятельство обозначают следующим образом: $a \rightarrowtail b$.

Замечание. Весьма общее понятие соответствия в этом учебном пособии трактуется с программистских позиций. Если сказано, что между множествами A и B установлено соответствие, то подразумевается, что задан способ по любому элементу $a \in A$ определить соответствующий ему элемент $b \in B$. Способ может быть любым, если возможность его применения не вызывает сомнений. Например, это может быть массив **array** [A] **of** B, или процедура типа **proc** (A) : B, или же выражение, зависящее от переменной типа A и доставляющее значение типа B.

Пример. Соответствие $n \rightarrowtail 2n$ устанавливает взаимно-однозначное соответствие между множеством натуральных чисел $\mathbb{N}$ множеством чётных натуральных чисел $2\mathbb{N}$, $\mathbb{N} \sim 2\mathbb{N}$.

Нетрудно видеть, что:

1) любое множество взаимно-однозначно соответствует самому себе: $A \sim A$ — достаточно рассмотреть соответствие $a \rightarrowtail a$, где $a \in A$;

2) если $A \sim B$, то $B \sim A$ — достаточно использовать соответствие $a \rightarrowtail b$ для построения соответствия $b \rightarrowtail a$, где $a \in A$, $b \in B$;

3) если $A \sim B$ и $B \sim C$, то $A \sim C$ — соответствие устанавливается с использованием промежуточного элемента $b \in B$: $a \rightarrowtail b \rightarrowtail c$, где $a \in A$, $b \in B$, $c \in C$.

Если между двумя множествами A и B может быть установлено взаимно-однозначное соответствие, то говорят, что множества имеют одинаковую *мощность*, или что множества *равномощны*, и записывают это так: $|A| = |B|$. Другими словами,

$$|A| = |B| \mathbf{:=} A \sim B.$$

Из определения и отмеченных свойств взаимно-однозначного соответствия непосредственно следует

Теорема. *Равномощность множеств обладает следующими свойствами*:

1) $\forall A\ (|A| = |A|)$.

2) $\forall A, B\ (|A| = |B| \Rightarrow |B| = |A|)$.

3) $\forall A, B, C\ (|A| = |B| \ \&\ |B| = |C| \Rightarrow |A| = |C|)$.

Примеры.

1) Множество десятичных цифр равномощно множеству пальцев на руках человека (для подавляющего большинства людей), но не равномощно множеству пальцев на руках и на ногах.

2) Множество чётных натуральных чисел равномощно множеству всех натуральных чисел.

1.2.3. Конечные и бесконечные множества

Для приложений дискретной математики в программировании наибольшее значение имеют множества с конечным количеством элементов. Вопрос о том, чем конечное отличается от бесконечного, неподготовленного человека может поставить в тупик. Здесь мы рассматриваем применительно к множествам один из возможных ответов на этот вопрос. С античных времен известен принцип: "часть меньше

целого". Оказывается, этот принцип можно использовать в качестве характеристического свойства конечных множеств. Для бесконечных множеств этот принцип не имеет места.

Множество A называется *конечным*, если у него нет равномощного собственного подмножества:

$$\forall B\ ((B \subset A\ \&\ |B| = |A|) \Rightarrow (B = A)).$$

Для конечного множества A используется запись $|A| < \infty$. Все остальные множества называются *бесконечными*. Взяв отрицание условия конечности, получаем, что для бесконечного множества A

$$\exists B\ (B \subset A\ \&\ |B| = |A|\ \&\ B \neq A),$$

то есть бесконечное множество равномощно некоторому своему собственному подмножеству. Для бесконечного множества A используется запись $|A| = \infty$.

Пример. Множество натуральных чисел бесконечно, $|\mathbb{N}| = \infty$, поскольку оно равномощно своему собственному подмножеству чётных чисел.

Теорема. *Множество, имеющее бесконечное подмножество, бесконечно*:

$$(B \subseteq A\ \&\ |B| = \infty) \Rightarrow (|A| = \infty).$$

Доказательство. Множество B бесконечно, то есть существует взаимно-однозначное соответствие $B \sim C$ между множеством B и некоторым его собственным подмножеством C. Обозначим это соответствие $x \rightarrowtail x'$. Построим соответствие между множеством A и его собственным подмножеством D:

$$X \rightarrowtail \textbf{if } x \in B \textbf{ then } x' \textbf{ else } x \textbf{ end if}.$$

Другими словами, на элементах из B мы пользуемся заданным соответствием, а остальным элементам сопоставляем их самих (рис. 1.1). Это взаимно-однозначное соответствие между множеством A и его собственным подмножеством D, и значит, $|A| = \infty$. □

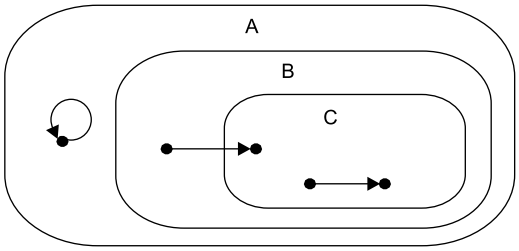


Рис. 1.1. К доказательству теоремы 1.2.3

Замечание. В обозначениях параграфа 1.2.6 $D = C \cup (A \backslash B)$.

Следствие. *Все подмножества конечного множества конечны*.

1.2.4. Добавление и удаление элементов

Если A — множество, а x — элемент, причём $x \notin A$, то элемент x можно *добавить* в множество A:

$$A + x := \{y \mid y \in A \vee y = x\}.$$

Аналогично, если A — множество, а x — элемент, причём $x \in A$, то элемент x можно *удалить* из множества A:

$$A - x := \{y \mid y \in A \,\&\, y \neq x\}.$$

Легко видеть, что при удалении и добавлении конечного числа элементов конечные множества остаются конечными, а бесконечные — бесконечными.

На самом деле операции добавления и удаления элементов являются частными случаями операций объединения и разности множеств, $A + x = A \cup \{x\}$, $A - x = A \backslash \{x\}$, которые рассматриваются в параграфе 1.2.6 и потому, строго говоря, излишни. В стандартных учебниках по теории множеств такие операции не рассматриваются и не

упоминаются. Здесь они введены для упрощения обозначений. Такой подход акцентирует "программистское" отношение к математике: мы вводим новые операции, если это практически удобно, даже если это теоретически излишне.

1.2.5. Мощность конечного множества

Теорема 1. *Любое непустое конечное множество равномощно некоторому отрезку натурального ряда*:

$$\forall A\ (A \neq \varnothing\ \&\ |A| < \infty \Rightarrow \exists k \in \mathbb{N}\ (|A| = |1..k|)).$$

Доказательство. Рассмотрим следующую программу:

$i := 0$ // счётчик элементов
while $A \neq \varnothing$ **do**
  select $x \in A$ // выбираем элемент
  $i := i + 1$ // увеличиваем счётчик
  $x \longmapsto i$ // ставим элементу в соответствие его номер
  $A := A - x$ // удаляем элемент из множества
end while

Если эта программа не заканчивает работу, то она даёт соответствие $B \sim \mathbb{N}$ для некоторого множества $B \subseteq A$, что невозможно ввиду конечности A. Значит, процедура заканчивает работу при $i = k$. Но в этом случае построено взаимно-однозначное соответствие $A \sim 1..k$. □

Лемма. *Любое непустое подмножество множества натуральных чисел содержит наименьший элемент.*

Доказательство. Пусть A — произвольное подмножество множества натуральных чисел, конечное или бесконечное. Рассмотрим задание множества A следующей порождающей процедурой:

$A := \{n \in \mathbb{N} \mid n := 0;$
while true do $n := n + 1$; **if** $n \in A$ **then yield** n **end if end while**$\}$.

Ясно, что множество A действительно порождается этой процедурой, причём тот элемент, который порождается первым, является наименьшим. □

Теорема 2. Любой отрезок натурального ряда конечен:

$$\forall n \in \mathbb{N}\ (|1..n| < \infty).$$

Доказательство. От противного. Пусть существуют бесконечные отрезки натурального ряда. Рассмотрим наименьшее n такое, что $|1..n| = \infty$. Тогда отрезок $1..n$ равномощен некоторому своему собственному подмножеству A, $|1..n| = |A|$, то есть существует взаимно-однозначное соответствие между отрезком $1..n$ и подмножеством A. Пусть при этом соответствии $n \rightarrowtail i$. Рассмотрим соответствие между отрезком $1..(n-1)$ и его собственным подмножеством $A - i$, задаваемое соответствием между $1..n$ и A. Это соответствие является взаимно-однозначным, а значит, отрезок $1..(n-1)$ изоморфен своему собственному подмножеству и является бесконечным, что противоречит выбору n. □

Следствие. *Различные отрезки натурального ряда неравномощны*:

$$n \neq m \Rightarrow |1..n| \neq |1..m|.$$

Доказательство. Пусть для определённости $n > m$. Тогда $1..m \subseteq 1..n$ и $1..m \neq 1..n$. Если $|1..n| = |1..m|$, то $|1..m| = \infty$, что противоречит теореме. □

Говорят, что конечное множество A имеет *мощность* k (обозначения: $|A| = k$, card $A = k$, $\#A = k$), если оно равномощно отрезку $1..k$:

$$|A| = k := A \sim 1..k.$$

Замечание. Таким образом, если множество A конечно, $|A| = k$, то элементы A всегда можно *перенумеровать*, то есть поставить им в соответствие номера из отрезка $1..k$ с помощью некоторой процедуры.

Наличие такой процедуры подразумевается, когда употребляется запись $A = \{a_1, \ldots, a_k\}$.

По определению $|\varnothing|$ **:=** 0.

1.2.6. Операции над множествами

Обычно рассматриваются следующие операции над множествами:

объединение: $A \cup B$ **:=** $\{x \mid x \in A \vee x \in B\}$;

пересечение: $A \cap B$ **:=** $\{x \mid x \in A \,\&\, x \in B\}$;

разность: $A \backslash B$ **:=** $\{x \mid x \in A \,\&\, x \notin B\}$;

симметрическая разность:

$$A \triangle B \text{ := } (A \cup B) \backslash (A \cap B) = \{x \mid (x \in A \,\&\, x \notin B) \vee (x \notin A \,\&\, x \in B)\};$$

дополнение: A' **:=** $\{x \mid x \notin A\}$.

Операция дополнения подразумевает, что задан некоторый универсум U: $A' = U \backslash A$. В противном случае операция дополнения не определена.

Пример. Пусть $A := \{1, 2, 3\}$, $B := \{3, 4, 5\}$. Тогда $A \cup B = \{1, 2, 3, 4, 5\}$, $A \cap B = \{3\}$, $A \backslash B = \{1, 2\}$, $A \triangle B = \{1, 2, 4, 5\}$. Если определён универсум $U := \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, то $A' = \{0, 4, 5, 6, 7, 8, 9\}$, $B' = \{0, 1, 2, 6, 7, 8, 9\}$.

Если множества A и B конечны, то из определений нетрудно видеть, что

$$|A \cup B| = |A| + |B| - |A \cap B|,$$
$$|A \backslash B| = |A| - |A \cap B|,$$
$$|A \triangle B| = |A| + |B| - 2|A \cap B|.$$

Операции пересечения и объединения двух множеств допускают следующее обобщение. Пусть задано семейство множеств $\{A_i\}_{i \in I}$. Тогда

$$\bigcup_{i \in I} A_i \text{ := } \{x \mid \exists i \in I\, (x \in A_i)\},$$

$$\bigcap_{i \in I} A_i := \{x \mid \forall i \in I\, (x \in A_i).$$

1.2.7. Разбиения и покрытия

Пусть $\mathcal{E} = \{E_i\}_{i \in I}$ — некоторое семейство подмножеств множества M, $E_i \subseteq M$. Семейство $\mathcal{E}$ называется *покрытием* множества M, если каждый элемент M принадлежит хотя бы одному из E_i:

$$\forall x \in M\, (\exists i \in I\, (x \in E_i)).$$

Семейство $\mathcal{E}$ называется *дизъюнктным*, если элементы этого семейства попарно не пересекаются, то есть каждый элемент множества M принадлежит не более чем одному из множеств E_i:

$$\forall i, j \in I\, (i \neq j \Rightarrow E_i \cap E_j = \varnothing).$$

Дизъюнктное покрытие называется *разбиением* множества M. Элементы разбиения, то есть подмножества множества M, часто называют *блоками* разбиения.

Пример. Пусть $M := \{1, 2, 3\}$. Тогда $\{\{1, 2\}, \{2, 3\}, \{3, 1\}\}$ является покрытием, но не разбиением; $\{\{1\}, \{2\}, \{3\}\}$ является разбиением (и покрытием), а семейство $\{\{1\}, \{2\}\}$ является дизъюнктным, но не является, ни покрытием, ни разбиением.

Теорема. *Если $\mathcal{E} = \{E_i\}_{i \in I}$ есть дизъюнктное семейство подмножеств множества M, то существует разбиение $\mathcal{B} = \{B_i\}_{i \in I}$ множества M такое, что каждый элемент дизъюнктного семейства $\mathcal{E}$ является подмножеством блока разбиения $\mathcal{B}$*:

$$\forall i \in I\, (E_i \subseteq B_i).$$

Доказательство. Выберем произвольный элемент $i0 \in I$ и построим семейство $\mathcal{B} = \{B_i\}_{i \in I}$ следующим образом:

$$B_{i0} = M \setminus \bigcup_{i \in I - i0} Ei, \qquad \forall i \in I - i0\, (B_i = E_i).$$

Семейство $\mathcal{B}$ по построению является дизъюнктным покрытием, то есть разбиением. Ясно, что $E_{i0} \subseteq B_{i0}$, а для остальных элементов требуемое включение имеет место по построению. □

Пример. Пусть $M := \{1, 2, 3\}$. Тогда элементы дизъюнктного семейства $\{\{1\}, \{2\}\}$ являются подмножествами блоков разбиения $\{\{1\}, \{2,3\}\}$.

1.2.8. Булеан

Множество всех подмножеств множества M называется *булеаном* множества M и обозначается 2^M:

$$2^M := \{A \mid A \subseteq M\}.$$

Теорема. *Если множество M конечно, то* $|2^M| = 2^{|M|}$.

Доказательство. Индукция по $|M|$. База: если $|M| = 0$, то $M = \varnothing$ и $2^{\{\varnothing\}} = \{\varnothing\}$. Следовательно, $|2^{\{\varnothing\}}| = |\{\varnothing\}| = 1 = 2^0 = 2^{|\varnothing|}$. Индукционный переход: пусть $\forall M\ (|M| < k \Rightarrow |2^M| = 2^{|M|})$. Рассмотрим $M = \{a_1, \ldots, a_k\}$, $|M| = k$. Положим $\mathcal{M}_1 := \{X \subseteq 2^M \mid a_k \notin X\}$ и $\mathcal{M}_2 := \{X \subseteq 2^M \mid a_k \in X\}$. Имеем $2^M = \mathcal{M}_1 \cup \mathcal{M}_2$, $\mathcal{M}_1 \cap \mathcal{M}_2 = \varnothing$, $\mathcal{M}_1 \sim \mathcal{M}_2$ за счёт взаимнооднозначного соответствия $X \rightarrowtail X + a_k$. Обозначим $L := \{a_1, \ldots, a_{k-1}\}$. В таких обозначениях $\mathcal{M}_1 \sim 2^L$. По индукционному предположению $|2^L| = 2^{k-1}$, и значит, $|\mathcal{M}_1| = |\mathcal{M}_2| = 2^{k-1}$. Следовательно, $|2^M| = |\mathcal{M}_1| + |\mathcal{M}_2| = 2^{k-1} + 2^{k-1} = 2^k = 2^{|M|}$. □

Пересечение, объединение и разность подмножеств множества U (универсума) являются его подмножествами, то есть применение операций к подмножествам универсума не выводит за его пределы.

Множество всех подмножеств множества U с операциями пересечения, объединения, разности и дополнения образует *алгебру подмножеств U*.

1.2.9. Свойства операций над множествами

Операции над множествами обладают целым рядом важных свойств, которые рассматриваются в этом параграфе. Пусть задан универсум U. Тогда $\forall\, A, B, C \subseteq U$ и выполняются следующие равенства:

1) *идемпотентность*:

$$A \cup A = A, \qquad A \cap A = A;$$

2) *коммутативность*:

$$A \cup B = B \cup A, \quad A \cap B = B \cap A;$$

3) *ассоциативность*:

$$A \cup (B \cup C) = (A \cup B) \cup C,\ A \cap (B \cap C) = (A \cap B) \cap C;$$

4) *дистрибутивность*}:

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C),\ A \cap (B \cup C) =$$
$$= (A \cap B) \cup (A \cap C);$$

5) *поглощение*:

$$(A \cap B) \cup A = A, \qquad (A \cup B) \cap A = A;$$

6) свойства нуля:

$$A \cup \varnothing = A, \qquad A \cap \varnothing = \varnothing;$$

7) свойства единицы:

$$A \cup U = U, \qquad A \cap U = A;$$

8) *инволютивность*:

$$(A')' = A$$

9) *законы де Моргана*:

$$(A \cap B)' = A' \cup B', \qquad (A \cup B)' = A' \cap B';$$

10) свойства дополнения:

$$A \cup A' = U, \qquad A \cap A' = \varnothing;$$

11) выражение для разности:

$$A \setminus B = A \cap B'.$$

В справедливости перечисленных равенств можно убедиться различными способами. Например, можно провести формальное рассуждение для каждого равенства. Рассмотрим для примера первое ра-

венство: $A \cup A = A$. Возьмем произвольный элемент x, принадлежащий левой части равенства, $x \in A \cup A$. По определению операции объединения $\cup$ имеем $x \in A \vee x \in A$. В любом случае $x \in A$. Взяв произвольный элемент из множества в левой части равенства, обнаруживаем, что он принадлежит множеству в правой части. Отсюда по определению включения множеств получаем, что $A \cup A \subseteq A$. Пусть теперь $x \in A$. Тогда, очевидно, верно $x \in A \vee x \in A$. Отсюда по определению операции объединения имеем $x \in A \cup A$. Таким образом, $A \subseteq A \cup A$. Следовательно, по определению равенства множеств, $A \cup A = A$. Это рассуждение можно записать короче:

$$x \in A \cup A \Leftrightarrow x \in A \vee x \in A \Leftrightarrow x \in A.$$

Аналогичные рассуждения нетрудно провести и для остальных равенств.

1.3. ПРЕДСТАВЛЕНИЕ МНОЖЕСТВ В ПРОГРАММАХ

Термин "представление" применительно к программированию означает следующее. Представить в программе какой-либо объект (в данном случае множество) — это значит описать в терминах системы программирования структуру данных, используемую для хранения информации о представляемом объекте, и алгоритмы над выбранными структурами данных, которые реализуют присущие данному объекту операции. Таким образом, применительно к множествам определение представления подразумевает описание способа хранения информации о принадлежности элементов множеству и описание алгоритмов для вычисления объединения, пересечения и других введённых операций.

Следует подчеркнуть, что, как правило, один и тот же объект может быть представлен многими разными способами, причём нельзя указать способ, который является наилучшим для всех возможных случаев. В одних случаях выгодно использовать одно представление, а в других — другое. Выбор представления зависит от целого ряда

факторов: особенностей представляемого объекта, состава и относительной частоты использования операций в конкретной задаче и т. д. Умение выбрать наилучшее для данного случая представление является основой искусства практического программирования. Хороший программист отличается тем, что он знает *много* разных способов представления и умело выбирает наиболее подходящий.

1.3.1. Битовые шкалы

Пусть задан конечный универсум U и число элементов в нем не превосходит разрядности компьютера, $|U| < n$.

Элементы универсума нумеруются:

$U = \{u_1, \ldots, u_n\}$

Подмножество A универсума U представляется *кодом* (машинным *словом* или *битовой шкалой*) C: **array** [1..n] of 0..1, в котором:

$C[i]$ = **if** $u_i \in A$ **then** 1 **else** 0

Замечание. Указанное представление можно описать короче, задав *инвариант* (условие, которому удовлетворяют все элементы): $\forall i \in 1..n$ ($C[i] = u_i \in A$). В дальнейшем предпочтение отдаётся более короткой форме записи описания представлений.

Код пересечения множеств A и B есть поразрядное логическое произведение кода множества A и кода множества B. Код объединения множеств A и B есть поразрядная логическая сумма кода множества A и кода множества B. Код дополнения множества A есть инверсия кода множества A. В большинстве компьютеров для этих операций есть соответствующие машинные команды. Таким образом, операции над небольшими множествами выполняются весьма эффективно. В некоторых языках программирования, например в Паскале, это представление множеств непосредственно включено в состав типов данных языка.

Замечание. Если мощность универсума превосходит размер машинного слова, но не очень велика, то для представления множеств

используются массивы битовых шкал. В этом случае операции над множествами реализуются с помощью циклов по элементам массива.

Эту же идею можно использовать для представления мультимножеств. Если $\underline{\underline{X}} = [x_1^{a1}, \ldots, x_n^{an}]$ — мультимножество над множеством $X = \{x_1, \ldots, x_n\}$, то массив A: **array** [1..n] **of integer**, где $\forall i \in 1..n\ (A[i] = a_i)$, является представлением мультимножества $\underline{\underline{X}}$.

Замечание. Представление индикатора совпадает с представлением его состава. Полезно сравнить это наблюдение с первым замечанием в параграфе 1.2.1.

1.3.2. Генерация всех подмножеств универсума

Во многих переборных алгоритмах требуется последовательно рассмотреть все подмножества заданного множества $\{a_1, \ldots, a_k\}$. В большинстве компьютеров целые числа представляются кодами в двоичной системе счисления, причём число $2^k - 1$ представляется кодом, содержащим k единиц, а все числа, меньшие $2^k - 1$, представляются кодами, имеющими не более k ненулевых разрядов. Таким образом, код числа 0 является представлением пустого множества $\varnothing$, код числа 1 является представлением подмножества, состоящего из первого элемента, и т. д., код числа $2^k - 1$ является представлением всего множества $\{a_1, \ldots, a_k\}$. Это наблюдение позволяет сформулировать следующий тривиальный алгоритм, который перечисляет все подмножества n-элементного множества.

Алгоритм 1.1. Генерация всех подмножеств n-элементного множества

Вход: $n \geq 0$ — мощность множества.

Выход: последовательность кодов подмножеств i.

For i **from** 0 **to** $2^n - 1$ **do**

yield I // код очередного подмножества

end for

Обоснование. Алгоритм выдает 2^n различных целых чисел, следовательно, 2^n различных кодов от 00...0 до 11...1. Таким образом, все подмножества генерируются, причём ровно по одному разу.

Недостаток этого алгоритма состоит в том, что порядок генерации подмножеств никак не связан с их составом. Например, вслед за подмножеством с кодом 0111 (состоящим из трёх элементов) будет перечислено подмножество с кодом 1000 (состоящее из одного элемента).

Замечание. Во многих переборных задачах требуется рассмотреть все подмножества некоторого множества и найти среди них то, которое удовлетворяет заданному условию. При этом проверка условия часто может быть весьма трудоёмкой и зависеть от состава элементов очередного рассматриваемого подмножества. В частности, если очередное рассматриваемое подмножество незначительно отличается по составу элементов от предыдущего, то иногда можно воспользоваться результатами оценки элементов, которые рассматривались на предыдущем шаге перебора. В таком случае, если перебирать подмножества в подходящем порядке, можно значительно ускорить работу переборного алгоритма.

1.3.3. Алгоритм построения бинарного кода Грея

Алгоритм 1.2 генерирует последовательность всех подмножеств n-элементного множества, причём каждое следующее подмножество получается из предыдущего удалением или добавлением в точности одного элемента.

Вход $n \geq 0$ — мощность множества.

Выход: последовательность кодов подмножеств B.

```
B: array [1..n] of 0..1   // битовая шкала
                          // для представления подмножеств
for i from 1 to n do
    B[i] := 0 // инициализация
end for
yield B // пустое множество
for i from 1 to 2^(n–1) do
    p := Q(i)     // определение номера элемента,
                  // подлежащего добавлению или удалению
```

$B[p] := 1 - B[p]$ // добавление или удаление элемента

yield B // очередное подмножество

end for

Функция Q, с помощью которой определяется номер разряда, подлежащего изменению, возвращает количество нулей на конце двоичной записи числа i\$, увеличенное на 1.

Алгоритм 1.3. Функция Q определения номера изменяемого разряда

Вход i — номер подмножества.

Выход: номер изменяемого разряда.

$q := 1; j := i$

while j чётно **do**

$j := j / 2; q := q + 1$

end while

return q

Обоснование. Для $n = 1$ искомая последовательность кодов есть 0, 1. Пусть известна искомая последовательность кодов B_1, …, B_k, $k = 2^n$, для некоторого $n > 1$. Тогда последовательность кодов $B_1 0$, ..., $B_k 0$, $B_k 1$, ..., $B_1 1$ будет искомой последовательностью для $n + 1$. Действительно, в последовательности $B_1 0$, ..., $B_k 0$, $B_k 1$, ..., $B_1 1$ имеется 2^{n+1} кодов, они все различны и соседние различаются ровно в одном разряде по построению. Именно такое построение и осуществляет данный алгоритм. На нулевом шаге алгоритм выдаёт правильное подмножество B (пустое). Пусть за первые $2^k - 1$ шагов алгоритм выдал последовательность значений B. При этом $B[k + 1] = B[k + 2] = ... = B[n] = 0$. На 2^k-м шаге разряд $B[k + 1]$ изменяет своё значение с 0 на 1. После этого будет повторена последовательность изменений значений $B[1..k]$ в обратном порядке, поскольку $Q(2^k + m) = Q(2^k - m)$ для $0 \le m \le 2^k - 1$.

1.3.4. Представление множеств упорядоченными списками

Если универсум очень велик (или бесконечен), а рассматриваемые подмножества универсума не очень велики, то представление с

помощью битовых шкал не является эффективным с точки зрения экономии памяти. В этом случае множества обычно представляются списками элементов. Элемент списка при этом представляется записью с двумя полями: информационным и указателем на следующий элемент. Весь список задаётся указателем на первый элемент.

```
elem = record
    i: info;      // информационное поле;
                  // тип info считается определённым
    n: ↑elem      // указатель на следующий элемент
end record
```

При таком представлении трудоёмкость операции $\in$ составит $O(n)$, а трудоёмкость операций $\subseteq$, $\cap$, $\cup$ составит $O(nm)$, где n и m — мощности участвующих в операции множеств.

Замечание. Используемый здесь и далее символ O в выражениях вида $f(n) = O(g(n))$ читается "О большое" и означает, что функция f асимптотически ограничена по сравнению с g (ещё говорят, что f "имеет тот же порядок малости", что и g):

$$f(n) = O(g(n)) = \exists n_0,\, c > 0\ (\forall n > n_0\,(f(n) \le c\, g(n))).$$

Если элементы в списках упорядочить, например, по возрастанию значения поля i, то трудоёмкость всех операций составит $O(n + m)$. Эффективная реализация операций над множествами, представленными в виде упорядоченных списков, основана на весьма общем алгоритме, известном как *алгоритм слияния*. Алгоритм слияния параллельно просматривает два множества, представленных упорядоченными списками, причём на каждом шаге продвижение происходит в том множестве, в котором текущий элемент меньше.

1.3.5. Проверка включения слиянием

Рассмотрим алгоритм слияния, который определяет, является ли множество A подмножеством множества B.

Алгоритм 1.4. Проверка включения слиянием.

Вход: проверяемые множества A и B, которые заданы указателями a и b.

Выход: true, если $A \subseteq B$, в противном случае **false**.

```
pa := a; pb := b // инициализация
while pa ≠ nil & pb ≠ nil do
    if pa.i < pb.i then
        return false      // элемент множества A
                          // отсутствует в множестве B
    elseif pa.i > pb.i then
        pb := pb.n        // элемент множества A, может быть,
                          // присутствует в множествеB
    else
        pa := pa.n        // здесь pa.i = pb.i, то есть
        pb := pb.n        // элемент множества A точно
                          // присутствует в множестве B
    end if
end while
return pa = nil // true, если A исчерпано, false — иначе
```

Обоснование. На каждом шаге основного цикла возможна одна из трёх ситуаций: текущий элемент множества A меньше, больше или равен текущему элементу множества B. В первом случае текущий элемент множества A заведомо меньше, чем текущий и все последующие элементы множества B, а потому он не содержится в множестве B и можно завершить выполнение алгоритма. Во втором случае происходит продвижение по множеству B в надежде отыскать элемент, совпадающий с текущим элементом множества A. В третьем случае найдены совпадающие элементы и происходит продвижение сразу в обоих множествах. По завершении основного цикла возможны два варианта: либо pa = **nil**, либо $pa \neq$ **nil**. Первый вариант означает, что для всех элементов множества A удалось найти совпадающие элементы в множестве B. Второй — что множество B закончилось

раньше, то есть не для всех элементов множества A удалось найти совпадающие элементы в множестве B.

1.3.6. Вычисление объединения слиянием

Рассмотрим алгоритм слияния, который вычисляет объединение двух множеств, представленных упорядоченными списками.

Алгоритм 1.5. Вычисление объединения слиянием.

Вход: объединяемые множества A и B, которые заданы указателями a и b.

Выход: объединение $C = A \cup B$, заданное указателем c.

```
pa := a; pb := B; c := nil; e := nil    // инициализация
while pa ≠ nil & pb ≠ nil do
    if pa.i < pb.i then
        d := pa.i; pa := pa.n      // добавлению подлежит
                                   // элемент множества A
    elseif pa.i > pb.i then
        d := pb.i; pb := pb.n      // добавлению подлежит
                                   // элемент множества B
    else
        d := pa.i                  // здесь pa.i = pb.i, и можно
                                   // взять любой из элементов
        pa := pa.n;          // продвижение
        pb := pb.n           // в обоих множествах
    end if
    Append(c, e, d) // добавление элемента d в конец списка c
end while
p := nil                             // указатель "хвоста"
if pa ≠ nil then
    p := pa       // нужно добавить в результат
                  //оставшиеся элементы множества A }
end if
if pb ≠ nil then
    p := pb       // нужно добавить в результат
```

// оставшиеся элементы множества *B*

end if

while *p* ≠ **nil do**

Append(*c*, *e*, *p.i*) // добавление *pa.i* в конец списка *c*

p := *p.n* // продвижение указателя "хвоста"

end while

Обоснование. На каждом шаге основного цикла возможна одна из трёх ситуаций: текущий элемент множества *A* меньше, больше или равен текущему элементу множества *B*. В первом случае в результирующий список добавляется текущий элемент множества *A* и происходит продвижение в этом множестве, во втором — аналогичная операция производится с множеством *B*, а в третьем случае найдены совпадающие элементы и происходит продвижение сразу в обоих множествах. Таким образом, в результат попадают все элементы обоих множеств, причём совпадающие элементы попадают ровно один раз. По завершении основного цикла один из указателей, *pa* и *pb* (но не оба вместе!), может быть не равен **nil**. В этом случае остаток соответствующего множества без проверки добавляется в результат. □

Вспомогательная процедура $\Append(*c*,*e*,*d*)$ присоединяет элемент *d* к хвосту *e* списка *c*.

Алгоритм. 1.6. Процедура Append — присоединение элемента в конец списка.

Вход: указатель *c* на первый элемент списка, указатель *e* на последний элемент списка, добавляемый элемент *d*.

Выход: указатель *c* на первый элемент списка, указатель *e* на последний элемент списка.

q := **new**(elem); *q.i* := *d*; *q.n* := **nil** // новый элемент списка

if *c* = **nil then**

c := *q* // пустой список

else

e.n := *q* // непустой список

end if

e := *q*

Обоснование. До первого вызова функции Append имеем: *c* = **nil** и *e* = **nil**. После первого вызова указатель *c* указывает на первый элемент списка, а указатель *e* — на последний элемент (который после первого вызова является тем же самым элементом). Если указатели *c* и *e* не пусты и указывают на первый и последний элементы правильного списка, то после очередного вызова функции Append это свойство сохраняется. Из текста основной программы видно, что, кроме инициализации, все остальные манипуляции с этими указателями выполняются только с помощью функции Append. □

1.3.7. Вычисление пересечения слиянием

Рассмотрим алгоритм слияния, который вычисляет пересечение двух множеств, представленных упорядоченными списками.

Алгоритм 1.7. Вычисление пересечения слиянием.

Вход: пересекаемые множества *A* и *B*, заданные указателями *a* и *b*.

Выход: пересечение $C = A \cap B$, заданное указателем *c*.

```
pa := a; pb := b; c := nil; e := nil // инициализация
while pa ≠ nil & pb ≠ nil do
    if pa.i < pb.i then
        pa := pa.n  // элемент множества A
                    // не принадлежит пересечению
    else if pa.i > pb.i then
        pb := pb.n  // элемент множества B
                    // не принадлежит пересечению
    else
// здесь pa.i = pb.i — данный элемент принадлежит пересечению
        Append(c, e, pa.i);
        pa := pa.n; pb := pb.n // добавление элемента
    end if
end while
```

Обоснование. На каждом шаге основного цикла возможна одна из трёх ситуаций: текущий элемент множества A меньше, больше или равен текущему элементу множества B. В первом случае текущий элемент множества A не принадлежит пересечению, он пропускается и происходит продвижение в этом множестве, во втором то же самое производится с множеством B. В третьем случае найдены совпадающие элементы, один экземпляр элемента добавляется в результат и происходит продвижение сразу в обоих множествах. Таким образом, в результат попадают все совпадающие элементы обоих множеств, причём ровно один раз. □

1.3.8. Представление множеств итераторами

В предыдущих параграфах рассмотрены представления, в которых структуры данных предполагаются известными и используются при программировании операций. Иногда это бывает неудобно, поскольку, во-первых, возникают затруднения при совместном использовании нескольких альтернативных представлений для одного типа объектов, и, во-вторых, необходимо заранее определять набор операций, которым открыт доступ к внутренней структуре данных объектов. (Такие операции часто называют *первичными*).

Применительно к множествам следует признать, что понятие принадлежности является первичным, и потому вряд ли может быть запрограммировано без использования структуры данных для хранения множеств, но остальные операции над множествами нельзя признать первичными, и их можно запрограммировать независимо от представления множеств. Для иллюстрации этой идеи рассмотрим один из возможных способов реализации операций над множествами, не зависящий от представления множеств. Видимо, можно считать, что с программистской точки зрения для представления множества достаточно *итератора* — процедуры, которая перебирает элементы множества и делает с ними то, что нужно. Синтаксически понятие итератора может быть реализовано самыми различными способами в зависимости от традиций, стиля и используемого языка программиро-

вания. Например, следующая программа выполняет процедуру S для всех элементов множества X:

while $X \neq \varnothing$ **do**

select $x \in X$ //выбор произвольного элемента из множества X

$S(x)$ // применение процедуры S к элементу x

$X := X - x$ // удаление элемента x с целью

// исключить его повторный выбор

end while

В этом варианте реализации итератора исходное множество "исчезает", что не всегда желательно. Очевидно, что для всякого конкретного представления множества можно придумать такую реализацию итератора, которая обеспечивает перебор элементов ровно по одному разу и не портит исходного множества. Мы полагаем (здесь и далее во всей книге), что наличие итератора позволяет написать цикл

for $x \in X$ **do**

$S(x)$ // где S — любой оператор или процедура,

// зависящие от x }

end for

и этот цикл означает применение оператора S ко всем элементам множества x по одному разу в некотором неопределённом порядке.

Замечание. В доказательствах цикл **for** $x \in X$ **do** $S(x)$ **end for** используется и в том случае, $|X| = \infty$. Это не означает реального процесса исполнения тела цикла в дискретные моменты времени, а означает всего лишь мысленную возможность применить оператор S ко всем элементам множества x независимо от его мощности.

В таких обозначениях итератор пересечения множеств $X \cap Y$ может быть задан алгоритмом 1.8.

Алгоритм 1.8. Итератор пересечения множеств.

for $x \in X$ **do**

for $y \in Y$ **do**

if $x = y$ **then**

$S(x)$ // тело цикла

next for x // следующий x

end if

end for

end for

Аналогично итератор разности множеств $X\backslash Y$ может быть задан алгоритмом 1.9.

Алгоритм 1.9. Итератор разности множеств.

for $x \in X$ **do**

for $y \in Y$ **do**

if $x = y$ **then**

next for x // следующий x

end if

end for

$S(x)$ // тело цикла

end for

Замечание. Оператор **next for** x — это оператор *структурного перехода*, выполнение которого в данном случае означает прерывание выполнения цикла по y и переход к следующему шагу в цикле по x.

Заметим, что $A \cup B = (A\backslash B) \cup B$ и $(A\backslash B) \cap B = \varnothing$. Поэтому достаточно рассмотреть итератор объединения дизъюнктных множеств $X \cup Y$ (алгоритм 1.10), где $X \cap Y = \varnothing$.

Алгоритм 1.10. Итератор объединения дизъюнктных множеств.

for $x \in X$ **do**

$S(x)$ // тело цикла

end for

for $y \in Y$ **do**

$S(y)$ // тело цикла

end for

Стоит обратить внимание на то, что итератор пересечения реализуется вложенными циклами, а итератор дизъюнктного объединения — последовательностью циклов.

Нетрудно видеть, что сложность алгоритмов для операций с множествами при использовании предложенного представления, не зависящего от реализации, составляет в худшем случае $O(nm)$, где m и n — мощности участвующих множеств, в то время как для представления упорядоченными списками сложность в худшем случае равна $O(n + m)$.

Замечание. На практике почти всегда оказывается так, что самая изощрённая реализация операции, не зависящая от представления, оказывается менее эффективной по сравнению с самой прямолинейной реализацией, ориентированной на конкретное представление. При современном состоянии аппаратной базы вычислительной техники оказывается, что во многих задачах можно пренебречь некоторым выигрышем в эффективности, например, различием между $O(nm)$ и $O(n + m)$. Другими словами, нынешние компьютеры настолько быстры, что часто можно не гнаться за самым эффективным представлением, ограничившись "достаточно эффективным".

1.4. ОТНОШЕНИЯ

Обычное широко используемое понятие "отношение" имеет вполне точное математическое значение, которое вводится в этом разделе.

1.4.1. Упорядоченные пары и наборы

Если a и b — объекты, то через (a, b) *обозначим упорядоченную пару*. *Равенство* упорядоченных пар определяется следующим образом:

$(a, b) = (c, d)$ **:=** $a = c$ & $b = d$.

Вообще говоря, $(a, b) \neq (b, a)$.

Замечание. Формально упорядоченные пары можно определить как множества, если определить их так: (a, b) **:=** $\{a,\{a, b\}\}$. Таким образом, введённое понятие упорядоченной пары не выводит рассмотрение за пределы теории множеств.

Аналогично упорядоченным парам можно рассматривать упорядоченные тройки, четвёрки и т. д. В общем случае подобные объекты называются *n*-ками, *кортежами*, *наборами* или (конечными) *последовательностями*. Упорядоченный набор из n элементов обозначается $(a_1, \ldots, a_n)$. Набор $(a_1, \ldots, a_n)$ можно определить рекурсивно, используя понятие упорядоченной пары:

$(a_1, \ldots, a_n) = ((a_1, \ldots, a_{n-1}), a_n)$.

Количество элементов в наборе называется его *длиной* и обозначается следующим образом: $|(a_1, \ldots, a_n)|$.

Теорема. *Два набора одной длины равны, если равны их соответствующие элементы*:

$$\forall n \left((a_1, \ldots, a_n) = (b_1, \ldots, b_n) \Leftrightarrow \forall i \in 1..n\ (a_i = b_i)\right)$$

Доказательство. Индукция по n. База: при $n = 2$ по определению равенства упорядоченных пар. Пусть теперь

$$(a_1, \ldots, a_{n-1}) = (b_1, \ldots, b_{n-1}) \Leftrightarrow \forall i \in 1..(n-1)\ (a_i = b_i).$$

Тогда

$(a_1, \ldots, a_{n-1}) = (b_1, \ldots, b_{n-1}) \Leftrightarrow ((a_1, \ldots, a_{n-1}), a_n) =$

$= ((b_1, \ldots, b_{n-1})\ b_n) \Leftrightarrow (a_1, \ldots, a_{n-1}) =$

$= (b_1, \ldots, b_{n-1})\ \&\ a_n = b_n \Leftrightarrow \forall i \in 1..n\ (a_i = b_i)$. □

Отсюда следует, что порядок "отщепления" элементов в рекурсивном определении кортежа не имеет значения.

Замечание. Наиболее естественным представлением в программе *n*-ки с элементами из множества A является массив **array**[1..*n*] **of** A.

1.4.2. Прямое произведение множеств

Пусть A и B — два множества. *Прямым* (*декартовым*) *произведением* двух множеств A и B называется множество всех упорядоченных пар, в которых первый элемент принадлежит A, а второй принадлежит B:

$$A \times B := \{(a, b) \mid a \in A\ \&\ b \in B\}.$$

Замечание. Точка на плоскости может быть задана упорядоченной парой координат, то есть двумя точками на координатных осях. Таким образом, $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$. Своим появлением метод координат обязан Декарту, отсюда и название "декартово произведение".

Теорема. *Для конечных множеств A и B* $|A \times B| = |A|\,|B|$.

Доказательство. Первый компонент упорядоченной пары можно выбрать $|A|$ способами, второй — $|B|$ способами, причём независимо от выбора первого элемента. Таким образом, всего имеется $|A|\,|B|$ различных упорядоченных пар. □

Лемма. $(A \times B) \times C \sim A \times (B \times C)$.

Доказательство. $(A \times B) \times C = \{((a,\ b),\ c) \mid (a,\ b) \in A \times B\ \&\ c \in C\} =$
$= \{(a,\ b,\ c) \mid a \in A\ \&\ B \in B\ \&\ c \in C\} \sim \{(a,\ (b,c))\ \ a \in A\ \&\ (b,\ c) \in B \times C\} =$
$= A \times (B \times C)$, где требуемый изоморфизм устанавливается соответствием $(a, b, c) \rightarrowtail (a, (b, c))$. □

Понятие прямого произведения допускает обобщение. Прямое произведение множеств $A_1, \ldots, A_n$ — это множество наборов (*кортежей*):

$$A_1 \times \ldots \times A_n := \{(a_1, \ldots, a_n) \mid a_1 \in A_1\ \&\ \ldots\ \&\ a_n \in A_n\}.$$

Множества A_i не обязательно различны.

Степенью множества A называется его n-кратное прямое произведение самого на себя. Обозначение: A^n. Соответственно, $A^1 := A$, $A^2 := A \times A$ и вообще $A^n := A \times A^{n-1}$.

Следствие. $|A^n| = |A|^n$.

1.4.3. Бинарные отношения

Пусть A и B — два множества. *Бинарным отношением* между множествами A и B называется тройка $\langle A, B, R\rangle$, где R — подмножество прямого произведения A и B:

$$R \subseteq A \times B.$$

При этом R называется *графиком* отношения, A называется *областью отправления*, а B — *областью прибытия* отношения. Если множества A и B определены контекстом, то часто просто говорят, что задано отношение R. При этом для краткости отношение обозначают тем же символом, что и график.

Замечание. Принято говорить "отношение между множествами", хотя порядок множеств имеет значение, и отношение между A и B совсем не то же самое, что отношение между B и A. Поэтому иногда, чтобы подчеркнуть это обстоятельство, употребляют оборот "отношение R из множества A в множество B".

Среди элементов множеств A и B могут быть такие, которые не входят ни в одну из пар, определяющих отношение R. Множество элементов области отправления, которые присутствуют в парах отношения, называется *областью определения* отношения R и обозначается $\mathrm{Dom}\ R$, а множество элементов области прибытия, которые присутствуют в парах отношения, называется *областью значений* отношения R и обозначается $\backslash\mathrm{Im}\ R$:

$$\mathrm{Dom}\ R := \{a \in A \mid \exists b \in B\ ((a, b) \in R)\},$$

$$\mathrm{Im}\ R := \{b \in B \mid \exists a \in A\ ((a, b) \in R)\}.$$

Если $A = B$ (то есть $R \subseteq A^2$), то говорят, что R есть отношение *на* множестве A. Для бинарных отношений обычно используется *инфиксная* форма записи:

$$aRb := (a, b) \in R \subseteq A \times B.$$

Инфиксная форма позволяет более кратко записывать некоторые формы утверждений относительно отношений:

$$aRbRc := (a, b) \in R\ \&\ (b, c) \in R.$$

Примеры. Пусть задано множество U. Тогда $\in$ (принадлежность) — отношение между элементами множества U и элементами

булеана 2^U, а $\subseteq$ (включение) и = (равенство) — отношения на 2^U. Хорошо известны отношения =, <, ≤, >, ≥, ≠, определённые на множестве вещественных чисел.

Пусть R есть отношение между A и B: $R \subseteq A \times B$. Введём следующие понятия:

обратное отношение: $R^{-1} := \{(a, b) \mid (b, a) \in R\} \subseteq B \times A$,

дополнение отношения: $R' := \{(a, b) \mid (a, b) \notin R\} \subseteq A \times B$,

тождественное отношение: $I := \{(a, a) \mid a \in A\} \subseteq A^2$,

универсальное отношение: $U := \{(a, b) \mid a \in A \ \& \ b \in B\} = A \times B$.

Замечание. График тождественного отношения на множестве A иногда называют *диагональю* в $A \times A$.

Введем обобщённое понятие отношения: n-местное отношение R — это подмножество прямого произведения n множеств, то есть множество упорядоченных наборов (*кортежей*):

$$R \subseteq A_1 \times ... \times A_n = \{(a_1, \ldots, a_n) \mid a_1 \in A_1 \ \& \ ... \ \& \ a_n \in A_n\}.$$

Замечание. Число n, то есть длину кортежей отношения, называют иногда *вместимостью*.

Многоместные отношения используются, например, в теории баз данных. Само название "реляционная" база данных происходит от слова relation (отношение). Далее рассматриваются только двуместные (бинарные) отношения, при этом слово "бинарные" опускается.

1.4.4. Композиция отношений

Пусть $R_1 \subseteq A \times C$ — отношение между A и C, а $R_2 \subseteq C \times B$ — отношение между C и B. *Композицией* двух отношений R_1 и R_2 называется отношение $R \subseteq A \times B$ между A и B, определяемое следующим образом:

$$R := R_1 \circ R_2 := \{(a, b) \mid a \in A \ \& \ b \in B \ \& \ \exists c \in C \ (aR_1c \ \& \ cR_2b)\}.$$

Другими словами,

$$a \ R_1 \circ R_2 \ b \Leftrightarrow \exists c \in C \ (aR_1c \ \& \ cR_2b).$$

Теорема. Композиция отношений ассоциативна, то есть

$$\forall\, R_1 \subseteq A\times B,\, R_2 \subseteq B\times C,\, R_3 \subseteq C\times \mathrm{D}\ ((R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)).$$

Доказательство.

$a\ (R_1 \circ R_2)\circ R_3\ d \Leftrightarrow \exists c \in C\ (aR_1 \circ R_2\, c\ \&\ cR_3 d \Leftrightarrow$

$\Leftrightarrow \exists c \in C\ ((\exists b \in B\ (aR_1 b\ \&\ bR_2 c)\ \&\ cR_3 d) \Leftrightarrow$

$\Leftrightarrow \exists b\ \in B, c \in C\ (aR_1 b\ \&\ bR_2 c\ \&\ cR_3 d) \Leftrightarrow$

$\Leftrightarrow \exists b\ \in B\ (aR_1 b\ \&\ (\exists c \in C\ (bR_2 c\ \&\ cR_3 d))) \Leftrightarrow$

$\Leftrightarrow \exists b\ \in B\ (aR_1 b\ \&\ bR_2 \circ R_3 d) \Leftrightarrow aR_1 \circ (R_2 \circ R_3) d.$ □

Композиция отношений на множестве A является отношением на множестве A.

Пример. Композиция отношений $<$ и $\le$ на множестве вещественных чисел совпадает с отношением $<$: $<\circ\le\ =\ <$.

Замечание. В общем случае композиция отношений не коммутативна, то есть $R_1 \circ R_2 \neq R_2 \circ R_1$.

Пример. На множестве людей определены отношения кровного родства и супружества. Отношения "кровные родственники супругов" и "супруги кровных родственников" различны.

1.4.5. Степень отношения

Пусть R — отношение на множестве A. *Степенью* отношения R на множестве A называется его n-кратная композиция с самим собой. Обозначение: R^n. Соответственно, $R^0 = I$, $R^1 = R$, $R^2 = R \circ R$ и вообще $R^n = R^{\mathrm{n}-1} \circ R$.

Теорема. *Если некоторая пара* (a, b) *принадлежит какой-либо степени отношения* R *на множестве* A *мощности* n, *то эта пара принадлежит и некоторой степени* R *не выше* $n - 1$:

$$R \subseteq A^2\ \&\ |A| = n \Rightarrow \forall a, b \in A\ (\exists k\ aR^k b \Rightarrow \exists k < n\ aR^k b)).$$

Доказательство. Обозначим $c_0 := a$, $c_k := b$. По определению

$$(a, b) \in R^k \Rightarrow \exists\, c_1, \ldots, c_{k-1} \in A\ (c_0 R c_1 R c_2 R \ldots R c_{k-1} R c_k).$$

Далее, если $|A| = n$ & $k \geq n$, то $\exists i, j\ (c_i = c_j$ & $i \neq j)$, и значит,

$$c_0 R c_1 R \ldots R c_i R c_{j+1} R \ldots R c_{k-1} R c_k,$$

то есть $(a, b) \in R^{k-(j-i)}$. Обозначим через d процедуру, которая вычисляет пару чисел (i, j). Существование требуемого числа k (степени отношения R) обеспечивается следующим построением:

while $k \geq n$ **do**

$c_0 := a$; $ck := b$

$(i, j) := d()$

$k := k - (j - i)$

end while

□

Следствие. $R \subseteq A^2$ & $|A| = n \Rightarrow R^1 \cup \ldots \cup R^{n-1} =$
$= R^1 \cup \ldots \cup R^{n-1} \cup R^n \cup \ldots$.

1.4.6. Свойства отношений

Пусть $R \subseteq A^2$. Тогда отношение R называется:

рефлексивным, если $\forall a \in A\ (aRa)$;

антирефлексивным, если $\forall a \in A\ (\neg aRa)$;

симметричным, если $\forall a, b \in A\ (aRb \Rightarrow bRa)$;

антисимметричным, если $\forall a, b \in A\ (aRb$ & $bRa \Rightarrow a = b)$;

транзитивным, если $\forall a, b, c \in A\ (aRb$ & $bRc \Rightarrow aRc)$;

линейным, если $\forall a, b \in A\ (a = b \vee aRb \vee bRa)$.

Замечание. Иногда линейное отношение R называют *полным*. Такой термин является оправданным, поскольку для любых двух различных элементов A и B либо пара (a, b), либо пара (b, a) принадлежит отношению R. Отношение, не обладающее свойством полноты, при этом вполне естественно называть частичным. Однако использование термина "полное отношение" может порождать терминологиче-

скую путаницу. Дело в том, что устойчивое словосочетание "вполне упорядоченное множество", рассматриваемое в параграфе 1.8.6, оказывается созвучным словосочетанию "множество с полным порядком", хотя эти словосочетания обозначают существенно различные объекты. Во избежание неоднозначности мы используем термин "линейное отношение" как антоним термина "частичное отношение".

Теорема. Пусть $R \subseteq A \times A$ — отношение на A. Тогда:

1) R рефлексивно $\Leftrightarrow I \subseteq R$;

2) R симметрично $\Leftrightarrow R = R^{-1}$;

3) R транзитивно $\Leftrightarrow R \circ R \subseteq R$;

4) R антисимметрично $\Leftrightarrow R \cap R^{-1} \subseteq I$;

5) R антирефлексивно $\Leftrightarrow R \cap I = \varnothing$;

6) R линейно $\Leftrightarrow R \cup I \cup R^{-1} = U$.

Доказательство.

Используя определения свойств отношений, имеем:

[1.$\Leftrightarrow$] $\forall a \in A\ (aRa) \Leftrightarrow \forall a \in A\ ((a, a) \in R) \Leftrightarrow I \subseteq R$.

[2.$\Leftrightarrow$] $\forall a, b \in A\ (aRb \Rightarrow bRa) \Leftrightarrow \forall a, b \in A\ ((a, b) \in R \Rightarrow (b, a) \in R) \Leftrightarrow \forall a, b \in A\ (((a, b) \in R \Rightarrow (b, a) \in R)\ \&\ ((b, a) \in R \Rightarrow (a, b) \in R)) \Leftrightarrow \forall a, b \in A\ (((a, b) \in R \Rightarrow (a, b) \in R^{-1})\ \&\ ((a, b) \in R^{-1} \Rightarrow (a, b) \in R)) \Leftrightarrow R \subseteq R^{-1}\ \&\ R^{-1} \subseteq R \Leftrightarrow R = R^{-1}$.

[3.$\Leftrightarrow$] $\forall a, b, c \in A\ (aRb\ \&\ bRc \Rightarrow aRc) \Leftrightarrow$

$\Leftrightarrow \forall a, b, c \in A\ ((a, b) \in R\ \&\ (b, c) \in R \Rightarrow (a, c) \in R) \Leftrightarrow$

$\Leftrightarrow \forall a, c \in A\ (\exists b \in B\ ((a, b) \in R\ \&\ (b, c) \in R) \Rightarrow (a, c) \in R) \Leftrightarrow$

$\Leftrightarrow \forall a, c \in A\ ((a, c) \in R \circ R \Rightarrow (a, c) \in R\} \Leftrightarrow$

$\Leftrightarrow \forall a, c \in A\ (a\ R \circ R\ c \Rightarrow aRc) \Leftrightarrow R \circ R \subseteq R$.

[4.$\Rightarrow$] От противного. $R \cap R^{-1} \neq I \Rightarrow \exists a \neq b\ (aRb\ \&\ aR^{-1}b) \Rightarrow$

$\Rightarrow \exists a \neq Bb\ (aRb\ \&\ bRa)$

[4.$\Leftarrow$] $R \cap R^{-1} \subseteq I \Rightarrow (aRb\ \&\ aR^{-1}b \Rightarrow a = b) \Rightarrow (aRb\ \&\ \Rightarrow a = b)$.

[5.$\Rightarrow$] От противного. $R \cap I \neq \varnothing \Rightarrow \exists a \in A\ (aRa\ \&\ aIa) \Leftrightarrow$

$\Leftrightarrow \exists a \in A\ (aRa) \Leftrightarrow \neg \forall a \in A\ (\neg aRa)$.

[5.$\Leftarrow$] $R \cap I = \varnothing \Rightarrow \neg\exists a \in A\,(aRa) \Rightarrow \forall a \in A\,(\neg aRa)$.

[6.$\Leftrightarrow$] $\forall a, b \in A\,(a = b \vee aRb \vee bRa) \Leftrightarrow$

$\Leftrightarrow \forall a, b \in A\,((a, b) \in I \vee (a, b) \in R \vee (a, b) \in R^{-1}) \Leftrightarrow$

$\Leftrightarrow U \subseteq I \cup R \cup R^{-1} \Leftrightarrow U = R \cup I \cup R^{-1}$. □

1.4.7. Ядро отношения

Если $R \subseteq A \times B$ — отношение между множествами A и B, то композиция $R \circ R^{-1}$ называется *ядром* отношения R и обозначается $\ker R$. Другими словами, $a_1 \ker R\ a_2 \Leftrightarrow \exists\, b \in B\,(a_1 R b\ \&\ a_2 R b)$. Ядро отношения R между A и B является отношением на A: $R \subseteq A \times B \Rightarrow \ker R \subseteq A^2$.

Примеры.

1) Пусть отношение N_1 "находиться на расстоянии не более 1" на множестве целых чисел определено следующим образом:

$$N_1 := \{(n, m) \mid |n - m| \le 1).$$

Тогда ядром этого отношения является отношение N_2 "находиться на расстоянии не более 2":

$$N_2 := \{(n, m) \mid |n - m| \le 2).$$

2) Ядром отношения "быть знакомыми" является отношение "быть знакомыми или иметь общего знакомого".

3) Ядро отношения $\in$ между U и 2^U универсально:

$$\forall a, b \in U\,(\{a, b\} \in 2^U).$$

4) Ядро отношения $\subseteq$ на 2^U также универсально.

5) Рассмотрим отношение "тесного включения" на 2^U:

$$X \subseteq^{+1} Y := X \subseteq Y\ \&\ |X| + 1 = |Y|.$$

Ядром отношения $\subseteq^{+1}$ является отношение "отличаться не более чем одним элементом":

$$\ker \subseteq^{+1} = \{X, Y \in 2^U \mid |X| = |Y|\ \&\ |X \cap| \ge |X| - 1).$$

Замечание. Термин "ядро" и обозначение ker используются также и для других объектов. Их не следует путать — из контекста обычно ясно, в каком смысле используется термин "ядро".

Теорема. *Ядро любого отношения рефлексивно и симметрично на области определения*:

$$\forall R \subseteq A\times B(\forall a \in \mathrm{Dom}\, R(a \ker R\, a)\ \&\ \forall a, b \in \mathrm{Dom} R(a \ker R\, b \Rightarrow \Rightarrow b \ker R\, a)).$$

Доказательство.

[Рефлексивность] Пусть $a \in \mathrm{Dom}\, R$. Тогда $\exists b \in B\ (aRb) \Rightarrow$

$\Rightarrow \exists b \in B\ (aRb\ \&\ bR^{-1}a) \Rightarrow (a, a) \in R\circ R^{-1} \Rightarrow a \ker R\, a$.

[Симметричность] Пусть $a, b \in \mathrm{Dom}\, R\ \&\ a \ker R\, B$.

Тогда $\exists c \in B\ (aRc\ \&\ cR^{-1}b) \Rightarrow \exists c \in B(bRc\ \&\ cR^{-1}a) \Rightarrow$

$\Rightarrow (b, a) \in R\circ R^{-1} \Rightarrow b \ker R\, a$. □

1.5. ПРЕДСТАВЛЕНИЕ ОТНОШЕНИЙ В ПРОГРАММАХ

Отношения являются множествами, точнее, подмножествами декартовых произведений других множеств. Поэтому, вообще говоря, для представления отношений можно использовать те же приёмы, что и для представления множеств. Однако использование специальных представлений для отношений, в частности, булевых матриц, позволяет получать в некоторых случаях более эффективные представления.

1.5.1. Представление бинарных отношений булевыми матрицами

Пусть R — бинарное отношение на A, $R \subseteq A^2$ и $|A| = n$. Перенумеруем элементы множества A, $A = \{a_1, \ldots, a_n\}$. Тогда отношение R можно представить *булевой* матрицей R: **array** [1..n, 1..n] **of** 0..1, где $\forall i, j \in 1..n\ (R[i, j] = a_i R a_j)$.

Замечание. Термин "булева матрица" означает, что элементы матрицы — булевские значения и операции над ними выполняются по соответствующим правилам. В частности, если A и B — булевы матрицы, то операции над ними определяются следующим образом:

транспонирование: $A^T[i,j] := A[j, i]$;

вычитание: $(A - B)[i,j] := A[i,j] \,\&\, (1 - B[i,j])$;

инвертирование: $A'[i,j] := 1 - A[i,j]$;

умножение: $(A \times B)[i,j] := (A[i,1] \,\&\, B[1,j]) \vee \ldots \vee (A[i,n] \,\&\, B[n,j])$;

дизъюнкция: $(A \vee B)[i,j] := A[i,j] \vee B[i,j]$.

В следующих утверждениях предполагается, что все рассматриваемые отношения определены на множестве A, причём $|A| = n$.

Теорема 1. $R^{-1} = \mathrm{R}^T$.

Доказательство. $(b, a) \in R^{-1} \Leftrightarrow (a, b) \in R \Leftrightarrow$

$\Leftrightarrow R[a, b] = 1 \Leftrightarrow R^T[b, a] = 1$.□

В универсальное отношение U входят все пары элементов, поэтому все элементы матрицы универсального отношения U равны 1.

Теорема 2. $R' = U - R$.

Доказательство. $(a, b) \in R' \Leftrightarrow (a, b) \notin R \Leftrightarrow R[a, b] = 0 \Leftrightarrow$

$\Leftrightarrow U[a, b] - R[a, b] = 1 \Leftrightarrow (U - R)[a, b] = 1$. □

Следствие. $R' = R'$.

Теорема 3. $R_1 \circ R_2 = R_1 \times R_2$.

Доказательство. $(a, b) \in R_1 \circ R_2 \Leftrightarrow \exists c \in A\ (aR_1c \,\&\, cR_2b) \Leftrightarrow$

$\Leftrightarrow \exists c \in A\ (R_1[a, c] = 1 \,\&\, R_2[c, b] = 1) \Leftrightarrow$

$\Leftrightarrow (R_1[a, 1] \,\&\, R_2[1, b]) \vee \ldots \vee (R_1[a, n] \,\&\, R_2[n, b]) = 1 \Leftrightarrow$

$\Leftrightarrow (R_1 \times R_2)[a, b] = 1$. □

Следствие. $R^k = R^k$.

Теорема 4. $R_1 \cup R_2 = R_1 \vee R_2$.

Доказательство. $(a, b) \in R_1 \cup R_2 \Leftrightarrow aR_1b \vee aR_2b \Leftrightarrow$

$R_1\ [a, b] = 1 \vee R_2\ [a, b] = 1 \Leftrightarrow (R_1 \vee R_2)[a, b] = 1.$ □

Замечание. Представление отношений с помощью булевых матриц — это только один из возможных способов представления отношений. Другие варианты рассматриваются при обсуждении представления графов во второй части книги.

1.5.2. Замыкания отношений

Замыкание является весьма общим математическим понятием, рассмотрение конкретных вариантов которого начинается в этом параграфе и продолжается в других главах книги. Неформально говоря, замкнутость означает, что многократное выполнение допустимых шагов не выводит за определённые границы. Например, предел сходящейся последовательности чисел из *замкнутого* интервала [*a, b*] обязательно принадлежит этому интервалу, а предел сходящейся последовательности чисел из открытого (то есть *не* замкнутого) интервала (*a, b*) может лежать вне этого интервала. В некоторых случаях можно "расширить" незамкнутый объект так, чтобы он оказался замкнутым.

Пусть R и S — отношения на множестве M. Отношение S называется *замыканием* отношения R относительно свойства C, если:

1) S обладает свойством C: $C(S)$;

2) S является надмножеством R: $R \subseteq S$;

3) S является наименьшим таким объектом: $C(T)$ & $R \subseteq T \Rightarrow S \subseteq T$.

Пусть R^+ — объединение положительных степеней R, а R^* — объединение неотрицательных степеней R.

Теорема. R^+ — *транзитивное замыкание* R.

Доказательство. Проверим выполнение свойств замыкания при условии, что свойство C — это транзитивность.

[$C(R^+)$] Пусть aR^+b & bR^+c. Тогда $\exists n\ (aR^nb)$ & $\exists m\ (bR^mc)$. Следовательно, $\exists\ c_1, \ldots, c_{n-1}\ (aRc_1R...Rc_{n-1}Rb)$ и $\exists\ d_1, \ldots, d_{m-1}\ (bRd_1R...Rd_{m-1}Rc)$. Таким образом, $aRc_1R...Rc_{n-1}RbRd_1...Rd_{m-1}Rc \Rightarrow aR^{n+m+1}c \Rightarrow aR^+c$.

[$R \subseteq R^+$] $R = R^1 \Rightarrow R \subseteq R^1 \cup \ldots \cup R^{n-1} \cup R^n \cup \ldots \Rightarrow R \subseteq R^+$.

[$C(T)$ & $R \subseteq T \Rightarrow R^+ \subseteq T$] $aR^+b \Rightarrow \exists k\ (aR^kb) \Rightarrow$

$\Rightarrow \exists c_1,\ldots, c_{k-1}(aRc_1\ldots Rc_{k-1}Rb)$; $R \subseteq T \Rightarrow aTc_1T \ldots Tc_{k-1}Tb \Rightarrow aTb$. Таким образом, $R^+ \subseteq T$. □

Следствие. R^* - *рефлексивное транзитивное замыкание R.*

Доказательство. $R^* = I \cup R^+$, значит R^* рефлексивно. □

Вычислить транзитивное замыкание заданного отношения можно следующим образом:

$R^+ = R^1 \cup \ldots \cup R^{n-1} \cup R^n \cup \ldots = R^1 \cup \ldots \cup R^{n-1} \Rightarrow$

$R^+ = R^1 \vee \ldots \vee R^{n-1} = R^1 \vee \ldots \vee R^{n-1}$.

Такое вычисление будет иметь сложность $O(n^4)$. В следующем параграфе рассматривается более эффективный алгоритм, идея которого имеет удивительно широкую область применения.

1.5.3. Алгоритм Уоршалла

Рассмотрим алгоритм Уоршалла вычисления транзитивного замыкания отношения R на множестве M, $|M| = n$, имеющий сложность $O(n^3)$.

Алгоритм 1.11. Вычисление транзитивного замыкания отношения.

Вход: отношение, заданное матрицей R: **array**[1..n, 1..n] **of** 0..1.

Выход: транзитивное замыкание отношения, заданное матрицей T: **array**[1..n, 1..n] **of** 0..1.

```
S := R
for i from 1 to n do
  for j from 1 to n do
    for k from 1 to n do
      T[j, k] := S[j, k] ∨ S[j, i] & S[i, k]
    end for
  end for
  S := T
end for
```

Обоснование. На каждом шаге основного цикла (по i) в транзитивное замыкание добавляются такие пары элементов с номерами j и

k (то есть $T[j, k] := 1$), для которых существует последовательность промежуточных элементов с номерами в диапазоне $1..i$, связанных отношением R. Действительно, последовательность промежуточных элементов с номерами в диапазоне $1..i$, связанных отношением R, для элементов с номерами j и k существует в одном из двух случаев: либо уже существует последовательность промежуточных элементов с номерами в диапазоне $1..(i - 1)$ для пары элементов с номерами j и k, либо существуют *две* последовательности элементов с номерами в диапазоне $1..(i - 1)$ — одна для пары элементов с номерами j и i и вторая для пары элементов с номерами i и k. Именно это отражено в операторе $T[j, k] := S[j, k] \vee S[j, i] \,\&\, S[i, k]$. После окончания основного цикла в качестве промежуточных рассмотрены все элементы, и, таким образом, построено транзитивное замыкание. □

1.6. ФУНКЦИИ

Понятие "функция" является одним из основополагающих в математике. В данном случае подразумеваются, прежде всего, функции, отображающие одно конечное множество объектов в другое конечное множество. Термин "отображение" мы считаем синонимом термина "функция", но различаем контексты употребления. Термин "функция" используется в общем случае и в тех случаях, когда элементы отображаемых множеств не имеют структуры или она не важна. Термин "отображение" применяется, напротив, только в тех случаях, когда структура отображаемого множества имеет первостепенное значение. Например, мы говорим "булева функция", но "отображение групп". Такое предпочтение слову "функция" оказывается в расчёте на постоянное сопоставление читателем математического понятия функции с понятием функции в языках программирования.

1.6.1. Функциональные отношения

Пусть f — отношение между A и B, такое, что

$$\forall a\, ((a, b) \in f \,\&\, (a, c) \in f \Rightarrow b = c).$$

Такое свойство отношения называется *однозначностью*, или *функциональностью*, а само отношение называется функцией из A в B, причём для записи используется следующая форма: $f\colon A \to B$.

Если $f\colon A \to B$, то обычно используется *префиксная* форма записи: $b = f(a)$ **:=** $(a, b) \in f$. Если $b = f(a)$, то a называют *аргументом*, а b — *значением* функции. Если из контекста ясно, о какой функции идет речь, то иногда используется обозначение $a \rightarrowtail b$. Поскольку функция является отношением, для функции $f\colon A \to B$ можно использовать уже введенные понятия: множество A называется *областью отправления*, множество B — *областью прибытия* и

область определения функции: $\mathrm{Dom}\, f$ **:=** $\{a \in A \mid \exists b \in B(b = f(a))\}$;

область значений функции: $\mathrm{Im}\, f$ **:=** $\{b \in B \mid \exists a \in A\ (b = f(a))\}$.

Область определения является подмножеством области отправления, $\mathrm{Dom}\, f \subseteq A$, а область значений является подмножеством области прибытия, $\mathrm{Im}\, f \subseteq B$. Если $\mathrm{Dom}\, f = A$, то функция называется *тотальной*, а если $\mathrm{Dom}\, f \neq A$ — *частичной*. *Сужением* функции $f\colon A \to B$ на множество $M \subseteq A$ называется функция $f|_M$, определяемая следующим образом: $f|_M$ **:=** $\{(a, b) \mid (a, b) \in f \,\&\, a \in M\} = f \cap (M \times B)$. Функция f называется продолжением g, если g является сужением f.

Замечание. В математике, как правило, рассматриваются тотальные функции, а частичные функции объявлены "ущербными". Программисты же имеют дело с частичными функциями, которые определены не для всех допустимых значений. Например, математическая функция x^2 определена для всех x, а стандартная функция языка программирования sqr даёт правильный результат отнюдь не для всех возможных значений аргумента.

Далее, если явно не оговорено противное, речь идёт о тотальных функциях, поэтому области отправления и определения совпадают. Тотальная функция $f\colon M \to M$ называется *преобразованием* над M. Функция $\to f\colon A_1 \times \ldots \times A_n \to B$ называется функцией n аргументов, или n-местной. Такая функция отображает кортеж $(a_1, \ldots, a_n) \in A_1 \times \ldots \times A_n$

в элемент $b \in B$, $b = f((a_1, ..., a_n))$. При записи лишние скобки обычно опускают: $b = f(a_1, ..., a_n)$. Понятие функции удобно использовать для построения самых разнообразных математических моделей.

Примеры.

1) Всякому подмножеству A универсума U можно взаимно-однозначно сопоставить тотальную функцию 1_A: $U \to 0..1$. Эта функция называется *характеристической* функцией подмножества и определяется следующим образом: $1_A(a)$ **:= if** $a \in A$ **then** 1 **else** 0 **end if**. Характеристическая функция результата операции над множествами легко выражается через характеристические функции операндов:

$$1_{A \cup B} = \max(1_A, 1_B), \qquad 1_{A \cap B} = \min(1_A, 1_B),$$
$$1_{A'} = 1 - 1_A, \qquad 1_{A \setminus B} = \backslash\min(1_A, 1 - 1_B).$$

2) Всякому отношению R между A и B ($R \subseteq A \times B$) можно взаимно-однозначно сопоставить тотальную функцию 1_R: $A \times B \to 0..1$ (эта функция называется *характеристической* функцией отношения), полагая $R(a, b)$ **:= if** aRb **then** 1 **else** 0 **end if**. Характеристическая функция результата операции над отношениями легко выражается через характеристические функции операндов:

$$1_{R^{-1}}(a, b) = 1_R(b, a), \qquad 1_{R'} = 1 - 1_R.$$

1.6.2. Инъекция, сюръекция и биекция

Пусть f: $A \to B$. Тогда функция f называется:

инъективной, или *инъекцией*, если $b = f(a_1)$ & $b = f(a_2) \Rightarrow a_1 = a_2$;

сюръективной, или *сюръекцией*, если $\forall b \in B(\exists a \in A\ (b = f(a)))$;

биективной, или *биекцией*, если она инъекция и сюръекция.

Замечание. Биективную функцию также называют *взаимно-однозначной*. Введенное в параграфе 1.2.2 взаимно-однозначное соответствие есть биекция.

Понятия инъективности, сюръективности и тотальности применимы к любым отношениям, а не только к функциям. Таким образом,

получаем четыре парных свойства отношения $f \subseteq A \times B$, которые для удобства запоминания можно свести в следующую таблицу.

A	B
Функциональность $\forall a \in A\,((a, b) \in f \,\&\, (a, c) \in f \Rightarrow b = c)$	Инъективность $\forall b \in B\,((a, b) \in f \,\&\, (c, b) \in f \Rightarrow a = c)$
Тотальность $\forall a \in A\,(\exists b \in B\,((a, b) \in f))$	Сюръективность $\forall b \in B\,(\exists a \in A\,((a, b) \in f))$

Теорема. *Если* f: $A \to B$ — тотальная биекция, то отношение $f^{-1} \subseteq B \times A$ (*обратная функция*) является биекцией.

Доказательство. Поскольку f — биекция, имеем:

$((b_1 = f(a) \,\&\, b_2 = f(a) \Rightarrow b_1 = b_2)$ и $((b = f(a_1) \,\&\, b = f(a_2) \Rightarrow a_1 = a_2)$, а также $(\forall b \in B\,(\exists a \in A\,(b = f(a)))$. Покажем, что f^{-1} — функция. Имеем $f^{-1} = \{(b, a \mid a \in A \,\&\, b \in B \,\&\, b = f(a)\}$. Пусть $a_1 = f^{-1}(b) \,\&\, a_2 = f^{-1}(b)$. Тогда $b = f(a_1) \,\&\, = f(a_2) \Rightarrow a_1 = a_2$. Покажем, что f^{-1} — инъекция. Пусть $a = f^{-1}(b_1) \,\&\, a = f^{-1}(b_2)$. Тогда $b_1 = f(a) \,\&\, b_2 = f(a) \Rightarrow b_1 = b_2$. Покажем от противного, что f^{-1} — сюръекция. Пусть $\exists a \in A\,(\neg\exists b \in B\,(a = f^{-1}(b)))$. Тогда $\exists a \in A\,(\forall b \in B\,(a \neq f^{-1}(b)))$. Обозначим этот элемент a_0. Имеем $\forall b\,(a_0 \neq f^{-1}(b) \Rightarrow \forall b\,(b \neq f(a_0) \Rightarrow$ $\Rightarrow a_0 \notin f_A \subseteq A \Rightarrow a_0 \notin A$. □

Следствие. *Если* f: $A \to B$ *— инъективная функция, то отношение* $f^{-1} \subseteq B \times A$ *является функцией* (*возможно, частичной*), f^{-1}: $B \to A$.

1.6.3. Образы и прообразы

Пусть f: $A \to B$ и пусть $A_1 \subseteq A$, $B_1 \subseteq B$. Тогда множество

$$F(A_1) := \{b \in B \mid \exists a \in A_1 (b = f(a))\}$$

называется *образом* множества A_1 (при отображении f), а множество

$$F^{-1}(B_1) := \{a \in A \mid \exists b \in B_1\,(b = f(a))\}$$

называется *прообразом* множества B_1 (при отображении f). Заметим, что F является отношением между множествами $2^{\mathrm{Dom}\, f}$ и $2^{\mathrm{Im}\, f}$

$$F = \{(A_1, B_1) \mid A_1 \subseteq A \;\&\; B_1 \subseteq B \;\&\; B_1 = F(A_1)\},$$

а отношение F^{-1} действительно является обратным для отношения F^{-1}, и, таким образом, выбранное обозначение правомерно.

Замечание. Иногда используют альтернативные обозначения $f_A := \mathrm{Dom}\, f, f_B := \mathrm{Im}\, f$.

Теорема. *Если* $f\colon A \to B$ *— функция*, то $F\colon 2^{\mathrm{Dom}\, f} \to 2^{\mathrm{Im}\, f}$ (*переход к образам*) и $F^{-1}\colon 2^{\mathrm{Im}\, f} \to 2^{\mathrm{Dom}\, f}$ (*переход к прообразам*) *— тоже функции*.

Доказательство. Непосредственно следует из определений. □

Замечание. Фактически, это означает, что функцию можно применять не только к элементам области определения, а и к любым её подмножествам. На практике в целях упрощения записи для обозначения применения функции к подмножеству используют ту же самую букву, что и для применения функции к отдельному элементу.

Пусть задана функция $f : M \to M$, а функции F, $F^{-1} : 2^M \to 2^M$ обозначают функции перехода к образам и прообразам соответственно. Тогда нетрудно показать, что $\forall X, Y \subset M\, (F(X \cap Y) \subset F(X) \cap F(Y))$ и $\forall X, Y \subset M\, (F(X \cup Y) \subset F(X) \cup F(Y))$. Обычно по определению полагают, что $F(\varnothing) = F^{-1}(\varnothing) = \varnothing$.

1.6.4. Суперпозиция функций

Поскольку функции являются частным случаем отношений, для них определена композиция. Композиция функций называется *суперпозицией*. Для обозначения суперпозиции применяют тот же знак $\circ$, но операнды записывают в обратном порядке: если $f\colon A \to B$ и $g\colon B \to C$, то суперпозиция функций f и g записывается так: $g \circ f$. Такой способ записи суперпозиции функций объясняется тем, что обозначение функции принято писать слева от списка аргументов: $(f \circ g)(x) = f(g(x))$.

Теорема. *Суперпозиция функций является функцией*:

$$f\colon A \to B \;\&\; g\colon B \to C \;\Rightarrow g \circ f\colon A \to B \;\&\; g\colon B \to C.$$

Доказательство. Пусть $b = (g \circ f)(a)$ и $c = (g \circ f)(a)$. Тогда $b = g(f(a))$ и $c = g(f(a))$. Но f и g функции, поэтому $b = c$. □

Поскольку функция является отношением, можно ввести степень функции относительно суперпозиции. *Степенью n* функции f (обозначение f^n) называется n-кратная суперпозиция функции f. Соответственно, $f^0(x) = x, f^1(x) = f(x)$, и вообще $f^n = f \circ f^{n-1}$.

Замечание. В математическом анализе часто используют такое же обозначение для степени значения функции относительно умножения. Например, $\sin^2(x) = \sin(x) \cdot \sin(x)$. Следует различать по контексту, в каком смысле используется степень функции. В частности, в контексте данной книги $\sin^2(x) = \sin(\sin(x))$.

1.6.5. Представление функций в программах

Пусть $f\colon A \to B$, множество A конечно и не очень велико, $|A| = n$. Наиболее общим представлением такой функции является массив **array** [A] **of** B, где A — тип данных, значения которого представляют элементы множества A, а B — тип данных, значения которого представляют элементы множества B. Если среда программирования допускает массивы только с натуральными индексами, то элементы множества A нумеруются (то есть $A = \{a_1, \ldots, a_n\}$) и функция представляется с помощью массива **array** [1..n] **of** B. Функция нескольких аргументов представляется многомерным массивом.

Замечание. Представление функции с помощью массива является эффективным по времени, поскольку реализация массивов в большинстве случаев обеспечивает вычисление значения функции для заданного значения аргумента (индекса) за постоянное время, не зависящее от размера массива и значения индекса.

Если множество A велико или бесконечно, то использование массивов для представления функций является неэффективным с точки зрения экономии памяти. В таком случае для представления функций используется особый вид процедур, возвращающих единственное значение для заданного значения аргумента. Обычно такие процедуры также называются "функциями".

В некоторых языках программирования определения функций вводятся ключевым **function**. Многоместные функции представляются с помощью нескольких формальных параметров в определении функции. Свойство функциональности обеспечивается *оператором возврата*, часто обозначаемым ключевым словом **return**, который прекращает выполнение тела функции и одновременно возвращает значение.

Замечание. В языке программирования Фортран и в некоторых других языках вызов функции и обращение к массиву синтаксически неразличимы, что подчеркивает родственность этих понятий.

1.7. ОТНОШЕНИЯ ЭКВИВАЛЕНТНОСТИ

Различные встречающиеся на практике отношения могут обладать (или не обладать) теми или иными свойствами. Свойства, введенные в параграфе 1.4.6, встречаются особенно часто (потому им и даны специальные названия). Более того, оказывается, что некоторые устойчивые комбинации этих свойств встречаются настолько часто, что заслуживают отдельного названия и специального изучения. Здесь рассматриваются классы отношений, обладающих определённым набором свойств. Такое "абстрактное" изучение классов отношений обладает тем преимуществом, что, один раз установив некоторые следствия из наличия у отношения определённого набора свойств, далее эти следствия можно автоматически распространить на все конкретные отношения, которые обладают данным набором свойств. Рассмотрение начинается отношениями эквивалентности (в этом разделе) и продолжается отношениями порядка (в следующем разделе).

1.7.1. Классы эквивалентности

Рефлексивное симметричное транзитивное отношение называется отношением *эквивалентности*. Обычно отношение эквивалентности обозначают знаком ≡.

Примеры. Отношения равенства чисел, равенства множеств являются отношениями эквивалентности. Отношение равномощности множеств также является отношением эквивалентности. Другие, более интересные примеры отношений эквивалентности, приведены в последующих главах книги.

Пусть ≡ — отношение эквивалентности на множестве M и $x \in M$. Подмножество элементов множества M, эквивалентных x, называется *классом эквивалентности* для x:

$$[x]_{\equiv} := \{y \in M \mid y \equiv x\}.$$

Если отношение подразумевается, то нижний индекс у обозначения класса эквивалентности (знак отношения) обычно опускают.

Лемма 1. $\forall a \in M\,([a] \neq \varnothing)$.

Доказательство. $a \equiv a \Rightarrow a \in [a]$. □

Лемма 2. $a \equiv b \Rightarrow [a] = [b]$.

Доказательство. Пусть $a \equiv b$. Тогда $x \in [a] \Leftrightarrow x \equiv a \Leftrightarrow x \equiv b \Leftrightarrow x \in [b]$. □

Лемма 3. $\neg\, a \equiv b \Rightarrow [a] \cap [b] = \varnothing$.

Доказательство. От противного:

$[a] \cap [b] \neq \varnothing \Rightarrow \exists c \in M\,(c \in [a] \cap [b]) \Rightarrow c \in [a]$ & $c \in [b] \Rightarrow$

$c \equiv a$ & $c \equiv b \Rightarrow a \equiv c$ & $c \equiv b \Rightarrow a \equiv b$, противоречие. □

Теорема. *Если ≡ — отношение эквивалентности на множестве M, то классы эквивалентности по этому отношению образуют разбиение множества M, причём среди элементов разбиения нет пустых; и обратно, всякое разбиение $\mathcal{B} = \{B_i\}$ множества M, не содержащее пустых элементов, определяет отношение эквивалентности на множестве M, классами эквивалентности которого являются элементы разбиения.*

Доказательство.

[⇒] Построение требуемого разбиения обеспечивается следующим алгоритмом.

Вход: множество M, отношение эквивалентности ≡ на M.

Выход: разбиение $\mathcal{B}$ множества M на классы эквивалентности.

$\mathcal{B} := \varnothing$ // вначале разбиение пусто

for $a \in M$ **do**

for $B \in \mathcal{B}$ **do**

select $b \in B$ // берём представителя из B

if $a \equiv b$ **then**

$B := B+a$ // пополняем существующий класс

// эквивалнтности

next for A // переходим к рассмотрению

// следующего элемента из M

end if

end for

$\mathcal{B} := \mathcal{B}+\{a\}$ // вводим новый класс эквивалентности

end for

[⇒l] Положим $a \equiv b$ **:=** $\exists i(a \in B_i \,\&\, B \in B_i)$. Тогда отношение ≡ рефлексивно, поскольку $M = \cup B_i \Rightarrow \forall a \in M\ (\exists i\ (a \in B_i))$ и поскольку $a \in B_i \Rightarrow a \in B_i \,\&\, a \in B_i \Rightarrow a \equiv a$. Отношение ≡ симметрично, поскольку $a \equiv b \Rightarrow \exists i(a \in B_i \,\&\, b \in B_i) \Rightarrow \exists i\ (b \in B_i \,\&\, a \in B_i) \Rightarrow b \equiv a$. Наконец, отношение ≡ транзитивно, поскольку $a \equiv b \,\&\, b \equiv c \Rightarrow [a] = [b] \,\&\, [b] = [c] \Rightarrow [a] = [c] \Rightarrow a \equiv c$. □

1.7.2. Фактормножества

Если R — отношение эквивалентности на множестве M, то множество классов эквивалентности называется *фактормножеством* множества M относительно эквивалентности R и обозначается M/R:

$$M/R := \{[x]_R\}_{x \in M}.$$

Фактормножество является подмножеством булеана: $M/R \subseteq 2^M$. Функция nat R: $M \to M/R$ называется *отождествлением* и определяется следующим образом:

$$\text{nat } R(x) := [x]_R.$$

1.7.3. Ядро функционального отношения и множества уровня

Всякая функция f, будучи отношением, имеет ядро $f \circ f^{-1}$, которое является отношением на области определения функции.

Замечание. Даже если f — функция, f^{-1} отнюдь не обязательно функция, поэтому здесь $\circ$ — знак композиции отношений, а не суперпозиции функций.

Термин "ядро" применительно к функции f и обозначение ker f обычно резервируются для других целей, поэтому в формулировке следующей теоремы вместо слова "функция" употреблен несколько тяжеловесный оборот "функциональное отношение".

Теорема. *Ядро функционального отношения является отношением эквивалентности на определения*.

Доказательство. Пусть $f: A \to B$. Достаточно рассматривать случай, Dom $f = A$ и Im $f = B$.

[Рефлексивность] Пусть $a \in A$. Тогда $\exists b \in B\ (b = f(a)) \Rightarrow$ $\Rightarrow a \in f^{-1}(b) \Rightarrow (a, b) \in f$ & $(b, a) \in f^{-1} \Rightarrow (a, a) \in f \circ f^{-1}$.

[Симметричность] Пусть $(a_1, a_2) \in f \circ f^{-1}$. Тогда

$\exists b\ (b = f(a_1))$ & $a_2 \in f^{-1}(b) \Rightarrow a_1 \in f^{-1}(b)$ & $b = f(a_2) \Rightarrow$
$\Rightarrow b = f(a_2)$ & $a_1 \in f^{-1}(b) \Rightarrow (a_2, a_1) \in f \circ f^{-1}$.

[Транзитивность] Пусть $(a_1, a_2) \in f \circ f^{-1}$ и $(a_2, a_3) \in f \circ f^{-1}$. Это означает, что $\exists b_1 \in B\ (b_1 = f(a_1))$ & $a_2 \in f^{-1}(b_1)$ и $\exists b_2 \in B (b_2 =$ $= f(a_2))$ & $a_3 \in f^{-1}(b_2)$. Тогда $b_1 = f(a_1)$ & $b_1 = f(a_2)$ & $b_2 = f(a_2)$ & $b_2 =$

$= f(a_3) \Rightarrow b_1 = b_2$. Положим $b := b_1$ (или $b := b_2$). Тогда $b = f(a_1)$ & $b = f(a_3) \Rightarrow b = f(a_1)$ & $a_3 \in f^{-1}(b) \Rightarrow (a_1, a_3) \in f \circ f^{-1}$. □

Пусть $f: A \to B$ — функция и $f \circ f^{-1}$ — отношение эквивалентности на Dom f. Рассмотрим фактормножество Dom f / $(f \circ f^{-1})$. Класс эквивалентности (элемент фактормножества) — это подмножество элементов A, которые отображаются в один и тот же элемент $b \in B$. Такие множества называются *множествами уровня* функции f. Ясно, что Dom f / $(f \circ f^{-1})$| = |Im f|. Неформально множество уровня функции f, соответствующее значению c, — это множество решений уравнения $f(x) = c$.

Пример. Множество уровня 0 для функции $\sin(x)$ — это $\pi\mathbb{Z}$.

Замечание. Термин "множество уровня" опирается на очевидные геометрические ассоциации. Если нарисовать график функции одной переменной в обычной декартовой системе координат и провести горизонтальную прямую на определённом уровне, то абсциссы точек пересечения с графиком функции дадут множество уровня функции.

Пример. Пусть задано множество M, $|M| = n$. Рассмотрим функциональное отношение P между булеаном 2^M и множеством неотрицательных целых чисел (мощность):

$$P \subseteq 2^M \times 0..n, \text{ где } P := \{(X, k) \mid X \subseteq M \,\&\, k \in 0..n \,\&\, |X| = k\}.$$

Тогда ядром такого функционального отношения является отношение равномощности, а множествами уровня — семейства равномощных подмножеств.

1.8. ОТНОШЕНИЯ ПОРЯДКА

В этом разделе определяются различные варианты отношений порядка. Отношение порядка позволяет *сравнивать* между собой различные элементы одного множества. Фактически, интуитивные пред-

ставления об отношениях порядка уже были использованы при описании работы с упорядоченными списками.

1.8.1. Определения отношений упорядоченности

Рефлексивное транзитивное отношение на множестве M называется отношением *предпорядка*. Обычно отношение предпорядка обозначают знаком , а множество M называют *предупрядоченным.* Антирефлексивное транзитивное отношение на множестве M называют отношением *строгого предпорядка* (или *антирефлексивным предпорядком*) и обозначают знаком , а множество M называют *строго предупрядоченным*. Последовательность элементов $\langle a_1, \ldots, a_i, a_{i+1}, \ldots\rangle$ предупорядоченного множества M называется *цепочкой* в M, если $\forall i\ (a_{i+1} \quad a_i \ \&\ a_{i+1} \neq a_i)$.

Антисимметричное транзитивное отношение называется отношением *порядка*. Отношение порядка может быть рефлексивным, и тогда оно называется отношением *нестрогого порядка*. Ясно, что всякий нестрогий порядок является предпорядком. Отношение порядка может быть антирефлексивным, и тогда оно называется отношением *строгого порядка*. Ясно, что всякий строгий порядок является строгим предпорядком. Отношение порядка может быть линейным, и тогда оно называется отношением *линейного порядка*. Отношение порядка может не обладать свойством линейности, и тогда оно называется отношением *частичного порядка*. Обычно отношение строгого порядка (линейного или частичного) обозначается знаком $<$, а отношение нестрогого порядка — знаком $\leq$. Отношение порядка в общем случае обозначается знаком $\prec$.

Замечание. Для строгого порядка свойство антисимметричности можно определить следующим образом: $\forall a, b\ (a < b \Rightarrow \neg(b < a)) \equiv$ $\equiv \forall a, b\ (\neg(a < b) \vee \neg(b < a)) \equiv \forall a, b\ \neg\ (a < b\ \&\ b < a)$.

Примеры. Отношение $<$ на множестве вещественных чисел является отношением строгого линейного порядка. Отношение $\leq$ на множестве вещественных чисел является отношением нестрогого линейного порядка. Отношение $\subseteq$ на булеане 2^M является отношением нестрогого частичного порядка.

Множество, на котором задано отношение частичного порядка, называется *частично упорядоченным*. Множество, на котором задано отношение линейного порядка, называется *линейно упорядоченным*.

Пример. Множество вещественных чисел упорядочено линейно, а булеан упорядочен частично.

Теорема 1. *Если $\prec$ — отношение частичного порядка, то обратное отношение также является отношением частичного порядка*.

Доказательство.

[**антисимметричность**] $\forall a, b\ ((a \prec b)\ \&\ (b \prec a) \Rightarrow a = b) \Leftrightarrow$

$\forall a,\ b\ ((b \prec a)\ \&\ (a \prec b) \Rightarrow a = b) \Leftrightarrow \forall a,\ b\ ((a \quad b)\ \&\ (b \quad a) \Rightarrow a = b)$.

[**транзитивность**] $\forall a, b, c\ ((a \prec b)\ \&\ (b \prec c) \Rightarrow a \prec c) \Leftrightarrow$

$\forall a, b, c\ ((b \quad a) \& (c \quad b) \Rightarrow c \quad a) \Leftrightarrow$

$\Leftrightarrow \forall a, b, c\ ((c \quad b) \& (b \quad a) \Rightarrow c \quad a)$. □

Следствие. *Если $\prec$ — отношение частичного порядка, то отношения $\prec \backslash I$ и $\quad \backslash I$ являются отношениями строгого частичного порядка*.

Доказательство. Ясно, что $\forall R,\ I\ ((R \backslash I) \cap I = \varnothing)$ и по теореме 1.4.6. пункт 5 отношения $\prec \backslash I$ и $\quad \backslash I$ антирефлексивны. □

Теорема 2. *Если $<$ — отношение строгого частичного порядка, то дополнительное отношение $\geq$ является отношением нестрогого частичного порядка*.

Доказательство. $\neg(I \subseteq <) \Leftrightarrow I \subseteq <' \Leftrightarrow I \subseteq \geq$. □

Следствие. *Если* $\leq$ *– отношение нестрогого частичного порядка, то дополнительное отношение* $>$ *является отношением строгого частичного порядка*.

Даже если частично упорядоченное множество не является линейно упорядоченным, в нем могут найтись линейно упорядоченные подмножества.

Пример. Рассмотрим семейство подмножеств $\{C_i\}$ множества A такое, что $\forall i\ (C_i \subset C_{i+1}\ \&\ C_i \neq C_{i+1})$. Такое семейство образует цепочку относительно порядка $\subseteq$ (который является и предпорядком). Любая цепочка образует линейно упорядоченное подмножество частично упорядоченного множества 2^A.

Линейно упорядоченное подмножество $\{a_1, \ldots, a_i, \ldots\}$ можно рассматривать как последовательность $\langle a_1, \ldots, a_i, \ldots\rangle$, в которой $\forall i\ (a_i \prec a_{i+1}\ \&\ a_i \neq a_{i+1})$. Такие последовательности называются *строго монотонно возрастающими*, или просто *возрастающими*. Последовательности могут быть конечными или бесконечными. Если последовательность конечная, количество элементов в ней называется *длиной*. Говорят, что частично упорядоченное множество *имеет конечную высоту k*, если любая возрастающая последовательность имеет длину не более *k*.

1.8.2. Минимальные и наименьшие элементы

Элемент x множества M с отношением порядка $\prec$ называется *минимальным* в множестве M, если в M не существует меньших элементов: $\neg\exists y \in M\ (y \prec x\ \&\ y \neq x)$, иначе говоря, $\forall y \in M\ (\neg(y \prec x) \vee y = x)$.

Пример. Пустое множество $\varnothing$ является минимальным элементом булеана по включению.

Теорема 1. *Во всяком конечном непустом частично упорядоченном множестве существует минимальный элемент*.

Доказательство. От противного.

Пусть $\neg(\exists x \in M\,(\neg\exists y \in M\,(y \prec x \,\&\, y \neq x)))$.

Тогда $\forall x \in M\,(\exists y \in M\,(y \prec x))$.

Следовательно, $\exists u_1, \ldots u_n, \ldots\ (\forall I\,(u_{i+1} \prec u_i)\ \&\ u_{i+1} \neq u_i)$. Поскольку $|M| < \infty$, имеем $\exists i,\ j\ \ (i < j\ \&\ u_i = u_j)$. Но по транзитивности $u_i\ \ u_{i+1}\ \ \ldots\ \ u_j$, и значит $u_{i+1}\ \ u_j = u_i$. Таким образом, $u_{i+1} \prec u_i\ \&\ u_{i+1}\ \ u_i\ \ \Rightarrow\ u_{i+1} = u_i$ — противоречие. □

Замечание. Линейно упорядоченное конечное множество содержит ровно один минимальный элемент, а в произвольном конечном частично упорядоченном множестве их может быть несколько.

Теорема 2. *Всякий частичный порядок на конечном множестве может быть дополнен до линейного*.

Доказательство. См. следующий параграф. □

Замечание. В данном случае слова "может быть дополнен" означают, что существует отношение линейного порядка, которое является надмножеством заданного отношения частичного порядка.

Пусть M — частично упорядоченное множество с отношением порядка $\prec$, а X — его подмножество, $X \subseteq M$. Элемент a называется *наименьшим* в X (обозначение $a = \min X$), если $a \in X\ \&\ \forall x \in X\,(x \neq a \Rightarrow a \prec x)$. Из определения и антисимметричности порядка следует, что если наименьший элемент существует, то он единственен. Очевидно, что наименьший элемент, если он существует, является минимальным. Но элемент может быть минимальным, не будучи наименьшим. Более того, конечное частично упорядоченное множество, имея минимальные элементы, может не иметь наименьшего элемента.

Пример. Пусть в множестве $\{a, b, c, d\}$ задано отношение порядка $\{(a, b), (c, d)\}$. Тогда элементы a и c минимальные, но наименьшего элемента не существует.

Теорема. *Наименьший элемент, если он существует, является единственным*.

Доказательство. Пусть существуют два наименьших элемента a и b. Тогда $a \prec b$, поскольку a наименьший и, аналогично, $b \prec a$, поскольку b – наименьший. По антисимметричности, $a = b$.

В частично упорядоченном множестве с отношением порядка $\prec$ наряду с минимальными и наименьшими элементами можно определить *максимальные* и *наибольшие* элементы, используя обратное отношение .

Замечание. Используя теоремы параграфа 1.8.1. нетрудно показать, что все утверждения относительно минимальных и наименьших элементов имеют место и относительно максимальных и наибольших элементов, соответственно, то есть имеет место *двойственность*.

1.8.3. Алгоритм топологической сортировки

Рассмотрим алгоритм дополнения частичного порядка до линейного на конечном множестве.

Алгоритм

Вход: конечное частично упорядоченное множество U.

Выход: линейно упорядоченное множество W.

while $U \neq \varnothing$

    $m := \min(U)$ // min возвращает минимальный элемент

    yield m // включаем элемент m в множество W

    $U := U - m$ // элемент m более не рассматривается

end while

Обоснование. Всякая процедура, генерирующая объекты с помощью оператора **yield**, определяет линейный порядок на множестве своих результатов. Действительно, линейный порядок — это последовательность, в которой объекты генерируются во время работы процедуры. □

Функция min возвращает минимальный элемент, существование которого гарантируется теоремой 1 параграфа 1.8.2.

Вход: конечное частично упорядоченное множество U.

Выход: минимальный элемент m.

```
select m ∈ U        // выбираем любой элемент в качестве
                    // кандидата в минимальные,
for u ∈ U do
    if u ≺ m then
        m := u // меняем кандидата в минимальные
    end if
end for
return m            // элемент m минимальный
```

Обоснование. Рассмотрим последовательность элементов, которые присваивались m во время работы min. Эта последовательность не может быть бесконечной, и она линейно упорядочена. Рассмотрим последний элемент этой последовательности. Этот элемент m — минимальный. Действительно, от противного, пусть существует такой элемент u, что $u \prec m \;\&\; u \neq m$ и u не входит в последовательность. Тогда по транзитивности u меньше любого члена последовательности, и он был бы включен в последовательность в момент своего рассмотрения. □

Замечание. Если отношение порядка представлено матрицей, то функцию min можно реализовать, например, так — найти в матрице отношения первую строку, содержащую только нули.

1.8.4. Верхние и нижние границы

Пусть $X \subseteq M$ — подмножество упорядоченного множества M с отношением порядка $\prec$. Элемент $m \in M$ называется *нижней границей* для подмножества X, если $\forall x \in X\,(m \prec x)$. Элемент $m \in M$ называется *верхней границей* для подмножества X, если $\forall\, x \in X\,(m \prec x)$. Верхние и нижние границы не обязаны существовать для любого множества и

если существуют, то не всегда единственны. Если существует наибольшая нижняя граница, то она называется *инфимумом* и обозначается $\inf(X)$. Если существует наименьшая верхняя граница, то она называется *супремумом* и обозначается $\sup(X)$.

Пример. Рассмотрим множество рациональных чисел $\mathbb{Q}$ с обычным отношением порядка $<$ и его подмножество $X := \{x \in \mathbb{Q} \mid x > 0 \ \& \ x^2 > 2\}$. Тогда $17/12 \in X$, а $17/13$ является одной из нижних границ множества X. Инфимума это множество не имеет.

Теорема. *Пусть M — частично упорядоченное множество, а X — любое его подмножество. Тогда:*

1) *Если существует наименьший элемент* $a = \min X$, *то* $a = \inf X$;

2) *Если существует инфимум* $a = \inf X$ *и* $a \in X$, *то* $a = \min X$.

Доказательство.

[1] $\forall x \in X \ (a \prec x)$, следовательно, a – нижняя граница X. Пусть b — любая нижняя граница X, тогда $b \prec a$, так как $a \in X$. Следовательно, a — наибольшая нижняя граница.

[2] $\forall x \in X \ (a \prec x)$, поскольку a — нижняя граница X, и значит это наименьший элемент, поскольку $a \in X$.

Следствие. *Пусть M — частично упорядоченное множество, а X — любое его подмножество. Тогда:*

1) *Если существует наибольший элемент* $a = \max X$, то $a = \sup X$;

2) *Если существует супремум* $a = \sup X$ *и* $a \in X$, *то* $a = \max X$.

Говорят, что частично упорядоченное множество *линейно полно*, если любое его линейно упорядоченное подмножество имеет супремум.

Пример. Если множество X конечно, то 2^X — линейно полно относительно $\subseteq$.

1.8.5. Монотонные и непрерывные функции

Пусть A и B — упорядоченные множества и $f\colon A \to B$.

Тогда если

$$a_1 \prec a_2 \;\&\; a_1 \neq a_2 \Rightarrow f(a_1) \prec f(a_2) \vee f(a_1) = f(a_2),$$

то функция f называется *монотонно возрастающей*. Если

$$a_1 \prec a_2 \;\&\; a_1 \neq a_2 \Rightarrow f(a_2) \prec f(a_1) \vee f(a_1) = f(a_2),$$

то функция f называется *монотонно убывающей*. Если

$$a_1 \prec a_2 \;\&\; a_1 \neq a_2 \Rightarrow f(a_1) \prec f(a_2) \vee f(a_1) \neq f(a_2),$$

то функция f называется *строго монотонно возрастающей*.

Если $a_1 \prec a_2 \;\&\; a_1 \neq a_2 \Rightarrow f(a_2) \prec f(a_1) \vee f(a_1) \neq f(a_2),$

то функция f называется *строго монотонно убывающей*. Монотонно возрастающие и убывающие функции называются *монотонными*, а строго монотонно возрастающие и убывающие функции называются *строго монотонными* соответственно. Очевидно, что строго монотонная функция монотонна, но обратное неверно.

Примеры.

1) Тождественная функция $y = x$ для чисел является строго монотонно возрастающей, а функция знака числа

sign(x) **:= if** $x < 0$ **then** –1 **elseif** $x > 0$ **then** 1 **else** 0 **end if**

является монотонно возрастающей.

2) Пусть 2^M — булеан над линейно упорядоченным конечным множеством M, а частичный порядок на булеане — это включение. Тогда функция min(X), $X \subseteq M$ из алгоритма 1.12, доставляющая минимальный элемент, является монотонно убывающей, но не строго монотонной.

3) Пусть порядок на булеане 2^M — это собственное включение. Тогда функция, которая сопоставляет подмножеству $X \subseteq M$ его мощность, $X \rightarrowtail |X|$, является строго монотонно возрастающей.

4) Пусть $f : M \to M$ – некоторая функция. Тогда функция перехода к образам (см. 1.6.3.) $F : 2^M \to 2^M$ является монотонно возрастающей по включению.

Теорема 1. *Суперпозиция одинаково монотонных функции монотонна в том же смысле*.

Доказательство. Пусть $a \prec b$ и g монотонно возрастает, тогда $g(a) \prec g(b)$. Но f также монотонно возрастает и значит $f(g(a)) \prec f(g(b))$.

Функция $f : M \to M$, где M – частично упорядоченное множество, называется *непрерывной*, если для любой возрастающей последовательности $a_1 \prec \ldots \prec a_n \prec \ldots$, для которой существует супремум, выполнено равенство: $f(\sup\{a_i\}) = \sup\{f(a_i)\}$.

Замечание. Учитывая теоремы параграфа 1.8.1 и последнее замечание параграфа 1.8.2, непрерывность можно определить через строго убывающие и имеющие инфимум последовательности.

Теорема 2. *Непрерывная функция монотонно возрастает*.

Доказательство. Пусть $a \prec b$. Тогда $\sup\{a, b\} = b$. Имеем $f(b) = f(\sup\{a, b\}) = \sup\{f(a), f(b)\}$. Следовательно, $f(a) \prec f(b)$.

Теорема 3. *Суперпозиция непрерывных функций непрерывна*.

Доказательство. Рассмотрим строго монотонно возрастающую последовательность $\{a_n\}$, имеющую супремум $a = \sup\{a_n\}$. $f(g(a)) = f(g(\sup\{a_n\})) = f(\sup\{g(a_n)\}) = \sup\{f(g(a_n))\}$.

1.8.6. Наименьшая неподвижная точка функции

Рассмотрим функцию $f : M \to M$. Элемент $x \in M$ называется *неподвижной точкой* функции f, если $f(x) = x$.

Замечание. Неподвижная точка функции f — это то же, что и *корень* уравнения $f(x) = x$.

Пример. Пусть $f : X \to X$. Рассмотрим функцию перехода к образам F: $2^X \to 2^X$. Тогда множество $A = \cap\{Y \subseteq X \mid F(Y) \subseteq Y\}$ (т. е. пересечение всех таких подмножеств Y множества X, образы которых являются подмножествами аргументов) является неподвижной точкой функции F. Покажем, что $F(A) \subseteq A$. Действительно,

$$F(A) = F(\bigcap\{Y \subseteq X \mid F(Y) \subseteq Y\}) \subseteq \bigcap F(Y) = A.$$

Отсюда по монотонности F имеем $F(F(A)) \subseteq F(A)$, то есть $F(A)$ удовлетворяет условию для множеств Y и значит $A \subset F(A)$. Имеем $A = F(A)$, значит A — неподвижная точка функции F.

Замечание. Если $A = \bigcap \{Y \subseteq X \mid F(Y) \subseteq Y\} = \varnothing$, то $F(\varnothing) = \varnothing$ и пустое множество является неподвижной точкой функции перехода к образам по определению. Вообще говоря, функция f: $X \to X$ может не иметь неподвижных точек, иметь одну неподвижную точку или иметь их несколько, может быть, даже бесконечно много. Если множество X упорядочено, и функция f: $X \to X$ имеет неподвижные точки, то среди них можно выделить *наименьшую неподвижную точку*. Однако даже если функция имеет неподвижные точки, среди них может не быть наименьшей, просто потому, что не всякое, даже конечное, частично упорядоченное множество имеет наименьший элемент.

Пример. Наименьшей неподвижной точкой для функции перехода к образам $F : 2^X \to 2^X$ является множество $A = \bigcap\{Y \subseteq X \mid F(Y) \subseteq Y\}$, введенное в предыдущем примере. Действительно, пусть $\exists B \subseteq X(F(B) = B \ \& \ A \not\subset B)$. Тогда $F(B) \subseteq B$, и значит, множество B является одним из множеств Y. Получается, что пересечение не является подмножеством одного их пересекаемых множеств — противоречие.

Теорема [о наименьшей неподвижной точке]. *Если множество X частично упорядочено, линейно полно, имеет наименьший элемент* $\mathbf{0} = \min X$, *и функция f: $X \to X$ непрерывна и тотальна*, $\mathrm{dom}\, f = X$, *то* $a = \sup\{f^n(\mathbf{0}) \mid n \geq 0\}$ *является наименьшей неподвижной точкой функции f.*

Доказательство. Поскольку функция f тотальна, элементы $f^n(\mathbf{0})$ определены для любого $n \geq 0$. Рассмотрим подмножество $Y := \{f^n(\mathbf{0}) \mid n \geq 0\}$. Имеем: $f^0(\mathbf{0}) = \mathbf{0} \prec f^1(\mathbf{0})$, поскольку $\mathbf{0}$ – наименьший элемент. Поскольку функция f непрерывна, она и монотонна, а значит $f^1(\mathbf{0}) = f(f^0(\mathbf{0})) \prec f^2(\mathbf{0}) = f(f^1(\mathbf{0})), f^2(\mathbf{0}) = f(f^1(\mathbf{0})) \prec f^3(\mathbf{0}) = f(f^2(\mathbf{0}))$

и вообще $f^n(\mathbf{0}) \prec f^{n+1}(\mathbf{0})$ для любого $n \geq 0$. Таким образом, подмножество Y линейно упорядочено, а значит элемент $a = \sup\{f^n(\mathbf{0}) \mid n \geq 0\}$ определен по условию теоремы. Покажем теперь, что элемент a действительно является неподвижной точкой функции f, то есть $f(a) = a$.

Имеем:

$$f(a) = f(\sup\{f^n(\mathbf{0}) | n \geq 0\}) = \sup\{f(f^n(\mathbf{0})) | n \geq 0\} =$$

$$= \sup\{f^n(\mathbf{0}) | n \geq 1\} = \sup\{\mathbf{0}, a\} = a.$$

Покажем теперь, что a — наименьшая неподвижная точка. Пусть b — любая неподвижная точка. Поскольку $f(b) = b$, имеем $\forall n \geq 0$ $(f^n(b) = b)$. Но $\mathbf{0} \prec b$ и функция f монотонна, следовательно $\forall n \geq 0 (f^n(\mathbf{0}) \prec b)$, то есть b — верхняя граница множества $\{f^n(\mathbf{0}) \mid n \geq 0\}$, а значит $a \prec b$.

Замечание. Требование непрерывности в формулировке теоремы нужно только для того, чтобы гарантировать существование супремумов. Вообще говоря, если супремумы заведомо существуют, то от функции требуется только монотонность.

Фактически, приведенная теорема о наименьшей неподвижной точке служит обоснованием одного из случаев, когда для решения уравнения $x = f(x)$ правомерно применять метод итераций: $x_{i+1} := f(x_i)$.

1.8.7. Вполне упорядоченные множества

Частично упорядоченное множество X называется *вполне упорядоченным*, если любое его непустое подмножество имеет наименьший элемент. В частности, для любых двух элементов $a, b \in X$ один из них обязан быть наименьшим в подмножестве $\{a, b\}$, а значит, вполне упорядоченное множество упорядочено линейно.

Замечание. Не надо путать вполне упорядоченные множества и множества, на которых определён линейный (или полный) порядок. Линейно упорядоченное множество может не быть вполне упорядоченным.

Примеры.

1) Всякое конечное линейно упорядоченное множество вполне упорядочено.

2). Множество натуральных чисел $\mathbb{N}$ с обычным отношением порядка вполне упорядочено.

3) Множество рациональных чисел $\mathbb{Q}$ с обычным отношением порядка не является вполне упорядоченным, так как множество $X := \{x \in \mathbb{Q} \mid x^2 > 2\}$, равно как и само множество $\mathbb{Q}$, не имеет минимального элемента.

4) Множество вещественных чисел $\mathbb{R}$ с обычным отношением порядка не является вполне упорядоченным, так как множество $X := \{x \in \mathbb{R} \mid x > 0\}$, равно как и само множество $\mathbb{R}$, не имеет минимального элемента.

Говорят, что два линейно упорядоченных множества A и B *изоморфны* (обозначение $A \sim B$), если между ними существует взаимно-однозначное соответствие, сохраняющее порядок:

$$A \sim B := |A| = |B| \;\&\; (\forall a_1 \prec a_2 \in A\ (a_1 \prec a_2 \;\&\; a_1 \rightarrowtail b_1 \;\&\; a_2 \rightarrowtail b_2 \Rightarrow$$

$$\Rightarrow b_1 \prec b_2)).$$

Другими словами, два линейно упорядоченных множества A и B изоморфны, если между ними существует строго монотонно возрастающее взаимно-однозначное соответствие.

Замечание. Поскольку вполне упорядоченные множества упорядочены линейно, понятие *изоморфизма* линейно упорядоченных множеств применимо и к вполне упорядоченным множествам.

Пример. Множества натуральных чисел и чётных чисел с обычным порядком $<$ изоморфны, поскольку соответствие $n \rightarrowtail 2n$ является строго монотонно возрастающим.

1.8.8. Индукция

Важность вполне упорядоченных множеств определяется тем, что для них можно обобщить принцип *индукции*, применяемый обычно для множества натуральных чисел.

Теорема. [Индукция по вполне упорядоченному множеству] *Пусть X — вполне упорядоченное множество и x_0 - его минимальный элемент, а P - некоторый предикат, зависящий от элементов X. Тогда если*

$$P(x_0) \,\&\, \forall x_1 \in X \left((\forall x \in X\,(x \prec x_1 \Rightarrow P(x)\right) \Rightarrow P(x_1),$$

то

$$\forall\, x \in X\,(P(x)).$$

Доказательство. Обозначим $P := \{x \in X \mid \neg\, P(x)\}$, $P' \subseteq X$. Далее от противного. Пусть $P' \neq \varnothing$. Поскольку X вполне упорядоченно, P' имеет наименьший элемент, обозначим его x_1. Тогда $\forall x \in X\,(x \prec x_1 \Rightarrow$ $\Rightarrow P(x))$ по выбору x_1 и значит $P(x_1)$ по условию теоремы, что противоречит выбору x_1. □

Замечание. Математическая индукция соответствует индукции по вполне упорядоченному множеству натуральных чисел $\mathbb{N}$.

Индукция по вполне упорядоченному множеству сильнее обычного принципа математической индукции в силу следующей замечательной теоремы.

Теорема. *Любое множество может быть вполне упорядочено*.

Данное утверждение эквивалентно так называемой *аксиоме выбора* в теории множеств и само может быть принято за аксиому. Мы опускаем доказательство эквивалентности этой теоремы аксиоме выбора.

Пример. Рассмотрим множество положительных рациональных чисел $\mathbb{Q}_+ := \{m/n \mid m, n \in \mathbb{N}\}$, где m и n взаимно просты. Определим отношение $\prec$ на множестве $\mathbb{Q}_+$ следующим образом:

$$m_1/n_1 \prec m_2/n_2 := m_1 < m_2 \vee \big(m_1 = m_2 \;\&\; n_1 < n_2\big),$$

где $<$ — обычное отношение порядка на $\mathbb{N}$.

Нетрудно видеть, что множество $\mathbb{Q}_+$ с отношением $\prec$ является вполне упорядоченным, в то время как с обычным отношением порядка $<$ оно таковым не является.

ВЫВОДЫ

Обозначения и понятия теории множеств — *множества*, *отношения* и *функции* — являются основой того языком, на котором формулируются прикладные математические модели.

2. ВЫЧИСЛИМОСТЬ И СЛОЖНОСТЬ

Предметом этой главы являются строгие математические понятия, которые формализуют наши представления о том, что некоторые объекты или функции могут быть вычислены с помощью подходящей программы, в то время как объекты заданные "неэффективным" определением такими программами не вычисляются, по крайней мере, без привлечения других, более эффективных методов определения. Говоря о вычислимости или невычислимости, мы будем главным образом интересоваться существованием "программы" вычисляющей описанную функцию. Проблема объема памяти или длины соответствующего вычисления выйдет на передний план в разделах, посвященных анализу сложности алгоритмов. Областями определения и значениями функций будут главным образом целые положительные числа или конечные наборы таких чисел. Конечно, это связано с тем, что все реальные компьютеры имеют дело с числами, ограниченными длиной ячейки памяти, прибегая к какой-либо процедуре кодирования, иногда довольно изощренной, для представления других объектов.

2.1. ИНТУИТИВНАЯ ВЫЧИСЛИМОСТЬ

В этом разделе мы даём представление о вычислимости, используя наивное представление о программе или же алгоритме, как некоторой процедуре, получающей некоторые данные на входе и вырабатывающей некоторые данные на выходе. Мы увидим в дальнейшем, что самые различные способы формализации этих понятий приводят по существу к одному и тому же результату, описывая фактически один и тот же класс функций, как эффективно вычислимых. Этот глубокий и далеко не очевидный факт отражен в знаменитом тезисе Чёрча, который не надо рассматривать как теорему, а скорее как некоторый экспериментальный факт, находящий постоянно все новые подтверждения.

2.1.1. Частичные функции

Пусть X и Y — два множества. *Частичной функцией* из X в Y будем называть пару $\langle \mathrm{Dom}\, f, f\rangle$, состоящую из подмножества $\mathrm{Dom}\, f \subseteq X$ и отображения f: $\mathrm{Dom}\, f \to Y$. Напомним, что $\mathrm{Dom}\, f$ называется *областью определения* f. Будем говорить, что f определена в точке x, если $x \in \mathrm{Dom}\, f$. Если $\mathrm{Dom}\, f$ пусто, то f нигде не определена. В качестве областей определения и значения рассматриваемых функций будем рассматривать подмножества $\mathbb{Z}_+^n$, т. е. множества векторов, состоящих из положительных целых чисел. Частичная функция f: $\mathbb{Z}_+^m \to \mathbb{Z}_+^n$ называется "интуитивно" вычислимой, если существует такая "программа", что при подаче на ее вход вектора $x \in \mathbb{Z}_+^m$ мы получим на выходе $f(x)$, если $x \in \mathrm{Dom}\, f$ и 0, если $x \notin \mathrm{Dom}\, f$. Здесь 0 не принадлежит множеству значений f, а служит указателем на то, что x не входит в область определения f. Вместо 0 можно было бы использовать любой другой символ. Частичная функция f: $\mathbb{Z}_+^m \to \mathbb{Z}_+^n$ называется *полувычислимой*, если существует "программа", вырабатывающая значение $f(x)$, если $x \in \mathrm{Dom}\, f$ или же программа работает бесконечно долго, если $x \notin \mathrm{Dom}\, f$. Все вычислимые функции полувычислимы. Всюду определенные полувычислимые функции вычислимы.

2.1.2. Невычислимые функции

Если частичная функция не удовлетворяет условиям из предыдущего параграфа, то она называется *невычислимой*.

Теорема. *Существуют невычислимые функции*.

Доказательство. (Нестрогое) Каждая программа — это текст в конечном алфавите, поэтому множество программ счётно, тогда как множество всех функций $\mathbb{Z}_+ \to \mathbb{Z}_+$ несчётно. □

Замечание. В этом доказательстве нет уточнения понятия "программа", поэтому оно не является вполне строгим. Оно станет таковым, как только мы дадим какое-либо формальное определение в одной из возможных моделей вычислений.

2.2. РЕКУРСИЯ

В этом разделе дана формализация класса вычислимых и полувычислимых функций, которая остается адекватной при любом разумном определении понятия "программа" или "вычислительная процедура".

2.2.1. Простейшие функции

Мы будем строить вычислимые и полувычислимые функции, начиная с простейших функций и используя простые операции для построения все более сложных. В качестве простейших функций мы возьмем следующие:

Suc: $\mathbb{Z}_+ \to \mathbb{Z}_+$, $\mathrm{Suc}(x) = x + 1$, прибавление единицы,

$\mathbb{I}^n$: $\mathbb{Z}_+{}^n \to \mathbb{Z}_+$, $\mathbb{I}^n(x_1, x_2, ..., x_n) = 1$, "единичная" функция.

pr_i^n: $\mathbb{Z}_+{}^n \to \mathbb{Z}_+$, $\mathrm{pr}_i^n(x_1, x_2, ..., x_n) = x_i$, проекция на i-ую координату.

2.2.2. Элементарные операции над частичными функциями

Здесь мы введем элементарные операции над частичными функциями. В дальнейшем при формализации понятия "программа" этим операциям над функциями будут соответствовать операции над программами так, что результатом применения таких операций к вычислимым (полувычислимым) функциям будет снова вычислимая (полувычислимая) функция.

а) *Композиция*. Это обычная композиция функций. Она ставит в соответствие паре функций f: $\mathbb{Z}_+{}^m \to \mathbb{Z}_+{}^n$ и g: $\mathbb{Z}_+{}^n \to \mathbb{Z}_+{}^k$ функцию $h = g \circ f$: $\mathbb{Z}_+{}^m \to \mathbb{Z}_+{}^k$ там, где последняя определена.

$\mathrm{Dom}\,(g \circ f) = f^{-1}(\mathrm{Dom}\, g) =$

$= \{x \in \mathbb{Z}_+^{\,m} \mid x \in \mathrm{Dom}\ f\ \&\ f(x) \in \mathrm{Dom}\ g\}$. Напомним, что $(g \circ f)(x) = g(f(x))$.

b) *Соединение*. Эта операция ставит в соответствие частичным функциям f_i: $\mathbb{Z}_+^{\,m} \to \mathbb{Z}_+^{\,n_i}$ $i = 1, ..., k$ функцию

$$(f_1, ..., f_k)\colon \mathbb{Z}_+^{\,m} \to \mathbb{Z}_+^{\,n_1} \times \mathbb{Z}_+^{\,n_2} \times ... \times \mathbb{Z}_+^{\,n_k}$$

следующим образом

$$\mathrm{Dom}\,(f_1, ..., f_k) := \mathrm{Dom}\ f_1 \cap \mathrm{Dom}\ f_2 \cap ... \cap \mathrm{Dom}\ f_k,$$

$$(f_1, f_2 ... f_k)\,(x_1, ..., x_m) := (f_1(x_1, ..., x_m), f_2(x_1, ..., x_m), ... f_k(x_1, ..., x_m)).$$

c) *Рекурсия*. Эта операция ставит в соответствие паре функций f: $\mathbb{Z}_+^{\,n} \to \mathbb{Z}_+$ и g: $\mathbb{Z}_+^{\,n+2} \to \mathbb{Z}_+$ функцию h: $\mathbb{Z}_+^{\,n+1} \to \mathbb{Z}_+$, определяемую следующим образом^

$$h(x_1, x_2 ... x_n, 1) := f(x_1, ..., x_n),$$

$$h(x_1, x_2 ... x_n, k + 1) := g(x_1, x_2 ... x_n, k, h(x_1, ... x_n, k)) \text{ при } k \geq 1.$$

Эти уравнения называются *уравнениями рекурсии*. Область определения Dom h описывается следующим образом:

$$(x_1 ... x_n, 1) \in \mathrm{Dom}\, h \Leftrightarrow (x_1, x_2, ..., x_n) \in \mathrm{Dom}\, f$$

$$(x_1 ... x_n, k + 1) \in \mathrm{Dom}\, h \Leftrightarrow (x_1 ... x_n, k) \in \mathrm{Dom}\, h.$$

2.2.3. Лемма об индуктивных определениях

Существование и единственность функции h из предыдщего параграфа обеспечивается следующей теоремой, которая часто называется "Леммой об индуктивных определениях".

Теорема. *Пусть* f: $\mathbb{Z}_+^{\,n} \to \mathbb{Z}_+$ *и* g: $\mathbb{Z}_+^{\,n+2} \to \mathbb{Z}_+$ *частичные функции. Тогда существует единственная функция* h: $\mathbb{Z}_+^{\,n+1} \to \mathbb{Z}_+$, *удовлетворяющая уравнениям рекурсии*.

Доказательство. Без доказательства. □

Примеры.

1) Факториал.

Имеем $1! = 1$ и $(n + 1)! = n!\cdot(n + 1)$, следовательно функция $n!$ определена рекурсией из функции $\mathbb{I}$, где функция $g(n, e) = e\cdot(n + 1)$.

2) Сложение.

Так как $x + 0 = x$, а $x + (y + 1) = (x + y) + 1$, то функция $h(x, y) = x + y$ определена рекурсией над функциями $f(x) = x$ и $g(x, y, z) = z + 1$.

Замечание. Рекурсия, основанная на применении указанных правил, называется примитивной рекурсией. В математической логике рассматриваются и более общие методы рекурсивных определений, нежели примитивная рекурсия.

2.3. МАШИНА С НЕОГРАНИЧЕННЫМИ РЕГИСТРАМИ

В предыдущих разделах мы неформально описали понятие алгоритма, эффективной процедуры или программы. Теперь мы дадим формальное определение в терминах некоторого "идеализированного" компьютера, способного выполнять программы. Архитектура нашего компьютера, который называется *машиной с неограниченными регистрами* (МНР) напоминает архитектуру современных компьютеров.

2.3.1. Описание МНР

МНР содержит бесконечное число регистров R_1, R_2, R_3, ..., в каждом из которых в любой момент времени может храниться натуральное число. Число, содержащееся в R_n, мы будем обозначать r_n. МНР изменяет содержимое регистров в ответ на некоторую команду. Эти команды соответствуют простейшим операциям над числами.

Конечный список команд образует "программу" P. МНР имеет четыре вида команд.

1) *Команды обнуления*. Для каждого $n = 1, 2, ...$ имеется команда обнуления $Z(n)$. При исполнении команды $Z(n)$ содержимое регистра R_n заменяется на 0. Остальные регистры остаются без изменения. Мы записываем реакцию МНР на команду $Z(n)$ как $0 \to R_n$ или $r_n := 0$.

2) *Команды прибавления единицы*. Для каждого $n = 1, 2, ...$ имеются команды $S(n)$. Под действием команды $S(n)$ МНР увеличивает R_n на 1. Остальные регистры остаются без изменений. Реакция на команду $S(n)$ может быть записана как $r_n + 1 \to R_n$ или $r_n := r_n + 1$.

3) *Команды переадресации*. Для каждой пары натуральных чисел (m, n) имеется команда переадресации $T(m, n)$. Исполняя команду $T(m, n)$, МНР заменяет содержимое регистра R_n на число r_m, содержащееся в R_m. Все остальные регистры, включая R_m, не изменяются.

4) *Команды условного перехода*. Эти команды моделируют команды ветвления. Для каждой тройки натуральных чисел (m, n, q) имеется команда $J(m, n, q)$. Исполняя команду $J(m, n, q)$ МНР сравнивает содержимое регистров R_m и R_n. Если при этом $r_m = r_n$, то МНР переходит к выполнению q-й команды из списка команд; если $r_m \neq r_n$, то МНР переходит к выполнению следующей команды из списка. Если условный переход невозможен, ввиду того, что программа P содержит менее q команд, то МНР прекращает работу.

Команды обнуления, прибавления единицы и переадресации называются *арифметическими*.

Пример. Пусть МНР находится в конфигурации

R_1	R_2	R_3	R_4	R_5
3	0	15	21	17

и выполняет программу P, состоящую из команд

$Z(3)$

$S(5)$

$T(5, 4)$

$J(2, 3, 5)$.

Результатом работы будет конфигурация

R_1	R_2	R_3	R_4	R_5
3	0	0	18	18

В результате выполнения последней команды $J(2, 3, 5)$ МНР остановится, так как в нашем списке только четыре команды.

2.3.2. Вычисление на МНР

Чтобы производить вычисления на МНР, задают программу P и начальную конфигурацию, то есть последовательность $r_1, r_2 \ldots r_n \ldots$ натуральных чисел в регистрах $R_1, R_2 \ldots R_n \ldots$. Работа МНР заканчивается когда либо исчерпан список команд, либо очередная команда условного перехода $J(m, n, q)$ не может быть выполнена. Обратите внимание на то, что программа в памяти машины *не хранится* и никак не может быть изменена в процессе вычисления.

Замечание. Наша МНР в некотором смысле мощнее реальных компьютеров. Она обладает бесконечной памятью. Кроме этого, в каждом из регистров может храниться любое сколь угодно длинное натуральное число. Это обстоятельство сильно облегчает анализ процесса вычисления на МНР, так как не надо следить за размером чисел, помещаемых в регистры.

Теперь можно придать точный смысл понятию вычислимой функции. Пусть P некоторая программа для МНР. Для произвольного набора чисел $(a_1, a_2, \ldots, a_n) \in \mathbb{Z}_+^{\,n}$ вычисление на МНР по программе P из начальной конфигурации $a_1, a_2, \ldots, a_n, 0, 0, \ldots$ мы обозначаем $P(a_1,a_2,\ldots,a_n)$. Если программа P заканчивает работу в процессе вычисления $P(a_1, \ldots, a_n)$, то мы пишем $P(a_1, \ldots, a_n)\downarrow$. В противном случае мы пишем $P(a_1, \ldots, a_n)\uparrow$. Мы будем записывать

$$P(a_1, a_2, \ldots, a_n) \downarrow (b_1, b_2, \ldots, b_m),$$

если процесс $P(a_1, a_2, \ldots, a_n)$ оканчивается и в заключительной конфигурации $r_1 = b_1,\ r_2 = b_2,\ \ldots\ r_m = b_m$. Пусть задана частичная функция

f: $\mathbb{Z}_+^n \to \mathbb{Z}_+^m$. Говорят, что программа P для МНР *вычисляет* функцию f, если

$$\forall(a_1, a_2,..., a_n) \in \mathbb{Z}_+^n, (b_1, b_2,...,b_m) \in \mathbb{Z}_+^m \, (P(a_1,a_2,...a_n) \downarrow (b_1,b_2,...,b_m)) \Leftrightarrow$$

$$\Leftrightarrow (a_1, a_2, ..., a_n) \in \mathrm{Dom}\, f \,\&\, f(a_1, ..., a_n) = (b_1, ..., b_m).$$

В частности это означает, что $P(a_1,...,a_n)\downarrow \Leftrightarrow (a_1,a_2,...,a_n) \in \mathrm{Dom}\, f$. Функция f: $\mathbb{Z}_+^n \to \mathbb{Z}_+^m$ называется МНР-*вычислимой*, если существует программа для МНР, вычисляющая f.

Пример. Функция $x + y$ МНР-вычислима. Пусть начальная конфигурация МНР есть

R_1	R_2	R_3	R_4	...
x	y	0	0	0

Тогда следующая программа

$J(3, 2, 5)$

$S(1)$

$S(3)$

$J(1, 1, 1)$

вычисляет $x + y$. Легко увидеть, что эта программа прибавляет к $r_1 = x$ единицу до тех пор, пока содержимое регистра R_3, используемое как счетчик, не станет равным y. Заметим, что команда $J(1, 1, 1)$ здесь всегда возвращает нас к выполнению первой команды, а останавливается программа только в результате выполнения команды $J(3, 2, 5)$ в тот момент, когда $r_3 = r_2$.

Каждая конкретная МНР-вычислимая функция может быть вычислена различными программами.

2.3.3. Разрешимые предикаты

Многие математические проблемы формулируются как вопросы о том, обладают ли некоторые наборы чисел заданными свойствами. Алгоритм для решения такой задачи должен являться процедурой,

дающей ответ "да" или "нет", в зависимости от заданного набора чисел, то есть предикатом. Понятие характеристической функции позволяют свести вопрос к вычислению некоторой функции. Пусть $Q(x_1, x_2, ..., x_n)$ — n-местный предикат на натуральных числах. *Характеристической функцией* этого предикта $\chi_Q(x_1, ..., x_n)$ называется функция

$$\chi_Q(x_1, ..., x_n) \textbf{ := if } Q(x_1, ..., x_n) \textbf{ then } 1 \textbf{ else } 0 \textbf{ end if}.$$

Предикат $M(x_1, ..., x_n)$ *разрешим*, если его характеристическая функция $\chi_M(x_1, ..., x_n)$ — МНР-вычислима.

Замечание. Говоря о разрешимости, мы имеем дело с всюду определенными функциями.

Сопоставив несложные МНР-программы, можно показать, что следующие предикаты разрешимы:

a) $<x \neq y>$

b) $<x = 0>$

c) $<x = y^2>$

Свойства или предикаты мы иногда будем называть *проблемами*. Примеры показывают, что существуют разрешимые проблемы. Далее мы увидим, что существуют и неразрешимые проблемы.

2.3.4. Вычислимость в других областях

Наше определение вычислимости и разрешимости дано для функций и предикатов от натуральных чисел. Эти понятия распространяются на другие виды объектов с помощью кодирования натуральными числами. *Кодированием области D* называется явное и эффективное отображение α: $D \to \mathbb{N}$. При этом говорят, что объект d кодируется натуральным числом $\alpha(d)$. Если теперь задана функция f: $D \to D$, то f естественно кодируется функцией f^*: $\mathbb{N} \to \mathbb{N}$, которая код объекта $d \in \mathrm{Dom}\, f$ отображает в код объекта $f(d)$. Это можно записать как $f^* = \alpha^{-1} \circ f \circ \alpha$. Если задано кодирование области D, то опре-

деление МНР вычислимости переносится на функции из D в D. Функция $f\colon D \to D$ — МНР-вычислима, если f^* — МНР-вычислима, как функция натуральных чисел.

Пример. Явное кодирование положительных элементов поля рациональных чисел $\mathbb{Q}$ можно задать следующей формулой

$$\alpha(p/q) = (p + q + 1)(p + q)/2 + q.$$

Приведенное определение приводит сразу и к определению разрешимости предиката на области D. Предикат Q на области D называется *разрешимым*, если разрешим предикат $Q^*(d) = Q(\alpha(d))$.

2.4. ОПЕРАТОР МИНИМИЗАЦИИ

Введенные ранее операции над частичными функциями такие, как композиция, соединение или рекурсия, будучи примененными к всюду определенным функциям, снова дают всюду определенные функции. В то же время любой программист знает, как нетрудно составить программу, которая для некоторых начальных данных заканчивает работу и вырабатывает некоторое значение, а для других данных работает бесконечно. Это означает, что реальные вычисления приводят к частично определенным функциям. Здесь мы вводим еще одну операцию, применение которой существенно расширяет класс получающихся функций.

2.4.1. Определения

Рассмотрим операцию, которую будем называть *оператором* μ (или *оператором минимизации*). Она ставит в соответствие частичной функции $f\colon \mathbb{Z}_+^{n+1} \to \mathbb{Z}_+$ частичную функцию $h\colon \mathbb{Z}_+^{n} \to \mathbb{Z}_+$, которая определяется следующим образом.

$$\mathrm{Dom}\ h := \{(x_1,\ldots,x_n) \mid \exists y \geq 1(f(x_1, \ldots x_n, y) = 1\ \&\ \forall k \leq$$
$$\leq y\ (x_1,x_2,\ldots x_n,k) \in \mathrm{Dom}\ f\}$$
$$h(x_1 \ldots x_n) = \min\ \{y \mid f(x_1, x_2 \ldots x_n, y) = 1\}$$

Эта функция записывается как μy ($f(x_1 \ldots x_n, y) = 1$). Эту запись следует читать так "минимальное y такое, что $f(x_1 \ldots x_n, y) = 1$".

Замечание. Следует обратить внимание на то, что мы требуем, чтобы значения $f(x_1 \ldots x_m, k)$ были бы определены для всех $k \leq y$.

2.4.2. Частично рекурсивные функции

Последовательность частичных функций $f_1, f_2, \ldots, f_n$ называется *частично-рекурсивным описанием функции* f, если $f = f_n$, f_1 — одна из простейших функций; f_i для всех $i \geq 2$ либо является простейшей функцией, либо получается применением элементарных операций композиции, соединения, рекурсии или минимизации к некоторым из функций $f_1, \ldots, f_{i-1}$. Если оператор μ не используется, то такая последовательность называется *примитивно-рекурсивным описанием функции* f. Функция f называется *частично-рекурсивной*, если она допускает частично-рекурсивное описание. Если функция допускает примитивно-рекурсивное описание, то она называется *примитивно-рекурсивной*. Запас частично-рекурсивных функций необычайно широк. В частности, любая МНР-вычислимая функция допускает частично-рекурсивное описание.

2.4.3. МНР-вычислимость частично-рекурсивных функций

Легко видеть, что все простейшие функции — единичная функция, прибавление единицы и функция i-ой проекции МНР-вычислимы. Им соответствуют следующие программы

a) 1: $Z(1)$, $S(1)$.

b) Suc : $S(1)$.

c) pr_i^u : $T(i, 1)$.

Сейчас мы рассмотрим методы, позволяющие строить новые программы из уже имеющихся программ. Допустим, что мы имеем две программы P и Q и хотим выполнить их последовательно. Если мы просто объединим списки команд P и Q в один список, то можем встретиться с такой ситуацией, когда вычисление по программе $P = I_1, I_2, \ldots, I_s$ завершается, когда следующая команда в вычислении

имеет номер v, где $v > s$. Нам в этом случае надо перейти к первой команде Q. Это произойдет только, если $v = s + 1$. Поэтому мы сейчас ограничимся только программами I_1, I_2... I_s, которые останавливаются только тогда, когда следующей командой является I_{s+1}. Назовем программу $P = I_1, I_2, ..., I_s$ *стандартной*, если любая команда условного перехода $J(m, n, q)$ в P удовлетворяет условияю $q \le s + 1$.

Лемма. *Для каждой программы P существует стандартная программа* P^*, *такая, что всякое вычисление по программе P идентично соответствующему вычислению по программе* P^* *за исключением способа останова. В частности для любых* $a_1, ..., a_n$ и $b_1, ..., b_m$

$$P(a_1, a_2, ..., a_n)\downarrow(b_1, b_2, ..., b_m) \Leftrightarrow P^*(a_1, a_2, ..., a_n)\downarrow(b_1, b_2, ..., b_m).$$

Доказательство. Мы просто изменим все команды условного перехода $J(m, n, q)$ в программе P так, чтобы все остановы происходили только при переходе к команде I_{s+1}.

$$I_k^* := \textbf{if } q \le s+1 \textbf{ then } I_k \textbf{ else } J(m, n, s + 1) \textbf{ end if}. \ \square$$

Пусть теперь P и Q имеют стандартный вид и имеют длины s и t соответственно. *Соединением* или *конкатенацией* программ P и Q называется программа $I_1, I_2, ..., I_s, I_{s+1}, ..., I_{s+t}$, где $P = I_1 I_2... I_s$, а команды $I_{s+1}, ..., I_{s+t}$ суть команды программы Q, у которых каждый условный переход $J(m, n, q)$ заменен на $J(m, n, s + q)$. Ясно, что при таком определении результат действия PQ совпадет с соответствующим вычислением по программе P, за кокоторым следует вычисление по программе Q, начатое в заключительной конфигурации программы P. Нетрудно заметить теперь, что операция соединения программ в точности соответствует операции соединения частично рекурсивных функций. То есть, если P вычисляет функцию f_1, а Q вычисляет функцию f_2, то PQ вычисляет функцию (f_1, f_2). Оказывается для остальных элементарных операций над частично рекурсивными функциями можно также определить соответствующие операции над МНР-программами. Это приводит нас к следующей теореме.

Теорема. *Всякая частично-рекурсивная функция* f: $\mathbb{Z}_+^n \to \mathbb{Z}_+^m$ *является* МНР-*вычислимой*.

Доказательство. Без доказательства. □

Замечание. Доазательство этой теоремы состоит в проверке того, что класс частично рекурсивных функций содержит простейшие функции и замкнут относительно соединения, композиции, примитивной рекурсии и оператора минимизации.

2.4.4. Тезис Чёрча

В предыдущих разделах мы ввели класс частично рекурсивных функций и показали, что любая такая функция является МНР-вычислимой. В течение предыдущего столетия в математической логике и теории вычислимости были предложены многие другие модели вычислений, отличные от вычислений с помощью МНР. Всегда оказывалось, что если модель была достаточно мощной, то класс функций, который мог быть вычислен с помощью данной модели вычислений, совпадал с классом частично-рекурсивных функций. Это говорит о том, что наше "наивное" определение понятий вычислимой и полувычислимой функции *не зависит* от конкретизации слова "программа", под которой можно понимать любую эффективную вычислительную процедуру.

Тезис Чёрча: Интуитивно и неформально определенный класс эффективно вычислимых функций совпадает с классом частично-рекурсивных функций (или, что то же самое, с классом МНР-вычислимых функций).

Тезис Чёрча надо воспринимать как экспериментальный факт, в пользу которого есть целый ряд сильных аргументов. Существует огромный набор рекурсивных описаний различных полувычислимых функций. Каждый раз, когда кто-нибудь пытался найти частично-рекурсивное описание какой-нибудь неформально эффективно полувычислимой функции, такое описание обязательно находилось.

Все другие схемы детерминированной переработки информации при применении какой-либо эффективной процедуры кодирования "входов" и "выходов" натуральными числами приводили к тому же самому классу частично-рекурсивных функций.

Такие схемы задолго до появления МНР модели предлагались Тьюрингом, Постом, Марковым и другими математиками.

2.5. ВЫЧИСЛИМОСТЬ ПО ТЬЮРИНГУ

Здесь мы рассмотрим другой подход к вычислимости, предложенный в 1926 г. А. Тьюрингом. Процесс вычисления с помощью рассмотренной нами МНР близок к процессу вычислений на современных компьютерах, в то время как подход Тьюринга основан на анализе процесса вычислений, осуществляемых человеком при помощи карандаша и бумаги. Если разложить процесс осуществления вычислительного алгоритма человеком на элементарные шаги, то можно такой процесс представить как последовательность следующих действий.

a) Запись или стирание символа.

b) Переход к обозреванию нового символа.

c) Принятие решения о том, что делать на следующем шаге.

Абстрактная машина Тьюринга моделирует такой процесс.

2.5.1. Определение машины Тьюринга

Машина Тьюринга представляет собой устройство, которое производит операции на бесконечной в обе стороны ленте, разделенной на ячейки. В каждой ячейке может поместиться один символ. Машина снабжена читающей головкой, которая в каждый момент обозревает ровно один символ. Пустой символ обозначим ε. Будем считать, что символы принадлежат конечному алфавиту T. Машина может выполнить три типа операций.

1) Стереть обозреваемый символ и заменить его другим символом алфавита T.

2) Сдвинуть читающую головку вправо на 1 символ.

3) Сдвинуть читающую головку влево на 1 символ.

В каждый момент машина находится в одном из состояний q_1, q_2, ..., q_m, количество которых фиксировано. Каждая операция может изменить состояние машины M. Действие, которое машина M совершает, зависит от текущего состояния и символа, обозреваемого в настоящий момент. Формально машину Тьюринга M можно задать как семерку $(Q, T, I, \delta, \varepsilon, q_0, q_t)$, где

Q — множество состояний.

T — множество символов на ленте.

I — множество входных символов $I \subseteq T$.

ε — пустой символ.

q_0 — начальное состояние.

q_t — заключительное состояние.

δ — функция перехода, δ: $Q \times T \to Q \times T \times \{L, R, S\}$.

Эта функция по состоянию машины и читаемому символу выдаст новое состояние, записываемый символ и направление сдвига головки: L — влево, R — вправо, S — оставить на месте.

Замечание. Существуют варианты этого определения, например, многоленточные машины Тьюринга. Наша машина будет снабжена только одной лентой.

2.5.2. Языки, распознаваемые машиной Тьюринга

Так как машина Тьюринга работает с символами некоторого конечного алфавита — она может распознавать языки. Алфавит рассматриваемого языка играет роль входных символов. Вначале на ленте записано слово из входных символов. Все клетки справа от клеток, занимаемых входными словами — пусты. Слово из входных символов *допускается* машиной Тьюринга M, если машина, начав работу в начальном состоянии, сделает последовательность шагов, приводящую к конечному состоянию. Языком, допускаемым машиной Тьюринга, называется множество всех слов из входных символов, допускаемых машиной.

2.5.3. Универсальная машина Тьюринга

Под универсальной машиной Тьюринга понимают машину Тьюринга, которая может выполнить действия, производимые любой другой машиной Тьюринга. Алгоритм, выполняемый конкретной машиной Тьюринга, может быть закодирован с помощью символов некоторого алфавита (не обязательно совпадающего с алфавитом, использующимся для записи данных). Тогда такой код машины может быть использован, как данные для универсальной машины. Получив на вход код некоторой машины Тьюринга и входные данные для нее же, универсалльная машина вычисляет ответ, который вычислила бы по входным данным машина Тьюринга, чья программа была дана на вход.

Дадим теперь формальное определение универсальной машины.

Конечный алфавит, использующийся для кодирования машин Тьюринга, обозначим А. Тогда универсальной машиной Тьюринга U для класса машин с алфавитом В и k входными лентами называется машина Тьюринга с $k + 1$ входной лентой и алфавитом $A \cup B$ такая, что если подать на первые k лент входные данные, а на $k + 1$ —код некоторой машины Тьюринга M_1, то U выдаст тот же ответ, какой выдала бы на этих входных данных M_1, или же будет работать бесконечно долго, если M_1 на этих данных не останавливается.

Можно доказать, что такая машина существует и моделирует все другие машины и даже оценить ее сложность. Оказывается, что существует константа c, такая, что если исходная машина произвела t шагов до получения результата, то универсальная машина получит этот же результат не более, чем за ct^2 шагов. Доказательство у этой теоремы конструктивное. Такая машина строится явно. Теорема о существовании универсальной машины была предложена и доказана Тьюрингом в 1947 г.

Идея самой универсальной машины близка к идее современного компилятора. По сути дела любая универсальная машина Тьюринга и является компилятором, для машины, способной производить вычис-

ления по любому записанному на ленте алгоритму. Интересно, что как сама теорема Тьюринга, так и явная конструкция универсальной машины появились задолго до эры электронно-вычислительных машин и языков программирования высокого уровня.

2.5.4. Меры сложности алгоритмов

Машины Тьюринга удобны для анализа временной и пространственной сложности алгоритмов. *Временная сложность* $T(h)$ машины Тьюринга M равна наибольшему числу шагов, сделанных ею при обработке входа длины n. Если машина не останавливается хотя бы для одного из входов длины n, то для этого n значение $T(n)$ не определено. *Пространственная* (*ёмкостная*) сложность $S(n)$ машины Тьюринга M равна наибольшему расстоянию от крайнего левого рассмотренного символа до крайнего правого рассмотренного символа для всех входов длины n, если это число определено.

2.5.5. Функции, вычислимые по Тьюрингу

А. Тьюрингом было установлено, что если рассматривать машины Тьюринга для вычислений функций $f\colon \mathbb{Z}_+ \to \mathbb{Z}_+$ используя простейший алфавит {0, 1} для записи натуральных чисел на ленту, то *вычислимыми по Тьюрингу* функциями также окажутся частично-рекурсивные функции. Чтобы машина M могла вычислять числовые функции, надо договориться о представлении чисел на ленте. Самый простой способ использует единственный входной символ 1 и кодирует число x как x единиц, идущих подряд.

Частичную функцию, вычисляемую машиной Тьюринга M можно определить следующим образом: $f(x)$ — это количество вхождений символа 1 на ленте в заключительный момент, если вычисление останавливается.

Частичная функция называется *вычислимой по Тьюрингу*, если существует вычисляющая ее машина Тьюринга. Следующая теорема устанавливает связь между вычислимостью по Тьюрингу, МНР-вычислимостью и частично рекурсивными функциями.

Теорема. *Класс функций вычислимых по Тьюрингу $\mathcal{T}$ совпадает с классом $\mathcal{R}$ — частично-рекурсивных функций и с классом $\mathcal{G}$ —* МНР-*вычислимых функций*.

Доказательство. Без доказательства. □

Замечание. В отличие от тезиса Чёрча, все классы функций в этой теореме определены строго.

2.5.6. Функция Аккермана

Можно было бы подумать, что всюду определенные частично-рекурсивные функции примитивно рекурсивны. Это означало бы, что оператор минимизации нужен только для того, чтобы расширить класс примитивно-рекурсивных функций за счет добавления не всюду определенных функций. Однако это не так. Существуют всюду определенные частично-рекурсивные функции, не являющиеся примитивно-рекурсивными. Определим функцию $\psi(x, y)$ следующей системой уравнений:

$$\psi(1, y) = y + 1,$$

$$\psi(x + 1, 1) = \psi(x, 2),$$

$$\psi(x + 1, y + 1) = \psi(x, \psi(x + 1, y)).$$

Вычислимость функции ψ в неформальном смысле следует из того, что значения $\psi(x, y)$ $(x > 1)$ определены в терминах "более ранних" значений $\psi(x_1, y_1)$ с $x_1 < x$ или $x_1 = x$, но $y_1 < y$. Индукцией по x и y показывается, что для определения $\psi(x, y)$ необходимо знать только конечное число таких более ранних значений. Например, $\psi(2, 2) = 3$, а $\psi(3, 2) = 5$. Можно показать, однако, что функция $\psi(x, y)$ растет быстрее любой примитивно-рекурсивной функции. Эта функция называется *функцией Аккермана*.

2.6. ПЕРЕЧИСЛИМЫЕ И РАЗРЕШИМЫЕ МНОЖЕСТВА

2.6.1. Перечислимые множества

Множество $E \subseteq \mathbb{Z}_+^n$ называется *перечислимым*, если существует такая частично-рекурсивная функция f, что $E = \mathrm{Dom}\, f$. Интуитивный смысл перечислимости таков: существует алгоритм, распознающий элементы x, принадлежащие E, возможно не дающий никакого ответа для x, не принадлежащих E.

Теорема 1. *Следующие два класса множеств совпадают*

a) *Перечислимые множества*.

b) *Множества* 1-*уровня частично-рекурсивных функций*.

Доказательство.

[$a \to b$] Пусть E перечислимо, $E = \mathrm{Dom}\ f$ тогда E равно 1-уровень функции $1^{(1)} \cdot f$ (соединение единичной функции и f).

[$b \to a$] Пусть E — 1-уровень частично-рекурсивной функции $f(x_1 \ldots x_n)$. Положим $g(x_1, x_2, \ldots, x_n) = \min \{y \mid (f(x_1 \ldots x_n) - 1)^2 + y = 1\}$. Очевидно, что g — частично рекурсивна и $E = \mathrm{Dom}\, f$. □

Оказывается, что верно более сильное утверждение.

Теорема 2. *Класс перечислимых множеств совпадает с проекциями множеств уровня примитивно рекурсивных функций*.

Доказательство. Без доказательства.

Отсюда непосредственно следует важное для программистов утверждение.

Теорема 3. *Если E перечислимо, то существует программа, порождающая элементы из E*.

Доказательство. Пусть E — проекция на первой n-координат 1-уровня примитивно рекурсивной функции $f(x_1\ x_2 \ldots x_n, y)$. Порождающая E программа перебирает векторы $(x_1 \ldots x_n, y)$, вычисляет и подает на выход $(x_1 \ldots x_n)$ в том и только в том случае, если $f = 1$. Такая программа выписывает все элементы $(x_1 \ldots x_n)$ принадлежащие E, так как функция f примитивно рекурсивна, а значит, всюду определена.

Теорема 4. *Класс перечислимых множеств замкнут относительно следующих операций: конечное прямое произведение, конечное пересечение, конечное объединение, проекция*.

Доказательство. Без доказательства. □

2.6.2. Разрешимые множества

Множество $E \subseteq \mathbb{Z}_+^n$ называется *разрешимым*, если оно и его дополнение перечислимы. Оказывается, существуют перечислимые, но не разрешимые множества.

Пример.

2.6.3. Диофантовы множества

Выше было объяснено, что перечислимые множества суть проекции множеств уровня примитивно рекурсивных функций. Можно поставить вопрос о том, нельзя ли здесь ограничиться еще более узким классом функций. Ответом на этот вопрос явилось решение Ю. В. Матиясевичем знаменитой 10-ой проблемы Гильберта. Оказывается, что перечислимые множества являются проекциями множеств уровня полиномов с целыми коэффициентами. Множество $E \subseteq \mathbb{Z}_+^n$ называется *диофантовым*, если существует полином $P(y_1, y_2, ..., y_n)$ такой, что $(x_1, x_2... x_n) \in E \Leftrightarrow \exists\, y_1\, y_2... y_n,\ P(x_1\, x_2... x_n, y_1... y_n) = 1$. С античных времен уравнения в целых числах называются *диофантовыми*. Таким образом, диофантово множество это проекция множества решений некоторого диофантова уравнения на первые n координат.

Теорема. [Матиясевича] *Перечислимые множества диофантовы*.

Доказательство. Без доказательства. □

Замечание. Эта теорема явилась последним шагом в доказательстве 10-ой проблемы Гильберта, которая была сформулирована им на II Международном конгрессе математиков, проходившем в Париже в августе 1900 года в числе 23 проблем, во многом определивших развитие математики на столетие вперед. 10-ая проблема была

сформулирована так: "Пусть задано диофантово уравнение с произвольными неизвестными и целыми числовыми коэффициентами. Указать способ, при помощи которого возможно после конечного числа операций установить разрешимо ли это уравнение в целых числах".

Отрицательное решение 10-ой проблемы следует из вышеприведенной теоремы Матиясевича и из существования перечислимого, но не разрешимого множества. Действительно, пусть $E \subseteq \mathbb{Z}_+$ перечислимое, но не разрешимое множество. Так как E — диофантово, то

$$t \in E \Leftrightarrow \exists\, x_1\, x_2 \ldots\, x_n\, f(t, x_1, \ldots, x_n) = 0$$

для некоторого полинома f. Так как E не разрешимо, то отсюда следует, что не существует алгоритма, определяющего имеет ли для данного t уравнение $f(t,\ x_1 \ldots\ x_n) = 0$ какое-либо решение в целых числах $x_1, \ldots, x_n$. Отметим здесь, что конструкция диофантова представления любого перечислимого множества E абсолютно эффективна. По заданному рекурсивному описанию функции f с $\mathrm{Dom}\, f = E$ либо функции g с $g(\mathbb{Z}_+) = E$ соответствующий многочлен может быть выписан явно.

2.6.4. Полиномиальное представление множеств

Оказывается, что любое диофантово множество $E \subseteq \mathbb{Z}_+$ является множеством положительных значений некоторого полинома.

Теорема. *Пусть $E \subseteq \mathbb{Z}_+$ диофантово множество. Тогда существует такой многочлен $q \in Z(x_0, \ldots, x_n)$, что E совпадает с множеством положительных значений g в точках из $\mathbb{Z}_+^{n+1}$.*

Доказательство. Пусть $E \subseteq \mathbb{Z}_+ = \{x_0 \mid \exists\, x_1, \ldots, x_n\ (f(x_0, x_1 \ldots\ x_n)=0)\}$. Положим $g = x_0\,(1 - f^2(x_0, x_1, \ldots, x_n))$. Ясно, что $g > 0$ только тогда, когда $f(x_0, x_1 \ldots\ x_n) = 0$. □

Этот факт, вообще говоря, просто поразителен. Например, множество всех факториалов или множество всех простых чисел можно представить одним полиномом.

Пример.

1) Множество простых чисел — это в точности множество положительных значений следующего полинома степени 25 от 26 переменных.

$(k + 2)(1\ (\mathrm{wz} + h + j - q)^2 - ((gk + 2g + k + 1)(h + j) + h - z)^2 -$
$- (2n + p + q + z - e)^2 - (16(k + 1)\text{^}3(k + 2)(n + 1)^2 + 1 - f^2)^2 -$
$- (e\text{^}3(e + 2)(a + 1)^2 + 1 - 0^2)^2 - ((a^2 - 1)y^2 + 1 - x^2)^2 - (16\ r^2\ y\text{^}4(a^2 - 1) +$
$+ 1 - u^2)^2 - (((a + u^2(u^2 - a))^2 - 1)(n + 4dy)^2 + 1 - (x + cu)^2)^2 -$
$- (n + e + v + y)^2 - ((a^2 - 1)e^2 + 1 - m^2)^2 - (ai + le + 1 - l - i)^2 -$
$- (p + \mathrm{l}(a - n - 1) + b\ (2\mathrm{a}n + 2\mathrm{a} - n^2 - 2n - 2) - m)^2 -$
$- (q + y(a - p - 1) + s(2ap + 2a - p^2 - 2p - 2) - x)^2 - (z + pe(a - p) +$
$+ t(2pa - p^2 - 1) - pm)^2).$

2) Для чисел Фибоначчи можно указать совсем простой многочлен пятой степени от двух переменных

$$2a^4 b + a^3 b^2 - 2a^2 b^3 - a^5 - ab^4 + 2a.$$

Эта формула в качестве положительных значений дает только числа Фибоначчи u_n: $u_0 = u_1 = 1$, $u_{n+1} = u_n + u_{n-1}$, 1, 1, 3, 5, 8, 13, 21, 34....

2.7. МЕРЫ СЛОЖНОСТИ

Мы уже ввели для машины Тьюринга (МТ) понятия временной и емкостной сложности $T(n)$ и $S(n)$. Данные функции от длины входа описывают время вычислений и требуемую память только для данной машины Тьюринга. В принципе может оказаться, что другие модели, например, МНР или многоленточные МТ для той же задачи требуют меньше памяти или же работают гораздо быстрее. Оказывается, что на самом деле в некотором вполне определенном смысле, эти модели вычислений эквивалентны. Если для какой-то задачи время $T(n)$ для любой машины Тьюринга растет как экспонента от n, то и любая

МНР не сможет выполнить такую задачу быстрее, чем за экспоненциальное время, возможно, правда, с другим показателем. В этом разделе мы приведем аргументы в пользу утверждения, что некоторый класс задач содержит только задачи, не решающиеся за полиномиальное время на МТ или МНР.

2.7.1. Недетерминированные вычисления

Определение недетерминированной МТ (или НМТ) буквально повторяет определение МТ за исключением того, что функция переходов δ представляет собой отображение множества $Q\times T$ в множество подмножеств $Q\times(T\times \{L, R, S\})$. Иными словами функция δ выдает конечное количество вариантов для каждого следующего шага. Процесс вычисления надо представлять себе так, что на каждом шаге все варианты далее проверяются параллельно. Ясно, что в реальности такое распараллеливание работы на каждом шаге невозможно. Тем не менее НМТ является очень удобным средством описания временных и ёмкостных характеристик алгоритмов. Говорят, что НМТ M имеет \odmkey{временную}{Сложность!временная} сложность $T(n)$, если для всякой допускаемой входной цепочки длины n найдется последовательность, состоящая не более, чем из $T(n)$ шагов, которая приводит в конечное состояние. Говорят, что НМТ M имеет \odmkey{ёмкостную}{Сложность!ёмкостная} сложность $S(n)$, если для всякой допускаемой входной цепочки длины n найдется последовательность шагов, приводящая к конечному состоянию, в котором число просмотренных головкой ячеек не превосходит $S(n)$.

2.7.2. Детерминированное моделирование НМТ

2.7.3. Классы $\mathcal{P}$ и $\mathcal{NP}$

Определение $\mathcal{P}$-time, как множества языков, допускаемых ДМТ с полиномиальной временной сложностью и $\mathcal{NP}$-time, как множества языков, допускаемых НМТ с полиномиальной временной сложно-

стью. Мы будем писать просто $\mathcal{P}$ и $\mathcal{NP}$ там, где ясно, что речь идет о временных характеристиках. В настоящее время неизвестно, является ли $\mathcal{P}$ собственным подклассом класса $\mathcal{NP}$. Однако можно доказать, что некоторые языки не менее "трудны", чем любой язык из класса $\mathcal{NP}$. Такие языки (и, соответственно, задачи распознавания) называются $\mathcal{NP}$-полными.

ВЫВОДЫ

Интуитивное понятие алгоритма может быть определно формально, и различные способы определения алгоритма дют один и тот же результат.

3. АЛГЕБРАИЧЕСКИЕ СТРУКТУРЫ

Алгебраические методы описания моделей находят самое широкое применение при формализации различных предметных областей. Грубо говоря, при построении модели предметной области все начинается с введения подходящих обозначений для операций и отношений с последующим исследованием их свойств. Владение алгебраической терминологией, таким образом, входит в арсенал средств, необходимых для абстрактного моделирования, предшествующего практическому программированию задач конкретной предметной области. Материал этой главы, помимо введения в терминологию общей алгебры, содержит некоторое количество примеров конкретных алгебраических структур. Вначале рассматриваются классические структуры, которые обычно включатся в курсы общей алгебры (в том числе описывается язык абстрактной теории категорий), а затем обсуждаются некоторые более специальные структуры, наиболее часто применяемые в конкретных программных построениях.

3.1. ОПЕРАЦИИ И АЛГЕБРЫ

Словом "алгебра" обозначают, вообще говоря, не только раздел математики, но и один из конкретных объектов, изучаемых в этом разделе. К счастью, "алгебры" в узком смысле здесь не рассматриваются, а потому для краткости и без риска возникновения недоразумений слово "алгебра" используется как родовое понятие для обозначения разнообразных алгебраических структур.

3.1.1. Алгебраические структуры

Всюду определенная (тотальная) функция $\varphi: M^n \to M$ называется n-арной (n-местной) операцией на M. Если операция φ — бинарная (то есть $\varphi: M \times M \to M$), то будем писать $a\varphi b$ вместо $\varphi(a, b)$ или $a*b$, где * — знак операции.

Замечание. Такая форма записи называется инфиксной.

Множество M вместе с набором операций $\Sigma = \{\varphi_1, ..., \varphi_m\}$, $\varphi_i : M^{n_i} \to M$, где n_i — арность операции φ_i, называется *алгебраической структурой*, *универсальной алгеброй* или просто *алгеброй*. Множество M называется *основным* (*несущим*) множеством или *основой* (*носителем*; вектор арностей $(n_1, ..., n_m)$ называется *типом*; множество операций Σ называется *сигнатурой*. Запись: $\langle M; \Sigma\rangle$ или $\langle M; \varphi_1, ..., \varphi_m\rangle$. Операции φ_i *конечноместны*, сигнатура Σ конечна. Носитель не обязательно конечен, но не пуст.

Замечание. Далее для обозначения алгебры везде, где это возможно, используется прописная рукописная буква, а для обозначения ее носителя — соотвествующая обычная прописная буква: $\mathcal{A} = \langle A; \Sigma\rangle$. Такое соглашение позволяет использовать вольности в обозначениях, не вводя явно две буквы для алгебры или для носителя, а подразумевая одну из них по умолчанию. Например, выражение "алгебра A" означает алгебру $\mathcal{A}$ с носителем A и сигнатурой, которая ясна из контекста, а запись $a \in \mathcal{A}$ означает элемент a из носителя A алгебры $\mathcal{A}$.

Если в качестве φ_i допускаются не только функции, но и отношения, то множество M вместе с набором операций и отношений называется *моделью*. В приложениях обычно используется следующее обобщение понятия алгебры. Пусть $M = \{M_1, ..., M_n\}$ — множество основ, $\Sigma = \{\varphi_1, ..., \varphi_m\}$ — сигнатура, причем φ_i: $M_{i_1} \times ... \times M_{in_i} \to M_j$. Тогда $\langle M; \Sigma\rangle$ называется *многоосновной* алгеброй. Другими словами, многоосновная алгебра имеет несколько носителей, а операция сигнатуры действует из прямого произведения некоторых носителей в некоторый носитель.

3.1.2. Замыкания и подалгебры

Подмножество $X \subseteq M$ называется *замкнутым* относительно операции φ, если

$\forall x_1, ..., x_n \in X\, (\varphi(x_1, ..., x_n) \in X)$.

Если X замкнуто относительно всех $\varphi \in \Sigma$, то $\langle X; \Sigma_X \rangle$ называется *подалгеброй* $\langle M; \Sigma \rangle$, где $\Sigma_X = \{\varphi^X{}_i\}$, $\{\varphi^X{}_i\} = \varphi_i|_{X_k}$, $k = n_i$.

Примеры.

1) Алгебра $\langle \mathbb{R}; +, \cdot \rangle$ — *поле действительных чисел*. Тип — (2, 2). Все конечные подмножества, кроме $\{0\}$, не замкнуты относительно сложения и все конечные подмножества, кроме $\{0\}$ и $\{0, 1\}$, не замкнуты относительно умножения. *Поле рациональных чисел* $\langle \mathbb{Q}; +, \cdot \rangle$ образует подалгебру.

2) Алгебра $\langle 2^M; \cup, \cap, \bar{} \rangle$ — *алгебра подмножеств* над множеством M. Тип — (2, 2, 1). $\forall X \subseteq M$ ($\langle \{2^X; \cup, \cap, \bar{} \rangle$) — подалгебра.

3) Алгебра гладких функций $\langle \{f \mid f : \mathbb{R} \to \mathbb{R}\}; d/dx \rangle$, где d/dx — операция дифференцирования. Множество полиномов одной переменной x образует подалгебру, которая обозначается $\mathbb{R}[x]$.

Теорема. *Непустое пересечение подалгебр образует подалгебру этой же алгебры*.

Доказательство. Пусть $\langle X_i; \Sigma_{X_i} \rangle$ — подалгебра $\langle M; \Sigma \rangle$. Обозначим $X := \cap X_i$. Тогда

$$\forall i \, (\forall j \, (\varphi^{Xi}{}_j(x_1, ..., x_{n_j}) \in X_i \Rightarrow \forall j \, (\varphi^{Xi}{}_j(x_1, ..., x_{n_j}) \in \quad X_i)$$

Замыканием множества $X \subseteq M$ относительно сигнатуры Σ (обозначается $[X]_\Sigma$) называется множество всех элементов (включая сами элементы X), которые можно получить из X, применяя операции из Σ. Если сигнатура подразумевается, ее можно не указывать.

Пример. В алгебре $\langle \mathbb{Z}; +, \cdot \rangle$ замыканием числа 2 являются четные числа, то есть $[\{2\}] = \{n \in \mathbb{Z} \mid n = 2k \,\&\, k \in \mathbb{Z}\}$.

Свойства замыкания:

1) $X \subseteq Y \Rightarrow [X] \subseteq [Y]$;

2) $X \subseteq [X]$;

3) $[[X]] = [X]$;

4) $[X] \cup [Y] \subseteq [X \cup Y]$.

Пусть $\mathcal{A} = \langle A; \Sigma \rangle$ — некоторая алгебра и $X_1, ..., X_n \subseteq A$ — некоторые подмножества носителя, а $\varphi \in \Sigma$ — одна из операций алгебры. Тогда используется следующее соглашение об обозначениях:

$$\varphi(X_1, ..., X_n) := \{\varphi(x_1, ..., x_n) \mid x_1 \in X_1 \,\&\, ... \,\&\, x_n \in X_n\},$$

то есть алгебраические операции можно применять не только к отдельным элементам, но и к множествам (подмножествам носителя), получая, соответственно, не отдельные элементы, а множества (подмножества носителя).

3.1.3. Системы образующих

Пусть заданы набор функциональных букв $\Phi = \{\varphi_1, ...,\varphi_m\}$, служащих обозначениями функций некоторой сигнатуры Σ типа $N = (n_1, ..., n_m)$, и множество переменных $V = \{x_1, x_2, ...\}$. Определим множество *термов* T следующим образом:

1) $V \subseteq T$;

2) $t_1, \ldots, t_{n_i} \in T \,\&\, \varphi_i \in \Phi \Rightarrow \varphi_i () t_1, \ldots, t_{n_i}) \in T$.

Алгебра $\langle T; \Phi \rangle$ называется *свободной алгеброй термов*. Носителем этой алгебры является множество термов, то есть формальных выражений, построенных с помощью знаков операций сигнатуры Σ.

Замечание. Алгебры термов используются в программировании для определения абстрактных типов данных.

Множество $M' \subseteq M$ называется *системой образующих* алгебры $\langle M; \Sigma \rangle$, если $[M']_\Sigma = M$. Если алгебра имеет конечную систему образующих, то она называется *конечно-порожденной*. Бесконечные алгебры могут иметь конечные системы образующих.

Пример. Алгебра *натуральных чисел* — $\langle \mathbb{N}; + \rangle$ — имеет конечную систему образующих $1 \in \mathbb{N}$.

3.1.4. Свойства операций

Некоторые часто встречающиеся свойства операций имеют специальные названия. Пусть задана алгебра $\langle M; \sum \rangle$ и a, b, $c \in M$; $*, \lozenge \in \sum$; $*, \lozenge : M \times M \to M$. Тогда:

1) *Ассоциативность*: $(a * b) * c = a * (b * c)$.

2) *Коммутативность*: $a * b = b * a$.

3) *Дистрибутивность* $\lozenge$ *относительно* $*$ *слева*: $a \lozenge (b * c) = (a \lozenge b) * (a \lozenge c)$.

4) *Дистрибутивность* $\lozenge$ *относительно* $*$ *справа*: $(a * b) \lozenge c = (a \lozenge c) * (a \lozenge b)$.

5) *Поглощение* ($\lozenge$ поглощает $*$): $(a * b) \lozenge a = a$.

6) *Идемпотентность*: $a * a = a$.

Примеры.

1) Ассоциативные операции: сложение и умножение чисел, объединение и пересечение множеств, композиция отношений. Неассоциативные операции: возведение чисел в степень, вычитание множеств.

2) Коммутативные операции: сложение и умножение чисел, объединение и пересечение множеств. Некоммутативные операции: умножение матриц, композиция отношений, возведение в степень.

3) Дистрибутивные операции: умножение относительно сложения чисел. Недистрибутивные операции: возведение в степень дистрибутивно относительно умножения справа, но не слева: $((ab)^c = a^c b^c$, $a^{bc} \neq a^b a^c))$.

4) Пересечение поглощает объединение, объединение поглощает пересечение. Сложение и умножение не поглощают друг друга.

5) Идемпотентные операции: наибольший общий делитель натуральных чисел, объединение и пересечение множеств. Неидемпотентные операции: сложение и умножение чисел.

3.2. МОРФИЗМЫ

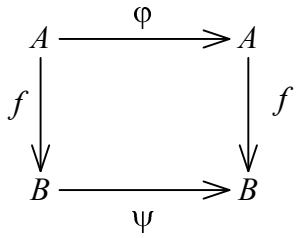
Понятие морфизма является одним из ключевых понятий алгебры. Каждая алгебраическая структура определенным образом выделяет класс "разумных" отображений между объектами с данной структурой, согласованных с операциями этой структуры. Такие отображения называются *морфизмами*. В этом разделе дается определение морфизмов для алгебраических структур. Более общее определение, пригодное для самых разных математических объектов (множеств, графов, топологических пространств и т. д.) дает теория.

3.2.1. Гомоморфизмы

Алгебры с различными типами имеют различное *строение*. Пусть $\mathcal{A} = \langle A; \varphi_1, ..., \varphi_m \rangle$ и $\mathcal{B} = \langle B; \psi_1, ..., \psi_m \rangle$ — две алгебры одинакового типа. Если существует функция $f: A \to B$, такая что

$$\forall i \in 1..m \ (f((\varphi_i(a_1), ... , a_n)) = \psi_i(f(a_1), ... , f(a_n))),$$

то говорят, что f — *гомоморфизм* из $\mathcal{A}$ в $\mathcal{B}$. Пусть $\mathcal{A} = \langle A; \varphi \rangle$, $\mathcal{B} = \langle B; \psi \rangle$, тип =(1) и $f: A \to B$. Действие этих функций можно изобразить с помощью следующей диаграммы:



Пусть f — гомоморфизм. Тогда если взять конкретное значение $a \in A$ и двигаться по двум различным путям на диаграмме, то получится один и тот же элемент $b \in B$ (так как $f(\varphi(a))=\psi(f(a))$). В таком случае диаграмма называется *коммутативной*. Коммутативной диаграмма называется потому, что условие гомоморфизма можно переписать так:

$$f \circ \varphi = \psi \circ f,$$

где ∘ — суперпозиция функций.

Замечание. Образно говорят, что гомоморфизм "уважает" операции.

Пример. Пусть $\mathcal{A} = \langle \mathbb{N}; + \rangle$, $\mathcal{B} = \langle \mathbb{N}_{10}; +_{10} \rangle$, где $\mathbb{N}_{10}=\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, а $+_{10}$ — сложение по модулю 10. Тогда $f: a \to (a \ \mathbf{mod}\ 10)$ — гомоморфизм из $\mathcal{A}$ в $\mathcal{B}$.

a) Гомоморфизмы, обладающие дополнительными свойствами, имеют специальные названия:

b) Гомоморфизм, который является инъекцией, называется *мономорфизмом*.

c) Гомоморфизм, который является сюръекцией, называется *эпиморфизмом*.

d) Гомоморфизм, который является биекцией, называется *изоморфизмом*.

e) Если $A = B$, то гомоморфизм называется *эндоморфизмом*, а изоморфизм называется *автоморфизмом.*

3.2.2. Изоморфизмы

Пусть $\mathcal{A} = \langle A; \varphi_1, ..., \varphi_m \rangle$ и $\mathcal{B} = \langle B; \psi_1, ..., \psi_m \rangle$ — две алгебры одного типа, и $f: A \to B$ — изоморфизм.

Теорема. *Если* $f: A \to B$ *— изоморфизм, то* $f^{-1}: B \to A$ *— тоже изоморфизм.*

Доказательство. Рассмотрим произвольную операцию φ из сигнатуры $\mathcal{A}$ и соответствующую ей операцию ψ из сигнатуры $\mathcal{B}$. Пусть вместимость этих операций n. Рассмотрим произвольные элементы $a_1, \ldots, a_n \in A$. Обозначим

$b_1 := f(a_1), \ldots, b_n := f(a_n)$, где $b_1, \ldots, b_n \in B$. Поскольку f — биекция, имеем $a_1 = f^{-1}(b_1), \ldots, a_n = f^{-1}(b_n)$. Тогда

$$f^{-1}(\psi(b_1, \ldots, b_n)) = f^{-1}\ \psi(f(a_1), \ldots, f(a_n))) =$$
$$= f^{-1}\ (f(\varphi(a_1, \ldots, a_n))) = \varphi(a_1, \ldots, a_n) = \varphi(f^{-1}(b_1), \ldots, f^{-1}(b_n)).$$

Если $f: A \to B$ — изоморфизм, то алгебры $\mathcal{A}$ и $\mathcal{B}$ называют *изоморфными* и обозначают так: $f: \mathcal{A} \sim \mathcal{B}$. Если f ясно из контекста или просто неважно в конкретном рассуждении, то пишут $\mathcal{A} \sim \mathcal{B}$.

Теорема. *Отношение изоморфизма на множестве однотипных алгебр является эквивалентностью*.

Доказательство.

[Рефлексивность] $f: \mathcal{A} \sim \mathcal{A}$, где f — тождественное отображение.

[Симметричность] $f: \mathcal{A} \sim \mathcal{B} \Rightarrow f^{-1}: \mathcal{B} \sim \mathcal{A}$.

[Транзитивность] $f: \mathcal{A} \sim \mathcal{B}$ & $g: \mathcal{B} \sim \mathcal{G} \Rightarrow g \circ f: \mathcal{A} \sim \mathcal{G}$.

Примеры.

1) Пусть $\mathcal{A} = \langle \mathbb{N}; + \rangle$, $\mathcal{B} = \langle \{n \mid n = 2k, k \in \mathbb{N}\}; + \rangle$ — *четные* числа. Тогда $\mathcal{A} \sim \mathcal{B}, f(n) = 2n$.

2) $\mathcal{A} = \langle 2^M; \cap, \cup \rangle \sim \mathcal{B} = \langle 2^M; \cup, \cap \rangle$, $f(X) = X'$.

3) $\mathcal{A} = \langle \mathbb{R}_+; \rangle \sim \mathcal{B} = \langle \mathbb{R}; + \rangle, f(x) = \log x$.

Понятие изоморфизма является одним из центральных понятий, обеспечивающих применимость алгебраических методов в различных областях. Действительно, пусть $\mathcal{A} = \langle A; \Sigma_\varphi \rangle$, $\mathcal{B} = \langle B; \Sigma_\psi \rangle$ и $\mathcal{A} \sim \mathcal{B}$. Пусть в алгебре $\mathcal{A}$ установлено свойство $\Phi_1 = \Phi_2$, где Φ_1 и Φ_2 — некоторые формулы в сигнатуре Σ_φ. Поскольку алгебры $\mathcal{A}$ и $\mathcal{B}$ изоморфны, отсюда немедленно следует, что в алгебре $\mathcal{B}$ справедливо свойство $\Psi_1 = \Psi_2$, где Ψ_1 и Ψ_2 — формулы, полученные из формул Φ_1 и Φ_2 заменой операций из сигнатуры Σ_φ на соответствующие операции из сигнатуры Σ_ψ. Таким образом, достаточно установить некоторое свойство в одной алгебре, и оно автоматически распространяется на все изоморфные алгебры. Алгебраические структуры принято рас-

сматривать *с точностью до изоморфизма*, то есть рассматривать классы эквивалентности по отношению изоморфизма.

3.3. ПОЛУГРУППЫ И МОНОИДЫ

Естественно начать изучение алгебраических структур с наиболее простых. Самой простой структурой является алгебра с одной унарной операцией. Здесь этот случай не рассматривается, хотя он также достаточно содержателен. Следующим по порядку является случай алгебры с одной бинарной операцией $* : M \to M \times M$, который и рассматривается в этом разделе.

3.3.1. Полугруппы

Полугруппа — это алгебра с одной ассоциативной бинарной операцией:

$$a * (b * c) = (a * b) * c.$$

Примеры.

1) Множество непустых слов A^+ в алфавите A образует полугруппу относительно операции *конкатенации*.

2) Всякое множество тотальных функций одного аргумента, замкнутое относительно суперпозиции, является полугруппой.

Если в полугруппе существует система образующих, состоящая из одного элемента, то такая полугруппа называется *циклической*.

Пример. $\langle \mathbb{N}; + \rangle$ является циклической полугруппой, поскольку $\{1\}$ является системой образующих.

3.3.2. Определяющие соотношения

Пусть $P = \langle M; * \rangle$ — полугруппа с конечной системой образующих A, $A \subseteq M$, $A = \{a_1, \ldots, a_n\}$.

Тогда $\forall\, x \in M\, (\exists\, y_1, \ldots, y_k \in A\ (x = y_1 * \ldots * y_k))$. Если опустить обозначение операции $*$, то всякий элемент $a \in M$ можно представить как слово α в алфавите A, то есть $M \subseteq A^+$. Обозначим через $\underline{\alpha}$ элемент, оп-

ределяемый словом α. Слова α и β могут быть различны, но соответствующие им элементы $\underline{\alpha}$ и $\underline{\beta}$ могут быть равны: $\underline{\alpha} = \underline{\beta}$, $\alpha, \beta \in A^+$, $\underline{\alpha}, \underline{\beta} \in M$. Такие равенства называются *определяющими соотношениями*. Если в полугруппе нет определяющих соотношений, и все различные слова, составленные из образующих, суть различные элементы носителя, то полугруппа называется *свободной*. Всякая полугруппа может быть получена из свободной введением определяющих соотношений. Два слова в алфавите A считаются равными, если одно из другого получается применением определяющих соотношений. Отношение равенства слов в полугруппе с определяющими соотношениями является отношением эквивалентности. Классы эквивалентности по этому отношению соответствуют элементам полугруппы.

Примеры.

1) В полугруппе $\langle \mathbb{N}; + \rangle$ имеется конечная система образующих $\{1\}$. Другими словами, каждое натуральное число можно представить, как последовательность знаков 1. Очевидно, что различные слова в алфавите $\{1\}$ суть различные элементы носителя, то есть эта полугруппа свободна.

2) Пусть полугруппа $\mathcal{P}$ задана системой двух образующих $\{a, b\}$ и двумя определяющими соотношениями, $aa = a$ и $bb = b$. Следующий элементарный алгоритм определяет, равны ли два слова α и β в полугруппе $\mathcal{P}$.

Вход: входные слова α: **array** [1..s] **of** $\{a, b\}$ и β: **array** [1..t] **of** $\{a, b\}$;

Выход: значение выражения α=β в полугруппе $\mathcal{P}$.

if α[1] ≠ β[1] **then**
    return false // первые буквы не совпадают
end if
N_α := 0 // счетчик изменений буквы в α
for i **from** 2 **to** s **do**

if $\alpha[i] \neq \alpha[i-1]$ **then**

$N_\alpha := N_\alpha + 1$ // буква изменилась

end if

end for

$N_\beta := 0$ // счетчик изменений буквы в β

for i **from** 2 **to** t **do**

if $\beta[i] \neq \beta[i-1]$ **then**

$N_\beta := N_\beta + 1$ // буква изменилась

end if

end for

return $N_\alpha = N_\beta$ // слова равны, если значения счетчиков одинаковы

Теорема [Марков-Пост]. *Существует полугруппа, в которой проблема распознавания равенства слов алгоритмически неразрешима.*

Доказательство. (Без доказательства). □

Пример. Г. С. Цейтин нашел пример очень простой полугруппы, в которой проблема равенства слов алгоритмически неразрешима. В этой полугруппе всего пять образующих $\{a, b, c, d, e\}$ и семь определяющих соотношений:

$ac = ca;\ ad = da;\ bc = cb;\ bd = db;\ abac = abace;\ eca = ae;\ edb = be.$

Замечание. Некоторые программисты полагают, что если в условиях задачи все дискретно и конечно, то для решения такой задачи программу можно составить в любом случае (используя метод "полного перебора"). Предшествующие теорема и пример показывают, что это мнение ошибочно.

3.3.3. Моноиды

Моноид — это полугруппа с *единицей*:

$\exists e\ (\forall a\ (a * e = e * a = a))$.

Примеры.

1) Множество слов A^* в алфавите A с операцией конкатенации вместе с пустым словом ε образует моноид.

2) Пусть T — множество термов над множеством переменных V и сигнатурой $\sum$. *Подстановкой*, или *заменой переменных*, называется множество пар $\sigma = \{t_i // v_i\}_{i \in I}$, где t_i — термы, а v_i — переменные. Результатом применения подстановки σ к терму t (обозначается $t\sigma$) называется терм, который получается заменой всех вхождений переменных v_i на соответствующие термы t_i. *Композицией* подстановок $\sigma_1 = \{t_i // v_i\}_{i \in I}$ и $\sigma_2 = \{t_j // v_j\}_{j \in J}$ называется подстановка $\sigma_1 \circ \sigma_2 := \{t_k // v_k\}_{k \in K}$, где $K = I \cup J$, а $t_k = t_i\sigma_2$, если $k \in I$ и $t_k = t_j$, если $k \notin I$.

Множество подстановок образует моноид относительно композиции, причем тождественная подстановка $\{v_i // v_i\}$ является единицей.

Теорема. *Единица моноида единственна*.

Доказательство. Пусть существует две единицы $e_1, e_2 \; \forall \{a\}\{\}\}$ Тогда $e_1 * e_2 = e_1 \; \& \; e_1 * e_2 = e_2 \Rightarrow e_1 = e_2$.

Теорема. *Всякий моноид над M изоморфен некоторому моноиду преобразований над M.*

Доказательство. Пусть $\mathcal{M} = \langle M; * \rangle$ — моноид над $M = \{e, a, b, c, ...\}$. Построим $\mathcal{F} = \langle F; \circ \rangle$ — моноид преобразований над M, где $F := \{f_m : M \to M \mid f_m(x) := x * m\}_{m \in M}$, а $\circ$ — суперпозиция функций, $h : M \to F\}$, $h(m) := f_m$. Тогда $\mathcal{F}$ — моноид, поскольку суперпозиция функций ассоциативна, f_e — тождественная функция (так как $f_e(x) = x * e = x$), и F замкнуто относительно $\circ$ (так как $f_a \circ f_b = f_{a*b}$: $f_{a*b}(x) = x * (a * b) = (x * a) * b = f_a(x) * b = f_b(f_a(x))$. Далее, h — гомоморфизм (так как $h(a * b) = f_{a*b} = f_a \circ f_b = h(a) \circ h(b)$). И наконец, h — биекция, поскольку h — сюръекция по построению, и h — инъекция (так как $(f_a(e) = e * a = a \; \& \; f_b(e) = e * b = b) \Rightarrow (a \neq b \Rightarrow f_a \neq f_b)$).

3.4. ГРУППЫ

Теория групп, некоторые положения которой рассматриваются в этом разделе, является ярчайшим примером мощи алгебраических методов, позволяющих получить из немногих базовых понятий и аксиом множество важнейших следствий.

3.4.1. Определение и основные свойства

Группа — это моноид, в котором

$$\forall a\,(\exists a^{-1}\,(a * a^{-1} = a^{-1} * a = e)).$$

Элемент a^{-1} называется *обратным*, а операция $*$ называется *умножением*.

Замечание. Так как операция умножения в группе не обязательно коммутативна, то, вообще говоря, свойства $a^{-1} * a = e$ и $a * a^{-1} = e$ не равносильны. Можно было бы различать обратные слева и обратные справа элементы. Введенная аксиома требует существования двустороннего обратного элемента. Можно показать, что она следует из более слабых аксиом существования левой единицы и левого обратного элемента.

Мощность носителя G группы $\mathcal{G}$ называется *порядком группы* и обозначается $|G|$.

Пример. Множество невырожденных квадратных матриц порядка n образует группу относительно операции умножения матриц. Единицей группы является единичная матрица (единицы на главной диагонали, остальные элементы равны нулю). Обратным элементом является обратная матрица.

Множество перестановок на множестве M, то есть множество взаимно однозначных функций $f\colon M \to M$ является группой относительно операции суперпозиции. Единицей группы является тождественная функция, а обратным элементом — обратная функция.

Теорема 1. *Обратный элемент единственен*.

Доказательство. Пусть $a* a^{-1} = a^{-1}* a = e$ & $a* b = b* a = e$.
Тогда
$a^{-1} = a^{-1}* e = a^{-1}*(a* b) = (a^{-1}* a)* b = e* b = b$.

Теорема 2. *В группе выполняются следующие соотношения*:
1) $(a* b)^{-1} = b^{-1} * a^{-1}$;
2) $a * b = a * c \Rightarrow b = c$;
3) $b * a = c * a \Rightarrow b = c$;
4) $(a^{-1})^{-1} = a$.
Доказательство.
[1] $(a*b)*(b^{-1}*a^{-1}) = a*(b*b^{-1})* a^{-1} = a*e*a^{-1} = a*a^{-1} = e$.
[2] $a*b = a*c \Rightarrow a^{-1}*(a*b) = a^{-1}*(a*c) \Rightarrow (a^{-1}*a)*b = (a^{-1}*a)*c \Rightarrow$
$\Rightarrow e*b = e*c \Rightarrow b = c$.
[3] $b*a = c*a \Rightarrow (b*a)*a^{-1} = (c*a)*a^{-1} \Rightarrow b*(a*a^{-1}) = c*(a*a^{-1}) \Rightarrow$
$\Rightarrow b*e = c*e \Rightarrow b = c$.
[4] $(a^{-1})*a = a^{-1}*a = e$.

Теорема 3. *В группе можно однозначно решить уравнение*

$$a * x = b, \text{ (}\textit{решение}\text{: } x = a^{-1} * b\text{)}.$$

Доказательство. $a*x = b \Rightarrow a^{-1}*(a*x) = a^{-1}*b \Rightarrow$
$\Rightarrow (a^{-1}*a)*x = a^{-1}*b \Rightarrow$
$\Rightarrow e*x = a^{-1}*b \Rightarrow x = a^{-1}*b$.
Коммутативная группа, то есть группа в которой

$$a * b = b * a,$$

называется *абелевой*. В абелевых группах приняты следующие обозначения: групповая операция обычно обозначается +, обратный элемент к a обозначается $-a$, единица группы обозначается 0 и называется *нулем*.

Примеры.

1) $\langle \mathbb{Z}; + \rangle$ — множество целых чисел образует абелеву группу относительно сложения. Нулем группы является число 0. Обратным элементом является число с противоположным знаком: $x^{-1} := -x$.

2) $\langle \mathbb{Q}_+; \cdot \rangle$ — множество положительных рациональных чисел образует абелеву группу относительно умножения. Нулем группы является число 1. Обратным элементом является обратное число: $(m/n)^{-1} := n/m$.

3) $\langle 2^M; \quad \rangle$ — булеан образует абелеву группу относительно симметрической разности. Нулем группы является пустое множество $\varnothing$. Обратным элементом к элементу x является он сам: $x^{-1} := x$.

Далее для обозначения знака операции в группе, наряду со знаком $*$, используется знак $\cdot$ или же знак операции опускается, там, где это не приводит к недоразумениям.

3.4.2. Гомоморфизмы групп

Взятие обратного элемента можно рассматривать как дополнительную унарную операцию. Следующая теорема показывает, что это никак не влияет на определения морфизмов.

Лемма. *Пусть* $\mathcal{G}$ и $\mathcal{G}'$ *— группы*, e *и* e' *их единичные элементы*, *соответственно*, *и* $f\colon G \to G'$ *— гомоморфизм групп*. *Тогда* $f(e) = e'$.

Доказательство. Имеем $f(e)\cdot f(e) = f(e\cdot e) = f(e)$ и $e'\cdot f(e) = f(e)$. Но уравнение $x\cdot a = b$ имеет одно и только одно решение, значит $e' = f(e)$.

Теорема. *Пусть* $\mathcal{G}$ *и* $\mathcal{G}'$ *— группы*, *и* $f\colon G \to G'$ *— гомоморфизм групп*. *Тогда* $f(x^{-1}) = f(x)^{-1}$.

Доказательство. Если e и e' — единичные элементы в $\mathcal{G}$ и $\mathcal{G}'$, то $e' = f(e) = f(x\cdot x^{-1}) = f(x)\, f(x^{-1})$.

Нетрудно видеть, что *множество автоморфизмов группы* $\mathcal{G}$ (обозначается $\mathrm{Aut}(\mathcal{G})$) само является группой, если в качестве умножения берется суперпозиция функций.

Пример. Пусть $\mathcal{G}$ — группа и $g \in G$. Тогда отображение $f\colon \mathbb{Z} \to G$, где $f(n) := g^n$, является гомоморфизмом.

Если $\mathcal{G}$ — абелева группа, то отображение $x \rightarrowtail x^n$ группы G в себя является гомоморфизмом. Оно называется *возведением в n-ю степень*. Заметим, что для некоммутативной группы это не всегда верно.

Взятие обратного элемента $x \rightarrowtail x^{-1}$ также является гомоморфизмом.

Если g — некоторый элемент G, то отображение $x \rightarrowtail gxg^{-1}$ является изоморфизмом и называется *сопряжением*.

Два элемента x и y группы $\mathcal{G}$ называются *сопряжёнными*, если $\exists g \in G\ (gxg^{-1} = y)$. Отношение сопряжённости является отношением эквивалентности. Можно показать, что если $S \subseteq G$ — множество образующих $\mathcal{G}$, и $f\colon S \to G'$ — отображение S в некоторую группу $\mathcal{G}'$, то f допускает не более чем одно *продолжение до гомоморфизма* $f'\colon G \to G'$, такого что $f'(s) = f(s)$ для $s \in S$. Таким образом, имеется эффективный способ задания гоморфизмов. Любой гомоморфизм будет однозначно определен, если его задать на образующих. Этот важнейший принцип применим и к другим алгебраическим системам.

3.4.3. Ядро и образ

Пусть $f\colon G \to G'$ — гомоморфизм групп, e и e' — единичные элементы в $\mathcal{G}$ и $\mathcal{G}'$. *Ядром гомоморфизма f* называется подмножество в G, состоящее из тех x, для которых $f(x) = e'$. Ядро гоморфизма f обозначается $\ker f$.

$$\ker f := \{g \in G \mid f(g) = e'\}.$$

Из определения следует, что $\ker f$ — подгруппа в $\mathcal{G}$. *Образ гомоморфизма* f — это образ группы G при отображении f (обозначение $\mathrm{Im}\, f$). Легко проверить, что $\mathrm{Im}\, f$ — подгруппа в $\mathcal{G}'$.

Теорема. *Гомоморфизм, ядро которого тривиально, инъективен*.

Доказательство. Предположим, что ядро гомоморфизма f тривиально, то есть $\ker f = \{e\}$. Допустим, что $f(x) = f(y)$. Тогда $f(x \cdot y^{-1}) = f(x) \cdot f(y^{-1}) = f(y) \cdot f(y^{-1}) = f(y) \cdot f(y)^{-1} = e'$. Следовательно, $x \cdot y^{-1} \in \ker f$, то есть $x \cdot y^{-1} = e$ и значит $x = y$.

Оказывается, что не все подгруппы группы $\mathcal{G}$ могут служить ядрами гомоморфизмов. В следующем параграфе мы опишем такие подгруппы.

3.4.4. Нормальные делители и факторгруппы

Пусть f: $G \to G'$ — гомоморфизм групп, и $H = \ker f$. Нетрудно видеть, что $\forall x \in G\ (x \cdot H \cdot x^{-1} = H)$. Подгруппа $H \subseteq G$, удовлетворяющая условию $\forall x \in G\ (xHx^{-1} = H)$, называется *нормальной подгруппой* или *нормальным делителем* в G. Оказывается, что все нормальные делители и только они служат ядрами гомоморфизмов. Пусть $H \subseteq G$ — подгруппа группы G. *Левым смежным классом* по H в группе G называется множество вида xH, где $x \in G$. Аналогично определяются *правые смежные классы* Hx. Определение нормального делителя $xHx^{-1} = H$ можно написать эквивалентным способом: $xH = Hx$. Это обстоятельство позволяет не различать правые и левые смежные классы, если H — нормальный делитель. В этом случае слова "правый" или "левый" опускаются. Определим теперь на множестве G' смежных классов по нормальному делителю H операцию умножения:

$$(xH) \cdot (yH) := xyH.$$

Теорема 1. *Если H — нормальный делитель в группе G, то множество смежных классов G' является группой относительно введенной операции умножения*.

Доказательство. Действительно, класс eH служит единичным элементом, а класс $x^{-1}H$ — обратным к классу xH.

Ассоциативность умножения классов следует из ассоциативности умножения в группе G:

$((xH)\cdot(yH))\cdot(zH) = (xyH)\cdot(zH) = xyzH$,

$(xH)\cdot((yH)\cdot(zH)) = (xH)\cdot(yzH) = xyzH$.

Рассмотрим отображение f: $G \to G'$, где $f(x) := xH$, Очевидно, что f является гомоморфизмом, а его ядром является H, $\ker f = H$.

Теорема 2. *Пусть f: $G \to G'$ — гомоморфизм групп, и $H = \ker f$. Тогда H является нормальным делителем в группе G.*

Доказательство. Покажем, то $\forall x\ (xHx^{-1} = H)$. Пусть $h \in H$. Тогда $f(xhx^{-1}) = f(x)\cdot e'\cdot f(x^{-1}) = f(x)\cdot f(x^{-1}) = f(xx^{-1}) = f(e) = e'$. Таким образом, $\forall x\ (xHx^{-1} \subseteq H)$. Обратно, пусть $h_1 = x^{-1}hx$, где $h \in H$. Тогда ясно, что $h_1 \in H$, но $h = xhx^{-1}$, следовательно $\forall x\ (H \subseteq xHx^{-1})$.

Группа смежных классов называется *факторгруппой* группы G по нормальной подгруппе H и обозначается G/H.

Пример. $G/\{e\} = G$ — факторгруппа группы G по тривиальной подгруппе равна самой группе G. $G/G = \{eG\}$ — факторгруппа группы по себе самой состоит из одного элемента. Таким образом, операция взятия факторгруппы напоминает операцию деления.

Если $G = \mathbb{Z}$, а $H = n\mathbb{Z}$ ($n\mathbb{Z}$, очевидно, является подгруппой в $\mathbb{Z}$), то факторгруппа $\mathbb{Z}/n\mathbb{Z}$ состоит из множеств вида $a + n\mathbb{Z}$, где $a \in [0..n-1]$. Элементы факторгруппы представлены всеми различными остатками по модулю n. Группа $\mathbb{Z}/n\mathbb{Z}$ обозначается $\mathbb{Z}_n$.

Замечание. Смежные классы — это в точности классы эквивалентности группы G по отношению $(x \Leftrightarrow y)$ **:=** $(xy^{-1} \in H)$. Таким обра-

зом, факторгруппа есть фактормножество G по этому отношению эквивалентности с естественной операцией умножения.

3.4.5. Теорема о гомоморфизме

Следующая теорема имеет фундаментальное значение, несмотря на всю ее простоту.

Теорема. *Пусть* $f\colon G \to G'$ *— гомоморфизм групп*. *Тогда*

$$\operatorname{Im} f \sim G / \ker f.$$

Другими словами, гомоморфный образ группы изоморфен факторгруппе по ядру гомоморфизма.

Доказательство. Пусть $H = \ker f$ и $G' = \operatorname{Im} f$. Чтобы определить изоморфизм φ групп G/H и G', рассмотрим xH — смежный класс по H. Так как $\forall y\ (f(xy) = f(x))$, то есть значение $f(x)$ зависит только от смежного класса xH, которому принадлежит x, положим $\varphi(xH) := f(x)$. Гоморофизм φ сюръективен по построению. Далее, гоморофизм φ имеет тривиальное ядро и инъективен. Следовательно, $G' \sim G/H$.

Пример. Множество ненулевых вещественных чисел $G = \mathbb{R}\backslash\{0\}$ образует группу относительно умножения. $H=\mathbb{R}^+$ — ее подгруппа. Тогда факторгруппа G/H есть группа из двух элементов, изоморфная $\mathbb{Z}_2$.

3.4.6. Действие группы на множестве

Пусть S — множество, а $\mathcal{G}$ — группа. *Действием группы* $\mathcal{G}$ *на множестве* S называется отображение $G\times S \to S$, такое, что если обозначить через gs образ пары (g, s) при этом отображении, то

$$\forall g_1, g_2 \in G\ (\forall s \in S\ ((g_1 \cdot g_2)s = g_1(g_2\, s))\ ),\ \forall s \in S(es = s).$$

В этом случае также говорят, что $\mathcal{G}$ *действует* на множестве S или что S есть $\mathcal{G}$-множество. Если $s \in S$, то множество

$Gs = \{s' \in S \mid s' = gs \;\&\; g \in G\}$, то есть множество элементов вида gs, где $g \in G$, называется *орбитой* (или *траекторией*) элемента s.

Пример. Пусть S_1 — единичная окружность, $\varphi \in 0..2\pi$ — некоторый угол. Определим действие группы $\mathbb{Z}$ на множестве S_1 следующим образом: элемент $n \in \mathbb{Z}$ действует на точку $x \in S_1$ как поворот точки x против часовой стрелки на угол $n\varphi$. Если φ соизмерим с числом π, то все орбиты состоят из конечного числа элементов. Если же φ/π иррационально, то все орбиты бесконечны.

Теорема. *Если G действует на множестве S, то различные орбиты либо не пересекаются, либо совпадают*.

Доказательство. Действительно, если Gs_1 и Gs_2 — две орбиты с общим элементом s, то $s = gs_1$ для некоторого g и $Gs = Ggs_1 = Gs_1$. Аналогично $Gs = Gs_2$. Здесь использовано очевидное равенство $Gg = G$.

Таким образом, множество S есть объединение попарно непересекающихся орбит:

$$S = \bigcup_{i \in I} Gs_i,\ Gs_i \cap Gs_j = \varnothing,$$

где I — некоторое множество индексов, а s_i — элементы разных орбит. Это разбиение дает полезную формулу $|S| = \Sigma_{i \in I} |Gs_i|$, которая называется *формулой разложения на орбиты*.

Пример. Каждая группа G действует на самой себе с помощью умножения. Здесь $S = G$, а отображение действия $G \times S \rightarrowtail S$ является умножением в группе. Это действие имеет всего одну орбиту.

Если $H \subseteq G$ — подгруппа, то H также действует на G. Орбитами этого действия являются смежные классы Hx. Все они имеют одинаковое количество элементов, равное порядку группы H, а их количество называется *индексом* группы H в G и обозначается $(G{:}H)$. Формула разложения на орбиты в этом случае дает $|G| = |H| \cdot (G{:}H)$.

Замечание. Из этого следует, что порядок подгруппы является делителем порядка группы.

Задать действие группы на множестве — все равно, что рассматривать элементы группы как преобразования этого множества. Исторически группы и возникли как группы преобразований.

3.4.7. Группа перестановок

В этом параграфе рассматривается одна из важнейших групп, называемая группой *перестановок*, или *симметрической* группой.

Взаимнооднозначная функция $f\colon X \to X$ называется *перестановкой* множества X.

Замечание. Если множество X конечно ($|X| = n$), то, не ограничивая общности, можно считать, что $X = 1..n$. В этом случае перестановку $f\colon 1..n \to 1..n$ удобно задавать таблицей из двух строк. В первой строке — значения аргументов, во второй — соответствующие значения функции. Такая таблица называется *подстановкой*. В сущности, перестановка и подстановка — синонимы.

Пример.

$f =$	1	2	3	4	5
	5	2	1	4	3

$g =$	1	2	3	4	5
	4	1	2	3	5

Произведением перестановок f и g (обозначается fg) называется их суперпозиция $g \circ f$.

Пример.

$fg =$	1	2	3	4	5
	5	1	4	3	2

Тождественная перестановка — это перестановка e, такая что $e(x) = x$. *Обратная* перестановка — это обратная функция, которая всегда существует, поскольку перестановка является биекцией.

Замечание. Таблицу обратной подстановки можно получить, если просто поменять местами строки таблицы исходной подстановки.

Таким образом, множество перестановок n-элементного множества образует группу относительно операции суперпозиции. Эта группа называется *симметрической* степени n и обозначается S_n.

Теорема. *Любая конечная группа G есть подгруппа группы перестановок*.

Доказательство. Так как множество G конечно и G действует на этом множестве с помощью умножения, то каждый элемент группы G задает некоторую перестановку самого множества G. Результаты действия перестановок, задаваемых элементами g_1, $g_2 \in G$, совпадают: $g_1(g_2\, x) = (g_1\, g_2)x$.

Таким образом, G отображается в группу перестановок. Обозначим это отображение φ. Единственный элемент, который оставляет на месте все элементы G, является единичным. Следовательно, ядро этого отображения состоит из единицы, и значит, оно инъективно. Согласно теореме о гомоморфизме φ изоморфно отображает G на свой образ.

Замечание. Таким образом, все конечные группы являются подгруппами симметрических групп S_n. Не надо думать, что описание подгрупп симметрической группы является простой задачей. Это реально пока только при небольших значениях n.

3.4.8. Простые группы

Группа, порожденная одним элементом x, называется *циклической*. Элементами этой группы являются всевозможные степени элемента x. Очевидно, что циклическая группа — абелева.

Пример. Циклическая группа порядка n изоморфна $\mathbb{Z}_n$.

Простой называется группа G, не имеющая никаких нормальных подгрупп, кроме $\{e\}$ и $\{G\}$.

Теорема. *Конечная циклическая группа является простой тогда и только тогда, когда ее порядок — простое число*.

Доказательство. Пусть $|G| = n$ и a — образующая группы G.

[Необходимость] Так как порядок подгруппы делит порядок группы, то любая подгруппа имеет порядок, равный либо n, либо 1. В первом случае подгруппа совпадает со всей группой. Во втором случае она тривиальна, то есть состоит только из единичного элемента.

[Достаточность] От противного. Пусть n не простое. Тогда $n = dd_1$, где $1 < D < n$. Нетрудно видеть, что элементы e, a^d, a^{2d}, ..., $a^{d_1 d}$ образуют подгруппу группы G, состоящую из d_1 элементов. Так как группа G абелева, то эта подгруппа — нормальный делитель, что противоречит тому, что группа G — простая.

Замечание. Описание всех остальных простых конечных групп было одним из крупнейших достижений математики XX века. Классификация всех простых конечных групп была завершена открытием и описанием свойств простой группы порядка

$$808017424794512875886459904961710757005754368000000000 =$$
$$= 2^{46} \cdot 3^{20} \cdot 5^9 \cdot 7^6 \cdot 11^2 \cdot 13^3 \cdot 17 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 41 \cdot 47 \cdot 59 \cdot 71,$$

называемой *группой Грисса–Фишера* или *Монстром*. Доказательство существования и описание свойств этой группы было бы невозможно без использования современных мощных компьютеров и реализации многих теоретико-групповых алгоритмов.

3.4.9. Свободная группа

Одной из важнейших групп является *свободная группа*, порожденная некоторым набором элементов X, причем все элементы этого набора считаются независимыми. Пусть $X = \{a, b, c, ...\}$ — множество символов некоторого алфавита. Дополним алфавит X элементами a^{-1}, b^{-1}, c^{-1}, ... и специальным элементом ε, который будет служить обозначением единичного элемента: $\underline{X} = X \cup \{\varepsilon\} \cup \{x^{-1} \mid x \in X\}$.

Здесь x^{-1} пока просто абстрактные символы. Рассмотрим множество F_X всевозможных слов в алфавите $\underline{X}$ со следующими правилами сокращения, справедливыми для любого символа $x \in X$:

$$\varepsilon x \rightarrowtail x,\ x\varepsilon \rightarrowtail x,\ xx^{-1} \rightarrowtail \varepsilon,\ x^{-1}x \rightarrowtail \varepsilon\ .$$

Два слова назовем *эквивалентными*, если одно из них получается из другого с помощью указанных правил. Очевидно, что данное отношение рефлексивно, симметрично и транзитивно и действительно является эквивалентностью.

Пример. Слово $\mathrm{abaa}^{-1}\mathrm{abbb}^{-1}\mathrm{baa}$ эквивалентно слову ababbaa.

Множество классов эквивалентности слов образует группу, если в качестве умножения слов использовать конкатенцию. Такая группа называется *свободной* группой с образующими из X и обозначается F_X. Легко видеть, что элементы из X порождают F_X. Таким образом, свободная группа — это просто набор слов.

Пример. $F_{\{a\}}$ является бесконечной циклической группой, порожденной элементом a, $F_{\{a\}} \sim \mathbb{N}$.

Замечание. Свободная группа уже с двумя образующими поистине огромна. В качестве подгруппы она содержит группу, изоморфную свободной группе с бесконечным числом образующих.

Важность свободных групп определяется следующей теоремой.

Теорема. *Всякая группа есть факторгруппа некоторой свободной группы*.

Доказательство. Без доказательства.

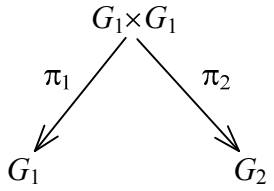
3.4.10. Прямое произведение групп

Большие группы можно строить из меньших, используя их в качестве строительных элементов. Пусть G_1 и G_2 — группы. *Прямым произведением* групп G_1 и G_2 называется множество пар $(g_1, g_2) \in G_1 \times G_2$, где умножение определено покомпонентно:

$$(g_1, g_2)\cdot(h_1, h_2) := (g_1 h_1, g_2 h_2).$$

Если e_1 и e_2 — единичные элементы групп G_1 и G_2, то единичным элементом в группе $G_1 \times G_2$ служит элемент (e_1, e_2). Легко проверить, что все аксиомы группы выполнены, таким образом, $G_1 \times G_2$ — группа. Отображения $\pi_1(g_1, g_2) := g_1$ и $\pi_2(g_1, g_2) := g_2$ являются гомо-

морфизмами группы $G = G_1 \times G_2$ в G_1 и G_2 соответственно, называются *проекциями*. Таким образом, мы имеем следующую диаграмму:



Здесь π_1 и π_2 — эпиморфизмы.

Пример. Если числа m и n взаимно просты, то группа $Z_m \times Z_n$ изоморфна Z_{nm}.

3.5. КАТЕГОРИИ И ФУНКТОРЫ

В предыдущих разделах рассматриваются объекты разного рода: множества, группы, упорядоченные множества, моноиды, полугруппы. В последующих разделах рассматриваются и другие виды объектов. Для объектов каждого рода определяются специальные отображения между ними, например гомоморфизмы. При этом многие формальные свойства таких объектов и отображений оказываются общими в разных случаях. Понятие категории призвано в максимальной общности описывать взаимоотношения между объектами и отображениями в различных математических теориях.

3.5.1. Определение категории

Категория $\mathcal{A}$ включает в себя:

1) Класс объектов $\mathrm{Ob}(\mathcal{A})$.

2) Для всяких двух объектов $A, B \in \mathrm{Ob}(\mathcal{A})$ множество $\mathrm{Mor}(A, B)$, называемое *множеством морфизмов*.

3) Для всяких трех объектов $A, B, C \in \mathrm{Ob}(\mathcal{A})$ *закон композиции*, то есть отображение $\circ$: $\mathrm{Mor}(B, C) \times \mathrm{Mor}(A, B) \to \mathrm{Mor}(A, C)$. Здесь и в дальнейшем композицию морфизмов f и g обозначаем $f \circ g$.

При этом должны выполняться следующие аксиомы:

[A_1] Множества морфизмов не пересекаются, за исключением случая, когда они равны:

$$A \neq A' \vee B \neq B' \Rightarrow \mathrm{Mor}(A, B) \cap \mathrm{Mor}(A', B') = \varnothing.$$

[A_2] Для каждого объекта имеется "тождественный" морфизм со свойством единичного элемента:

$$\forall A \in \mathrm{Ob}(\mathcal{A})(\exists \mathrm{id}_A(f \in \mathrm{Mor}(A,B) \Rightarrow f \circ \mathrm{id}_A = f) \& (g \in \mathrm{Mor}(B,A) \Rightarrow \mathrm{id}_A \circ g = f)))$$

[A_3] Закон композиции ассоциативен:

$$\forall A, B, C, D \in \mathrm{Ob}(\mathcal{A})\ (f \in \mathrm{Mor}(A,B)\ \&\ g \in \mathrm{Mor}(B,C)\ \&\ h \in \mathrm{Mor}(C,D)$$
$$\Rightarrow (h \circ g) \circ f = h \circ (g \circ f)).$$

Элемент $f \in \mathrm{Mor}(A, B)$ записывается также в виде $f\colon A \to B$ и иногда называется *стрелкой*.

Замечание. Иногда теорию категорий называют "теорией стрелок" или "абстрактной чепухой". Последний термин используется в позитивном смысле.

Замечание. Внимательный читатель может заметить, что введенное здесь обозначение для композиции расходится с тем, которое было дано в параграфе 1.5.4. Такое расхождение в обозначениях имеет определенные основания. Действительно, пусть нужно вычислить выражение $f(g(x))$, то есть суперпозицию функций. Функции являются частным случаем отношений, а суперпозиция — частным случаем композиции. В обозначениях параграфа 1.5.4 такая композиция запишется как $(g \circ f)(x)$. Данное обозначение имеет следующее "программно-вычислительное" оправдание: функции в композиции записаны в том порядке, в котором их нужно вычислять (в программе). С другой стороны, в подавляющем большинстве математических текстов композиция функций (или суперпозиция) $f(g(x))$ записывается так: $(f \circ g)(x)$. Такая запись естественно следует из традиции писать знак

функции слева от аргумента. В этом разделе мы придерживаемся традиционного способа записи композиции (суперпозиции).

Пример. Категория множеств **Set**. Объекты категории — произвольные множества. Множество морфизмов Mor(A, B) образуют произвольные отображения из A в B.

Упорядоченные множества **OrSet**. Объекты категории **OrSet** — множества с отношением порядка $\prec$, а Mor(A, B) — монотонные отображения, то есть сохраняющие порядок.

Группы **Gr**. Морфизмы в этой категории — это гомоморфизмы групп.

Замечание. Мы смело написали для объектов категории $A \in \mathrm{Ob}(\mathcal{A})$, хотя класс объектов не обязан быть множеством, например, в категории **Set** объект Ob(**Set**) — класс всех множеств, который множеством не является. В данном контексте это не приводит к противоречиям, подобным парадоксу Рассела (см. параграф. 1.1.3) и мы будем использовать знак $\in$ для обозначения принадлежности к классу.

3.5.2. Свойства морфизмов

Морфизм $f\colon A \to B$ называется *изоморфизмом*, если существует "обратный" морфизм $g\colon B \to A$, такой что $g \circ f$ и $f \circ g$ являются тождественными морфизмами, то есть $g \circ f = \mathrm{id}_A$ и $f \circ g = \mathrm{id}_B$. Если $A = B$, то изоморфизм называется *автоморфизмом*. Множество всех автоморфизмов объекта A обозначается Aut(A).

Пример. В категории **Set** изоморфизмами являются взаимно однозначные отображения (биекции).

Произвольные морфизмы объекта A в себя называются *эндоморфизмами*. Множество эндоморфизмов обозначается End(A). Множество End(A) относительно закона композиции $\circ$ является моноидом,

а множество Aut(A) — группой, что немедленно следует из определений.

3.5.3. Функторы

Функторы, определяемые в этом параграфе, призваны играть роль отображений между различными категориями. Они, как правило, не являются настоящими отображениями, так как определены на классах, а не на множествах. Излишняя педантичность в этом вопросе приводит к необходимости рассматривать громоздкие конструкции, подобные универсуму, замкнутому относительно всех рассматриваемых операций. В противном случае сразу же неизбежны противоречия, подобные парадоксу Рассела. Чтобы избежать этих сложностей, мы обращаемся с классами как с множествами, заранее зная, что все конструкции, используемые в данном учебном пособии, к противоречиям не приводят.

Пусть $\mathcal{A}$ и $\mathcal{B}$ — категории. *Ковариантный функтор* $\mathcal{F}$ из $\mathcal{A}$ в $\mathcal{B}$ — это правило, сопоставляющее каждому объекту A в $\mathcal{A}$ некоторый объект $\mathcal{F}(A)$ в $\mathcal{B}$ и каждому морфизму $f\colon X \to Y$ категории $\mathcal{A}$, действующему из объекта $X \in \mathrm{Ob}(\mathcal{A})$ в объект $Y \in \mathrm{Ob}(\mathcal{A})$, морфизм $\mathcal{F}(f)\colon \mathcal{F}(X) \to \mathcal{F}(Y)$ таким образом, что выполняются следующие условия:

1) $\forall A \in \mathcal{A}\ (\mathcal{F}(\mathrm{id}_A) = \mathrm{id}_{\mathcal{F}(A)})$;

2) $f \in \mathrm{Mor}(A, B)\ \&\ g \in \mathrm{Mor}(B, C) \Rightarrow \mathcal{F}(g \circ f) = \mathcal{F}(g) \circ \mathcal{F}(f)$.

Слово "ковариантный" здесь означает, что функтор $\mathcal{F}$ сохраняет направления стрелок. Если $f \in \mathrm{Mor}(A, B)$, то есть $f\colon A \to B$, то $\mathcal{F}(f) \in \mathrm{Mor}(\mathcal{F}(A), \mathcal{F}(B))$, то есть $\mathcal{F}(f)\colon \mathcal{F}(A) \to \mathcal{F}(B)$. Можно определить понятие *контравариантного функтора*, используя то же самое условие 1) и обращая стрелки в условии 2). При этом каждому морфизму $f\colon A \to B$ контравариантный функтор $\mathcal{F}$ сопоставляет морфизм в об-

ратном направлении $\mathcal{F}(f)$: $\mathcal{F}(B) \to \mathcal{F}(A)$, так что если f: $A \to B$ и g: $B \to C$, то $\mathcal{F}(g \circ f) = \mathcal{F}(f) \circ \mathcal{F}(g)$.

Пример. Пусть **Gr** — категория групп, а **Set** — категория множеств. Сопоставим каждой группе $G \in \mathrm{Ob}(\mathbf{Gr})$ ее носитель как множество, а каждому гомоморфизму сопоставляется он сам как отображение множеств. Получаем функтор из категории групп в категорию множеств. Он как бы "забывает" структуру группы на G. Такой функтор называется *стирающим* (или *забывающим*).

Следующий пример сложнее и гораздо содержательнее. Пусть $\mathcal{A}$ — некоторая категория и $A \in \mathrm{Ob}(\mathcal{A})$ — объект. Зафиксируем этот объект A. Определим функтор $\mathcal{M}_A$ следующим образом. Для любого объекта $X \in \mathrm{Ob}(\mathcal{A})$ положим $\mathcal{M}_A(X) := \mathrm{Mor}(A, X)$. Если теперь f: $X \to X'$ — морфизм, то определим $\mathcal{M}_A(\varphi)$: $\mathrm{Mor}(A, X) \to \mathrm{Mor}(A, X')$ — отображение, задаваемое правилом $\mathcal{M}_A(g) := \varphi \circ g$ для любого $g \in \mathrm{Mor}(A, X)$:

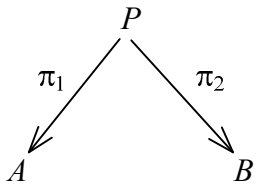
$$g\colon A \to X,\ \varphi\colon X \to X'.$$

Все свойства, определяющие функтор, выполнены. Такой функтор (из произвольной категории в категорию множеств) называется *представляющим*. Он хранит самую существенную информацию об объекте A произвольной категории.

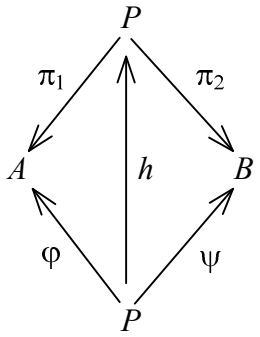
3.5.4. Прямое произведение

Занимаясь множествами, мы определили понятие прямого произведения множеств. Если множества несут на себе дополнительную структуру, например структуру порядка, моноида или группы, то в каждом из этих случаев также можно дать подходящее определение прямого произведения. Теория категорий позволяет дать самое общее определение, применимое во всех случаях. Ключевым свойством прямого произведения $P = A \times B$ при этом оказывается наличие двух

отображений $\pi_1: P \to A$ и $\pi_2: P \to B$, которые называются (каноническими) *проекциями*. Это приводит к следующему определению. Пусть $\mathcal{A}$ — категория и $A, B \in \mathrm{Ob}(\mathcal{A})$. *Прямым произведением* объектов A и B в категории $\mathcal{A}$ (обозначается $A \times B$) называется тройка $\langle P, \pi_1, \pi_2 \rangle$,



состоящая из объекта $P \in \mathrm{Ob}(\mathcal{A})$ и двух морфизмов $\pi_1: P \to A$ и $\pi_2: P \to B$, удовлетворяющих следующему условию: для любого объекта $C \in \mathrm{Ob}(\mathcal{A})$ и для любых морфизмов $\varphi: C \to A$ и $\psi : C \to B$ существует и единственен морфизм $h: C \to P$, для которого следующая диаграмма коммутативна, то есть $\varphi = \pi_1 \circ h$ и $\psi = \pi_2 \circ h$:



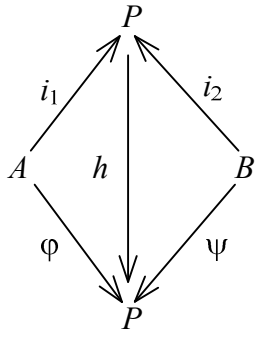
Пример. Пусть X_1 и X_2 — множества. Тогда $X_1 \times X_2$ в категории **Set** — это знакомое нам декартово произведение, то есть множество пар (x_1, x_2), где $x_1 \in X_1$, $x_2 \in X_2$ с проекциями $\pi_1(x_1, x_2) := x_1$ и $\pi_2(x_1, x_2) := x_2$.

3.5.5. Прямая сумма

Обращение направления стрелок в определении прямого произведения приводит к определению *прямой суммы* объектов категории. Ключевым свойством прямой суммы $P = A \oplus B$ при этом оказывается

наличие двух отображений i_1: $A \to P$ и i_2: $B \to P$, которые называются (каноническими) *вложениями*.

Пусть $\mathcal{A}$ — некоторая категория и $A, B \in \mathrm{Ob}(\mathcal{A})$. *Прямой суммой* объектов A и B в категории $\mathcal{A}$ (обозначаемой $A \oplus B$) называется тройка $\langle P, i_1, i_2 \rangle$, состоящая из объекта $P \in \mathrm{Ob}(\mathcal{A})$ и двух морфизмов i_1: $A \to P$ и i_2: $B \to P$, таких, что для любого объекта $C \in \mathrm{Ob}(\mathcal{A})$ и для любых двух морфизмов φ: $C \to A$ и $\psi : C \to B$ существует и единственен морфизм h: $P \to C$, такой что следующая диаграмма коммутативна, то есть $\varphi = h \circ i_1$ и $\psi = h \circ i_2$:



Пример. Прямой суммой множеств A и B в категории множеств является дизъюнктивное объединение $(A \times \{1\}) \cup (B \times \{2\})$. Здесь мы сначала домножаем множества A и B на разные элементы $\{1\}$ и $\{2\}$, чтобы сделать их непересекающимися, а затем объединяем. Нетрудно проверить, что определенная таким образом сумма действительно является прямой суммой в смысле теории категорий.

3.5.6. Единственность сумм и произведений

Все рассуждения этого раздела относятся к наиболее общим свойствам математических объектов, поэтому они неизбежно имеют чрезвычайно высокую степень абстракции. Осозновая тот факт, что программисту обычно легче мыслить конкретными объектами и понятиями, мы, тем не менее, сочли возможным и нужным представить основные понятия и методологию теории категорий. Чтобы показать

мощь и универсальность такого подхода, мы в этом параграфе докажем, что прямые суммы и прямые произведения определены однозначно с точностью до изоморфизма. Читатель, готовый принять это утверждение на веру, может без ущерба пропустить этот параграф, хотя мы настоятельно рекомендуем осмыслить приведенные доказательства. Пусть $\mathcal{A}$ — категория. Объект I называется *начальным* в категории $\mathcal{A}$, если каждое из множеств $\mathrm{Mor}(I, X)$ состоит только из одного элемента. То есть из объекта I в любой другой объект ведет только одна стрелка.

Пример. Группа из одного элемента — начальный объект в категории групп.

В категории множеств начальных объектов нет, так как множество функций с пустой областью определения пусто.

Объект F называется *финальным* в категории $\mathcal{A}$, если для каждого A множества $\mathrm{Mor}(A, F)$ состоят из единственного элемента.

Пример. Одноэлементное множество в категории множеств — финальный объект.

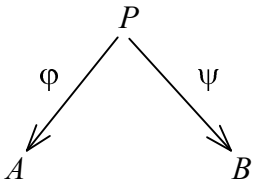
Одноэлементная группа в категории групп является как финальным, так и начальным объектом.

Теорема. *Начальный объект I и финальный объект F, если они существуют, определены с точностью до изоморфизма*.

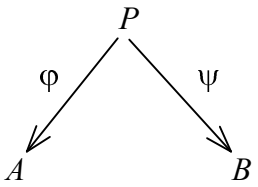
Доказательство. Пусть I и I_1 — два начальных объекта категории $\mathcal{A}$. Так как они начальные, то $\exists f\colon I \to I_1$ и $\exists g\colon I_1 \to I$. Далее, в $\mathrm{Mor}(I, I)$ существует единственный элемент I, а в $\mathrm{Mor}(I_1, I_1)$ существует единственный элемент I_1. Поэтому $f \circ g = I_1$, а $g \circ f = I$ и, следовательно, f изоморфизм. Для финального объекта доказательство то же самое с обращением всех стрелок.

Теорема. *Если в категории $\mathcal{A}$ существует прямое произведение объектов A и B, то оно определено с точностью до изоморфизма*.

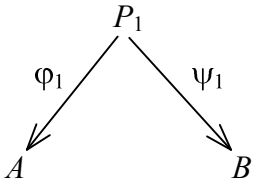
Доказательство. Рассмотрим новую категорию $\mathcal{B}$, взяв в качестве ее объектов диаграммы следующего вида:



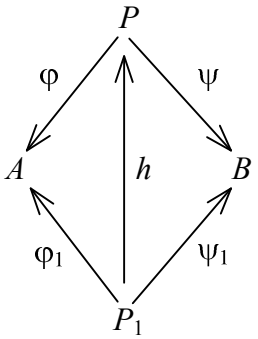
Здесь $P \in \mathrm{Ob}(\mathcal{A})$. Под морфизмом диаграммы



в диаграмму



будем понимать морфизм h: $P \to P_1$, такой что диаграмма



коммутативна. Все аксиомы категории выполнены. Согласно определению, произведение $P = A \times B$ является финальным объектом в категории $\mathcal{B}$, то есть определяется с точностью до изоморфизма.

Однозначность определения прямых сумм получается обращением стрелок. Все операции для конкретных объектов, рассматривае-

мые далее и называемые прямой суммой или прямым произведением (колец, полей, векторных пространств и т. д.), правильны в смысле теории категорий.

Замечание. В категории групп определить понятие суммы не так легко, как понятие произведения. Например, прямой суммой двух циклических групп $\{a^i\}_{i\in Z}$ и $\{b^i\}_{i\in Z}$ будет свободная группа $F\{a,b\}$.

3.6. КОЛЬЦА И ПОЛЯ

В этом разделе мы спускаемся с небес абстрактной теории категорий, чтобы вернуться к объектам, знакомым читателю со школы. Основными примерами колец и полей являются множества целых, рациональных и вещественных чисел с операциями сложения и умножения.

3.6.1. Кольца

Кольцо — это множество M с двумя бинарными операциями + и ∗ (они называются сложением и умножением, соответственно), в котором $\exists 0 \in M \ \ \forall a \ (a + 0 = 0 + a = a)$ существует нуль и выполняются следующие соотношения:

1) $(a + b) + c = a + (b + c)$ сложение ассоциативно;

2) $\exists 0 \in M \ (\forall a \ a + 0 = 0 + a = a)$ существует нуль;

3) $\forall a \ \exists\text{-}a \ (a + \text{-}a = 0)$ существует обратный элемент;

4) $a + b = b + a$ сложение коммутативно, то есть кольцо — абелева группа по сложению;

5) $a * (b * c) = (a * b) * c$ умножение ассоциативно, то есть кольцо — полугруппа по умножению;

6) $a * (b + c) = (a * b) + (a * c)$ умножение дистрибутивно

$(a + b) * c = (a * c) + (b * c)$ слева и справа.

Кольцо называется *коммутативным*, если

7) $a * b = b * a$ умножение коммутативно.

Коммутативное кольцо называется кольцом *с единицей*, если

8) $\exists 1 \in M \ (a * 1 = 1 * a = a)$ существует единица,

то есть кольцо *c* единицей — моноид по умножению.

Теорема. *В кольце выполняются следующие соотношения*:

1) $0 * a = a * 0 = 0$;

2) $a * (-b) = (-a) * b = -(a * b)$;

3) $(-a) * (-b) = a * b$.

Доказательство. $0 * a = (0 + 0) * a = (0 * a) + (0 * a) \Rightarrow$

$\Rightarrow -(0 * a) + (0 * a) = -(0 * a) + ((0 * a) + (0 * a)) =$

$= (-(0 * a) + (0 * a)) + (0 * a) \Rightarrow 0 = 0 + (0 * a) = 0 * a$.

$(a * (-b)) + (a * b) = a * (-b + b) = a * 0 = 0 \Rightarrow$

$\Rightarrow a * (-b) = -(a * b), (a * b) + ((-a) * b) = (a + (-a)) * b = 0 * b = 0 \Rightarrow$

$\Rightarrow (-a) * b = -(a * b)$.

$(-a)*(-b) = -(a*(-b)) = -(-(a * b)) = a * b$. □

Пример. $\langle \mathbb{Z}; +, * \rangle$ — коммутативное кольцо с единицей. Более того, $\forall n \langle \mathbb{Z}_n; +, * \rangle$ — коммутативное кольцо с единицей. В частности, машинная арифметика целых чисел $\langle \mathbb{Z}_{2^{15}}; +, * \rangle$ — коммутативное кольцо с единицей.

3.6.2. Области целостности

Если в кольце $\exists x \neq 0 \; \exists y \neq 0 \; (x * y = 0)$, то *x* называется *левым*, а *y* — *правым делителем нуля*.

Пример. В машинной арифметике $\langle \mathbb{Z}_{2^{15}}; +, * \rangle$ имеем 256*128 =0.

В группе $a * b = a * c \Rightarrow b = c$, однако, в произвольном кольце это не так.

Теорема. *Пусть* $a \neq 0$. *Тогда*

$$((a * b = a * c \Rightarrow b = c) \;\&\; (b * a = c * a \Rightarrow b = c)) \Leftrightarrow$$
$$\Leftrightarrow \forall x \neq 0, y \neq 0 \; (x * y \neq 0).$$

Доказательство.

[$\Rightarrow$] От противного. Пусть $\exists x \neq 0, y \neq 0 \; (x * y = 0)$.

Тогда $x \neq 0$ & $x*y = 0$ & $x*0 = 0 \Rightarrow y = 0$, $y \neq 0$& $x* y = 0$& $0* y = 0 \Rightarrow x = 0$.

[$\Leftarrow$] $0 = (a* b) + (-(a* b)) = (a* b) + (-(a*c))=(a*b) + (a*(-c)) = a*(b + (-c))$, $a*(b + (-c)) = 0$& $a \neq 0 \Rightarrow b + (-c) = 0 \Rightarrow b = c$. □

Коммутативное кольцо с единицей, не имеющее делителей нуля, называется *областью целостности*.

Пример. Целые числа $\langle \mathbb{Z}; +, * \rangle$ являются областью целостности, а машинная арифметика $\langle \mathbb{Z}_{2^{15}}; +, * \rangle$ — не является.

3.6.3. Поля

Поле — это множество M с двумя бинарными операциями + и *, такими что:

1) $(a + b) + c = a + (b + c)$ сложение ассоциативно;

2) $\exists 0 \in M\ (a + 0 = 0 + a = a)$ существует нуль;

3) $\forall a \exists -a (a + -a = 0)$ существует обратный элемент по сложению;

4) $a + b = b + a$ сложение коммутативно,
то есть поле — абелева группа по сложению;

5) $a * (b * c) = (a * b) * c$ умножение ассоциативно;

6) $\exists 1 \in M (a*1 = 1*a = a)$ существует единица;

7) $\forall a \neq 0 \exists a^{-1} (a^{-1}*a = 1)$ существует обратный по умножению;

8) $a * b = b * a$ умножение коммутативно, то есть ненулевые элементы образют абелеву группу по умножению;

9) $a * (b + c) = (a * b) + (a * c)$ умножение дистрибутивно относительно сложения.

Примеры.

1) $\langle \mathbb{R}; +, \cdot \rangle$ — поле вещественных чисел.

2) $\langle \mathbb{Q}; +, \cdot \rangle$ — поле рациональных чисел.

3) Пусть E_2 **:=** $\{0, 1\}$. Определим операции + , $\cdot$: $E_2 \times E_2 \to E_2$ следующим образом: $0 \cdot 0 = 0$, $0 \cdot 1 = 0$, $1 \cdot 0 = 0$, $1 \cdot 1 = 1$ (конъюнкция),

$0 + 0 = 0$, $0 + 1 = 1$, $1 + 0 = 1$, $1 + 1 = 0$ (сложение по модулю 2). Тогда $\mathbb{Z}_2 := \langle E_2; +, \cdot \rangle$ является полем и называется *двоичной арифметикой*.

В двоичной арифметике нуль — это 0, единица — это 1, $-1 := 0$, $-0 := 1$, $1^{-1} := 1$, а 0^{-1} — не определено.

Теорема. *В поле выполняются следующие соотношения*:

1) $(-a) = a * (-1)$;

2) $-(a + b) = (-a) + (-b)$;

3) $a \neq 0 \Rightarrow \{(a^{-1})\}^{-1} = a$;

4) $a * b = 0 \Rightarrow a = 0 \vee b = 0$.

Доказательство.

[1] $(a*(-1)) + a = (a*(-1)) + (a * 1) = a*(-1 + 1) = a* 0 = 0$.

[2] $(a+b) + ((-a)+(-b)) = (a+b) + ((-b)+(-a)) = a + (b+(-b)) + (-a) = a + 0 + (-a) = a + (-a) = 0$.

[3] $a^{-1}*a = 1$.

[4] $a*b = 0 \;\&\; a \neq 0 \Rightarrow b = 1*b = (a^{-1}*a)*b = a^{-1}*(a*b) = a^{-1}*0 = 0$,

$a*b = 0 \& b \neq 0 \Rightarrow a = 1*a = (b^{-1}*b)*a = b^{-1}*(b*a) = b^{-1}*(a*b) = b^{-1}*0 = 0$. □

Теорема. *Если* $a \neq 0$, *то в поле единственным образом разрешимо уравнение* $a * x + b = 0$, (*решение* $x = -(a^{-1}) * b$).

Доказательство.

$a*x + b = 0 \Rightarrow a*x + b + (-b) = 0 + (-b) \Rightarrow a*x + (b + (-b)) = -b \Rightarrow$

$\Rightarrow a*x + 0 = -b \Rightarrow a*x = -b \Rightarrow a^{-1}*(a*x) = a^{-1}*(-b) \Rightarrow$

$\Rightarrow (a^{-1}*a)*x = -(a^{-1}* b) \Rightarrow 1*x = -(a^{-1}*b) \Rightarrow x = -(a^{-1}* b)$. □

3.6.4. Гомоморфизмы колец

У нас уже появлялись морфизмы групп и моноидов, а теперь мы определим гомоморфизмы колец с тем, чтобы кольца образовывали категорию. Пусть R и S — кольца. Гомоморфизмом колец $\Theta: R \to S$ называется отображение, для которого

$$\Theta(x + y) = \Theta(x) + \Theta(y),\ \Theta(x*y) = \Theta(x)*\Theta(y).$$

В категории колец с единицей будем дополнительно требовать $\Theta(1_R) = 1_S$.

Теорема. *Пусть* $\Theta: R \to S$ *— гомоморфизм колец. Тогда* $\Theta(0_R) = 0_S$, $\Theta(-x) = -\Theta(x)$, *и если* x *обратим*, то $\Theta(x^{-1}) = \Theta(x)^{-1}$.

Доказательство. Заметим, что в абелевой группе $a + x = a \Rightarrow x = 0$.

$\Theta(0_R) = \Theta(0_R+0_R) = \Theta(0_R)+\Theta(0_R) \Rightarrow \Theta(0_R) = 0_S$.

$\Theta(-x)+\Theta(x) = \Theta(-x+x) = \Theta(0_R) = 0_S \Rightarrow \Theta(-x) = -\Theta(+x)$.

$\Theta(x)\cdot\Theta(x^{-1}) = \Theta(x\cdot x^{-1}) = \Theta(1_R) = 1_S \Rightarrow \Theta(x^{-1}) = (\Theta(x))^{-1}$. □

Инъективный гомоморфизм будем называть *мономорфизмом* колец, а сюръективный — *эпиморфизмом*. Далее рассматриваются кольца с единицей, если не оговорено обратное. *Характеристикой* кольца R называется порядок подгруппы, порожденной единицей в аддитивной группе $\langle R; + \rangle$, то есть характеристика кольца R — это минимальное натуральное число n, такое что $n\cdot 1 = 0$. Если таких n не существует, то характеристику кольца будем считать равной нулю. В кольце R с ненулевой характеристикой n $\forall a \in R$ $(n\cdot a = a+a+ \ldots +a = 0)$. Действительно, $na = a+a+a+\ldots+a = 1\cdot a+1\cdot a+\ldots+1\cdot a = (1+1+\ldots+1)\cdot a = 0\cdot a = 0$.

3.6.5. Идеалы

Левый идеал $\mathfrak{a}$ кольца A — это аддитивная подгруппа в A (то есть подмножество A, являющееся подгруппой группы по сложению), такая, что $A\mathfrak{a} \subseteq \mathfrak{a}$. *Правый идеал* $\mathfrak{a}$ — это аддитивная подгруппа в A такая, что $\mathfrak{a}A \subseteq \mathfrak{a}$. Так как мы предполагаем, что кольца имеют единицу, то на самом деле $A\mathfrak{a} = \mathfrak{a}$, так как $1\cdot\mathfrak{a} = \mathfrak{a}$. Идеал, содержащий единицу, очевидно, совпадает со всем кольцом. *Двусторонние* идеалы называются просто *идеалами*. Если A — кольцо, а $a \in A$ — некоторый элемент, то $\mathfrak{a} = aA$ есть левый идеал, который называется *главным* идеалом с образующей a. Аналогично, AaA — *главный двусторонний* идеал. В коммутативном кольце всякий левый или правый идеал яв-

ляется двусторонним. Коммутативное кольцо, в котором всякий идеал главный и $1 \neq 0$, называется *кольцом главных идеалов*.

Теорема. $\langle \mathbb{Z}; +, \cdot \rangle$ — *кольцо главных идеалов*.

Доказательство. Пусть $\mathfrak{a}$ — идеал, $\mathfrak{a} \neq \mathbb{Z}$ и $\mathfrak{a} \neq \varnothing$. Если $n \in \mathfrak{a}$, то и $-n \in \mathfrak{a}$, то есть $\mathfrak{a}$ — симметричное относительно 0 множество. Пусть d — наименьшее натуральное число в $\mathfrak{a}$.

Тогда $\forall n \in \mathfrak{a}\ \exists q,r\ (0 \leq r \leq d\ \&\ n = dq + r)$. Так как $\mathfrak{a}$ — идеал, то $r = n - dq \in \mathfrak{a}$, и значит $r = 0$. Таким образом, $\mathfrak{a}$ состоит из всех чисел, кратных d, и наш идеал — главный. Поскольку $\mathfrak{a}$ — произвольный идеал, $\mathbb{Z}$ — кольцо главных идеалов. □

Если $\{\mathfrak{a}_i\}_{i \in I}$ — семейство идеалов, то их пересечение $\bigcap_{i \in I} \mathfrak{a}_i$ — также идеал. То же самое имеем для левых или правых идеалов. Пусть $a_1, a_2, \ldots, a_n$ — элементы кольца A. Обозначим $(a_1, \ldots, a_n)$ идеал, являющийся пересечением всех идеалов, содержащих элементы $a_1, \ldots, a_n$. Мы называем $(a_1, \ldots, a_n)$ идеалом, *порожденным* элементами $a_1, \ldots, a_n$. Это множество состоит из всех элементов кольца A, представимых в виде $x_1 a_1 + x_2 a_2 + \ldots + x_n a_n$, где $x_i \in A$.

Примеры.

1) Множество чётных чисел $2\mathbb{Z}$ является идеалом в кольце $\mathbb{Z}$.

2) $(3, 5) = \mathbb{Z}$, то есть идеал в $\mathbb{Z}$, порожденный элементами 3 и 5 — это все кольцо $\mathbb{Z}$. Действительно, $1 = 3 \cdot 2 - 1 \cdot 5$, следовательно, $1 \in (3, 5)$, и значит, $(3, 5) = \mathbb{Z}$.

3.6.6. Факторкольца

Пусть A и B — кольца, а $f\colon A \to B$ — гомоморфизм колец. Как и для групп, определим *ядро* $\ker f$ следующим образом:

$$\ker f := \{a \in A \mid f(a) = 0\}.$$

Теорема. *Если* $f\colon A \to B$ *— гомоморфизм колец, то* $\ker f$ *— идеал кольца* A.

Доказательство. Проверим, например, что $\ker f$ — подгруппа абелевой группы $(A, +)$. Пусть $x_1, x_2 \in \ker f$, тогда $f(x_1+x_2) = f(x_1) + f(x_2) = 0 + 0 = 0 \Rightarrow x_1 + x_2 \in \ker f$. Пусть теперь $x \in \ker f$ и $a \in A$. Тогда $f(ax) = f(a)\cdot f(x) = f(a)\cdot 0 = 0$, то есть $A \ker f \subseteq \ker f$. □

Таким образом, ядра гомоморфизмов колец являются идеалами. Обратно, пусть $\mathfrak{a}$ — идеал кольца A. Построим *факторкольцо* $A/\mathfrak{a}$ следующим образом. Рассмотрим на A отношение эквивалентности $x \equiv y := x - y \in \mathfrak{a}$. Обозначим через $A/\mathfrak{a}$ множество классов эквивалентности по этому отношению, $A/\mathfrak{a} := A/\equiv$. Каждый класс состоит из всех элементов $x+\mathfrak{a}$ для некоторого $x \in A$. Пусть $x_1+\mathfrak{a}$ и $x_2+\mathfrak{a}$ — два класса эквивалентности. Определим операции сложения и умножения классов эквивалентности: $(x_1+\mathfrak{a})+(x_2+\mathfrak{a}) := x_1+x_2+\mathfrak{a}$, $(x_1+\mathfrak{a})*(x_2+\mathfrak{a}) := x_1 x_2+\mathfrak{a}$.

Легко проверить, что таким образом определенные сложение и умножение не зависят от выбора представителей в классах $x_1+\mathfrak{a}$ и $x_2+\mathfrak{a}$ и удовлетворяют аксиомам кольца.

3.6.7. Прямые произведения колец

Пусть $\mathcal{A}$ и $\mathcal{B}$ — кольца с единицей. *Прямым произведением колец* $\mathcal{A}$ и $\mathcal{B}$ назовем декартово произведение $A \times B$ с покомпонентными умножением и сложением:

$(a_1, b_1) + (a_2, b_2) := (a_1 + a_2, b_1 + b_2)$,

$(a_1, b_1) * (a_2, b_2) := (a_1\, a_2, b_1\, b_2)$.

Единицей прямого произведения служит элемент $(1_A, 1_B)$. Так определенное произведение является прямым произведением в смысле теории категорий.

3.6.8. Простые и максимальные идеалы

Далее мы рассматриваем только коммутативные кольца с единицей. Пусть $\mathcal{A}$ — кольцо и $\mathfrak{p}$ — его идеал. Идеал $\mathfrak{p}$ называется *простым*, если $x \in \mathcal{A} \,\&\, y \in \mathcal{A} \,\&\, xy \in \mathfrak{p} \Rightarrow (x \in \mathfrak{p}) \vee (y \in \mathfrak{p})$.

Теорема. *Если $\mathfrak{p}$ — простой идеал, то $\mathcal{A}/\mathfrak{p}$ — область целостности*.

Доказательство. Если существуют x и y, такие что $x \notin \mathfrak{p} \,\&\, y \notin \mathfrak{p}$, но $xy \in \mathfrak{p}$, то классы $x + \mathfrak{p}$ и $y + \mathfrak{p}$ — ненулевые элементы в $A/\mathfrak{p}$ с нулевым произведением, то есть в $A/\mathfrak{p}$ они являются делителями нуля.

Пусть $\mathfrak{m}$ — идеал в $\mathcal{A}$. Идеал $\mathfrak{m}$ называется *максимальным*, если $\mathfrak{m} \neq \mathcal{A}$ и не существует идеала $\mathfrak{a}$, промежуточного между $\mathfrak{m}$ и $\mathcal{A}$.

$$\mathfrak{m} \text{ — максимальный } := \mathfrak{a} \text{ — идеал } \& \ \mathfrak{m} \subseteq \mathfrak{a} \ \Rightarrow \mathfrak{m} = \mathcal{A}.$$

Теорема. *Всякий максимальный идеал прост*.

Доказательство. Пусть $\mathfrak{m}$ — максимальный идеал, и пусть $\exists x, y \in A\ (xy \in \mathfrak{m})$. Пусть $x \in \mathfrak{m}$. Пусть $\mathfrak{m} + \mathcal{A}x$ — идеал, строго содержащий $\mathfrak{m}$ и, следовательно, равный $\mathcal{A}$. Так как он содержит 1, то $1 = u + ax$, где $u \in \mathfrak{m}$, $a \in \mathcal{A}$. Умножим последнее равенство на y, получим $y = yu + axy$, отсюда $y \in \mathfrak{m}$ и, следовательно, идеал $\mathfrak{m}$ — простой. □

Пример. В $\mathbb{Z}$ простыми и одновременно максимальными идеалами являются идеалы вида $p\mathbb{Z}$, где p — простое число. Факторкольцо $\mathbb{Z}/p\mathbb{Z}$ — это кольцо вычетов по модулю p.

3.7. ВЕКТОРНЫЕ ПРОСТРАНСТВА И МОДУЛИ

Понятие векторного пространства должно быть известно читателю из курса средней школы и других математических курсов. Обычно это понятие ассоциируется с геометрической интерпретацией векторов в пространствах $\mathbb{R}^2$ и $\mathbb{R}^3$. В этом разделе даны и другие примеры векторных пространств, которые используются в последующих главах для решения задач, весьма далеких от геометрической интерпретации.

3.7.1. Векторное пространство

Пусть $\mathcal{F} = \langle F\ ;\ +,\ \cdot \rangle$ — некоторое поле с операцией сложения +, операцией умножения $\cdot$, *аддитивной* единицей 0 и *мультипликативной* единицей 1. Пусть $\mathcal{V} = \langle V;\ + \rangle$ — некоторая абелева группа с операцией + и единицей **0**. Если существует операция $F \times V \to V$ (знак этой операции опускается), такая что для любых $a, b \in F$ и для любых $\mathbf{x}, \mathbf{y} \in V$ выполняются соотношения:

1) $(a + b)\mathbf{x} = a\mathbf{x} + b\mathbf{x}$,
2) $a(\mathbf{x} + \mathbf{y}) = a\mathbf{x} + a\mathbf{y}$,
3) $(a \cdot b)\mathbf{x} = a(b\mathbf{x})$,
4) $1\mathbf{x} = \mathbf{x}$,

то $\mathcal{V}$ называется *векторным пространством* над полем $\mathcal{F}$, элементы F называются *скалярами*, элементы V называются *векторами*, а необозначенная операция $F \times V \to V$ называется *умножением вектора на скаляр*.

Примеры.

1) Пусть $\mathcal{F} = \langle F; +, \cdot \rangle$ — некоторое поле. Рассмотрим множество кортежей F^n. Тогда $\mathcal{F}^n = \langle F^n; + \rangle$, где

$$(a_1, \ldots, a_n) + (b_1, \ldots, b_n) := (a_1 + b_1, \ldots, a_n + b_n)$$

является абелевой группой, если положить

$$-(a_1, \ldots, a_n) := (-a_1, \ldots, -a_n) \text{ и } \mathbf{0} := (0, \ldots, 0).$$

Положим

$$a(a_1, \ldots, a_n) := (a \cdot a_1, \ldots, a \cdot a_n).$$

Тогда $\mathcal{F}^n$ является векторным пространством над $\mathcal{F}$ для любого (конечного) n. В частности, $\mathbb{R}^n$ является векторным пространством для любого n. Векторное пространство $\mathbb{R}^2$ (и $\mathbb{R}^3$) имеет естественную геометрическую интерпретацию (рис. 3.1).

2) Двоичная арифметика $\mathbb{Z}_2 = \langle E_2; +_2, \cdot \rangle$ является полем, а булеан $\langle 2^M; \quad \rangle$ с симметрической разностью является абелевой группой. Положим $1X := X$, $0X := \varnothing$. Тогда булеан с симметрической разностью является векторным пространством над двоичной арифметикой.

3) Пусть $X = \{x_1, \ldots, x_n\}$ — произвольное множество, а $\mathcal{F}$ — поле. Рассмотрим множество V_X всех формальных конечных сумм вида $\sum_{x \in X} \lambda_x x$, где $\lambda_x \in F$. Положим $\sum_{x \in X} \lambda_x x + \sum_{x \in X} \mu_x x$ & := $= \sum_{x \in X} (\lambda_x + \mu_x) x$, $\alpha \sum_{x \in X} \lambda_x x := \sum_{x \in X} (\alpha \lambda_x) x$, где $\alpha, \lambda_x, \mu_x \in F$.

Ясно, что V_X является векторным пространством, а X является его множеством образующих. В таком случае говорят, что V_X — это векторное пространство, "натянутое" на множество X. Можно дока-

зать, что V_X является прямой суммой $|X|$ экземпляров поля $\mathcal{F}$ в категории векторных пространств над $\mathcal{F}$.

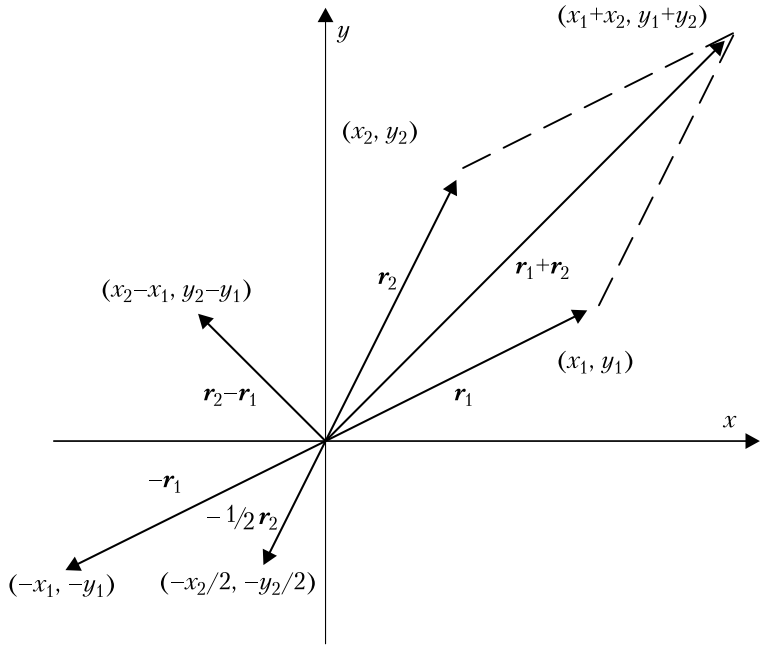


Рис. 3.1. Операции над векторами в $\mathbb{R}^2$

3.7.2. Свойства нуль-вектора

Единица группы $\mathcal{V}$ (если $\mathcal{V}$ — векторное пространство) называется *нуль-вектором* и обозначается **0**.

Теорема. 1) $\forall \mathbf{x} \in V\,(0\mathbf{x} = \mathbf{0})$, 2) $\forall a \in F\,(a\mathbf{0} = \mathbf{0})$.

Доказательство.

[1] $0\mathbf{x} = (1 - 1)\mathbf{x} = 1\mathbf{x} - 1\mathbf{x} = \mathbf{x} - \mathbf{x} = \mathbf{0}$.

[2] $a\mathbf{0} = a(\mathbf{0} - \mathbf{0}) = a\mathbf{0} - a\mathbf{0} = (a - a)\mathbf{0} = 0\mathbf{0} = \mathbf{0}$. □

3.7.3. Линейные комбинации

Если $\mathcal{V}$ — векторное пространство над полем $\{\mathcal{F}\}$, S — некоторое множество векторов, $S \subseteq V$, то сумма вида $\sum_{i=1}^{n} a_i\mathbf{x}_i$, $a_i \in F$, $\mathbf{x}_i \in S$

называется *линейной комбинацией* векторов из S. Пусть $X = \{\mathbf{x}_1,\ldots,\mathbf{x}_k\}$ — конечное множество векторов. Если

$$\sum_{i=1}^{k} a_i\mathbf{x}_i = \mathbf{0} \Rightarrow a_1 = a_2 = \ldots = a_k = 0,$$

то множество X называется *линейно независимым*. В противном случае, то есть если

$$\exists a_1, \ldots, a_k \in F\ (\vee_{i=1}^{k} a_i \neq 0\ \&\ \sum_{i=1}^{k} a_i\mathbf{x}_i = \mathbf{0}),$$

то множество X называется *линейно зависимым*.

Теорема. *Линейно независимое множество векторов не содержит нуль-вектора*.

Доказательство. От противного. Пусть линейно независимое множество $S = \{\mathbf{0}, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ содержит нуль-вектор и для определенности $\mathbf{x}_1 = \mathbf{0}$. Положим $a_1 := 1$, $a_2 := 0$ и т. д., $a_k := 0$. Тогда имеем $1\mathbf{0} + 0\mathbf{x}_2 + \ldots + 0\mathbf{x}_k = \mathbf{0} + \mathbf{0} + \ldots + \mathbf{0} = \mathbf{0}$. □

3.7.4. Базис и размерность

Подмножество векторов $S \in V$, такое, что любой элемент V может быть представлен в виде линейной комбинации элементов S, называется *порождающим* множеством пространства $\mathcal{V}$. Конечное линейно независимое порождающее множество называется *базисом* векторного пространства. Пусть векторное пространство $\mathcal{V}$ имеет базис $B = \{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$.

Теорема 1. *Каждый элемент векторного пространства имеет единственное представление в данном базисе*.

Доказательство. Пусть $\mathbf{x} = \sum_{i=1}^{n} a_i\mathbf{e}_i$ и $\mathbf{x} = \sum_{i=1}^{n} b_i\mathbf{e}_i$.

Тогда $\mathbf{0} = \mathbf{x}–\mathbf{x} = \sum_{i=1}^{n} a_i\mathbf{e}_i – \sum_{i=1}^{n} b_i\mathbf{e}_i = \sum_{i=1}^{n} (a_i–b_i)\mathbf{e}_i \Rightarrow$

$\Rightarrow \forall i \in 1..n\ (a_i–b_i = 0) \Rightarrow \forall i\ (a_i = b_i)$. □

Коэффициенты разложения вектора в данном базисе называются его *координатами*.

Теорема 2. *Пусть* $B = \{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$ — *базис*, *а* $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ — *линейно независимое множество пространства* $\mathcal{V}$. *Тогда* $n \geq m$.

Доказательство. От противного. Пусть $n < m$ и B — базис. Тогда $\exists\ a_1, \ldots, a_k \in F$ $(\mathbf{x}_1 = a_1\mathbf{e}_1 + \ldots + a_n\mathbf{e}_n)$. Поскольку $\mathbf{x}_1 \neq \mathbf{0}$ имеем $\vee_{i=1}^{n} a_i \neq 0$. Пусть для определенности $a_1 \neq 0$. Тогда (свойства поля)

$$\mathbf{e}_1 = a_1^{-1}\mathbf{x}_1 - (a_1^{-1}a_2)\mathbf{e}_2 - (a_1^{-1}a_n)\mathbf{e}_n.$$

Так как B порождает $\mathcal{V}$, то и $\{\mathbf{x}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$ тоже порождает $\mathcal{V}$. Аналогично $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{e}_3, \ldots, \mathbf{e}_n\}$ порождает $\mathcal{V}$. Продолжая процесс, получаем, что $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ порождает $\mathcal{V}$. Следовательно,

$$\mathbf{x}_{n+1} = \sum_{i=1}^{n} b_i\mathbf{x}_i \,\&\, \vee_{i=1}^{n} b_i \neq 0 \Rightarrow \mathbf{x}_{n+1} - \sum_{i=1}^{n} b_i\mathbf{x}_i = \mathbf{0}$$

и X — линейно зависимое множество. □

Теорема 3. *Пусть* B_1 *и* B_2 — *базисы векторного пространства* $\mathcal{V}$, *тогда* $|B_1| = |B_2|$.

Доказательство. B_1 — базис, и B_2 — линейно независимое множество. Следовательно, $|B_1| \geq |B_2|$. С другой стороны, B_2 — базис, и B_1 — линейно независимое множество. Следовательно, $|B_2| \geq |B_1|$. Имеем $|B_1| = |B_2|$. □

Если векторное пространство $\mathcal{V}$ имеет базис B, то количество элементов в базисе называется *размерностью* векторного пространства и обозначается $\dim(\mathcal{V})$. Векторное пространство, имеющее конечный базис, называется *конечномерным*. Векторное пространство, не имеющее базиса (в смысле приведенного определения), называется *бесконечномерным*.

Примеры.

1) Одноэлементные подмножества образуют базис булеана, $\dim 2^M = |M|$.

2) Кортежи вида (0, ..., 1, 0, ..., 0) образуют базис пространства $\mathcal{F}^n$, $\dim\mathcal{F}^n = n$.

Замечание. Если определить базис как максимальное линейно-независимое множество (то есть множество, которое нельзя расширить, не нарушив свойства линейной независимости), то понятие базиса можно распространить и на бесконечномерные пространства.

3.7.5. Модули

Понятие модуля во многом аналогично понятию векторного пространства, с той лишь разницей, что векторы умножаются не на элементы из поля, а на элементы из произвольного кольца. Пусть $\mathcal{R} = \langle R; +, * \rangle$ — некоторое кольцо с операцией сложения +, операцией умножения $*$, нулевым элементом 0 и единичным элементом 1. Абелева группа $\mathcal{M} = \langle M; + \rangle$ называется *модулем* над кольцом $\mathcal{R}$, если задана операция умножения на скаляр $R \times M \to M$ (никак не обозначаемая), такая что:

1) $(a + b)\mathbf{x} = a\mathbf{x} + b\mathbf{x}$,

2) $a(\mathbf{x} + \mathbf{y}) = a\mathbf{x} + a\mathbf{y}$,

3) $(a * b)\mathbf{x} = a(b\mathbf{x})$,

4) $1 * \mathbf{x} = \mathbf{x}$,

где $a, b \in R$, а $\mathbf{x}, \mathbf{y} \in M$. Как и в случае векторных пространств, элементы кольца $\mathcal{R}$ называются *скалярами*.

Примеры.

1) Векторное пространство $\mathcal{V}$ над полем $\mathcal{F}$ является модулем над $\mathcal{F}$, так как поле, очевидно, является кольцом.

2) Множество векторов с целочисленными координатами в $\mathbb{R}^n$ является модулем над кольцом $\mathbb{Z}$.

3) Любая абелева группа есть модуль над кольцом $\mathbb{Z}$, если операцию умножения на скаляр $n \in \mathbb{Z}$ определить следующим образом: $n\mathbf{v} := \mathbf{v} + \mathbf{v} + ... + \mathbf{v}$, n раз.

Теорема. *Любой идеал кольца R является модулем над кольцом R.*

Доказательство. Пусть $\mathfrak{a} \subseteq \mathcal{R}$ — идеал. Он является абелевой группой, так что $c \in R \,\&\, a \in \mathfrak{a} \Rightarrow ca \in \mathfrak{a}$ по определению идеала. □

Следствие. *Любое кольцо является модулем над собой.*

3.7.6. Прямая сумма модулей

Пусть $\mathcal{R}$ — кольцо, а $\mathcal{M}_1$ и $\mathcal{M}_2$ — два модуля над $\mathcal{R}$. Декартово произведение $M_1 \times M_2$ с покомпонентной операцией сложения $(v_1, v_2) + (w_1, w_2) := (v_1 + w_1, v_2 + w_2)$ и операцией умножения на скаляр $r(v_1, v_2) := (rv_1, rv_2)$ называется *прямой суммой* $\mathcal{M}_1$ и $\mathcal{M}_2$ и обозначается $\mathcal{M}_1 \oplus \mathcal{M}_2$. Каждый из модулей $\mathcal{M}_1$ и $\mathcal{M}_2$ каноническим образом вкладывается в прямую сумму $\mathcal{M}_1 \oplus \mathcal{M}_2$ с помощью отображений $i_1: \mathcal{M}_1 \to \mathcal{M}_1 \oplus \mathcal{M}_2$ и $i_2: \mathcal{M}_2 \to \mathcal{M}_1 \oplus \mathcal{M}_2$: $i_1(m_1) := (m_1, 0)$, $i_2(m_2) := (0, m_2)$.

Таким образом, прямая сумма модулей правильно определена в смысле теории категорий, так как эти два отображения удовлетворяют условиям раздела 3.4. Прямая сумма $\mathcal{R} \oplus \mathcal{R} \oplus ... \oplus \mathcal{R}$ (n раз) называется *свободным* модулем ранга n над кольцом $\mathcal{R}$. Свободные модули являются естественным обобщением понятия векторного простанства.

3.8. РЕШЕТКИ

Решетки иногда называют "структурами", но слово "структура" перегружено, и мы не будем использовать его в этом значении. Решетки сами по себе часто встречаются в разных программистских за-

дачах, но еще важнее то, что понятие решетки непосредственно подводит нас к понятию булевой алгебры, значение которой для основ современной двоичной компьютерной техники трудно переоценить.

3.8.1. Операции решётки

Решётка — это множество M с двумя бинарными операциями $\cap$ и $\cup$, такими что выполнены следующие условия (аксиомы решетки):

1) идемпотентность:

$$a \cap a = a,\ a \cup a = a;$$

2) коммутативность:

$$a \cap b = b \cap a,\ a \cup b = b \cup a;$$

3) ассоциативность:

$$(a \cap b) \cap c = a \cap (b \cap c),\ (a \cup b) \cup c = a \cup (b \cup c);$$

4) поглощение:

$$(a \cap b) \cup a = a,\ (a \cup b) \cap a = a;$$

5) Решетка называется *дистрибутивной*, если

$$a \cup (b \cap c) = (a \cup b) \cap (a \cup c),\ a \cap (b \cup c) = (a \cap b) \cup (a \cap c).$$

3.8.2. Ограниченные решетки

Если в решетке $\exists 0 \in M\ (\forall a(0 \cap a = 0))$, то 0 называется *нулем* (или *нижней гранью*) решетки. Если в решетке $\exists 1 \in M\ (\forall a\ (1 \cup a = 1))$, то 1 называется *единицей* (или *верхней гранью*) решетки. Решетка с верхней и нижней гранями называется *ограниченной*.

Теорема. *Если нижняя (верхняя) грань существует, то она единственна*.

Доказательство. Пусть 0' — еще один нуль решетки. Тогда $0 \cap 0' = 0'$ и $0' \cap 0 = 0$. Следовательно, $0 = 0'$. □

Теорема. $a \cap b = b \Leftrightarrow a \cup b = a$.

Доказательство.

[$\Rightarrow$] Пусть $a \cap b = b$. Тогда $a \cup b = a \cup (a \cap b) = (a \cap b) \cup a = a$.

[$\Leftarrow$] Пусть $a \cup b = a$. Тогда $a \cap b = (a \cup b) \cap b = (b \cup a) \cap b = b$. □

Следствие. $0 \cap a = 0 \Leftrightarrow 0 \cup a = a$, $1 \cup a = 1 \Leftrightarrow 1 \cap a = a$.

3.8.3. Решетка с дополнением

В ограниченной решетке элемент a' называется *дополнением* элемента a, если $a \cap a' = 0$ и $a \cup a' = 1$. Если

$$\forall a \in M\,(\exists a \in M\,(a \cap a' = 0\ \&\ a \cup a' = 1)),$$

то ограниченная решетка называется *решеткой с дополнением*. Вообще говоря, дополнение не обязано существовать и не обязано быть единственным.

Теорема [о свойствах дополнения]. *В ограниченной дистрибутивной решетке с дополнением выполняется следующее*:

1) *дополнение a' единственно*;

2) *дополнение инволютивно*: $a'' = a$;

3) *грани дополняют друг друга*: $1' = 0$, $0' = 1$;

4) *выполняются законы де Моргана*: $(a \cup b)' = a' \cap b'$, $(a \cap)' = a' \cup b'$.

Доказательство.

[1] Пусть x, y — дополнения a. Тогда $a \cap x = 0$, $a \cup x = 1$, $a \cap y = 0$, $a \cup y = 1$. Имеем: $(x = x \cap 1 = x \cap (a \cup y) = (x \cap a) \cup (x \cap y) = 0 \cup (x \cap y) = x \cap y$, $y = y \cap 1 = y \cap (a \cup x) = (y \cap a) \cup (y \cap x) = 0 \cup (y \cap x) = y \cap x = x \cap y) \Rightarrow x = y$.

[2] $(a \cup a' = 1 \Rightarrow a' \cup a = 1,\ a \cap a' = 0 \Rightarrow a' \cap a = 0) \Rightarrow a = a''$.

[3] $(1 \cap 0 = 0,\ 0' \cap 0 = 0) \Rightarrow 1 = 0'$, $(1 \cup 0 = 1,\ 1 \cup 1' = 1) \Rightarrow 0 = 1'$.

$(a \cap b) \cap (a' \cup b') = (a \cap b \cap a') \cup (a \cap b \cap b') = (0 \cap b) \cup (a \cap 0) = 0 \cup 0 = 0$, $(a \cap b) \cup (a' \cup b') = (a \cup a' \cup b') \cap (b' \cup a' \cup b') = (1 \cup b') \cap (1 \cup a') = 1 \cap 1 = 1$. □

3.8.4. Частичный порядок в решетке

В любой решетке можно естественным образом ввести частичный порядок, а именно: $a \prec b := a \cap b = a$.

Теорема 1. *Пусть $a \prec b$ **:=** $a \cap b = a$. Тогда $\prec$ является отношением частичного порядка*.

Доказательство.

[Рефлексивность] $a \cap a = a \Rightarrow a \prec a$.

[Антисимметричность] $a \prec b$ & $b \prec a \Rightarrow\ a \cap b = a$ & $b \cap a = b \Rightarrow$

$\Rightarrow a = a \cap b = b \cap a = b$.

[Транзитивность] $a \prec b$ & $b \prec c \Rightarrow a \cap b = a$ & $b \cap c = b \Rightarrow$

$\Rightarrow a \cap c = (a \cap b) \cap c = a \cap (b \cap c) = a \cap b = a \Rightarrow a \prec c$. □

Наличие частичного порядка в решетке не случайно, это ее характеристическое свойство. Более того, обычно решетку определяют, начиная с частичного порядка, следующим образом. Пусть M — частично упорядоченное множество с частичным порядком $\prec$. Элемент x называется *нижней границей* для a и b, если $x \prec a$ & $x \prec b$. Аналогично, y называется *верхней границей* для a и b, если $a \prec y$ & $b \prec y$. Элемент x называется *нижней гранью* (наибольшей нижней границей) элементов a и b, если x — нижняя граница элементов a и b и для любой другой нижней границы v элементов a и b выполняется $v \prec x$. Обозначение: $x = \inf(a, b)$. Аналогично, y называется *верхней гранью* (наименьшей верхней границей) элементов a и b, если y — верхняя граница элементов a и b и для любой другой верхней границы u элементов a и b выполняется $y \prec u$. Обозначение: $y = \sup(a, b)$.

Теорема 2. *Если нижняя (верхняя) грань существует, то она единственна*.

Доказательство. $x = \inf(a, b)$ & $y = \inf(a, b) \Rightarrow y \prec x$ & $x \prec y \Rightarrow x = y$.

Теорема 3. *Если в частично упорядоченном множестве для любых двух элементов существуют нижняя и верхняя грани, то это множество образует решетку относительно* inf *и* sup (*то есть* $x \cap y$ **:=** $\inf(x, y)$, $x \cup y$ **:=** $\sup(x, y)$).

Доказательство.

[1] $\inf(x, x) = x \Rightarrow x \cap x = x$, $\sup(x, x) = x \Rightarrow x \cup x = x$;

[2] $\inf(x, y) = \inf(y, x) \Rightarrow x \cap y = y \cap x$, $\sup(x, y) = \sup(y, x) \Rightarrow x \cup y = y \cup x$;

[3] $\inf(x, \inf(y, z)) = \inf(\inf(x, y), z) \Rightarrow x \cap (y \cap z) = (x \cap y) \cap z$,
$\sup(x, \sup(y, z)) = \sup(\sup(x, y), z) \Rightarrow x \cup (y \cup z) = (x \cup y) \cup z$;
[4] $\sup(\inf(a, b), a) = a \Rightarrow (a \cap b) \cup a = a$,
$\inf(\sup(a, b), a) = a \Rightarrow (a \cup b) \cap a = a$. □

3.8.5. Булевы алгебры

Дистрибутивная ограниченная решетка, в которой для каждого элемента существует дополнение, называется *булевой алгеброй*. Свойства булевой алгебры: \

1) $a \cup a = a$, $a \cap a = a$

по определению решетки;

2) $a \cup b = b \cup a$, $a \cap b = b \cap a$

по определению решетки;

3) $a \cup (b \cup c) = (a \cup b) \cup c$, $a \cap (b \cap c) = (a \cap b) \cap c$

по определению решетки;

4) $(a \cap b) \cup a = a$, $(a \cup b) \cap a = a$

по определению решетки;

5) $a \cup (b \cap c) = (a \cup b) \cap (a \cup c)$, $a \cap (b \cup c) = (a \cap b) \cup (a \cap c)$

по свойству дистрибутивности;

6) $a \cup 1 = 1$, $a \cap 0 = 0$

по свойству ограниченности;

7) $a \cup 0 = a$, $a \cap 1 = a$

по следствию из теоремы ограниченности;

8) $a'' = a$

по теореме о свойствах дополнения;

9) $(a \cap b)' = a' \cup b'$, $(a \cup b)' = a' \cap b'$

по теореме о свойствах дополнения;

10) $a \cup a' = 1$, $a \cap a' = 0$

так как дополнение существует.

Примеры.

1) $\langle 2^M;\ \cap, \cup, ' \rangle$ — булева алгебра, причем M — верхняя грань, $\varnothing$ — нижняя грань, $\subseteq$ — естественный частичный порядок.

2) $\langle E_2; \&, \vee, \neg \rangle$ — булева алгебра, причем 1 — верхняя грань, 0 — нижняя грань, $\Rightarrow$ — естественный частичный порядок.

3.9. КОМПЬЮТЕРНАЯ АЛГЕБРА

Долгое время основными объектами вычислений в компьютере являлись целые и вещественные числа фиксированной разрядности, откуда и произошло название: компьютер — это вычислительная машина. Фиксированная разрядность влечёт ограниченную точность вычислений, и вычисления (особенно с вещественными числами) оказываются приближёнными. Аппаратура процессора действительно имеет фиксированную разрядность, но программное обеспечение позволяет преодолеть это ограничение. Компьютер может не только приближенно производить арифметические действия с числами, но и выполнять так называемые *символьные вычисления*.

Пример. Символьное решение уравнение с буквенными коэфициентами намного ценнее приближённого отыскания корня для конкретных числовых значений коэффициентов. Действительно, символьное решение выявляет зависимость корня от коэффициентов, что само по себе может быть полезным знанием, и в то же время позволяет легко и быстро вычислить значение корня для заданных значений коэффициентов. Численное решение уравнения даёт приближенное значение корня для конкретных значений коэффициентов и более ничего.

Компьютерная алгебра изучает способы представления и алгоритмы операций над разнообразными математическими объектами, заданными в форме символьных выражений. Слово "алгебра" в названии "компьютерная алгебра" используется традиционно, хотя современные системы компьютерной алгебры умеют работать не только с алгебраическими структурами, рассмотренными в этой главе, но и практически с любыми математическими объектами.

3.9.1. Системы компьютерной алгебры

Следующие средства обычно включаются в состав ядра современной системы компьютерной алгебры.

1) Операции с целыми, рациональными, вещественными и комплексными числами с неограниченной точностью. Неограниченная точность означает, что операции выполняются без потери точности (без округлений).

2) Операции с полиномами от одной или нескольких переменных. К операциям относятся не только сложение и умножение, но и вычисление наибольшего общего делителя, разложение на множители, решение систем полиномиальных уравнений и т. д.

3) Вычисления с матрицами с символьными или числовыми коэффициентами. Помимо сложения и умножения обычно реализуют обращение, транспонирование, вычисление определителей, собственных значений и т. д.

4) Операции с функциями, заданными выражениями. К таким операциям относятся символьное дифференцирование и интегрирование, разложение в ряды, аппроксимация и т. д.

5) Преобразование формул, упрощение выражений, подстановки, вычисление значений.

Кроме того, могут быть предусмотрены возможности оперирования с другими объектами, например, группы, кольца, тензоры, алгебраичекие кривые и поверхности и многое другое. Конечно, в коротком разделе невозможно описать представление всех типов данных, которые используются в компьютерной алгебре, но на примере важнейших объектов мы показываем основные принципы, используемые и для представления более сложных типов.

3.9.2. Длинные целые

Первым типом объектов являются *длинные целые числа*, представляющие целые числа произвольной (неограниченной) разрядности.

Замечание. На самом деле разрядность длинных целых также ограничена, она не может быть бесконечной, потому что в компьютере не может быть ничего бесконечного, но разрядность ограничена не ёмкостью регистров процессора, а размером памяти, отводимой для размещения данных. В современных компьютерах это означает увеличение допустимой разрядности в миллиарды раз, что оправдывает использование эпитета "неограниченный".

Длинное целое можно представить записью типа *L*, состоящей из знака числа, его длины в словах, и массива слов переменной длины:

type *L* = **record**

s: {–1, 1} // знак числа

n: **integer** // количество "цифр"

w: **array** [1..∞] of **integer** // массив "цифр"

end record

Замечание. Использование массива переменной длины, который обозначен **array** [1..∞], означает, что память для длинного целого числа распределяется динамически, во время выполнения, и не может быть распределена статически, во время компиляции. Аналогичное обстоятельство имеет место для большинства объектов компьютерной алгебры.

Если длина одного слова, отводимого для хранения значения типа **integer** составляет *k* битов, то представленную структуру данных можно рассматривать как запись числа *N* в системе счисления по основанию 2^k:

$$N = \sum_{i=0}^{n-1} (2^k)^i * w[i],$$

где значения *w*[*i*] являются *цифрами* этой системы счисления.

Замечание. Значения *n* и *w*[*i*] заведомо неотрицательны. Поэтому если система программирования допускает целые без знака (**unsigned integer**), то их использование позволяет несколько экономить память.

Алгоритмы обычных арифметических действий для длинных целых чисел не столь просты, как для обычных целых. Рассмотрим, для примера, сложение. Для выполнения сложения нужно иметь возможность складывать две цифры (в системе счисления по основанию 2^k) и получать их сумму s и, возможно, значение p переноса единицы в старший разряд ().

Алгоритм 3.1. Сложение длинных целых

Вход: два длинных целых числа M и N одного знака, представленных соответствующими записями.

Выход: длинное целое число K, $K = M + N$.

$s := (M.w[i] + N.w[i] + p)$ **mod** 2^k

$p := (M.w[i] + N.w[i] + p)$ **div** 2^k

При выполнении умножения имеет место эффект быстрого роста длины промежуточных результатов, поскольку длина произведения пропорциональна сумме длин сомножителей. В этой ситуации наивные алгоритмы умножения и деления чисел (умножение "в столбик" и деление "уголком") выполняются слишком долго и на практике не применяются.

3.9.3. Длинные рациональные

Рациональное число $a = p/q$ неограниченной разрядности можно представить парой длинных целых (p, q), второе из которых положительно. Легко видеть, что при этом каждое рациональное число имеет бесконечное множество различных представлений:

$$1/2 = 2/4 = 3/6 = ...$$

Выполнению ариметических операций неединственность представления не препятствует:

$$(p_1, q_1) + (p_2, q_2) := (p_1*q_2 + p_2*q_1, q_1*q_2),$$

при этом полученная пара, очевидно, является одним из представлений суммы рациональных чисел. Однако неединственность представления порождает другие проблемы, например, проверку равенства.

Если потребовать несократимости рациональной дроби p/q (то есть взаимной простоты p и q), то каждое рациональное число будет представлено единственным образом. Представление, которое является единственным для каждого объекта, называется *каноническим*.

Пример. Представление подмножеств заданного перенумерованного множества с помощью битовых шкал (см. 1.3) является каноническим.

Однако нахождение канонического представления для рационального числа требует нахождения наибольшего делителя и знаменателя. На практике часто оказывается выгоднее проводить промежуточные выкладки, не вычисляя наибольшего общего делителя, и приводить к каноническому виду только окончательный результат. Имея представления для чисел из $\mathbb{Z}$ и $\mathbb{Q}$ можно легко построить и более сложные объекты, например, векторы и матрицы с целыми и рациональными коэффициентами.

3.9.4. Представления полиномов

Следующим по сложности объектом являются полиномы от одной или нескольких переменных с целыми или рациональными коэффициентами. В компьютерной алгебре различают два типа представлений: плотные и разреженные. Представление называется *разреженным*, если в нём не присутствуют явно нулевые составляющие. В противном случае, то есть если хранятся все составлющие, представление называется *плотным*.

Пример. Представление множеств битовыми шкалами является плотным, а упорядоченными списками — разреженным.

Наиболее простым плотным представлением полинома

$$f = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$$

является его представление массивом коэффициентов

array [0..*n*] **of integer**.

В случае нескольких переменных можно использовать многомерный массив. Нетрудно видеть, что для хранения полинома степени n от m переменных потребуется хранить $m(n + 1)$ чисел, вне зависимости от того, сколько слагаемых фактически присутствует в полиноме. По этой причине плотное представление полиномов имеет ограниченное употребление. Для построения разреженного предсставления используются списочные структуры. Пусть задан полином

$$P = \sum a_{i_1\, i_2 \ldots\, i_n} \cdot x_1^{\,i_1} \cdot x_2^{\,i_2} \cdot \ldots \cdot x_n^{\,i_n}.$$

Введем обозначения. Степень полинома P по переменной x_s обозначим $d(s)$. Таким образом, $\forall s \in 1..n\ (i_s \in 1..d(s))$. Вектор индексов $(i_1, \ldots, i_n)$ называется *мультиндексом* и обозначается одной буквой I. Если $X = (x_1, ..., x_n)$ — вектор переменных, а I — мультииндекс, то X^I означает произведение переменных в соответствующих степенях.

В таких обозначениях весь полином P записывают в виде

$$P = \sum_{I \in M} a_I X^I,$$

где I пробегает некоторое множество мультииндексов. Каждое слагаемое $a_I X^I$ называют *мономом*. Полином P можно представить в виде упорядоченного списка мономов, причем каждый моном представлен записью типа M постоянной длины:

type *M* = **record**
    c: ↑**number** // ссылка на коэффициент
    I: **array** [1..*n*] **of integer** // мультиндекс
    n: ↑*M* // ссылка на следующий моном
end record

Мономы в списке упорядочены. Моном определяется мультииндексом, поэтому порядок на мономах определяется порядком в $\mathbb{Z}^n$. Обычно рассматривают лексикографическое упорядочение и упорядочение по полной степени. *Лексикографическое упорядочение* $<_{\text{lex}}$ на

множестве мономов задаётся следующим образом. Сначала фиксируется некоторый линейный порядок на множестве переменных:

$$x_1 >_{\text{lex}} x_2 >_{\text{lex}} \ldots >_{\text{lex}} x_n.$$

Переменная x_1 называется *старшей* переменной. Порядок на переменных продолжается до порядка $<_{\text{lex}}$ на мономах за счёт рассмотрения лексикографического порядка на мультииндексах, как на векторах в $\mathbb{Z}^n$. Сравнить два монома можно с помощью слдющего алгоритма.

Алгоритм 3.2. Лексиграфический порядок на мономах.

Вход: Два монома $a_I X^I$ и $a_J X^J$.

Выход: Результат сравнения мономов:

-1, если $a_I X^I <_{\text{lex}} a_J X^J$

$+1$, если $a_I X^I >_{\text{lex}} a_J X^J$

0, если $a_I X^I =_{\text{lex}} a_J X^J$, то есть мономы подобны.

for s **from** 1 **to** n **do**

  if $I[s] < J[s]$ **then**

    return -1 // $a_I X^I <_{\text{lex}} a_J X^J$

  end if

  if $I[s] > J[s]$ **then**

    return $+1$ // $a_I X^I >_{\text{lex}} a_J X^J$

  end if

end for

return 0 // мультииндексы равны — мономы подобны

Порядок *полной степени* определяется следующим образом. *Полной степенью* $d(I)$ монома $a_I X^I$ называется сумма степеней мультииндекса I: $d(I) := \sum_{s=1}^{n} I[s]$. Далее порядок $<_{\text{tord}}$ определяется следующим образом:

$$a_I X^I <_{\text{tord}} a_J X^J := d(I) < d(J) \vee (d(I) = d(J) \& a_I X^I <_{\text{lex}} a_J X^J).$$

Для разреженного списочного представления критически важным оказывается способ упорядочения мономов.

3.9.5. Представление выражений

Для представления объектов более сложных, чем полиномы, используют другие структуры. Например, для представления произвольных выражений, содержащих кроме логических операций некоторые функции, такие как sin, log, можно использовать так называемое *рекурсивно-полиномиальное* представление. При этом представлении фиксируется набор исходных переменных $\{x_1, \ldots, x_k\}$, которые называются *полиномиальными* и вводятся по мере необходимости новые переменные $\{y_1, \ldots, y_k\}$, которые называются *функциональными*. Представление строится рекусивно следующим образом.

Выражение, не содержащее функциональных символов, представляется в виде упорядоченного списка мономов, как обычный полином. Если выражение содержит подвыражение $f(S)$, и представление для S построено, то определяется новая функциональная переменная y, которой сопоставляется дескриптор $y=f(S)$, и подвыражение $f(S)$ заменяется на переменную y.

Пример. Выражение

$$F = \sin(a + \log(b + c)) + 2d$$

рекурсивно-полиномиально представляется в виде

$$F = 1 \cdot y_1 + 2 \cdot d.$$

Функциональная переменная y_1 имеет дескриптор $y_1 = \sin(F_1)$, где

$$F_1 = 1 \cdot a + 1 \cdot y_2.$$

Переменная y_2 в свою очередь имеет дескриптор $y_2 = \log(F_2)$, где F_2 уже является обычным полиномом:

$$F_2 = 1 \cdot b + 1 \cdot c.$$

ВЫВОДЫ

Компьютерная алгебра является важнейшим приложением алгебраических методов.

4. КОМБИНАТОРИКА

Предмету комбинаторики не так просто дать краткое исчерпывающее определение. В некотором смысле слово "комбинаторика" можно понимать как синоним термина "дискретная математика", то есть исследование дискретных конечных математических структур. На школьном уровне с термином "комбинаторика" связывают просто набор известных формул, служащих для вычисления так называемых *комбинаторных чисел*, о которых идёт речь в первых разделах этой главы. Может показаться, что эти формулы полезны только для решения олимпиадных задач и не имеют отношения к практическому программированию. На самом деле это далеко не так. Вычисления на дискретных конечных математических структурах, которые часто называют *комбинаторными вычислениями*, требуют комбинаторного анализа для установления свойств и выявления оценки применимости используемых алгоритмов. Рассмотрим элементарный жизненный пример.

Пример. Пусть некоторое агентство недвижимости располагает базой данных из n записей, причём каждая запись содержит одно предложение (что имеется) и один запрос (что требуется) относительно объектов недвижимости. Требуется найти все такие пары записей, в которых предложение первой записи совпадает с запросом второй и одновременно предложение второй записи совпадает с запросом первой. (На бытовом языке это называется подбором вариантов обмена.) Допустим, что используемая СУБД позволяет проверить вариант за одну миллисекунду. Нетрудно сообразить, что при "лобовом" алгоритме поиска вариантов (каждая запись сравнивается с каждой) потребуется $n(n-1)/2$ сравнений. Если $n = 100$, то все в порядке — ответ будет получен за 4, 95 секунды. Но если $n = 100\,000$ (более реальный случай), то ответ будет получен за 4 999 950 секунд, что составляет без малого 1389 часов и вряд ли может считаться приемлемым. Обратите внимание на то, что мы оценили только трудоёмкость подбора

прямых вариантов, а существуют ещё варианты, когда число участников сделки больше двух...

Этот пример показывает, что практические задачи и алгоритмы требуют предварительного анализа и количественной оценки. Задачи обычно оцениваются с точки зрения *размера*, то есть общего количества различных вариантов, среди которых нужно найти решение, а алгоритмы оцениваются с точки зрения *сложности*. При этом различают *сложность по времени*, то есть количество необходимых шагов алгоритма, и *сложность по памяти*, то есть объём памяти, необходимый для работы алгоритма. Во всех случаях основным инструментом такого анализа оказываются формулы и методы, рассматриваемые в этой главе.

4.1. КОМБИНАТОРНЫЕ ЗАДАЧИ

Во многих практических случаях возникает необходимость подсчитать количество возможных комбинаций объектов, удовлетворяющих определённым условиям. Такие задачи называются *комбинаторными*. Разнообразие комбинаторных задач не поддаётся исчерпывающему описанию, но среди них есть целый ряд особенно часто встречающихся, для которых известны способы подсчёта.

4.1.1. Комбинаторные конфигурации

Для формулировки и решения комбинаторных задач используются различные модели *комбинаторных конфигураций*. Рассмотрим следующие две наиболее популярные:

1) Дано n предметов. Их нужно разместить по m ящикам так, чтобы выполнялись заданные ограничения. Сколькими способами это можно сделать?

2) Рассмотрим множество функций $f: X \to Y$, где $|X| = n$, $|Y| = m$. Не ограничивая общности, можно считать, что $X = \{1, ..., n\}$, $Y = \{1, ..., m\}$, $F = \langle F(1), ..., F(n)\rangle$, $1 \le F(i) \le m$. Сколько существует функций F, удовлетворяющих заданным ограничениям?

Замечание. Большей частью соответствие конфигураций, описанных на "языке ящиков" и на "языке функций", очевидно, поэтому доказательство правильности способа подсчёта (вывод формулы) можно провести на любом языке. Если сведение одной модели к другой не очевидно, то оно включается в доказательство.

4.1.2. Размещения

Число всех функций (при отсутствии ограничений), или число всех возможных способов разместить n предметов по m ящикам, называется *числом размещений* и обозначается $U(m, n)$.

Теорема. $U(m, n) = m^n$.

Доказательство. Поскольку ограничений нет, предметы размещаются независимо друг от друга и каждый из n предметов можно разместить m способами. □

Замечание. В комбинаторных задачах часто используются два правила, которые называются *правилом суммы* и *правилом произведения*. Неформально эти правила можно сформулировать следующим образом. Пусть существуют некоторые возможности построения комбинаторной конфигурации. Если эти возможности взаимно исключают друг друга, то их количества следует складывать, а если возможности независимы, то их количества следует перемножать.

Пример. При игре в кости бросаются две кости и выпавшие на верхних гранях очки складываются. Какова вероятность выбросить 12 очков? Каждый возможный исход соответствует функции

$$F: \{1, 2\} \to \{1, 2, 3, 4, 5, 6\}$$

(аргумент — номер кости, результат — очки на верхней грани). Таким образом, всего возможно $U(6, 2) = 6^2 = 36$ различных исходов. Из них только один (6 + 6) даёт двенадцать очков. Вероятность 1/36.

4.1.3. Размещения без повторений

Число инъективных функций, или число способов разместить n предметов по m ящикам, не более чем по одному в ящик, называется

числом размещений без повторений и обозначается $A(m, n)$, или $[m]_n$, или $(m)_n$.

Теорема. $A(m, n) = m! / (m - n)!$.

Доказательство. Ящик для первого предмета можно выбрать m способами, для второго — $m - 1$ способами и т. д. Таким образом,

$A(m, n) = m \cdot (m - 1) \cdot \ldots \cdot (m - n + 1) = m! / (m - n)!$. □

По определению считают, что $A(m,n) := 0$ при $n > m$ и $A(m, 0) := 1$.

Замечание. Простые формулы, выведенные для числа размещений без повторений, дают повод поговорить об элементарных, но весьма важных вещах. Рассмотрим две формулы:

$$A(m, n) = m \cdot (m - 1) \cdot \ldots \cdot (m - n + 1) \quad \text{и} \quad A(m, n) = m! / (m - n)!.$$

Формула слева выглядит сложной и незавершённой, формула справа — изящной и "математичной". Но формула — это частный случай алгоритма. При практическом вычислении левая формула оказывается намного предпочтительнее правой. Во-первых, для вычисления по левой формуле потребуется $m - 1$ умножение, а по правой — $2m - n - 2$ умножений и одно деление. Поскольку $n < m$, левая формула вычисляется существенно быстрее. Во-вторых, число $A(m, n)$ растёт довольно быстро и при больших m и n может не поместиться в разрядную сетку. Левая формула работает правильно, если результат помещается в разрядную сетку. При вычислении же по правой формуле переполнение может наступить "раньше времени" (то есть промежуточные результаты не помещаются в разрядную сетку, в то время как окончательный результат мог бы поместиться), поскольку факториал — очень быстро растущая функция.

Пример. В некоторых видах спортивных соревнований исходом является определение участников, занявших первое, второе и третье места. Сколько возможно различных исходов, если в соревновании участвуют n участников? Каждый возможный исход соответствует функции $\{F\colon \{1, 2, 3\} \to \{1..n\}$ (аргумент — номер призового места,

результат — номер участника). Таким образом, всего возможно $A(n, 3) = n(n - 1)(n - 2)$ различных исходов.

4.1.4. Перестановки

Если $|X| = |Y| = n$, то существуют взаимно-однозначные функции $f: X \to Y$. Число взаимно-однозначных функций, или *число перестановок n* предметов, обозначается $P(n)$.

Теорема. $P(n) = n!$.

Доказательство.

$P(n) = A(n, n) = n \cdot (n - 1) \cdot ... \cdot (n - n + 1) = n \cdot (n - 1) \cdot ... \cdot 1 = n!$. □

Пример. Последовательность $\mathcal{E} = (E_1, ..., E_m)$ непустых подмножеств множества E ($\mathcal{E} \subseteq 2^E$, $E_i \subseteq E$, $E_i \neq \varnothing$) называется *цепочкой множеств* в E, если $\forall i \in 1..(m - 1)$ ($E_i \subseteq E_{i+1}$ & $E_i \neq E_{i+1}$). Цепочка $\mathcal{E}$ называется *полной* цепочкой в E, если $|\mathcal{E}| = |E|$. Сколько существует полных цепочек? Очевидно, что в полной цепочке каждое следующее подмножество E_{i+1} получено из предыдущего подмножества E_i добавлением ровно одного элемента из E и таким образом, $|E_1| = 1$, $|E_2| = 2$, ..., $|E_m| = |E| = m$. Следовательно, полная цепочка определяется порядком, в котором элементы E добавляются для образования очередного элемента полной цепочки. Отсюда количество полных цепочек — это количество перестановок элементов множества E, равное $m!$.

4.1.5. Сочетания

Число строго монотонно возрастающих функций, или число размещений n неразличимых предметов по m ящикам не более чем по одному в ящик, то есть число способов выбрать из m ящиков n ящиков с предметами, называется *числом сочетаний* и обозначается $C(m, n)$, или C_m^n.

Теорема. $C(m, n) = m! / (n! \, (m - n)!)$.

Доказательство.

[Обоснование формулы] Сочетание является размещением без повторений неразличимых предметов. Следовательно, число сочетаний определяется тем, какие ящики заняты предметами, поскольку перестановка предметов при тех же занятых ящиках считается одним сочетанием. Таким образом, $C(m, n) = A(m, n) / P(n) = m! / (n! \, (m–n)!)$.

[Сведение моделей] Число сочетаний является числом строго монотонно возрастающих функций, потому что любая строго монотонно возрастающая функция $\{F: 1..n \to 1..m$ определяется набором своих значений, причём $1 \le F(1) < ... < F(n) \le m$. Другими словами, каждая строго монотонно возрастающая функция определяется выбором n чисел из диапазона $1..m$. Таким образом, число строго монотонно возрастающих функций равно числу n-элементных подмножеств m-элементного множества, которое, в свою очередь, равно числу способов выбрать n ящиков с предметами из m ящиков.

По определению $C(m, n) := 0$ при $n > m$.

Пример. В начале игры в домино каждому играющему выдаётся 7 костей из имеющихся 28 различных костей. Сколько существует различных комбинаций костей, которые игрок может получить в начале игры? Очевидно, что искомое число равно числу 7-элементных подмножеств 28-элементного множества. Имеем: $C(28, 7) = 28! / (7!(28–7)!) = (28 \cdot 27 \cdot 26 \cdot 25 \cdot 24 \cdot 23 \cdot 22)/(7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1) = 1\,184\,040$.

4.1.6. Сочетания с повторениями

Число монотонно возрастающих функций, или число размещений n неразличимых предметов по m ящикам, называется *числом сочетаний с повторениями* и обозначается $V(m, n)$.

Теорема. $V(m, n) = C(n + m – 1, n)$.

Доказательство. Монотонно возрастающей функции $f: 1..n \to 1..m$ однозначно соответствует строго монотонно возрастающая функция $\{f': 1..n \to 1..(n + m – 1)$. Это соответствие устанавливается следующей формулой: $f'(k) = f(k) + k – 1$.

Пример. Сколькими способами можно рассадить n вновь прибывших гостей среди m гостей, уже сидящих за круглым столом? Очевидно, что между m сидящими за круглым столом гостями имеется m промежутков, в которые можно рассаживать вновь прибывших. Таким образом, число способов равно

$$V(m, n) = C(m + n - 1, n) = (m + n - 1)!/(n!(m - 1)!).$$

4.2. ПЕРЕСТАНОВКИ

Для вычисления количества перестановок в \ref{7.1.4} установлена очень простая формула: $P(n) = n!$. Применяя эту формулу при решении практических задач, не следует забывать, что факториал — это {\e*m* очень\/} быстро растущая функция, в частности, факториал растёт быстрее экспоненты. Действительно, используя известную из математического анализа *формулу Стирлинга*

$$n^n \pi (2\pi n)^{1/2} e^{-n} < n! < n^n (2\pi n)^{1/2} e^{-n+1/(12n)},$$

нетрудно показать, что

$$\lim_{n\to+\infty} (n!)/(2^n) = +\infty.$$

4.2.1. Циклы в перестановках

Взаимно-однозначные функции (перестановки), удобно задавать таблицами подстановки. В таблице подстановки нижняя строка (значения функции) является перестановкой элементов верхней строки (значения аргумента). Если принять соглашение, что элементы верхней строки (аргументы) всегда располагаются в определённом порядке (например, по возрастанию), то верхнюю строку можно не указывать — подстановка определяется одной нижней строкой. Таким образом, подстановки (таблицы) взаимно-однозначно соответствуют перестановкам (функциям). Напомним, что множество перестановок образует группу относительно суперпозиции. Перестановку f (и соот-

ветствующую ей подстановку) элементов 1, ..., n будем обозначать $\langle a_1, \ldots, a_n \rangle$, где a_i — все различные числа из диапазона $1..n$.

Если задана перестановка f, то *циклом* называется последовательность элементов $x_0, \ldots, x_k$, такая, что

$$f(x_i) = \textbf{if } 0 \le i < k \textbf{ then } x_{i+1} \textbf{ else } x_0 \textbf{ end if}.$$

Цикл длины 2 называется *транспозицией*.

4.2.2. Инверсии

Если в перестановке $f = \langle a1, \ldots, an \rangle$ для элементов a_i и a_j имеет место неравенство $a_i > a_j$ при $i < j$, то пара (a_i, a_j) называется *инверсией*. Обозначим $I(f)$ — *число инверсий* в перестановке f.

Теорема. *Произвольную перестановку f можно представить в виде суперпозиции I(f) транспозиций соседних элементов*.

Доказательство. Пусть $f = \langle a_1, \ldots, 1, \ldots, a_n \rangle$. Переставим 1 на первое место, меняя её местами с соседними слева элементами. Обозначим последовательность этих транспозиций через t_1. При этом все инверсии (и только они), в которых участвовала 1, пропадут. Затем переставим 2 на второе место и т. д. Таким образом, $f \circ t_1 \circ \ldots \circ t_n = e$ и по свойству группы $f = t_{n-}{}^{1} \circ \ldots \circ t_1^{-1}$, причём $|t_1| + |t_2| + \ldots + |t_n| = I(f)$. □

Следствие. *Всякая сортировка может быть выполнена перестановкой соседних элементов*.

Замечание. Доказанная теорема утверждает, что произвольную перестановку можно представить в виде композиции определённого числа транспозиций, но не утверждает, что такое представление является эффективным. Метод *сортировки*, основанный на предшествующей теореме, известен как *метод пузырька*. Заметим, что при перемещении элемента на своё место транспозициями соседних элементов все элементы остаются на своих местах, кроме двух соседних элементов, которые, возможно, меняются местами. Таким образом, метод пузырька может быть выражен в форме алгоритма 4.1. Этот алгоритм прост, но является далеко не самым эффективным алгоритмом сортировки.

Алгоритм 4.1. Сортировка методом пузырька.

Вход: массив A: **array** [1..n] **of** B, где значения элементов массива расположены в произвольном порядке и для значений типа B задано отношение <.

Выход: массив A: **array** [1..n] **of** B, в котором значения расположены в порядке возрастания.

```
for i from 2 to n do
    for j from n downto i do
        if A[j] < A[j–1] then
            A[j]↔A[j–1] //транспозиция соседних элементов
        end if
    end for
end for
```

4.2.3. Генерация перестановок

На множестве перестановок естественным образом можно определить порядок на основе упорядоченности элементов. А именно, говорят, что перестановка $\langle a_1, ..., a_n\rangle$ *лексикографически* предшествует перестановке $\langle b_1, ..., b_n\rangle$, если $\exists k \le n\ (a_k < b_k\ \&\ \forall i < k\ \ (a_{i=}b))$. Аналогично, говорят, что перестановка $\langle a_1, ..., a_n\rangle$ *антилексикографически* предшествует перестановке $\langle b_1, ..., b_n\rangle$ если $\exists k \le n\ (a_k > b_k\ \&\ \forall i > k\ (a_{i=}b_i))$. Следующий алгоритм генерирует все перестановки элементов 1, ..., n в антилексикографическом порядке. Массив P: **array** [1..n] **of** 1..n является глобальным и предназначен для хранения перестановок.

Алгоритм 5.2. Генерация перестановок в антилексикографическом порядке

Вход: n — количество элементов

Выход: последовательность перестановок элементов 1, ..., n в антилексикографическом порядке.

```
for i from 1 to n do
    P[i] := i // инициализация
```

end for

Antilex(*n*) // вызов рекурсивной процедуры Antilex

Основная работа по генерации перестановок выполняется рекурсивной процедурой Antilex.

Вход: *m* — параметр процедуры — количество первых элементов массива *P*, для которых генерируются перестановки.

Выход: последовательность перестановок элементов 1, ..., *m* в антилексикографическом порядке.

if *m* = 1 **then**
    yield *P* // очередная перестановка
else
    for *i* **from** 1 **to** *m* **do**
        Antilex (*m*–1) // рекурсивный вызов
        if *i*< *m* **then**
            P[*i*] ↔ *P*[*m*] // следующий элемент
            Reverse(*m*–1) // изменение порядка элементов
        end if
    end for
end if

Вспомогательная процедура Reverse переставляет элементы заданного отрезка массива *P* в обратном порядке.

Вход: *k* — номер элемента, задающий отрезок массива *P*, подлежащий перестановке в обратном порядке.

Выход: первые *k* элементов массива *P* переставлены в обратном порядке

j := 1 // нижняя граница обращаемого диапазона
while *j* < *k* **do**
    P[*j*] ↔ *P*[*k*] // меняем местами элементы
    j := *j* + 1 // увеличиваем нижнюю границу
    k := *k* – 1 // уменьшаем верхнюю границу
end while

Обоснование. Заметим, что искомую последовательность перестановок n элементов можно получить из последовательности перестановок $n-1$ элементов следующим образом. Нужно выписать n блоков по $(n-1)!$ перестановок в каждом, соответствующих последовательности перестановок $n-1$ элементов в антилексикографическом порядке. Затем ко всем перестановкам в первом блоке нужно приписать справа n, во втором — $n-1$ и т. д. в убывающем порядке. Затем в каждом из блоков (кроме первого), к перестановкам которого справа приписан элемент i, нужно в перестановках блока заменить все вхождения элемента i на элемент n. В полученной последовательности все перестановки различны, и их $n(n-1)! = n!$, то есть перечислены все перестановки. При этом антилексикографический порядок соблюден для последовательностей внутри одного блока, потому что этот порядок был соблюден в исходной последовательности, а для последовательностей на границах двух блоков — потому что происходит уменьшение самого правого элемента. Обратимся к процедуре Antilex — легко видеть, что в ней реализовано указанное построение. В основном цикле сначала строится очередной блок — последовательность перестановок первых $m-1$ элементов массива P (при этом элементы $P[m],..., P[n]$ остаются неизменными). Затем элемент $P[m]$ меняется местами с очередным элементом $P[i]$. Вызов вспомогательной процедуры Reverse необходим, поскольку последняя перестановка в блоке является обращением первой, а для генерации следующего блока на очередном шаге цикла нужно восстановить исходный порядок. □

Пример. Последовательность перестановок в антилексикографическом порядке для $n = 3$: (1, 2, 3), (2, 1, 3), (1, 3, 2), (3, 1, 2), (2, 3, 1), (3, 2, 1).

4.2.4. Двойные факториалы

Содержание этого параграфа не имеет отношения к основной теме раздела — перестановкам и включено в раздел по сходству обозначений и с целью привести две формулы, которые иногда исполь-

зуются в практических комбинаторных расчётах и требуются в последующих разделах. *Двойным факториалом* натурального числа n (обозначается $n!!$) называется произведение числа n и всех меньших натуральных чисел той же чётности.

Пример. $10!! = 10 \cdot 8 \cdot 6 \cdot 4 \cdot 2 = 3840$.

Теорема 1. $(2k)!! = 2^k \cdot k!$.

Доказательство. $(2k)!! = 2 \cdot 4 \cdot \ldots \cdot (2k - 2) \cdot 2k =$

$= (2 \cdot 1) \cdot (2 \cdot 2) \cdot \ldots \cdot (2 \cdot (k - 1)) \cdot (2 \cdot k) = (2 \cdot 2 \cdot \ldots \cdot 2 \cdot 2) \cdot (1 \cdot 2 \cdot \ldots \cdot (k - 1) \cdot k) = 2^k \cdot k!$. □

Теорема 2. $(2k + 1)!! = (2k + 1)! / (2^k \cdot k!)$.

Доказательство. Заметим, что $(2k + 1)! = (2k + 1)!!(2k)!!$. □

4.3. БИНОМИАЛЬНЫЕ КОЭФФИЦИЕНТЫ

Число сочетаний $C(m, n)$ — это число различных n-элементных подмножеств m-элементного множества. Числа $C(m, n)$ обладают целым рядом свойств, рассматриваемых в этом разделе, которые оказываются очень полезными при решении различных комбинаторных задач.

4.3.1. Элементарные тождества

Основная формула для числа сочетаний $C(m, n) = m!/[n!(m - n)!]$ позволяет получить следующие простые тождества.

Теорема.

1) $C(m, n) = C(m, m - n)$.

2) $C(m, n) = C(m - 1, n) + C(m - 1, n - 1)$.

3) $C(n, i)C(i, m) = C(n, m)C(n - m, i - m)$.

Доказательство.

[1] $C(m, m - n) = m!/[(m - n)!\ (m - (m - n))!] = m!/[(m - n)!n!] =$

$= C(m, n)$.

[2] $C(m - 1, n) + C(m - 1, n - 1) =$

$= (m - 1)!/[n!\ (m - n - 1)!] + (m - 1)!/[(n - 1)\ (m - 1 - (n - 1))!] =$

$= (m - 1)!/[n(n - 1)!(m - n - 1)!] + (m - 1)!/[(n - 1)!(m - n)(m - n - 1)!] = [(m - n)(m - 1)! + n(m - 1)!] / [n(n - 1)!(m - n)(m - n - 1)!] =$

$= [(m - n + n)(m - 1)!] / [n!(m - n)!] = m! / [n!(m - n)!] = C(m, n)$.

[3] $C(n, i)C(i, m) = n!/[i!(n - i)!]\cdot i!/[m!(i - m)!] =$
$= n!/[m!(i - m)!(n - i)!] = [n!(n - m)!] / [m!(i - m)!(n - i)!(n - m)!] =$
$= n!/[m!(n - m)!] \cdot (n - m)!/[(i - m)!(n - i)!] =$
$= C(n, m)C(n - m, i - m)$. □

4.3.2. Бином Ньютона

Числа сочетаний $C(m, n)$ называются также *биномиальными коэффициентами*. Смысл этого названия устанавливается следующей теоремой, известной также как формула *бинома Ньютона*.

Теорема. $(x + y)^m = \sum^{m}_{n=0} C(m, n)\, x^n\, y^{m-n}$.

Доказательство. По индукции. База, $m = 1$:

$(x + y)^1 = x + y = 1x^1y^0 + 1x^0y^1 = C(1,0)x^1y^0 + C(1,1)x^0y^1 =$
$= \sum^{1}_{n=0} C(1,n)x^n y^{1-n}$.

Индукционный переход:

$(x + y)^m = (x + y)(x + y)^{m-1} = (x + y)\sum^{m-1}_{n=0} C(m - 1, n)x^n y^{m-n-1} =$
$= \sum^{m-1}_{n=0} x\, C(m - 1, n)\, x^n y^{m-n-1} + \sum^{m-1}_{n=0} y\, C(m - 1, n)\, x^n y^{m-n-1} =$
$= \sum^{m-1}_{n=0} C(m - 1, n)\, x^{n+1} y^{m-n-1} + \sum^{m-1}_{n=0} C(m - 1, n)\, x^n y^{m-n} =$
$= \sum^{m}_{n=1} C(m - 1, n - 1)\, x^n y^{m-n} + \sum^{m-1}_{n=0} C(m - 1, n)\, x^n\, y^{m-n} =$
$= C(m - 1, 0)\, x^0\, y^m + \sum^{m-1}_{n=1} (C(m - 1, n - 1) + C(m - 1, n))\, x^n\, y^{m-n} +$
$+ C(m - 1, m - 1)\, x^m\, y^0 = \sum^{m}_{n=0} C(m, n)\, x^n\, y^{m-n}$. □

Следствие 1. $\sum^{m}_{n=0} C(m, n) = 2^m$.

Доказательство. $2^m = (1+1)^m = \sum^{n}_{n=0} C(m, n)1^n 1^{m-n} = \sum^{m}_{n=0} C(m, n)$. □

Следствие 2. $\sum^{m}_{n=0} (-1)^n\, C(m, n) = 0$.

Доказательство.

$0 = (-1 + 1)^m = \sum^{m}_{n=0} C(m,n)(-1)^n 1^{m-n} = \sum^{m}_{n=0} (-1)^n\, C(m, n)$. □

4.3.3. Свойства биномиальных коэффициентов

Биномиальные коэффициенты обладают целым рядом замечательных свойств.

Теорема 1. $\sum^{m}_{n=0} n\mathrm{C}(m, n) = m2^{m-1}$.

Доказательство. Рассмотрим следующую последовательность, составленную из чисел 1,..., m. Сначала выписаны все подмножества длины 0, потом все подмножества длины 1 и т. д. Имеется $C(m, n)$ подмножеств мощности n, и каждое из них имеет длину n, таким образом, всего в этой последовательности $\sum^{m}_{n=0} n\mathrm{C}(m, n)$ чисел. С другой стороны, каждое число x входит в эту последовательность $2^{|\{1,..., m\}\{x\}|} = 2^{m-1}$ раз, а всего чисел m.

Теорема 2. $C(m + n, k) = \sum^{k}_{i=0} C(m, i)C(n, k - i)$.

Доказательство. $C(n + m, k)$ — это число способов выбрать k предметов из $m + n$ предметов. Предметы можно выбирать в два приема: сначала выбрать i предметов из первых m предметов, а затем выбрать недостающие $k - i$ предметов из оставшихся n предметов. Отсюда общее число способов выбрать k предметов составляет

$$\sum^{k}_{i = 0} C(m, i)C(n, k - i).$$

Последнее свойство известно как *тождество Коши.*

Замечание. Данные свойства биномиальных коэффициентов нетрудно доказать, проводя алгебраические выкладки обычным образом. Например, первое свойство можно получить так:

$\sum^{m}_{n=0} n\mathrm{C}(m, n) = \sum^{m}_{n=1} [n\, m!] / [(m - n)!n!] =$

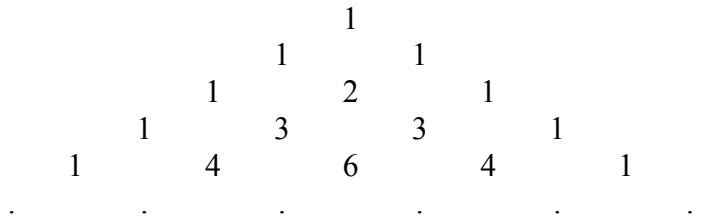
$= \sum^{m}_{n=1} [m\, (m - 1)!] / [(m - n)!(n - 1)!] =$

$= m\sum^{m-1}_{n=0} (m - 1)!/[(m - n - 1)!n!] =$

$= m \sum^{m-1}_{n=0} C(m - 1, n) = m2^{m - 1}$. В доказательстве использованы рассуждения "в комбинаторном духе", чтобы проиллюстрировать "неалгебраические" способы решения комбинаторных задач.

4.3.4. Треугольник Паскаля

Из второй формулы теоремы 4.3.1 вытекает эффективный способ рекуррентного вычисления значений биномиальных коэффициентов, который можно представить в графической форме, известной как *треугольник Паскаля*:



В этом равнобедренном треугольнике каждое число (кроме единиц на боковых сторонах) является суммой двух чисел, стоящих над ним. Число сочетаний $C(m, n)$ находится в $(m + 1)$-м ряду на $(n + 1)$-м месте.

4.3.5. Генерация подмножеств

Элементы множества $\{1, ..., m\}$ естественным образом упорядочены. Поэтому каждое n-элементное подмножество этого множества также можно рассматривать как упорядоченную последовательность. На множестве таких последовательностей определяется лексикографический порядок. Следующий простой алгоритм генерирует все n-элементные подмножества m-элементного множества в лексикографическом порядке.

Алгоритм 5.3. Генерация n-элементных подмножеств m-элементного множества

Вход: n — мощность подмножества, m — мощность множества, $m \geq n > 0$.

Выход: последовательность всех n-элементных подмножеств m-элементного множества в лексикографическом порядке.

for i **from** 1 **to** m **do**

    $A[i] := i$ // инициализация исходного множества

end for

```
if m=n then
        return A[1..n]}   // единственное подмножество
end if
p := n                    // p — номер первого изменяемого элемента
while p ≥ 1 do
        yield A[1..n]     // очередное подмножество
                          // в первых n элементах массива A
        if A[n] = m then
                p := p – 1   // нельзя увеличить последний элемент
        else
                p := n       // можно увеличить последний элемент
        end if
        if p ≥ 1 then
                for i from n downto p do
                        A[i] := A[p]+i–p+1 // увеличение элементов
                end for
        end if
end while
```

Обоснование. Заметим, что в искомой последовательности n-элементных подмножеств (каждое из которых является возрастающей последовательностью n чисел из диапазона $1..m$) вслед за последовательностью $\langle a_1, ..., a_n\rangle$ следует последовательность

$$\langle b_1, ..., b_n \rangle = \langle a_1, ..., a_{p-1}, a_p + 1, a_p + 2, ..., a_p + n - p + 1\rangle,$$

где p — максимальный индекс, для которого $b_n = a_p + n - p + 1 \le m$. Другими словами, следующая последовательность получается из предыдущей заменой некоторого количества элементов в хвосте последовательности идущими подряд натуральными числами, но так, чтобы последний элемент не превосходил m, а первый изменяемый элемент был на 1 больше, чем соответствующий элемент в предыдущей последовательности. Таким образом, индекс p, начиная с которого следует изменить "хвост последовательности", определяется по зна-

чению элемента $A[n]$. Если $A[n] < m$, то следует изменять только $A[n]$, и при этом $p := n$. Если же уже $A[n] = m$, то нужно уменьшать индекс $p := p - 1$, увеличивая длину изменяемого хвоста.

Пример. Последовательность n-элементных подмножеств m-элементного множества в лексикографическом порядке для $n = 3$ и $m = 4$: (1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4).

Замечание. Довольно часто встречаются задачи, в которых требуется найти в некотором множестве элемент, обладающий определёнными свойствами. При этом исследуемое множество может быть велико, а его элементы устроены достаточно сложно. Если неизвестно заранее, как именно следует искать элемент, то остаётся перебирать все элементы, пока не попадётся нужный (поэтому такие задачи называют *переборными*. При решении переборных задач совсем не обязательно и, как правило, нецелесообразно строить всё исследуемое множество (*пространство поиска*) в явном виде. Достаточно иметь *итератор*, перебирающий элементы множества в подходящем порядке. Фактически, алгоритмы 1.2, 5.2 и 5.3 являются примерами таких итераторов для некоторых типичных пространств поиска. Чтобы использовать эти алгоритмы в качестве итераторов при переборе, достаточно вставить вместо оператора **yield** проверку того, что очередной элемент является искомым.

4.3.6. Мультимножества и последовательности

В параграфе 4.1.5 показано, что число n-элементных подмножеств m-элементного множества равно $C(m, n)$. Поскольку n-элементные подмножества — это n-элементные индикаторы, ясно, что число n-элементных индикаторов над m-элементным множеством также равно $C(m, n)$. Общее число n-элементных мультимножеств $[(n_1)x_1, ..., (n_m)x_m]$, в которых $n_1 + ... + n_m = n$, над m-элементным множеством $X = \{x_1, \ldots, x_m\}$ нетрудно определить, если заметить, что это число способов разложить n неразличимых предметов по m ящикам, то есть число сочетаний с повторениями $V(m, n) = C(n + m - 1, n)$.

Пусть задано множество $X = \{x_1, \ldots, x_m\}$. Рассмотрим последовательность $Y = \langle y_1, \ldots, y_m \rangle$, где $\forall i \ (y_i \in X)$ и в последовательности Y элемент x_i встречается n_i раз. Тогда мультимножество $\underline{\underline{X}} = [(n_1)x_1,..., (n_m)x_m]$ называется *составом* последовательности Y. Ясно, что состав любой последовательности определяется однозначно, но разные последовательности могут иметь один и тот же состав.

Пример. Пусть $X = \{1, 2, 3\}$, $Y_1 = \langle 1, 2, 3, 2, 1 \rangle$, $Y_2 = \langle 1, 1, 2, 2, 3 \rangle$.

Тогда последовательности Y_1 и Y_2 имеют один и тот же состав $\underline{\underline{X}} = [(2)1, (2)2, (1)3]$.

Замечание. Из органической химии известно, что существуют различные вещества, имеющие один и тот же химический состав. Они называются изомерами.

Все последовательности над множеством $X = \{x_1, \ldots, x_m\}$, имеющие один и тот же состав $\underline{\underline{X}} = [(n_1)x_1,..., (n_m)x_m]$, имеют одну и ту же длину $n = n_1 + ... + n_m$. Обозначим число последовательностей одного состава $C(n; n_1, \ldots, n_m)$.

Теорема. $C(n; n_1, \ldots, n_m) = n! / [n_1!... n_m!]$.

Доказательство. Мультимножество $\underline{\underline{X}} = [(n_1)x_1, ..., (n_m)x_m]$ можно записать в виде последовательности $\langle x_1, \ldots, x_1, x_2, \ldots, x_2, ..., x_m, \ldots, x_m \rangle$, где элемент x_i встречается n_i раз. Ясно, что все последовательности одного состава $\underline{\underline{X}}$ получаются из указанной последовательности перестановкой элементов. При этом, если переставляются одинаковые элементы, например, x_i, то получается та же самая последовательность. Таких перестановок $n_i!$. Таким образом, общее число перестановок $n_1!... n_m!C(n; n_1, \ldots, n_m)$. С другой стороны, число перестановок n элементов равно $n!$. □

4.3.7. Мультиномиальные коэффициенты

Числа $C(n; n_1, ..., n_m)$ встречаются в практических задачах довольно часто, поскольку обобщают случай индикаторов, которые описываются биномиальными коэффициентами, на случай произвольных мультимножеств. В частности, справедлива формула, обоб-

щающая формулу бинома Ньютона, поэтому числа $C(n;\ n_1,\ ...,\ n_m)$ иногда называют *мультиномиальными коэффициентами*.

Теорема. $(x_1+ \dots +x_m)^n = \sum_{n1+\dots+nm=n} C(n;\ n_1,\ ...,\ n_m)\ x_1^{n_1}\cdot\dots\ x_m^{n_m}$.

Доказательство. Индукция по m. База: при $m = 2$ по формуле бинома Ньютона имеем $(x_1 + x_2)^n = \sum^{n}_{i=0} C(n,\ i)x_1^{\ i}\ x_2^{\ n-i} =$

$= \sum^{n}_{i=0} n!/[i!(n-i)!]\cdot x_1^{\ i}x_2^{\ n-i} = \sum_{n_1+n_2=n} n!\ /\ [n_1!\ n_2!]\ x_1^{\ n_1}\ x_2^{\ n_2}$.

Пусть теперь $(x_1+\dots+x_{m-1})^n=\sum_{n_1+\dots+n_{m-1}=n}C(n;n_1,\dots,n_{m-1})x_1^{\ n_1}\dots x_{m-1}^{\ n_{m-1}}$.

Рассмотрим $(x_1+\dots+x_m)^n$. Имеем: $(x_1+\dots+x_m)^n = ((x_1+\dots+x_{m-1})+x_m)^n =$

$= \sum^{n}_{i=0} C(n,\ i)(x_1 +\dots+ x_{m-1})^i\ x_m^{\ n-i} =$

$= \sum^{n}_{i=0} n!/[i!(n-i)!]\ \left(\sum_{n_1+\dots+n_{m-1}=i} i!/[n_1!\cdot\dots\ n_{m-1}!]\ x_1^{\ n_1}\dots x_{m-1}^{\ n_{m-1}}\right) x_m^{\ n-i=}$

$= \sum^{n}_{i=0} \sum_{n_1+\dots+n_{m-1}=i}[n!i!]\ /\ [i!(n-i)!n_1!\ \dots\ n_{m-1}!]\ x_1^{\ n_1}\dots x_{m-1}^{\ n_{m-1}}\ x_m^{\ n-i} =$

$= \sum_{n_1+\dots+n_{m-1}+n_m=n} n!\ /\ [n_1!\cdot\dots\ n_{m-1}!n_m!]\ x_1^{\ n_1}\dots x_{m-1}^{\ n_{m-1}}\ x_m^{\ n-i}$,

где $n_m := n - i$. □

Следствие. $\sum_{n_1+\dots+n_m=n} n!\ /\ [n_1!\cdot\dots\ !n_m!] = m^n$.

Доказательство. $m^n = (1+\dots+1)^n = \sum_{n_1+\dots+n_m=n} n!/[n_1!\dots n_m!\}1^{n_1}\ 1^{n_m} =$

$= {}_{n_1+\dots+n_m=n}\ n!\ /\ [n_1!\ \dots\ n_m!]$.

Пример. При игре в преферанс трём играющим сдаётся по 10 карт из 32 карт и 2 карты остаются в "прикупе". Сколько существует различных раскладов?

$C(32;\ 10,\ 10,\ 10,\ 2) = [32!]\ /\ [10!10!10!2!] = 2\ 753\ 294\ 408\ 504\ 640$.

4.4. РАЗБИЕНИЯ

Разбиения не рассматривались среди типовых комбинаторных конфигураций в разделе 4.1, потому что получить для них явную формулу не так просто, как для остальных. В этом разделе отмечаются основные свойства разбиений, а окончательные формулы приведены в параграфе 4.6.3.

4.4.1. Числа Стирлинга второго рода

Пусть $\mathcal{B} = \{B_1, ..., B_n\}$ — разбиение множества X из m элементов на n подмножеств: $B_i \subseteq X$, $B_1 \cup \ldots \cup B_n = X$, $B_i \neq \varnothing$, $B_i \cap B_j = \varnothing$ при $i \neq j$.

Подмножества B_i называются *блоками* разбиения. Между разбиениями с непустыми блоками и отношениями эквивалентности существует взаимнооднозначное соответствие (см. 1.7.1). Если E_1 и E_2 — два разбиения X, то говорят, что разбиение E_1 есть *измельчение* разбиения E_2, если каждый блок E_2 есть объединение блоков E_1. Измельчение является частичным порядком на множестве разбиений.

Число разбиений m-элементного множества на n блоков называется *числом Стирлинга второго рода* и обозначается $S(m, n)$. По определению положим

$S(m, m) := 1$, $S(m, 0) := 0$ при $m > 0$, $S(0, 0) := 1$, $S(m, n) := 0$ при $n > m$.

Теорема 1. $S(m, n) = S(m - 1, n - 1) + nS(m - 1, n)$.

Доказательство. Пусть $\mathcal{B}$ — множество всех разбиений множества $M := \{1, ..., m\}$ на n блоков. Положим

$$\mathcal{B}_1 := \{X \in \mathcal{B} \mid \exists B \in X\,(B = \{m\})\},\ \mathcal{B}_2 := \{X \in \mathcal{B} \mid \neg\exists B \in X\,(B = \{m\})\},$$

то есть в $\mathcal{B}_1$ входят разбиения, в которых элемент m образует отдельный блок, а в $\mathcal{B}_2$ — все остальные разбиения. Заметим, что $\mathcal{B}_2 = \{X \in \mathcal{B} \mid m \in X \Rightarrow |X| > 1\}$. Тогда $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{B}_1 \cap \mathcal{B}_2 = \varnothing$. Имеем $|\mathcal{B}_1| = S(m - 1, n - 1)$, $|\mathcal{B}_2| = n\,S(m - 1, n)$, так как все разбиения $\mathcal{B}_2$ получаются следующим образом: берём все разбиения множества $\{1, ..., m - 1\}$ на n блоков (их $S(m - 1, n)$) и в каждый блок по очереди помещаем элемент m. Следовательно, $S(m, n) = |\mathcal{B}| = |\mathcal{B}_1| + |\mathcal{B}_2| = S(m - 1, n - 1) + nS(m - 1, n)$. □

Теорема 2. $S(m, n) = \sum^{m-1}\_\{i = n - 1\}\ C(m - 1, i)S(i, n - 1)$.

Доказательство. Пусть $\mathcal{B}$ — множество всех разбиений множества $\mathrm{M} := \{1, \ldots, m\}$ на n блоков. Рассмотрим семейство $\underline{B} := \{B \subseteq 2^m \mid m \in B\}$.

Тогда $\mathcal{B} = \cup_{B \in \underline{B}} \mathcal{B}_B$, где $\mathcal{B}_B := \{X \mid X \in \mathcal{B} \,\&\, B \in X\}$, причём $\mathcal{B}_{B'} \cap \mathcal{B}_{B''} = \varnothing$, если $B' \neq B''$. Пусть $B \in \underline{B}$ и $b := |B|$. Тогда $|\mathcal{B}_B| = S(m-b, n-1)$. Заметим, что $|\{B \in \underline{B} \mid b = |B|\}| = C(m-1, b-1)$. Имеем:

$$S(m, n) \,\&=\, |\mathcal{B}| = \sum_{b=1}^{m-(n-1)} \left| \cup_{B \in \underline{B} \,\&\, |B|=b} \mathcal{B}_B \right| =$$

$$= \sum_{b=1}^{m-(n-1)} C(m-1, b-1)\, S(m-b, n-1) =$$

$$= \sum_{i=m-1}^{n-1} C(m-1, m-i-1)\, S(i, n-1) =$$

$$= \sum_{i=n-1}^{m-1} C(m-1, i)\, S(i, n-1),$$

где $i := m - b$. □

4.4.2. Числа Стирлинга первого рода

Число сюръективных функций, то есть число размещений m предметов по n ящикам, таких, что все ящики заняты, называется *числом Стирлинга первого рода* и обозначается $s(m, n)$.

Теорема. $s(m, n) = n!\, S(m, n)$.

Доказательство. Каждое разбиение множества $\{1, \ldots, m\}$ соответствует семейству множеств уровня сюръективной функции и обратно (см. 1.7.3). Таким образом, число различных семейств множеств уровня сюръективных функций — это число Стирлинга второго рода $S(m, n)$. Всего сюръективных функций $s(m, n) = n!\, S(m, n)$, так как число сюръективных функций с заданным семейством множеств уровня равно числу перестановок множества значений функции. □

4.4.3. Число Белла

Число всех разбиений m-элементного множества называется *числом Белла* и обозначается $B(m)$,

$$B(m) := \sum^{m}_{n=0} S(m, n),\ B(0) := 1.$$

Теорема. $B(m + 1) = \sum^{m}_{i=0} C(m, i)\, B(i)$.

Доказательство. Пусть $\mathcal{B}$ — множество всех разбиений множества $M_1 = 1..(m + 1)$. Рассмотрим множество подмножеств множества M_1, содержащих элемент $m + 1$: $\underline{B} := \{B \subseteq 2^{M_1} \mid m + 1 \in B\}$. Тогда $\cup_{B \in \underline{B}} \mathcal{B}_B$, где $\mathcal{B}_B := \{X \in \mathcal{B} \mid B \in X\}$. Пусть $B \in \underline{B}$ и $b = |B|$. Тогда $|\mathcal{B}_B| = B(m + 1 - b)$. Заметим, что $|\{B \in \underline{B} \mid b = |B|\}| = C(m, b - 1)$. Следовательно,

$$B(m + 1) = |\mathcal{B}| = \sum^{m+1}_{b=1} C(m, b - 1)\, B(m - b + 1) =$$

$$= \sum^{0}_{i=m} C(m, m - i)\, B(i) = \sum^{m}_{i=0} C(m, i)\, B(i), \text{ где } i := m - b + 1. \square$$

4.5. ВКЛЮЧЕНИЯ И ИСКЛЮЧЕНИЯ

Приведённые в предыдущих четырёх разделах формулы и алгоритмы дают способы вычисления комбинаторных чисел для некоторых распространённых комбинаторных конфигураций. Практические задачи не всегда прямо сводятся к известным комбинаторным конфигурациям. В этом случае используются различные методы сведения одних комбинаторных конфигураций к другим. В этом и двух следующих разделах рассматриваются три наиболее часто используемых метода. Мы начинаем с самого простого и прямолинейного, но имеющего ограниченную область применения метода включений и исключений.

4.5.1. Объединение конфигураций

Часто комбинаторная конфигурация является объединением других, число комбинаций в которых вычислить проще. В таком случае требуется уметь вычислять число комбинаций в объединении. В простых случаях формулы для вычисления очевидны:

$$|A \cup B| = |A| + |B| - |A \cap B|,$$

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C|.$$

Пример. Сколько существует натуральных чисел, меньших 1000, которые не делятся ни на 3, ни на 5, ни на 7? Всего чисел, меньших тысячи, 999. Из них:

999:3 = 333 делятся на 3,

999:5 = 199 делятся на 5,

999:7 = 142 делятся на 7,

999:(3*5) = 66 делятся на 3 и на 5,

999:(3*7) = 47 делятся на 3 и на 7,

999:(5*7) = 28 делятся на 5 и на 7,

999:(3*5*7) = 9 делятся на 3, на 5 и на 7.

Имеем: 999 – (333 + 199 + 142 – 66 – 47 – 28 + 9) = 457.

4.5.2. Формула включений и исключений

Следующая формула, известная как *формула включений и исключений*, позволяет вычислить мощность объединения нескольких множеств, если известны их мощности и мощности всех возможных пересечений.

Теорема. $\left|\cup^{n}_{i=1} A\right| = \sum^{n}_{i=1}|A_i| - \sum_{1\le i<j\le n}|A_i \cap A_j| +$

$+ \sum_{1 \le i<j<k \le n}|A_i \cap A_j \cap A_k| - \ldots + (-1)^{n-1}|A_1 \cap \ldots \cap A_n|.$

Доказательство. Индукция по n. Для n = 2, 3 теорема проверена в предыдущем параграфе. Пусть

$\left|\cup^{n-1}_{i=1} A_i\right| = \sum^{n-1}_{i=1}|A_i| - \sum_{1 \le i<j \le n-1}|A_i \cap A_j| + \ldots + (-1)^{n-2}|A_i \cap \ldots \cap A_{n-1}|.$

Заметим, что $\left(\cup^{n-1}_{i=1} A_i\right) \cap A_n = \cup^{n-1}_{i=1}(A_i \cap A_n)$, и по индукционному предположению

$\left|\cup^{n-1}_{i=1}(A_i \cap A_n)\right| = \sum^{n-1}_{i=1}|A_i \cap A_n| - \sum_{1 \le i<j \le n-1}|A_i \cap A_j \cap A_n| + \ldots +$

$+ (-1)^{n-2}|A_1 \cap \ldots \cap A_{n-1} \cap A_n|.$

Тогда

$$\left|\cup^{n}_{i=1}A_i\right| = \left|\left(\cup^{n-1}_{i=1}A_i\right) \cup A_n\right| =$$
$$= \left|\cup^{n-1}_{i=1}A_i\right| + |A_n| - \left|\left(\cup^{n-1}_{i=1}A_i\right) \cap A_n\right| =$$
$$= \left(\sum^{n-1}_{i=1}|A_i| - \sum_{1\le i<j\le n-1}|A_i \cap A_j| + ... + (-1)^{n-2}|A_1 \cap ... \cap A_{n-1}|\right) + |A_n| -$$
$$- \left(\sum^{n-1}_{i=1}|A_i \cap A_n| - \sum_{1\le i<j\le n-1}|A_i \cap A_j \cap A_n| + ... +$$
$$+ (-1)^{n-2}|A_1 \cap ... \cap A_{n-1} \cap A_n|\right) =$$
$$= \left(\sum^{n-1}_{i=1}|A_i| + |A_n|\right) - \left(\sum_{1\le i<j\le n-1}|A_i \cap A_j| + \sum^{n-1}_{i=1}|A_i \cap A_n|\right) + ... -$$
$$- (-1)^{n-2}|A_1 \cap ... \cap A_{n-1} \cap A_n| =$$
$$= \sum^{n}_{i=1}|A_i| - \sum_{1\le i<j\le n}|A_i \cap A_j| + ... + (-1)^{n-1}|A_1 \cap ... \cap A_n|. \quad \square$$

Замечание. Обозначения сумм с неравенствами в пределах суммирования, использованные в формулировке и доказательстве теоремы, являются не более чем сокращённой формой записи кратных сумм. Например, $\sum_{1\le i<j\le n}$ означает $\sum^{n-1}_{i=1}\sum^{n}_{j=i+1}$.

4.5.3. Число булевых функций, существенно зависящих от всех своих переменных

Рассмотрим применение формулы включений и исключений на примере следующей задачи. Пусть p_n **:=** $|P_n| = 2^{2^n}$ — число всех булевых функций n переменных, а q_n — число булевых функций, существенно зависящих от всех n переменных. Пусть P^i_n — множество булевых функций, у которых переменная x_i фиктивная (кроме x_i могут быть и другие фиктивные переменные). Имеем:

$$q_n = |P_n \setminus (P^1_n \cup ... \cup P^n_n)| = |P_n \setminus \cup^{n}_{i=1}P^i_n|.$$

С другой стороны, $|P^i_n| = 2^{2^{n-1}}$, более того, $|P^{i1}_n \cap ... \cap P^{ik}_n| = 2^{2^{n-k}}$. Следовательно,

$$q_n = 2^{2^n} - \left(\sum^{n}_{i=1}|P^i_n| - \sum_{1\le i<j\le n}|P^i_n \cap P^j_n| + ... + (-1)^{n-1}|P^1_n \cap ... \cap P^n_n|\right) =$$
$$= 2^{2^n} - \left(C(n,1)2^{2^{n-1}} - C(n,2)2^{2^{n-2}} + ... + (-1)^{n-1}C(n,n)2\right) =$$

$= \sum^{n}_{i=0} (-1)^i C(n, i) 2^{2^{n-i}}$.

4.6. ФОРМУЛЫ ОБРАЩЕНИЯ

Очень полезную, но специфическую группу приёмов образуют различные способы преобразования уже полученных комбинаторных выражений. В этом разделе рассматривается один частный, но важный случай.

4.6.1. Теорема обращения

Пусть $a_{n,k}$ и $b_{n,k}$ — некоторые (комбинаторные) числа, зависящие от параметров n и k, причём $0 \le k \le n$. Если известно выражение чисел $a_{n,k}$ через числа $b_{n,k}$, то в некоторых случаях можно найти и выражение чисел $b_{n,k}$ через числа $a_{n,k}$, то есть решить комбинаторное уравнение.

Теорема. *Пусть*

$$\forall n \, (\forall k \le n \, (a_{n,\,k} = \sum^{n}_{i=0} \lambda_{n,\,k,\,i} \, b_{n,\,i}))$$

и пусть

$$\exists \mu_{n,\,k,\,i} \Big(\forall k \le n \Big(\forall m \le n \Big(\sum^{n}_{i=0} \mu_{n,\,k,\,i} \, \lambda_{n,\,i,\,m} = \text{if} \;\; = \\ = k \; \textbf{then} \; 1 \; \textbf{else} \; 0 \; \textbf{end if} \Big) \Big) \Big).$$

Тогда

$$\forall k \le n \Big(b_{n,\,k} = \sum^{n}_{i=0} \mu_{n,\,k,\,i} \, a_{n,\,i} \Big).$$

Доказательство. $\sum^{n}_{i=0} \mu_{n,\,k,\,i} \, a_{n,\,i} = \sum^{n}_{i=0} \mu_{n,\,k,\,i} \Big(\sum^{n}_{m=0} \lambda_{n,\,i,\,m} \, b_{n,\,m} \Big) =$

$= \sum^{n}_{m=0} \Big(\sum^{n}_{i=0} \mu_{n,\,k,\,i} \, \lambda_{n,\,i,\,m} \Big) b_{n,\,m} = b_{n,\,k}$. □.

4.6.2. Формулы обращения для биномиальных коэффициентов

Применение теоремы обращения предполагает отыскание для заданных чисел $\lambda_{n,\,k,\,i}$ (коэффициентов комбинаторного уравнения) соответствующих чисел $\mu_{n,\,k,\,i}$, удовлетворяющих условию теоремы обращения. Особенно часто числами $\lambda_{n,\,k,\,i}$ являются биномиальные коэффициенты.

Лемма. $\sum_{i=0}^{n} (-1)^{i-m} C(n, i)C(i, m) =$ **if** $m = n$ **then** 1 **else** 0 **end if**.

Доказательство. Используя формулу 3 из параграфа 5.3.1 и тот факт, что $C(n - m, i - m) = 0$ при $i \le m$, имеем:

$$\sum_{i=0}^{n} (-1)^{i-m} C(n, i)C(i, m) = \sum_{i=0}^{n} (-1)^{i-m} C(n, m)C(n - m, i - m) =$$

$$= \sum_{i=m}^{n} (-1)^{i-m} C(n, m)C(n - m, i - m) =$$

$$= C(n, m)\sum_{i=m}^{n} (-1)^{i-m} C(n - m, i - m).$$

Но при $m < n$ имеем:

$$\sum_{i=m}^{n} (-1)^{i-m} C(n - m, i - m) = \sum_{j=0}^{n-m} (-1)^{j} C(n - m, j) = 0,$$

где $j := i - m$. С другой стороны, при $m = n$ имеем:

$$C(n, m)\sum_{i=m}^{n} (-1)^{i-m} C(n - m, i - m) = C(n, n)(-1)^{n-n} C(0, 0) = 1. \square$$

Теорема 1.

Если $a_{n,k} = \sum_{i=0}^{k} C(k, i)b_{n,i}$, *то* $b_{n,k} = \sum_{i=0}^{k} (-1)^{k-i} C(k, i)a_{n,i}$.

Доказательство. Здесь $\lambda_{n,k,i} = C(k, i)$ и $\mu_{n,k,i} = (-1)^{k-i} C(k, i)$.

При $k \le n$, $m \le n$ имеем:

$$\sum_{i=0}^{n} \mu_{n,k,i} \lambda_{n,i,m} = \sum_{i=0}^{n} (-1)^{k-i} C(k, i)\, C(i, m) =$$

$$= \sum_{i=0}^{k} (-1)^{k-i+m-m} C(k, i)\, C(i, m) =$$

$$= (-1)^{k-m} \sum_{i=0}^{k} (-1)^{i-m} C(k, i)\, C(i, m) =$$

= **if** $k = n$ **then** 1 **else** 0 **end if**, так как $C(k, i) = 0$ при $i > k$. $\square$

Теорема 2.

Если $a_{n,k} = \sum_{i=k}^{n} C(i, k)b_{n,i}$, *то* $b_{n,k} = \sum_{i=k}^{n} (-1)^{i-k} C(i, k)a_{n,i}$.

Доказательство. Здесь $\lambda_{n,k,i} = C(i, k)$ и $\mu_{n,k,i} = (-1)^{i-k} C(i, k)$.

При $k \le n$, $m \le n$ имеем:

$$\sum_{i=0}^{n} \mu_{n,k,i} \lambda_{n,i,m} = \sum_{i=0}^{n} (-1)^{i-k} C(i, k)\, C(m, i) =$$

$$= \sum_{i=0}^{n} (-1)^{i-k} C(m, i)\, C(i, k) = \textbf{if } k = n \textbf{ then } 1 \textbf{ else } 0 \textbf{ end if}. \quad \square$$

4.6.3. Формулы для чисел Стирлинга

В качестве примера использования формул обращения рассмотрим получение явных формул для чисел Стирлинга первого и второго рода. Рассмотрим множество функций f: $A \to B$, где $|A| = n$ и $|B| = k$. Число всех таких функций равно k^n. С другой стороны, число функций f, таких, что $|f(A)| = i$, равно $s(n, i)$, поскольку $s(n, i)$ — это число сюръективных функций f: $1..n \to 1..i$. Но множество значений функции (при заданном i) можно выбрать $C(k, i)$ способами. Поэтому

$$k^n = \sum_{i=0}^{k} C(k, i)s(n, i).$$

Обозначив $a_{n,\,k} := k^n$ и $b_{n,\,i} := s(n, i)$, имеем по теореме 1 предыдущего параграфа $s(n, k) = \sum_{i=0}^{k} (-1)^{k-i} C(k, i)i^n$. Учитывая связь чисел Стирлинга первого и второго рода, окончательно имеем:

$$S(n, k) = [1/n!] \sum_{i=0}^{k} (-1)^{k-i} C(k, i)i^n.$$

4.7. ПРОИЗВОДЯЩИЕ ФУНКЦИИ

Для решения комбинаторных задач в некоторых случаях можно использовать методы математического анализа. Разнообразие применяемых здесь приёмов весьма велико и не может быть в полном объёме рассмотрено в рамках этой книги. В данном разделе рассматривается только основная идея метода производящих функций, применение которой иллюстрируется тремя простыми примерами. Более детальное рассмотрение можно найти в литературе.

4.7.1. Основная идея

Пусть есть последовательность комбинаторных чисел a_i и последовательность функций $\varphi_i(x)$. Рассмотрим формальный ряд

$$\mathcal{F}(x) := \sum_i a_i \varphi_i(x).$$

$\mathcal{F}(x)$ называется *производящей функцией* (для заданной последовательности комбинаторных чисел a_i относительно заданной последовательности функций $\varphi_i(x)$). Обычно используют $\varphi_i(x) := x^i$ или $\varphi_i(x) := x^i/i!$.

Пример. Из формулы бинома Ньютона при $y = 1$ имеем:

$$(1 + x)^n = \sum^n_{i=0} C(n, i)\, x^i.$$

Таким образом, $(1 + x)^n$ является производящей функцией для биномиальных коэффициентов.

4.7.2. Метод неопределённых коэффициентов

Из математического анализа известно, что если

$$\mathcal{F}(x) = \sum_i a_i\varphi_i(x) \text{ и } \mathcal{F}(x) = \sum_i b_i\varphi_i(x),$$

то $\forall i\ (a_i = b_i)$ (для рассматриваемых здесь систем функций φ_i). В качестве примера применения производящих функций докажем следующее тождество.

Теорема. $C(2n, n) = \sum^n_{k=0} C(n, k)^2$.

Доказательство. Имеем: $(1 + x)^{2n} = (1 + x)^n(1 + x)^n$. Следовательно,

$$\sum^{2n}_{i=0} C(2n, i)x^i = \sum^n_{i=0} C(n, i)x^i \cdot \sum^n_{i=0} C(n, i)x^i.$$

Приравняем коэффициент при x^n:

$$C(2n, n) = \sum^n_{k=0} C(n, k)C(n, n-k) = \sum^n_{k=0} C(n, k)^2. \quad \square$$

4.7.3. Числа Фибоначчи

Числа Фибоначчи $F(n)$ определяются следующим образом:

$$F(0) := 1,\ F(1) := 1,\ F(n + 2) := F(n + 1)+F(n).$$

Найдём выражение для $F(n)$ через n. Для этого найдём производящую функцию $\varphi(x)$ для последовательности чисел $F(n)$. Имеем:

$$\varphi(x) = \sum^{\infty}_{n=0} F(n)x^n =$$

$$= F(0)x^0 + F(1)x^1 + \sum^{\infty}_{n=2} \big(F(n-2) + F(n-1)\big)x^n =$$

$$= 1 + x + \sum^{\infty}_{n=2} F(n-2)x^n + \sum^{\infty}_{n=2} F(n-1)x^n =$$

$$= 1 + x + x^2\sum^{\infty}_{n=2} F(n-2)x^{n-2} + x\sum^{\infty}_{n=2} F(n-1)x^{n-1} =$$

$$= 1 + x + x^2\sum^{\infty}_{n=0} F(n)x^n + x\Big(\sum^{\infty}_{n=0} F(n)x^n - F(0)\Big) =$$

$$= 1 + x + x^2\varphi(x) + x\big(\varphi(x) - 1\big).$$

Решая это функциональное уравнение относительно $\varphi(x)$, получаем, что $\varphi(x) = 1/(1 - x - x^2)$. Последнее выражение нетрудно разложить в ряд по степеням x. Действительно, уравнение $1 - x - x^2 = 0$ имеет корни $x_1 = -(1 + \sqrt{5})/2$ и $x_2 = -(1 - \sqrt{5})/2$, причём, как нетрудно убедиться, $1 - x - x^2 = (1 - ax)(1 - bx)$, где $a = (1 + \sqrt{5})/2$, $b = (1 - \sqrt{5})/2$. Далее, легко видеть что $1/[(1 - ax)(1 - bx)] = \alpha/(1 - ax) + \beta/(1 - bx)$, где $\alpha = a/(a - b)$, $\beta = -b/(a - b)$.

Из математического анализа известно, что для малых x

$$1/(1 - \gamma x) = \sum^{\infty}_{n=0} \gamma^n x^n.$$

Таким образом,

$$\varphi(x) = \alpha/(1 - ax) + \beta/(1 - bx) = \alpha\sum^{\infty}_{n=0} a^n x^n + \beta\sum^{\infty}_{n=0} b^n x^n =$$

$$= \sum^{\infty}_{n=0} [a/(a - b)]a^n x^n - \sum^{\infty}_{n=0} [b/(a - b)]b^n x^n =$$

$$= \sum^{\infty}_{n=0} [a^{n+1} - b^{n+1}] / [a - b]\, x^n.$$

Окончательно получаем

$$F(n) = [a^{n+1} - b^{n+1}]/[a - b] = 1/\sqrt{5}\Big(\big([1 + \sqrt{5}]/2\big)^{n+1} - \big([1 - \sqrt{5}]/2\big)^{n+1}\Big).$$

4.7.4. Числа Каталана

Число Каталана $C(n)$ можно определить следующим образом:

$$C(0) := 1,\ C(n) := \sum^{n-1}_{k=0} C(k)C(n - k - 1).$$

Числа Каталана используются при решении различных комбинаторных задач.

Пример. Пусть $\langle S, * \rangle$ — полугруппа и нужно вычислить элемент $s_1 * \ldots * s_n$. Сколькими способами это можно сделать? То есть сколькими способами можно расставить скобки, определяющие порядок вычисления выражения? Обозначим число способов $C(n)$. Ясно, что $C(0) = 1$. При любой схеме вычислений на каждом шаге выполняется некоторое вхождение операции * над соседними элементами и результат ставится на их место. Пусть последней выполняется то вхождение операции *, которое имеет номер k в исходном выражении. При этом слева от выбранного вхождения знака * находилось и было выполнено $k-1$ знаков операции *, а справа, соответственно, $(n-1)-(k-1) = n-k$ знаков операции *. Тогда ясно, что

$$C(n) = \sum_{k=1}^{n} C(k-1)C(n-k) = \sum_{k=0}^{n-1} C(k)C(n-k-1),$$

и ответом на поставленный вопрос является число Каталана $C(n)$.

Числа Каталана выражаются через биномиальные коэффициенты. Получим это выражение, используя метод производящих функций.

Теорема. $C(n) = C(2n, n) / (n+1)$.

Доказательство. Найдём производящую функцию для чисел Каталана $\varphi(x) = \sum_{n=0}^{\infty} C(n)\, x^n$. Для этого рассмотрим квадрат этой функции: $\varphi^2(x) = \left(\sum_{n=0}^{\infty} C(n)\, x^n\right)^2 = \sum_{m=0}^{\infty} C(m)\, x^m \cdot \sum_{n=0}^{\infty} C(n)\, x^n =$

$= \sum_{m,\, n=0}^{\infty} C(m)C(n)\, x^{m+n} = \sum_{n=0}^{\infty}\sum_{k=0}^{n} C(k)\, C(n-k)\, x^n =$

$= \sum_{n=0}^{\infty} C(n+1)\, x^n = \sum_{n=0}^{\infty} C(n+1)\, x^{n+1}/x = 1/x\left(\sum_{n=0}^{\infty} C(n)\, x^n - C(0)\right) =$

$= 1/x\left(\varphi(x) - 1\right)$.

Решая уравнение $\varphi(x) = x\varphi^2(x) + 1$ относительно функции $\varphi(x)$, имеем:

$$\varphi(x) = [1 \pm (1-4x)^{1/2}]/[2x].$$

Обозначим $f(x) := \sqrt{(1 - 4x)}$ и разложим $f(x)$ в ряд по формуле Тейлора:

$$f(x) = f(0) + \sum^{\infty}_{k=1} [f^{(k)}(0)]/[k!]x^k.$$

Имеем: $f^{(k)}(x) = 1/2\cdot(1/2 - 1)\cdot\ldots\cdot(1/2 - k + 1)\cdot(1 - 4x)^{1/2}\cdot(-4)^k =$

$= -2^k\cdot 1\cdot 3\cdot\ldots\cdot(2k - 3)\cdot(1 - 4x)^{1/2 - k} = -2^k\cdot(2k - 3)!!\cdot(1 - 4x)^{1/2 - k} =$

$= -2^k[(2k - 3)!]/[2^{k-2}(k - 2)!](1 - 4x)^{1/2 - k} =$

$= -2^2[(2k - 2)!(k - 1)(k - 1)!]/[(2k - 2)(k - 1)!(k - 1)!](1 - 4x)^{1/2 - k} =$

$= -2(k - 1)![(2k - 2)!]/[(k - 1)!(k - 1)!](1 - 4x)^{1/2 - k} =$

$= -2(k - 1)!C(2k - 2, k - 1)(1 - 4x)^{1/2 - k}.$

Таким образом,

$$f^{(k)}(0) = -2(k - 1)!C(2k - 2, k - 1)$$

и

$$f(x) = 1 - 2\sum^{\infty}_{k=1} [1/k]\cdot C(2k - 2, k - 1)x^k.$$

Подставляя выражение $f(x)$ в формулу для $\varphi(x)$, следует выбрать знак "минус" перед корнем, чтобы удовлетворить условию $C(0) = 1$. Окончательно имеем: $\sum^{\infty}_{n=0} C(n)x^n = \varphi(x) = [1 - f(x)]/[2x] =$

$= [1 - 1 + 2\cdot\sum^{\infty}_{n=1} 1/n\cdot C(2n - 2, n - 1)\,x^n] / [2x] =$

$= \sum^{\infty}_{n=1} 1/n\cdot C(2(n - 1), n - 1)\,x^{n-1} = \sum^{\infty}_{n=0} C(2n, n)/(n + 1)x^n$, и по методу неопределённых коэффициентов $C(n) = C(2n, n)/(n + 1)$. □

ВЫВОДЫ

Комбинаторные формулы являются основным инструментов оценки сложности практических алгоритмов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Дэвенпорт Дж., Сирэ И., Турнье Э. Компьютерная алгебра. — М.: Мир, 1991.

2. Кокс Д., Литтл Дж., О'Ши Д. Идеалы, многообразия и алгоритмы. — М. Мир, 2000.

3. Р. Лиддл, Г. Нидеррайтер. Конечные поля. т. 1, 2. — М.: Мир, 1988.

4. Атья М., Макдональд И. Введение в коммутативную алгебру. — М: Мир, 1972.

5. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. Перевод с английского. — М.: Мир, 1979.

6. Кнут Д. Искусство программирования для ЭВМ. Т. 2. Получисленные алгоритмы. Т. 2. — М.: Мир, 1977.

7. Акритас А. Основы компьютерной алгебры с приложениями. — М.: Мир, 1994.

8. Новиков Ф.А. Дискретная математика для программистов: Учебник для вузов. 3-е изд. — СПб.: Питер, 2009.

9. Босс В. Лекции по математике. Т. 6: От Диофанта до Тьюринга. — М.: КомКнига, 2006.

10. Мальцев А. Н. Алгоритмы и рекурсивные функции. — М., Наука, 1986.

11. Гиндикин С. Г. Алгебра логики в задачах. — М., Наука, 1972.

12. Гаврилов Г. П., Сапоженко А. А. Задачи и упражнения по дискретной математике: Учеб. пособие. 3-е изд., перераб. — М.: ФИЗМАТЛИТ, 2005.

13. Успенский В. А., Верещагин Н. К., Плиско В. Е. Вводный курс математической логики. 2-е изд. — М.: ФИЗМАТЛИТ, 2004.

14. Аржанцев И. В. Базисы Гребнера и системы алгебраических уравнений. — М.: МЦНМО, 2003.

15. Мендельсон Э. Введение в математическую логику. — М., Наука, 1971.

16. Виленкин Н. Я. Комбинаторика. — М.: Наука, 1969.

17. Хаггарти Р. Дискретная математика для программистов — М.:

Техносфера, 2003.

18. Кузнецов О. П.. Адельсон-Вельский Г. М. Дискретная математика дли инженера — М.: Энергия, 1980.

19. Липский В. Комбинаторика для программистов. — М.: Мир, 1988.

20. Кук Д., Бейз Г. Компьютерная математика. — М., Наука, 1990.

21. Р. Грэхем, Д. Кнут, О. Паташник. Конкретная математика. Основание информатики. — М., Мир, 1998.

22. Асанов М. О., Баранский В. А., Расин В. В. Дискретная математика: графы, матроиды, алгоритмы. — Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001.

23. Андерсон Джеймс А. Дискретная математика и комбинаторика. — М.: Вильямс, 2004.

Васильев Николай Николаевич
Новиков Федор Александрович

КОМПЬЮТЕРНАЯ АЛГЕБРА.

Часть I.
ДИСКРЕТНАЯ МАТЕМАТИКА,
ТЕОРИЯ АЛГОРИТМОВ

Учебное пособие

Лицензия ЛР № 020593 от 07.08.97

Налоговая льгота – Общероссийский классификатор продукции
ОК 005-93, т. 2; 95 3005 – учебная литература

Подписано в печать . .2011. Формат 60×84/16 Печать цифровая.
Усл. печ. л. . Уч.-изд. л. . Тираж . Заказ

Отпечатано с готового оригинал-макета, предоставленного автором
в цифровом типографском центре Издательства Политехнического
университета.
195251, Санкт-Петербург, Политехническая ул., 29.
Тел. (812) 540-40–14
Тел./факс: (812) 927-57-76