В.Д.Гоппа

ВВЕДЕНИЕ В АЛГЕБРАИЧЕСКУЮ ТЕОРИЮ ИНФОРМАЦИИ

М.: Наука. Физматлит, 1995.—112 с.

Развивается алгебраический подход к теории информации. Теория информации трактуется как абстрактная теория слов со своими специфическими задачами, связанными с хранением слов в памяти компьютера, обработкой слов и их передачей по каналам связи. На множестве слов канонически присутствует алгебраическая структура, связанная с действием симметрической группы на словах. Эта структура используется для определения информации слова с различными приложениями к информатике.

Книга предназначена для широкого круга читателей.

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

В книге развивается алгебраический подход к теории информации. Теория информации трактуется как абстрактная теория слов со своими специфическими задачами, связанными с хранением слов в памяти компьютера, обработкой слов и их передачей по каналам связи. На множестве слов канонически присутствует алгебраическая структура, связанная с действием симметрической группы на словах. Эта структура используется для определения информации слова с различными приложениями к информатике.

Книга состоит из пяти глав и исторического очерка.

В первой главе вводятся основные понятия теории информации и доказываются теоремы кодирования. Во второй главе веденные понятия применяются для распознавания образов. Приводятся алгоритмы вычисления информативности признака и кластерного анализа в пространстве признаков. Третья глава посвящена реляционным базам данных. Основное внимание уделяется синтезу баз данных на основе декомпозиции. В четвертой главе анализируется связь между словами и графами. Рассматриваются основные алгоритмы на графах, которые находят применение при сжатии информации, распознавании образов и синтезе баз данных. В последней главе изучается память слов и основные процессы, связанные с хранением и преобразованием генетической информации (молекулярная биология с точки зрения информатики).

Книга написана на основе курса лекций, прочитанных автором в 1986—1990 г.г. на Специальном факультете информатики в науках о Земле Московского геологоразведочного института. Автор благодарен декану факультета А.А. Виноградову за любезное приглашение прочитать этот курс лекций и слушателям, чьи замечания автор учел при написании книги.

Лекции сопровождались лабораторными занятиями в компьютерном классе, которые проводил Е.Н. Городничев. Многие результаты, включенные в книгу, появились в процессе нашей совместной работы, так что его можно считать полноправным соавтором книги.

Чрезвычайно полезными оказались беседы с О.Н. Ковалевой о проблемах, связанных с реляционными базами данных. Консультантом по генетике явился мой сын В.В. Гоппа, окончивший биофак МГУ. Всем этим лицам выражаю глубокую благодарность.

1. ИНФОРМАЦИЯ СЛОВ И ТЕОРЕМЫ КОДИРОВАНИЯ

1.1. Информация по Хартли

Пусть Ω—конечное множество, $|\Omega|$—его мощность. *Информация элемента* $\omega \in \Omega$ определяется по Хартли [17] следующим образом:

$$I(\omega) = \log_2 |\Omega|.$$

Эта величина равна с точностью до округления числу двоичных символов необходимых для записи элемента ω (индивидуализации элемента множества).

1.2. Отношение эквивалентности

Пусть теперь на множестве Ω задано отношение эквивалентности , т.е. рефлексивное, симметричное и транзитивное бинарное отношение. Это отношение порождает разбиение Ω на непересекающиеся классы.

Множество классов эквивалентности называется фактормножеством и обозначается $\Omega/\approx$. Пусть π: $\Omega \to \Omega/\approx$ обозначает каноническое проектирование, которое каждому элементу $\omega \in \Omega$ ставит в соответствие его класс эквивалентности $\pi(\omega)$. Информацию элемента ω определим в виде

$$I(\omega) = \log_2 |\pi(\omega)|.$$

В предельном случае, когда существует только один класс эквивалентности, информация каждого элемента одинакова и совпадает с информацией по Хартли.

Предложение 1.1. *Если N обозначает число классов эквивалентности*:

$$N = |\Omega/\approx|,$$

то

$$\sum_{\omega \in \Omega} 2^{-(I(\omega)+\log N)} = 1.$$

Доказательство. Пусть W_i обозначает i-й класс. Тогда

$$\sum_{i=1}^{N} \left(2^{-\log |W_i| - \log N}\right) |W_i| = \sum_{i=1}^{N} \frac{1}{N} = 1.$$

1.3. Неравномерное кодирование слов

Пусть X—конечное множество (алфавит), $|X| = q$, X^n—множество слов длины n в алфавите X. Обозначим через B двоичный алфавит, так что $B = \{0, 1\}$, а через B^*—множество всех слов конечной длины в алфавите B. *Неравномерным кодом* (*сжимающим отображением*) называют отображение

$$\Psi: X^n \to B^*,$$

определенное для любого $n = 1, 2, \ldots$

Код называется *префиксным* или *дешифрируемым*, если $\psi(x)$ не является началом никакого другого кодового слова $\psi(x')$, $x' \neq x$. Длину кодового слова $\psi(x)$ обозначим через $l(x)$.

Слова префиксного кода можно записывать в одну цепочку, не используя разделительных символов (запятых). Любую такую цепочку можно единственным образом разложить на ее компоненты.

Предложение 1.2 (неравенство Крафта). *Для существования префиксного кода с длинами слов* $n_1, \ldots, n_N$ *необходимо и достаточно, чтобы*

$$\sum_{i=1}^{N} 2^{-n_i} \leq 1.$$

Доказательство. Если у префиксного кода существует слово длины j, то все 2^{n-j} слов длины n ($n > j$), получающиеся добавлением $n - j$ символов, не могут входить в префиксный код. Если обозначить через S_j число кодовых слов длины j, то получаем неравенство

$$S_n \leq 2^n - S_1 \cdot 2^{n-1} - S_2 \cdot 2^{n-2} - \ldots - S_{n-1} \cdot 2,$$

справедливое для любого $n = 2, 3, \ldots$ Ясно, что это условие является также и достаточным для построения префиксного кода с соответствующими $\{S_j\}$. Разделив на 2^n, получаем

$$\sum_{j=1}^{n} S_j \cdot 2^{-j} \leq 1,$$

что и требовалось доказать, поскольку

$$\sum_{j=1}^{n} S_j \cdot 2^{-j} = \sum_{n_j \leq n} 2^{-n_j}.$$

Сопоставляя предложения 1 и 2, заключаем, что существует префиксный код с длинами двоичных слов

$$l(\omega) = I(\omega) + \log N, \qquad l(\omega) =]I(\omega) + \log N[,$$

где $]a[$ обозначает ближайшее целое, большее или равное a.

Ясно также, как осуществить кодирование: сначала нужно указать номер класса W_i, содержащего ω (для этого потребуется префикс, содержащий $]\log N[$ двоичных символов), а затем указать номер элемента ω в классе W_i (для этого потребуется еще $]\log |W_i|[$ символов). В дальнейшем основание логарифма предполагается равным 2.

1.4. Действие группы на множестве

Рассмотрим наряду с множеством Ω конечную группу G. Говорят, что G *действует на* Ω, если задано отображение $G \times \Omega$ такое, что

$$(\sigma\tau)\omega = \sigma(\tau\omega), \qquad e\omega = \omega$$

для всех $\sigma, \tau \in G$ и $\omega \in \Omega$; здесь $\tau\omega$ обозначает образ пары (τ, ω) при этом отображении, а e—единица группы G.

В этом случае Ω называют *G-множеством*. Множество элементов $\tau \in G$, для которых $\tau\omega = \omega$ есть, очевидно, подгруппа в G. Она называется *стационарной подгруппой элемента* ω и обозначается G_ω. Подмножество в Ω, состоящее из всех элементов вида $\sigma\omega$ (где $\sigma \in G$) обозначается $G\omega$ и называется *орбитой элемента* ω. Если σ и τ лежат в одном и том же (смежном) классе по $H = G_\omega$, то $\sigma\omega = \tau\omega$, и обратно. Таким образом, существует взаимно однозначное соответствие между левыми смежными классами G/H и элементами орбиты $G\omega$. Следовательно, порядок орбиты $G\omega$ совпадает с индексом $[G:G_\omega]$. Заметим, что G_ω не является, вообще говоря, нормальной подгруппой в G.

Две орбиты группы G либо не пересекаются, либо совпадают. Таким образом, на G-множестве существует каноническое разбиение

$$\Omega = \sum_{i=1}^{N} G\omega_i.$$

Поэтому на Ω можно определить информацию разбиения

$$I(\omega) = \log |G\omega| = \log [G:G_\omega].$$

1.5. 0-информация слова

Пусть X—конечное множество (алфавит), $X^n = \Omega$—множество всех слов (последовательностей букв) длины n в алфавите X. На

X^n канонически действует симметрическая группа S_n, переставляющая позиции букв слова. Поэтому X^n является S_n-множеством и можно ввести информацию разбиения. Для произвольного слова $\omega \in X^n$, $\omega = (a_1, a_2, \ldots, a_n)$ и произвольной подстановки $\sigma \in S_n$ действие S_n описывается как

$$\sigma(\omega) = (a_{\sigma(1)}, a_{\sigma(2)}, \ldots, a_{\sigma(n)}).$$

Если буква $a_i \in X$ входит в слово $\omega \in X^n$ ровно m_i раз, $i = 1, 2, \ldots, |X| = q$, то разбиение $n = m_1 + \ldots + m_q$ определяет *композицию* $(m_1, \ldots, m_q)$ слова ω. Ясно, что все слова одной и той же орбиты имеют одну и ту же композицию и обратно. Величину

$$I_0 = \log |G\omega|,$$

где $G = S_n$, назовем 0-*информацией слова* ω.

Стационарная подгруппа G_ω является в данном случае прямым произведением симметрических групп

$$S_{m_1} \times S_{m_2} \times \ldots \times S_{m_q},$$

поэтому

$$I_0(\omega) = \log \frac{n!}{m_1! m_2! \ldots m_q!}.$$

Пример 1.1. Слово $\omega_1 = (aaaa)$ имеет композицию (4), поэтому $I_0(\omega_1) = 0$. Слово $\omega_2 = (abaa)$ имеет композицию (3, 1), поэтому

$$I_0(\omega_2) = \log \frac{4!}{3!1!} = 2 \text{ бита}.$$

Для слова $\omega_3 = (abab)$ имеем композицию (2, 2) и

$$I_0(\omega) = \log \frac{4!}{2!2!} = \log 6.$$

Таким образом, чем меньше симметрий в слове, тем больше его 0-информация. Термин «0-информация» подсказывает, что далее будут даны обобщения понятия информации, основанного на изучении симметрий в расположении букв. Наличие симметрий позволяет эффективно кодировать (сжимать) это слово. С другой стороны, случайно выписанная последовательность букв не содержит симметрий.

Минимально возможное значение 0-информации равно нулю. Это значение достигается только для слов, состоящих из одной и той же буквы.

Максимальное значение 0-информация принимает на словах, в которых все буквы встречаются одинаково часто (это следует из известных свойств полиномиальных коэффициентов).

1.6. Условная 0-информация слова

Рассмотрим наряду с алфавитом X и множеством X^n еще один алфавит Y (который может совпадать с X) и соответственно множество слов Y^n.

Пусть $y \in Y^n$, G_y—стационарная подгруппа этого слова. Группа G_y, являясь подгруппой симметрической группы S_n, также действует на X^n, так что X^n является G_y-множеством и разбивается на «условные орбиты». Информацию, связанную с этим разбиением, назовем *условной 0-информацией слова* $x \in X^n$ при условии $y \in Y^n$ и обозначим $I_0(x/y)$. Ясно, что

$$I_0(x/y) = \log [G_y:(G_x \cap G_y)].$$

Введем еще одно определение. Декартово произведение $X^n \otimes Y^n$ можно трактовать как множество слов Z^n в алфавите $X \otimes Y = Z$. Если $x = (a_1, \ldots, a_n)$, $y = (b_1, \ldots, b_n)$, то слово $x \otimes y = ((a_1, b_1), \ldots, (a_n, b_n))$ будем называть *произведением слов* x и y.

Предложение 1.3.

$$I_0(x \otimes y) = I_0(x) + I_0(y/x) = I_0(y) + I_0(x/y).$$

Доказательство. Легко проверяется, что $G_{x \otimes y} = G_x \cap G_y$. Далее применяем теорему Лагранжа о подгруппах:

$$[G:(G_x \cap G_y)] = [G:G_x]\,[G_x:(G_x \cap G_y)] = [G:G_y]\,[G_y:(G_x \cap G_y)].$$

Переходя к логарифмам, получаем требуемый результат.

Предложение 1.4.

$$I_0(y/x) \le I_0(y), \quad I_0(x \otimes y) \le I(x) + I(y).$$

Доказательство. Первое неравенство очевидно, поскольку условная орбита элемента y является подорбитой безусловной орбиты ($G_x y \subseteq Gy$). Второе неравенство является следствием первого и предложения 1.3.

Предложение 1.5.

$$I_0(x/y \otimes z) \le I_0(x/y), \quad I_0(x \otimes y/z) = I_0(x/z) + I_0(y/x \otimes z).$$

Доказательство. $G_{y \otimes z} = G_y \cap G_z \subseteq G_y$. Отсюда следует, что орбита любого элемента x под действием $G_{y \otimes z}$ является подорбитой орбиты этого элемента под действием G_y. Тем самым доказано первое неравенство. Доказательство второго утверждения следует из цепочки равенств:

$$[G_z:(G_z \cap G_{x \otimes y})] = [G_z:(G_x \cap G_y \cap G_z)] =$$

$$= [G_z:(G_z \cap G_x)] \times [(G_z \cap G_x):(G_x \cap G_y \cap G_z)].$$

Предложение 1.6 (функциональная зависимость). *Следующие утверждения равносильны:*

1) $I_0(x/y) = 0$, 2) $G_y \subseteq G_x$, 3) $b_{j_1} = b_{j_2} \Rightarrow a_{j_1} = a_{j_2}$, 4) $y \Rightarrow x$.

Доказательство. Если

$$I_0(x/y) = \log [G_y:(G_x \cap G_y)] = 0,$$

то

$$G_y = G_x \cap G_y, \quad \text{т. е. } G_y \subseteq G_x.$$

Это означает, что каждая буква b слова y, на какой бы позиции j_1 она ни стояла, всегда переходит в одну и ту же букву a слова x. Это значит, что существует функциональная зависимость $y \to x$ между словами x и y.

Пример 1.2.

$$x = (a\,c\,c\,b\,a\,b\,c\,c),$$

$$y = (d\,a\,a\,e\,d\,e\,a\,b).$$

Мы видим, что всегда $d \Rightarrow a$, $a \Rightarrow c$, $e \Rightarrow b$, $b \Rightarrow c$.

Таким образом, *значение каждой координаты слова x однозначно определяется соответствующей координатой слова y, что и называется, по определению, функциональной зависимостью $y \to x$.*

Обратной зависимости $x \to y$ в данном примере не существует, поскольку буква c переходит иногда в букву a, а иногда в букву b слова y. Поэтому

$$I_0(y/x) \neq 0.$$

$I_0(x/y)$ и $I_0(y/x)$ одновременно равны нулю тогда и только тогда, когда слово y получено из слова x в результате переименования букв, *причем разные буквы слова x* имеют различные новые названия.

Два слова x и y, связанные взаимной функциональной зависимостью, будем называть *эквивалентными*:

$$x \approx y \Leftrightarrow I_0(y/x) = I_0(x/y) = 0 \Leftrightarrow x \Rightarrow y, \quad y \Rightarrow x.$$

Предложение 1.7. *$I_0(y/x)$ является метрикой на орбите.*

Доказательство. Докажем неравенство треугольника:

$$I_0(x/y) + I_0(y/z) \geq I_0(x/y \otimes z) + I_0(y/z) = I_0(x \otimes y/z) \geq I_0(x/z).$$

Отсюда следует, что

$$I_0(x/z) \leq I_0(x/y) + I_0(y/z).$$

Далее, если x и y принадлежат одной и той же орбите, то

$$I_0(x/y) = I_0(y/x).$$

Действительно, в этом случае $I_0(x) = I_0(y)$, а

$$I_0(x \otimes y) = I_0(x) + I_0(y/x) = I_0(y) + I_0(x/y).$$

Поэтому $I_0(x/y)$ симметрична относительно x и y. Наконец, $I_0(x/y) = 0$ означает, что $x \approx y$. Таким образом, $I_0(x/y)$ является метрикой на классах эквивалентных слов.

Метрику $I_0(x/y)$ будем называть *информационной.* Если x и y не принадлежат одной орбите, то $I_0(y/x)$ не равно $I_0(x/y)$. Можно положить

$$\rho(x, y) = \frac{1}{2}(I_0(y/x) + I_0(x/y)),$$

и эта величина будет метрикой уже для любых слов одной и той же длины.

Введем, наконец, следующее определение. Величину

$$I_0(x{:}y) = I_0(x) + I_0(y) - I_0(x \otimes y)$$

назовем *взаимной 0-информацией слов x и y, или информацией, содержащейся в слове x относительно слова y.*

Из предложения 1.4 следует, что эта величина неотрицательна, а из предложения 1.3 следует, что

$$I_0(x{:}y) = I_0(x) - I_0(x/y) = I_0(y) - I_0(y/x).$$

1.7. Вычисление условной информации

Поскольку мы рассматриваем слова с точностью до эквивалентности, то каждое слово однозначно определяется своей композицией и перечислением номеров позиций, занятых одной и той же буквой.

П р и м е р 1.3. Слово $x = (abaacbca)$ можно задать в виде

$$x = \begin{vmatrix} 1 & 3 & 4 & 8 \\ 2 & 6 & & \\ 5 & 7 & & \end{vmatrix}.$$

Зная такую таблицу, можно однозначно восстановить слово x с точностью до переименования букв: на позициях 1, 3, 4 и 8 устанавливаем некоторую букву d:

d		*d*	*d*				*d*
1	2	3	4	5	6	7	8

Далее, на позициях 2 и 6 помещаем некоторую другую букву a:

d	*a*	*d*	*d*		*a*		*d*
1	2	3	4	5	6	7	8

Наконец, оставшиеся позиции занимаем третьей буквой, например b:

d	*a*	*d*	*d*	*b*	*a*	*b*	*d*
1	2	3	4	5	6	7	8

В результате получаем слово y, полученное из первоначального слова x переименованием букв: $a \to d$, $b \to a$, $c \to b$ (в криптографии такой прием называется шифром Цезаря).

Таким образом, каждое слово разбивается на несколько *канонических подслов* нулевой 0-информации. Например, слово x «склеивается» из следующих подслов:

a		*a*	*a*				*a*
1	2	3	4	5	6	7	8

	b				*b*		

				c		*c*	

Стационарная подгруппа G_x состоит из всех перестановок, оставляющих неподвижной каждую строку в таблице Юнга, или, что то же самое, каждое каноническое подслово. Поскольку

$$G_x = S_{m_1} \times S_{m_2} \times \ldots \times S_{m_q}$$

и каждая из этих симметрических групп действует независимо на своем множестве позиций, орбита слова y под действием G_x оказывается произведением орбит под действием канонических подслов слова x. Но условная орбита под действием канонического подслова совпадает с безусловной орбитой. Тем самым вычисление

условной информации сводится к вычислению безусловной информации.

Пусть x и y имеют композиции $(m_1, \ldots, m_q)$, $(n_1, \ldots, n_k)$ соответственно. Составляем матрицу переходов букв слова x в буквы слова y:

$$\begin{aligned} m_{11} + m_{12} + \ldots + m_{1k} &= m_1, \\ m_{21} + m_{22} + \ldots + m_{2k} &= m_2, \\ &\cdots\cdots\cdots \\ m_{q1} + m_{q2} + \ldots + m_{qk} &= m_q. \end{aligned}$$

Тогда

$$I_0(y/x) = \log \frac{m_1!}{m_{11}! \ldots m_{1k}!} + \log \frac{m_2!}{m_{21}! \ldots m_{2k}!} + \ldots + \log \frac{m_q!}{m_{q1}! \ldots m_{qk}!}.$$

Пример 1.4.

$$x = (a\, b\, a\, a\, c\, b\, c\, a),$$
$$y = (1\ 2\ 2\ 1\ 3\ 2\ 1\ 3),$$

	1	2	3	
a	2	1	1	4
b		2		2
c	1		1	2
	3	3	2	8

Матрица переходов составлена следующим образом: замечаем, что буква a переходит в символ 1 два раза, и по одному разу в символы 2 и 3 слова y. Тем самым получаем первую строку матрицы и подсчитываем сумму элементов этой строки. В конце определяем также сумму элементов каждого столбца. Тем самым в матрице присутствуют композиции слов x и y:

$$x = (4, 2, 2), \qquad y = (3, 3, 2),$$

а также общая длина слова:

$$n = 4 + 2 + 2 = 3 + 3 + 2 = 8.$$

$$I_0(y/x) = \log \frac{4!}{2!\,1!\,1!} + \log \frac{2!}{2!} + \log \frac{2!}{1!\,1!} = \log 12 + \log 2 = \log 24.$$

Производя такие же вычисления по столбцам, получаем $I_0(x/y)$:

$$I_0(x/y) = \log \frac{3!}{2!\,1!} + \log \frac{3!}{2!\,1!} + \log \frac{2!}{1!\,1!} = 2 \cdot \log 3 + \log 2 = \log 18.$$

Для безусловной информации находим значения

$$I_0(x) = \log \frac{8!}{4!\,2!\,2!} = \log 420, \quad I_0(y) = \log \frac{8!}{3!\,3!\,2!} = \log 560.$$

Вычисляем взаимную информацию:

$$I_0(x{:}y) = I_0(x) - I_0(x/y) = \log \frac{420}{18} = \log \frac{70}{3} =$$

$$= I_0(y) - I_0(y/x) = \log \frac{560}{24} = \log \frac{70}{3}.$$

Далее, слово $x \otimes y$ имеет композицию (2, 2, 1, 1, 1, 1,), поэтому

$$I_0(x \otimes y) = \log \frac{8!}{2!\,2!}.$$

1.8. Группировка наблюдений (квантование)

Выясним теперь, как меняется информация слова при отождествлении некоторых букв алфавита. При переходе к алфавиту $X' = X/\approx$ некоторые канонические подслова склеиваются. Пусть в примере 1.3 буквы b и c отождествляются. Тогда получаем вместо исходного слова x новое слово x':

$$x' = (a\,b\,a\,a\,b\,b\,b\,a),$$
$$x = (a\,b\,a\,a\,c\,b\,c\,a).$$

Имеет место функциональная зависимость $x \to x'$, вызванная тем, что $c \to b$. Поэтому

$$I_0(x{:}x') = I_0(x') - I_0(x'/x) = I_0(x').$$

С другой стороны,

$$I_0(x{:}x') = I_0(x) - I_0(x/x').$$

При вычислении $I_0(x/x')$ можно ограничиться лишь буквами b и c, которые отождествляются между собой:

	b	c	
b	2	2	4

$$I_0(x/x') = \log \frac{4!}{2!\,2!} = \log 6,$$

$$I_0(x') = I_0(x) - I_0(x/x') = \log \frac{420}{6} = \log 70.$$

Вычисляя непосредственно $I_0(x')$, получаем

$$I_0(x') = \log \frac{8!}{4!\,4!} = \log 70.$$

Резюмируем сказанное.

Предложение 1.8. *Информация слова уменьшается в результате отождествления некоторых букв алфавита (группировки наблюдений). Если x и x' обозначают соответственно исходное и новое слово, то*

$$I_0(x') = I_0(x) - I_0(x/x') = I_0(x{:}x').$$

Пусть в результате наблюдений получено слово (последовательность данных)

$$x = (0.16,\ 0.1,\ 0.7,\ 0.1,\ 0.18,\ 1.5).$$

Требуется превратить это слово в двоичное слово, выбрав соответствующие два интервала квантования. Естественное требование заключается в том, чтобы новое слово x' содержало как можно больше информации об исходном слове, т. е. чтобы как можно меньше информации было бы утеряно при квантовании. Таким образом, величина $I_0(x{:}x')$ должна быть максимальной. Но эта величина совпадает, согласно предложению 1.8, с величиной $I_0(x')$. Последняя величина примет наибольшее значение, когда 1 и 0 будут распределены поровну в слове x'. Приходим к квантованию по принципу «вариационного ряда»: нужно упорядочить все данные по возрастанию, т. е.

0.16	0.1	0.7	0.1	0.18	1.5
3	1	5	2	4	6

Первые три значения относим к первому классу, что соответствует интервалу $(-\infty;\ 0.16]$, а остальные три значения относим ко второму классу, что соответствует интервалу квантования $(0.16;\ \infty)$. В результате получаем слово

$$x = (0\,0\,1\,0\,1\,1).$$

1.9. Нахождение числа орбит

Каждая орбита однозначно определяется композицией $(m_1, \ldots, m_q)$, где $\sum m_i = n$. Таким образом, число различных орбит совпадает с числом всевозможных представлений числа n в виде суммы целых слагаемых (при этом допускается $m_i = 0$ для некоторых i). Приходим к классической комбинаторной задаче: сколькими способами можно разделить n одинаковых шаров (позиций слова) по q различным урнам (буквам)?

Пусть, например, $n = 4$, $q = 3$. Возможны следующие варианты:

4	0	0		2	2	0
0	4	0		2	0	2
0	0	4		2	2	0
3	1	0		2	1	1
3	0	1		1	2	1
1	3	0		1	1	2
0	3	1				
1	0	3				
0	1	3				

Упорядочим три буквы a, b, c следующим образом: $a = 1$, $b = 2$, $c = 3$. Тогда варианту распределения (4 0 0) соответствует слово $(a\,a\,a\,a) = (1\,1\,1\,1)$, варианту (0 4 0)—слово $(b\,b\,b\,b) = (2\,2\,2\,2)$, а варианту (2 1 1)—слово $(a\,a\,b\,c) = (1\,1\,2\,3)$. При этом слова (1 1 2 3) и (1 2 3 1) отождествляются, поскольку лежат в одной и той же орбите. Отсортируем каждое слово и прибавим (0 1 2 ...):

$$\begin{array}{c} (1\,1\,2\,3) \\ + \\ (0\,1\,2\,3) \end{array} \Rightarrow (1\,2\,4\,6).$$

Получается взаимно однозначное отображение всех n-слов q-ичного алфавита на множество всех n-слов в $(n + q - 1)$-ичном алфавите, причем в словах-образах нет повторяющихся символов. Отсюда получаем формулу для числа орбит:

$$N = \binom{n + q - 1}{n} = \binom{n + q - 1}{q - 1}.$$

В предыдущем примере имеем

$$N = \binom{4 + 3 - 1}{4} = \binom{6}{2} = 15.$$

Если рассматривать слова с точностью до переименования букв, то останутся лишь четыре варианта орбит:

$$(4),\ (3\,1),\ (2\,2),\ (2\,1\,1).$$

Всевозможные разбиения $\alpha, \beta, \ldots$ числа n принято упорядочивать лексикографически (как в словаре), так что

$$\alpha = (m_1 + m_2 + \ldots) > \beta = (n_1 + n_2 + \ldots)$$

тогда и только тогда, когда первая неисчезающая разность $m_i - n_i$ положительна. Например, при $n = 7$ имеем

$$(7) > (6, 1) > (5, 2) > (5, 1, 1) > (4, 2, 1) > (4, 1, 1, 1) >$$
$$> (3, 3, 1) > (3, 2, 2) > (3, 2, 1, 1) > (3, 1, 1, 1, 1) >$$
$$> (2, 2, 2, 1) > (2, 2, 1, 1, 1) > (2, 1, 1, 1, 1, 1) > (1, 1, 1, 1, 1; 1, 1).$$

Для числа таких разбиений не существует точной формулы.

Пусть слово x имеет композицию $(m_1, \ldots, m_q)$. Каждое каноническое подслово длины m_i слова x порождает условное разбиение, совпадающее с безусловным, поэтому на позициях, соответствующих этому подслову, число орбит равно

$$\binom{m_i + q - 1}{q - 1}.$$

Так как все подслова действуют независимо, то общее число условных орбит под действием слова x равно

$$N_x = \prod_{i=1}^{q} \binom{m_i + q - 1}{q - 1}.$$

Пример 1.5.

$x = (1\,0\,0\,1\,1), \quad Y = X = \{0, 1\}, \quad m_1 = 3, \; m_2 = 2.$

Число безусловных орбит равно

$$N = \binom{5 + 2 - 1}{1} = 6$$

(каждая орбита однозначно определяется числом единиц двоичного слова).

Число условных орбит при условии x равно

$$N_x = \binom{3 + 2 - 1}{1} \cdot \binom{2 + 2 - 1}{1} = 12.$$

Ниже показаны все безусловные орбиты и их расщепление на условные:

I	00000	IV 1)	10011
II 1)	00001	2)	11010
	10000		11001
	00010		01011
2)	01000		10110
	00100		10101
III 1)	10001		00111
	00011	3)	01101
2)	01100		01110
3)	11000	V 1)	11110
	01001		11101
	01010		01111
	10100	2)	11011
	00101		10111
	00110	VI	11111

Число N_x совпадает в данном случае с числом всевозможных матриц 2×2 вида

	1	0	
1	m_{11}	m_{12}	3
0	m_{21}	m_{22}	2

Здесь m_{ij}—целые числа, такие, что

$$m_{11} + m_{12} = 3, \quad m_{21} + m_{22} = 2.$$

1.10. Сжатие по Фитингофу

Перейдем к описанию эффективного метода кодирования слов. Рассмотрим сначала случай двоичного алфавита ($q = 2$). Каждая орбита здесь полностью определяется числом единиц (весом Хэмминга) слова:

$$d = 0, 1, \ldots, n.$$

Это число будем считать номером орбиты и записывать его в виде префикса постоянной длины $]\log(n+1)[$. Для эффективной нумерации всех слов одной и той же d-орбиты расположим их в лексикографическом порядке, считая, что $1 > 0$:

1 1 1 1 0 0 0
1 1 1 0 1 0 0
1 1 1 0 0 1 0
1 1 1 0 0 0 1
1 1 0 1 1 0 0
.....
0 0 0 1 1 1 1

Пусть в слове x единицы занимают места $n_1 < n_2 < \ldots < n_d$, считая справа. Например, для слова (1 1 0 1 1 0 0) $n_1 = 3$, $n_2 = 4$, $n_3 = 6$, $n_4 = 7$. Подсчитаем, сколько слов лежит ниже слова x. Во-первых, нужно учесть все слова, у которых левые отрезки совпадают вплоть до самой первой единицы (исключая ее). Тогда на n_1 месте в этих словах должен стоять 0, а последняя единица может занимать любые из $(n_1 - 1)$ мест. Таким образом, всего существует $\binom{n_1 - 1}{1}$ таких слов.

Учтем, далее, все слова, которые совпадают с x левее n_2-й позиции. Если у этих слов на месте n_2 также стоит 1, то приходим к предыдущему случаю. Если же на месте n_2 стоит 0, то остальные

две единицы могут занимать любое из $\binom{n_2 - 1}{2}$ возможных мест.

Продолжая эти рассуждения, приходим к выводу, что число слов, лежащих ниже слова x в лексикографическом упорядочении, равно

$$N^x = \binom{n_1 - 1}{1} + \binom{n_2 - 1}{2} + \ldots + \binom{n_d - 1}{d}.$$

Это число и будем считать номером слова в d-й орбите.

Пример 1.6. Для слова 1111000 $n_1 = 4$, $n_2 = 5$, $n_3 = 6$, $n_4 = 7$, поэтому его номер равен

$$\binom{3}{1} + \binom{4}{2} + \binom{5}{3} + \binom{6}{4} = 34.$$

Слово же 0001111 имеет 0-й номер, поскольку $n_1 = 1$, $n_2 = 2$, $n_3 = 3$, $n_4 = 4$. Номер слова 1101100 равен

$$\binom{2}{1} + \binom{3}{2} + \binom{5}{3} + \binom{6}{4} = 30.$$

Для записи номера слова d-й орбиты требуется $\left]\log \binom{n}{d}\right[$ двоичных символов. Таким образом, каждое кодовое слово состоит в данном случае из постоянной части (префикса длины $]\log (n+1)[$) и переменной части длины $\left]\log \binom{n}{d}\right[$.

Общий случай недвоичного алфавита сводится к двоичному с помощью последовательного отождествления букв. Пусть слово имеет композицию $(m_1, \ldots, m_q)$. Выделим букву a_1, а остальные буквы отождествим. Получим двоичное слово, в которое входит m_1 единиц. Это слово закодируем описанным выше способом, а затем применим те же рассуждения к каноническому подслову длины $n - m_1$, записанному в старом алфавите. В результате получим кодовое слово длины

$$l(x) =]\log (n+1)[+ \left]\log \binom{n}{m_1}\right[+$$

$$+]\log (n - m_1 + 1)[+ \left]\log \binom{n - m_1}{m_2}\right[+ \ldots$$

$$\ldots +]\log (n - m_1 - \ldots - m_{q-2} + 1)[+ \left]\log \binom{n - m_1 - \ldots - m_{q-2}}{m_{q-1}}\right[.$$

Величина

$$\binom{n}{m_1}\binom{n-m_1}{m_2}\cdots\binom{n-m_1-\ldots-m_{q-2}}{m_{q-1}} = \frac{n!}{m_1!m_2!\ldots m_q!} = 2^{I_0(x)}.$$

Кроме того,

$$(n+1)(n-m_1+1)\ldots(n-m_1-\ldots-m_{q-2}+1) \leq (n+1)^{q-1}.$$

Последняя оценка имеет простое объяснение: для того, чтобы указать номер орбиты, достаточно выделить $q-1$ упорядоченных ящиков, в каждый из которых записывать числа $m_1, \ldots, m_{q-1}$. Поскольку каждое из этих чисел может принимать значения от 0 до n, имеется ровно $(n+1)^{q-1}$ возможностей. Эта оценка является достаточно точной для числа орбит при больших n и фиксированном q. Описанный метод сжатия требует вычисления биномиальных коэффициентов $\binom{n}{j}$. Известное тождество

$$\binom{n}{j} = \binom{n-1}{j} + \binom{n-1}{j-1}$$

позволяет находить эти коэффициенты с помощью треугольника Паскаля:

	1				2	3	4	5	6	7	8
8	7				21	35	35	21	7	1	0
7	6				15	20	15	6	1	0	
6	5				10	10	5	1	0		
5	4				6	4	1	0			
4	3				3	1	0				
3	2	→	⊗	←	1	0					
2	1	→	⊗	←	0						
1	0										

Треугольник строится снизу вверх и слева направо сложением соседних элементов. На пересечении i-й строки и j-го столбца стоит элемент $\binom{i-1}{j}$. Допустим, требуется вычислить номер слова 1 1 0 1 0 0. Здесь $n_1 = 3$, $n_2 = 4$, $n_3 = 6$, $n_4 = 7$. Находим в первом столбце треугольника элемент 3-й строки: он равен 2, затем во втором столбце ищем элемент 4-й строки, он равен 3 и т.д. Складывая эти числа, получаем искомый номер слова. Следовательно, если хранить треугольник в памяти компьютера, то процесс сжатия происходит практически мгновенно. Конечно, при больших n приходится для построения треугольника Паскаля работать с числами большой разрядности, реализуя соответствующую арифметику программным путем.

Рассмотрим теперь, как происходит процесс декодирования. Требуется по номеру

$$N^x = \binom{n_1 - 1}{1} + \binom{n_2 - 1}{2} + \ldots + \binom{n_d - 1}{d}$$

восстановить числа $n_1, n_2, \ldots, n_q$. Для этого воспользуемся следующим свойством треугольника Паскаля:

$$\binom{n+1}{d} = \binom{n}{d} + \binom{n-1}{d-1} + \ldots + \binom{n-d+1}{1} + 1.$$

Это простое следствие основного тождества для биномиальных коэффициентов можно интерпретировать так: если просуммировать d элементов главной диагонали, отличной от нулевой, слева направо, то получится число, на единицу меньше того числа, которое стоит выше последнего диагонального элемента в d-м столбце. Например, $1 + 1 + 1 + 1 = 5 - 1$, $2 + 3 + 4 + 5 = 15 - 1$, $3 + 6 + 10 + 15 = 35 - 1$.

Получаем следующий способ декодирования. Пусть $N^x = 30$ (предполагается, что n и d фиксированы, $n = 7$, $d = 4$). Находим в 4-м столбце наибольший элемент, не превосходящий 30. Им оказывается число 15 в 7-й строке. Это число обязательно входит в разложение N^x. Действительно, если бы нужным элементом оказалось бы число 5 из 6-й строки, то, поскольку $n_1 < n_2 < \ldots < n_d$, N^x не превосходило бы суммы чисел диагонали, содержащей число 5, т. е. величины $5 + 4 + 3 + 2$, которая меньше 15. Таким образом заключаем, что $n_4 = 7$. Вычитаем 15 из 30 и находим в 3-м столбце максимальный элемент, не превосходящий 15. Им оказывается число 10 в 6-й строке. Заключаем, что $n_3 = 6$. Точно так же находим номера $n_2 = 4$ и $n_1 = 3$.

Подытожим сказанное

Т е о р е м а 1.1. *Существует префиксный код полиномиальной трудоемкости, у которого длина кодового слова асимптотически равна его* 0*-информации*:

$$l(x) = I_0(x) + O(n), \qquad \text{где } |O(n)| \leq (q-1)\log(n+1) + 2(q-1).$$

1.11. Независимость

Величины $I_0(y/x)$ и $I_0(x{:}y)$ характеризуют степень зависимости слов. Один предельный случай, когда $I_0(y/x) = 0$, был уже рассмотрен в этой главе. Он соответствует самой сильной зависимости—функциональной зависимости $x \to y$. В этом случае взаимная информация $I_0(x{:}y)$ принимает максимальное значение,

равное $I_0(y)$. Для эквивалентных слов расстояние $\rho(x, y) = 0$, а $I_0(x{:}y) = I_0(x) = I_0(y)$.

Рассмотрим теперь другой крайний случай, когда $I_0(x{:}y) = 0$.

Предложение 1.9. *Следующие определения эквивалентны:*

1) *Слова x и y независимы*; 2) $I_0(x{:}y) = 0$,

3) $I_0(x/y) = I_0(x)$, 4) $I_0(y/x) = I_0(y)$.

Таким образом, если x и y независимы, то в слове x не содержится никакой информации относительно слова y.

Примером независимых слов являются слова

$$x = (1\,1\,1\,1), \quad y = (a\,b\,b\,a).$$

Информацию слова мы будем вычислять с точностью до $O(n)$, так что независимыми оказываются также слова

$$x = (1\,1\,1\,1\,0\,0\,0\,0), \quad y = (0\,1\,0\,1\,0\,1\,0\,1).$$

По определению $I_0(y/x) = \log [G_x{:}(G_x \cap G_y)]$. Когда слово x фиксировано, то $I_0(y/x)$ принимает наибольшее значение, если $G_x \cap G_y = e$. При этом

$$I_0(y/x) = \log |G_x|, \quad I_0(x/y) = \log |G_y|, \quad I_0(x \otimes y) = \log |G|.$$

Такую пару слов будем называть *сильно независимой.* Ясно, что в этом случае переходная матрица состоит из единиц и нулей.

1.12. Канал с шумом. Теоремы кодирования Шеннона

Когда говорят о канале с шумом, то имеют в виду пары слов x и y, $x \in X^n$, $y \in Y^n$. При этом X называют *входным алфавитом*, а Y*—алфавитом на выходе канала.*

Канал связи преобразует последовательности символов на входе в выходные последовательности. Преобразование слова x в слово y управляется переходной матрицей

		a	b	c	...	
	a	m_{11}	m_{12}	...		m_1
	b	m_{21}	m_{22}	...		m_2
	c	...	...	...		...
$A =$	...					...
	...					...
	...					...
	...	n_1	n_2	...		

каждый элемент которой указывает на число переходов i-й буквы слова x в j-ю букву слова y.

Матрица A однозначно определяет композицию $(m_1, \ldots, m_q)$ входного слова x_0 и композицию $(n_1, \ldots, n_k)$ выходного слова y_0. Тем самым определена некоторая орбита U_n на входе (0-*источник на входе*) и орбита U_n' на выходе (0-*источник на выходе*).

Кроме того, определена условная информация $I_0(y_0/x_0) = d$, которая характеризует уровень помех в канале, и взаимная информация $C = I_0(x_0{:}y_0)$, которую назовем *пропускной способностью канала.*

Пару $x \otimes y$, $x \in U_n$, $y \in U_n'$, назовем *допустимой*, если переходная матрица для этих слов совпадает с матрицей A. В этом случае $I_0(y/x) = d$. Для фиксированного слова x все допустимые слова y (назовем это множество *шаром радиуса d*) составляют условную орбиту $G_x y = V_d(x)$. Действительно, если слова $x \otimes y$ и $x \otimes u$ допустимы, то эти слова имеют одну и ту же композицию, так что $x \otimes y = \sigma(x \otimes u)$. Но $\sigma x = x$, поэтому $\sigma \in G_x$.

Таким образом, слово x воспринимается на выходе как некоторое слово шара $V_d(x)$.

Описанная математическая модель (назовем ее *алгебраическим каналом*) является идеализацией реальных каналов связи. В дальнейшем мы обобщим эту модель, приближая ее к реальности, но основные идеи и методы помехоустойчивого кодирования удобно анализировать на этой модели, поскольку она допускает алгебраизацию с помощью групп подстановок.

Под *кодом* будем понимать любое подмножество $D \subseteq U_n$, $D = \{x_1, \ldots, x_N\}$. Предполагается, что по каналу передаются лишь кодовые слова. Величину N назовем *мощностью кода.*

Если шары $V_d(x_i)$ и $V_d(x_j)$ с центрами в кодовых словах пересекаются, то возникает неоднозначность декодирования. Выберем для каждого x_i множество $B_i \subseteq V_d(x_i)$ так, чтобы все B_i не пересекались. Систему $\{B_i\}$, $i = 1, \ldots, N$, назовем *решающей схемой для кода.*

Каждое слово $y \in B_i$ декодируется в кодовое слово x_i. Ясно, что величина $e_i = 1 - |B_i|/|V_d|$ характеризует ошибку декодирования, если было передано слово x_i. Точнее говоря, когда $y \in V_d - B_i$, наступает отказ от декодирования, который мы будем трактовать как ошибку декодирования.

Величину $e = \operatorname{Max} e_i$ назовем *максимальной ошибкой решающей схемы*, а величину

$$\bar{e} = \frac{1}{N} \sum_{i=1}^{N} e_i$$

средней ошибкой решающей схемы.

Введем нормированную меру $\mu(x, y)$ на $U_n \times U_n'$ следующим образом:

$$\mu(x) = \frac{1}{|U_n|}, \quad \text{для всех} \quad x \in U_n,$$

$$\mu(y/x) = \begin{cases} \dfrac{1}{|V_d|}, & \text{если } y \in V_d, \\ 0, & \text{если } y \notin V_d. \end{cases}$$

Ясно, что для любого множества $M \subseteq U_n'$

$$\mu(M/x) = \sum_{y \in M} \mu(y/x) = |M \cap V_d(x)| / |V_d|.$$

В частности, $e_i = 1 - \mu(B_i/x_i)$. Далее, если $y \in V_d(x)$, то $I_0(y/x) = d$. Поскольку

$$I_0(x) + I_0(y/x) = I_0(y) + I_0(x/y),$$

то

$$I_0(x/y) = d' = I_0 - I_0' + d.$$

т. е. $x \in V_{d'}(y)$.

Другими словами, с каждым каналом связан двойственный канал, матрица которого получается транспонированием матрицы A. Поэтому $y \in V_d(x)$ равносильно $x \in V_{d'}(y)$ для двойственного канала. Поскольку $|U_n||V_d| = |U_n'||V_{d'}|$, имеем

$$\mu(y) = \sum_{x \in U_n} \mu(x)\mu(y/x) = \frac{|V_{d'}|}{|U_n||V_d|} = \frac{1}{|U_n'|}.$$

Наконец,

$$\mu(V_d(x)) = \frac{|V_d|}{|U_n'|}.$$

Теорема 1.2. *Существует код мощности N и решающая схема с максимальной ошибкой e, для которых справедливо неравенство*

$$N > 2^c \cdot e.$$

Доказательство. Выберем $x_1 \in U_n$ в качестве первого кодового слова и множество $V_d(x_1)$ в качестве B_1. Если для некоторой точки x_2 выполняется неравенство

$$|V_d(x_1) \cap V_d(x_2)| \leq e|V_d|,$$

то x_2 включим в код и положим

$$B_2 = V_d(x_2) - V_d(x_2) \cap V_d(x_1).$$

На N-м шаге включаем точку x_N в код, если

$$\left| V_d(x_N) \cap \left(\bigcup_{i-1}^{N-1} V_d(x_i) \right) \right| \leq e|V_d|,$$

при этом полагаем

$$B_N = V_d(x_N) - V_d(x_N) \cap \left(\bigcup_{i=1}^{N-1} V_d(x_i) \right).$$

Ясно, что при таком расширении кода будем всегда получать $e_i \leq e$.

Процесс расширения заканчивается, как только $\mu\left(\bigcup_{i=1}^{N} V_d(x_i) \right)$ становится больше e.

Действительно, пусть $M = U_n' - \bigcup_{i=1}^{N} V_d(x_i)$. Если $\mu(M) \geq 1 - e$, то, поскольку

$$\mu(M) = \sum_{x \in U_n} \mu(x)\mu(M/x) = \sum \frac{|M \cap V_d(x)|}{|V_d||U_n|},$$

найдется такая точка, что

$$\frac{|M \cap V_d(x)|}{|V_d|} \geq 1 - e$$

и эту точку можно было бы включить в код.

Таким образом, для максимального кода выполняется неравенство

$$e < \mu\left(\bigcup_{i=1}^{N} V_d(x_i) \right) \leq N\mu(V_d(x_i)) = \frac{N|V_d|}{|U_n'|}.$$

Вспоминая, что $|V_d| = 2^{I_0(y_0/x_0)}$, а $|U_n'| = 2^{I_0(y_0)}$, получаем

$$e < N/2^{I_0(x_0:y_0)} = N/2^c.$$

Теорема 1.3. *Для любого кода и любой решающей схемы выполняется неравенство*

$$N \leq 2^c/(1-\bar{e}) \leq 2^c/(1-e).$$

Доказательство. Заметим прежде всего, что

$$1 - \bar{e} = \frac{1}{|V_d|N} \sum_{i=1}^{N} |B_i|.$$

Поскольку все B_i не пересекаются, то

$$2^{I_0'} = |U_n'| \geq |\cup B_i| = \sum_{i=1}^{N} |B_i| = N|V_d|(1-\bar{e}).$$

Остается заметить, что $\bar{e} \leq e$.

Построенная в процессе доказательства теоремы 1.2 решающая схема имеет достаточно сложную структуру, так что приходится хранить в памяти все множества $\{B_i\}$. В то же время для любого кода существует *каноническая решающая схема*

$$\left\{B_i = V_d(x_i) - V_d(x_i) \cap \left(\bigcup_{j \neq i} V_d(x_j)\right)\right\},$$

имеющая следующее простое описание в виде алгоритма декодирования: слово y на выходе декодируется в кодовое слово x_i, если $I_0(y/x_i) \leq d$. Если существует несколько кодовых слов, удовлетворяющих этому условию, то происходит ошибка декодирования. Удобно исключить параметр d из этого алгоритма, применяя *каноническое декодирование*: слово y декодируется в кодовое слово x_i, для которого $I_0(y/x_i)$ минимально. Поскольку

$$I_0(x_i/y) = I_0 - I_0' + I_0(y/x_i),$$

то каноническое декодирование равносильно декодированию по минимуму $I_0(x_i/y)$.

Разницу между решающей схемой теоремы 1.2 и канонической решающей схемой показывает рис. 1:

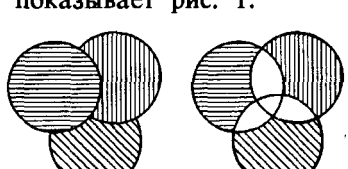


Рис. 1

Теорема 1.4. *Существует код, для которого мощность и ошибка канонического декодирования связаны неравенством*

$$N > 2^c \bar{e}.$$

Доказательство. Рассмотрим ансамбль $\{D_1, D_2, \ldots, D_M\}$ всех кодов мощности N. Ясно, что этот ансамбль состоит из

$$M = \binom{|U_n|}{N}$$

элементов. Обозначим через $\bar{e}_{\text{ср}}$ ошибку, осредненную по всему ансамблю. Таким образом, если $\bar{e}^j$—средняя ошибка канонической решающей схемы для j-го кода, то

$$\bar{e}_{\text{ср}} = \frac{1}{M} \sum_{i=1}^{M} \bar{e}^j.$$

Каждое допустимое слово $x \otimes y$, $x \in U_n$, $y \in V_d(x)$, назовем допустимым ребром (рис. 2):

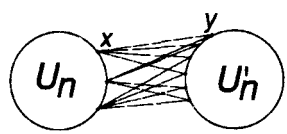


Рис. 2

Ясно, что всего существует $|U_N| \cdot |V_d| = |U'_n| \cdot |V_{d'}| = 2^{I_0(x \otimes y)}$ таких ребер. Пусть $\{B_i^j\}$, $i = 1, \ldots, N$,—каноническая решающая схема для j-го кода ансамбля. Допустимое ребро (x, y) назовем регулярным для кода D_j, если $x = x_i \in D_j$, $y \in B_i^j$. Из определения $\bar{e}^j$ следует, что

$$1 - \bar{e}^j = \frac{1}{N} \sum \frac{|B_i^j|}{|V_d|}, \quad 1 - \bar{e}_{\text{ср}} = \frac{1}{MN|V_d|} \sum_{j=1}^{M} \sum_{i=1}^{N} |B_i^j|.$$

Величина

$$k_j = \sum_{i=1}^{N} |B_i^j|$$

равна числу регулярных ребер для j-го кода, а

$$\sum_{j=1}^{M} k_j = k$$

совпадает с общим числом ребер, регулярных для кодов ансамбля. Если k_j трудно подсчитать—эта величина различна для разных кодов, то k уже можно легко вычислить.

Действительно, пусть m—число кодов ансамбля, для которых оказывается регулярным фиксированное допустимое ребро (x, y). Ребро (x, y) регулярно для некоторого кода, если шар $V_{d'}(y)$ не содержит никакой кодовой точки, кроме x (по определению канонической решающей схемы). Поэтому

$$m = \binom{|U_n| - \|V_{d'}\|}{N-1}.$$

Ясно, что

$$k = |U_n||V_d|m.$$

Имеем

$$1 - \bar{e}_{\text{ср}} = \frac{1}{MN|V_d|}|U_n||V_d|m = \frac{|U_n|}{N}\frac{m}{M} =$$

$$= \frac{|U_n|(|U_n| - |V_{d'}|)(|U_n| - |V_{d'}| - 1)\ldots(|U_n| - |V_{d'}| - N + 2)N!}{N|U_n|(|U_n| - 1)\ldots(|U_n| - N + 1)(N-1)!} =$$

$$= \left(1 - \frac{|V_{d'}| - 1}{|U_n| - 1}\right)\left(1 - \frac{|V_{d'}| - 1}{|U_n| - 2}\right)\cdots\left(1 - \frac{|V_{d'}| - 1}{|U_n| - (N-1)}\right) \geq$$

$$\geq \left(1 - \frac{|V_{d'}| - 1}{|U_n| - (N-1)}\right)^{N-1} \geq 1 - (N-1)\frac{|V_{d'}| - 1}{|U_n| - (N-1)}.$$

Таким образом,

$$\bar{e}_{\text{ср}} \leq (N-1)\frac{|V_{d'}| - 1}{|U_n| - (N-1)},$$

и найдется код из ансамбля, для которого

$$\bar{e} \leq (N-1)\frac{|V_{d'}| - 1}{|U_n| - (N-1)},$$

что равносильно неравенству

$$N \geq 1 + \bar{e}\frac{|U_n|}{|V_{d'}| - (1 - \bar{e})} > \bar{e}\frac{|U_n|}{|V_{d'}|} = \bar{e}\frac{|U'_n|}{|V_d|}.$$

В доказательстве использовано неравенство

$$(1 - x)^n \geq 1 - nx.$$

Приведем для полноты вывод этого неравенства.

Л е м м а. *Оценка* $(1 - x)^n \geq 1 - nx$ *справедлива при четном целом положительном* n *для всех вещественных* x, *а при нечетном* n—*для всех* $x \leq 2$.

Доказательство. Рассмотрим функцию

$$f(x) = (1 - x)^n - (1 - nx).$$

Вычислим производные:

$$f'(x) = n[1 - (1 - x)^{n-1}], \quad f''(x) = n(n - 1)(1 - x)^{n-2}.$$

Первая производная обращается в 0 в единственной точке $x = 0$, если n четно. При этом $f''(0) > 0$, $f(0) = 0$. Таким образом, в точке $x = 0$ достигается минимум $f(x)$ и $f(x) \geq 0$ для всех x.

Если же n нечетно, то появляется еще одна стационарная точка $x = 2$. При этом $f''(2) < 0$, $f(2) = 2(n - 1) \geq 0$. Следовательно, этой точке соответствует максимум и отрицательные значения $f(x)$ может принимать лишь при $x > 2$.

При доказательстве теоремы 1.4 мы всегда можем предполагать, что $N - 1$ четно, так как все оценки рассматриваются с точностью до округления.

1.13. Асимптотическое поведение информации. Энтропия

При больших длинах n вычисление 0-информации слова становится громоздким, поэтому интерес представляют приближенные асимптотические выражения. Приближения основаны на известной формуле Стирлинга для факториалов.

Лемма (формула Стирлинга)

$$n! \equiv \sqrt{2\pi n}\, n^n e^{-n}, \quad \sqrt{2\pi n}\, n^n e^{-n} \exp \frac{1}{12n + 1} < n! < \sqrt{2\pi n}\, n^n \exp \frac{1}{12n}.$$

Для произвольного слова x длины n с композицией $(m_1, \ldots, m_k)$ определим *энтропию слова* $H(x)$ посредством формулы

$$H(x) = -\sum_i \frac{m_i}{n} \log \frac{m_i}{n},$$

считая, что $0 \cdot \log 0 = 0$.

Применяя формулу Стирлинга к выражению для $I_0(x)$, получаем

Предложение 1.10. *Для любого слова x длины n в алфавите из q букв справедливо*

$$I_0(x) = n(H(x) - O(1)),$$

где, как обычно, $O(1) \to 0$ при $n \to \infty$, причем

$$O(1) \approx \frac{q - 1}{2} \frac{\log n}{n}.$$

Всюду в дальнейшем для вычисления $I_0(x)$ мы будем считать, что

$$I_0(x) = nH(x),$$

даже при небольших длинах n. Поскольку

$$H(x) = \lim_{n \to \infty} \frac{1}{n} I_0(x),$$

энтропия обладает всеми доказанными ранее свойствами 0-информации, которые, в свою очередь, являются следствиями элементарной теории групп.

Асимптотическое выражение для условной информации получается следующим образом. С каждой парой слов x и y свяжем, как и ранее, матрицу (m_{ij}) (число переходов i-й буквы слова x в j-ю букву слова y):

	a	b	...	...	Сумма в строке
a	m_{11}	m_{12}	...	m_{1k}	m_1
b	m_{21}	m_{22}	...	m_{2k}	m_2
...			...		...
...			...		...
...	m_{r1}	m_{r2}	...	m_{rk}	m_r
Сумма в столбце	n_1	n_2	...	n_k	n

Здесь $(m_1, \ldots, m_r)$—композиция слова x, $(n_1, \ldots, n_k)$—композиция слова y и

$$\sum_i m_i = \sum_j n_j = n.$$

Пусть X_i—i-я строка матрицы. Тогда

$$H_i = H(X_i) = H(m_{i1}/m_i, m_{i2}/m_i, \ldots, m_{ik}/m_i).$$

Обозначим

$$H(y/x) = \frac{m_1}{n} H_1 + \frac{m_2}{n} H_2 + \ldots + \frac{m_r}{n} H_r.$$

Предложение 1.11.

$$I_0(y/x) = n(H(y/x) - O(1)), \quad \text{где } O(1) = \frac{(q-1)q}{2} \frac{\log n}{n}.$$

Предложение 1.12. *$H(x)$ минимальна и равна нулю при $x = (a, a, \ldots, a)$. Максимум достигается, когда $m_1/n = \ldots = m_r/n = 1/r$.* При этом $H(x) = \log r$.

Доказательство. Рассмотрим логарифмическую функцию $\ln x$. Поскольку $(\ln x)' = 1/x$, то справедливо неравенство

$\ln x \leq x - 1$, которое означает, что график функции $\ln x$ лежит ниже прямой $y = x - 1$, являющейся касательной к этой функции в точке $x = 1$. Равенство достигается лишь в точке $x = 1$. Далее,

$$U(x) = H\left(\frac{m_1}{n}, \ldots, \frac{m_r}{n}\right) - \log r =$$

$$= \sum_i \frac{m_i}{n} \log \frac{n}{m_i} - \sum_i \frac{m_i}{n} \log r = \sum_i \frac{m_i}{n} \log \frac{n}{m_i r}.$$

Поскольку

$$\log \frac{n}{m_i r} = \frac{1}{\ln 2} \ln \frac{n}{m_i r} \leq \frac{1}{\ln 2} \left(\frac{n}{m_i r} - 1\right),$$

получаем

$$U(x) \leq \frac{1}{\ln 2} (1 - 1) = 0.$$

Здесь равенство имеет место только лишь при $n/(m_i r) = 1$.

Пример вычисления:

$$x = (bbcc), \quad y = (abab), \quad I_0(y) = 4H(1/2) = 4.$$

	a	b	
b	1	1	$2H(1/2) = 2$
c	1	1	$2H(1/2) = 2$
	2	2	4

$$I_0(y/x) = 4, \quad I_0(x{:}y) = 0.$$

Следовательно, слова x и y независимы.

Теоремы кодирования в асимптотической форме принимают следующий вид:

Т е о р е м а 1.5. *Если* $\log N < C$, *то найдется код мощности* N, *для которого* $e \to 0$ *с ростом* n. *Если* $\log N > C$, *то для любого кода мощности* N $e \to 1$ *с ростом* n.

1.14. Прямое произведение слов

Введем в заключение одну важную операцию над словами, которая позволяет «соединить» два слова произвольной длины. Пусть

$$x = (a_1, a_2, \ldots, a_m), \quad y = (b_1, b_2, \ldots, b_n).$$

Прямым произведением $x \times y$ *слов* x, y *назовем слово*

$$x \times y = ((a_1, b_1), (a_1, b_2), \ldots, (a_1, b_n), (a_2, b_1), \ldots, (a_m, b_n)).$$

Полученное слово имеет длину mn в алфавите, который является декартовым произведением алфавитов. Удобно представлять слово $x \times y$ в виде двух столбцов:

x'	y'
a_1	b_1
a_1	b_2
...	...
a_1	b_n
a_2	b_1
...	...
a_2	b_n
...	...
a_m	b_n

Если слово x имеет композицию $(m_1, \ldots, m_r)$, а слово y—композицию $(n_1, \ldots, n_k)$, то слово x' имеет композицию $(m_1 n, \ldots, m_r n)$, а слово y'—композицию $(n_1 m, \ldots, n_k m)$. Поэтому

$$H(x') = H(m_1 n/mn, \ldots, m_r n/mn) = H(m_1/m, \ldots, m_r/m) = H(x),$$

$$H(y') = H(y).$$

Очевидно,

$$x \times y = x' \otimes y',$$

$$H(x \times y) = H\left(\frac{m_1 n_1}{mn}, \frac{m_1 n_2}{mn}, \ldots, \frac{m_r n_k}{mn}\right) = -\sum_{i,j} \frac{m_i n_j}{mn} \log \frac{m_i n_j}{mn} =$$

$$= -\sum_{i,j} \frac{m_i}{m} \frac{n_j}{n} \log \frac{m_i}{m} - \sum_{i,j} \frac{m_i}{m} \frac{n_j}{n} \log \frac{n_j}{n} =$$

$$= \sum_j \frac{n_j}{n} H(x) + \sum_i \frac{m_i}{m} H(y) = H(x) + H(y).$$

Таким образом, доказано следующее утверждение:

Т е о р е м а 1.6. *Прямое произведение слов $x \times y$, где m—длина слова x, а n—длина слова y, может быть представлено в виде*

$$x \times y = x' \otimes y',$$

где x' и y'—взаимно независимые слова длины mn, причем

$$H(x') = H(x), \quad H(y') = H(y), \quad H(x \times y) = H(x) + H(y).$$

2. РАСПОЗНАВАНИЕ ОБРАЗОВ

2.1. Постановка задачи распознавания. Информационная матрица

Задача распознавания образов была впервые сформулирована Ф. Розенблаттом [14] при попытке моделирования операций, выполняемых живыми организмами в процессе коммуникации и восприятия окружающего мира. Им же была построена обучающая машина, известная под названием «Персептрон», позволившая решить задачу автоматической классификации.

Каждый объект описывается набором признаков $X1$, $X2$, ..., Xn. Различают качественные признаки, когда $X = \{1, 2, \ldots, m\}$, и количественные, когда X—любое действительное число. В последнем случае признак с помощью квантования можно превратить в качественный, т. е. представить в виде конечной последовательности целых чисел (слова). (Эта процедура описана в пункте 1.8.) Кроме того, каждый контрольный объект должен быть отнесен к тому или иному классу из некоторого множества $Y = \{1, 2, \ldots, k\}$.

Информация о признаках представляется в виде таблицы (*информационной матрицы*), которая используется для обучения.

Пример 2.1.

Номер объекта	Признаки								Класс
	$X1$	$X2$	$X3$	$X4$	$X5$	$X6$	$X7$	$X8$	Y
1	2	2	2	1	2	2	1	1	1
2	2	2	2	2	2	2	1	1	1
3	2	2	2	1	1	2	1	1	1
4	2	1	1	2	1	1	1	2	2
5	2	1	1	2	1	1	2	1	2
6	1	1	1	2	1	1	1	2	2
7	2	1	1	1	2	2	1	2	3
8	2	1	1	1	2	2	2	2	3
9	2	2	1	1	2	2	1	2	3

В обучающую выборку, как правило, включается только часть контрольных объектов, остальные используются для настройки (т. е. для достижения наиболее четкого распознавания). Например, из 60 объектов 40 можно использовать для обучения

и 20 для контроля при настройке, причем последние 20 объектов должны равномерно распределяться между классами.

Пример 2.2 (прогнозирование запаса руды для месторождения [17]).

Значения признаков:

Y—запас руды ($Y = 1 \Leftrightarrow Y > 1$ млн тонн, $Y = 0 \Leftrightarrow Y < 1$ млн тонн),

$X1$—приуроченность к горизонтам слюдистых сланцев,

$X2$—приуроченность к горизонтам амфиболитов,

$X3$—близость к контакту со слюдистыми сланцами,

$X4$—близость к контакту с амфиболитами,

$X5$—присутствие тел амфиболитов,

$X6$—обилие даек,

$X7$—пиритизация,

$X8$—пропилитизация,

$X9$—ожелезнение,

$X10$—аргиллитизация,

$X11$—наличие вторичных ореолов свинца и цинка,

$X12$—развитие складок.

Информационная матрица

	$X1$	$X2$	$X3$	$X4$	$X5$	$X6$	$X7$	$X8$	$X9$	$X10$	$X11$	$X12$	Y
1	0	1	0	0	0	1	1	1	0	0	1	1	1
2	0	1	1	1	0	0	1	0	1	0	1	1	1
3	0	1	0	1	1	1	0	1	0	0	0	0	1
4	0	0	1	0	1	0	1	0	1	1	1	1	1
5	1	1	0	1	0	1	1	0	1	0	1	0	1
6	1	0	1	0	1	1	0	1	0	1	0	1	0
7	1	0	1	0	1	1	1	0	0	1	0	0	0
8	1	0	0	1	1	1	0	1	0	0	0	0	0
9	1	0	1	1	1	1	0	0	1	0	0	1	0
10	0	1	1	0	0	0	0	1	0	1	0	1	0

Контроль

11	0	0	1	1	1	1	0	0	0	0	0	1	?
12	0	0	0	1	0	1	0	0	1	0	1	1	?

Таким образом, мы собираем информацию о разведанных месторождениях, обучаемся на основе этой информации, а затем пытаемся прогнозировать запас руды для неизвестного (контрольного) месторождения, опираясь на косвенные признаки $X1 \leftrightarrow X12$, которые не требуют глубокой разведки.

Пример 2.3 (управление технологическим процессом). Рассмотрим производство карбида кальция на мощной электрической печи. Конечный продукт можно отнести к одному из трех сортов. С помощью математического моделирования очень трудно

установить, как влияет на сортность карбида характер технологического режима. Поэтому мы отказываемся от моделирования и применяем метод распознавания образов: собираем информацию о технологическом режиме и полученной при этом сортности карбида за несколько рабочих смен и составляем информационную матрицу для обучения. При этом набор признаков, предположительно оказывающих влияние на качество карбида, формируется на основе консультаций со специалистами (экспертами). Например, можно выбрать такие признаки:

$X1$—содержание влаги в коксе,
$X2$—гранулометрический состав кокса,
$X3$—гранулометрический состав извести,
$X4$—количество шихтовой извести,
$X5$—сила тока,
$X6$—напряжение трансформатора,
$X7$—общая активная мощность печи.

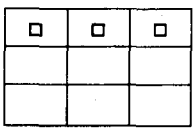
В информационную матрицу следует включать как можно больше признаков. В дальнейшем останутся лишь информативные признаки, действительно влияющие на качество конечного продукта.

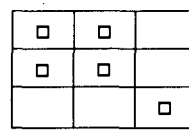
Пример 2.4 (распознавание изображений). Предположим, что экран разделен на квадратные клетки, каждой из которых соответствует свой светочувствительный элемент:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Тогда любое изображение можно превратить в двоичное слово, считая i-м признаком:

$$X_i = \begin{cases} 1, & \text{если изображение пересекает } i\text{-ю клетку,} \\ 0, & \text{если не пересекает,} \end{cases}$$


⇒ (111000000),


⇒ (110110001).

Можно поставить задачу отнесения изображения к одному из классов, основываясь на предварительном обучении.

2.2. Вычисление информативности признака

Каждый признак X является столбцом информационной матрицы, т. е. является словом в конечном алфавите. Например,

$$X1 = (2, 2, 2, 2, 2, 1, 2, 2, 2).$$

Мы хотим по этому признаку определить значение основного признака Y, т. е. узнать, к какому классу принадлежит данный объект. Таким образом, *информативность признака X равна количеству информации, содержащейся в этом признаке относительно основного признака Y*:

$$I_0(X:Y) = I_0(Y) - I_0(Y/X) = I_0(X) - I_0(X/Y).$$

Например, если $Y = (1, 1, 1, 2, 2, 2, 3, 3, 3)$, то $I_0(X1:Y) = 1.77$ бита.

Для вычисления информативности составляется матрица перехода букв слова X в буквы слова Y:

	1	2	3	
1	0	1	0	1
2	3	2	3	8
	3	3	3	9

Для матрицы из примера 2.1 получаем следующую информативность признаков:

$X1$	$X2$	$X3$	$X4$	$X5$	$X6$	$X7$	$X8$
1.77	6.16	8.26	6.16	6.16	8.26	1.37	6.16

Признак X, для которого $I_0(X:Y) = I_0(Y)$, назовем *чистым*. Для такого признака $I_0(Y/X) = 0$, т. е. признак полностью раскрывает неопределенность. В этом случае существует функциональная зависимость $X \to Y$. Если же $I_0(X:Y) = 0$, то признак называется *шумом*. Такой признак не дает никакой информации относительно принадлежности объекта к какому-либо классу, поскольку он независим от Y.

Введем *первый параметр настройки* α, определяющий минимально допустимый уровень информативности признака, и отбросим все признаки, информативность которых меньше α. Параметр α обычно задается в процентах от максимально возможной информативности $I(Y)$. Если в примере 2.1 выбрать $\alpha = 24\ \%$, то, поскольку $I(Y) = 14.26$, получаем $\alpha = 3.42$. Таким образом, считаем шумовыми признаки $X1$ и $X7$.

В примере 2.2 информативность признаков равна

$X1$	$X2$	$X3$	$X4$	$X5$	$X6$	$X7$	$X8$	$X9$	$X10$	$X11$	$X12$
2.8	2.8	1.24	0.3	1.24	0.3	2.8	0.3	1.24	1.24	6.1	0

(здесь $I(Y) = 10$ бит). Если выбрать $\alpha = 25\ \%$, т. е. $\alpha = 2.5$, то для дальнейшего анализа остаются четыре признака:

$X11$	$X1$	$X2$	$X7$
6.1	2.8	2.8	2.8

2.3. Кластерный анализ в пространстве признаков

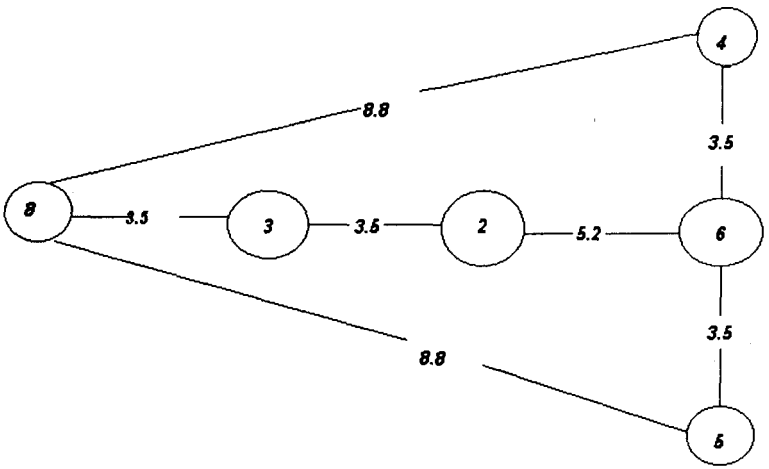
Некоторые признаки могут повторять друг друга. Сходные признаки образуют подмножество в пространстве признаков, называемое *кластером*. Для анализа зависимости признаков естественно воспользоваться информационной метрикой

$$\rho(X_i, X_j) = \frac{1}{2}\,(I_0(X_j/X_i) + I_0(X_i/X_j)).$$

Если расстояние $\rho(X_i, X_j)$ мало, то X_i и X_j зависимы (повторяют друг друга). Для независимых признаков $\rho(X_i, X_j)$ приближается к максимуму.

В примере 2.1 имеем

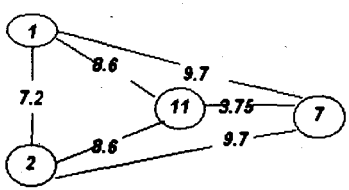
	*X*2	*X*3	*X*4	*X*5	*X*6	*X*8
*X*2	0	3.5	8.4	8.4	5.2	6.8
*X*3		0	8.4	8.4	6	3.5
*X*4			0	6.8	3.5	8.8
*X*5				0	3.5	8.8
*X*6					0	8.4
*X*8						0



Таким образом, существуют два кластера: (*X*8, *X*3, *X*2) и (*X*4, *X*6, *X*5).

Для примера 2.2 имеем соответственно

	*X*11	*X*1	*X*2	*X*7
*X*11	0	8.6	8.6	3.75
*X*1		0	7.2	9.7
*X*2			0	9.7
*X*7				0



Построенные графы дают лишь предварительную информацию о взаимной зависимости признаков. Метрика $\rho(X_i, X_j)$ показывает, насколько далеки признаки X_i и X_j от взаимной эквивалентности. В предельном случае, когда $\rho(X_i, X_j) = 0$, существует функциональная зависимость в обе стороны:

$$X_i \to X_j, \qquad X_j \to X_i.$$

Например, можно считать, что признаки

X_i	X_j
1	2
1	2
1	2
1	2
0	1
0	1

повторяют друг друга. Более глубокий кластерный анализ связан с понятием условной независимости.

Скажем, *что признак (слово) Z дублирует признак X относительно Y, если*

$$I_0(Y: (X \otimes Z)) = I_0(Y:X).$$

Другими словами, если известен признак X, то добавление признака Z не приносит новой информации относительно признака Y. Очевидно, это определение равносильно равенству

$$I_0(Y/X \otimes Z) = I_0(Y/X).$$

Естественно назвать при этом признаки Y и Z *условно независимыми при условии X.*

Это отношение симметрично, т. е. Y и Z можно поменять местами:

$$I_0(Z/X \otimes Y) = I_0(Z/X).$$

Доказательство тривиально:

$$I_0(X \otimes Z) + I_0(Y/X \otimes Z) = I_0(X \otimes Z) + I_0(Y/X),$$

$$I_0(X \otimes Z \otimes Y) = I_0(X) + I_0(Z/X) + I_0(Y/X),$$

$$I_0(X \otimes Z \otimes Y) = I_0(X \otimes Y) + I_0(Z/X),$$

$$I_0(X \otimes Y) + I_0(Z/X \otimes Y) = I_0(X \otimes Y) + I_0(Z/X),$$

$$I_0(Z/X \otimes Y) = I_0(Z/X).$$

Отношение «Z дублирует признак X относительно Y» мы будем обозначать двойной стрелкой:

$$X \underset{Y}{\to\to} Z.$$

В частности, если имеет место функциональная зависимость

$$X \to Z,$$

то

$$I_0(Z/X) = 0,$$

так что для любого Y имеем

$$I_0(Z/X \otimes Y) = 0.$$

Поэтому

$$X \underset{Y}{\to\to} Z.$$

Приведем пример условно независимых признаков. Если X—тривиальное слово, состоящее из одной буквы:

$$X = (a, a, \ldots, a),$$

то понятие условной независимости совпадает с безусловной независимостью, поскольку в этом случае

$$I_0(Y/X) = I_0(Y)$$

для любого слова Y.

Поэтому мы можем строить условно независимые слова для произвольного слова X, рассматривая его канонические подслова, полученные ограничением на позиции, занятые одной и той же буквой:

X'	Z'	Y'
a	b	a
a	b	b
a	c	a
a	c	b

X''	Z''	Y''
b	a	c
b	a	d

Здесь Z' и Y' независимы, точно так же независимы Z'' и Y''. Теперь «склеиваем» эти слова:

X	Z	Y
a	b	a
a	b	b
a	c	a
a	c	b
b	a	c
b	a	d

В результате получаем условно независимые слова Y и Z.

Для фиксированного признака X рассмотрим множество $S(X)$, состоящее из всех признаков, дублирующих X относительно Y с точностью до $\varepsilon > 0$:

$$S(X) = \{Z \mid I_0(Y{:}(X \otimes Z)) - I_0(Y{:}X) < \varepsilon\}.$$

Это множество естественно назвать *кластером, соответствующим признаку* X. Кластер состоит из признаков, сходных с признаком X по его способности распознавать значение признака Y, т. е. проводить классификацию. Множество $S(X)$ состоит из

всех признаков Z, для которых с некоторой погрешностью выполняется зависимость

$$X \underset{Y}{\twoheadrightarrow} Z.$$

Все признаки из $S(X)$ непригодны для образования сложного признака на основе признака X.

2.4. Формирование сложных признаков

Обычно один отдельно взятый признак X_i раскрывает лишь частично неопределенность относительно признака Y, поэтому приходится рассматривать наборы признаков (сложные признаки).

В обозначениях главы 1 сложный признак U, составленный из простых признаков X и Z,—это слово

$$U = X \otimes Z.$$

Теорема 2.1. *Если X_1 и X_2 независимы, то информативность сложного признака $X_1 \otimes X_2$ не меньше суммы информативностей простых признаков, его составляющих.*

Более точно,

$$I_0(Y{:}(X_1 \otimes X_2)) \geq I_0(Y{:}X_1) + I_0(Y{:}X_2) - I_0(X_1{:}X_2).$$

Доказательство.

$$I_0(Y{:}(X_1 \otimes X_2)) = I_0(Y) - I_0(Y/X_1 \otimes X_2),$$

$$I_0(Y/X_1 \otimes X_2) = I_0(X_1 \otimes X_2 \otimes Y) - I_0(X_1 \otimes X_2) =$$

$$= I_0(X_1) + I_0(X_2 \otimes Y/X_1) - I_0(X_1) - I_0(X_2/X_1) =$$

$$= I_0(Y/X_1) + I_0(X_2/X_1 \otimes Y) - I_0(X_2/X_1) =$$

$$= I_0(Y/X_1) + I_0(X_2) - I_0(X_2/X_1) - (I_0(X_2) - I_0(X_2/X_1 \otimes Y)).$$

Поскольку

$$I_0(X_2/X_1 \otimes Y) \leq I_0(X_2/Y),$$

то

$$I_0(Y/X_1 \otimes X_2) \leq I_0(Y/X_1) + I_0(X_2) - I_0(X_2/X_1) - I_0(X_2{:}Y) \leq$$

$$\leq I_0(Y/X_1) + I_0(X_2{:}X_1) - I_0(X_2{:}Y),$$

так что

$$I_0(Y{:}(X_1 \otimes X_2)) \geq I_0(Y{:}X_1) + I_0(X_2{:}Y) - I_0(X_2{:}X_1).$$

Если признаки независимы, то $I_0(X_2{:}X_1) = 0$ и в этом случае получаем

$$I_0(Y{:}(X_1 \otimes X_2)) \geq I_0(Y{:}X_1) + I_0(Y{:}X_2).$$

Если же признаки повторяют друг друга, то величина $I_0(X_1{:}X_2)$ становится большой, а информативность сложного

признака оказывается такой же, что и информативность простого признака. Поэтому при *образовании сложного признака не следует объединять признаки, лежащие в одном и том же кластере.*

Введем *второй параметр настройки* β, равный минимально допустимой информативности сложного признака. Величина β колеблется обычно от 60 % $I_0(Y)$ до $I_0(Y)$. При выбранном значении β найдем все сложные признаки, информативность которых превышает β.

Сначала находим все двойные признаки, удовлетворяющие этому условию. В примере 2.1 при $\beta = 80\ \%\ I_0(Y) = 0.8 \cdot 14.26 = 11.4$ сочетание $X_3 \otimes X_6$ достаточно информативно, поэтому остается для дальнейшей обработки. Сочетание же $X_2 \otimes X_4$ имеет информативность, меньшую β, поэтому добавляется третий признак X_8. Тройной признак $X_2 \otimes X_4 \otimes X_8$ имеет уже достаточную информативность.

Для отобранных сложных признаков U и V выполняется условие

$$U \not\subset V,\ V \not\subset U.$$

Для каждого отобранного сложного признака составляем матрицу перехода букв этого признака (слова) в буквы признака Y. Например, для $X_2 \otimes X_4 \otimes X_8$ в примере 2.1, если заменить $1 \to 0,\ 2 \to 1,\ 3 \to 2$, получаем

	0	1	2	Сумма
100	1			1
110	1			1
100	1			1
011		2		2
010		1		1
001			2	2
101			1	1

(2.1)

Эта матрица применяется на следующем этапе распознавания.

Для примера 2.2 эти матрицы имеют вид ($\beta = 60\ \%$):

1) сочетание $X_{11} \otimes X_1$

11	1	Y
1	0	1
1	0	1
0	0	1
1	0	1
1	1	1
0	1	0
0	1	0
0	1	0
0	1	0
0	0	0

	0	1	
10	0	3	3
00	1	1	2
11	0	1	1
01	4	0	4
	5	5	10

$I_0(Y{:}(X_{11} \otimes X_1)) = 8$ бит;

2) сочетание $X_{11} \otimes X_2$

11	2	Y
1	1	1
1	1	1
0	1	1
1	0	1
1	1	1
0	0	0
0	0	0
0	0	0
0	0	0
0	1	0

	0	1	
11	0	3	3
01	1	1	2
10	0	1	1
00	4	0	4
	5	5	10

$I_0(Y{:}(X_{11} \otimes X_2)) = 8$ бит;

3) сочетание $X_1 \otimes X_2 \otimes X_7$

1	2	7	Y
0	1	1	1
0	1	1	1
0	1	0	1
0	0	1	1
1	1	1	1
1	0	0	0
1	0	1	0
1	0	0	0
1	0	0	0
0	1	0	0

	0	1	
011	0	2	2
010	1	1	2
001	0	1	1
111	0	1	1
100	3	0	3
101	1	0	1
	5	5	10

$I_0(Y{:}(X_1 \otimes X_2 \otimes X_7)) = 8$ бит;

4) сочетание $X_{11} \otimes X_7$

11	7	Y
1	1	1
1	1	1
0	0	1
1	1	1
1	1	1
0	0	0
0	1	0
0	0	0
0	0	0
0	0	0

	0	1	
11	0	4	4
00	4	1	5
01	1	0	1
	5	5	10

$I_0(Y{:}(X_{11} \otimes X_7)) = 6.4$ бит.

2.5. Переход в новое пространство признаков. Метрика Хэмминга

Будем считать новым признаком сложный признак, отобранный на предыдущем этапе. Переход в новое признаковое пространство осуществляется следующим образом.

Введем *третий параметр настройки* γ, *принимающий целочисленные значения* 1, 2, ... Действие этого параметра

поясним на примере. Пусть у некоторого объекта в старом пространстве признаки X_2, X_4, X_8 принимают значения 0,1,1 соответственно. Находим в соответствующей переходной матрице (2.1) сочетание (011) и соответствующую строку (C_0, C_1, C_2) (в данном случае $C_0 = 0$, $C_1 = 2$, $C_2 = 0$). Находим максимальное значение, скажем, C_1. Если окажется, например, что $C_1 - C_0 \geq \gamma$, $C_1 - C_2 \geq \gamma$, то соответствующий признак в новом пространстве принимает значение, равное 1. Если же в строке (C_0, C_1, C_2) нет «лидера», то записываем пробел (отказ от распознавания по данному признаку для данного объекта). В нашем примере при $\gamma = 2$ новый признак принимает значение 1, а при $\gamma = 3$—пробел. Таким образом, параметр γ определяет «степень доверия» к данному сложному признаку.

Размер нового пространства признаков совпадает с числом отобранных сложных признаков. В новом пространстве все объекты (образы) кластеризуются вокруг стандартных векторов (0, 0, ..., 0), (1, 1, ..., 1), (2, 2, ..., 2) и т. д.

Метрика Хэмминга определяется для двух слов как количество координат, в которых буквы этих слов отличаются друг от друга. Эта величина удовлетворяет всем аксиомам метрики. Если в пространстве признаков основной является информационная метрика $\rho(X_i, X_j)$, то в новом пространстве образы (объекты) кластеризуются относительно метрики Хэмминга $d(k, m)$: два объекта k и m, лежащие в одном и том же классе, оказываются близкими в смысле $d(k, m)$ в новом пространстве. При этом степень кластеризации регулируется параметром β, а параметр γ определяет реальный размер нового пространства признаков (с учетом отказов).

2.6. Распознавание

Перейдем к последнему этапу распознавания. Сущность предыдущих этапов заключается в том, что выполняется некоторое нелинейное преобразование исходного пространства, так что в новом пространстве сходные объекты становятся близкими по метрике Хэмминга.

Переведем контрольный объект в новое пространство и вычисляем расстояния $d(i, 0)$, $d(i, 1)$, ... от этого объекта до стандартных слов $0 = (0, 0, \ldots)$, $1 = (1, 1, \ldots)$, ... Если $d(i, j)$ минимально, то считаем, что i-й объект принадлежит j-му классу. Для примера 2.2 имеем при $X = 1$

Номер объекта	$11 \otimes 1$	$11 \otimes 2$	$1 \otimes 2 \otimes 7$	$11 \otimes 7$	Y
1	1	1	1	1	1
7	0	0	0	0	0
11	—	0	—	0	0
12	1	1	—	—	1

Настройку программы осуществляют, меняя γ, затем β и, наконец, α. Конкретные значения этих параметров зависят от общего размера обучающей выборки, числа исходных признаков и их информативности. Всегда следует стремиться к возможно большим значениям γ, но при этом появляется много отказов и, как следствие, уменьшается число признаков в новом пространстве. Это уменьшение можно компенсировать уменьшением параметра β, но тогда ухудшается кластеризация по Хэммингу (образы «расплываются»). Параметр α, отсекающий «шум», не следует выбирать слишком малым, так как при этом происходит искажение результатов распознавания.

Если, тем не менее, не удается настроить программу, то следует изменить обучающую выборку: некоторые объекты, использованные ранее для контроля, применить для обучения и наоборот.

3. РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

3.1. Отношения

В гл. 1 рассматривались отдельные слова и были введены некоторые характеристики слов и пар слов. В гл. 2 основным объектом была информационная матрица, строки которой назывались образами, а столбцы—признаками (атрибутами). В информационной матрице присутствует обычно некоторая избыточность, позволявшая предсказывать значение некоторого признака, если известны значения других признаков.

В этой главе мы будем рассматривать совокупность информационных матриц (будем называть их отношениями). Такой способ описания базы данных был предложен Е. Коддом.

Пример 3.1

Табельный номер	Ф.И.О.	Должность	Номер комнаты	Телефон	Дети	
					Имя	Возраст
010	Петров Н.Л.	Инженер	10	016	Саша	6
					Игорь	4
					Вова	2
102	Иванов А.И.	Зав. лаб.	20	010	Аня	10
					Игорь	8
080	Сергеев С.Н.	Ст. инженер	10	016	Вова	6

Каждому атрибуту X_i ставится в соответствие множество его значений D_i (*домен атрибута*). Сложный признак $K = X_1 \otimes X_2 \otimes \ldots \otimes X_m$ называется *возможным ключом*, если он однозначно определяет все остальные значения признаков. Другими словами, не существует двух строк, имеющих одно и то же значение на всех атрибутах из K, так что имеет место функциональная зависимость

$$K \to Y_i$$

для любого атрибута Y_i и данного отношения.

Кроме того, предполагается, что ни один атрибут X_i не может быть удален без нарушения этого свойства. Другими словами, возможный ключ—это минимальный сложный признак, полностью раскрывающий неопределенность относительно других признаков.

Обычно один из возможных ключей фиксируется: значения его атрибутов остаются неизменными. Такой ключ используется в качестве идентификатора объекта при поиске информации и называется *первичным ключом.*

Состояние отношения изменяется во времени. Прежде всего, может добавляться новая строка (операция «Вставка»). Кроме того, некоторая строка может стираться (операция «Удаление»). Можно изменять лишь часть строки. Модификация достигается с помощью операции «Изменение».

База данных должна реагировать на запросы. Выбор—это элементарная операция над отношением. Результатом ее применения является другое отношение, которое представляет собой подмножество строк с определенными значениями в выделенных атрибутах.

Можно выбрать также подмножество столбцов (атрибутов). Такая операция называется «Проекцией». При создании базы данных стремятся избегать аномалий при обновлении данных и избавиться от информационной избыточности в отношениях.

Обратимся к примеру 3.1 [3]. Предположим, что у комнаты 10 изменился номер телефона. При обновлении базы данных потребовалось бы найти все строки, у которых соответствующий атрибут принимает значение 016 (аномалия изменения). Кроме того, здесь наблюдается избыточность в хранении информации, поскольку номер телефона 016 дублируется во многих строках. Возникает желание заменить отношение «Сотрудники отдела» на два отношения:

Отношение «Сотрудники»

Табельный номер	Ф.И.О.	Должность	Номер комнаты	Дети	
				Имя	Возраст
010	Петров Н.Л.		10		
102	Иванов А.И.		20		
080	Сергеев С.Н.		10		

Отношение «Комнаты»

Номер комнаты	Телефон
6	004
7	001
8	007
10	016
20	010

Представление исходного отношения в виде пары отношений называется *декомпозицией.*

Восстановить первоначальное отношение можно с помощью операции «Соединение»: в обоих отношениях должен присутствовать общий атрибут (в данном случае «номер комнаты»). Находим строки с одним и тем же значением этого атрибута в

обоих отношениях и расширяем первое отношение, добавляя новый атрибут «Телефон».

Любое априорное знание о различного рода зависимостях между атрибутами может принести большую пользу для оптимального выбора схемы отношений.

3.2. Функциональные зависимости

Функциональная зависимость (*F*-зависимость) $X \to Y$ была определена в гл. 2 для двух слов X и Y посредством равенства $I_0(Y/X) = 0$. Если известны некоторые *F*-зависимости для атрибутов, то можно вывести остальные, используя некоторые правила вывода, которые являются следствиями предложений 1.3—1.7.

F1. *Рефлексивность*: $X \to X$.

Доказательство.

$$I_0(X/X) = 0.$$

F2. *Пополнение*: $X \to Y$ *влечет за собой*

$$X \otimes Z \to Y.$$

Доказательство.

$$I_0(Y/X \otimes Z) \leq I_0(Y/X) = 0.$$

F3. *Аддитивность*: $X \to Y$ *и* $X \to Z$ *влечет за собой* $X \to Y \otimes Z$.

Доказательство.

$$I_0(Y \otimes Z/X) = I_0(Y/X) + I_0(Z/X \otimes Y) =$$

$$= I_0(Z/X \otimes Y) \leq I_0(Z/X) = 0.$$

F4. *Проективность*: $X \to Y \otimes Z$ *влечет за собой* $X \to Y$.

Доказательство.

$$0 = I_0(Y \otimes Z/X) = I_0(Y/X) + I_0(Z/X \otimes Y),$$

так что

$$I_0(Y/X) = 0, \quad I_0(Z/X \otimes Y) = 0.$$

F5. *Транзитивность*: $X \to Y$ и $Y \to Z$ *влечет за собой* $X \to Z$.

Доказательство.

$$I_0(Z/X) \leq I_0(Z/Y) + I_0(Y/X) = 0.$$

F6. *Псевдотранзитивность*: $X \to Y$ и $Y \otimes Z \to W$ *влечет за собой* $X \otimes Z \to W$.

Доказательство.

$$I_0(W/X \otimes Z) \le I_0(W/Y \otimes Z) + I_0(Y \otimes Z/X \otimes Z) =$$

$$I_0(Y \otimes Z/X \otimes Z) = I_0(Y/X \otimes Z) + I_0(Z/Y \otimes X \otimes Z) \le$$

$$\le I_0(Y/X) + I_0(Z/Z) = 0.$$

Выше приведенные правила позволяют строить новое множество, отправляясь от некоторого множества F-зависимостей и добавляя новые F-зависимости. Пусть, например,

$$F = \{X \to Y, X \otimes Y \to Z \otimes U, U \to V\}.$$

Используя пополнение, получаем

$$Z \otimes U \to V.$$

Используя транзитивность, имеем

$$X \otimes Y \to V.$$

Из рефлексивности

$$X \otimes Y \to X \otimes Y$$

следует

$$X \otimes Y \to X \otimes Y \otimes V.$$

Наконец, применяя аддитивность, получаем

$$X \otimes Y \to X \otimes Y \otimes V \otimes Z \otimes U.$$

Обозначим через A множество правил

$$A = \{F1, F2, \ldots, F6\}.$$

Применяя эти правила к некоторому множеству F-зависимостей несколько раз, получаем расширенное множество зависимостей, которое стабилизируется на некотором шаге. Назовем его замыканием множества F и обозначим F_A^*. Очевидно

$$\left(F_A^*\right)^* = F_A^*.$$

Добавим к множеству A некоторое новое правило. В результате получаем множество $B \supseteq A$. Применяя правила из B к множеству F несколько раз, получаем множество зависимостей, которое обозначим F_B^*. Ясно, что $F_A^* \subseteq F_B^*$, а если $F_A^* = F_B^*$ для всех F и $B \supseteq A$, то систему правил A можно считать *полной*.

Теорема 3.1. *Система правил вывода* F1—F6 *является полной.*

Доказательство. Допустим, что $F_A^* \neq F_B^*$ для некоторых F и $B \supseteq A$. Это означает, что существует зависимость $X \to Y$, которая не может быть выведена из F с помощью правил F1—F6. Это должно быть справедливо для всех отношений с

данным множеством атрибутов. Тем не менее, мы построим отношение, которое удовлетворяет всем зависимостям из F_A^*, но не удовлетворяет зависимости $X \to Y$.

Поскольку $X \to X$, а из $X \to Z$ следует $X \to X \otimes Z$ ввиду аддитивности, то существует максимальное подмножество X^* (сложный признак), такое, что $X \to X^*$ и для любой другой зависимости $X \to Z$ справедливо включение $Z \subseteq X^*$.

Пусть $\{X_1, X_2, \ldots, X_n\}$—множество атрибутов. Отношение, которое мы строим, состоит из двух строк t и t'. Через $t(W)$ будем обозначать ограничение строки t на простых признаках, входящих в W. Каждый столбец отношения имеет вид

$$\begin{pmatrix} a_i \\ a_i \end{pmatrix}, \text{ если } X_i \in X^*; \quad \begin{pmatrix} a_i \\ b_i \end{pmatrix}, \ a_i \neq b_i, \text{ если } X_i \notin X^*.$$

Покажем, что полученное отношение удовлетворяет всем зависимостям $W \to Z$ из F^*. Если $W \not\subseteq X^*$, то $t(W) \neq t'(W)$ и зависимость $W \to Z$ выполняется тривиально. Поэтому предположим, что $W \subseteq X^*$. Тогда $t(W) = t'(W)$. Ясно, что $X^* \to W$ ввиду рефлексивности и проективности. Поскольку $X \to X^*$, применение транзитивности дает $X \to Z$. Следовательно, $Z \subseteq X^*$, так что $t(Z) = t'(Z)$ как только $t(W) = t'(W)$. Это означает, что отношение удовлетворяет зависимости $W \to Z$.

Предположим, что выполняется также $X \to Y$. Тогда $Y \subseteq X^*$. Но так как зависимость $X \to X^*$ входит в F_a^*, то, согласно проективности, $(X \to Y) \in F_A^*$. Поскольку мы предположили, что $(X \to Y) \notin F_A^*$, приходим к противоречию.

3.3. Декомпозиция на основе функциональных зависимостей

Все признаки в отношении должны быть атомарными (неделимыми), т.е. значения в домене не являются ни списками, ни множествами простых или сложных значений. В примере 3.1 признак «Дети», очевидно, не удовлетворяет этому требованию. Соответствующее отношение следует представить, расщепляя сложный признак на его составные части.

Пример 3.2.

Отношение «Сотрудники отдела»

Табельный номер	Имя	Ф.И.О.	Возраст	Должность	Номер комнаты	Телефон
010	Саша	Петров Н.Л.	6	Инженер	10	016
010	Игорь	Петров Н.Л.	4	Инженер	10	016
010	Вова	Петров Н.Л.	2	Инженер	10	016
102	Аня	Иванов А.И.	10	Зав. лаб.	20	010
102	Игорь	Иванов А.И.	8	Зав. лаб.	20	010
080	Вова	Сергеев С.Н.	6	Ст. инженер	10	016

Говорят, что *отношение приведено при этом к 1-й нормальной форме.* В таблице выделен первичный ключ [Табельный номер, Имя]. Других возможных ключей здесь нет. Можно было бы вместо атрибута «Табельный номер» использовать атрибут «Ф.И.О.» (для данного состояния отношения эти атрибуты находятся во взаимно однозначной зависимости). Однако мы должны иметь в виду возможность появления нового сотрудника с точно такой же фамилией, именем и отчеством—ему будет присвоен свой табельный номер. Поэтому функциональной связи Ф.И.О.→ Табельный номер, вообще говоря, не существует.

Составляем граф функциональных зависимостей для данного отношения:

Говорят, что атрибут X *полностью зависит* от ключа K, если не существует функциональной зависимости $Y \to X$ для любого собственного подмножества $Y \subseteq K$. В противном случае говорят о *частичной (неполной) зависимости.*

В нашем примере атрибут «Возраст ребенка» полно зависит от ключа, а остальные атрибуты частично. Наличие неполной зависимости приводит к избыточности данных и создает аномалии при обновлении.

Из отношения следует исключить атрибуты, которые не находятся в полной зависимости от ключа. После этого нужно построить дополнительно одну или несколько проекций на часть составного ключа и атрибуты, функционально зависящие от этой части ключа. В нашем примере следует разбить исходное отношение на два:

Отношение «Сотрудники»

Табельный номер	Ф.И.О.	Должность	Номер комнаты	Телефон
010	Петров Н.Л.	Инженер	10	016
102	Иванов А.И.	Зав. лаб.	20	010
080	Сергеев С.Н.	Ст. инженер	10	016

Отношение «Дети сотрудников»

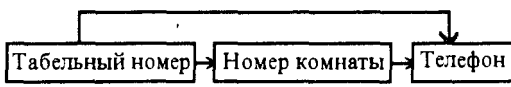
Табельный номер	Имя	Возраст
010	Саша	6
010	Игорь	4
010	Вова	2
102	Аня	10
102	Игорь	8
080	Вова	6

Говорят, что *отношение приведено ко 2-й нормальной форме*, если каждый непервичный атрибут (т.е. атрибут, не входящий ни в один из ключей) функционально полно зависит от каждого возможного ключа.

Дальнейшее устранение избыточности связано с понятием транзитивной зависимости. Обратимся к отношению «Сотрудники». Граф зависимостей для этого отношения имеет вид

Скажем, что *имеет место транзитивная зависимость между признаками* X, Y и Z, если $X \to Y$, $Y \to Z$, но $Z \not\to Y$ или $Y \not\to X$.

В нашем примере транзитивная зависимость имеет вид



Устранение такой зависимости осуществляется дальнейшим расщеплением отношения:

Отношение «Сотрудники»

Табельный номер	Ф.И.О.	Должность	Номер комнаты
010	Петров Н.Л.	Инженер	10
102	Иванов А.И.	Зав. лаб.	20
080	Сергеев С.Н.	Ст. инженер	10

Отношение «Комнаты»

Номер комнаты	Телефон
10	016
20	010

После устранения транзитивности зависимостей отношение попадает в *3-ю нормальную форму.*

Задача нахождения транзитивных зависимостей тесно связана с поиском остовного дерева графа. Соответствующий алгоритм будет рассмотрен в следующей главе.

3.4. Декомпозиция и условная независимость

В гл. 2, когда речь шла о кластерном анализе в пространстве признаков, было введено понятие условной независимости признаков.

Рассмотрим пример:

X	Z	Y
a	b	a
a	b	b
a	c	a
a	c	b
b	b	a
b	b	c

в котором Y и Z независимы при условии X. Присвоим переменным X, Y, Z следующие значения:

X—фамилия лектора,

Z—день недели,

Y—номер студенческой группы.

Получаем отношение «Расписание»

Ф.И.О.	День недели	Номер группы
Петров Н.Л.	Понедельник	Ф-17
Петров Н.Л.	Понедельник	М-32
Петров Н.Л.	Среда	Ф-17
Петров Н.Л.	Среда	М-32
Иванов А.И.	Понедельник	Ф-17
Иванов А.И.	Понедельник	К-10

Здесь существует единственный возможный ключ, содержащий все три признака, отношение находится в 3-й нормальной форме. Тем не менее, возможна дальнейшая декомпозиция:

X	Z
a	b
a	c
b	b

X	Y
a	a
a	b
b	a
b	c

Ф.И.О.	День недели
Петров Н.Л.	Понедельник
Петров Н.Л.	Среда
Иванов А.И.	Понедельник

Ф.И.О.	Номер группы
Петров Н.Л.	Ф-17
Петров Н.Л.	М-32
Иванов А.И.	Ф-17
Иванов А.И.	К-10

Очевидно, с помощью соединения можно однозначно восстановить исходное отношение, так что декомпозиция осуществляется без потерь.

Рассмотрим два отношения $R_1(X, Z)$ и $R_2(X, Y)$, каждое из которых содержит по два признака. Обозначим через $R(X, Z, Y)$ отношение, полученное соединением этих отношений:

$$R(X, Z, Y) = R_1(X, Z) \times R_2(X, Y).$$

Обозначим через $\overline{R}_1$, $\overline{R}_2$, $\overline{R}$ проекции этих отношений на строки с фиксированным значением атрибута X, общего для обоих отношений R_1, R_2. В нашем примере для $X = a$ имеем

$\overline{R}_1$:

$\overline{X}_1$	$\overline{Z}_1$
a	b
a	c

$\overline{R}_2$:

$\overline{X}_2$	$\overline{Y}_2$
a	a
a	b

$\overline{R}$:

$\overline{X}$	$\overline{Z}$	$\overline{Y}$
a	b	a
a	b	b
a	c	a
a	c	b

Очевидно, слово $\overline{Z} \otimes \overline{Y}$ является прямым произведением слов:

$$\overline{Z} \otimes \overline{Y} = \overline{Z}_1 \times \overline{Y}_2.$$

(определение прямого произведения и его свойства даны в пункте 1.14). Таким образом, $\overline{Z}$ и $\overline{Y}$ независимы (см. теорему 1.9). Для $X = b$ получаем

$\overline{\overline{R}}_1$:

$\overline{\overline{X}}_1$	$\overline{\overline{Z}}_1$
b	b

$\overline{\overline{R}}_2$:

$\overline{\overline{X}}_2$	$\overline{\overline{Y}}_2$
b	a
b	c

$\overline{\overline{R}}$:

$\overline{\overline{X}}$	$\overline{\overline{Z}}$	$\overline{\overline{Y}}$
b	b	a
b	b	c

Слово $\overline{\overline{Z}} \otimes \overline{\overline{Y}}$ является также прямым произведением, а $\overline{\overline{Z}}$ и $\overline{\overline{Y}}$ независимы.

Таким образом, при соединении отношений $R_1(X, Z)$ и $R_2(X, Y)$ получаются слова Z и Y, условно независимые при условии X. При образовании прямого произведения информация соответствующих слов суммируется. Если исходное отношение $R(X, Z, Y)$ можно "расщепить" на два отношения, а затем "соединить", возвращаясь к исходному отношению, то Z и Y должны быть условно независимы. Тем самым доказан следующий результат.

Теорема 3.2. *Для отношения* $R(X, Z, Y)$ *возможна декомпозиция без потерь тогда и только тогда, когда* Z *и* Y *условно независимы при условии* X. *В этом случае* $R(X, Z, Y)$ *является соединением отношений* $R_1(X, Z)$ *и* $R_2(X, Y)$.

3.5. Многозначная зависимость

Пусть $R\{X_1, X_2, \ldots, X_n\}$—некоторое множество атрибутов (отношение). Подмножество $X \subseteq R$ будем отождествлять со сложным атрибутом, а $X \otimes Y$ соответствует объединению множеств $X \cup Y$.

Обозначим $Z = R - (X \otimes Y)$. Если Y и Z условно независимы при условии X, то будем говорить, что X и Y связаны *многозначной зависимостью*

$$X \twoheadrightarrow Y.$$

Как было показано в гл. 2, это равносильно тому, что

$$X \twoheadrightarrow Z.$$

Согласно определению, в этом случае

$$I_0(Z/X \otimes Y) = I_0(Z/X), \qquad I_0(Y/X \otimes Z) = I_0(Y/X)$$

и возможна декомпозиция отношения R без потери информации (см. теорему 3.2.).

Для многозначной зависимости существуют правила вывода, аналогичные правилам для функциональных зависимостей. Эти правила позволяют находить новые зависимости, если известно некоторое множество многозначных зависимостей.

М1. *Рефлексивность*: $X \twoheadrightarrow X$.

Доказательство.

$$I_0(X/X \otimes Z) = 0 = I_0(X/X).$$

М2. *Пополнение*: *Если* $X \twoheadrightarrow Y$, *то* $X \otimes U \twoheadrightarrow Y$ *для любого* U.

Доказательство. Достаточно предположить, что U совпадает с простым атрибутом: $U = X_i$. Если $U \in X$, то доказательство тривиально. Если $U \in Y$, то

$$I_0(Z/X \otimes Y \otimes U) = I_0(Z/X \otimes Y) = I_0(Z/X) = I_0(Z/X \otimes U).$$

Если $U \in Z$, то

$$I_0(Y/X \otimes Z \otimes U) = I_0(Y/X \otimes Z) = I_0(Y/X) = I_0(Y/X \otimes U).$$

М3. *Аддитивность*: *Если* $X \twoheadrightarrow Y$ и $X \twoheadrightarrow U$, *то* $X \twoheadrightarrow Y \otimes U$.

Доказательство. $Z = R - X \otimes Y \otimes U$.

$$I_0(Y \otimes U/X \otimes Z) = I_0(Y/X \otimes Z) + I_0(U/X \otimes Z \otimes Y) =$$

$$= I_0(Y/X) + I_0(U/X) =$$

$$= I_0(Y/X) + I_0(U/X \otimes Y) = I_0(Y \otimes U/X).$$

M4. *Проективность*: *Если* $X \twoheadrightarrow Y$ *и* $X \twoheadrightarrow U$, *то* $X \twoheadrightarrow \twoheadrightarrow Y \cap U$.

Доказательство. Пусть $Z_1 = R - (X \otimes Y)$. $Z_2 = R - (X \otimes U)$ и $Z = R - (X \otimes (Y \cap U))$. Обозначим через $\overline{X}$, как обычно, дополнение множества X, т. е. $\overline{X} = R - X$. Тогда

$$Z_1 = \overline{X \cup Y}, \; Z_2 = \overline{X \cup U},$$

$$Z = \overline{X \cup (Y \cap U)} = \overline{(X \cup Y) \cap (X \cup U)} = \overline{X \cup Y} \cup \overline{X \cup U} =$$

$$= Z_1 \cup Z_2 = Z_1 \otimes Z_2,$$

$$I_0(Z/X \otimes (Y \cap U)) = I_0(Z_1 \otimes Z_2/X \otimes (Y \cap U)) =$$

$$= I_0(Z_1/X \otimes (Y \cap U)) + I_0(Z_2/X \otimes (Y \cap U) \otimes Z_1) =$$

$$= I_0(Z_1/X) + I_0(Z_2/X) = I_0(Z_1/X) + I_0(Z_2/X \otimes Z_1) =$$

$$= I_0(Z_1 \otimes Z_2/X) = I_0(Z/X).$$

M5. *Транзитивность*: *Если* $X \twoheadrightarrow Y$ *и* $X \twoheadrightarrow U$, то

$$X \twoheadrightarrow Y - U.$$

Доказательство. $Y = (Y - U) \cup (Y \cap U)$. Докажем утверждение, из которого будет следовать нужное нам свойство относительно $Y - U$. Если $X \twoheadrightarrow Y$ и $X \twoheadrightarrow U$, где $U \subseteq Y$, то $X \twoheadrightarrow Y - U$. Имеем

$$Y = U \otimes Y', \qquad Y' = Y - U.$$

$$I_0(Y/X) = I_0(Y/X \otimes Z_1) = I_0(U \otimes Y'/X \otimes Z_1) =$$

$$= I_0(U/X \otimes Z_1) + I_0(Y'/X \otimes Z_1 \otimes U) =$$

$$= I_0(U/X) + I_0(Y'/X \otimes Z_1 \otimes U).$$

Поскольку

$$I_0(Y/X) = I_0(U \otimes Y'/X) = I_0(Y'/X) + I_0(U/X \otimes Y') =$$

$$= I_0(Y'/X) + I_0(U/X),$$

получаем

$$I_0(Y'/X \otimes Z_1 \otimes U) = I_0(Y'/X).$$

M6. *Псевдотранзитивность*: *Если* $X \twoheadrightarrow Y$ *и* $Y \otimes W \twoheadrightarrow U$, *то*

$$X \otimes W \twoheadrightarrow U - (Y \otimes W).$$

Доказательство. Свойство $X \twoheadrightarrow Y$ в сочетании с пополнением дает

$$X \otimes W \twoheadrightarrow Y.$$

Далее,

$$X \otimes W \twoheadrightarrow W, \qquad I_0(W/X \otimes W \otimes Z) = 0 = I_0(W/X \otimes W).$$

Применяя аддитивность, получаем,

$$X \otimes W \twoheadrightarrow Y \otimes W.$$

Поскольку

$$Y \otimes W \twoheadrightarrow U,$$

из транзитивности следует

$$X \otimes W \twoheadrightarrow U - (Y \otimes W).$$

Так же, как и в случае функциональных зависимостей, можно показать, что система правил M1—M6 является полной, так что, применяя эти правила, можно получить все многозначные зависимости, характерные для данного отношения. После устранения этих зависимостей отношение оказывается в *4-й нормальной форме.*

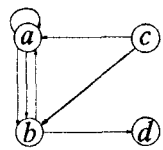
4.1. Граф алгебраического канала

В гл. 1 алгебраический канал был определен посредством пары слов $x \otimes y$. Этот канал преобразует слова орбиты O_x в слова орбиты O_y, а закон преобразования определяется переходной матрицей, соответствующей словам x и y.

В то же время паре слов $x \otimes y$ соответствует некоторый граф, если считать его ребром столбец $\binom{a}{b}$ из пары $x \otimes y$. Например, если

$$\begin{matrix} x \\ y \end{matrix} \left| \begin{matrix} a & a & a & b & b & c & c \\ a & b & b & a & d & b & a \end{matrix} \right),$$

то граф выглядит следующим образом:



Заметим что мы получили граф самого общего вида—ориентированный граф с кратными ребрами и петлями. Такой граф обычно называют *мультиграфом*. Обратно, каждому мультиграфу соответствует некоторая пара слов $x \otimes y$. Поскольку граф не зависит от того, в каком порядке выписаны его ребра, пара $x \otimes y$ и пара $\sigma x \otimes \sigma y$, где σ—некоторая подстановка, определяют один и тот же граф. Мы будем всегда определять граф посредством пары слов $x \otimes y$, где x отсортировано (его буквы лексикографически упорядочены).

Таким образом, *понятие алгебраического канала эквивалентно понятию графа.*

Переходная матрица канала является матрицей инцидентности соответствующего графа:

	a	b	c	d	
a	1	2			3
b	1			1	2
c	1	1			2
d					0
	3	3		1	7

Множество ребер графа обычно обозначают через E, а величина $|E|$ определяет сложность объекта «граф»: чтобы задать граф, нужно каким-то образом перечислить его ребра.

Граф, у которого каждая пара вершин соединена ребром, называется *полным*. Очевидно, у полного графа $|E| = |V|^2$, где V—множество вершин. Таким образом, всегда выполнено неравенство

$$|E| \leq |V|^2.$$

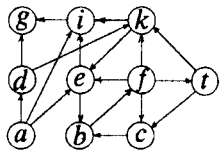
Мультиграф удобно записывать парой слов $x \otimes y$ с указанием внизу кратности соответствующей дуги:

$$\begin{pmatrix} a & a & b & b & c & c \\ a & b & a & d & b & a \\ \hline 1 & 2 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Таким образом, мультиграф—это обычный граф, у которого каждое ребро снабжается некоторым целым числом (весом ребра).

4.2. Поиск пути

Основной алгоритм на графе называется «Поиск пути». Этот алгоритм позволяет найти путь из вершины a в вершину t, состоящий из минимального количества ребер:



Компактная запись этого графа:

$$\begin{pmatrix} a & a & a & b & c & d & d & e & e & f & f & f & f & i & k & k & t & t \\ d & e & i & f & b & g & k & b & i & c & e & k & t & g & e & i & c & k \end{pmatrix}.$$

Поиск пути начинаем с того, что определяем вершины 1-го поколения. Это множество вершин, в которые можно попасть из начальной вершины. Каждой из этих вершин ставим в соответствие метку—вершину, из которой мы пришли:

	Метка
a	
b	
c	
d	a
e	a
f	
g	
i	a
k	
t	

Теперь находим вершины 2-го поколения, просматривая все вершины, помеченные на предыдущем этапе. Если вершина уже помечена, то новая метка не ставится.

	Метка 1-го поколения	Метка 2-го поколения
a		
b		*e*
c		
d	*a*	
e	*a*	
f		
g		*d*
i	*a*	
k		*d*
t		

Продолжаем этот процесс, пока не будет помечена конечная вершина *t*.

	Метка 1-го поколения	Метка 2-го поколения	Метка 3-го поколения	Метка 4-го поколения
a				
b		*e*		
c				
d	*a*			
e	*a*			
f			*b*	
g		*d*		
i	*a*			
k		*d*		
t				*f*

Для наглядности мы «разнесли» метки разных поколений по различным столбцам. На самом деле все метки следует хранить в одном столбце. Как только конечная вершина *t* оказалась помеченной, процесс расстановки меток прекращается. Искомый путь строим, отправляясь от конечной вершины. Находим метку вершины *t*—она равна *f*. Теперь находим метку вершины *f*—она равна *b*, и т. д. В результате получаем

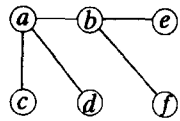
$$\text{Путь} = (a,\ e,\ b,\ f,\ t).$$

Граф называется *связным*, если для любой пары вершин (i, j) существует путь из i в j. Описанный алгоритм можно применить для определения связности графа, но проще эту задачу решить с помощью остовного дерева.

4.3. Остовное дерево графа

Деревом называется неориентированный связный граф без циклов.

Пример дерева:



В каждом дереве найдется хотя бы одна вершина, которая связана лишь одним ребром с остальной частью графа (такая вершина называется *висячей*). Действительно, допустим, что каждая вершина имеет по крайней мере два ребра. Тогда можно «войти» в вершину i по одному ребру, а «выйти» по другому. В результате получим новую вершину j. С этой вершиной можно поступить аналогичным образом. В результате получим последовательность разных вершин, которая не может бесконечно увеличиваться—обязательно наступит повторение вершин, а это означает, что в графе существует цикл.

Если удалить висячую вершину из дерева, состоящего из $|V|$ вершин и $|E|$ ребер, то останется дерево, состоящее из $|V|-1$ вершины и $|E|-1$ ребра. Продолжая этот процесс удаления, прийдем в конце концов к тривиальному дереву из одной вершины. Таким образом, *любое дерево удовлетворяет условию*

$$|V| = |E| + 1.$$

Пусть теперь G—произвольный связный граф, $e = \binom{s}{t}$—его ребро. Допустим, что существует цикл, содержащий это ребро. Тогда существует путь из s в t, не проходящий через ребро e. Поэтому можно удалить это ребро из графа, не нарушая связности. Продолжая этот процесс, прийдем в конце концов к связному подграфу без циклов с тем же множеством V вершин, т.е. к дереву. Такой подграф называется *остовным деревом графа* или *остовом*. Ясно также, что равенство

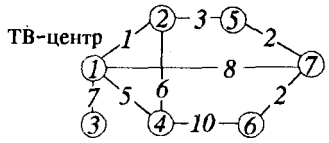
$$|V| = |E| + 1$$

выполняется только для деревьев. У произвольного графа, не являющегося деревом, число ребер больше:

$$|E| \geq |V| - 1.$$

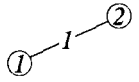
Существует простой алгоритм для нахождения остова. Мы рассмотрим даже более общую *задачу о минимальном остове*, когда каждому ребру поставлено в соответствие некоторое неотрицательное число (расстояние) и требуется найти остов с минимальным суммарным расстоянием.

Можно представить себе проектируемую телевизионную кабельную сеть для района-новостройки, где для каждого ребра указано расстояние в км:

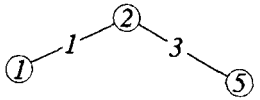


Кажется естественным осуществлять реализацию этого плана поэтапно: вначале соединить все пункты минимальной сетью, а затем наращивать дополнительные связи.

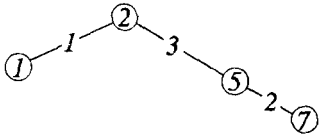
Начинаем поиск остовного дерева с произвольной вершины, скажем, с 1. Находим все ребра, выходящие из этой вершины, и выбираем ребро с минимальным расстоянием:



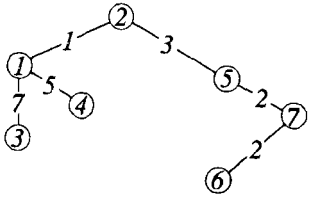
Теперь просматриваем все ребра, входящие в множество {1, 2}, и находим минимальное ребро:



Аналогично поступаем с множеством {1, 2, 5}:



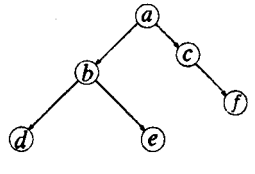
Продолжая этот процесс, получаем в конце остовное дерево с минимальным суммарным расстоянием $R = 20$ км:



4.4. Ордерево. Иерархические структуры

Ориентированный граф (орграф) называется *ордеревом*, если граф, получаемый при игнорировании ориентации дуг, является деревом.

Пример ордерева:



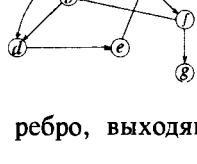
В таком графе отсутствуют циклы и закорачивания—нельзя, например, из вершины a провести дугу в вершину d (используя терминологию предыдущей главы, мы получили бы транзитивную зависимость, которая при игнорировании ориентации приводила бы к циклу (a, b, d)).

В ордереве в каждую вершину, кроме одной, которая называется *лидером* или *корнем*, входит ровно одна дуга.

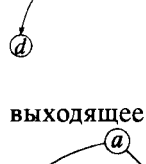
Ордерево описывает обычно иерархическую структуру (отношение подчиненности). Представленное на рисунке дерево называют *деревом, выходящим из корня a.* Можно изменить ориентацию каждой дуги на противоположную. Тогда получится *дерево, входящее в корень a.* Все пути в таком графе заканчиваются вершиной a.

Чтобы найти остовное дерево, выходящее из вершины a для заданного орграфа (или же определить, что его не существует), поступаем так же, как и в неориентированном случае.

Пусть, например, задан орграф:

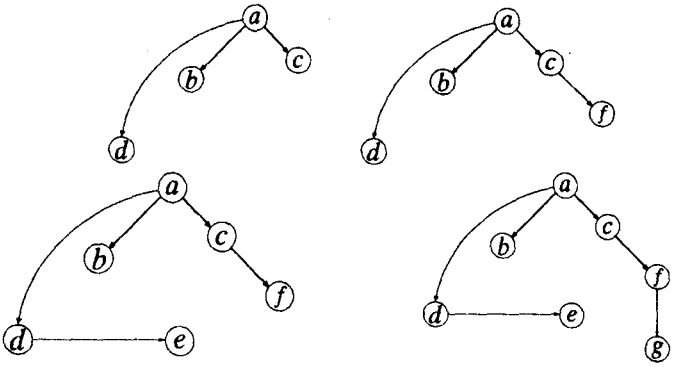


Выбираем какое-либо ребро, выходящее из вершины a:



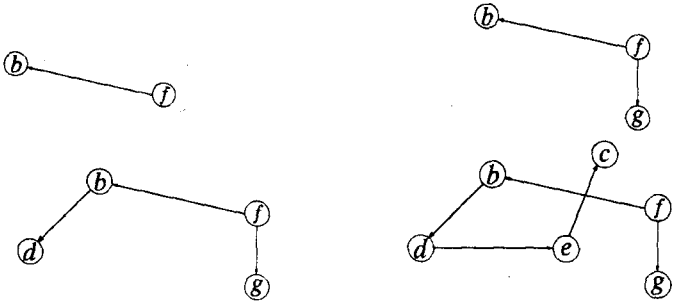
Теперь находим ребро, выходящее из множества $\{a, d\}$:

То же самое проделаем для множества $\{a, b, c\}$ и т. д.:



Таким образом, существует остовное дерево, выходящее из вершины a.

В то же время не существует остовного дерева, выходящего из вершины f:



Дальнейшее продолжение невозможно, так что вершину a мы не можем присоединить к данному дереву.

Это можно было предвидеть заранее—в вершину a не входит ни одна дуга (такая вершина называется *корневой*). Таким образом:

Если в орграфе присутствует корневая вершина, то любое остовное дерево может выходить только из этой вершины.

Если в орграфе присутствует больше одной корневой вершины, то не существует остовного дерева.

Компактная запись ордерева в памяти компьютера осуществляется с помощью слов $x \otimes y$:

$$\begin{matrix} x \\ y \end{matrix}\begin{pmatrix} a & a & a & c & d & f \\ b & c & d & f & e & g \end{pmatrix}.$$

Чтобы узнать, что $x \otimes y$ действительно соответствует некоторому ордереву, нужно подсчитать общее число различных букв в обоих словах. Это число равно числу вершин графа:

$$|V| = 7.$$

Затем следует сравнить это число с длиной слова, которая совпадает с числом ребер:

$$|E| = 6.$$

Если выполняется соотношение

$$|V| = |E| + 1,$$

то заключаем, что $x \otimes y$ определяет ордерево.

4.5. Бинарный поиск

Пусть $R = \{a, b, c, \ldots\}$—множество из N элементов. Допустим, что относительно некоторого объекта x известно лишь, что он является элементом множества R. Требуется идентифицировать этот объект, т.е. установить, какой букве он соответствует. В простейшем случае можно по очереди выполнить сравнения

$$x = a?\ x = b?, \ \ldots\ ,$$

пока не получим положительный ответ. Очевидно, в этом случае может потребоваться N сравнений. Ускорить поиск можно, разбивая R на два непересекающихся подмножества:

$$R = A + \overline{A}, \quad A \subseteq R,$$

и задавая вопрос:

$$x \in A?$$

В случае положительного ответа разбиваем A на два подмножества

$$A = A' + \overline{A}', \quad A' \subseteq A$$

и спрашиваем:

$$x \in A'?$$

и т.д., пока не получим точное значение элемента x. Возникает вопрос: как следует выбирать множество A? С точки зрения распознавания образов требуется раскрыть неопределенность признака (слова) Y, где

$$Y = (a, b, c, d, e, f).$$

Мы пытаемся раскрыть эту неопределенность с помощью косвенного признака

$$X_A = (1, 1, 0, 0, 0, 0),$$

который получается, когда мы отождествляем все буквы из множества A, приписывая им значение 1. При этом все буквы из $\overline{A}$ получают значение 0. Информативность признака X_A, равная $I_0(X_A{:}Y)$, должна быть максимальна.

Переход от Y к X_A соответствует группировке наблюдений (квантованию). В предложении 1.8 было показано, что

$$I_0(X_A:Y) = I_0(X_A) \leq N.$$

Таким образом, максимальное количество информации мы получаем, когда разбиваем исходное множество R на два равномощных множества:

$$|A| = N/2, \quad I_0(X_A) = N.$$

Следовательно, идентифицировать объект x можно, задавая $]\log N[$ бинарных вопросов, на которые можно ответить лишь «да» или «нет». Меньшее количество таких вопросов не позволит нам раскрыть неопределенность признака Y. Действительно, энтропия слова Y равна

$$I_0(Y) = N \log N,$$

а признак X_A уменьшает эту неопределенность максимум на N бит.

Пусть теперь каждый элемент множества R соответствует некоторой записи в компьютере. Каждая запись однозначно определяется своим ключом K_i (целым числом). Множество R записей называется *файлом*. Предполагается, что файл отсортирован:

$$K_1 < K_2 < \ldots < K_N.$$

Требуется найти запись с ключом K. Как мы показали только что, самый быстрый поиск связан с делением файла пополам.

4.6. Быстрая сортировка

Произвольный файл, в котором встречаются одинаковые записи, можно представить в виде слова длины N:

$$R = (acbaabcdabda).$$

Допустим, что композиция этого слова равна

$$(m_1, m_2, \ldots, m_k).$$

Всего существует

$$N = n!/m_1! \ldots m_k!$$

слов, являющихся перестановками слова R, и только одно из них является отсортированным:

$$R_0 = (aaaaabbbccdd).$$

В этом слове все буквы лексикографически упорядочены.

Отсортировать слово R—это значит распознать подстановку σ, с помощью которой это слово переводится в слово R_0:

$$R_0 = \sigma R.$$

Как было показано в предыдущем пункте, для этого потребуется по крайней мере

$$\log N = I_0(R)$$

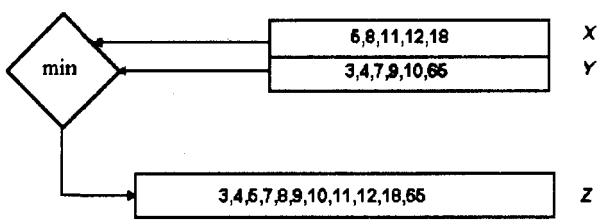
бинарных вопросов (сравнений). Таким образом, *сложность сортировки не может быть меньше, чем* $I_0(R)$. В том случае, когда все записи различны, имеем

$$I_0(R) = n \log n,$$

где n—длина файла.

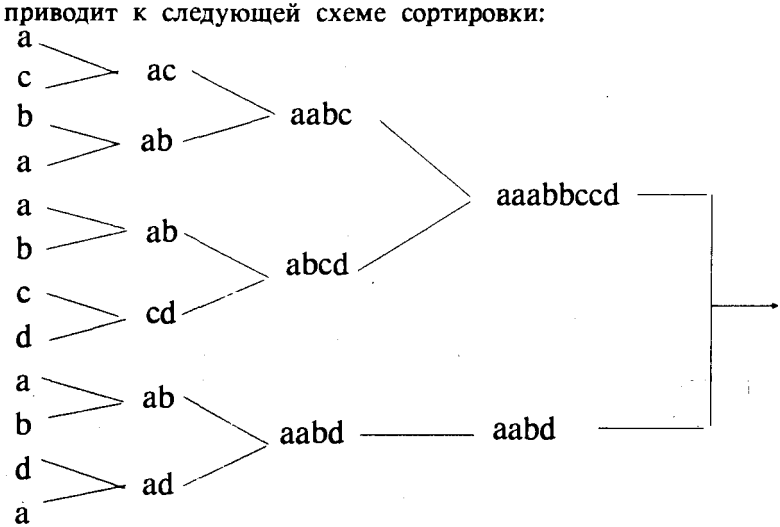
Существует простой метод сортировки, на котором достигается эта граница—сортировка слиянием.

Пусть заданы два отсортированных файла длины m и n соответственно:



Оба файла хранятся в регистре сдвига. Первые числа обоих регистров поступают на устройство сравнения. Минимальное из этих чисел вводится в регистр Z на выходе. Если это число было взято из регистра Y, то содержимое этого регистра сдвигается влево и сравнению подвергаются новые числа.

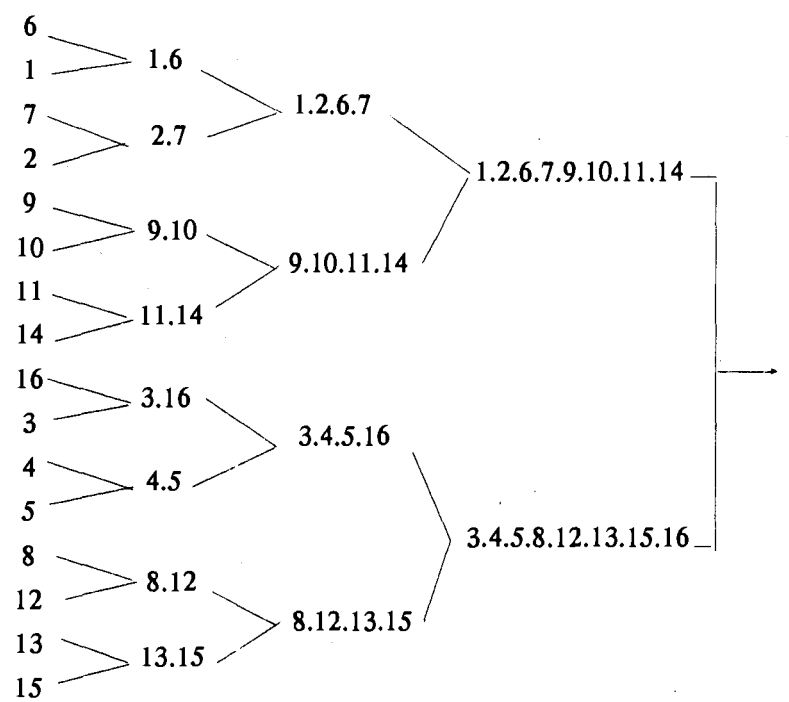
В результате после $m + n$ сравнений получаем отсортированный файл длины $m + n$. Последовательная реализация этой идеи приводит к следующей схеме сортировки:



Допустим для простоты, что $n = 2^k$. Исходный файл разбивается на $n/2$ отрезков длины 2. Каждый из отрезков сортируется; для этого потребуется $n/2$ сравнений. Затем образуется $n/4$ отрезков длины 4 путем слияния. Для этого потребуется еще $(n/4)\cdot 4 = n$ сравнений. Затем $n/8$ отрезков длины 8 потребуют еще n сравнений, и т. д. В результате получаем отсортированный файл после

$$n/2 + n(\log n - 1)$$

сравнений. Ниже показана сортировка слиянием 16 различных чисел:



4.7. Дерево поиска

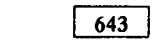
Бинарный поиск является самым быстрым методом поиска, но для его применения требуется, чтобы исходный файл был отсортирован.

В том случае, когда файл является динамическим, т.е. регулярно добавляются новые записи или стираются старые, частое применение сортировки снижает эффективность бинарного поиска. В этом случае более пригодной является иерархическая организация записей файла, при которой поиск осуществляется с помощью бинарного дерева.

Допустим, что записи с ключами K_i поступают в следующем порядке:

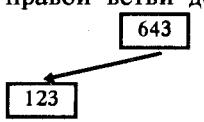
K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8
643	123	276	850	720	087	061	321

Первую запись K_1 объявляем корнем дерева:

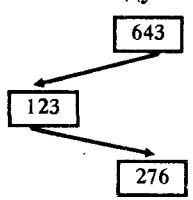


При поступлении записи с ключом K_2 производим сравнение $K_2 < K_1$?

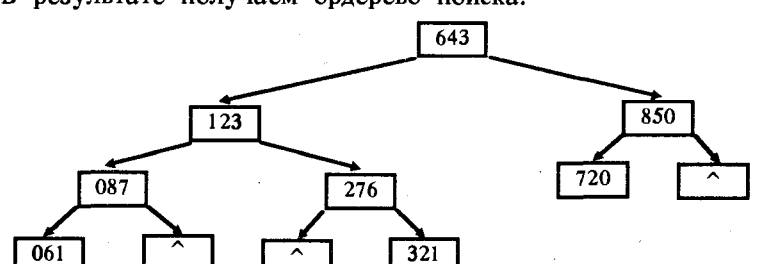
При положительном ответе помещаем запись K_2 в левой ветви, в противном случае—в правой ветви дерева:



Точно так же поступаем со следующей записью:



В результате получаем ордерево поиска:



Полученное дерево хранится в памяти компьютера в виде набора записей (узлов) *P*. Каждый узел *P* содержит следующие поля:

Address(*P*) —адрес в памяти;
Key (*P*) —ключ;
L(*P*) —указатель левого поддерева, равный адресу соответствующего узла;
R(*P*) —указатель правого поддерева;
Inform(*P*) —информация, содержащаяся в данной записи.

Корень дерева обозначается *Root*, а пустой узел—символом ^.

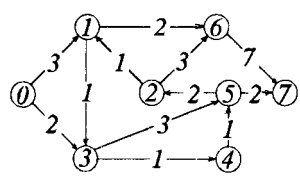
Например, возможна следующая таблица записей, образующих бинарное дерево:

Address	*Key*	*L(P)*	*R(P)*	*Inform*
0001	643	0010	0011	
0010	123	0101	0111	
0011	850	0110	^	
0101	087	1000	^	
1000	061	^	^	
0111	276	^	1001	
1001	321	^	^	
0110	720	^	^	

Поиск со вставкой по дереву осуществляется следующим способом. Допустим, что требуется найти запись с ключом $K = 276$. Сравниваем этот ключ с ключом, содержащимся в корне дерева: $K < Key\ (Root)$? В данном случае $276 < 643$, поэтому обращаемся к левому указателю $L(Root) = 0010$. Теперь сравниваем ключ с величиной $Key(0010)$: $276 < 123$? Неравенство не выполняется. Убедившись, что $276 \neq 123$, обращаемся к правому указателю $R(0010) = 0111$. Сравнение $K < Key(0111) = 276$ позволяет обнаружить нужный ключ. Если бы оказалось $R(0010) = \wedge$, это свидетельствовало бы о том, что записи с ключом K не существует, и можно было бы «подвесить» эту запись к нашему дереву (вставка записи на свободное место в памяти).

4.8. Кратчайший маршрут (алгоритм Дикстры)

Допустим, что в орграфе заданы расстояния $\rho(x, y)$ для каждого ребра (неотрицательные вещественные числа):



x	0	0	1	1	2	2	3	3	4	5	5	6	7
y	1	3	3	6	1	6	4	5	5	2	7	7	^
ρ	3	2	1	2	1	3	1	3	1	2	2	7	^

Требуется найти кратчайший маршрут $0 \rightarrow 7$, т.е. маршрут с минимальным суммарным расстоянием.

Алгоритм, который решает эту задачу, является модификацией алгоритма «Поиск пути» и заключается в расстановке меток. Каждая метка представляет собой пару $(s; d)$, где s—номер

вершины, из которой мы попадаем в данную вершину, а d—накопленное расстояние к данной вершине.

Начнем с объявления «ведущей» стартовой вершины маршрута. Затем находим вершины 1-го поколения:

		Метка	
		s	*d*
–	0		
	1	0	3
	2		
	3	0	2
	4		
	5		
	6		
	7		

(1)

Теперь «закрываем» вершину 0, приписывая ей слева знак «+».

Ведущая вершина назначается следующим образом. Просматривая все помеченные, но не закрытые вершины, находим вершину с наименьшим значением d (ею оказалась вершина 3). Эта вершина определяет 2-е поколение вершин: {4, 5}.

Метка $d(4)$ для вершины 4 получается теперь как сумма $d(3) + \rho(3, 4) = 2 + 1 = 3$:

		Метка	
		s	*d*
+	0		
	1	0	3
	2		
–	3	0	2
	4	3	3
	5	3	5
	6		
	7		

(2)

Закрываем вершину 3 и находим новую ведущую вершину. Ею может быть выбрана любая из двух вершин {1, 4}.

Допустим, что мы выбрали вершину 4:

		Метка	
		s	*d*
+	0		
	1	0	3
	2		
+	3	0	2
–	4	3	3
	5	3 4	5 4
	6		
	7		

(3)

У вершины 5 появляется, кроме старой (3, 5), новая метка (4, 4). Из этих двух меток оставляем одну—ту, у которой величина d меньше. У закрытых вершин меток не меняем.

Следующие итерации не требуют дополнительных пояснений:

		Метка	
		s	d
+	0		
−	1	0	3
	2		
+	3	0	2
+	4	3	3
	5	4	4
	6	1	5
	7		

(4)

		Метка	
		s	d
+	0		
+	1	0	3
	2	5	6
+	3	0	2
+	4	3	3
−	5	4	4
	6	1	5
	7	5	6

(5)

		Метка	
		s	d
+	0		
+	1	0	3
	2	5	6
+	3	0	2
+	4	3	3
+	5	4	4
−	6	1	5
	7	5	6

(6)

		Метка	
		s	d
+	0		
+	1	0	3
−	2	5	6
+	3	0	2
+	4	3	3
+	5	4	4
+	6	1	5
	7	5	6

(7)

		Метка	
		s	d
+	0		
+	1	0	3
+	2	5	6
+	3	0	2
+	4	3	3
+	5	4	4
+	6	1	5
+	7	5	6

(8)

Процесс заканчивается, когда все помеченные вершины оказываются закрытыми. Строим искомый маршрут:

Путь = (0, 3, 4, 5, 7) с длиной $d(7) = 6$.

Точно так же по заключительной таблице мы можем найти кратчайший путь до любой вершины, а соответствующая метка d равна длине этого пути. Например, найдем путь $0 \to 2$:

(0, 3, 4, 5, 2).

То, что алгоритм Дикстры действительно решает поставленную задачу, можно доказать следующим образом.

Пусть на некотором шаге алгоритма W обозначает множество закрытых вершин. Тогда для любой помеченной вершины x, как следует из описания алгоритма, метка $d(x)$ равна кратчайшему пути из 0 в x, в котором *все промежуточные вершины принадлежат* W. Выбираем теперь очередную ведущую вершину $x \notin W$. Допустим, что существует путь $0 \to x$, длина которого меньше, чем $d(x)$. Тогда этот путь должен проходить через последовательность вершин из W, затем через некоторую вершину $y \notin W$ и далее через некоторый отрезок $y \to x$. Должно, таким образом, выполняться неравенство

$$d(y) + \rho(y, x) < d(x).$$

Но это невозможно, поскольку $\rho(x, y) \geq 0$ и $d(x) \leq d(y)$ по определению ведущей вершины.

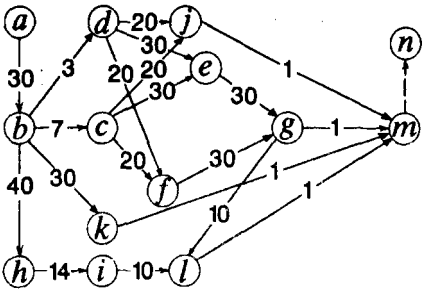
Таким образом, когда мы закрываем очередную вершину x, мы уже знаем кратчайший маршрут $0 \to x$ (его длина равна $d(x)$, и эта метка не может измениться в дальнейшем).

4.9. Максимальный маршрут. Сетевая модель комплекса операций.

При планировании работ обычно составляется перечень необходимых операций с указанием продолжительности и непосредственно предшествующих операций. Можно представить себе, например, комплекс мероприятий, который выполняет режиссер любительской студии для постановки спектакля:

Номер	Наименование операции	Продолжительность (в днях)	Непосредственно предшествующие операции
a	Выбор пьесы	30	-
b	Чтение пьесы в студии	1	*a*
c	Распределение ролей	7	*b*
d	Размножение пьесы	3	*b*
e	Репетиции с главными героями	30	*c, d*
f	Репетиции с другими участниками	20	*c, d*
g	Совместные репетиции	30	*e, f*
h	Закупка декораций	40	*b*
i	Установка декораций	14	*h*
j	Заказ костюмов	20	*c, d*
k	Подготовка музыкального оформления	30	*b*
l	Работа с осветителями	10	*i, g*
m	Генеральная репетиция	1	*g, j, k, l*
n	Премьера спектакля	1	*n*

Составляем граф, отражающий взаимозависимость операций:



Поскольку операция не может начаться, пока не будут выполнены все предшествующие операции, общее время, необходимое для выполнения всей работы, определяется максимальным маршрутом в графе (он называется *критическим путем*).

Граф не является ордеревом: в некоторые вершины могут входить несколько дуг. В тоже время в графе отсутствуют циклы: для любых двух операций всегда можно однозначно решить, какие из них предшествуют другой (или же они независимы).

Отсутствие циклов приводит к тому, что в графе (и в любом его подграфе) найдется хоть одна корневая вершина. Поэтому для нахождения максимального маршрута можно применить алгоритм Дикстры со следующими модификациями:

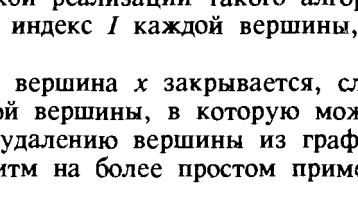
a) в качестве ведущей выбирать корневую вершину подграфа, который получается после удаления всех закрытых к этому времени вершин вместе с их ребрами.

b) когда у какой-либо вершины появляются две метки (s_1, d_1) и (s_2, d_2), то оставлять метку с большим значением d.

При практической реализации такого алгоритма удобно вначале подсчитывать индекс I каждой вершины, равный числу входящих дуг.

После того как вершина x закрывается, следует уменьшить на 1 индекс каждой вершины, в которую можно попасть из x. Это соответствует удалению вершины из графа.

Поясним алгоритм на более простом примере:



1) Ведущая вершина *a*

		Индекс	*s*	*d*
–	*a*	0		
	b	1	*a*	10
	c	0	*a*	20
	d	2		
	e	2		
	f	2		

2) Ведущая вершина *c*

		Индекс	*s*	*d*
+	*a*	0		
	b	0	*a c*	30
–	*c*	0	*a*	20
	d	2		
	e	1	*c*	22
	f	2		

3) Ведущая вершина *b*

		Индекс	*s*	*d*
+	*a*	0		
–	*b*	0	*c*	30
+	*c*	0	*a*	20
	d	1	*b*	55
	e	0	*b*	35
	f	2		

4) Ведущая вершина *e*

		Индекс	*s*	*d*
+	*a*	0		
+	*b*	0	*c*	30
+	*c*	0	*a*	20
	d	0	*b*	55
–	*e*	0	*b*	35
	f	1	*e*	51

5) Ведущая вершина *d*

		Индекс	*s*	*d*
+	*a*	0		
+	*b*	0	*c*	30
+	*c*	0	*a*	20
–	*d*	0	*b*	55
+	*e*	0	*b*	35
	f	0	*d*	58

6) Ведущая вершина f

		Индекс	s	d
+	a	0		
+	b	0	c	30
+	c	0	a	20
+	d	0	b	55
+	e	0	b	35
+	f	0	d	58

Теперь все вершины закрыты и в окончательной таблице величина $d(x)$ равна максимальной длине пути $a \to x$.

Критический путь $a \to f$ имеет вид

$$\text{Путь} = (a, c, b, d, f),$$

а длина его равна 58.

То, что описанный алгоритм решает поставленную задачу, можно доказать следующим образом.

Пусть W—множество уже закрытых вершин на некотором шаге алгоритма. Тогда для любой помеченной вершины x метка $d(x)$ равна максимальной длине пути из a в x, в котором *все промежуточные вершины принадлежат* W. Выбираем теперь очередную ведущую вершину $x \notin W$. Ее индекс равен 0, и таковым он стал благодаря удалению некоторых вершин из множества W.

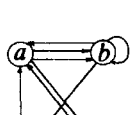
Таким образом в вершину x можно попасть лишь из некоторой вершины $y \in W$. В свою очередь вершина y была когда-то ведущей, и в нее можно попасть только из множества W. Таким образом: любой путь, ведущий в x, состоит из вершин множества W. Поэтому $d(x)$ является максимальной длиной по всем путям $a \to x$.

4.10. Эйлеров граф

Мультиграф называется *эйлеровым*, если существует замкнутый путь, проходящий через каждое ребро ровно один раз. Эйлеров граф можно нарисовать, не отрывая карандаш от бумаги. Можно представить себе также путешественника, который отправляется в путь, начиная с некоторой вершины, по ребрам (мостам) графа, имеющим одностороннее движение (даже для пешеходов). После того, как очередной мост пройден, он взрывается. Если граф эйлеров, то существует такой вариант обхода, при котором в конце путешествия путник оказывается в начальной вершине, а все мосты уничтожены. Соответствующий обход называется *эйлеровым контуром.*

Пусть x—некоторое слово, Tx—его циклический сдвиг. Пара слов $x \otimes Tx$, очевидно, соответствует некоторому эйлерову графу:

$$\begin{matrix} x \\ Tx \end{matrix}\begin{pmatrix} abadabbcdc \\ badabbcdca \end{pmatrix},$$



Каждое ребро графа соответствует паре соседних букв слова x, так что слово x и является эйлеровым контуром.

С другой стороны, если задан граф, то мы превращаем его в пару слов $x_0 \otimes y_0$, где слово x_0 отсортировано:

$$\begin{matrix} x_0 \\ y_0 \end{matrix}\begin{pmatrix} aaabbbccdd \\ bbdabcadac \end{pmatrix}.$$

Кроме того, мы знаем переходную матрицу:

	a	b	c	d	
a		2		1	3
b	1	1	1		3
c	1			1	2
d	1		1		2
	3	3	2	2	10

В то же время совсем не ясно, существует ли такое слово x, что $x_0 \otimes y_0$ и $x \otimes Tx$ определяют один и тот же граф. Одно несомненно:

Граф $x_0 \otimes y_0$ является эйлеровым тогда и только тогда, когда

$$\sigma(x_0 \otimes y_0) = x \otimes Tx$$

для некоторого слова x *и некоторой подстановки* σ.

Отсюда заключаем:

Если граф $x_0 \otimes y_0$ эйлеров, то он является связным графом и слова x_0 и y_0 имеют одну и ту же композицию.

Это значит, что в переходной матрице суммарный столбец совпадает с суммарной строкой (в нашем примере 3, 3, 2, 2).

Оказывается, что эти условия являются также достаточными:

Связный граф $x_0 \otimes y_0$, *у которого* x_0 и y_0 *имеют одну и ту же композицию, является эйлеровым.*

Для доказательства мы должны построить эйлеров путь, т.е. найти такое слово x, что $\sigma(x_0 \otimes y_0) = x \otimes Tx$.

Рассмотрим отображение

$$\varphi\colon x \otimes Tx \to x_0 \otimes y_0,$$

которое заключается в том, что сортируются столбцы из $x \otimes Tx$ по верхней букве, причем если у двух столбцов верхние буквы совпадают, то их положения не меняются:

$$\begin{matrix} x \\ Tx \end{matrix}\begin{pmatrix} abadabbcdc \\ badabbcdca \end{pmatrix} \xrightarrow{\varphi} \begin{pmatrix} aaabbbccdd \\ bdbabcdaac \end{pmatrix}\begin{matrix} x_0 \\ y_0 \end{matrix}.$$

Выделим слева для каждой буквы, отличной от начальной, тот столбец, который соответствует последнему вхождению этой буквы в слово x:

$$D = \begin{pmatrix} bcd \\ cac \end{pmatrix}.$$

Очевидно, если проделать такую же операцию справа, то получится тот же набор столбцов D.

В графе D число ребер на 1 меньше числа вершин, поэтому он является деревом. В то же время начальная буква a отсутствует в верхней строке, следовательно, из нее не выходит ни одного ребра. Таким образом, *D является деревом, входящим в начальную вершину a.*

Если мы хотим построить отображение φ^{-1}, обратное к отображению φ, то мы должны, очевидно, возвращаться от $(x_0 \otimes y_0)$ к $(x \otimes Tx)$ следующим образом:

$$\begin{matrix} x_0 \\ y_0 \end{matrix}\begin{pmatrix} aaabbbccdd \\ bdbabcdaac \end{pmatrix} \xrightarrow{\varphi^{-1}} \begin{pmatrix} ab... \\ ba... \end{pmatrix}.$$

Выделяем первый столбец $\begin{pmatrix} a \\ b \end{pmatrix}$ и вычеркиваем его. Нижняя буква b указывает на то, что нужно найти первый столбец с буквой b вверху. Вычеркиваем этот столбец. Теперь очередь за столбцом $\begin{pmatrix} a \\ d \end{pmatrix}$ и т. д.

Применим теперь отображение φ^{-1} к произвольному графу $x_0 \otimes y_0$, где x_0 и y_0 имеют одну и ту же композицию, а x_0 отсортировано. После очередного вычеркивания для всех букв, кроме буквы a, выполняется свойство: число вхождений буквы в нижнее слово на 1 меньше числа вхождений этой буквы в верхнее слово. Поэтому процесс вычеркивания не может прерваться из-за того, что не хватит некоторой буквы в верхнем слове.

Иначе обстоит дело с буквой a: число вхождений этой буквы в верхнее слово на каждом шаге алгоритма на 1 меньше числа вхождений в нижнее слово. Поэтому может оказаться, что внизу появился запрос на эту букву, а ее лимит вверху уже исчерпан:

$$\begin{pmatrix} aaabbbccdd \\ bdbabcadac \end{pmatrix} \rightarrow (abadabc).$$
1352 67 4

Здесь дальнейшее продолжение невозможно: требуется буква a вверху, а все такие буквы уже вычеркнуты.

Такая ситуация не может возникнуть, когда D является деревом, входящим в вершину a. Действительно, допустим, что еще не вычеркнут некоторый столбец из D с нижней буквой, отличной от a, скажем, $\binom{b}{c}$. Это значит, что вверху еще присутствует буква c, а значит еще не вычеркнут столбец из D с буквой c вверху. Допустим, что этот столбец имеет вид $\binom{c}{d}$. Заключаем, что еще не вычеркнут столбец из D с буквой d вверху. Поскольку все пути в графе D ведут к вершине a, найдется невычеркнутый столбец из D с буквой a внизу, скажем, $\binom{d}{a}$.

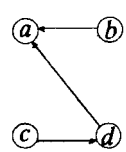
Таким образом, когда D является деревом, входящим в a, столбец из D с буквой a внизу вычеркивается последним, когда не останется ни одного столбца в исходном графе $x_0 \otimes y_0$.

Отсюда возникает следующий *алгоритм построения эйлерова контура*:

1) Найти остовное дерево D, входящее в a, где a—произвольная вершина, выбранная в качестве начальной. Поскольку граф связен, такое дерево всегда существует и для его построения можно использовать алгоритм, описанный в п. 4.4.

2) Для фиксированного отсортированного слова x_0 построить нижнее слово y_0, имеющее ту же композицию, что и x_0. Для этого буквы слова y_0, соответствующие последним вхождениям каждой буквы в слово x_0, заполняем в соответствии с деревом D. Остальные буквы слова y_0 заполняем произвольным образом в соответствии с переходной матрицей графа.

3) Найти эйлеров контур, применяя алгоритм вычеркивания. В нашем примере остовное дерево можно выбрать следующим образом:



$$D = \binom{bcd}{ada}.$$

Вписываем это дерево в $x_0 \otimes y_0$:

$$\begin{matrix} x_0 \\ y_0 \end{matrix}\begin{pmatrix} aaabbbccdd \\ \quad a\ d\ a \end{pmatrix}.$$

Заполняем остальные места слова y_0:

$$\begin{matrix} x_0 \\ y_0 \end{matrix}\begin{pmatrix} aaabbbccdd \\ bbdcbaadca \end{pmatrix}.$$

Применяем алгоритм вычеркивания:

$$\begin{pmatrix} a\,a\,a\,b\,b\,b\,c\,c\,d\,d \\ b\,b\,d\,c\,b\,a\,a\,d\,c\,a \\ 1\,4\,7\,2\,5\,6\,3\,9\,8\,10 \end{pmatrix} \rightarrow (abcabbadcd).$$

4.11. Максимальный поток в сети

Под *сетью* будем понимать орграф, в котором каждой дуге ставится в соответствие неотрицательное целое число $c(u, v)$, называемое *пропускной способностью дуги*. В сети выделяются две вершины, которые называются *источником* s и *стоком* t. В источник не входит ни одна дуга, а из стока не выходят дуги.

Говорят, что задан поток из s в t в сети, если каждой дуге поставлено в соответствие целое число $f(u, v)$ со следующими свойствами:

1) Для всех дуг (u, v) $0 \leq f(u, v) \leq c(u, v)$.

2) Для каждой вершины $x \neq s, t$ $\sum\limits_{u \to x} f(u, x) = \sum\limits_{x \to v} f(x, v)$.

Здесь $\sum\limits_{u \to x}$ означает суммирование по всем вершинам, из которых можно попасть в x. Соответственно, $\sum\limits_{x \to v}$ означает суммирование по всем вершинам, в которые можно попасть из x.

Таким образом, второе условие представляет собой «закон сохранения» для каждой вершины, отличной от s и t.

Соединим теперь s и t фиктивной дугой (t, s) и зададим значение $f(t, s)$ таким образом, чтобы закон сохранения выполнялся и для вершин s и t:

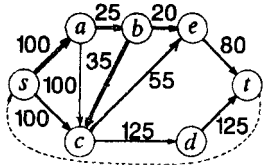
$$f(t, s) = \sum_{s \to v} f(s, v) = \sum_{u \to t} f(u, t);$$

$f(t, s)$ называется *величиной потока из s в t*.

Требуется найти поток максимальной величины.

Такая задача возникает при пересылке товаров по железной дороге (без хранения их на промежуточных станциях), при управлении потоками автомобилей в сети автострад, при проектировании нефтепроводов, при передаче информации в информационных сетях и т. д.

Рассмотрим в качестве примера задачу транспортировки некоторого продукта из пункта s в пункт t по сети, состоящей из железных дорог и автострад штриховой кривой:



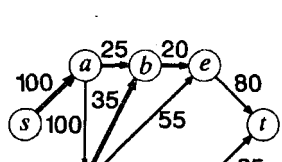
Здесь указаны пропускные способности дуг, а пунктиром указана фиктивная дуга. Ее пропускная способность предполагается неограниченной.

Когда определен некоторый поток в сети, величина $f(u, v)$ может трактоваться как кратность соответствующей дуги, а с учетом фиктивной дуги законы сохранения превращают сеть в эйлеров мультиграф. Поэтому задача о максимальном потоке трансформируется в задачу отыскания эйлерова подграфа с максимальным значением $f(t, s)$. Соответствующий эйлеров контур можно найти, проходя из s в t по некоторому маршруту, разрушая пройденные «мосты». Затем можно вернуться в s по фиктивной дуге и выбрать новый маршрут по изменившейся сети. Поскольку мы хотим найти максимальный контур, разрушения каждый раз должны быть минимальны, так что следует выбирать маршрут с наименьшим количеством промежуточных вершин. Таким образом, алгоритм состоит из многократного применения алгоритма «Поиск пути».

1-й маршрут:

s			
a	s		
b		a	
c	s		
d		c	
e		c	
t			d

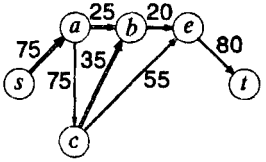
$\begin{pmatrix} 100 & 125 & 125 \\ s, & c, & d, & t \end{pmatrix}$, $f_1 = 100$.



2-й маршрут:

s				
a	s			
b		a		
c		a		
d			c	
e			b	
t				d

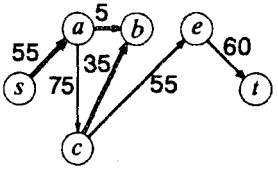
$\begin{pmatrix} 100 & 100 & 25 & 25 \\ s, & a, & c, & d, & t \end{pmatrix}$, $f_2 = 25$.



3-й маршрут:

s				
a	s			
b		a		
c		a		
d				
e			b	
t				e

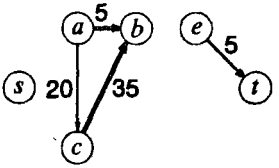
$\begin{pmatrix} 75 & 25 & 20 & 80 \\ s, & a, & b, & e, & t \end{pmatrix}$, $f_3 = 20$.



4-й маршрут:

s				
a	s			
b		a		
c		a		
d				
e			c	
t				e

$\begin{pmatrix} 55 & 75 & 55 & 60 \\ s, & a, & c, & e, & t \end{pmatrix}$; $f_4 = 55$.



Теперь пути из s в t не существует. Максимальный поток равен $f_1 + f_2 + f_3 + f_4 = 100 + 25 + 20 + 55 = 200$.

Объединяем все маршруты в переходной матрице, вписывая туда для всех дуг i-го маршрута значения f_j. Если для разных маршрутов i и j встречаются одинаковые дуги, то соответствующие значения f_i и f_j суммируются:

	s	a	b	c	d	e	t	
s		100		100				200
a			20	80				100
b						20		20
c					125	55		180
d							125	125
e							75	75
t	200							200
	200	100	20	180	125	75	200	

При правильном проведении вычислений суммарная строка должна совпадать с суммарным столбцом, поскольку мы получили эйлеров мультиграф.

5. ПАМЯТЬ В СЛОВАХ

5.1. 1-информация слова

В любом слове x присутствует некоторая память, которая проявляется в том, что появление буквы на некоторой позиции зависит от того, какая буква занимает предшествующую позицию. Обозначим, как и ранее, через Tx циклический сдвиг слова x. *Назовем величину* $I_1(x) = I_0(x/Tx)$ 1-*информацией слова* x.

Ясно, что $I_1(x) \le I_0(x)$, а равенство $I_0(x) = I_1(x)$ равносильно отсутствию памяти в слове, поскольку слово x в этом случае не зависит от своего сдвига, так что появление определенной буквы на некоторой позиции совершенно не влияет на следующую букву. Рассмотрим, например, слово

$$x = (010011110010).$$

Поскольку число нулей здесь совпадает с числом единиц, то

$$I_0(x) = H(x) = 12H(1/2) = 12 \text{ бит}.$$

Чтобы вычислить $I_1(x)$, составляем переходную матрицу $x \to Tx$:

	0	1	
0	3	3	6
1	3	3	6
	6	6	12

Мы видим, что $I_1(x) = I_0(Tx/x) = 12$ бит. Таким образом, в этом слове отсутствует память на один шаг (соседние буквы независимы).

Наоборот, в слове

$$x = (01010101),$$

обладающем регулярной структурой, память очень сильна: появление буквы 0 означает, что следующей буквой станет обязательно 1. В переходной матрице $x \to Tx$ это отражается следующим образом:

	0	1	
0	0	4	4
1	4	0	4
	4	4	8

Мы видим, что $I_1(x) = 0$ (функциональная зависимость $x \to Tx$). В то же время $I_0(x)$ максимально: $I_0(x) = 8$ бит.

В гл. 1 мы видели, что $I_0(x)$ равно логарифму числа слов 0-орбиты слова x. 0-орбита состоит из всех слов с одной и той же композицией.

По аналогии назовем 1-*орбитой слова* x множество слов x' с такой же композицией $x' \otimes Tx'$, что и композиция $x \otimes Tx$. Иначе говоря, два слова x и x' принадлежат одной и той же 1-орбите тогда и только тогда, когда переходная матрица $x \to Tx$ совпадает с переходной матрицей $x' \to Tx'$. Ясно, что 1-орбита является подмножеством 0-орбиты, так что 0-орбита разбивается на непересекающиеся 1-орбиты.

Т е о р е м а 5.1. *Логарифм числа* $N^{(1)}$ *слов* 1-*орбиты асимптотически совпадает с* 1-*информацией:*

$$\log N^{(1)} = I_1(x) - o(n),$$

$$| o(n) | \leq (q-1) \log n,$$

где q—размер алфавита.

Д о к а з а т е л ь с т в о . Отсортируем пару $x \otimes Tx$, применяя отображение φ, введенное в п. 4.10:

$$x \otimes Tx \xrightarrow{\varphi} x_0 \otimes y_0.$$

Если в слове $x_0 \otimes y_0$ фиксировать последний столбец для каждой буквы, кроме начальной (фиксировать дерево D), а затем подействовать на оставшиеся позиции слова y_0 подстановкой из G_x, то получим новое слово y_0', которому соответствует при алгоритме вычеркивания (отображение φ^{-1}) эйлеров контур, т.е. слово x' из 1-орбиты слова x. Допустим, что слово x имеет композицию $(m_1, \ldots, m_q)$, а переходная матрица $x \to Tx$ имеет вид $\|m_{ij}\|$, где $\sum_j m_{ij} = m_i$. Если удалить из пары $x \otimes Tx$ дерево D, то остается матрица $\|m_{ij}'\|$, где

$$\sum_j m_{ij}' = m_i' = m_i - 1, \qquad m_{ij}' = m_{ij}$$

за исключением одного индекса $j = k$, для которого

$$m_{ik}' = m_{ik} - 1.$$

Число различных слов x' из 1-орбиты, которые мы можем

получить описанным способом, равно

$$\prod_i \frac{m_i'!}{m_{i1}'!\ldots m_{iq}'!} = \prod_i \frac{(m_i-1)!}{m_{i1}!\ldots(m_{ik}-1)!\ldots m_{iq}!} \geq$$

$$\geq \frac{1}{m_1\ldots m_{q-1}} \prod_i \frac{m_i!}{m_{i1}!\ldots m_{iq}!}.$$

Таким образом,

$$\log N^{(1)} \geq I_0(x/Tx) - \sum_{i=1}^{q-1} \log m_i \geq I_1(x) - (q-1)\log n.$$

С другой стороны, если в $x_0 \otimes y_0$ не фиксировать дерево D, а подействовать на y_0 подстановкой из G_{x_0}, то получим все слова условной орбиты слова y_0. Логарифм мощности этой орбиты равен

$$I_0(y_0/x_0) = I_0(x/Tx) = I_1(x).$$

Таким образом,

$$\log N^{(1)} \leq I_1(x).$$

Следовательно,

$$I_1(x) - (q-1)\log n \leq \log N^{(1)} \leq I_1(x).$$

5.2. 1-сжатие по Фитингофу

В п. 1.10 был описан способ сжатия слов (назовем его 0-*сжатием*), при котором длина кодового слова асимптотически равна его 0-информации. Этот способ эффективен в том случае, когда буквы в слове распределены неравномерно. Для слова

$$x = (01010101)$$

этот способ не дает эффекта сжатия, поскольку $I_0(x) = n$. В то же время сильная зависимость на один шаг между буквами этого слова наводит на мысль применить следующий способ сжатия (назовем его 1-*сжатием*): кодовое слово состоит из двух частей: префикса фиксированной длины, в котором записывается номер 1-орбиты, и переменной части, в которой записывается номер слова внутри 1-орбиты.

Каждая 1-орбита полностью определяется переходной матрицей $x \to Tx$. Для ее записи потребуется место для q^2 чисел m_{ij}. Поскольку каждое из чисел m_{ij} потребует не больше $\log n$ бит, всего для префикса нужно будет отвести $q^2 \log(n)$ бит. Для

записи же номера слова внутри 1-орбиты потребуется не больше $I_1(x)$ бит вследствие теоремы 5.1. Таким образом, справедлив следующий результат:

Т е о р е м а 5.2. *Существует префиксный код, у которого длина кодового слова асимптотически равна* 1-*информации слова*:

$$l(x) = I_1(x) + o(n),$$

где

$$| o(n) | \leq q^2 \log n.$$

Практически сжатие удобно осуществлять сведением к 0-сжатию. Допустим, что

$$x = (acabcabbc).$$

Используя отображение φ, получаем пару слов $x_0 \otimes y_0$:

$$\begin{matrix} x \\ Tx \end{matrix} \begin{pmatrix} acabcabbc \\ cabcabbca \end{pmatrix} \to \begin{pmatrix} aaabbbccc \\ cbbcbcaaa \end{pmatrix} \begin{matrix} x_0 \\ y_0 \end{matrix}.$$

Сначала кодируем ту часть $y_0^{(1)}$ слова y_0, которая соответствует фиксированной букве a вверху, т.е. записываем в сжатой форме слово (cbb). Для этого применяем 0-сжатие, так что кодовое слово состоит из двух частей: префикса длины $q \log n$, в котором записываем числа $m_{11}, m_{12}, \ldots, m_{1q}$. Затем идет переменная часть длины $I_0\left(y_0^{(1)}\right)$. Далее записываем в аналогичной форме ту часть $y_0^{(2)}$ слова y_0, которая соответствует фиксированной букве b вверху, и т.д. В результате получаем переменную часть длины

$$I_0\left(y_0^{(1)}\right) + I_0\left(y_0^{(2)}\right) + \ldots = I_0(y_0/x_0) = I_1(x).$$

5.3. m-информация слова

В слове может отсутствовать память на один шаг, но пара соседних букв может однозначно определять следующую букву. Введем понятие 2-*информации слова* следующим образом:

$$I_2(x) = I_0(x/Tx \otimes T^2x),$$

где T^kx означает сдвиг на k шагов. Более общим образом, m-*информацией слова* назовем величину

$$I_m(x) = I_0(x/Tx \otimes T^2x \otimes \ldots \otimes T^mx).$$

Вычислим для примера $I_2(x)$ для слова

$$x = (010011110010).$$

Переходная матрица $Tx \otimes T^2x \to x$ имеет вид

	0	1	
00		3	3
01	2	1	3
10	3		3
11	1	2	3
	6	6	12

$$I_2(x) = I_0(x/Tx \otimes T^2x) = 3H\left(\frac{1}{3}\right) + 3H\left(\frac{1}{3}\right) = 2 \cdot 3 \cdot 0.92 = 5.52 \text{ бит}.$$

Каждой паре букв (диграмме) присвоим новое обозначение:

$$00 — A$$

$$01 — B$$

$$10 — C$$

$$11 — D$$

Тогда слову x ставится в соответствие новое слово из большего алфавита:

$$X = (BCABDDDCABCA).$$

Чтобы по слову X восстановить слово x, нужно каждую букву слова X выписать в виде столбца:

$$X = \begin{pmatrix} 010011110010 \\ 100111100100 \end{pmatrix} = x \otimes Tx.$$

Верхняя строка совпадает со словом x. Очевидно,

$$I_1(X) = I_0(X/TX) = I_0(x \otimes Tx/Tx \otimes T^2x) = I_0(x/Tx \otimes T^2x) = I_2(x).$$

Таким образом, понятие 2-информации слова x сводится к 1-информации слова X.

Например, переходная матрица $X \to TX$ имеет вид

	A	B	C	D	
A		3			3
B			2	1	3
C	3				3
D			1	2	3
	3	3	3	3	12

$$I_1(X) = 6H\left(\frac{1}{3}\right).$$

Точно так же, когда речь идет о вычислении m-информации, можно по слову m из q-ичного алфавита построить слово X в

q^m-ичном алфавите, считая каждый блок из m соседних букв новой буквой. Тогда

$$I_m(x) = I_1(X).$$

1-орбиту слова X назовем в этом случае *m-орбитой слова x.*

Из теорем 5.1 и 5.2 получаем:

Теорема 5.3. *Логарифм числа слов m-орбиты асимптотически равен m-информации слова:*

$$\log N^{(m)} = I_m(x) - o(n),$$

$$|\, o(n) \,| \leq q^m \log n.$$

Существует префиксный код, у которого длина кодового слова асимптотически совпадает с m-информацией слова:

$$l(x) = I_m(x) + o(n),$$

$$|\, o(n) \,| \leq q^{m+1} \log n.$$

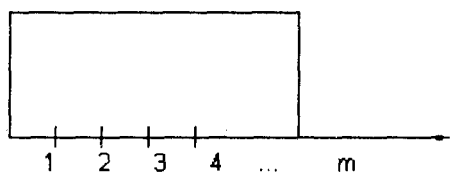
5.4. Информационная характеристика слова

Для каждого слова x можно составить невозрастающую цепочку величин:

$$I_0(x) \geq I_1(x) \geq I_2(x) \geq \ldots$$

Эту последовательность назовем *информационной характеристикой слова.* Последовательность обычно стабилизируется при некотором значении m. При этом величина $I_m(x)$ характеризует ту степень сжатия, которая может быть достигнута с помощью симметрий слова (симметрий в расположении букв).

Существуют двоичные последовательности с идеальной информационной характеристикой:



У этой последовательности

$$I_0(x) = I_1(x) = \ldots = I_m(x),$$

что означает следующее:

1) Число единиц равно числу нулей.

2) Каждая пара двоичных символов одинаковое число раз встречается в слове.

3) Каждая группа из $(m+1)$ символов одинаковое число раз встречается в слове.

Таким образом, m-информация слова максимальна и практически отсутствует память. Кроме того, в слове наблюдается минимальное количество симметрий. Отсутствие симметрий обычно воспринимается как случайность, поэтому последовательности с описанной идеальной информационной характеристикой называют *псевдослучайными*.

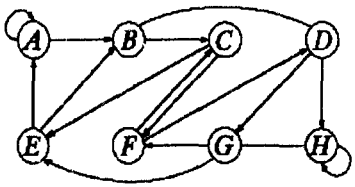
Принцип построения таких последовательностей поясним на примере $m = 3$, $n = 16$. Обозначим:

$$\begin{array}{ll} 000 - A & 100 - E \\ 001 - B & 101 - F \\ 010 - C & 110 - G \\ 011 - D & 111 - H \end{array}$$

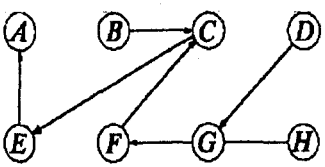
Мы хотим построить последовательность X этих букв, такую, что переходная матрица $X \to TX$ имеет следующий вид:

	A	B	C	D	E	F	G	H	
A	1	1							2
B			1	1					2
C					1	1			2
D							1	1	2
E	1	1							2
F			1	1					2
G					1	1			2
H							1	1	2
	2	2	2	2	2	2	2	2	16

Соответствующий эйлеров граф можно представить как



Найдем остовное дерево, входящее в A:



Вписываем это дерево в $x_0 \otimes y_0$:

$$\begin{array}{l} x_0 \\ y_0 \end{array} \begin{pmatrix} AA\ BB\ CC\ DD\ EE\ FF\ GG\ HH \\ C\ \ E\ \ G\ \ A\ \ C\ \ F\ \ G \end{pmatrix}.$$

Заполняем остальные буквы слова y_0 и применяем алгоритм вычеркивания:

$$\begin{matrix} x_0 & A & A & B & B & C & C & D & D & E & E & F & F & G & G & H & H \\ y_0 & A & B & D & C & F & E & H & G & B & A & D & C & E & F & H & G \\ & 1 & 2 & 3 & 9 & 10 & 15 & 4 & 12 & 8 & 16 & 11 & 14 & 7 & 13 & 5 & 6 \end{matrix} \rightarrow$$

$$\rightarrow (AABDHHGEBCFDGFCE).$$

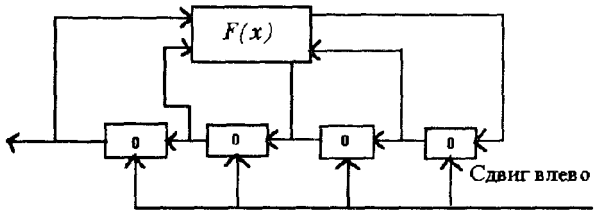
Выписываем теперь каждую букву в виде двоичного столбца:

$$\left(\begin{matrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ \hline A & A & B & D & H & H & G & E & B & C & F & D & G & F & C & E \end{matrix}\right).$$

Верхняя строка определяет нужную нам псевдослучайную последовательность:

$$x = (0000111100101101).$$

Эта последовательность порождается нелинейным регистром сдвига:



В зависимости от состояния регистра (x_1, x_2, x_3, x_4) формируется значение некоторой булевой функции $F(x_1, x_2, x_3, x_4)$, которое определяет следующее состояние:

$$0000 \rightarrow 0001$$
$$0001 \rightarrow 0011$$
$$\ldots\ldots\ldots$$

5.5. Генетическая информация

Наследственная информация (генетический материал) у всех растений, животных, микроорганизмов и большинства вирусов представляется словами большой длины в четырехбуквенном алфавите:

$$ATGGTTCGTTCTTAT\ldots$$

Каждое такое слово представляет собой полимерную молекулу ДНК, в состав которой входят четыре основания (нуклеотида):

A – аденин C – цитозин
G – гуанин T – тимин

Существуют особые водородные связи $A \leftrightarrow T$ и $G \leftrightarrow C$ (комплементарность). Такое попарное сопоставление нуклеотидов было выведено с помощью молекулярных моделей, на которых точно выдерживались все межатомные расстояния.

Природная молекула ДНК представляет собой две полимерные цепи попарно соединенных нуклеотидов, закрученные в форме двойной спирали (модель Уотсона—Крика):

$$\begin{pmatrix} ATGGTTCGT \ldots \\ TACCAAGCA \ldots \end{pmatrix}.$$

Если отождествить ДНК с внешней памятью компьютера, например с магнитной лентой, то можно сказать, что информация записывается избыточно, так что к исходному слову приписывается его шаблон в дополнительном коде.

Введем в рассмотрение конечное поле $F_4 = (0, 1, \alpha, \beta)$, в котором операции производятся по модулю неприводимого многочлена $x^2 + x + 1$, так что $\alpha + \beta = 1$, $\alpha\beta = 1$. Поэтому можно сопоставить нуклеотиды и элементы поля F_4:

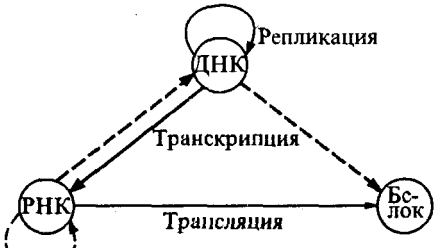
$$\begin{pmatrix} 0, 1, \alpha, \beta \\ A, T, G, C \end{pmatrix},$$

так что дополнительный нуклеотид определяется по правилу $x \to 1 + x$. Таким образом, $A \to A + 1 = T$, $G \to G + 1 = C$.

Очень длинные молекулы ДНК упакованы в клетке в небольшом объеме. Общая длина ДНК хромосом человека (около 10^9 нуклеотидов) упакована в ядре диаметром меньше микрометра. Это означает, что в клетке ДНК компактизована.

Определенные отрезки (подслова) слова ДНК, несущие смысловую нагрузку, называются *генами*. Каждый ген кодирует информацию, ответственную за создание определенного белка. Средняя длина гена у человека порядка 10^4 нуклеотидов (букв). Таким образом, у человека присутствует порядка 10^5 различных белков.

Пути переноса информации, закодированной в ДНК, обобщены Ф. Криком в виде *центральной догмы молекулярной биологии*:



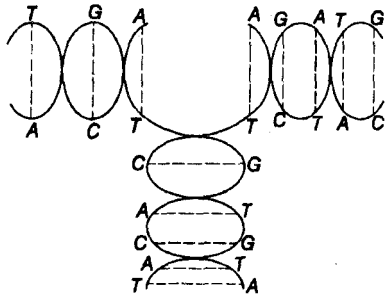
Сплошные стрелки обозначают здесь три основных великих канала передачи биологической информации: *репликация, транскрипция и трансляция.*

Продолжая аналогию с компьютером, в которой ДНК отождествляется с внешней памятью, можно сказать, что репликация соответствует копированию (размножению) магнитных лент. Транскрипция соответствует записи в оперативную память (РНК), а трансляция—обработке информации (построению белков).

Возможны и особые случаи переноса информации (на схеме они представлены штриховыми линиями. Так, наблюдалась репликация РНК и обратная транскрипция от РНК к ДНК. Далее, информация может непосредственно преобразовываться из ДНК в белок, минуя РНК (это наблюдалось в случае антибиотиков). «Запрещен» перенос информации от белков обратно к нуклеиновым кислотам. Это означает, что модификация белков (генных продуктов) не наследуется.

Способность ДНК к самовоспроизведению (репликации) лежит в основе размножения живых организмов. Модель двойной спирали позволила понять принцип удвоения ДНК. Представлялось логичным, что при удвоении ДНК две хромосомные нити «расплетаются», а затем каждая из них служит базисом, на котором выстраивается комлементарная ей новая цепь. В результате образуются две дочерние хромосомы (реплики), каждая из которых содержит по одной родительской хромосоме. Экспериментально показано, что именно так, по *полуконсервативному механизму*, происходит репликация ДНК:

Удивительная простота такой модели скрывает сложнейшие



биохимические процессы, в которых участвует множество белков (ферментов). Эти белки, как и все другие, закодированы в последовательности нуклеотидов ДНК. В процессе эволюции сформировался сложный репликационный аппарат, обеспечивающий максимальную точность передачи информации от родительских к дочерним молекулам ДНК. По существующим оценкам ошибки репликации, приводящие к появлению неправильного нуклеотида происходят с частотой порядка одной на

10^8-10^{10} нуклеотидов. В то же время синтез ДНК происходит с очень высокой скоростью—около 1000 нуклеотидов в секунду. Вся ДНК бактерий (порядка 10^7 нуклеотидов) воспроизводится за 20 минут.

Стабильность генетического материала объясняется существованием в клетках всех живых организмов специальных систем *репарации* (обнаружения и исправления ошибок). Каждый день появляются все новые данные, свидетельствующие о большом разнообразии систем репарации ДНК.

Перейдем теперь к описанию процесса транскрипции. При этом молекула ДНК превращается в молекулу РНК, в которой присутствуют также четыре основания. Основное отличие РНК от ДНК заключается в том, что вместо тимина *T* появляется *урацил U* , так что при транскрипции происходит взаимно однозначное преобразование слов:

ATGGTTCGTTC ...

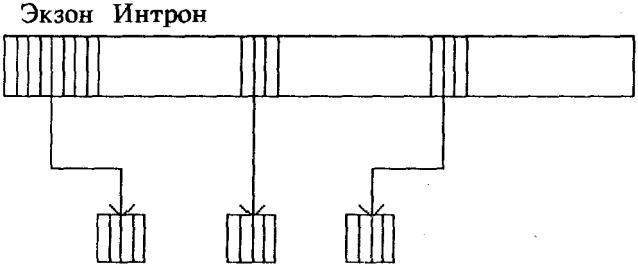
↓

AUGGUUCGUUC ...

Существует следующее объяснение такого преобразования.

Аденин *A* и гуанин *G* относятся к классу пуриновых оснований, а тимин и цитозин—к классу пиримидиновых оснований. В любом живом организме часто и спонтанно происходит *апуринизация*, т.е. уменьшение доли аденина и гуанина. Каждая клетка человеческого организма ежедневно теряет около 5000 пуринов. В то же время пиримидины *T* и *C* более устойчивы. Кроме того, постоянно происходит превращение одного кислотного основания в другое. При *дезаминировании* цитозин *C* превращается в урацил *U*, аденин *A* в гипоксантин, а гуанин—в ксантин. Чаще всего происходит превращение $C \to U$ (в каждой человеческой клетке за день происходит около 100 таких событий). Если бы для хранения генетической информации вместо ДНК использовалась бы РНК, т.е. набор букв $\{A, U, C, G\}$, то ошибочные переходы $C \to U$ невозможно было бы обнаружить. Если же выбрать алфавит $\{A, T, C, G\}$, то при дезаминировании всех оснований возникают основания, не характерные для ДНК. Это обстоятельство позволяет репаративной системе клетки узнавать продукт дезаминирования и удалять его. Можно сказать, что РНК является более естественной. Именно эта кислота участвует в основном процессе—в формировании белков. В то же время она неустойчива, поэтому заменяется на ДНК во «внешней памяти».

Вначале считали, что преобразованием $T \rightarrow U$ ограничивается весь процесс транскрипции (это справедливо для простейших организмов). Затем была обнаружена *мозаичная структура гена.* У некоторых организмов каждый ген разделен на чередующиеся участки: информационные слова (*экзоны*) и вставки (*интроны*). В процессе транскрипции все интроны вырезаются, а оставшиеся экзоны "сшиваются" (этот процесс называется *сплайсингом*):

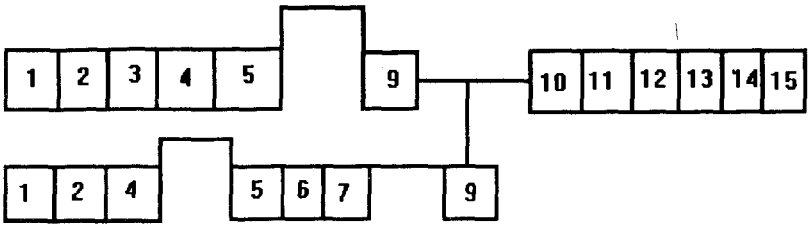


Число интронов может сильно варьироваться: от одного в гене дрожжей до нескольких десятков (17 в гене птиц и 50 в гене млекопитающих). Интроны могут составлять большую часть мозаичного гена. Например, в гене птиц экзоны составляют лишь восьмую часть общей длины гена, а в гене млекопитающих— лишь двадцатую.

Таким образом, ДНК содержит огромную избыточность: лишь небольшая часть букв используется для формирования белков. По-видимому, такая избыточность нужна для обнаружения и исправления ошибок, т.е. интроны участвуют в системе репарации. Все это напоминает запись на магнитной ленте, когда после информационного файла добавляется контрольная сумма.

В настоящее время известно, что в интронах могут находиться важные элементы регуляции транскрипции—*усилители*, или *энхансеры.* Эти элементы были выявлены в опытах с искусственными генными конструкциями, составленными из отрезков ДНК разного происхождения. Энхансеры представлены короткими отрезками ДНК (десятки букв). Извлеченные из вирусных генов и включенные в состав искусственных генетических конструкций, они увеличивали эффективность транскрипции в десятки и сотни раз. Особенность энхансеров состоит в том, что они способны действовать на больших расстояниях (более чем 1000 букв) и вне зависимости от ориентации по отношению к направлению транскрипции гена. Наконец, в 1987 г. был обнаружен так называемый *альтернативный сплайсинг*: различные отрезки ДНК могут несколькими способами сшиваться в РНК, образуя разные модификации (изотипы)

одного и того же белка. Разные типы образуются в разных тканях на определенных стадиях развития:



Таким образом, один ген может кодировать несколько различных белков.

Синтез молекул РНК начинается в определенных местах ДНК, называемыми *промоторами*, и завершается в *терминаторах*. Участок ДНК, ограниченный промотором и терминатором, представляет собой единицу транскрипции—*оперон*. В пределах каждого оперона копируется только одна из двух нитей ДНК, которая называется *значащей* или *матричной*. Во всех оперонах, считываемых в одном направлении, значащей является одна нить ДНК. В оперонах, считываемых в противоположном направлении, значащей является другая нить ДНК. Соседние опероны могут быть отделены друг от друга нетранскрибируемыми участками ДНК, а могут и перекрываться. Разбиение ДНК на множество оперонов обеспечивает возможность независимого считывания разных генов, их индивидуального включения и выключения. Транскрипция осуществляется специальным ферментом—*РНК—полимеразой*.

5.6. Генетический код

После того как на стадии транскрипции сформирована цепочка букв РНК:

AUGGUUCGUUC ...,

наступает этап превращения этого слова в последовательность белков (стадия трансляции).

Белки играют важнейшую роль в жизнедеятельности любых организмов. Некоторые белки являются *ферментами*, т.е. катализаторами биохимических реакций в живых организмах. Без участия ферментов подобные реакции не происходят или протекают слишком медленно. Другие белки (*структурные*) выполняют в организме роль строительных блоков. Некоторые белки, такие, например, как гемоглобин, участвуют в системе запасания и транспорта кислорода.

Белки—это большие полимерные молекулы, построенные из аминокислотных звеньев. В состав белков входят 20 различных видов аминокислот: Глицин (*Gly*), Аланин (*Ala*), Валин (*Val*) и

др. Таким образом, каждый белок представляет собой слово в 20-буквенном алфавите:

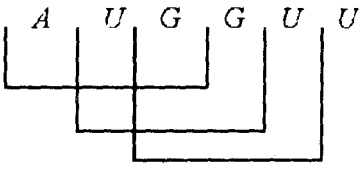
$$Met - Gln - Arg - Tyr - Glu - Ser - Leu - \ldots$$

В процессе трансляции происходит перевод четырех буквенного слова в 20-буквенное слово:

$$\underset{\downarrow}{AUG}\ \underset{\downarrow}{GUU}\ \underset{\downarrow}{CGU}\ \ldots$$
$$Met\quad Val\quad Arg$$

Сразу после того, как была признана модель строения ДНК, многие исследователи сосредоточили свое внимание на установлении истинной природы генетического кода. Было ясно, что, поскольку в состав белков входят 20 различных аминокислот, каждый кодон (набор букв, кодирующий аминокислоту), должен состоять не менее чем из трех нуклеотидов. Дублеты могли бы образовать не более 16 различных кодонов, в то время как на основе триплетов можно составить до 64 различных кодонов.

Можно представить себе несколько способов построения триплетного кода. Код мог бы быть *перекрывающимся*:



Возникла красивая гипотеза *кода без запятой*, обеспечивающего синхронизацию нуклеотидной последовательности (Ф. Крик).

С. Голомб предложил следующую формализацию.

Пусть A^n обозначает множество слов длины n в алфавите A из q букв. Рассмотрим два слова $x, y \in A^n$:

$$x = (a_1, \ldots, a_n),$$

$$y = (b_1, \ldots, b_n).$$

Соединяя эти слова последовательно:

$$a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n,$$

получаем $n - 1$ новых слов:

$$(a_2, \ldots, a_n,\ \ b_1),$$

$$(a_3, \ldots, b_1,\ \ b_2),$$

$$\ldots\ldots\ldots$$

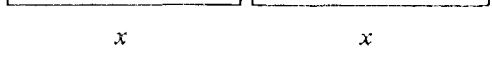
$$(a_n, \ldots, b_{n-2},\ \ b_{n-1}),$$

которые называются *перекрытиями* слов x и y.

Подмножество $D \subseteq A^n$ называют *кодом без запятой*, если для любых двух кодовых слов $x, y \in D$ все перекрытия не являются кодовыми словами. Такой код обеспечивает синхронизацию в длинной цепочке кодовых слов. Действительно, выберем некоторую букву такой цепочки в качестве стартовой позиции, отсчитаем n букв, в результате получим некоторое слово из A^n. Если мы попали в перекрытие, то это слово не будет кодовым (предполагается, что перед нашими глазами, точнее, перед глазами рибосомы, осуществляющей трансляцию, выписан весь кодовый словарь D). Сдвигаемся влево на одну позицию и повторяем процедуру до тех пор, пока не получим кодовое слово. Тем самым будет установлена правильная стартовая позиция.

Какова может быть максимальная мощность кода $|D|$ (обозначим ее $W_n(q)$)?

Простая оценка сверху может быть получена из рассмотрения периода слова. Пусть $T^k x$ обозначает, как и прежде, циклический сдвиг на k шагов. Минимальное число k, при котором $T^k x = x$, назовем *периодом* d слова x. Ясно, что $d \leq n$ и d является делителем числа n. Слова максимального периода $d = n$ назовем *основными*. Код без запятой может содержать лишь основные слова. Действительно, пусть x—слово периода $d < n$. Выписывая подряд это слово два раза, получаем перекрытие, совпадающее со словом x:



Если обозначить через $P_n(q)$ число основных слов из q букв, то получим оценку

$$W_n(q) \leq \frac{1}{n} P_n(q)$$

(циклический сдвиг кодового слова не может быть кодовым словом). Очевидно,

$$\sum_{d|n} P_d(q) = q^n.$$

Из этого равенства можно получить выражение для $P_d(q)$, используя формулу обращения Мёбиуса.

Пусть $f(n)$, $F(n)$— две целочисленные функции натурального аргумента, причем

$$F(n) = \sum_{d|n} f(d).$$

Тогда

$$f(n) = \sum_{d \mid n} \mu(d)\, F(n/d),$$

где функция Мёбиуса определяется следующим образом:

$$\mu(d) = \begin{cases} 1, & \text{если } d = 1, \\ 0, & \text{если } d \text{ делится на квадрат}, \\ (-1)^r, & \text{если } d = p_1, \ldots, p_r. \end{cases}$$

Отсюда получаем

$$P_n(q) = \sum_{d \mid n} \mu(d)\, q^{n/d},$$

так что

$$W_n(q) \leq \frac{1}{n} \sum_{d \mid n} \mu(d)\, q^{n/d}.$$

Подставляя $n = 3$, $q = 4$, получаем

$$W_3(4) \leq \frac{1}{3}\left(\mu(1) \cdot 4^3 + \mu(3) \cdot 4\right) = \frac{1}{3}(4^3 - 4) = 20.$$

Оказывается, при $n = 3$ и любом q полученная оценка для $W_n(q)$ является достижимой: следует выбирать кодовые слова (abc) по правилу $a < b \geq c$. Действительно, рассмотрим пару кодовых слов: *abcdef*. Тут выполняются неравенства

$$a < b \geq c,$$

$$d < e \geq f.$$

Первое перекрытие *bcd* не является кодовым словом, потому что $b \geq c$. Второе перекрытие *cde* не является кодовым словом, потому что $d < e$. Таким образом, мы действительно получили код без запятой. Период слова может быть равен или 1, или 3. Основные слова—это слова, в которых встречаются хотя бы две разные буквы.

Легко показать, что сдвигом основного слова всегда можно получить слово, удовлетворяющее условию $a < b \geq c$. В то же время слова вида (aaa), $bbb)$, ... не входят в код. Поэтому мощность кода в точности равна $(q^3 - q)/3$.

Таким образом, из 64 возможных слов можно было бы создать код из 20 слов, обеспечивающий синхронизацию. Однако природа не пошла по этому пути и, как всегда, оказалась права: ошибка хотя бы в одном символе (нуклеотиде) приводила бы к нарушению синхронизации и вела бы к летальному для организма исходу.

В результате тончайших биохимических экспериментов, проведенных Ф. Криком, М. Ниренбергом, С. Очоа, Г. Кораной

было установлено, что генетический код является неперекрывающимся, и был установлен смысл каждого кодона. Расшифровка генетического кода, полностью завершенная в 1966 г., явилась величайшим научным достижением. Окончательная таблица выглядит следующим образом:

	U	*C*	*A*	*G*	
U	*UUU* *UUC* } *Phe* *UUA* *UUG* } *Leu*	*UCU* *UCC* *UCA* *UCG* } *Ser*	*UAU* *UAC* } *Tyr* *UAA* } *Stop* *UAG* } *Stop*	*UGU* *UGC* } *Cys* *UGA Stop* *UGG Trp*	*U* *C* *A* *G*
C	*CUU* *CUC* *CUA* *CUG* } *Leu*	*CCU* *CCC* *CCA* *CCG* } *Pro*	*CAU* *CAC* } *His* *CAA* *CAG* } *Gln*	*CGU* *CGC* *CGA* *CGG* } *Arg*	*U* *C* *A* *G*
A	*AUU* *AUC* *AUA* } *Ile* *AUG Met*	*ACU* *ACC* *ACA* *ACG* } *Thr*	*AAU* *AAC* } *Asn* *AAA* *AAG* } *Lys*	*AGU* *AGC* } *Ser* *AGA* *AGG* } *Arg*	*U* *C* *A* *G*
G	*GUU* *GUC* *GUA* *GUG* } *Val*	*GCU* *GCC* *GCA* *GCG* } *Ala*	*GAU* *GAC* } *Asp* *GAA* *GAG* } *Glu*	*GGU* *GGC* *GGA* *GGG* } *Gly*	*U* *C* *A* *G*

Как следует из таблицы, код является вырожденным, т.е. существуют слова-синонимы: *GUU = GUC = Val*, *CGG = AGA = Arg* и т. д. Имеются три кодона: *UAA*, *UAG*, *UGA*, которые не несут смысловой нагрузки (non-sence). Они являются терминаторными кодонами: появление такого кодона в слове означает конец трансляции. Если такой кодон возникает в результате изменения какой-либо буквы смыслового кодона (мутации), это приводит обычно к летальному исходу.

Пожалуй, самой впечатляющей особенностью генетического кода является его *универсальность*. Приведенную таблицу можно с одинаковым успехом применить для расшифровки РНК человека, птицы, бактерии или табака. Из этого правила, как выяснилось в 1979 г., есть исключения: генетический код митохондрий (особых органов клетки) характеризуется отличными от обычных значениями некоторых кодонов и некоторыми особыми правилами узнавания кодонов.

Трансляцию осуществляет особый орган клетки—рибосома. Синхронизация (установка рамки считывания) осуществляется с помощью префикса, *AGGAGGU*, который называется *последовательностью Шайн-Долгарно*. Искажение этой последовательности приводит, конечно, к катастрофическим последствиям. Однако эта последовательность присутствует в слове в единственном числе. Поэтому вероятность искажения символов в определенных семи позициях достаточно мала.

5.7. Хромосомная база данных

Каждый ген кодирует некоторый белок, который в свою очередь определяет некоторый простой признак организма. При описании базы данных в гл. 3 каждому признаку ставился в соответствие домен признака—то множество значений, которое может принимать данный признак. В биологии понятие домена эквивалентно понятию *аллели*. Каждый простой признак (ген) может существовать в одной или нескольких альтернативных формах (аллелях). Например, в каждом растении гороха существует ген, влияющий на форму семян: семена могут быть гладкими или морщинистыми. Известно большое число множественного аллелизма. Классический пример—ген, определяющий окраску меха кролика. Здесь возможны по крайней мере четыре аллеля: шиншиловый, дикий тип, гималайский, альбинос.

Для хранения генетической информации во всех живых организмах используется реляционная база данных.

Каждое отношение $R(X_1, \ldots, X_n)$ состоит из двух строк (гомологичных хромосом). Хромосома содержит гены $X_1, \ldots, X_n$, которые образуют сложный признак X. Пара хромосом образуется при размножении: одна хромосома получается от отца, другая—от матери (*диплоидная пара*). У гомологичных хромосом все гены совпадают по своей функции, но могут отличаться несколькими нуклеотидами. Часто эти изменения вызваны нежелательной мутацией и проявляются в форме наследственного заболевания. Например, если в гене человека, кодирующем гемоглобин, заменить букву T на букву A в одной позиции, то возникает альтернативная форма гемоглобина, ведущая к так называемой серповидной анемии. Если значения признака совпадают в обеих строках, то особь называется *гомозиготной* по данному гену. В противном случае говорят о *гетерозиготности*.

Таким образом, гомозиготность характеризуется диплоидными парами $\binom{A}{A}$, $\binom{a}{a}$, а гетерозиготность—парами $\binom{A}{a}$, $\binom{a}{A}$. Число хромосом у любого организма равно $2n$, где n—*гаплоидное число*. Например, у человека $n = 23$. Это означает, что база данных состоит из 23 отношений, которые соединяются между собой в процессе, предшествующем размножению особи. У комнатной мухи $n = 6$, у краба $n = 127$. Наименьшее число $n = 1$ наблюдается у лошадиной аскариды (Ascaris megalocephata). Число хромосом остается постоянным для каждого вида. Под видом понимают обычно совокупность организмов, которые взаимодействуют между собой в процессе размножения.

Существуют два типа деления клеток: один для образования *соматических клеток* (или клеток тела) и другой для образования половых клеток (*гамет*). Соматические клетки образуются

в результате митоза и необходимы для нормального роста (развития) организма. При митозе все хромосомы удваиваются перед началом деления клетки (репликация ДНК) и распределяются поровну между двумя дочерними клетками. В результате все соматические клетки организма обладают одинаковым набором хромосом $2n$.

Гаметы нужны только для продолжения рода. Они образуются в процессе *мейоза*. При этом каждая клетка делится дважды, а число хромосом удваивается лишь один раз.

Вместо одной диплоидной пары $\binom{A}{a}$ образуется четыре гомологичных хромосомы, A, A, a, a, и они распределяются поровну между четырьмя образовавшимися гаметами. Каждая гамета получает также одну из хромосом B, B, b, b, соответствующих другому сложному признаку. Согласно Менделю такое распределение хромосом происходит независимо как между четырьмя гаметами, так и между разными признаками.

Другими словами, в процессе мейоза реализуется оператор СОЕДИНЕНИЕ для всех отношений, образующих реляционную базу данных генов, т.е. образуется прямое произведение всех отношений. Например, для трех хромосомных пар, X, Y, Z, получаем следующее отношение $R(X, Y, Z)$:

$$R(X): \binom{A}{a},$$

$$R(Y): \binom{B}{b},$$

$$R(Z): \binom{C}{c},$$

$R(X, Y, Z)$:

X	Y	Z
A	B	C
A	B	c
A	b	C
A	b	c
a	B	C
a	B	c
a	b	C
a	b	c

Здесь X обозначает сложный признак, включающий в себя все гены, расположенные в одной хромосоме. Они образуют так называемую *группу сцепления*.

Имеются четыре гаметы. Для каждой из них выбирается некоторая строка из этой таблицы. В результате гамета оказыва-

ется укомплектованной полным набором n негомологичных хромосом. Пара гамет (одна мужская половая клетка и одна женская) сливаются в процессе оплодотворения. Образующаяся при этом клетка имеет полный диплоидный набор $2n$. Возможно, например, такое сочетание:

$$\begin{pmatrix} A\ b\ C \\ a\ B\ C \end{pmatrix}.$$

При расшифровке генетического кода в процессе трансляции участвуют обе гомологичные хромосомы. Обычно один признак является *доминантным*, а другой *рецессивным*. Это означает следующее. Если ДНК одной хромосомы указывает на то, что нужно строить белок типа A, а другая хромосома—что нужно его строить в форме a, то предпочтение отдается доминантной аллели A. При этом аллель a блокируется. Существуют также *кодоминантные* аллели—в этом случае строится как белок A, так и белок a. Например, существуют четыре группы крови: 0, A, B и AB. Они определяются тремя аллелями одного гена: I^A, I^B и i. Аллели I^A и I^B доминанты по отношению к аллелю i, но кодоминанты по отношению друг к другу. Возможны следующие варианты:

$$\begin{pmatrix} A \\ A \end{pmatrix},\ \begin{pmatrix} A \\ B \end{pmatrix},\ \begin{pmatrix} A \\ i \end{pmatrix},\ \begin{pmatrix} B \\ A \end{pmatrix},\ \begin{pmatrix} B \\ B \end{pmatrix},\ \begin{pmatrix} B \\ i \end{pmatrix},\ \begin{pmatrix} i \\ A \end{pmatrix},\ \begin{pmatrix} i \\ B \end{pmatrix},\ \begin{pmatrix} i \\ i \end{pmatrix}.$$

Наборы

$$\begin{pmatrix} A \\ A \end{pmatrix},\ \begin{pmatrix} A \\ i \end{pmatrix},\ \begin{pmatrix} i \\ A \end{pmatrix}$$

отождествляются и образуют один генотип (группу крови) A. Точно так же отождествляются наборы

$$\begin{pmatrix} B \\ B \end{pmatrix},\ \begin{pmatrix} B \\ i \end{pmatrix},\ \begin{pmatrix} i \\ B \end{pmatrix}$$

(группа крови B). Варианты

$$\begin{pmatrix} A \\ B \end{pmatrix},\ \begin{pmatrix} B \\ A \end{pmatrix}$$

образуют генотип AB. Наконец, набор $\begin{pmatrix} i \\ i \end{pmatrix}$ соответствует группе крови 0.

Наиболее богатая генетическая база данных у гетерозигот.

Например, в случае гомозигот получаем

$$R(x): \begin{pmatrix} A \\ A \end{pmatrix},$$

$$R(Y): \begin{pmatrix} B \\ B \end{pmatrix},$$

$$R(Z): \begin{pmatrix} C \\ c \end{pmatrix},$$

$R(X, Y, Z)$:

X	Y	Z
A	B	C
A	B	c

Существует несколько копий этих строк, так что каждая гамета укомплектована, но число различных генотипов равно двум. Некоторое разнообразие генотипов достигается с помощью *кроссинговера.* В стадии мейоза гомологичные хромосомы конъюгируют, так что имеются два-три участка контакта (хиазмы), на которых происходит обмен генами. В результате сложный признак оказывается разделенным на несколько признаков.

ИСТОРИЧЕСКИЙ ОЧЕРК

Пионерской работой по теории информации является статья Р. Хартли 1928 г. В этой статье впервые была предложена логарифмическая мера информации.

Широкий интерес к этой области возник после появления статьи К. Шеннона 1948 г. [20], в которой была отчетливо сформулирована проблематика этой теории, связанная с кодированием сообщений. К. Шеннон рассматривает множество слов A^n вместе с распределением вероятностей $p(x)$. Величину

$$-\log p(x) = S(x)$$

он называет собственной информацией слова, а среднюю величину

$$H(x) = -\sum p(x) \log p(x)$$

энтропией, или информацией источника. В терминах $H(x)$ формулируется и доказывается теорема кодирования в отсутствие шума. Критерием качества кода является средняя длина кодовых слов

$$\bar{l} = \frac{1}{n} \sum p(x)\, l(x).$$

Для префиксного кода всегда

$$\bar{l} \geq H(x)$$

(следствие неравенства между средним арифметическим и средним геометрическим), и можно как угодно близко подойти к этой границе, устремляя $n \to \infty$. Для этого нужно упорядочить все слова по собственной информации:

$$S(x_1) \geq S(x_2) \geq \ldots \geq S(x_N),$$

и наименее вероятному слову приписывать слово из префиксного кода, имеющее наибольшую длину. Такой принцип применялся ранее в коде Морзе, а после работы Шеннона стало возможным сравнивать подобные коды по их эффективности. В частности, оказалось, что код Морзе близок к оптимальному.

Были предложены регулярные методы построения оптимальных кодов (код Шеннона—Фано, код Хаффмена). В этих методах предполагается, что вероятности сообщений (слов) известны точно. Если статистика не точна или же меняется в процессе передачи, то методы приводят к кодам, далеким от оптимальных.

Кодовые слова определяются вероятностной структурой, поэтому кодирование связано с экспоненциальным объемом вычислений.

Когда речь идет об эффективном кодировании отдельных букв с учетом неравномерной частоты их появления, упомянутые методы приводят к очень быстрым и красивым алгоритмам. На практике обычно ставится задача сжатия слова большой длины, причем в исходной постановке ничего не говорится о какой-либо вероятности.

Б.М. Фитингоф [16] обнаружил существование методов кодирования источника, устойчивых по отношению к распределению вероятностей. Он предложил упорядочить слова по величине $H(x)$, названной им квазиэнтропией. Эта величина применялась ранее как выборочная энтропия (статистическая оценка для энтропии). Б.М. Фитингоф впервые применил ее как вес, по которому следует упорядочить сообщения. Если слову с наименьшей энтропией приписать кодовое слово наименьшей длины, то получится код, оптимальный для целого класса вероятностных распределений.

При кодировании по методу Фитингофа вовсе не требуется знание статистики сообщений. А если распределение вероятностей меняется, то код по-прежнему остается оптимальным. Энтропия слова $H(x)$ или $I_0(x)$ выполняет при этом ту же функцию, что и собственная информация по Шеннону. В своей следующей статье Фитингоф назвал эту величину информацией слова. Задача кодирования источника была сведена им к нумерации слов одной и той же орбиты.

Реализация этого метода с использованием треугольника Паскаля принадлежит В.Ф. Бабкину. В результате был получен метод кодирования источника полиномиальной трудоемкости. Поскольку упорядоченность по $I_0(x)$ заменяет упорядоченность по вероятности, возникает мысль применить I_0 для целей декодирования в канале с шумом.

В шенноновской теории декодирование определяется в терминах $p(x)$. Декодирование максимального правдоподобия заключается в том, что слово y декодируется в такое слово x на входе, для которого апостериорная вероятность

$$p(x/y) = p(x, y)/p(y)$$

максимальна (байесовский подход). Декодер при этом настраивается на вероятности, которые известны неточно и могут меняться в процессе передачи.

Абонентам кажется, что декодирование осуществляется оптимальным способом, а на самом деле ввиду изменившейся статистики эффективность декодирования давно уже не поддается контролю. Если же для целей декодирования использовать метрику $I_0(x/y)$, то получается декодирование, оптимальное для

целого класса распределений вероятностей (устойчивое декодирование) [7].

Возникла мысль ввести вместо вероятностного канала с шумом более простую модель—алгебраический канал [7]. Если научиться строить оптимальный код для алгебраического канала, то тем самым будет решена задача построения оптимального кода в вероятностной модели (конструктивное доказательство теоремы кодирования Шеннона).

Впрочем, надобность в вероятностной модели отпадает, поскольку теория информации оказывается достаточно интересной и богатой приложениями в алгебраической постановке. Одним из таких приложений является распознавание образов.

Описанный в гл. 2 метод распознавания является развитием хорошо известного подхода М.М. Бонгарда [4]. Когда М.М. Бонгард создавал свой алгоритм, не было известно, как вычислять информацию одного признака (слова), содержащуюся в другом признаке. Алгебраический подход к теории информации дает естественное решение двух ключевых проблем распознавания образов—определения информативности признаков и кластерного анализа.

Программная реализация описанного метода распознавания была выполнена И.С. Борейко. Ряд специалистов в Московском геологоразведочном институте применили этот метод распознавания для решения различных геологических задач [12, 15, 16].

Реляционная модель базы данных была первоначально изложена в статье Е.Ф. Кодда [21]. Он обратил внимание на важность понятия функциональной зависимости для целей нормализации (декомпозиции). Правила вывода для F-зависимостей были предложены К. Делобелем и Кейси, их полноту доказал Армстронг.

Многозначные зависимости были введены Р. Фейджином. Наиболее полное изложение теории реляционных баз данных содержится в книге Д. Мейера [13] и К. Делобеля и М. Адиба [22].

Для математического описания реляционных баз данных обычно используется язык математической логики, причем создается своя «реляционная алгебра». В то же время речь идет в этой теории об устранении избыточности, так что напрашивалось обращение к теории информации. Традиционная вероятностная теория информации оказывалась при этом далекой от запросов информатики.

Иначе обстоит дело с алгебраической теорией информации. В рамках этой теории основные понятия теории реляционных баз данных получают естественное и простое объяснение. Функциональная зависимость оказывается эквивалентной обращению в нуль условной информации (энтропии). Многозначные зависимости получают прозрачное истолкование в терминах условной независимости слов.

Связь между словами и графами прослеживается во многих работах по теории графов, например в книге К. Бержа [2]. При описании задач поиска и сортировки мы следуем изложению Д. Кнута [10].

Задача построения эйлеровых контуров с использованием остовного дерева была решена Н. де Брейном и др. [18]. При изложении этих результатов в гл. 4 использовались идеи П. Кастелейна и Д. Фоата. При изложении основных алгоритмов на графах мы следуем книге В. Липского [11].

Гл. 5 книги, в которой изучается память слова с помощью циклического сдвига, основана на статье [8].

Мы включили в эту книгу также основные сведения о том, как хранится и обрабатывается в живых организмах генетическая информация с помощью слов, "буквами" которых являются сложные полимерные молекулы. При этом мы следуем изложению Ф. Айалы и Дж. Кайгера [1] и книге С.Г. Инге-Вечтомова [9].

СОДЕРЖАНИЕ

СПИСОК ЛИТЕРАТУРЫ

1. *Айала Ф., Кайгер Дж.* Современная генетика. В 3 т.—М.: Мир, 1987.
2. *Берж К.* Теория графов и ее применение.—М.: ИЛ, 1962.
3. *Бойко В.В., Савинков В.М.* Проектирование баз данных для информационных систем.—М.: Финансы и статистика, 1989.
4. *Бонгард М.М.* Проблема узнавания.—М.: Наука, 1967.
5. *Борейко И.С., Гоппа В.Д., Городничев Е.Н.* Алгебраико-информационный подход к распознаванию образов // Труды Международной конференции по автоматизации в геологии.—М., 1990.
6. *Васильев А.Г.* Анализ геохимических факторов модели распознавания образов.—М.: МГРИ, 1987.
7. *Гоппа В.Д.* Информация слов // Пробл. передачи информ.—1978.—Т. 14, № 3.—С. 3—17.
8. *Гоппа В.Д.* Коды и информация // УМН.—1984.—Т. 39, № 1—С. 77—120.
9. *Инге-Вечтомов С.Г.* Генетика с основами селекции.—М.: Высшая школа, 1989.
10. *Кнут Д.* Искусство программирования. Т. 3. Сортировка и поиск.— М.: Мир, 1978.
11. *Липский В.* Комбинаторика для программистов.— М.: Мир, 1988.
12. *Лисенков А.Б.* Выявление ведущих факторов формирования подземных вод.— М.: МГРИ, 1987.
13. *Мейер Д.* Теория реляционных баз данных.— М.: Мир. 1986.
14. *Розенблатт Ф.* Принципы нейродинамики. Персептроны и теория механизмов мозга.— М.: Мир, 1965.
15. *Сальников А.С.* Распознавание золотоносных месторождений.— М.: МГРИ, 1988.
16. *Фитингоф Б.М.* Оптимальное кодирование при меняющейся и неизвестной статистике сообщений // Пробл. передачи информ.—1966.—Т. 2, № 2.—С. 3—11.
17. *Фитингоф Б.М.* Сжатие дискретной информации // Пробл. передачи информ.—1967.—Т. 3, № 3.—С. 26—36.
18. *Харари Ф.* Теория графов.—М.: Мир, 1973.
19. *Харченков А.Г.* Принципы и методы прогнозирования минеральных ресурсов.— М.: Недра, 1987.
20. *Шеннон К.* Работы по теории информации и кибернетике.— М.: ИЛ, 1963.
21. *Codd E.F.* Relational Model of Data for Large Shared Data Banks // CACM.—1970.—13 : 6, June.—P. 377—387.
22. *Delobel C., Adiba M.* Relational Database Systems.—Benjamin, 1985 (translation of: Bases de Donnees et Systemes Relationnels.—Paris: Bordos, 1983).