Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

О. В. Герман, Н. Н. Дорожкина

ТЕОРИЯ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ И СИСТЕМ

**Тексты лекций
для студентов специальности 1-40 01 02-03
«Информационные системы и технологии
(издательско-полиграфический комплекс)»**

Минск 2007

УДК 004.4(075.8)
ББК 22.18.7я7
Г 1

Рассмотрены и рекомендованы к изданию редакционно-издательским советом университета

Рецензенты:
доцент кафедры информационных систем
и технологий БГУИР кандидат технических наук
А. М. Севернев;
доцент кафедры информатики и высшей
математики МГЭИ кандидат экономических наук
Ж. М. Анисимова

Герман, О. Г.

Г 1 Теория информационных процессов и систем : тексты лекций для студентов специальности 1-40 01 02-03 «Информационные системы и технологии (издательско-полиграфи-ческий комплекс)» / О. В. Герман, Н. Н. Дорожкина. – Минск : БГТУ, 2007. – 222 с.

ISBN 978-985-434-728-8

В текстах лекций рассматриваются базовые информационные процессы и системы: представление и хранение информации, поиск, сжатие, восстановление, шифрование, обработка и фильтрация информации, СУБД, клиент-серверные системы, системы диагностики и решения задач, формализованных на основе логических уравнений.

УДК 004.4(075.8)
ББК 22.18.7я7

ISBN 978-985-434-728-8

ПРЕДИСЛОВИЕ

Широкое использование информационных технологий, в том числе и таких сложных, как распределенное программирование, многомерные базы данных (OLAP-кубы), спутниковое телевидение, мобильная телефония, интернет-процессы и т. д., является важной особенностью жизни современного общества. Информационные технологии лежат в основе функционирования информационных систем (например, баз данных и знаний, систем радиолокации, поисковых систем в Интернете, систем телекоммуникаций и т. д.). Информационные технологии основаны на использовании процессов хранения, представления, обработки и передачи данных. Изучение этих процессов необходимо для подготовки специалистов в области информатики и информационных систем (технологий).
С учетом имеющегося дефицита учебной литературы по предмету подготовка настоящих материалов представляется актуальной.

Пособие представляет лекционный материал курса «Теория информационных процессов и систем», который излагался несколько лет студентам Белорусского государственного технологического университета (г. Минск).

Пособие состоит из двух частей. В первой части рассматриваются основы информационных процессов. Во второй части рассматриваются задачи создания, организации работы и практической реализации важных информационных систем: баз данных
и знаний, статистических (прогнозирования, принятия решений и т. п.), лингвистических (для работы с текстовой и гипертекстовой информацией, обработки языковых запросов, распознавания текста и пр.).

Замеченные ошибки и изъяны просим сообщать по адресу ovgerman@mail.ru.

Часть I
ТЕОРИЯ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ

Лекция 1
КРАТКОЕ ВВЕДЕНИЕ В ИСЧИСЛЕНИЕ ВЕРОЯТНОСТЕЙ

Цель. Дать определение вероятности. Привести основные формулы вычисления вероятностей. Привести определение функции распределения случайной величины. Пояснить материал на примерах.

1.1. Определение и вычисление вероятности

Теория вероятностей изучает количественные закономерности случайных явлений. Чтобы говорить о случайном предметно, определим понятие опыта (эксперимента).

Под *опытом* будем понимать осуществление определенных условий [1, 2]. В качестве примера возьмем подбрасывание монеты. Условиями опыта являются:

– одна и та же монета;

– один и тот же способ подбрасывания.

В опыте часть условий варьируется. В нашем примере – это сила подбрасывания (возможно что-либо еще). Таким образом, исход опыта заранее предсказать нельзя, поскольку имеются варьируемые (или неизвестные) условия. Результат опыта будем называть *случайным событием*. Любая числовая величина ξ, значение которой может меняться в зависимости от случайного события, называется *случайной величиной* [3].

В нашем примере пусть $\xi = 1$, если монета падает орлом вверх, и $\xi = 0$ – в противном случае. Таким образом, ξ есть случайная

величина. Оказывается, если один и тот же опыт повторять много раз, то событие, например $\xi = 1$, будет наступать с некоторой устойчивой частотой:

$$P(A) = \frac{m}{n}, \qquad (1.1)$$

где m – число опытов, в которых событие A (т. е. $\xi = 1$) наступило; n – число опытов.

Согласно Р. Мизесу, вероятность события A есть предел отношения (1.1) при $n \to \infty$. Подход Р. Мизеса на практике не может быть реализован, поскольку требует проведения бесконечного числа опытов. В связи с этим приведем иные способы вычисления вероятностей событий.

Определение. Два события A и B называются *несовместными*, если ни в каком опыте они не могут произойти вместе.

Например, события выпадения «орла» и «решки» несовместны.

Определение. События A_1, A_2, …, A_n образуют *полную группу несовместных событий*, если и только если результат каждого опыта есть одно и только одно событие из множества A_1, A_2, …, A_n.

Классический (лапласовский) способ вычисления вероятности основан на рассмотрении множества исходов опыта как полной группы несовместных и равновозможных событий.

Пусть, как и выше, A_1, A_2, …, A_n – множество всех равновозможных несовместных исходов опыта и пусть B – некоторое событие, связанное с данным опытом, которое возникает тогда и только тогда, когда опыт завершается любым из m различных, но зафиксированных исходов A_i. Тогда вероятность события B определяется как отношение $\frac{m}{n}$.

Пример. Рассмотрим опыт, состоящий из двух последовательных подбрасываний монеты. Требуется найти вероятность события B, состоящего в выпадении одного «орла» и одной «решки».

Обозначим через «О» («Р») событие, связанное с выпадением «орла» («решки») при одном подбрасывании монеты. Тогда множество всех исходов опыта имеет следующий вид:

{ОО, ОР, РО, РР}.

Здесь $n = 4$. Событие B имеет место в двух случаях: ОР и РО, т. е. $m = 2$. Следовательно, вероятность $P(B) = \frac{2}{4} = \frac{1}{2}$.

Приведенное классическое определение вероятности ограничено допущением о равновозможности и конечности числа исходов опыта. Наряду с классическим используют статистическое определение вероятности, основанное на понятии относительной частоты события.

Относительная частота события, наряду с вероятностью, принадлежит к основным понятиям теории вероятности [4] (приставку «относительная» будем далее опускать).

Исчисление частот подчиняется тем же формулам, что и исчисление вероятностей.

Определение. *Частотой события* называют отношение числа опытов, в которых событие появилось, к числу фактически произведенных опытов.

Таким образом, вычисление частот можно осуществить по экспериментальным таблицам.

При статистическом подходе за вероятность события принимают частоту события при условии, что число опытов достаточно велико [5]. Имеются еще геометрический и аксиоматический способы определения вероятности [1, 6].

1.2. Основные формулы

Определение. *Условной вероятностью* $P(A|B)$ *наступления события* A при условии наступления события B называется вероятность появления события A в тех и только тех опытах, в которых имело место событие B.

Определение. Событие A и B *независимы* (друг от друга), если и только если $P(A|B) = P(A)$ и $P(B|A) = P(B)$.

В качестве примера рассмотрим табл. 1.1.

Данная таблица, разумеется, служит чисто иллюстративным целям. Пусть событие B состоит в том, что человек старше 40 лет имеет в среднем более 15 нетрудоспособных дней в году по

болезни. Общее число таких лиц составляет 1. Следовательно, частота $P(B) = \frac{1}{10}$ (10 – общее число лиц в табл. 1.1).

Таблица 1.1

Пол	Старше 40 лет	Число нетрудоспособных дней по болезни за год
Ж	1	12
М	0	8
М	1	16
М	1	14
Ж	0	5
Ж	0	4
Ж	0	10
М	0	16
М	1	12
Ж	1	13

Из табл. 1.1 найдем:

$$P(\text{мужчина}) = \frac{5}{10};$$

$$P(\text{старше 40 лет}) = \frac{5}{10}.$$

Следовательно, события «быть мужчиной» и «старше 40 лет» не являются независимыми.

Для любых двух событий A и B имеет место:

$$P(AB) = P(A)P(B \mid A) = P(B)P(A \mid B). \qquad (1.2)$$

(для связки «или» используют знак «$\vee$»; для связки «и» – «&» (или «*», или вообще эту связку опускают).

Проверим это соотношение по табл. 1.1. Например,

$$P(\text{старше 40 лет и мужчина}) = \frac{5}{10} \cdot \frac{3}{5} = \frac{3}{10}.$$

В самом деле, число записей, где одновременно есть «Пол» = «М» и «Старше 40 лет» = «1», равно 3; всех записей – 10.

Вероятность наступления хотя бы одного события A или B (или обоих вместе) определяется из соотношения:

$$P(A \vee B) = P(A) + P(B) - P(AB).$$

Если события A и B независимы друг от друга, то

$$P(A \vee B) = P(A) + P(B) - P(A)P(B),$$

так как $P(AB) = P(A)P(B \mid A) = P(A)P(B).$

Если A и B несовместны, то $P(AB) = 0$ и $P(A \vee B) = P(A) + P(B).$

Определение. Событие $\overline{A}$ (читается «не A») называется *противоположным* событию A, если они образуют полную группу несовместных событий.

Из классического определения вероятности непосредственно выводим, что $P(A) + P(\overline{A}) = 1.$

В случае полной группы несовместных событий A_1, A_2, …, A_n получим

$$P(A_1) + P(A_2) + ... + P(A_n) = 1.$$

Событие S называется достоверно истинным, если $P(S) = 1,$ и достоверно ложным, если $P(S) = 0.$ Если S достоверно истинное событие, то для любого события A $P(A \vee S) = 1$ и $P(AS) = P(A).$ Далее можно установить непосредственно, что

$$P(A) = P(A \,\&\, B \vee A \,\&\, \overline{B}) = P(AB) + P(A\overline{B});$$

$$P(A \vee B) = P[A \vee (AB \vee \overline{A}B)] = P[(A \vee AB) \vee \overline{A}B] = P(A \vee \overline{A}B) =$$
$$= P(A) + P(\overline{A}B).$$

Аналогично,

$$P(A \vee B \vee C) = P(A) + P(\overline{A}B) + P(C\overline{A}\,\overline{B}) \text{ и т. д.}$$

Пусть A_1, A_2, …, A_n образуют полную группу несовместных событий. Тогда для произвольного события B, которое может наступить лишь при появлении одного из событий A_1, A_2, …, A_n, имеет место [1, 7]

$$P(B) = \sum_{i=1}^{n} P(B \mid A_i)P(A_i). \qquad (1.3)$$

Формула (1.3) допускает также следующую обобщенную формулировку [7]: пусть A_1, A_2, …, A_n образуют группу попарно несовместных событий, причем событие B может наступить лишь при появлении одного из событий A_1, A_2, …, A_n. Тогда справедли-

во соотношение (1.3), которое называется формулой полной вероятности.

Формула (1.3) доказывается так: поскольку A_1, A_2, …, A_n образуют полную группу несовместных событий, то событие $A_1 \vee A_2 \vee ... \vee A_n$ достоверно истинное. Поэтому

$$P(B) = P[B \,\&\, (A_1 \vee A_2 \vee ... \vee A_n)] = P(BA_1 \vee BA_2 \vee ... \vee BA_n) =$$
$$= P(BA_1) + P(BA_2) + ... + P(BA_n),$$

так как события BA_1, BA_2, …, BA_n попарно несовместны.

Итак,

$$P(B) = \sum_{i=1}^{n} P(BA_i) = \sum_{i=1}^{n} P(A_i)P(B \mid A_i).$$

Важную роль играет также теорема Байеса, которая устанавливает справедливость следующей формулы:

$$P(A_i|B) = \frac{P(A_i)P(B|A_i)}{\sum_{j=1}^{n} P(A_j)P(B|A_j)}. \qquad (1.4)$$

Так, используя табл. 1.1, найдем

$$P(\text{старше 40 лет} \mid \text{мужчина}) =$$
$$= P(\text{старше 40 лет})P(\text{мужчина} \mid \text{старше 40 лет}) / (P_1 + P_2),$$

где $P_1 = P(\text{старше 40 лет})P(\text{мужчина} \mid \text{старше 40 лет})$; $P_2 = P(\text{старше 40 лет})P(\text{женщина} \mid \text{старше 40 лет})$.

По таблице находим:

$$P(\text{старше 40 лет}) = \frac{5}{10};$$

$$P(\text{старше 40 лет} \mid \text{женщина}) = \frac{2}{5};$$

$$P(\text{мужчина} \mid \text{старше 40 лет}) = \frac{3}{5};$$

$$P(\text{женщина} \mid \text{старше 40 лет}) = \frac{2}{5};$$

$$P(\text{старше 40 лет} \mid \text{мужчина}) = \frac{\frac{5}{10}\cdot\frac{3}{5}}{\frac{5}{10}\cdot\frac{3}{5}+\frac{5}{10}\cdot\frac{2}{5}} = \frac{3}{5}.$$

Теорема Байеса интересна по следующей причине. Она позволяет оценить вероятность какого-либо события или явления при наличии дополнительных признаков.

Пусть E_1, E_2 – признаки. Тогда формулу Байеса запишем таким образом:

$$P(A \mid E_1, E_2) = \frac{P(A)P(E_1 \mid A)P(E_2 \mid E_1 A)}{P(E_1)P(E_2 \mid E_1)}. \qquad (1.5)$$

Пренебрегая условием возможной зависимости признаков, упростим (1.5) для случая независимых признаков:

$$P(A \mid E_1, E_2) = \frac{P(A)P(E_1 \mid A)P(E_2 \mid A)}{P(E_1)P(E_2)}. \qquad (1.6)$$

На практике часто считают, что признаки независимы, и используют формулу (1.6), которая в общем случае для $m > 2$ признаков принимает такой вид:

$$P(A \mid E_1, \ ..., \ E_m) = \frac{P(A)P(E_1 \mid A) \ \ ... \ \ P(E_m \mid A)}{P(E_1)P(E_2) \ \ ... \ \ P(E_m)}. \qquad (1.7)$$

1.3. Вероятностные распределения

Пусть ξ – случайная величина. Далее пусть $P(\xi < x)$ обозначает вероятность того, что в произвольном опыте значение случайной величины ξ не превзойдет x:

$$F(x) = P(\xi < x). \qquad (1.8)$$

Эта функция называется *функцией распределения случайной величины* [2]. Важнейшими из известных распределений являются

нормальное распределение (или распределение Гаусса), *распределение Пуассона* (для редких событий), *равномерное распределение* и др.

Из (1.8) имеем для $b > a$:

$$P(\xi < b) = P(\xi < a) + P(a \le \xi < b),$$

так как события $\xi < a$ и $a \le \xi < b$ несовместны. Следовательно,

$$P(a \le \xi < b) = F(b) - F(a). \tag{1.9}$$

Каждое вероятностное распределение характеризуется набором характеристик. Двумя важнейшими характеристиками являются математическое ожидание $M[\xi]$ и дисперсия $D[\xi]$. Пусть $F(x)$ – функция распределения. Если $F(x)$ непрерывна и дифференцируема, то можно найти ее производную:

$$f(x) = \frac{dF(x)}{dx}. \tag{1.10}$$

Функция $f(x)$ в (1.10) носит название *плотности распределения*. Ясно, что

$$F(x) = \int_{-\infty}^{x} f(x)dx. \tag{1.11}$$

Кроме того,

$$\int_{-\infty}^{\infty} f(x)dx = F(\infty) - F(-\infty) = 1.$$

Определение. *Математическим ожиданием* $M[\xi]$ и *дисперсией* $D[\xi]$ случайной величины ξ с непрерывной дифференцируемой функцией распределения называются величины

$$M[\xi] = \int_{-\infty}^{\infty} xf(x)dx \tag{1.12}$$

и

$$D[\xi] = \int_{-\infty}^{\infty} (x - M[\xi])^2 f(x) dx \qquad (1.13)$$

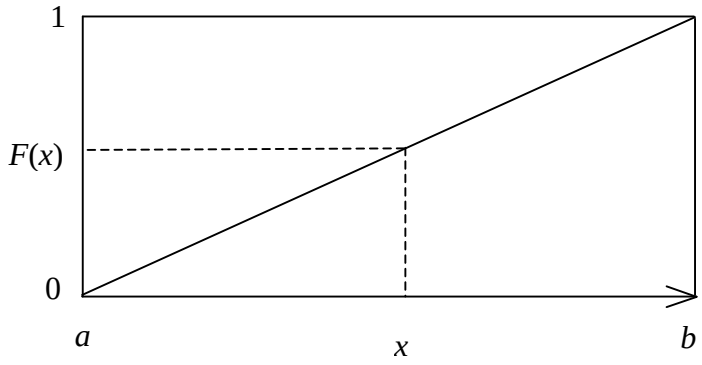
соответственно.

Важнейшим вероятностным распределением является нормальное распределение (распределение К. Гаусса). Для него

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-M[\xi])^2}{2\sigma^2}}, \qquad (1.14)$$

где $\sigma = \sqrt{D[\xi]}$ определяется как корень квадратный из дисперсии и называется *среднеквадратическим отклонением* случайной величины ξ. Формулу (1.14), разумеется, удобно использовать только тогда, когда σ и $M[\xi]$ известны заранее или могут быть предварительно оценены. Применим формулы (1.12), (1.13) для равномерно распределенной случайной величины, изменяющейся на отрезке [*a*, *b*].

Функцию распределения можно представить графически (см. рисунок).



Рисунок

Из простых геометрических соображений найдем:

$$F(x) = \frac{x - a}{b - a};$$

$$f(x) = \frac{dF(x)}{dx} = \frac{1}{b - a};$$

$$M[\xi]=\int_a^b \frac{1}{b-a}x dx=\frac{x^2}{2(b-a)}\Big|_a^b=\frac{b^2-a^2}{2(b-a)}=\frac{b+a}{2};$$

$$D[\xi]=\int_a^b\left(x-\frac{b+a}{2}\right)^2\frac{1}{b-a}dx=\frac{(b-a)^2}{12}.$$

Рассмотрим теперь дискретную случайную величину, т. е. такую величину, которая может принимать лишь конечное или счетное число значений $\ldots<x_{-1}<x_0<x_1<x_2<\ldots<x_k<\ldots$ Обозначим P_i вероятность того, что дискретная случайная величина ξ принимает значение x_i.

Функция распределения дискретной случайной величины ξ определяется так:

$$F(\xi)=\sum_{x_i<x}P(x_i). \tag{1.15}$$

Математическое ожидание дискретной случайной величины ξ находится из соотношения

$$M[\xi]=\sum_i x_i P(x_i), \tag{1.16}$$

а дисперсия из соотношения [4]

$$D[\xi]=\sum_i\left(x_i-M[\xi]\right)^2 P_i. \tag{1.17}$$

В (1.16), (1.17) индекс i «пробегает» по всему множеству индексов значений x_i.

За иллюстрацией обратимся к табл. 1.2, в которой приведены данные о сдаче экзамена студентами.

Таблица 1.2

Оценка	Число студентов, получивших оценку	P_i
3	5	5/25
4	3	3/25
5	6	6/25
6	4	4/25
7	3	3/25
8	2	2/25
9	1	1/25
10	1	1/25
∑	25	1

Найдем математическое ожидание оценки из (1.16):

$$M = 3 \cdot \frac{5}{25} + 4 \cdot \frac{3}{25} + 5 \cdot \frac{6}{25} + 6 \cdot \frac{4}{25} + 7 \cdot \frac{3}{25} + 8 \cdot \frac{2}{25} + 9 \cdot \frac{1}{25} + 10 \cdot \frac{1}{25} \approx 5{,}08.$$

Знание функции (плотности) распределения случайной величины позволяет отыскивать значения вероятности того, что случайная величина принимает то или иное значение. Например, вероятность того, что нормально распределенная случайная величина ξ с математическим ожиданием 1,7 и среднеквадратическим отклонением 0,05 окажется в диапазоне [0; 1,8] равна:

$$P(0 \le \xi \le 1{,}8) = \frac{1}{0{,}05\sqrt{2\pi}} \int_0^{1{,}8} e^{\frac{(x-1{,}7)^2}{2 \cdot 0{,}05^2}} dx = 0{,}72.$$

Лекция 2
ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ ТЕОРИИ ИНФОРМАЦИИ

Цель. Сформулировать понятие энтропии по Шеннону. Дать определение количества информации через энтропию. Привести основные формулы для вычисления условной энтропии.

2.1. Энтропия как мера неопределенности

Основы вычисления информации заложил Клод Шеннон в 1948 г. Он ввел следующую формулу для оценки степени (меры) неопределенности (сложности) системы S, которая с вероятностями P_1, P_2, …, P_n может находиться в одном из состояний s_1, s_2, …, s_n (соответственно):

$$H = -\sum_{i=1}^{n} P_i \log_2 P_i. \qquad (2.1)$$

Мера неопределенности (сложности) системы называется *энтропией*. Этот термин впервые ввел Клаузис в XIX веке.

Количество информации, содержащееся в сигнале, можно связать с изменением энтропии (неопределенности системы) при получении сигнала.

Пример. Вероятность того, что приемный пункт работает, составляет $P_1 = \frac{7}{8}$, а что не работает $P_2 = \frac{1}{8}$. Энтропия такой системы:

$$H_0 = -\frac{1}{8}\log_2 \frac{1}{8} - \frac{7}{8}\log_2 \frac{7}{8} = 0{,}544 \text{ (бита)}.$$

Любой сигнал, устанавливающий, что пункт работает или нет, полностью снимает энтропию. Поэтому такой сигнал несет информацию $I = H_0 = 0{,}544$ бита.

Теорема 1. Энтропия системы S имеет максимальное значение тогда и только тогда, когда вероятности ее состояний равны.

Доказательство дано в [8].

Пусть ξ, ω – две случайные величины.

Определение. Случайная величина ω называется независимой от случайной величины ξ, если закон распределения величины ω не зависит от того, какое значение приняла величина ξ [5].

Рассмотрим сначала случай дискретных случайных величин ξ и ω.

Пусть H_ξ – энтропия случайной величины ξ (соответственно H_ω – энтропия случайной величины ω). Имеют место следующие результаты [5, 8].

Теорема 2. Если ξ и ω – две независимые случайные величины, то совместная энтропия

$$H_{\xi,\omega} = H_\xi + H_\omega.$$

Под *совместной энтропией* понимается энтропия двумерной случайной величины $z = (\xi, \omega)$, представляющая собой случайный вектор с компонентами ξ и ω, являющимися случайными величинами.

В тексте далее для упрощения выкладок будем обозначать:

$$P(\xi = x) \text{ через } P(x);$$

$$P(\omega = y) \text{ через } P(y);$$

$$P(\xi = x, \omega = y) \text{ через } P(x, y);$$

$\sum_x \ldots$ означает, что суммирование ведется по всему множеству значений x, принимаемых случайной величиной ξ (соответственно $\sum_y \ldots$).

Д о к а з а т е л ь с т в о.

$$H(\xi,\omega) = -\sum_x \sum_y P(x,y)\log_2 P(x,y) =$$

$$= -\sum_x \sum_y P(x)P(y)[\log_2 P(x) + \log_2 P(y)] =$$

$$= -\sum_x \sum_y P(x)\log_2 P(x)P(y) - \sum_x \sum_y P(x)\log_2 P(y)P(y) =$$

$$= -\sum_x P(x)\sum_y P(y)\log_2 P(y) - \sum_y P(y)\sum_x P(x)\log_2 P(x) =$$

$$= +\sum_x P(x)H_\omega + \sum_y P(y)H_\xi = H_\omega + H_\xi,$$

так как $\sum_x P(x)H_\omega = H_\omega$ ввиду того, что $\sum_x P(x) = 1$.

Определение. *Условной энтропией* $H_{\xi|\omega}$ случайной величины ξ относительно случайной величины ω называется величина

$$H_{\xi|\omega} = -\sum_x P(x \mid y) \cdot \log_2 P(x \mid y). \tag{2.2}$$

Теорема 3. Если ξ и ω – две зависимые случайные величины, то их совместная энтропия равна:

$$H_{\xi,\omega} = H_\xi + H_{\omega|\xi} = H_\omega + H_{\xi|\omega}. \tag{2.3}$$

Д о к а з а т е л ь с т в о.

$$H_{\xi,\omega} = -\sum_x \sum_y P(x,y) \log_2 P(x,y) =$$

$$= -\sum_x \sum_y P(x)P(y \mid x)[\log_2 P(x) + \log_2 P(y \mid x)] =$$

$$= -\sum_x \sum_y P(x)P(y \mid x) \log_2 P(x) - \sum_x \sum_y P(x)P(y \mid x) \log_2 P(y \mid x) =$$

$$= -\sum_x \sum_y P(x) \frac{P(x \mid y)P(y)}{P(x)} \log_2 P(x) - \sum_x P(x) \sum_y P(y \mid x) \log_2 P(y \mid x) =$$

$$= -\sum_x \sum_y P(x \mid y)P(y) \log_2 P(x) + \sum_x P(x) H_{\omega|\xi} =$$

$$= -\sum_x \log_2 P(x) \sum_y P(y)P(x \mid y) + H_{\omega|\xi} =$$

$$= -\sum_x \log_2 P(x)P(x) + H_{\omega|\xi} = H_\xi + H_{\omega|\xi}.$$

Заметим, что $\sum_y P(y)P(x \mid y) = \sum_y P(x,y) = P(x)$.

Рассмотрим величину

$$I_{\xi,\omega} = H_\xi - H_{\xi|\omega}. \tag{2.4}$$

Эта величина называется *шенноновским количеством информации*, содержащимся в сигнале ω относительно сигнала ξ. Оценка (2.4) очень удобна при вычислении информативности признаков
в диагностических задачах. К этой оценке обратимся ниже.

Для вычисления энтропии непрерывной случайной величины ξ используют замену суммы (2.1) интегралом [5]:

$$H(\xi) = -\int_{-\infty}^{\infty} f(x)\log_2 f(x)dx, \qquad (2.5)$$

где $f(x)$ – функция плотности распределения случайной величины ξ.

2.2. Задача фильтрации

Под фильтрацией сложного сигнала понимают устранение определенных составляющих сигнала. Например, для передачи аналогового сигнала получают наложение (суперпозицию) этого сигнала на высокочастотный (мощный) сигнал, который играет роль несущего. На приемной стороне следует устранить эту высокочастотную составляющую, чтобы получить исходный сигнал.

Задачей фильтрации может быть устранение помехи (шума). При этом сигнал помехи имеет неизвестный закон распределения.

Интересной задачей фильтрации является задача сглаживания временных рядов данных. Временными рядами данных являются, например, еженедельные (месячные, квартальные) данные о цене на нефть на мировом рынке. Цена на нефть складывается из некоторой закономерной составляющей, характеризующей изменение уровня потребности в нефти в мире, и случайной составляющей, которая зависит, например, от политической обстановки в нефтедобывающих регионах. Целью сглаживания является устранение случайной составляющей и получение некоторой закономерной линии, называемой линией тренда. Простейшее уравнение фильтра, используемого для сглаживания временного ряда, следующее:

$$y_i = a_0 y_{i-1} + a_1 y_i + a_2 y_{i+1};$$

$$a_0 + a_1 + a_2 = 1.$$

Это уравнение соответствует функции сглаживания; здесь y_i – значение сигнала на i-м такте. Например, часто используют функцию сглаживания вида

$$y_i = \frac{1}{4} y_{i-1} + \frac{1}{2} y_i + \frac{1}{4} y_{i+1}.$$

Задача расчета фильтра состоит в определении коэффициентов a_i, которые рассчитываются исходя из требования, чтобы восстановленное значение сигнала как можно меньше отличалось от оригинального.

Лекция 3
ЗАДАЧА ПОСТРОЕНИЯ ДИАГНОСТИЧЕСКОГО ДЕРЕВА

Цель. Показать способ оценки информативности признаков через энтропию. Рассмотреть задачу построения диагностического (классифицирующего) дерева. Привести алгоритм построения дерева минимального размера.

3.1. Описание рабочего примера

Рассмотрим табл. 3.1 с данными, относящимися к сотрудникам некоторого предприятия

Таблица 3.1

Пол	Возраст	Вес	Напряженная работа	Число нетрудоспособных дней по болезни за год
Ж	25	51	1	12*
М	30	80	0	8
М	44	73	1	16*
М	22	81	1	14*
Ж	53	72	0	5
Ж	38	61	0	4
Ж	34	64	0	10
М	42	93	0	16*
М	37	62	1	12*
Ж	22	51	1	13*

Примечание. В столбце «Напряженная работа» «1» соответствует значению «да», «0» – «нет».

В табл. 3.1 нас будет интересовать столбец «Число нетрудоспособных дней по болезни за год», который характеризует состояние здоровья сотрудника в зависимости от других приведенных факторов и называется диагностическим. Звездочкой отмечены случаи, в которых количество нетрудоспособных дней по болезни превышает 10 (условно – это пороговая величина).

Рассмотрим построение диагностического дерева (понятие дерева мы пока дадим иллюстративно) для этой таблицы. Такое

дерево позволяет по значениям признаков (пол, возраст, вес, напряженная работа) конкретного работника выяснить, ожидается ли у него число нетрудоспособный дней в году более 10 или нет.

Разобьем диапазоны количественных признаков (к ним относятся возраст, вес) на поддиапазоны, число которых на практике принято брать равным $\sqrt{n}$, где n – число записей в таблице. Поскольку $n=10,\ \sqrt{10}\approx 3{,}2,$ возьмем три поддиапазона. В столбце «Возраст» найдем максимальное (h) и минимальное (l) значения: $h=53,\ \ l=22.$ Интервал [22; 53] разобьем на три поддиапазона, каждый из них имеет длину $\frac{53-22}{3}\approx 10{,}3.$ Имеем такое разбиение: [22; 32,3], (32,3; 42,6], (42,6; 53].

Аналогично поступим с признаком «Вес», разбив его диапазон значений на следующие поддиапазоны: [51; 65], (65; 79], (79; 93].

Теперь табл. 3.1 заменим табл. 3.2.

Таблица 3.2

Пол	Возраст	Вес	Напряженная работа	Число нетрудоспособных дней по болезни за год
Ж	[22; 32,3]	[51; 65]	1	*
М	[22; 32,3]	(79; 93]	0	
М	(42,6; 53]	(65; 79]	1	*
М	[22; 32,3]	(79; 93]	1	*
Ж	(42,6; 53]	(65; 79]	0	
Ж	(32,3; 42,6]	[51; 65]	0	
Ж	(32,3; 42,6]	(79; 93]	0	
М	(32,3; 42,6]	(79; 93]	0	*
М	(32,3; 42,6]	[51; 65]	1	*
Ж	[22; 32,3]	[51; 65]	1	*

Примером диагностического дерева может служить дерево, показанное на рис. 3.1.

На рис. 3.1 ребрам приписаны признаки. В узле *1* проверяем пол. Если «Ж», то двигаемся в узел *2*, в котором проверяем признак «НР» (напряженная работа) и «$\overline{\text{НР}}$» (не напряженная работа). Все ветки, которые заканчиваются символом «*» соответствуют людям, имеющим более 10 нетрудоспособных дней по болезни.

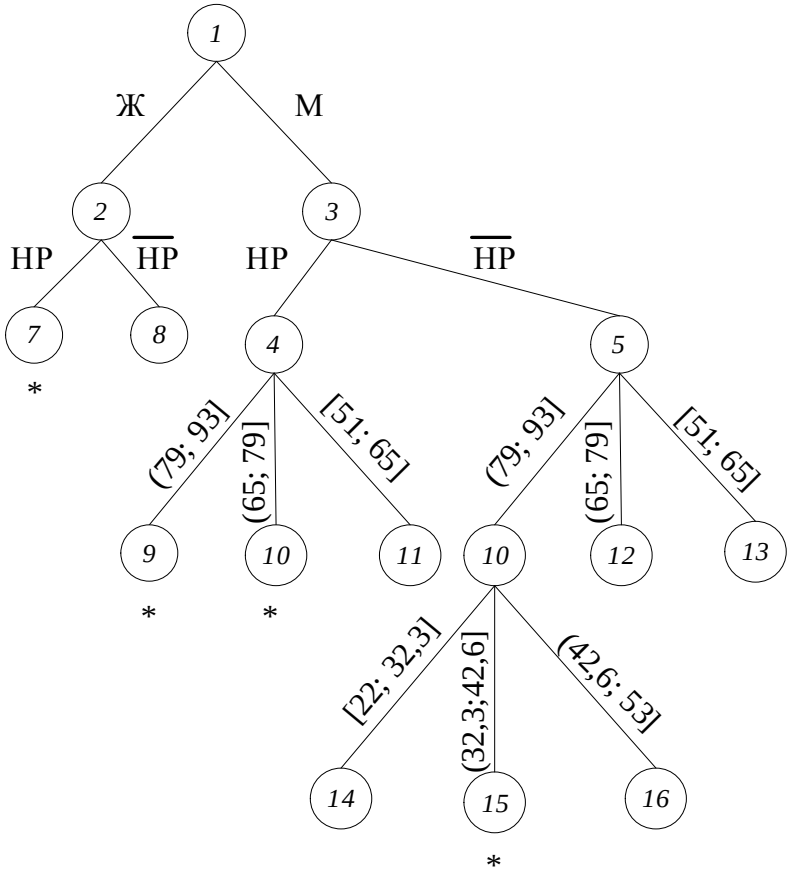


Рис. 3.1

Возникает важный практический вопрос: как построить дерево с минимальным числом узлов (минимальное диагностическое дерево)? Это очень сложная задача. Мы ограничимся эвристическим алгоритмом, который использует аналитическую оценку информативности признаков. Можно построить целое множество диагностических деревьев для одной и той же таблицы. Проблема состоит в отыскании наименьшего по размеру дерева.

3.2. Эвристический алгоритм построения минимального диагностического дерева

Идея алгоритма состоит в оценивании информативности признаков на основе шенноновской формулировки. Наиболее информативный признак следует использовать для ветвления в корневом узле диагностического дерева (вершина *1* на рис. 3.1).

Затем следует искать для каждой из вершин переходов *2* и *3* на рис. 3.1 следующие по величине информативности признаки и т. д. Для раскрытия этой идеи нужно разъяснить, как выполнить расчет информативности признаков по табл. 3.2. Напомним, что в табл. 3.2 в качестве диагностического используется признак «Число нетрудоспособных дней по болезни за год». В этой графе имеется символ «*». Соответственно, вероятность данного исхода $P_{\text{Н}} = \frac{6}{10}$. Вероятность противоположного события $P_{\overline{\text{Н}}} = \frac{4}{10}$. Энтропия H_0 исходной системы составляет $H_0 = -\frac{6}{10}\log_2\frac{6}{10} - \frac{4}{10}\log_2\frac{4}{10} \approx 0{,}97$.

Пользуясь табл. 3.2, проведем теперь расчет информативности каждого признака в отдельности. Возьмем первый признак «Пол». Этот признак принимает всего два значения: «М» и «Ж». Для каждого из этих значений построим таблицу, включив в нее только те значения из табл. 3.2, где признак «Пол» равен «М» (соответственно «Ж»). Так, для «Пол» = «М» искомая таблица принимает вид табл. 3.3.

Таблица 3.3

Пол	Число нетрудоспособных дней по болезни за год	[Другие признаки]
М		–
М	*	–
М	*	–
М	*	–
М	*	–

Графа «Другие признаки» нас, разумеется, не интересует, и объединяет все остальные признаки табл. 3.2. Найдем значение энтропии полученной таблицы.

$$H_{\text{М}} = -\frac{1}{5}\log_2\frac{1}{5} - \frac{4}{5}\log_2\frac{4}{5} \approx 0{,}72.$$

Примечание. В табл. 3.3 всего пять записей, из которых в четырех встречается значение «*».

Аналогичным образом строим таблицу для значения «Пол» = «Ж» (табл. 3.4) и вычисляем энтропию.

Таблица 3.4

Пол	Число нетрудоспособных дней по болезни за год	[Другие признаки]
Ж	*	–
Ж		–
Ж		–
Ж		–
Ж	*	–

$$H_{\text{Ж}} = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} \approx 0{,}97.$$

Теперь можно оценить среднестатистическое количество информации, доставляемое признаком «Пол». Для наглядности приводим рис. 3.2, на основании которого этот расчет и следует произвести.

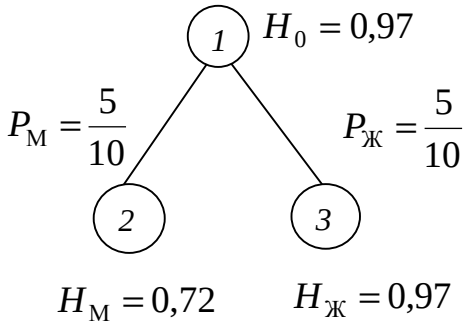


Рис. 3.2

Из рис. 3.2 видно, что если наблюдаемое лицо – мужчина (вероятность этого $P_{\text{М}} = 0{,}5$), то энтропия снижается с 0,97 до 0,72. Если же наблюдаемое лицо – женщина, то энтропия не изменяется. Взвешивая этот результат по вероятности, найдем, что энтропия «результирующей системы» после уяснения значения пола наблюдаемого лица составит

$$H_{\text{Пол}} = 0{,}5 \cdot 0{,}72 + 0{,}5 \cdot 0{,}97 = 0{,}845.$$

Отсюда количество информации, доставляемой признаком «Пол»,

$$I_{\text{Пол}} = H_0 - H_{\text{Пол}} = 0{,}97 - 0{,}845 = 0{,}125 \text{ бита}.$$

Аналогичные расчеты надлежит произвести по признакам «Возраст», «Вес», «Напряженная работа».

Опуская все сопутствующие выкладки, получаем следующий итоговый результат:

$$I_{\text{Пол}} = 0{,}125 \text{ бита};$$

$$I_{\text{Возраст}} = 0{,}046 \text{ бита};$$

$$I_{\text{Вес}} = 0{,}046 \text{ бита};$$

$$I_{\text{Напряженная работа}} = 0{,}61 \text{ бита}.$$

Замечание.

1. Обращаем внимание на то, что для признаков «Возраст» и «Вес» следует строить три таблицы (так как эти признаки имеют по три значения).

2. Энтропия равна 0, если вероятность символа «*» равна 1 или 0. Такая ситуация имеет место для признака «Напряженная работа».

Итак, мы видим, что наиболее информативным является признак «Напряженная работа». Этот признак мы помещаем в верхнюю часть искомого диагностического дерева (рис. 3.3)

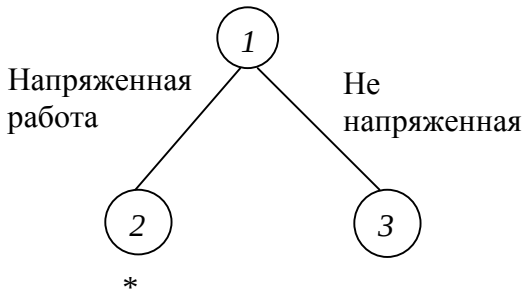


Рис. 3.3

Очевидно, что ветвь «Напряженная работа» сразу устанавливает диагноз: «Потерянные дни по нетрудоспособности» = «*». Поэтому надлежит продолжить построение дерева с узла *3* на рис. 3.3. Прежде всего, перепишем табл. 3.2 для узла *3*, оставив только записи, где значение признака «Напряженная работа» = 0 (т. е. «Не напряженная работа»).

Таблица 3.5

Пол	Возраст	Вес	Напряженная работа	Число нетрудоспособных дней по болезни за год
М	[22; 32,3]	(79; 93]	0	
Ж	[42,6; 53]	(65; 79]	0	
Ж	(32,3; 42,6]	[51; 65]	0	
Ж	(32,3; 42,6]	(79; 93]	0	
М	(32,3; 42,6]	(79; 93]	0	*

Теперь по табл. 3.5 нужно оценить информативность признаков «Пол», «Возраст», «Вес». Опуская все промежуточные, вычисления запишем итоговый результат.

$$I_{\text{Пол}} = 0{,}32 \text{ бита};\ I_{\text{Возраст}} = 0{,}17 \text{ бита};\ I_{\text{Вес}} = 0{,}17 \text{ бита}.$$

Наиболее информативным для табл. 3.5 является признак «Пол». С учетом этого дерево на рис. 3.3 изменяется, как показано на рис. 3.4.

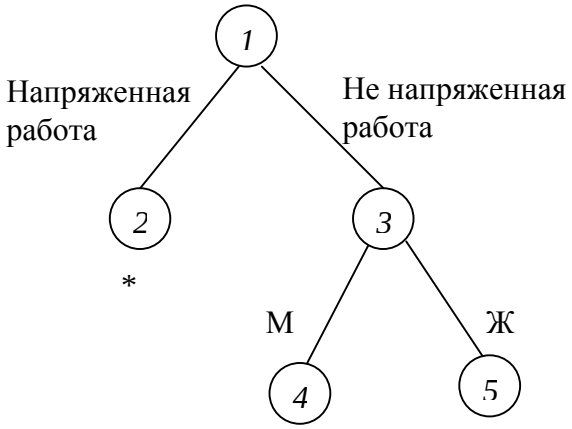


Рис. 3.4

В дереве на рис. 3.4 не ясна только ситуация в узле *4*. В узле *5* сразу ставим диагностический ответ «Потерянные дни по нетрудоспособности» = «нет». Для узла *4* строим новую табл. 3.6.

Таблица 3.6

Пол	Возраст	Вес	Напряженная работа	Число нетрудоспособных дней по болезни за год
М	[22; 32,3]	(79; 93]	0	
М	(32,3; 42,6]	(79; 93]	0	*

Из табл. 3.6 видно, что нужно проверять признак «Возраст», так как все остальные признаки не различимы (имеют каждый в своей графе одни и те же значения). Таким образом, итоговое диагностическое дерево представлено на рис. 3.5.

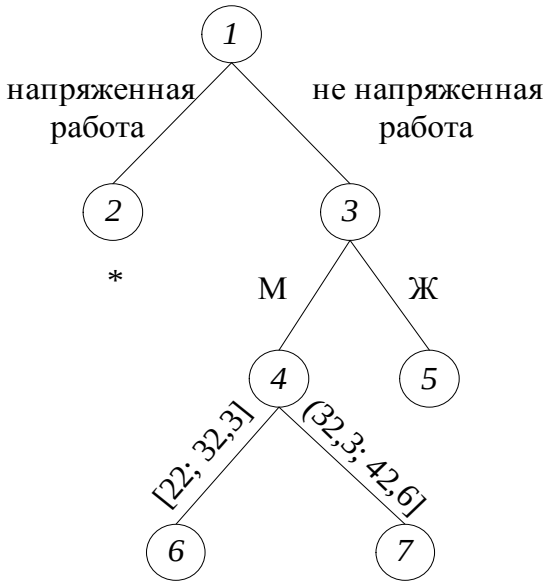


Рис. 3.5

В этом дереве всего семь узлов. Заметим, что на рис. 3.1 в дереве 16 узлов, так что нам удалось уменьшить размеры более чем в 2 раза.

Следует особо отметить, что не для всякой таблицы можно построить диагностическое дерево, которое бы позволяло однозначно давать заключения. Ситуации, в которых постановка диагноза сохраняет неоднозначность, будут обсуждаться далее.

Лекция 4
ПРИНЯТИЕ РЕШЕНИЙ НА ОСНОВЕ БАЙЕСОВСКОГО ПОДХОДА

Цель. Рассмотреть задачу принятия решений в условиях неопределенности. Привести теорему Байеса и показать ее применение для решения задачи выбора решения.

4.1. Описание рабочего примера

В качестве исходных данных снова рассмотрим табл. 3.2. В лекции 3 мы строили диагностическое дерево, однако, как мы отметили в той лекции, не всегда такое дерево может однозначно дать диагноз. Например, мы можем не знать, является ли работа сотрудника напряженной, или нет. В этом случае на помощь приходит теорема Байеса. Эта теорема может использоваться непосредственно либо на ее основе строят специальные методы. Рассмотрим оба варианта. Будем решать следующую задачу. Известно, что наблюдаемое лицо – мужского пола, 27 лет. Нам необходимо оценить вероятность того, что этот человек имеет более 10 нетрудоспособных дней в году по болезни.

Рассмотрим сначала байесовский статистический подход.

4.2. Использование Байесовской стратегии

Байесовская формула находит разнообразное применение. Одной из таких областей применения является задача распознавания. Эту задачу в общем случае можно сформулировать так: отнести наблюдаемый объект к одному из известных классов объектов
таким образом, чтобы оптимизировать значение критерия распознавания. Достаточно часто в качестве критерия распознавания
используют минимум штрафа за неправильное распознавание. Рассмотренная ранее задача построения диагностического дерева преследует ту же цель: распознать по наблюдаемым признакам

значение диагностического признака у объекта. Заметим, что из практических соображений использование диагностического дерева не всегда целесообразно, поскольку его размеры могут быть огромными. Поэтому применение той или иной байесовской стратегии не только целесообразно, но и нередко наиболее эффективно.

Из других известных подходов к распознаванию можно отметить подходы, использующие понятие расстояния (в метрическом пространстве); логические подходы; подходы, основанные на использовании разделяющей гиперплоскости, и др. Эти специальные вопросы в настоящих текстах лекций не рассматриваются.

Остановимся на методе минимизации риска неправильной классификации, который использует формулу Байеса. Отправной для этого метода является следующая формула:

$$R_k = \sum_{i=1}^{m} C_{ik} P(w_i \mid y). \qquad (4.1)$$

В этой формуле рассматривается множество классов w_1, w_2, …, w_m объектов или состояний. Применительно к рассматриваемой нами задаче пусть w_1 есть класс лиц, имеющих в году не более 10 нетрудоспособных дней, и w_2 – это класс лиц, имеющих в году более 10 нетрудоспособных дней. Далее, y – это наблюдаемое лицо. Более того, это вектор числовых характеристик наблюдаемого лица. Для простоты примем, что этот вектор представлен только одной характеристикой – возрастом (27 лет). Таким образом, $P(w_i \mid y)$ – это условная вероятность того, что наблюдаемое лицо принадлежит классу w_i с заданной характеристикой y. В нашем случае имеем P(«нетрудоспособных дней ≤ 10» | «возраст» = 27) и P(«нетрудоспособных дней > 10» | «возраст» = 27).

Далее, C_{ik} – это величина штрафа, который следует заплатить, если объект в действительности принадлежит к классу w_i, а его отнесли к классу w_k. Очень часто для упрощения выкладок принимают $C_{ik} = 1$, если и только если $i \neq k$, и $C_{ik} = 0$, если и только если $i = k$. Формулу (4.1) применяют для расчета R_1, R_2, …, R_m, после чего объект относят к тому классу, для которого величина штрафа R_i минимальна. Обратимся к табл. 3.2 за исходными данными.

Будем использовать представленное в ней разбиение интервала изменения возраста на поддиапазоны.

Преобразуем формулу (4.1), воспользовавшись формулой Байеса. Итак,

$$\sum_{i=1}^{m} C_{ik} P(w_i \mid y) = \sum_{i=1}^{m} C_{ik} \frac{P(w_i) P(y \mid w_i)}{P(y)}.$$

Вместо (4.1) будем применять формулу

$$R_k = \sum_{i=1}^{m} C_{ik} P(w_i) P(y \mid w_i). \qquad (4.2)$$

Замечание. Множитель $P(y)$ исключен, так как он одинаков при вычислении $R_1, R_2, \ldots, R_m$ для каждого класса.

Для рассматриваемого нами примера двух классов w_1 и w_2 формула (4.2) дает:

$$R_1 = C_{11} P(w_1) P(y \mid w_1) + C_{21} P(w_2) P(y \mid w_2);$$

$$R_2 = C_{12} P(w_1) P(y \mid w_1) + C_{22} P(w_2) P(y \mid w_2).$$

В силу нашего допущения о C_{ij} имеем:

$$R_1 = C_{21} P(w_2) P(y \mid w_2) = P(w_2) P(y \mid w_2);$$

$$R_2 = C_{12} P(w_1) P(y \mid w_1) = P(w_1) P(y \mid w_1).$$

Из таблицы (3.2) сразу находим: $P(w_1) = \frac{4}{10}$; $P(w_2) = \frac{6}{10}$; $P(y \mid w_1) = \frac{1}{4}$; $P(y \mid w_2) = \frac{3}{6}$.

Находим непосредственно

$$R_1 = \frac{6}{10} \cdot \frac{3}{6} = \frac{3}{10}; \; R_2 = \frac{4}{10} \cdot \frac{1}{4} = \frac{1}{10}.$$

Поскольку $R_1 > R_2$, то заключаем, что если возраст равен 27 годам, то следует выбрать класс w_2.

На практике часто вместо вероятности $P(y \mid w_i)$ используют плотность вероятности $f(y \mid w_i)$. Покажем, что такая замена правомочна.

В силу (1.9)

$$P(a \le y < a + \Delta) = \int_a^{a+\Delta} f(y)dy,$$

используя представление интеграла суммой, получим:

$$\int_a^{a+\Delta} f(y)dy = \sum_a^{a+\Delta} f(y_i)\Delta = f(y_k)\Delta$$

(в силу теоремы Лагранжа), $y_k \in [a, a+\Delta]$. Таким образом, при $\Delta \to 0$ можно использовать приближенное равенство

$$P(a \le y < a + \Delta) = f(a)\Delta.$$

Учитывая, что Δ фигурирует в каждой формуле для $P(y \mid w_i)$, просто заменяем $P(y \mid w_i)$ на $f(y \mid w_i)$. Может возникнуть вопрос, как эту методику применить в случае многомерных данных: претерпевает изменения формула (1.14) для расчета плотности $f(x)$. Например, для случая нормального распределения будем иметь

$$f(y \mid w_i) = \frac{1}{(2\pi)^{\frac{n}{2}} |C_i|^{\frac{1}{2}}} e^{\left[-\frac{1}{2}\cdot(y-\bar{y}_i)^T C_i^{-1}(y-\bar{y}_i)\right]}, \qquad (4.3)$$

где n – размерность вектора многомерных данных; C_i – корреляционная матрица, построенная по исходной таблице данных; $|C_i|$ – определитель этой матрицы; $\bar{y}_i$ – вектор средних значений признаков в классе w_i; C_i^{-1} – обратная матрица.

При использовании функции плотности вероятности нет необходимости разбивать интервал изменения возраста на поддиапазоны, что является несомненным достоинством. Однако теперь потребуется знать вероятностный закон распределения возраста. Определение такого закона представляет самостоятельную задачу.

В иллюстративных целях будем считать, что возраст распределен по нормальному закону (1.14). Для того чтобы знать закон распределения вероятности y в классе w_1 (w_2), нужно табл. 3.1 разбить на две части. В одну из них включим только записи, относящиеся к классу w_1 (лица, имеющие не более 10

нетрудоспособных дней в году), а во вторую – только записи из класса w_2 (лица, имеющие более 10 нетрудоспособных дней в году):

$w_1 =$

Возраст
30
53
38
34

$w_2 =$

Возраст
25
44
22
42
37
22

Воспользуемся выборочными средним и среднеквадратичным отклонениями в качестве оценок истинного среднего и истинного среднеквадратического отклонений в классе w_1 (w_2). В статистике этот вопрос требует применения специального критерия (например, χ^2), который рассматривать не будем.

Для класса w_1 найдем:

$$m_1 = 38{,}75;\ \ \sigma_1 = \sqrt{D_1} = 10{,}04;$$

$$f_1(27) = \frac{1}{\sqrt{2\pi}\cdot 10{,}04} e^{-\frac{(27-38{,}75)^2}{2\cdot 10{,}04^2}} \approx 0{,}021.$$

Для класса w_2 найдем:

$$m_2 = 32;\ \ \sigma_2 = 10{,}178;\ \ f_2(27) \approx 0{,}036.$$

Отсюда $R_1 = \frac{6}{10}\cdot 0{,}036;\ \ R_2 = \frac{4}{10}\cdot 0{,}021$ и $R_1 > R_2$.

И в этом случае вывод о зачислении лица в возрасте 27 лет в класс w_2 сохраняет силу.

Далее, рассмотрим один интересный подход, который позволяет (при определенных допущениях) свести многомерный случай к одномерному. Для этого нам потребуется функция, которая позволит многомерные целочисленные данные однозначно сопоставить с целым числом [9].

Теорема 4. Пусть x, y – целые положительные числа или 0. Тогда с каждой парой значений (x, y) однозначно сопоставимо целое число

$$z = \frac{1}{2}(x^2 + 2xy + y^2 + 3x + y). \qquad (4.4)$$

Справедливо и обратное. Каждому неотрицательному z однозначно соответствует целочисленный вектор (x, y), удовлетворяющий (4.4)

Рассмотрим, как воспользоваться этим результатом. Пусть дана табл. 4.1.

Таблица 4.1

Возраст	Вес	Число нетрудоспособных дней по болезни за год
25	52	12
30	80	8
44	73	16
22	81	14
…	…	… (данные из табл. 1.3)

Тогда, полагая x = «Возраст», y = «Вес», по формуле (4.4) получаем табл. 4.2.

Таблица 4.2

x	y	z	Число нетрудоспособных дней по болезни за год
25	52	3028	12
30	80	6135	8
44	73	6947	16
22	81	5378	14
…	…	…	…

Разумеется, значение результирующей величины z существенно увеличивается (как квадратичная функция). Однако мы всегда можем пронормировать z, т. е. получить ее значения в диапазоне 0–1 (или от –1 до 1), а во-вторых, ее можно просто промасштабировать, например, разделив все числа в графе z на 1000. Важно заметить, что x и y не обязательно должны быть целыми (они могут быть рациональными с заранее оговоренным числом знаков после запятой). Пусть число значащих цифр после запятой равно r, тогда получим целые числа, умножив исходные на 10^r.

Лекция 5
СТАТИСТИЧЕСКИЕ МЕТОДЫ ОЦЕНКИ ГИПОТЕЗ

Ц е л ь . Рассмотреть статистические критерии Фишера и χ^2.

При ограниченном числе опытов необходимы критерии, которые позволяли бы с определенной вероятностью ошибки принимать или отклонять гипотезы. Одной из основных является, например, гипотеза о том, что выборочные значения среднего M^* и дисперсии D^* являются подходящими оценками математического ожидания M и дисперсии D оцениваемого закона распределения. Другим важным вопросом является вопрос о том, насколько реальные данные и данные, полученные с помощью той или иной модели, близки, чтобы ту или иную модель можно было использовать, например, для предсказания.

В этой лекции мы познакомимся с двумя важнейшими статистическими критериями, применяемыми для получения ответов на поставленные и подобные им вопросы.

5.1. Критерий Фишера

Критерий Фишера позволяет сравнить два ряда чисел на предмет их близости (статистической адекватности) [10, 11]. Первый ряд чисел – это реальные данные (например, данные измерений или опытные данные). Второй ряд чисел – это данные, полученные с помощью модели для каждого соответствующего измерения или опыта. Критерий Фишера позволяет оценить, можно ли считать данную модель пригодной для получения оценок реальных величин. Задача построения модели является центральной в теории статистического моделирования и регрессионном анализе [10–12].

Итак, рассмотрим табл. 5.1, в которой содержатся данные о весе и росте некоторой совокупности людей.

Пусть построена модель следующего вида:

$$\text{Рост} = \text{Вес} + 100. \qquad (5.1)$$

Таблица 5.1

Вес	**Рост**	**Рост^{M}**
70	164	170
75	180	175
82	187	182
76	174	176
80	172	180
95	185	195
64	166	164
68	172	168

Эта модель позволяет по весу найти рост и наоборот. В табл. 5.1 в графе «Рост ^{M}» мы привели значения роста, вычисленные по модели. Хороша модель или плоха? Для ответа используем статистический критерий Фишера.

$$F_{\text{р}} = \frac{S^2_{\text{ад}}}{S^2_{\text{общ}}}; \tag{5.2a}$$

$$S^2_{\text{ад}} = \frac{1}{n-p}\sum_{i=1}^{n}(\text{Рост}_i^{\text{M}} - \text{Рост}_i)^2; \tag{5.2b}$$

$$S^2_{\text{общ}} = \frac{1}{n-1}\sum_{i=1}^{n}(\text{Рост}_i - \overline{\text{Рост}})^2, \tag{5.2c}$$

где $F_{\text{р}}$ – расчетное значение критерия Фишера, представленное в виде отношения дисперсии $S^2_{\text{ад}}$ (называемой дисперсией адекватности и оценивающей степень «разброса» модельных значений Рост_i^{M} от соответствующих значений Рост_i и дисперсии $S^2_{\text{общ}}$ (называемой общей дисперсией случайной величины «Рост» и измеряемой на основе приведенных в табл. 5.1 значений); n – число значений в таблице; Рост_i – значение роста реального человека в i-й записи в графе «Рост»; Рост_i^{M} – модельное значение роста также в i-й записи, найденное по значению «Вес» в этой записи по формуле (5.1); $\overline{\text{Рост}}$ – среднее значение роста, найденное по графе «Рост»; p – количество параметров модели.

Теперь следует выяснить, что понимать под параметром модели. Модель представляет собой некоторую формулу. Имеются определенные классы формул, например класс линейных формул вида

$$Y = a_0 + a_1 x_1 + a_2 x_2 + ... + a_n x_n, \qquad (5.3)$$

где x_i – переменные; a_i – коэффициенты.

Именно коэффициенты и являются параметрами, т. е. неизменными величинами, в отличие от переменных x_i, которые изменяются при переходе от одного опыта к другому. Итак, параметром является постоянная величина, входящая в формулу либо как коэффициент при переменной, либо как свободный член, либо как показатель степени, основание логарифма или предел интегрирования и т. д.

Модель (5.1) является частным случаем модели (5.3). Здесь Y = Рост, a_0 = 100, x_1 = Вес, a_1 = 1. Таким образом, модель (5.1) имеет два параметра: a_0 и a_1. Теперь нетрудно найти дисперсию адекватности $S^2_{\text{ад}}$ и общую дисперсию $S^2_{\text{общ}}$. Имеем,

$$S^2_{\text{ад}} = 45{,}7;\ S^2_{\text{общ}} = 70;\ F_{\text{р}} = \frac{45{,}7}{70} = 0{,}652\,381.$$

Итак, мы определили расчетное значение критерия Фишера $F_{\text{р}}$. Остается сравнить это значение с табличным значением $F_{\text{т}}$, найденным по статистическим таблицам по величинам $k_1 = n - p$ и $k_2 = n - 1$, а также принятого уровня ошибки, например, $\alpha = 0{,}05$. Воспользуемся для этой цели MS Excel (функция FРАСОБР). Получим $F_{\text{т}} = 3{,}6$. Проверяем условие

$$F_{\text{р}} \le F_{\text{т}}. \qquad (5.4)$$

Если оно выполняется, то это означает, что у нас нет оснований не доверять модели. Напротив, если $F_{\text{р}} > F_{\text{т}}$, то модель следует забраковать как негодную с вероятностью ошибки, не превосходящей $0{,}05$. В нашем конкретном случае модель вполне приемлема.

5.2. Критерий χ^2

Критерий χ^2 оперирует двумя рядами чисел: теоретической частотой (или вероятностью) попадания случайной величины в заданные интервалы (группы) и наблюдаемой частотой попадания этой величины в те же интервалы (группы). По степени расхождения частот попадания в интервалы можно судить о близости (соответствии) теоретического и эмпирического распределений. Приведем пример. На вопрос о том, какой цвет любимый, получены следующие данные (табл. 5.2).

Таблица 5.2

Число предпочтений	Цвет		
	Белый	Зеленый	Красный
Эмпирическое	40	25	31
Теоретическое	32	32	32

Сформулируем следующий вопрос: верно ли, что цвета предпочитают в целом равномерно, т. е. (в данном случае) теоретические частоты составят 32 для числа лиц, предпочитающих белый, зеленый и соответственно красный цвета.

Формула для расчета критерия χ^2 имеет такой вид:

$$\chi^2_{\text{р}} = \sum_i \frac{(k_i - k_i^*)}{k_i^*}. \qquad (5.5)$$

где k_i и k_i^* – соответственно опытное и теоретическое количество попаданий в *i*-ю группу.

Получаем

$$\chi^2_{\text{р}} = \frac{(40-32)^2}{32} + \frac{(25-32)^2}{32} + \frac{(31-32)^2}{32} = \frac{114}{32} = 3{,}56.$$

Теперь нужно найти табличное значение $\chi^2_{\text{т}}$. Для принятия гипотезы о равномерном предпочтении цветов необходимо, чтобы $\chi^2_{\text{р}} \le \chi^2_{\text{т}}$.

Табличное значение критерия χ^2 находим по значению принятого уровня, например, $\alpha = 0{,}05$ и числу степеней свободы *S*. Число степеней свободы

$$S = (m-1)(n-1),$$

где m – число строк в таблице; n – число столбцов в таблице (табл. 5.2).

У нас $m = 2$, $n = 3$ и $S = 2$. Для отыскания $\chi^2_т$ воспользуемся MS Excel (функция ХИ2ОБР). Получаем $\chi^2_т = 5{,}99$. Вывод: нет оснований считать, что предпочтения цветов неравномерны.

Лекция 6
ФИЗИЧЕСКИЕ ПРИНЦИПЫ ХРАНЕНИЯ ИНФОРМАЦИИ

Цель. Рассмотреть различные физические принципы записи и хранения информации на носителях.

6.1. Хранение информации на магнитном носителе

Запись на магнитный носитель основана на использовании закона электромагнитной индукции, открытого в 1831 г. Майклом Фарадеем:

$$E = -\frac{d\Phi}{dt}, \qquad (6.1)$$

где Ф – величина магнитного потока, проходящего через поверхность; E – возникающая при этом электродвижущая сила (эдс); t – время.

Из (6.1) видно, что величина эдс прямо пропорциональна скорости изменения магнитного потока.

Для записи информации используют магнитные материалы – ферромагнетики (окись железа, кобальт и др.), обладающие способностью сохранять намагниченное состояние после воздействия электромагнитного поля. Намагничивание производится вдоль узких полос – дорожек – путем воздействия на них изменяющегося магнитного поля в миниатюрных электромагнитных катушках, входящих в записывающие головки. В результате при вращении диска, на поверхность которого нанесен ферромагнитный слой, домены, расположенные вдоль дорожек, намагничиваются определенным образом. Итак, общая картина такова. Ток, несущий сигнал (0 или 1), проходит через электромагнитную катушку головки для записи. При этом

головка позиционируется непосредственно на дорожке вращающегося магнитного диска. В результате определенные домены дорожки намагничиваются одинаковым образом (имеют одинаковую ориентацию спинов).

Чтение информации с дорожки также выполняется в соответствии с законом электромагнитной индукции. При подведении читающей головки к дорожке вращающегося магнитного диска в электромагнитной катушке головки возникает ток, сила которого тем больше, чем больше скорость вращения диска (у современных дисковых накопителей эта скорость составляет около 3000 оборотов в секунду). Производители дисковой памяти для компьютеров обеспечивают возможность записи на магнитный диск до нескольких сотен гигабайт информации (1 байт = 8 бит; 1 Кбайт = 100 бит; 1 Мбайт = 10^6 бит; 1 Гбайт = 10^9 бит). Итак, основными достоинствами магнитной памяти являются следующие:

- можно записывать огромные массивы информации;
- информация после записи сохраняется на диске.

Однако память на магнитном носителе по скорости работы значительно уступает полупроводниковой памяти.

6.2. Полупроводниковая память на транзисторах

Рассмотрим электрическую схему инвертора, реализованную на транзисторе (рис. 6.1).

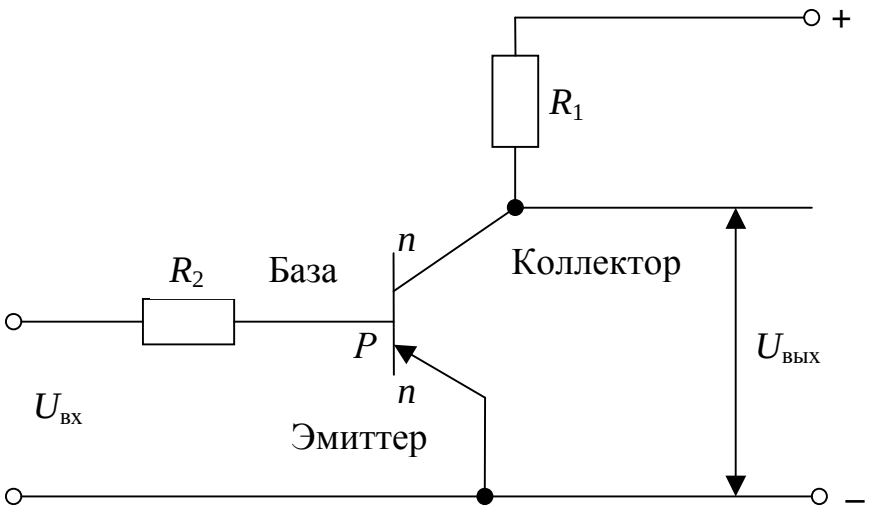
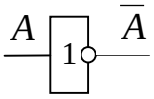


Рис. 6.1

Транзистор – это электрическая схема с двумя *p*-*n*-переходами, включенными встречно. Когда оба *p*-*n*-перехода открыты, то сопротивление транзистора мало, падение напряжения
на нем невелико. Будем считать, что в исходном состоянии $U_{вх} = 0$. Переход «база – коллектор» заперт, так как на коллектор подается положительный потенциал, а в коллекторе – избыток электронов (*n*-тип). Поэтому транзистор заперт. Сопротивление на нем высокое, поэтому высоко и падение напряжения $U_{вых}$ на выходе схемы. Напротив, если подать на вход схемы напряжение $U_{вх}$, достаточное для отпирания транзистора, то оба перехода станут открытыми, поэтому сопротивление транзистора станет малым и $U_{вых}$ резко упадет. Такой принцип работы схемы соответствует инвертору.

Если на входе инвертора высокое напряжение (или ток, что все равно), то на выходе будет низкое напряжение (ток), и наоборот. Функциональным обозначением инвертора является схема
на рис. 6.2.



$$A = 0 \rightarrow \overline{A} = 1;\ A = 1 \rightarrow \overline{A} = 0$$

Рис. 6.2

Эта схема с одним входом. Однако не сложно получить схемы инвертора с двумя, тремя и более входами (рис. 6.3).

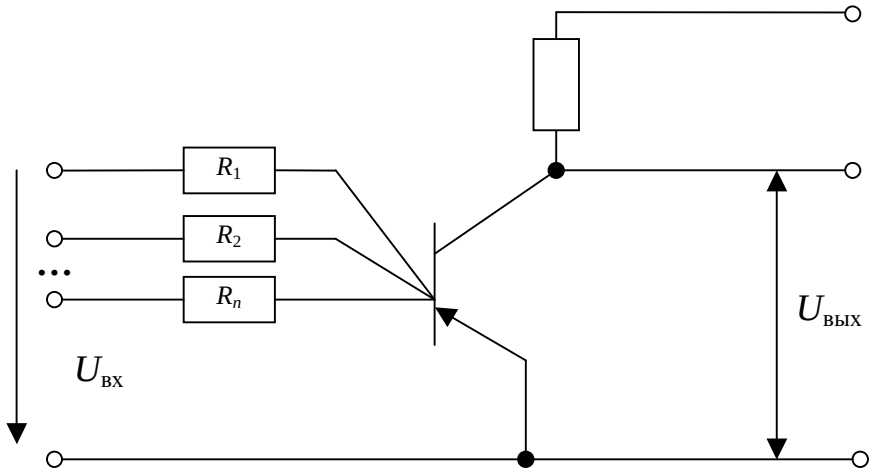


Рис. 6.3

Это достигается простым добавлением одинаковых входных сопротивлений R_i, подключенных параллельно. Функциональное обозначение инвертора с двумя входами показано на рис. 6.4.

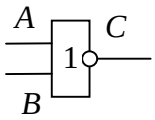


Рис. 6.4

Эта схема имеет высокое напряжение (ток) на выходе *C*, если на каждом из входов *A* и *B* низкое напряжение (ток). Если же на каком-то из входов *A* или *B* (или на обоих одновременно) высокое напряжение (ток), то на выходе *C* тока не будет. Таким образом, логику работы инвертора с двумя входами можно передать с помощью табл. 6.1.

Таблица 6.1

A	*B*	*C*
0	0	1
0	1	0
1	0	0
1	1	0

Примечание. 0 – нет тока; 1 – есть ток.

С помощью инверторов можно реализовать важнейшие функции компьютерной логики. Такими функциями являются функции **И**, **ИЛИ**. Реализация функции **И** должна удовлетворять табл. 6.2.

Таблица 6.2

A	*B*	*A* **И** *B*
0	0	0
0	1	0
1	0	0
1	1	1

Реализация функции **ИЛИ** должна удовлетворять табл. 6.3.

Таблица 6.3

A	*B*	*A* **ИЛИ** *B*
0	0	0
0	1	1
1	0	1
1	1	1

Функциональная реализация схемы «*A* **ИЛИ** *B*» представлена на рис. 6.5.

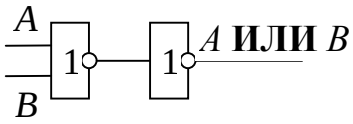


Рис. 6.5

Для реализации схемы «*A* **И** *B*» нужно воспользоваться правилом де Моргана:

A **И** *B* = **НЕ** (**НЕ** *A* **ИЛИ НЕ** *B*)

Здесь **НЕ** означает инвертирование сигнала. Для удобства используют следующие обозначения:

И = «&» (знак амперсанда);

ИЛИ = «∨»;

НЕ = « $\bar{}$ » (черта сверху).

Тогда приведенное выше правило де Моргана запишется так:

$$A \,\&\, B = \overline{\overline{A} \vee \overline{B}}$$

Напомним, что функциональная реализация инвертора показана на рис. 6.2. Таким образом, объединяя рис. 6.2 и 6.3, получаем функциональную схему для реализации операции $A \,\&\, B$ (рис. 6.6).

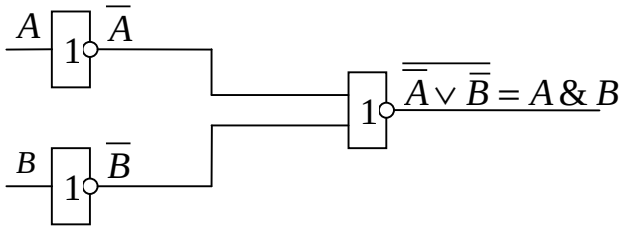


Рис. 6.6

Разумеется, из соображений удобства лучше использовать сокращенные функциональные представления для операций **И** и **ИЛИ** (рис. 6.7).

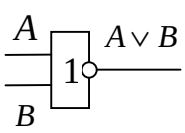


схема **ИЛИ**

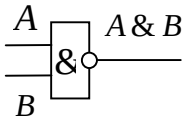


схема **И**

Рис. 6.7

Наконец, мы добрались до следующего важнейшего вопроса: как построить ячейку памяти для хранения сигнала. Принцип работы подобной ячейки элементарен: при подаче на ее вход «0» и сопровождающего управляющего единичного сигнала на выходе ячейки бу-дет «0». Наоборот, при подаче на вход ячейки «1» и единичного управляющего сигнала на выходе ячейки будет постоянно присутствовать «1». Интересующая нас схема называется RS-триггером; она представлена на рис. 6.8. Эта схема содержит только два управляющих входа (S – set, R – reset) и два выхода Q и $\overline{Q}$.

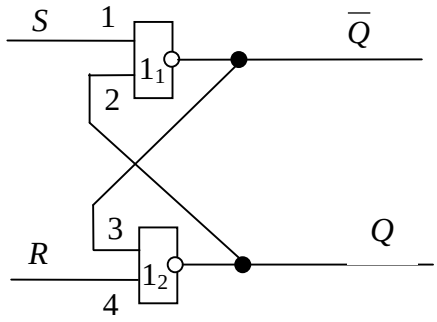


Рис. 6.8

Опишем работу этой схемы. Разрешимыми комбинациями на входах триггера являются: $(S=1,\ R=0)$; $(S=0,\ R=1)$; $(S=0,\ R=0)$. Комбинация $(S=0,\ R=0)$ ничего не изменяет. Комбинация $(S=1,\ R=1)$ является запрещенной. Рассмотрим комбинацию $(S=1,\ R=0)$. На выходе $\overline{Q}$ станет «0», на выходе Q станет «1». После того как сигнал на *S* пропадет, единичный сигнал с выхода *Q* поступает на вход 2 схемы 1_1 и обеспечивает «0» на выходе $\overline{Q}$, на выходе Q по-прежнему остается «1». Теперь пусть на входе триггера присутствует комбинация $(S=0,\ R=1)$, тогда на выходе Q станет «0», а на выходе $\overline{Q}$ – «1» и все изменится с точностью да наоборот.

Таким образом, простейшая схема на рис. 6.8 выполняет функцию запоминающего устройства и иллюстрирует принцип работы полупроводниковой памяти. Особенностями полупроводниковой памяти являются следующие:

- работает значительно быстрее магнитной;
- не обеспечивает сохранения информации после отключения питания.

6.3. Хранение информации на лазерных компакт-дисках (CD)

Лазерные компакт-диски напоминают обычные магнитные диски. Информация записывается и читается с дорожек читающей головкой. Обычная емкость компакт-диска составляет 650 Мбайт, его диаметр 12 см, толщина 1,2 мм. Компакт-диски изготавливаются из пластиковой основы, на которую наносится слой алюминия, покрываемый защитным слоем лака. Записываемая на дорожку информация представляет собой чередование ямок и плоских мест.

При записи информации ямки прожигаются в алюминиевом основании лазерным лучом. Логический «0» соответствует ямке, а логическая «1» – ровному месту на дорожке. При чтении информации лазерный луч направляется на дорожку и фокусируется с помощью управляющей катушки. Луч проникает через слой лака и отражается от алюминиевой поверхности. Характер отражения от плоского места и от ямки различный. При отражении от ямки луч рассеивается; при отражении от плоского места – нет. Отражающийся луч попадает через призму на светочувствительный диод, который преобразует световые импульсы в импульсы электрического тока. Скорость считывания информации с компакт-диска варьируется в диапазоне от 150 до 1200 Кбайт/с.

6.4. Принцип работы Flash-памяти

Flash-память представляет собой полупроводниковую микросхему, в которой, однако, информация сохраняется после отключения питания.

Ячейку Flash-памяти можно представить так, как показано на рис. 6.9. Эта ячейка обеспечивает хранение одного бита информации. Важной особенностью этой схемы является способность удерживать заряд на плавающем затворе. Управляющий затвор играет роль «базы». При подаче положительного напряжения на управляющий затвор между «стоком» и «истоком» начинается движение электронов (показано стрелкой). Часть этих электронов проходит через слой изолятора и попадает в область плавающего затвора. В этой области они накапливаются и могут храниться длительное время (несколько лет). Количество электронов, накопленное в плавающем затворе, зависит от уровня напряжения на управляющем затворе. При положительном напряжении на управляющем затворе число накапливаемых электронов велико, при отрицательном – мало. Таким образом, логический «0» и «1» отличаются числом электронов, накопленных в плавающем затворе. При считывании данных «0» и «1» различаются уровнем потенциала на плавающем затворе. Для стирания информации на плавающий затвор ячейки подается отрицательное напряжение, вследствие чего электроны с плавающего затвора через изолятор уходят на «исток».

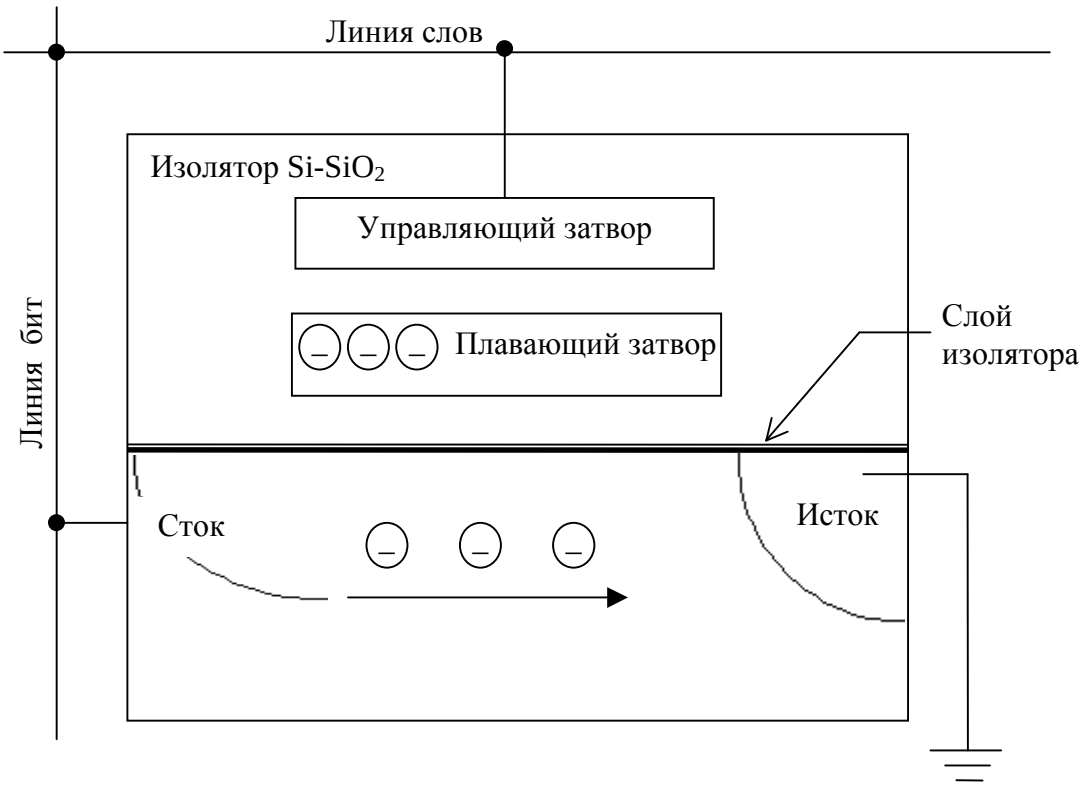


Рис. 6.9

Достоинством Flash-памяти является ее высокая компактность и быстродействие, характерное для полупроводниковой памяти. Современные Flash-микросхемы могут хранить несколько сотен мегабайт информации.

6.5. Голографическая память

Теоретические принципы голографии заложил американец венгерского происхождения Денис Габор. Он же и получил первые голограммы в 1948 г. Голограмма – это картинка, которая получается в фотоэмульсионном слое при наложении луча лазера на луч лазера с той же длиной волны, отраженный от объекта. Такое взаимодействие лучей называется интерференцией. Таким образом, голограмма – это набор интерференционных колец. Схема получения голограммы приведена на рис. 6.10.

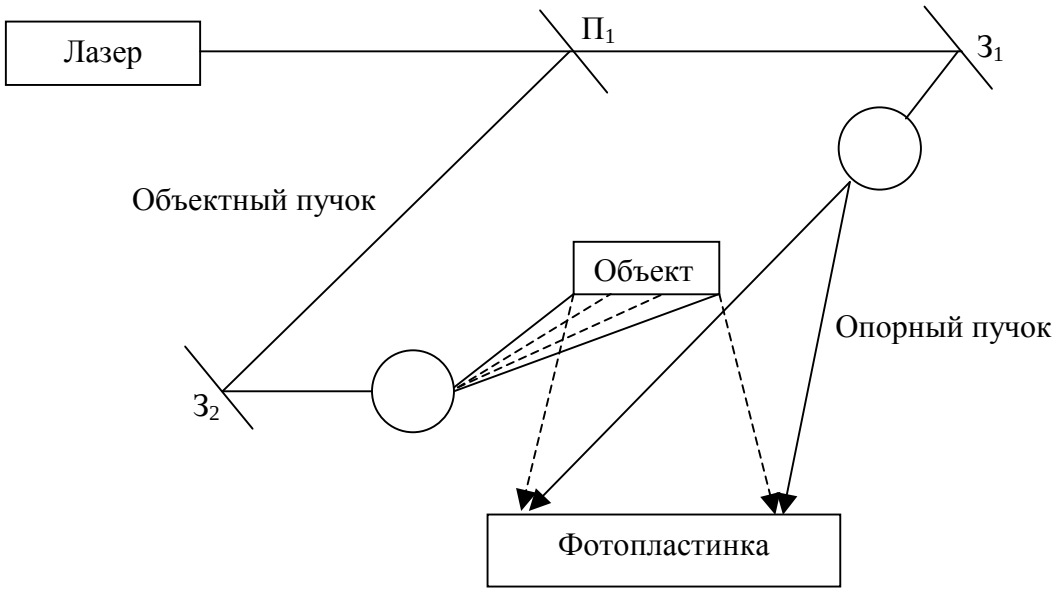


Рис. 6.10

Лазерный луч расщепляется призмой $П_1$ на два луча. Объектный пучок от зеркала $З_2$, отражаясь, попадает на объект и далее, отражаясь от объекта, попадает на фотопластинку. С другой стороны опорный пучок отражается от зеркала $З_1$ и тоже попадает на фотопластинку. Таким образом, объектный пучок и опорный пучок интерферируют и на фотопластинке отображаются интерференционные кольца. По сути, это и есть голографический образ объекта.

Для того чтобы воспроизвести записанное изображение, т. е. получить изображение объекта, применяют схему, показанную на рис. 6.11. В этой схеме на фотопластинку с записанными интерференционными кольцами падает лазерный пучок, и наблюдатель видит мнимое изображение объекта.

Идея использования голографической памяти в ЭВМ была озвучена еще в 1960-х гг. Принцип голографической записи/чтения не изменяется. Все дело в объекте, в качестве которого используется экран жидкокристаллического дисплея. При этом на экране светится матрица, состоящая из светлых и темных ячеек.

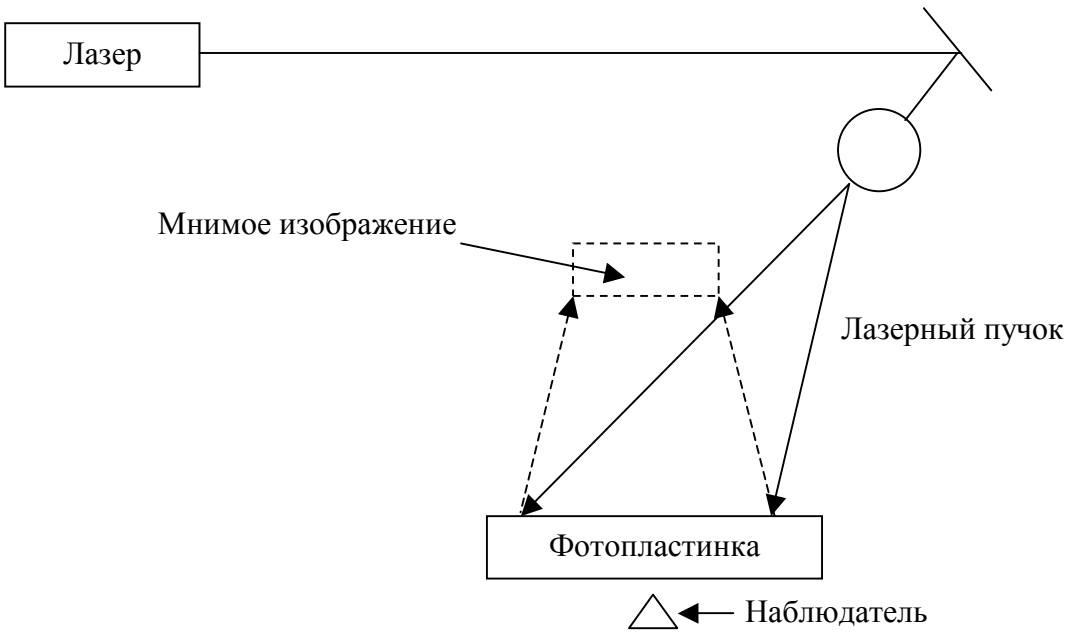


Рис. 6.11

Светлая ячейка соответствует «1», темная – «0». Голограмма обеспечивает сохранение содержимого экрана (вообще говоря, кристалл размером с кусок сахара способен хранить голограммы, содержащие до 1000 Гбайт информации).

Лекция 7
ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ЭВМ

Ц е л ь . Рассмотреть основные позиционные системы счисления и правила перехода из одной системы счисления в другую.

7.1. Позиционные системы счисления

Следует иметь в виду, что вся информация может быть условно разбита на следующие категории:

- числовая;
- текстовая;
- графическая.

Две последние категории можно свести к числовой. В позиционной системе счисления число X (целое) можно представить в виде

$$X = a_0 p^0 + a_1 p^1 + a_2 p^2 + ... + a_k p^k. \qquad (7.1)$$

Здесь каждое a_i – целое, причем $a_i < p$. Число p называется основанием системы счисления. В качестве основания p в ЭВМ используют 2, 8, 16. Например,

$$10 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0; \qquad (7.2a)$$

$$10 = 1 \cdot 8^1 + 2 \cdot 8^0; \qquad (7.2b)$$

$$10 = 10 \cdot 16^0. \qquad (7.2c)$$

Формулу (7.1) можно записать короче:

$$X = a_k a_{k-1} a_{k-2} ... a_1 a_0. \qquad (7.3)$$

В представлении (7.3) коэффициенты a_i перечисляются слева направо, начиная со старших индексов i. Например,

10 = 1010 (по основанию 2);

10 = 12 (по основанию 8);

10 = 10 (по основанию 16).

Двоичная система счисления ($p = 2$) играет главенствующую роль, поскольку технически наиболее просто реализовать

устройства с двумя состояниями. Так, транзистор может находиться
в открытом или закрытом состоянии, ферромагнетик может быть намагничен или размагничен, на входе (выходе) схемы сигнал может присутствовать или отсутствовать и т. п.

Рассмотрим, как получить число в двоичном представлении по заданному десятичному целому числу X. Опять, пусть $X = 10$. Алгоритм построения двоичного числа, равного данному десятичному числу, состоит в многократном выполнении одних и тех же действий. Делим X на два и выписываем остаток r и частное p. Если это действие выполняется на шаге i, то более удобно обозначить данные величины через r_i, p_i. Так, для $i = 0$ имеем:

$$r_0 = 0;\ p_0 = 5.$$

Далее, делим p_i на 2 и выписываем остаток r_{i+1} и частное p_{i+1}:

$$r_1 = 1,\ p_1 = 2 \text{ (частное от деления 5 на 2).}$$

Повторяем процедуру:

$$r_2 = 0;\ p_2 = 1.$$

$$r_3 = 1;\ p_3 = 0.$$

Очевидным образом процесс деления завершается тогда, когда $p_i = 0$. Теперь результирующее двоичное представление числа X получим в виде последовательности остатков $r_3 r_2 r_1 r_0$ (1010). Схематически процесс деления показан на рис. 7.1.

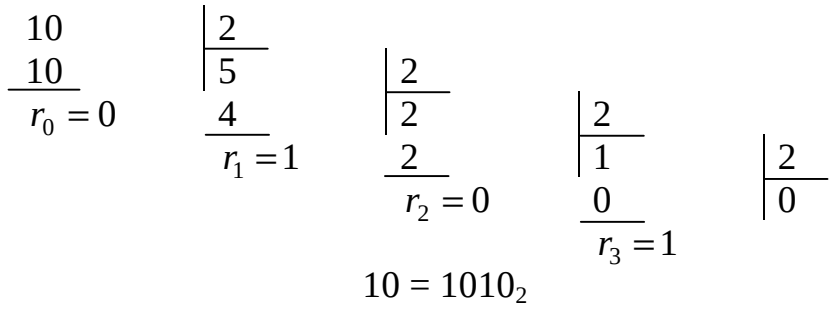


Рис. 7.1

Алгоритм деления с получением остатков применим для получения представления целого десятичного числа для любого целого положительного основания p. Например, пусть $p = 3$,

$X = 28$. Получение результирующего представления показано на рис. 7.2.

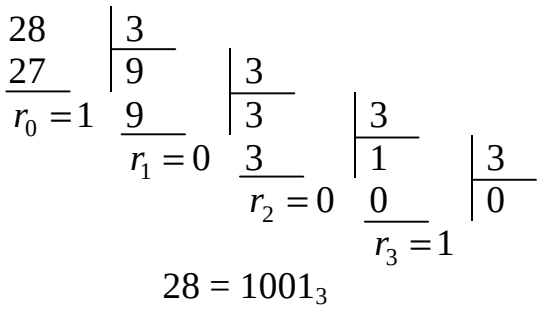


Рис. 7.2

Очень просто переводить двоичные числа в восьмеричные и шестнадцатеричные и наоборот. Для перевода двоичного числа в восьмеричные его нужно разбить на тройки (триады) битов и каждую тройку битов заменить соответствующим восьмеричным числом. Если битов не хватает, то двоичное число дополняют слева нужным числом нулей. Для простоты оперирования тройками битов приведем их десятичные эквиваленты:

000 – 0	100 – 4
001 – 1	101 – 5
010 – 2	110 – 6
011 – 3	111 – 7

(ясно, что в случае троек битов это будут соответствующие восьмеричные эквиваленты).

Пример. Пусть дано двоичное число 0101001. Для разбиения на тройки допишем слева два нуля:

$$\underbrace{000}\ \underbrace{101}\ \underbrace{001}$$

Заменяем каждую тройку эквивалентным троичным числом из (2.5):

$$051_8$$

Итак, 51_8 и есть восьмеричный эквивалент приведенного выше двоичного числа 0101001.

Для получения шестнадцатеричного числа используют не тройки, а четверки двоичных разрядов и таблицу эквивалентов:

0000 – 0	0100 – 4	1000 – 8	1100 – C
0001 – 1	0101 – 5	1001 – 9	1101 – D
0010 – 2	0110 – 6	1010 – A	1110 – E
0011 – 3	0111 – 7	1011 – B	1111 – F

Теперь наше двоичное число разбиваем на четверки таким образом:

$$\underbrace{0010}\ \underbrace{1001}$$

что дает шестнадцатеричный эквивалент – 29_{16}.

Числа с фиксированной точкой (дробной частью) переводятся в двоичное представление двумя способами. Наиболее простое – это так называемое экспоненциальное представление, например:

$$2{,}53 = 253 \cdot 10^{-2},$$

$$-270{,}132 = -270132 \cdot 10^{-3}.$$

Видим, что для представления числа с фиксированной дробной частью используется целая часть – мантисса и экспоненциальная часть (10^{-k}), или просто – экспонента. И мантисса, и экспонента являются целыми числами. Поэтому в данном случае для представления исходного числа нужно два двоичных числа: мантисса и экспонента.

Другой очевидный способ состоит в расширении представления (7.1) до (7.4):

$$X = a_0 p^0 + a_1 p^1 + \ldots + a_k p^k + a_{-1} p^{-1} + \ldots + a_{-m} p^{-m}. \qquad (7.4)$$

В любом случае и здесь нам потребуется для хранения X использовать два числа:

$$a_k a_{k-1} a_{k-2} \ldots a_1 a_0;$$

$$a_{-1} a_{-2} \ldots a_{-m}.$$

Разумеется, можно хранить отдельно целую часть числа и его дробную часть.

7.2. Машинные форматы текстовой и числовой информации

Для хранения текстовой информации в ЭВМ используют систему кодировки ASCII (1 байт на символ) и UniCode (2 байта на символ). Каждый символ, вводимый с клавиатуры, имеет собственный код, например:

32 – код пробела,

13 – код клавиши ENTER

27 – код клавиши ESCAPE и пр.

Коды чисел отличаются от самих чисел. Так, ASCII код 7 соответствует гудку (звуковому сигналу), а не числу 7. Для различения символов-чисел и настоящих чисел принято символьные представления брать в кавычки. Так, ‘7’ есть символ семерки, а 7 есть чис-ло семь.

По-разному хранятся отрицательные и положительные числа. В частности, первым разрядом (знаковым разрядом) положительного числа является 0, а отрицательного – 1. Существенна также и ширина ячеек, хранящих числа. Короткие положительные числа требуют 1 байт (1 байт = 8 бит). Обычно целые числа со знаком требуют для хранения 4 байта; вещественные – 8 байт.

7.3. Представление графической информации

Для кодирования изображений используют различные форматы. Рассмотрим три таких формата (модели): RGB, YCbCr, CMYK. Прежде всего, заметим, что изображение представляет собой матрицу, состоящую из огромного числа точек, называемых пикселами (pixel). Для каждого пиксела нужно запомнить цвет. В формате RGB любой цвет получают как сумму трех составляющих: R (red – красный), G (green – зеленый), B (blue – синий). Каждая составляющая передается числом от 0 до 255. Это число задает яркость. Например:

RGB (0, 0, 0) = черный цвет;

RGB (255, 255, 255) = белый цвет;

RGB (255, 255, 0) = желтый цвет;

RGB (255, 0, 255) = пурпурный цвет и т. д.

В цветовой модели YCbCr Y представляет яркость изображения в целом. Значение Cb задает синеву изображения, а Cr – красноту. Связь между RGB и YCbCr однозначно передается следующими соотношениями:

$$\begin{aligned} Y &= 0{,}299R + 0{,}587G + 0{,}114B; \\ Cb &= -0{,}1687R - 0{,}313G + 0{,}5B + 2^8; \\ Cr &= 0{,}5R - 0{,}4187G - 0{,}0813B + 2^8. \end{aligned} \qquad (7.5)$$

$$R = Y + 1{,}402Cr;$$
$$G = Y - 0{,}34414(Cb - 2^8) - 0{,}71414(Cr - 2^8); \quad (7.6)$$
$$B = Y + 1{,}7222(Cb - 2^8).$$

Сразу заметим, что в цветовой модели YCbCr основную (большую часть) информацию несет компонент Y. Это позволяет, в частности, выполнять более сильное сжатие файлов изображений в формате YCbCr (например, JPEG).

Еще одной цветовой моделью является CMYK (cyan, magenta, yellow, black – голубой, пурпурный, желтый, черный). Эта модель используется для печати цветных документов. Связь между CMYK и RGB задается следующими соотношениями:

$$K = (2^8 - 1) - MAX(R, G, B);$$
$$C = (2^8 - 1) - R - K;$$
$$Y = (2^8 - 1) - G - K; \quad (7.7)$$
$$M = (2^8 - 1) - B - K.$$

Модель CMYK подобрана таким образом, что если C = M = Y, то мы видим тот или иной оттенок серого цвета.

Лекция 8
ФУРЬЕ-ПРЕДСТАВЛЕНИЕ СЛОЖНЫХ СИГНАЛОВ

Цель. Изложить механизм Фурье-представления аналитического сигнала с помощью конечного и бесконечного спектра.

Фурье показал, что любой непрерывный ограниченный сложный периодический сигнал $s(t)$ с периодом $2T$ может быть представлен в виде бесконечной суммы простейших гармоник:

$$s(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos \omega_n t + b_n \sin \omega_n t), \tag{8.1}$$

где ω_n – угловая скорость n-й гармоники;

$a_0, a_1, ..., a_l, b_1, b_2, ..., b_l$ – коэффициенты, соответствующие амплитудам гармоник:

$$a_0 = \frac{1}{T} \int_{-T}^{T} s(t)\ dt; \tag{8.2}$$

$$a_l = \frac{1}{T} \int_{-T}^{T} s(t) \cos \omega_l t\ dt; \tag{8.3}$$

$$b_l = \frac{1}{T} \int_{-T}^{T} s(t) \sin \omega_l t\ dt. \tag{8.4}$$

Эти коэффициенты в своей совокупности называются *спектром сигнала s(t)*. Важным обстоятельством является тот факт, что можно получить отличную аппроксимацию (приближение) сигнала $s(t)$, даже если ограничиться конечной суммой первых n гармоник
в (8.1). Таким образом, можно воспользоваться представлением Фурье (8.1) следующим образом. Напомним, что угловая скорость ω_l связана с частотой l-й гармоники соотношением

$$\omega_l = 2\pi f_l,$$

но

$$\omega_l = l\omega = l2\pi f = l2\pi\frac{1}{2T} = \frac{l\pi}{T}.$$

Это позволяет переписать (8.3) и (8.4) таким образом:

$$a_l = \frac{1}{T}\int_{-T}^{T} s(t)\cos\frac{\pi l t}{T}\ dt; \tag{8.5}$$

$$b_l = \frac{1}{T}\int_{-T}^{T} s(t)\sin\frac{\pi l t}{T}\ dt. \tag{8.6}$$

Соотношения (8.2), (8.5), (8.6) записаны в интегральной форме. Воспользуемся теперь приближенным представлением интеграла в виде суммы

$$\int_a^b f(x)\,dx = \sum_{i=1}^{M} f(x_i)\Delta x, \tag{8.7}$$

где $x_i = a + (i-1)\Delta x$;

$$M = \frac{b-a}{\Delta x}.$$

Величина Δx должна быть достаточно малой. Ее выбор определяется теоремой Котельникова:

$$\Delta x \le \frac{1}{2 f_{\max}}, \tag{8.8}$$

В (8.8) величина $f_{\max}$ определяется максимальной частотой гармоники в конечном спектре сигнала. От того, сколько гармоник включено в конечный спектр сигнала, зависит точность его аппроксимации (приближенного представления). Применим формулу (8.7) к (8.2), (8.5), (8.6). Получим следующие дискретные соотношения для коэффициентов гармоник:

$$a_0 = \frac{1}{T}\sum_{i=1}^{M} s(t_i)\Delta t; \tag{8.9}$$

$$a_l = \frac{1}{T}\sum_{i=1}^{M} s(t_i)\cos\frac{\pi l t_i}{T}\Delta t; \tag{8.10}$$

$$b_l = \frac{1}{T}\sum_{i=1}^{M} s(t_i)\sin\frac{\pi l t_i}{T}\Delta t, \qquad (8.11)$$

где $t_i = -T + (i-1)\Delta t$,

$$M = \frac{2T}{\Delta t},$$

$\Delta t \le \dfrac{1}{2 f_{\max}} = \dfrac{\pi}{\omega_{\max}} = \dfrac{\pi}{\omega k}$ (k – номер максимальной гармоники).

Применение (8.9)–(8.11) состоит в следующем. В таблицу заносят значения сложного сигнала, измеренные на отрезке [0; 2T] в точках $t_i = 0, \Delta t, 2\Delta t, ..., N\Delta t$, где Δt определяется из теоремы Котельникова, т. е. $\Delta t \le \dfrac{1}{2 f_{\max}}$. Таким образом, имеем значения $s(t_i)$ в точках t_i. Из (8.9)–(8.11) находим коэффициенты a_0, a_l, b_l (l = 1, 2, …, k; k – номер максимальной гармоники). Теперь вместо того, чтобы хранить огромное число точек $s(t_i)$ достаточно хранить коэффициенты a_0, a_l, b_l, по которым всегда можно восстановить форму сигнала $s(t)$ по формуле (8.1).

Итак, еще раз отметим следующий факт: сложный сигнал характеризуется своим спектром. Спектр выполняет своего рода роль «визитной карточки» сигнала. По спектру можно выполнить опознание сигнала и, следовательно, источника сигнала. Ряд Фурье дает возможность определять спектр периодического сигнала. Для непериодического сигнала используется не ряд, а интеграл Фурье:

$$f(t) = \int_0^{\infty} \left[a(\omega)\cos\omega t + b(\omega)\sin\omega t \right] d\omega. \qquad (8.12)$$

Воспользуемся формулой Эйлера:

$$\cos\omega t = \frac{e^{j\omega t} + e^{-j\omega t}}{2};$$

$$\sin\omega t = \frac{e^{j\omega t} - e^{-j\omega t}}{2j};$$

$$e^{j\omega t} = \cos\omega t + j\sin\omega t.$$

Отсюда имеем

$$a(\omega)\cos\omega t + b(\omega)\sin\omega t = a(\omega)\frac{e^{j\omega t}+e^{-j\omega t}}{2} + b(\omega)\frac{e^{j\omega t}-e^{-j\omega t}}{2j} =$$

$$= a(\omega)\frac{e^{j\omega t}+e^{-j\omega t}}{2} - jb(\omega)\frac{e^{j\omega t}-e^{-j\omega t}}{2} =$$

$$= \frac{a(\omega)-jb(\omega)}{2}e^{j\omega t} - \frac{a(\omega)+jb(\omega)}{2}e^{-j\omega t} = c(\omega)e^{j\omega t} + c(-\omega)e^{-j\omega t},$$

где

$$c(\omega) = \frac{a(\omega)-jb(\omega)}{2};$$

$$c(-\omega) = \frac{a(\omega)+jb(\omega)}{2}.$$

Таким образом,

$$f(t) = \int_{-\infty}^{\infty} c(\omega)e^{j\omega t}d\omega. \qquad (8.13)$$

Это выражение известно как комплексная форма интеграла Фурье. При этом

$$c(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt\,. \qquad (8.14)$$

Интеграл

$$\&(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \qquad (8.15)$$

называется *спектральной плотностью* функции $f(t)$. Таким образом, вместо спектра, который был определен для периодического сигнала, мы имеем спектральную плотность $\&(\omega)$ для непериодического сигнала.

Пример. Найдем спектральную плотность сигнала:

$$f(t) = \begin{cases} 1, & \text{при } t > 0, \\ 0, & \text{при } t \le 0. \end{cases}$$

Получаем

$$\&(\omega) = \int_{-\infty}^{\infty} 1 e^{-j\omega t}\, dt = -\frac{1}{j\omega}\int_{0}^{\infty} e^{-j\omega t}\, d(-j\omega t) = \frac{j}{\omega} e^{-j\omega t}\Big|_{0}^{\infty} = 0 - j\frac{1}{\omega} = \frac{1}{\omega} e^{-j\frac{\pi}{2}}.$$

Лекция 9
СЛОЖНЫЕ СТРУКТУРЫ ДАННЫХ

Ц е л ь . Рассмотреть сложные структуры данных, такие как деревья и стеки, используемые в современных автоматизированных системах.

Помимо чисел и текста, имеются другие, более сложные структуры данных. Среди них следует отметить деревья, списки и массивы, а также стеки и очереди. Изучение списков и массивов вынесено в следующую лекцию. К знакомству с остальными видами данных мы непосредственно и переходим.

9.1. Деревья

Нам уже доводилось строить диагностическое дерево ранее, поэтому представление о том, что такое дерево, уже есть. Формально определим дерево следующим образом. Обозначим $M = \{m_1, m_2, ..., m_n\}$ – множество вершин (графически представленных кружками). Далее, обозначим $U = \{u_{ij}\}$ – множество дуг (направленных ребер), причем $i \in \{1, n\}$, $j \in \{1, n\}$. Если u_{ij} входит в U, то говорят, что имеется связь от вершины i к вершине j (т. е. m_i и m_j соединены дугой). Пара (M, U) называется *ориентированным графом* [13]. Если ребра не ориентированы, то прилагательное «ориентированный» в определении графа опускают.

Например, пусть

$$M = \{m_1,\ m_2,\ m_3,\ m_4\};\ U = \{u_{1,1},\ u_{1,3},\ u_{1,2},\ u_{3,4},\ u_{4,3},\ u_{3,2}\}.$$

Соответствующий ориентированный граф показан на рис. 9.1.

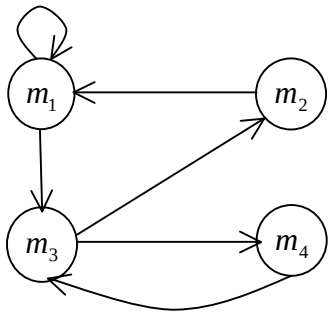


Рис. 9.1

Определение. Если в графе все ребра не ориентированы, то граф называется *неориентированным*.

Определение. Ребро, связывающее вершину саму с собой, называется *петлей*.

Определение. *Цепью* в графе называется последовательность ребер, такая, что каждая соседняя пара ребер в этой последовательности имеет общую вершину и ни одно ребро не образует петлю [14].

Цепь в ориентированном графе называется путем при условии, что любые две соседние дуги одинаково направлены.

Определение. *Неориентированным деревом* называется граф без петель, в котором каждые две вершины соединены одной и только одной цепью [14].

Принято одну вершину дерева выделить и считать *корневой* (главной). В ориентированном дереве, каждая вершина, не являющаяся корневой, соединена единственной цепью, исходящей из корневой вершины. Если специально не оговорено, то далее под деревом понимаем неориентированное дерево.

Пример дерева показан на рис. 9.2.

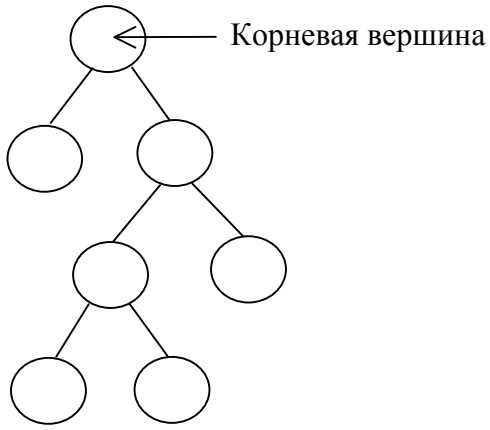


Рис. 9.2

Деревья применяются очень широко. С их помощью отображают иерархические структуры данных, например

организационную структуру предприятия (рис. 9.3). Такая структура характерна для иерархических баз данных.

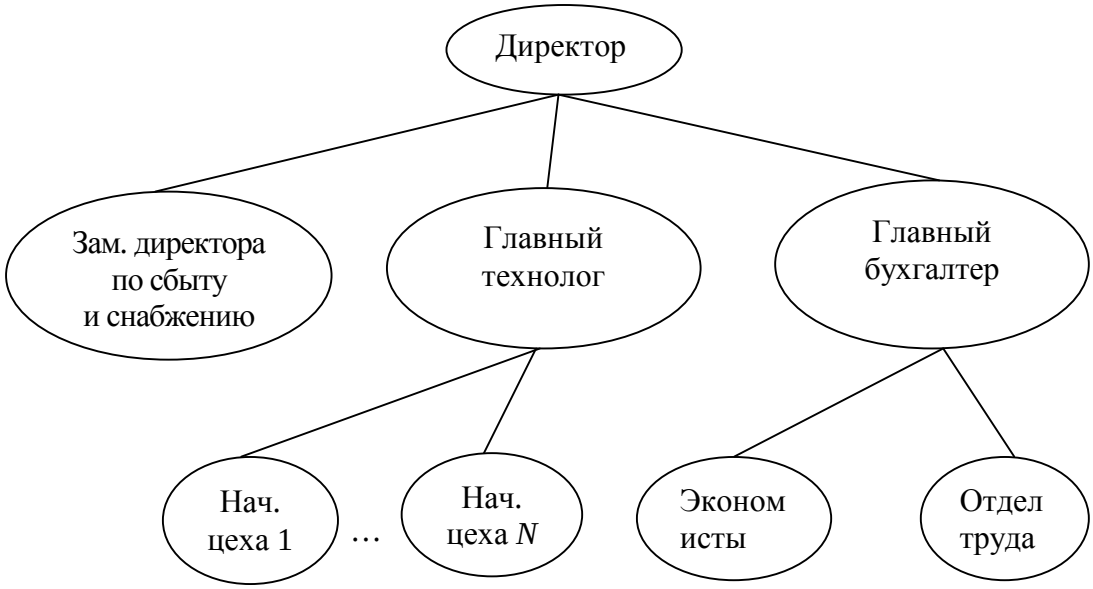
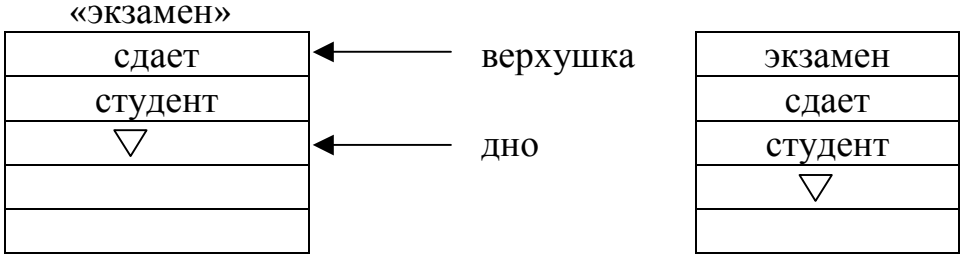


Рис. 9.3

Деревья используют для разбора языков программирования, для ускорения поиска, управления памятью и пр.

9.2. Стеки и очереди

Стек – это вид памяти, представляющий собой набор последовательно расположенных ячеек, в котором первая ячейка называется верхушкой стека, а последняя – дном. Занесение очередного слова информации в стек выполняется через его верхушку. При этом содержимое каждой последующей ячейки проталкивается вглубь на следующую ячейку, что иллюстрируется рис. 9.4.



До записи слова «экзамен» После записи слова «экзамен»

Рис. 9.4

Обратная операция чтения из стека выполняется путем выталкивания слова за словом из стека. Заметим, что доступным в стеке является слово, находящиеся в его верхушке.

Стеки имеют разнообразное применение. Одно из них связано с обработкой арифметических выражений при компиляции программ. Обычные арифметические выражения записываются в привычной нам инфиксной форме (знак операции стоит между операндами). Обработка выражения типа

$$((2+3,6)*4–3)*(3,2+4,3*1,5) \qquad (9.1)$$

затрудняется ввиду наличия скобок. Поэтому подобные обычные арифметические выражения сначала переводят в бесскобочную форму, называемую польской инверсной записью. Смысл такой записи состоит в том, что знак операции записывается после операндов, а скобки становятся ненужными. Например, a + b приводят к виду ab+. Выражение 2*c – 4 в польской записи получает форму с 2*4–. Выражение (9.1) получит такой вид

$$2\ \ 3,6+4*3–3,2\ \ 4,3\ \ 1,5*+*. \qquad (9.2)$$

Стек используется как для получения польской записи типа (9.2) из (9.1), так и для вычисления значения выражения типа (9.1).

Начнем с вычисления выражений типа (9.1). Каждый элемент входной строки, начиная с первого (самого левого), анализируется на предмет проверки того, является ли этот элемент числом или переменной, либо он есть знак операции. Если анализируемый элемент является числом или переменной, то он загружается в стек. Если же анализируемый элемент является знаком операции, то он приводит к выгрузке из стека стольких операндов, сколько операндов участвует в операции. Выполняется операция, а ее результат загружается в стек. Эти действия выполняются для каждого элемента входной строки.

Результирующее значение входной строки будет находиться в верхушке стека. Обработка строки (9.2) приведена на рис. 9.5.

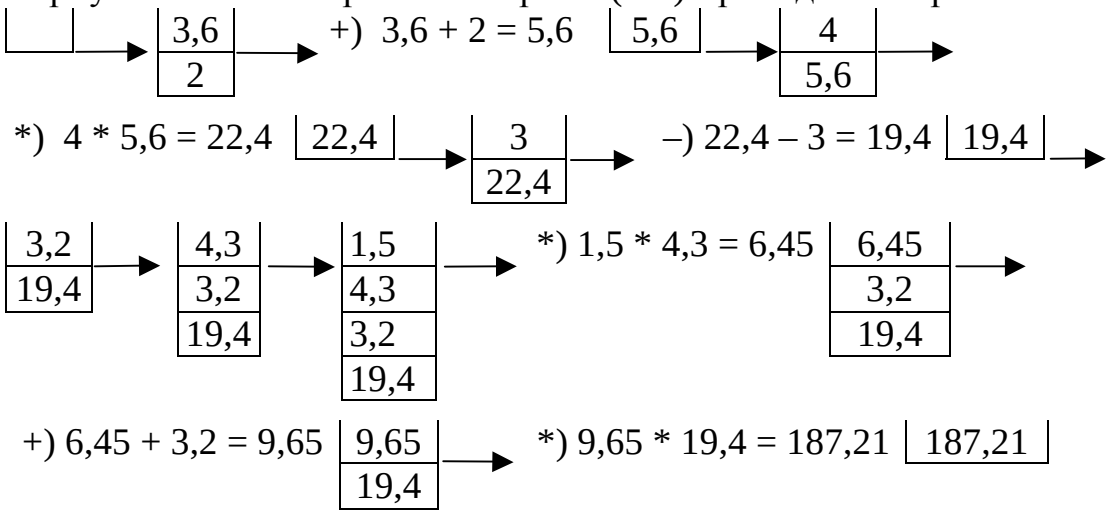


Рис. 9.5

Наконец, рассмотрим, как получить польскую инверсную запись. Используются следующие правила.

1. Если очередной элемент строки – число или переменная, то этот элемент поступает в конец получаемой (формируемой) записи.

2. Если очередной элемент – знак операции, то производится попытка загрузить его в стек. При этом если в верхушке находится менее приоритетная или равная по приоритету операция или скобка, то производится запись в стек. Заметим, что возведение в степень имеет наивысший приоритет; далее следуют умножение и деление (имеющие одинаковый приоритет), затем сложение и вычитание (также имеющие одинаковый приоритет). Скобка имеет наименьший приоритет.

3. Если очередной элемент – знак операции и ее приоритет ниже приоритета операции в верхушке стека, то производится последовательная выгрузка символов операций из стека в конец формируемой строки до тех пор, пока очередной символ операции не будет иметь более низкий приоритет, либо стек не опустеет.

4. Открывающая скобка всегда заносится в стек, независимо от того, что находится в верхушке стека.

5. Если приходит закрывающая скобка, то выполняется выгрузка содержимого стека до открывающей скобки, спаренной с этой закрывающей скобкой. Выгружаемое содержимое стека

ставится в конец формируемой строки. Скобки просто сбрасываются.

6. Если вся строка обработана, то содержимое стека выгружается в конец строки.

Обработку выражения (9.1) согласно приведенным правилам иллюстрирует рис. 9.6.

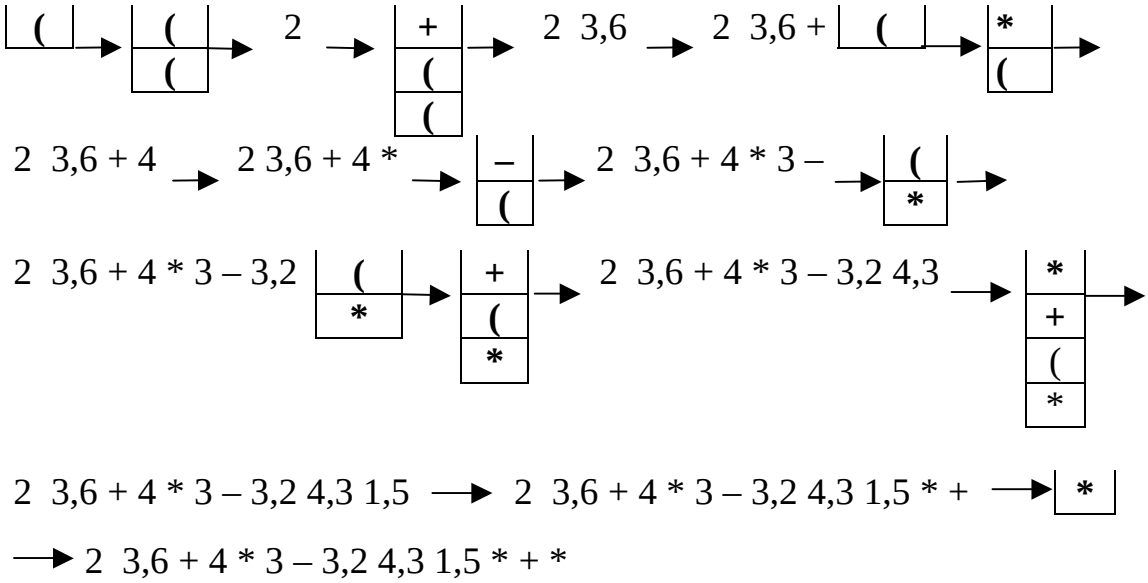


Рис. 9.6

Очередь – это модифицированный стек. Элементы записываются в очередь через верхушку, а считываются со дна. Очереди используются для управления процессами в ЭВМ. Широко используется циклическая очередь, в которой верхушка и дно находятся в одной и той же ячейке.

Лекция 10
СПИСКИ И МАССИВЫ

Ц е л ь. Рассмотреть рекурсивный способ определения списков и массивов, а также методы их спецификации.

Список – это именованное упорядоченное множество $N \geq 0$ элементов, каждый из которых имеет уникальный номер из диапазона $[1, 2, ..., N]$, и никакие два элемента не имеют одного и того же номера. Пусть F – список фамилий, содержащий три фамилии:
$F(1)$ = «Иванов», $F(2)$ = «Петров», $F(3)$ = «Сидоров». Обращение к элементам списка F выполняется через обозначение $F(i)$, где i – номер элемента. Число элементов списка F обозначается как $|F|$ или size(F) или length(F). Обозначением пустого списка служит ∅. Этот знак вообще служит для обозначения любого пустого множества. Первый элемент списка F часто обозначается как head(F). Часть элементов списка без первого обозначается как tail(F). Операция присоединения одного списка G к концу другого списка F записывается так: $F||G$ или $F+G$. Часто находит применение операция расщепления списка на левый и правый подсписки по элементу i. Такую операцию обозначим **:** (F, i). Запись

$$(G, H) = \mathbf{:}\ (F, i)$$

означает, что список F расщепляется на списки G и H следующим образом:

$$G = [F(1), F(2), \ldots, F(i)];$$

$$H = [F(i+1), F(i+2), \ldots, F(|F|)].$$

Наконец, упорядочение списка F по возрастанию его элементов задают как sort(F).

Списки весьма полезны в языках программирования для задания так называемых рекурсивных определений. Познакомимся с понятием рекурсивного определения на примере. Пусть дан следующий список F:

$$F = [1, 3, 4, -2, -3, 12].$$

Попробуем рекурсивно определить условия отсортированности списка: sorted.

Будем считать, что пустой список и список, состоящий из одного единственного элемента, всегда упорядочены. Имеем

$$\text{sorted}(F) \text{ если и только если } \begin{cases} |F| \le 1, \\ \text{head}(F) \le \text{head}(\text{tail}(F)), \\ \text{И sorted}(\text{tail}(F)). \end{cases}$$

Приведенное нами определение является рекурсивным, поскольку оно определяет, что список F отсортирован через условия, относящиеся к самому этому списку.

Законность данного определения следует проверить последовательно. Проверяем условие

$$\text{head}(F) \le \text{head}(\text{tail}(F)) \text{ И sorted}(\text{tail}(F))$$

или

$$1 \le 3 \text{ И sorted}([3, 4, -2, -3, 12]).$$

Здесь содержится условие sorted([3, 4, –2, –3, 12]), относящееся к хвостовой части списка. Это условие сводится к следующим:

$$3 \le 4 \text{ И sorted}([4, -2, -3, 12]).$$

В свою очередь, условие sorted([4, –2, –3, 12]) сводится к

$$4 \le -2 \text{ И sorted}([-2, -3, 12]),$$

что неверно, ибо 4 > –2. Поэтому получаем общий ответ, что список F не отсортирован. В качестве упражнения попытайтесь по аналогии определить наименьший (наибольший) элемент списка и сумму элементов списка, используя рекурсию.

Многомерный массив – это множество элементов, определяемых несколькими номерами: если массив двумерный, то двумя номерами, если трехмерный – тремя и т. д. Двумерные массивы называются *матрицами*. Элементы в матрицах располагаются по строкам и столбцам. Обозначение $F(i, j)$ определяет элемент матрицы F, стоящий на пересечении i-й строки и j-го столбца.

В материалах данных лекций мы будем использовать произведение одной матрицы на другую, а также обратные матрицы и определители матрицы. Объясним на примере

операцию произведения матрицы A на матрицу B. Для перемножения матриц нужно, чтобы число столбцов матрицы A равнялось числу строк матрицы B.

Пусть

$$A=\begin{bmatrix}1 & 2 & 3\\ 0 & -1 & 4\end{bmatrix};\ B=\begin{bmatrix}2 & 5\\ -1 & 0\\ 1 & -2\end{bmatrix}.$$

Обозначим $A(i,\bullet)$ – i-ю строку матрицы A; $B(\bullet, j)$ – j-й столбец матрицы B.

Результатом перемножения $A(i,\bullet)$ на $B(\bullet, j)$ будет элемент C_{ij} результирующей матрицы $C = AB$, определяемый как

$$C_{ij} = A(i,\bullet)B(\bullet, j) = \sum_{k=1}^{|A(i,\bullet)|} A(i,k)B(k,j). \qquad (10.1)$$

Например,

$$C_{1,2} = \begin{pmatrix}1 & 2 & 3\end{pmatrix}\times\begin{pmatrix}5\\ 0\\ 2\end{pmatrix} = 1\cdot 5 + 2\cdot 0 + 3\cdot 2 = 11;$$

$$C_{2,2} = \begin{pmatrix}0 & -1 & 4\end{pmatrix}\times\begin{pmatrix}5\\ 0\\ 2\end{pmatrix} = 0\cdot 5 + (-1)\cdot 0 + 4\cdot 2 = 8 \text{ и т. д.}$$

Таким образом, операция (10.1) позволяет найти любой элемент результирующей матрицы произведения.

Пусть дана квадратная матрица A, например,

$$A=\begin{bmatrix}1 & 0 & -1\\ 2 & 3 & 1\\ 0 & 1 & 4\end{bmatrix}. \qquad (10.2)$$

Дадим понятие *определителя* квадратной матрицы. Из школьного курса известно, что определитель квадратной матрицы B второго порядка считается как

$$|B| = b_{11}b_{22} - b_{12}b_{21}.$$

Обозначим $|A_{ij}|$ – минор элемента A_{ij} матрицы A как определитель подматрицы A, получаемой вычеркиванием строки i и столбца j. Так, для (10.2) найдем

$$|A_{23}| = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = 1;$$

$$|A_{33}| = \begin{vmatrix} 1 & 0 \\ 2 & 3 \end{vmatrix} = 3 \text{ и т. д.}$$

Для квадратной матрицы A любого порядка $n > 2$ ее определитель $|A|$ может быть найден как

$$|A| = \sum_{k=1}^{n} (-1)^{i+k} |A_{ik}| a_{ik} = \sum_{k=1}^{n} (-1)^{i+k} |A_{ki}| a_{ki}. \qquad (10.3)$$

Так, для (10.3) имеем

$$|A| = (-1)^{1+1} |A_{11}| \cdot 1 + (-1)^{2+1} |A_{21}| \cdot 2 + (-1)^{3+1} |A_{31}| \cdot 0.$$

Учитывая, что $|A_{11}| = \begin{vmatrix} 3 & 1 \\ 1 & 4 \end{vmatrix} = 11$, $|A_{21}| = \begin{vmatrix} 0 & -1 \\ 1 & 4 \end{vmatrix} = 1$, $|A_{31}| = \begin{vmatrix} 0 & -1 \\ 3 & 1 \end{vmatrix} = 3$,

получим $|A| = 1 \cdot 11 \cdot 1 - 1 \cdot 1 \cdot 2 + 1 \cdot 3 \cdot 0 = 9.$

С другой стороны,

$$|A| = (-1)^{1+1} |A_{11}| \cdot 1 + (-1)^{2+1} |A_{12}| \cdot 0 + (-1)^{3+1} |A_{13}| \cdot (-1) =$$
$$= 1 \cdot 11 \cdot 1 + 1 \cdot (-1) \cdot 2 = 9.$$

Далее определим операцию транспонирования прямоугольной матрицы A как такую, при которой i-я строка матрицы становится ее i-м столбцом (для каждой такой строки).

Пример. Пусть $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$.

Результат *транспонирования* матрицы A обозначается как A' или A^T и имеет вид:

$$A' = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}.$$

Таким образом, транспонирование, по существу, сводится к повороту матрицы.

Большую роль в статистике и теории информации играет так называемая ковариационная матрица. Рассмотрим табл. 5.1, в которой записаны две случайные величины: «Вес» и «Рост» для некоторой совокупности людей. Для удобства обозначим ξ_1 = ВЕС, ξ_2 = РОСТ. Ковариационная матрица для двух случайных величин определяется таким образом:

$$\mathrm{COV}_{\xi_1\xi_2} = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12} \\ \sigma_{21} & \sigma_{22}^2 \end{bmatrix}, \qquad (10.4)$$

где

$$\sigma_{ij}^2 = M\{(x_i - \mu_i)(x_j - \mu_j)\} \qquad (10.5)$$

определяет дисперсию (взаимную дисперсию при $i = j$); μ_i – математическое ожидание (среднее значение) случайной величины ξ_i; $M\{...\}$ означает взятие математического ожидания: $\mu_1 = M(\xi_1)$, $\mu_2 = M(\xi_2)$.

Так, из табл. 5.1 найдем:

$$\mu_1 = M(\text{Вес}) = 76{,}375;$$

$$\mu_2 = M(\text{Рост}) = 175.$$

Определим $\sigma_{11}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_{1i} - \mu_1)^2 = 80{,}23;$

$$\sigma_{12} = \sigma_{21} = \frac{1}{n}\sum_{i=1}^{n}(x_{1i} - \mu_1)(x_{2i} - \mu_2) = 55;$$

$$\sigma_{22}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_{2i} - \mu_2)^2 = 70.$$

На практике значительно шире применяют не ковариационную матрицу, а корреляционную матрицу, элементы которой находят как

$$\mathrm{COV}_{\xi_1\xi_2} = \begin{bmatrix} \dfrac{\sigma_{11}^2}{\sigma_{11}^2} & \dfrac{\sigma_{12}}{\sigma_{11}\sigma_{22}} \\ \dfrac{\sigma_{21}}{\sigma_{11}\sigma_{22}} & \dfrac{\sigma_{22}^2}{\sigma_{22}^2} \end{bmatrix} = \begin{bmatrix} 1 & \dfrac{\sigma_{12}}{\sigma_{11}\sigma_{22}} \\ \dfrac{\sigma_{21}}{\sigma_{11}\sigma_{22}} & 1 \end{bmatrix}. \qquad (10.6)$$

В нашем случае получим

$$\mathrm{COV}_{\xi_1\xi_2} = \begin{bmatrix} 1 & 0,734 \\ 0,734 & 1 \end{bmatrix}.$$

Коэффициенты корреляции показывают меру линейной связи (зависимости) между двумя случайными величинами (ξ_1 и ξ_2). Коэффициент корреляции принимает значение в диапазоне от –1 до +1. При этом 0 означает, что линейной связи между ξ_1 и ξ_2 нет; +1 (–1) означает линейную зависимость между ξ_1 и ξ_2. В нашем примере коэффициент корреляции между весом и ростом достаточно высок (0,734), что свидетельствует о существенной (почти линейной) связи между этими величинами.

Для нормированных случайных величин корреляционная и ковариационная матрица совпадают.

Лекция 11
ЗАДАЧА ПОИСКА И СОРТИРОВКИ ИНФОРМАЦИИ

Цель. Изучить механизм поиска информации на основе *B*-дерева и его модификаций.

Поиск информации является одной из центральных задач, решаемых в автоматизированных информационных системах – в базах данных. При этом основная задача поиска – быстро найти требуемую информацию. Критерий быстроты является важнейшей технической характеристикой системы поиска. Для ускорения поиска информация должна быть некоторым образом организована. Различают следу-ющие основные виды поиска: поиск по ключам на основе двоичного индексного дерева, ассоциативный поиск, поиск по ключевым словам, поиск информации как задача распознавания образов и пр.

11.1. *B*-дерево и его использование

Рассмотрим реализацию поиска информации на основе *B*-дерева (*Balanced Tree*). Это дерево также называют двоичным индексным деревом, если речь идет о базах данных. Предположим, имеется упорядоченный список *L* фамилий:

Агафов
Бердычев
Гвардиан
Ивановский
Илимов
Ичкайло
Пунтус
Твердынко
Уфимцев
Цар

Условия упорядоченности списка обязательно. Построим для этого списка *L* соответствующее *B*-дерево. Найдем середину списка *L* (или примерно середину) и соответствующую фамилию

поместим в корень дерева. Выведем из нее два ребра (влево и вправо). Влево отнесем все фамилии, стоящие выше, чем фамилия, помещенная в корень; вправо отнесем все остальные фамилии (рис. 11.1).

Рис. 11.1

С каждым из списков L_1 и L_2 поступим аналогично (рис. 11.2).

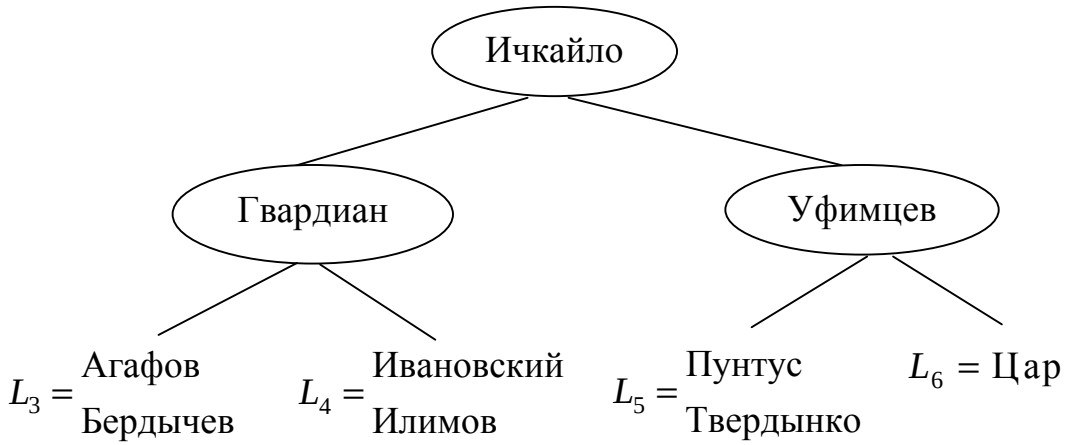


Рис. 11.2

Продолжим процесс для подсписков L_3, L_4, L_5, L_6. Опуская дальнейшие выкладки, приводим результирующее *B*-дерево (рис. 11.3).

Теперь покажем, как это дерево позволяет быстро отыскивать нужную фамилию. Например, пусть требуется найти фамилию Твердынко.

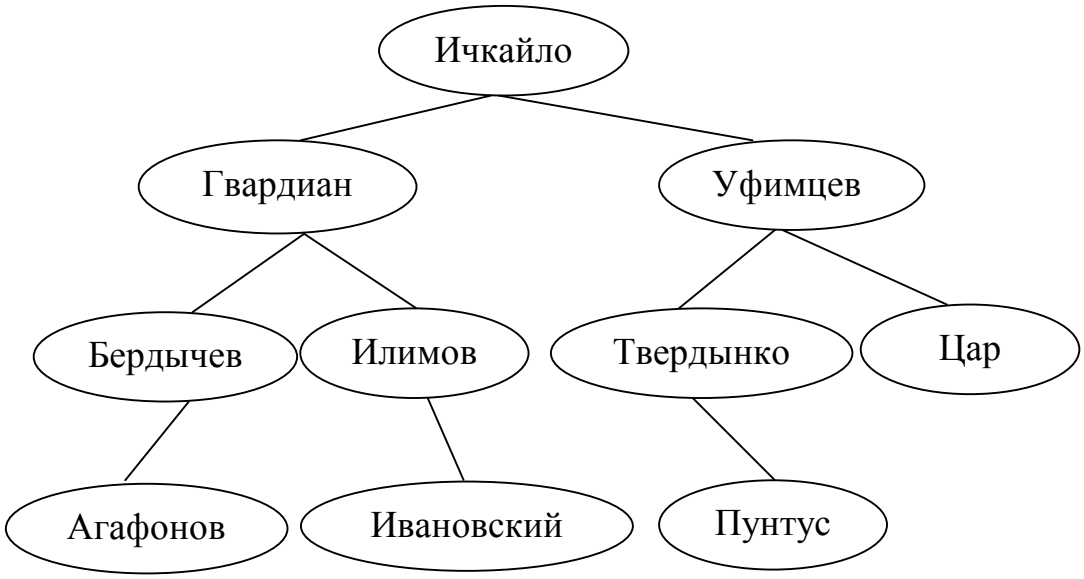


Рис. 11.3

Начинаем поиск с корневой вершины и сравниваем фамилии Ичкайло и Твердынко. Поскольку Твердынко по алфавиту стоит ниже Ичкайло, перемещаемся от корневой вершины вправо (рис. 3.3). Попадаем в вершину «Цар». Сравниваем фамилию Цар и Твердынко. Теперь Твердынко находится в списке выше, чем Цар. Следовательно, переходим из вершины «Цар» влево и попадаем в искомый узел с фамилией Твердынко.

Сложность операции поиска оценивается числом произведенных сравнений (пройденных вершин). Ясно, что в худшем случае придется пройти вершины самого длинного пути в B-дереве. Заметим, что слово *Balanced* (сбалансированный) подчеркивает тот факт, что все пути в B-дереве либо имеют одинаковую длину, либо отличаются на 1. Теперь, если число всех вершин N, то нетрудно показать, что длина наибольшего пути
в B-дереве составит

$$1+]\log_2 N[,$$

где $]x[$ – округленное до ближайшего меньшего целого значение x.

Итак, если, скажем, в *B*-дереве 32 768 вершин, то в худшем случае придется пройти $\log_2 32\,768 = 16$ вершин (т. е. сделать 16 сравнений). Если бы пришлось фамилии проверять одну за другой, то число сравнений в среднем составило бы 16 384. Итак, сложность поиска на *B*-дереве оценивается как $\log_2 N$.

Следует понимать, что фамилия (в нашем примере) – это признак, по которому должна быть найдена вся запись, которая может включать адрес, должность, год рождения, семейное положение
и пр. Обычно в базах данных используются индексы для поиска информации. По индексу отыскивается номер записи (адрес записи), а затем уже по найденному адресу выполняется чтение записи из внешней памяти.

При добавлении новых записей или удалении имеющихся *B*-дерево перестает быть сбалансированным. Поэтому в реальных системах *B*-дерево нужно периодически перестраивать.

11.2. B^+-дерево

B^+-дерево – это *B*-дерево, из каждой вершины которого выходит ровно $k > 2$ ребер. Построим «троичное» B^+-дерево для заданного списка *L* индексов:

$$L = [1, 3, 4, 16, 17, 23, 25, 42, 47, 49, 51, 54, 58, 67, 84]$$

Список нужно разбить на $k = 3$ (примерно) равных подсписка. В нашем случае сделаем такое разбиение:

$$L = \underset{L_1}{[1, 3, 4, 16, 17]}\ \underset{L_2}{[\ 23, 25, 42, 47, 49]}\ \underset{L_3}{[51, 54, 58, 67, 84]}$$

В корень формируемого B^+-дерева поместим первый элемент списка L_2, т. е. 23 (можно было также поместить в корень последний элемент списка L_1, последний элемент списка L_2 или первый элемент списка L_3). Наше дерево примет на этом этапе вид, показанный на рис. 11.4.

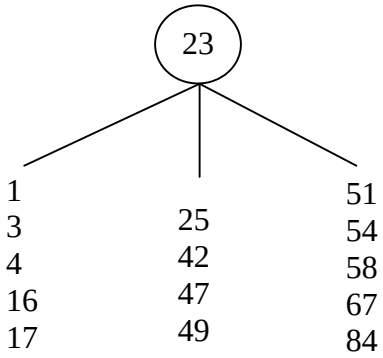


Рис. 11.4

Аналогичным образом поступим с каждым из подсписков, разбив их на три примерно равные части и поместив в формируемые вершины первые элементы средних списков (рис. 11.5).

Поясним сказанное на примере первого подсписка: [1, 3] [4, 16] [17], который мы разбили, как показано, на три примерно равных подсписка. Элемент 4 помещаем в узел. Остальные элементы распределены, как показано на рис. 11.5.

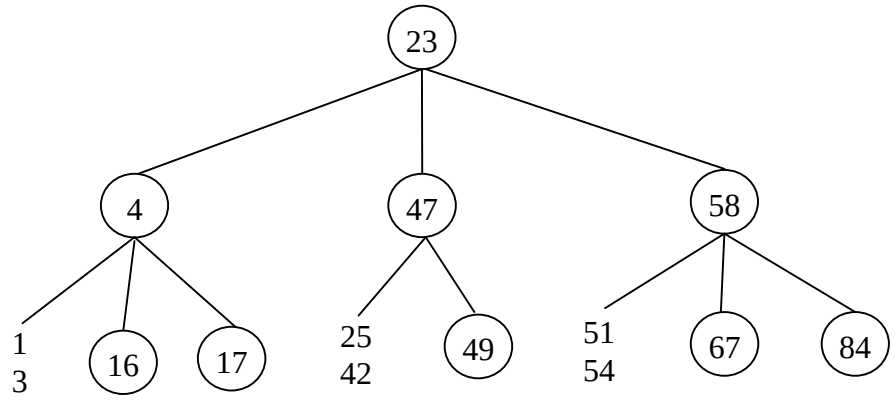


Рис. 11.5

Окончательный вариант дерева приведен на рис. 11.6.

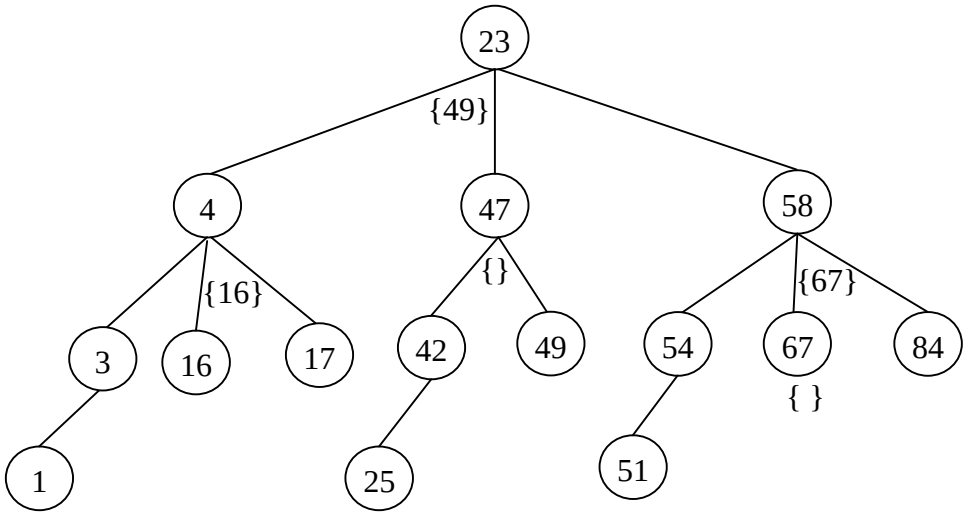


Рис. 11.6

Рассмотрим, как выполняется поиск, скажем, числа 67. Сравниваем 23 и 67, так как 67 > 23, то нужно перемещаться либо по средней ветке на элемент 47, либо по правой ветке на элемент 58. Но по какой именно? Нам не достает информации. Поэтому в B^+-дереве в каждой вершине, в которую осуществляется переход по средней ветке, записывают максимальный элемент, который может быть достигнут при движении по этому пути. Такие элементы нами помещены в фигурные скобки рядом с вершинами (рис. 3.6). Теперь ответ на вопрос, куда переходить при поиске элемента 67, решается сравнением с элементом в фигурных скобках рядом с вершиной 23, т. е. с 49. Поскольку 67 > 49, то двигаться по среднему пути в вершину 47 не имеет смысла: максимальный возможный номер на этом пути – 49. Поэтому переходим в вершину 58. Сравниваем 58 и 67. Опять та же ситуация. Сравниваем 67 с элементом в фигурных скобках, т. е. с 67 и получаем ответ, в какую вершину переходить.

Оценим теперь сложность поиска для троичного B^+-дерева. Не забудем о том, что мы производим дополнительные сравнения в каждых двух вершинах из трех пройденных. Ясно, что длина пути из корня в конечную вершину оценивается как $]1+\log_3 N[$. Для $\frac{2}{3}$ из этого числа вершин предстоит произвести дополнительные проверки. Значит, общее число проверок в худшем случае составит

$$]1+\log_3 N[+\frac{2}{3}\cdot]1+\log_3 N[=\frac{5}{3}\cdot]1+\log_3 N[.$$

Какое дерево лучше – двоичное или троичное? Ответ такой: в среднем лучше себя «ведет» двоичное дерево, однако примерно в $\frac{1}{3}$ всех случаев поиска троичное дерево обеспечивает бóльшую скорость поиска в сравнении с двоичным, и это ускорение составляет максимально

$$a=\frac{\log_2 N}{\log_3 N}=\log_2 3\approx 1{,}58$$

(но это максимальное ускорение, достигаемое только по единственному пути!).

Лекция 12
АССОЦИАТИВНЫЙ ПОИСК ДАННЫХ

Цель. Рассмотреть математические основы ассоциативного поиска как варианта поиска альтернативного поиску на основе *B*-дерева.

Ассоциативный поиск информации основан на получении адреса данных путем преобразования признака. Например, пусть признаком является фамилия. Возьмем фамилию «Иванов». Функция, которая получает адрес по фамилии, может быть построена различными способами. Один из наиболее простых – сложить ASCII-коды символов, образующих фамилию. Для фамилии «Иванов» эта сумма равна 1151. Итак, функция, получающая по значению признака число, называется ХЭШ-функцией (от англ. hash – мешанина). При ассоциативном поиске выделяют основную память и вспомогательную. Пусть размер основной памяти рассчитан на 300 фамилий. Разделим полученное значение 1151 на 300 и возьмем остаток от деления.
В нашем случае остаток равен 251. Поскольку остаток может быть нулевым, прибавим к нему 1. Итак, результатом наших действий является получение адреса в основной памяти, равного 252. В этой ячейке мы предполагаем найти фамилию «Иванов». Теперь можно заметить, что такая экзотическая фамилия как «Ивавон» даст точно такой же адрес в основной памяти, что приведет к конфликту. Это наблюдение указывает на общий недостаток ассоциативного поиска, для преодоления которого используется вспомогательная память. Размер вспомогательной памяти неограничен. При возникновении конфликта в основной ассоциативной памяти выполняется поиск во вспомогательной памяти последовательным просмотром ее ячеек, начиная с самой первой. Это обстоятельство также является недостатком. Для устранения данного недостатка вспомогательную память можно проиндексировать и организовать поиск на основе *B*-дерева. Таким образом, это позволяет объединить преимущества ассоциативной памяти (доступ за одно обращение) и *B*-дерева. Для эффективного функционирования ассоциативного механизма поиска нужно обеспечить выбор hash-функции, что является непростой задачей.

Рассмотрим следующий интересный подход к выбору hash-функции.

Для его реализации требуется, чтобы размер основной памяти был равен 2^r ($r > 0$). Пусть, например, $r = 5$ и основная область рассчитана на $2^5 = 32$ фамилии. Теперь предположим, что у нас имеется столько же независимых друг от друга функций f_1, …, f_5, для которых известны минимальные и максимальные значения, которые они принимают. С помощью этих функций можно получить число в диапазон 00000…1111. Каждая из функций f_1, …, f_5 просто берет сумму ASCII-кодов символов фамилии и умножает эту сумму на некоторый фиксированный множитель, скажем, f_1 умножает
на 17, f_2 на 53, f_3 на 71, f_4 на 113 и f_5 на 147. Желательно, чтобы множители были простыми числами. Далее выбираем среднюю цифру произведения (если число цифр четное, то добавляем слева 0). Значением функции является 0, если эта цифра меньше 5, и 1, если эта цифра больше или равна пяти. Общий результат – двоичное число, записанное как результат объединения в одно слово полученных двоичных цифр.

Так, для фамилии «Иванов» получим:

$$f_1 = 1 \ (19567 = 1151 \times 17);$$

$$f_2 = 0 \ (61003 = 1151 \times 53);$$

$$f_3 = 1 \ (81721 = 1151 \times 71);$$

$$f_4 = 0 \ (130063 = 1151 \times 113);$$

$$f_5 = 1 \ (169197 = 1151 \times 147).$$

Таким образом, результатом является адрес 10101. Одним важным вариантом задачи поиска является такой, когда используется несколько признаков. Его мы рассмотрим в следующей лекции.

Лекция 13
ПОИСК КАК ЗАДАЧА РАСПОЗНАВАНИЯ

Цель. Изложить технологию поиска многомерных данных, основанную на использовании систем разделяющих неравенств.

Пусть имеется два числовых признака x_1 и x_2 и множество значений, собранных в табл. 13.1.

Таблица 13.1

x_1	x_2
2	4
–1	3
0	1
2	5
3	–2
6	3
1	1
4	3

Разобьем общее число записей на две равные (примерно равные) половины, из которых одну обозначим как «0», а вторую – как «1». Затем также поступим с полученными половинами и т. д. результат такого разбиения представлен на рис. 13.1.

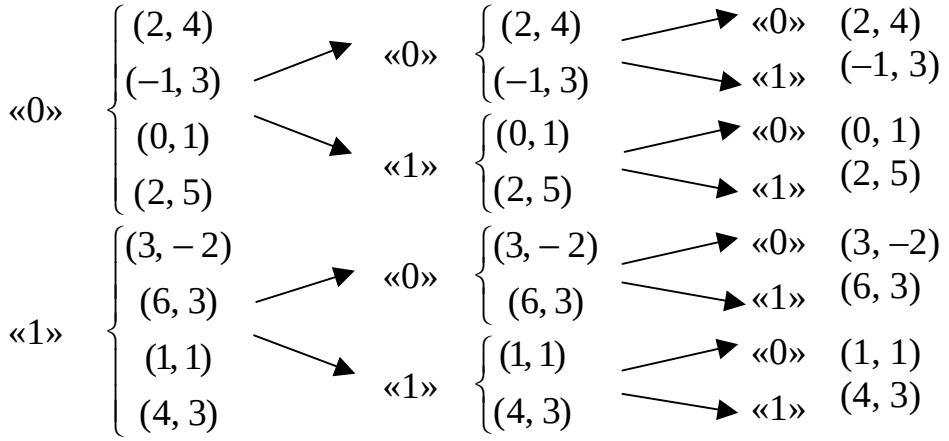


Рис. 13.1

Теперь допустим, что для разбиения каждого множества на две равных (примерно равных) половины мы используем линейное неравенство вида

$$a_1 x_1 + a_2 x_2 \geq 0. \qquad (13.1)$$

Коэффициенты a_1 и a_2 нам еще предстоит найти. Используя неравенство (13.1) мы вместо x_1 и x_2 подставляем значения, соответствующие отыскиваемому объекту из табл. 13.1. Если неравенство выполняется, то попадаем в половину, помеченную как «0». Если неравенство не выполняется, то попадаем в половину, помеченную как «1». Этот принцип можно эффективно обобщить, так что в лучшем случае число всех неравенств будет равно $1+]\log_2 N[$, где N – число всех записей. Однако эта наилучшая теоретическая оценка, к сожалению, не всегда выдерживается.

Итак, начнем с того, что покажем, как отыскать коэффициенты a_1 и a_2 в неравенстве (13.1). Составим таблицу для первых двух множеств:

x_1	x_2	y
2	4	≥ 0
–1	3	≥ 0
0	1	≥ 0
2	5	≥ 0
3	–2	< 0
6	3	< 0
1	1	< 0
4	3	< 0

Запишем следующую систему неравенств:

$$\begin{cases} 2a_1 + 4a_2 \geq 0, \\ -1a_1 + 3a_2 \geq 0, \\ 0a_1 + 1a_2 \geq 0, \\ 2a_1 + 5a_2 \geq 0, \\ 3a_1 - 2a_2 < 0, \\ 6a_1 + 3a_2 < 0, \\ 1a_1 + 1a_2 < 0, \\ 4a_1 + 3a_2 < 0. \end{cases} \qquad (13.2)$$

Неравенства (13.2) составлены согласно описанному принципу. Если найти коэффициенты a_1 и a_2, то все множество записей будет разделено нужным нам образом на две условные половины: «0» и «1». Имеются две проблемы. Во-первых, следует избавиться от жестких неравенств (<). Это сделать нетрудно, если ввести достаточно малую величину $\xi > 0$, такую, что

$$c_1 a_1 + c_2 a_2 < 0$$

можно заменить на

$$c_1 a_1 + c_2 a_2 \leq \xi.$$

Заметим, что такая замена может сделать систему неравенств несовместной, однако и исходная система может быть несовместна, так что наша основная задача – разрешить проблему несовместности системы неравенств. В иллюстративных целях зададим $\xi = 1$
и приведем все неравенства к виду

$$\begin{cases} 2a_1 + 4a_2 \geq 0, \\ -1a_1 + 3a_2 \geq 0, \\ 0a_1 + 1a_2 \geq 0, \\ 2a_1 + 5a_2 \geq 0, \\ -3a_1 + 2a_2 \geq 1, \\ -6a_1 - 3a_2 \geq 1, \\ -1a_1 - 1a_2 \geq 1, \\ -4a_1 - 3a_2 \geq 1. \end{cases} \quad (13.3)$$

Для решения мы используем алгоритм устранения невязок.

Неравенство с положительной правой частью называется *невязкой*. Если в системе нет невязок, то ее решение доставляется нулевыми значениями переменных. В противном случае описываемый алгоритм пытается избавиться от невязок. При этом выделяется две фазы. На первой фазе нужно получить систему базисных неравенств вида $a_i \geq 0$. Вторая фаза выполняется так же, как и первая, но уже при наличии базисных неравенств. Если на второй фазе в процессе итераций встречается невязка, причем все коэффициенты в левой части неположительны, то устанавливаем факт неразрешимости (несовместности) системы неравенств. Этот факт мы будем использовать специальным образом. Итак, возьмем любую невязку, например,

$$-3a_1 + 2a_2 \geq 1$$

и выразим из нее переменную a_1 так

$$-3a_1 \geq 1 + 2a_2;$$

$$a_1 \leq -\frac{1}{3} - \frac{2}{3}a_2;$$

$$a_1 = -\frac{1}{3} - \frac{2}{3}a_2 - z_1, \quad (13.4)$$

где z_1 – новая неотрицательная переменная.

Подставим выражение (13.4) в (13.3). Получим систему:

$$\begin{cases} \frac{8}{3}a_2 - 2z_1 \geq \frac{2}{3}, \\ \frac{11}{3}a_2 + z_1 \geq -\frac{1}{3}, \\ a_2 \geq 0, \\ \frac{11}{3}a_2 - 2z_1 \geq \frac{2}{3}, \\ z_1 \geq 0, \\ a_2 + 6z_1 \geq -1, \\ -\frac{1}{3}a_2 + z_1 \geq \frac{2}{3}, \\ -\frac{1}{3}a_2 + 4z_1 \geq -\frac{1}{3}. \end{cases} \quad (13.5)$$

Первая фаза завершена. Имеются два базисных неравенства $a_2 \geq 0$ и $z_1 \geq 0$. Вторая фаза выполняется с небольшим отличием от первой: из невязки выражаем переменную с положительным коэффициентом. Так, возьмем невязку:

$$\frac{8}{3}a_2 - 2z_1 \geq \frac{2}{3}.$$

Выражаем a_2:

$$a_2 \geq \frac{1}{4} + \frac{3}{4}z_1;$$

$$a_2 = \frac{1}{4} + \frac{3}{4}z_1 + z_2, \quad (13.6)$$

где z_2 – есть новая неотрицательная переменная.

Подставляем (13.6) в (13.5):

$$\begin{cases} z_2 \geq 0, \\ \frac{15}{4} z_1 + \frac{11}{3} z_2 \geq -\frac{5}{4}, \\ \frac{3}{4} z_1 + z_2 \geq -\frac{1}{4}, \\ \frac{3}{4} z_1 + \frac{11}{3} z_2 \geq -\frac{1}{4}, \\ z_1 \geq 0, \\ \frac{27}{4} z_1 + z_2 \geq -\frac{5}{4}, \\ \frac{3}{4} z_1 - \frac{1}{3} z_2 \geq \frac{3}{4}, \\ \frac{15}{4} z_1 - \frac{1}{3} z_2 \geq -\frac{1}{4}. \end{cases} \qquad (13.7)$$

Осталась одна единственная невязка:

$$\frac{3}{4} z_1 - \frac{1}{3} z_2 \geq \frac{3}{4};$$

$$z_1 = 1 + \frac{4}{9} z_2 + z_3. \qquad (13.8)$$

Подстановка (13.6) приводит к системе без невязок. В этой системе решение дает $z_2 = z_3 = 0$. Отсюда в силу подстановок (13.4), (13.6), (13.8) найдем: $a_1 = -2;\ a_2 = 1$.

Итак, первое разделяющее неравенство нами получено в форме

$$-2x_1 + x_2 \geq 0. \qquad (13.9)$$

Нетрудно убедиться, что для точек (2, 4), (–1, 3), (0, 1), (2, 5) мы получили условное множество «0» (неравенство (13.1) выполнимо). Для остальных точек это неравенство невыполнимо (условное множество «1»).

Теперь, согласно рис. 13.1, нам нужно провести новую «разбивку» точек.

x_1	x_2	y

2	4	≥ 0
–1	3	≥ 0
0	1	< 0
2	5	< 0
3	–2	≥ 0
6	3	≥ 0
1	1	< 0
4	3	< 0

Действительно, второй слой точек на рис. 13.1 разбит именно таким образом. Снова составляем систему неравенств (для переменных a_3 и a_4).

$$\begin{cases} 2a_3 + 4a_4 \geq 0, \\ -1a_3 + 3a_4 \geq 0, \\ 0a_3 - a_4 \geq 1, \\ -2a_3 - 5a_4 \geq 1, \\ 3a_3 - 2a_4 \geq 0, \\ 6a_3 + 3a_4 \geq 0, \\ -a_3 - a_4 \geq 1, \\ -4a_3 - 3a_4 \geq 1. \end{cases} \qquad (13.10)$$

Выполняем первую фазу алгоритма. Из невязки

$$0a_3 - a_4 \geq 1$$

получим

$$0a_3 - a_4 = 1 + z_1;$$

$$a_4 = -1 - z_1. \qquad (13.11)$$

Эта замена приводит к следующей системе

$$\begin{cases} 2a_3 - 4z_1 \geq 4, \\ -a_3 - 3z_1 \geq 3, \\ z_1 \geq 0, \\ -2a_3 + 5z_1 \geq -4, \\ 3a_3 + 2z_1 \geq -2, \\ 6a_3 - 3z_1 \geq 3, \\ -a_3 + z_1 \geq 0, \\ -4a_3 + 3z_1 \geq -2. \end{cases} \tag{13.12}$$

Из невязки

$$2a_3 - 4z_1 \geq 4$$

выразим переменную a_3:

$$a_3 = 2 + 2z_1 + z_2. \tag{13.13}$$

Система (13.12) примет следующий вид:

$$\begin{cases} \text{(a)} \quad z_2 \geq 0, \\ \text{(b)} \quad -5z_1 - z_2 \geq 5, \\ \text{(c)} \quad z_1 \geq 0, \\ \text{(d)} \quad z_1 - 2z_2 \geq 0, \\ \text{(e)} \quad 8z_1 + 3z_2 \geq -8, \\ \text{(f)} \quad 9z_1 + 6z_2 \geq -9, \\ \text{(g)} \quad -z_1 - z_2 \geq 2, \\ \text{(h)} \quad -5z_1 - 4z_2 \geq 6. \end{cases} \tag{13.14}$$

Теперь мы столкнулись со следующей ситуацией. Невязки (b), (g), (h) невыполнимы вместе с базисными неравенствами. Следовательно, (13.14) противоречива. Это означает, что нельзя подобрать переменные a_3, a_4 в (13.10), чтобы разрешить нужным образом записи на подмножества «0» и «1». Поступим так: просто исключим из (13.14) невязки (b), (g), (h), которые с базисными неравенствами дают противоречие.

В итоге не останется ни одной невязки, и процесс завершится. Найдем

$$z_1 = z_2 = 0; \;\; a_3 = 2; \;\; a_4 = -1,$$

снова получим разделяющее неравенство

$$2x_1 - x_2 \geq 0. \qquad (13.15)$$

Однако одного неравенства (13.15) будет недостаточно, так как оно неверно разделяет точки, соответствующие исключенным неравенствам. Неравенство (13.15) разбивает точки так, как показано на рис. 13.2.

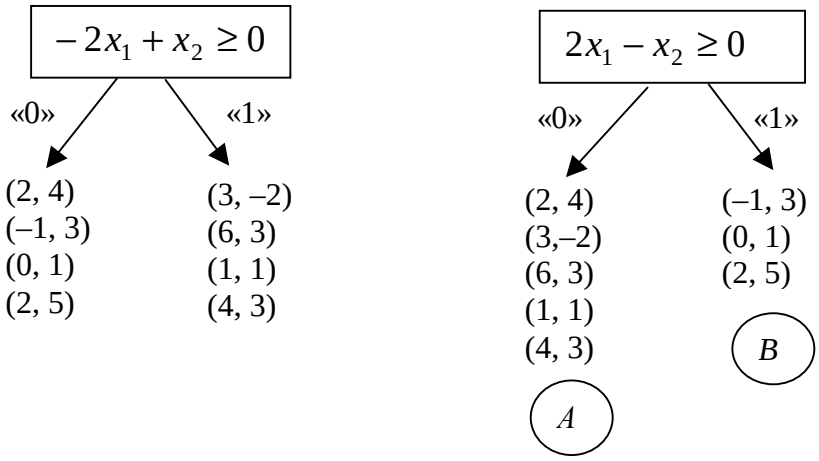


Рис. 13.2

Очевидно, что ни множество точек на ветви *A*, ни множество точек на ветви *B* не являются «правильными» в смысле рис. 13.1. Поэтому нам потребуется еще как минимум два неравенства (по одному на каждую из ветвей *A* и *B* на рис. 13.2), чтобы правильно «развести» точки. Сначала для ветви *A* сформируем систему

$$\begin{cases} 2a_5 + 4a_6 \geq 0, \\ 3a_5 - 2a_6 \geq 0, \\ 6a_5 + 3a_6 \geq 0, \\ -1a_5 - 1a_6 \geq 1, \\ -4a_5 - 3a_6 \geq 1. \end{cases} \qquad (13.16)$$

Эта система сформирована только для точек, вошедших в условную ветвь *A* на рис. 13.2. Составление неравенств (13.16) подчиняется обычным правилам: если точка (x_i, y_i) попадает в множество «0», то для него формируем неравенство

$$a_5 x_i + a_6 x_i \geq 0;$$

если же точка (x_i, y_i) попадает в множество «1», то для него формируем неравенство

$$-a_5 x_i - a_6 x_i \geq 1.$$

Опуская выкладки, запишем решение системы (13.16): $a_5 = 1$, $a_6 = -2$ и новое разделяющее неравенство

$$x_1 - 2x_2 \geq 0. \qquad (13.17)$$

При поиске решения (13.16) пришлось удалить одну из точек, поэтому ветвь *A* еще не обработана до конца (рис. 13.3).

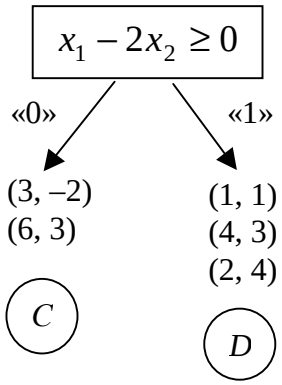


Рис. 13.3

Ветвь *C* на рис. 13.3 уже нет необходимости обрабатывать, поскольку она содержит «правильные» точки. Ветвь *D* содержит «неправильную» точку (2, 4), поэтому для нее составим систему

$$\begin{cases} -1a_7 - 1a_8 \geq 1, \\ -4a_7 - 3a_8 \geq 1, \\ 2a_7 + 4a_8 \geq 0. \end{cases} \qquad (13.18)$$

Теперь уже найдем окончательно: $a_7 = -2$, $a_8 = 1$ и разделяющее неравенство

$$-2x_1 + x_2 \geq 0. \qquad (13.19)$$

Вернемся к ветви *B* на рис. 13.2 и запишем систему

$$\begin{cases} -a_9 + 3a_{10} \geq 0, \\ -a_{10} \geq 1, \\ -2a_9 - 5a_{10} \geq 1. \end{cases} \qquad (13.20)$$

Находим решение: $a_9 = -3,\ a_{10} = -1$ и разделяющее неравенство

$$-3x_1 - x_2 \geq 0. \qquad (13.21)$$

Итак, со средним слоем на рис. 13.1 разобрались. Для этого слоя построим поисковое дерево, показанное на рис. 13.4.

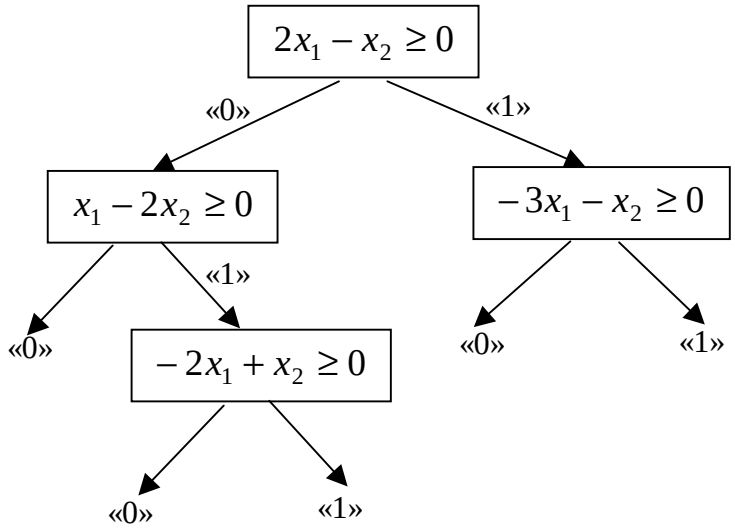


Рис. 13.4

Читатель, вероятно, догадался, почему мы употребили слова «поисковое дерево». В самом деле, движение по ветвям такого дерева вполне соответствует процессу поиска на рассмотренном нами ранее *B*-дереве. Обратим также внимание на аналогию в данном случае поиска и распознавания. Так, с помощью поискового дерева, типа изображенного на рис. 13.4, можно ответить на вопрос – к какому классу («0» или «1») принадлежит заданная точка? Этот вопрос является основным в теории распознавания образов. Отсюда мы и почерпнули аналогию, которую закрепили в названии данной лекции.

Нам осталось разобраться с последним, третьим слоем на рис. 13.1. Опуская все выкладки, даем вариант окончательного поискового дерева на рис. 13.5 для данного слоя.

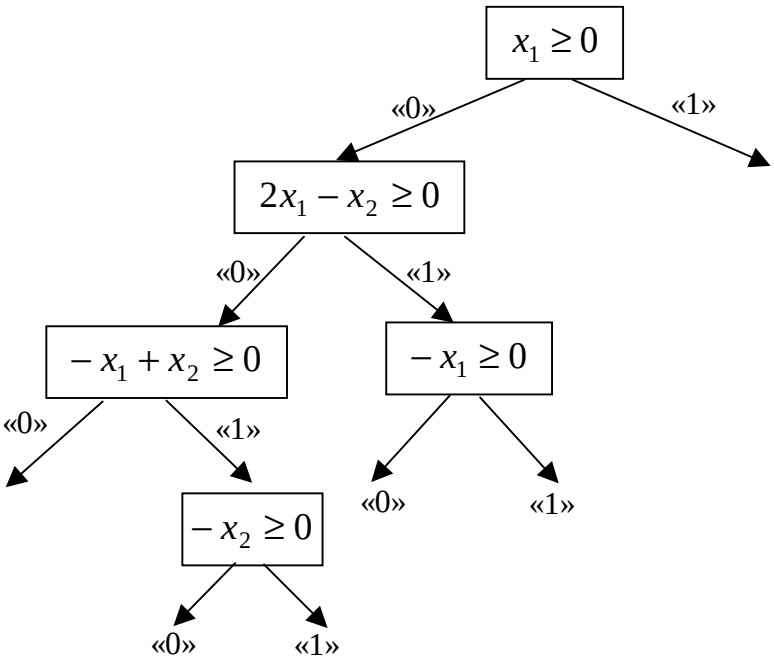


Рис. 13.5

Итак, все слои дерева на рис. 13.1 распознаются правильно. Остается продемонстрировать, в какой адрес переводится, например, точка (2, 5). На первом слое мы используем единственное разрешающее неравенство (13.9), которое выполняется для точки (2, 5), поэтому получаем первую цифру адреса – «0». На втором слое используем поисковое дерево (рис. 13.4). Двигаемся по этому дереву от вершины к вершине согласно правилу: если неравенство выполняется в данной точке, то идем по ветке «0», иначе – по ветке «1». Получаем вторую цифру адреса – «1». Наконец, используем третье поисковое дерево (рис. 13.5) и находим третью цифру адреса – «1». Итоговый адрес «011». Более подробно с механизмом поиска/распознавания читатель может ознакомиться в исследовании [15]. Заметим, что даже для систем с тысячами записей размер поискового дерева (для одного разряда) оценивается как натуральный логарифм от числа записей.

Поиск по ключевым словам. Поиск информации по ключевым словам широко используется в сети Интернет. Наиболее простым образом такой поиск можно реализовать выписав для каждого документа список ключевых слов. При поиске пользователь также записывает ключевые слова. Система поиска возвращает список документов, содержащих наибольшее число совпадений. Этот вариант

поиска достаточно примитивен, и далее в этом пособии мы рассмотрим более сложные варианты.

Лекция 14
ПРОБЛЕМА И МЕТОДЫ СОРТИРОВКИ

Цель. Изучить основные методы сортировки данных в автоматизированных системах.

Суть сортировки информации состоит в ее упорядочении, например в алфавитно-цифровом порядке. Необходимость сортировки ясна уже хотя бы для построения B-дерева (B^+-дерева). Кроме того, отсортированную информацию часто требуется выводить на печать или экран монитора. Таким образом, проблема сортировки должна быть эффективно решена. Основная задача состоит в разработке быстрых методов сортировки. На языке вычислительной математики такие методы называют эффективными.

14.1. Вычислительный аспект проблемы сортировки

Вычислительный аспект проблемы сортировки связан с подсчетом числа сравнений, которое производит алгоритм. Рассмотрим следующий список, подлежащий упорядочению:

$$L = [2, 1, 5, 6, 3, 9, 4]. \qquad (14.1)$$

В этом списке 7 элементов. Пусть алгоритм решает задачу упорядочения так. Сначала отыскивает наименьшее число в списке и ставит его в начало. Затем в оставшейся части списка находит снова наименьшее число и ставит его вторым по порядку и т. д. При нахождении наименьшего числа нужно произвести $N - 1$ сравнений в списке из N чисел. Следовательно, описанный нами алгоритм сортировки выполнит общее число сравнений, равное

$$(N-1)+(N-2)+(N-3)+\ldots+1=\frac{N(N-1)}{2}.$$

В случае нашего списка L будет сделано 21 сравнение. Это не очень хорошая оценка, поскольку при большом N число сравнений

растет как функция $N^2 - N$. В вычислительной математике методы, выполняющие число сравнений, зависящее от числа входных значений N как полином от N, называют *эффективными*. На практике, впрочем, уже оценка $O(N^3)$ является мало приемлемой, хотя теоретически эффективной.

Одним из наиболее широко распространенных методов сортировки является метод пузырьков. Оценка его вычислительной сложности есть $O(N^2)$ (читается «О» большое от N^2. Наилучшая из известных оценок для методов сортировки есть $(N \log_2 N)$.

14.2. Метод пузырьков

Будем излагать метод на примере списка L (14.1). Берем первый элемент и перемещаем его вправо до тех пор, пока он не окажется меньше очередного элемента. Результат наших действий таков:

$$L = [1, \underline{2}, 5, 6, 3, 9, 4]. \qquad (14.2)$$

Берем следующий элемент – 5 – и делаем то же самое:

$$L = [1, \underline{2}, \underline{5}, 6, 3, 9, 4]. \qquad (14.3)$$

Берем теперь 6 и повторяем процедуру:

$$L = [1, \underline{2}, \underline{5}, 3, \underline{6}, 9, 4]. \qquad (14.4)$$

Наконец перемещаем 9:

$$L = [1, \underline{2}, \underline{5}, 3, \underline{6}, 4, \boxed{9}\ ]. \qquad (14.5)$$

Число 9 стало на свое место и дальше не участвует в рассмотрении.

Теперь возобновляем движение элементов с последнего подчеркнутого числа, т. е. с 6:

$$L = [1, \underline{2}, \underline{5}, 3, 4, \boxed{6}, \boxed{9}\ ]. \qquad (14.6)$$

Это правило используем и далее: начинаем перемещать всегда последний подчеркнутый элемент. Наши операции дают следующий результат:

$$L = [1, \boxed{2}, 3, 4, \boxed{5}, \boxed{6}, \boxed{9}\ ]. \qquad (14.7)$$

Подчеркнутых элементов не осталось. Просматриваем список с самого начала и все выполняем по аналогии, пока все элементы не будут упорядочены.

Вычислительная сложность метода пузырьков есть $O(N^2)$, так что это далеко не лучший метод сортировки.

14.3. Метод Шелла

Опять возьмем список (14.1) и подсчитаем число его элементов $N = 7$. Определим

$$k = \left[\frac{N}{2}\right] = 4,$$

где $[x]$ – округленное до ближайшего большего целого значения число x.

В методе Шелла сначала сравниваем элементы, отстоящие друг от друга на число элементов k, т. е. сравниваем следующие пары элементов с номерами:

(1, 6); (2, 7).

При этом меняем сравниваемые элементы местами, если необходимо. Получаем следующий результат:

$$L = [2, 1, \underline{4}, 6, 3, 9, 5]. \qquad (14.8)$$

Теперь сокращаем интервал вдвое:

$$k = \left[\frac{k}{2}\right] = 2.$$

Сравниваем следующие элементы (по номерам):

(1, 4); (2, 5); (3, 6); (4, 7);

$$L = [2, 1, 4, 5, 3, 9, 6]. \qquad (14.9)$$

Наконец, $k = \left[\frac{k}{2}\right] = 1$. Сравниваем:

(1, 3); (2, 4); (3, 5); (4, 6); (5, 7);

$$L = [2, 1, 3, 5, 4, 9, 6]. \qquad (14.10)$$

На последнем проходе сравниваем соседей ($k = 0$):

$$L = [2, 1, 3, 4, 5, 6, 9]. \qquad (14.11)$$

Последний проход для $k = 0$ делают в общем случае несколько раз.

Число проходов равно $\lceil \log_2 N \rceil$. Число сравнений на каждом проходе не более $\frac{N}{2}$. Следовательно, сложность метода Шелла в лучшем случае не выше $O(N \log_2 N)$. Это очень быстрый метод.

14.4. Метод Хоара

Метод Хоара также считается одним из наиболее эффективных методов сортировки, хотя в худшем случае его сложность есть $O(N^2)$. Его идея достаточно проста. Она основана на том, что в списке выбирается произвольный элемент, а затем все элементы, меньшие выбранного, помещают в левый подсписок, а остальные – в правый подсписок. Затем аналогичным образом разбивают левый и правый подсписки и т. д. Опять обратимся за пояснением к списку (14.1). Как правило, в качестве первого элемента для разбиения списка берут наибольший из двух крайних элемент, т. е. в нашем случае – 4. Последовательно сравниваем число 4 с элементами списка, начиная с первого, до тех пор, пока очередной элемент не окажется больше 4. Результатом будет новое положение числа 4, полученное обменом мест:

$$L = [2, 1, \underline{4}, 6, 3, 9, \underline{5}]. \qquad (14.12)$$

Теперь сравниваем число 4 с правыми соседями, начиная с конца списка. Поскольку $3 < 4$, производим обмен:

$$L = [2, 1, \underline{3}, 6, \underline{4}, 9, \underline{5}]. \qquad (14.13)$$

Теперь сравниваем 4 с левыми соседями, начиная с подчеркнутого числа 3:

$$L = [2, 1, \underline{3}, \underline{4}, \underline{6}, 9, \underline{5}]. \qquad (14.14)$$

Сравнение продолжаем с правыми соседями, начиная от первого неподчеркнутого элемента справа (т. е. 9):

$$L = [2, 1, 3, \boxed{4}, 6, 9, 5]. \qquad (14.15)$$

Итак, число 4 разбило исходный список на два подсписка:

$$L_1 = [2, 1, 3] \text{ и } L_2 = [6, 9, 5].$$

Далее для каждого из этих подсписков следует применить описанную процедуру (мы оставляем это читателю).

14.5. Линейное упорядочение со вставкой

Рассмотрим, как за минимальное число шагов вставить требуемое число в уже упорядоченный список, например,

$$5 \to [1, 2, 3, \underline{8}, 17, 21].$$

Сравниваем середину списка (т. е. 8) со вставленным числом. Поскольку $5 < 8$, то отбрасываем правую половину списка за числом 8:

$$5 \to [1, \underline{2}, 3].$$

Находим середину списка, т. е. 2. Поскольку $2 < 5$, то отбрасываем левую половину:

$$5 \to [3].$$

Теперь вставка очевидна:

$$[3, 5].$$

Это значит, что число 5 нужно поместить непосредственно за числом 3. Ясно, что число сравнений равно $\log_2 N$. Теперь список (14.1) сортируем по следующей схеме:

$$2 \to [\ ],$$

$$1 \to [2],$$

$$5 \to [1, 2],$$

$$6 \to [1, 2, 5],$$

$$\dots$$

$$4 \to [1, 2, 3, 5, 6, 9],$$

т. е. очередной элемент списка вставляется в уже отсортированную часть. Общее число операций сравнения составит:

$$\sum_{i=1}^{N} \log_2 i = \log_2 N!. \qquad (14.16)$$

Заметим, что $N > \log_2 N$. Тем не менее при использовании метода линейного упорядочения со вставкой дополнительно теряется

время на перемещение элементов при вставке. Следовательно, общая сложность этого алгоритма оценивается как $\sum_{i=2}^{N-1} \frac{i}{2} + \log_2 N! = \mathrm{O}\left(\frac{N^2}{4}\right)$.

Лекция 15
АНАЛОГОВАЯ И ЦИФРОВАЯ ПЕРЕДАЧА ИНФОРМАЦИИ

Цель. Рассмотреть физические принципы приема и передачи информации.

Различают два основных способа передачи информации: аналоговый и цифровой. Аналоговая передача информации реализуется на основе несущей высокочастотной электромагнитной волны. Такая волна характеризуется амплитудой, фазой и частотой.
С помощью процесса модуляции можно изменить амплитуду, фазу и частоту электромагнитной волны по закону изменения амплитуды полезного (информационного) сигнала.
Соответственно различают амплитудную, фазовую и частотную модуляции. Аналоговый сигнал можно передавать на очень большие расстояния. В качестве примера приведем спутниковое телевизионное вещание. Спутник применяют для отражения телевизионного сигнала и направления его в нужную точку на удаленной территории. Цифровая передача используется в сетях ЭВМ. Образно говоря, «1» соответствует высокий уровень импульса тока, а «0» – низкий. Применяют также перепады напряжения (например, в манчестерском кодировании) для обозначения «1» и «0». Цифровая передача более качественная и более надежная. Однако она охватывает меньшую территорию, является достаточно дорогостоящей, в частности, если речь идет об оптической связи.

Модуляция и демодуляция аналоговых сигналов

Наиболее простым и распространенным видом модуляции является амплитудная.

Пусть несущий высокочастотный сигнал описывается уравнением

$$h(t) = H\cos(\omega t + \varphi_0), \qquad (15.1)$$

где H – амплитуда высокочастотного сигнала; ω – частота (скорость); φ_0 – начальная фаза.

Несущий высокочастотный сигнал передается на сторону приема. Высокочастотный сигнал обладает большой энергией и способен преодолевать значительные расстояния. Напротив, полезный сигнал, как правило, имеет низкую частоту, поэтому для передачи такого сигнала осуществляют изменение (модуляцию) амплитуды (фазы или частоты) высокочастотного несущего сигнала по закону изменения полезного сигнала.

Идея амплитудной модуляции состоит в том, что значение H изменяется по закону полезного (информационного) сигнала, т. е.

$$H = H(t), \tag{15.2}$$

откуда

$$h(t) = H(t)\cos(\omega t + \varphi_0). \tag{15.3}$$

Иллюстрацию физической картины амплитудной модуляции дает рис. 15.1.

Пусть полезный сигнал изменяется по гармоническому закону:

$$H(t) = A_0 \cos(\xi t + \gamma_0). \tag{15.4}$$

Тогда результирующий модулированный сигнал будет таким:

$$h(t) = A_0 \cos(\xi\ t + \gamma_0)\cos(\omega\ t + \varphi_0) =$$

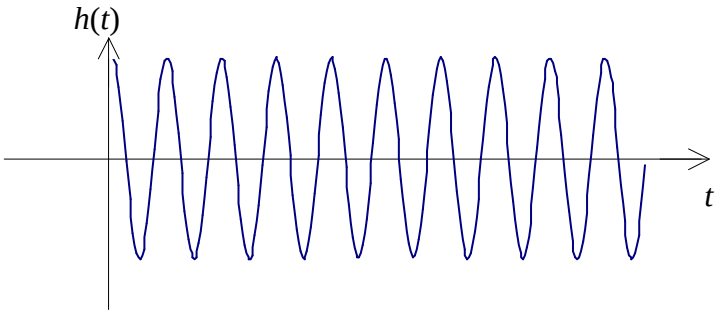
$$= \frac{A_0}{2}\left[\cos(\xi\ t + \omega\ t + \gamma_0 + \varphi_0) + \cos(\xi\ t + \gamma_0 - \omega\ t - \varphi_0)\right] =$$

$$= \frac{A_0}{2}\cos\left[(\xi\ + \omega)t + (\gamma_0 + \varphi_0)\right] + \frac{A_0}{2}\cos\left[(\omega - \xi)t + (\varphi_0 - \gamma_0)\right]. \tag{15.5}$$

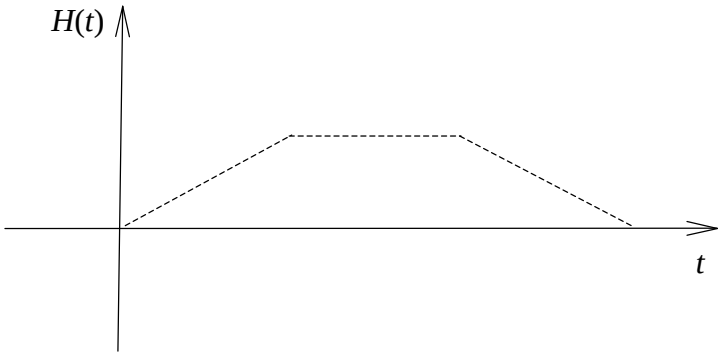
Мы видим, что спектр модулированного по амплитуде сигнала состоит из двух гармоник с частотами $\omega + \xi$ и $\omega - \xi$. Теперь, вспомнив преобразование Фурье, заметим, что любой сложный сигнал $H(t)$, который можно представить в виде ряда Фурье, будучи промодулированным по амплитуде, будет иметь конечный спектр частот $\xi_1 + \omega$, $\xi_2 + \omega$, …, $\xi_n + \omega$; $\omega - \xi_1$, $\omega - \xi_2$, …, $\omega - \xi_n$.

Таким образом, при амплитудной модуляции спектр модулированного сигнала располагается симметрично относительно несущей частоты ω, т. е. расщепляется на две части, симметрично расположенные относительно несущей частоты ω.

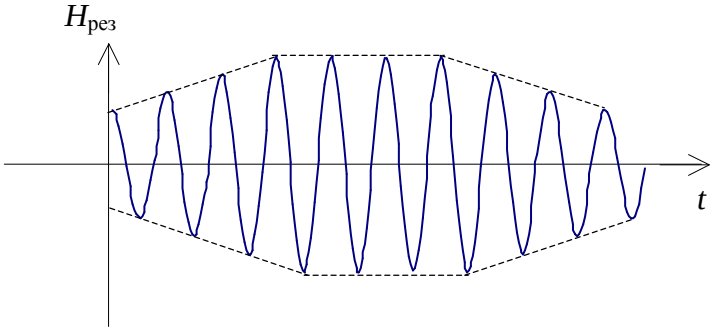
Для восстановления модулированного по амплитуде сигнала широко используют схему диодного детектора (рис. 15.2).



Высокочастотный гармоничный сигнал



Полезный сигнал



Результирующий сигнал, модулированный по амплитуде

Рис. 15.1

Входное напряжение подается на *LC*-контур. Параметры *L* и *C* подбирают так, чтобы возникало явление резонанса в контуре, при этом выполняется соотношение

$$\omega_{\text{р}} L = \frac{1}{\omega_{\text{р}} C} \text{ и } \omega_{\text{р}} = \sqrt{\frac{1}{LC}},$$

где $\omega_{\text{р}}$ – резонансная частота.

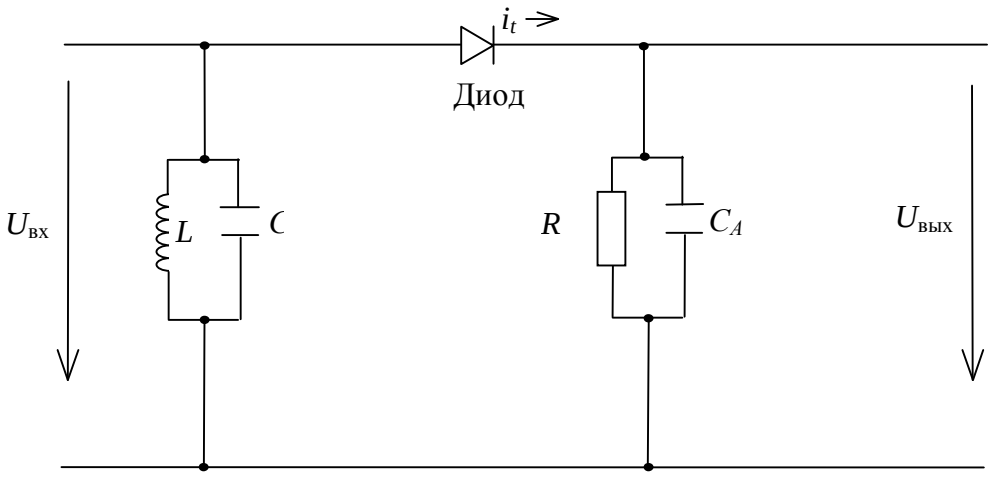


Рис. 15.2

Таким образом, входной сигнал с частотой $\omega_{\text{р}}$ будет резко усилен, а его амплитуда значительно превзойдет амплитуду составляющих, которые имеют другие частоты. Можно сказать, что входной *LC*-контур играет роль приемника, настроенного на заданную частоту $\omega_{\text{р}}$. В результате в контуре наводится эдс по закону электромагнитной индукции. Диод пропускает импульсы тока, изменяющиеся по закону модулированного сигнала (рис. 15.3).

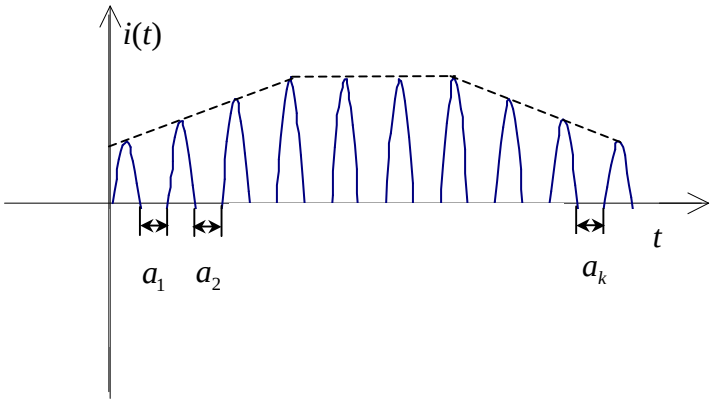


Рис. 15.3

Сигнал тока с диода несет в себе высокочастотную составляющую, «срезанную» по положительным полупериодам. Выходной *RC*-контур играет роль фильтра высоких частот. Поскольку конденсатор C_A не успевает разрядиться мгновенно на интервалах a_1, a_2, …, a_k (рис. 15.3), то напряжение $U_{вых}$ изменяется плавно, как показано на рис. 15.4.

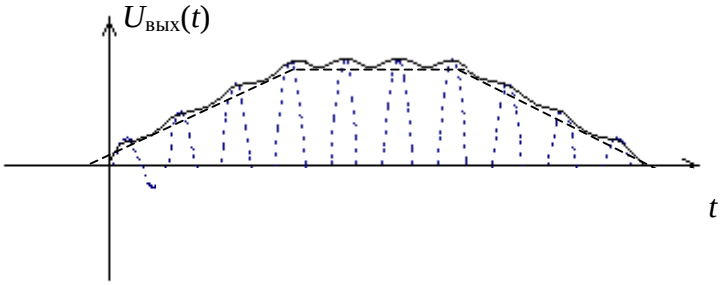


Рис. 15.4

Действительно, на интервалах a_1, a_2, …, a_k входной сигнал с приемного контура не поступает (ток i_1 отсутствует), так как диод не пропускает высокочастотные составляющие, которые соответствуют отрицательным полупериодам. Таким образом, входное напряжение в этот момент определяется напряжением на конденсаторе C_A, который медленно разряжается. В результате восстанавливается форма исходного полезного сигнала.

Лекция 16
ПРЕОБРАЗОВАНИЕ АНАЛОГОВЫХ СИГНАЛОВ В ЦИФРОВЫЕ И НАОБОРОТ

Цель. Рассмотреть схемы цифро-аналоговых и аналого-цифровых преобразователей и описать их работу.

Во многих случаях на практике необходимо преобразовывать аналоговые сигналы в цифровые и наоборот. Например, такая потребность связана с вводом данных в ЭВМ. Скажем, показатели датчиков температуры должны быть преобразованы в цифровую форму для ввода в ЭВМ. Преобразование аналогового сигнала в цифровой состоит в применении двух операций: дискретизации и квантования. Дискретизация означает переход от непрерывного сигнала к множеству значений, снимаемых через регулярные интервалы времени Δt. Напомним, что величина Δt должна удовлетворять теореме Котельникова:

$$\Delta t \le \frac{1}{2 f_{\max}},$$

где $f_{\max}$ – максимальная частота в конечном спектре аналогового сигнала.

Функцию дискретизации и квантования сигнала выполняют специальные микросхемы – аналого-цифровые преобразователи (АЦП). На один из входов АЦП поступает аналоговый сигнал из датчика. Существует большое число самых разнообразных датчиков. Например, в качестве датчика температуры может служить обычный диод (рис. 16.1).

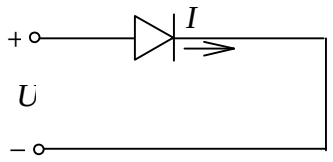


Рис. 16.1

Известно, что ток, проходящий через диод при заданном прямом напряжении U, определяется формулой

$$I = I_S(l^{\beta} - 1), \qquad (16.1)$$

где $\beta = \dfrac{qU}{2kT}$; k – постоянная Больцмана; T – температура (по Кельвину); I_S – обратный ток насыщения диода.

Логарифмируя (16.1), получим:

$$\ln(I + I_S) = \ln I_S + \frac{qU}{2kT},$$

откуда

$$U = \frac{[\ln(I + I_S) - \ln I_S]2kT}{q}. \qquad (16.2)$$

Из (16.2) видно, что напряжение на выходе схемы (16.1) линейно зависит от температуры.

Это выходное напряжение с датчика подается на аналоговый вход АЦП; на синхронизирующий вход АЦП подают тактовую частоту, в соответствии с которой выполняется дискретизация входного аналогового сигнала. Структурную схему АЦП можно представить так, как показано на рис. 16.2.

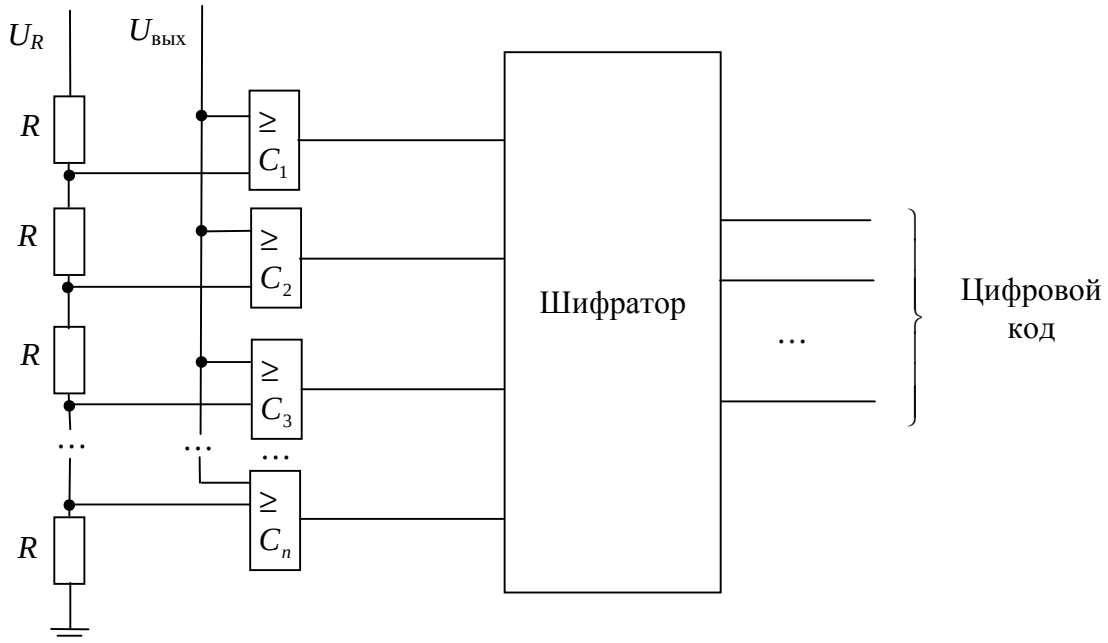


Рис. 16.2

На схеме (рис. 16.2) представлена линейка сопротивлений и соответствующих им компараторов (схем сравнения). По мере движения по линейке сопротивлений возрастает значение напряжения, подаваемого на компаратор с данного сопротивления. Компаратор выдает высокий уровень, если $U_{вх} \geq U_i$ (U_i – выходное напряжение с i-го по счету сопротивления R_i). На-против, компаратор выдает низкий уровень, если $U_{вх} \leq U_i$. Таким образом, на вход схемы шифратора подается набор общего вида:

$$\underbrace{11\ldots1}_{k}\underbrace{00\ldots0}_{m}$$

k единиц m нулей

$$k + m = 2^{n-1}$$

Этот набор преобразуется шифратором в соответствующее двоичное число на выходе. Заметим, что шифратор представляет булевскую схему, на входе которой имеется 2^{n-1} контакта, а на выходе – n.

Обратное преобразование цифрового сигнала в аналоговый реализует схема, которая называется цифро-аналоговый преобразователь (ЦАП). Схему ЦАП можно представить так, как показано на рис. 16.3.

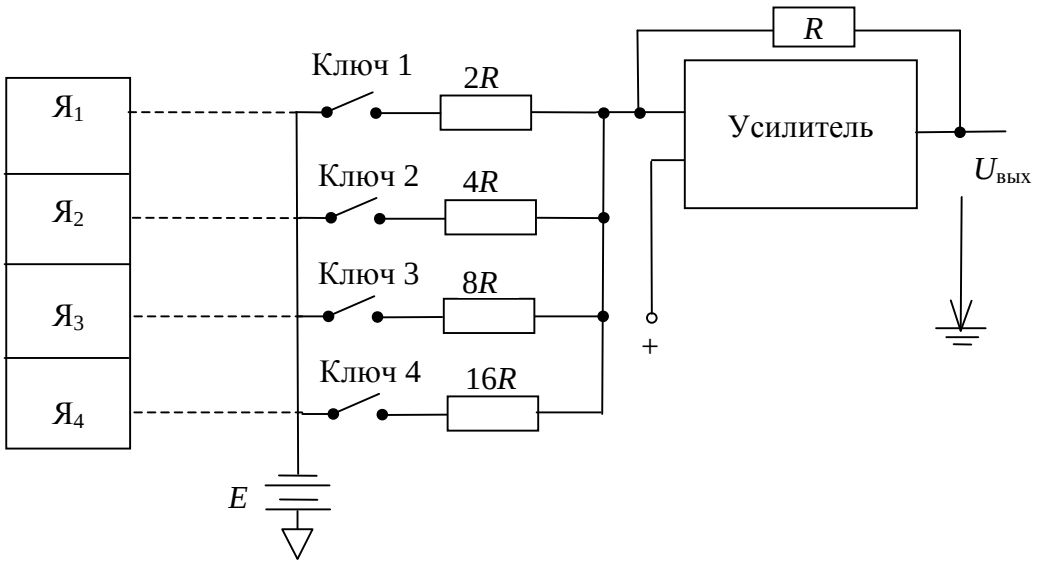


Рис. 16.3

Напряжение на выходе усилителя определяется по формуле

$$U_{\text{вых}} = \frac{R}{R_{\text{вх}}} E, \qquad (16.3)$$

где $R_{\text{вх}} = \alpha_1 2R + \alpha_2 4R + \alpha_3 8R + \alpha_4 16R$; $\alpha_i = 0$, если ключ разомкнут; $\alpha_i = 1$, если ключ замкнут.

Состоянием ключей управляют ячейки памяти $Я_1$, $Я_2$, …, $Я_4$. Если в ячейку записывается 1, то соответствующий ключ замыкается, иначе – размыкается. В зависимости от состояния ключей определяется $R_{\text{вх}}$ и значение $U_{\text{вых}}$ (16.3). Если все ключи разомкнуты, то на выходе усилителя нет напряжения.

Лекция 17
СЕМИУРОВНЕВАЯ МОДЕЛЬ ОТКРЫТОЙ СИСТЕМЫ

Цель. Изложить принципы семиуровневой архитектуры открытых систем. Привести описание назначения и функционирования каждого уровня.

В мае 1983 г. Международная организация стандартов приняла документ № 7498 «Базовая модель взаимосвязи открытых систем». Этот документ является стандартом, определяющим архитектуру современных сетей, услуги, которые должны предоставляться на каждом слое, и спецификации протоколов. В архитектуре различают следующие семь уровней (слоев):

1. физический;
2. канальный;
3. сетевой;
4. транспортный;
5. сеансовый;
6. представления данных;
7. программный.

На физическом уровне определяются требования к используемой физической среде передачи данных, электрическим характеристикам сигналов и волновых сопротивлений линий связи, а также способ связи с канальным уровнем.

Так, в качестве линии связи может использоваться витая пара проводов (телефонная линия связи), экранированный с помощью металлической оплетки электрический кабель, волоконно-оптический кабель для передачи световых сигналов и др. Для кодирования сигналов широко используется так называемая манчестерская кодировка, в которой «1» соответствует перепад уровня сигнала, «0» – отсутствие перепада (рис. 17.1).

Проверка перепада уровня сигнала «привязана» к тактовым синхронизирующим импульсам. Манчестерское кодирование является более надежным по сравнению с кодированием, в котором «1» соответствует высокий уровень, а «0» – низкий (или наоборот). Это объясняется тем, что наложение помехи на сигнал в целом не сказывается на величине перепада уровня его фронта.

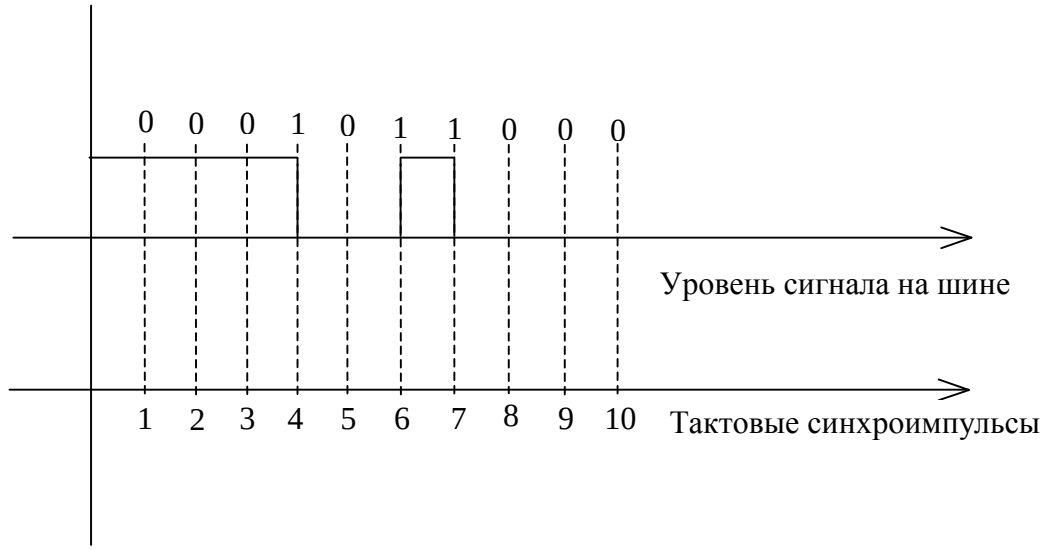


Рис. 17.1

В канальном уровне определяются требования по организации связи между двумя приемопередающими узлами (абонентами), подключенными к линии связи (каналу обмена). Проблема возникает при подключении многих абонентов к общему каналу. Используют различные стратегии доступа к каналу.

1. Метод передачи маркера. Под маркером понимают информационное сообщение, адресованное конкретному абоненту. Получив маркер, данный абонент становится «хозяином» линии связи (host-computer), т. е. может осуществлять передачу информации любому другому абоненту. Завершив передачу, host-машина передает маркер другому компьютеру (обычно соседнему).

2. Метод прослушивания несущей частоты. Данный метод основан на том, что каждый абонент прослушивает линию связи. Если передачи информации нет (слышна только тактовая синхронизирующая частота), то абонент начинает передачу информации. Может возникнуть ситуация, когда несколько узлов начнут передачу одновременно. В результате произойдет наложение одних данных на другие так, что получится jam. Приемник, естественно, не распознает направленных ему сообщений, и по истечении контрольного времени ЭВМ-передатчик установит наличие конфликта. ЭВМ-передатчик попытается повторно начать передачу, но через случайное время. В силу этого обстоятельства конфликтующие ЭВМ-передатчики повторно начнут передачу в

разное время, что и разрешит конфликт.

Существуют также другие методы обмена данными на канальном уровне.

Различают несколько видов (типов) сообщений на канальном уровне: сообщение-маркер, сообщение-данные, сообщение-прерывание.

Сообщение-данные, как правило, имеет следующий формат:

<Преамбула><Начало кадра><Адрес получателя>
<Адрес отправителя><Длина сообщения>
<Контрольный код><Данные><Конец кадра>

Преамбула указывает, что далее следуют сообщения. Начало каждого очередного сообщения устанавливает поле <Начало кадра>.

Назначение полей <Адрес получателя> и <Адрес отправителя> очевидно. Поле <Длина сообщения> задает число байт с данными. Поле <Контрольный код> содержит контрольные проверочные разряды для обнаружения и исправления ошибок данных. Например, в качестве контрольных кодов используют коды Хэмминга, циклические коды и др. Поле <Конец кадра> устанавливает конец передаваемого кадра данных. Большие сообщения разбиваются на кадры, передаваемые по отдельности.

Перейдем теперь к сетевому уровню. Сеть представляет собой множество каналов данных, соединяющих хосты, к которым подключаются локальные ЭВМ (рис. 17.2).

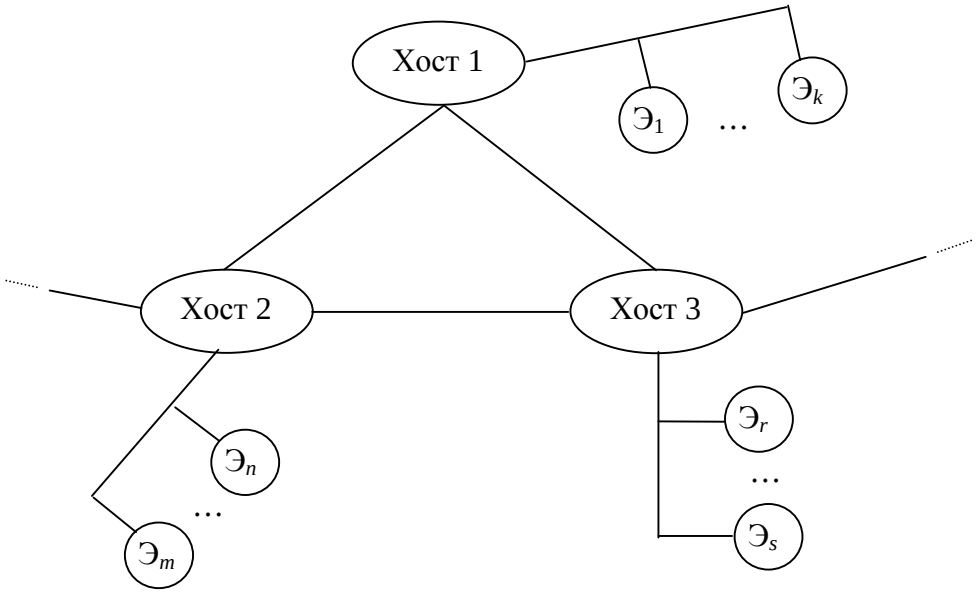


Рис. 17.2

Хост играет роль сервера-машины, обслуживающего подключенные к нему локальные ЭВМ ($Э_i$). В сети значительно усложняется задача управления трафиком сообщений. Различают два принципиально разных варианта управления трафиком: управление
на уровне каналов (сеть с коммутацией каналов) и управление на уровне пакетов (сеть с коммутацией пакетов). При коммутации каналов между передающей и принимающей ЭВМ на время передачи устанавливается жесткий путь по сети. Это означает, что другие ЭВМ не могут задействовать ни один из каналов передачи данных, входящих в назначенный путь. Такой вариант является не гибким, так как не учитывает потребности остальных ЭВМ. При коммутации пакетов большие сообщения разбиваются на отдельные пакеты, которые передаются независимо от других пакетов, входящих в сообщение. Путь, по которому доставляется каждый пакет, определяется динамически с учетом загрузки каналов, поэтому разные пакеты могут следовать по разным маршрутам. Итак, на сетевом уровне возникает задача управления трафиком. Решение этой важной задачи можно выполнить с помощью алгоритма пометок Дейкстры, который опишем на примере.

Рассмотрим сеть на рис. 17.3.

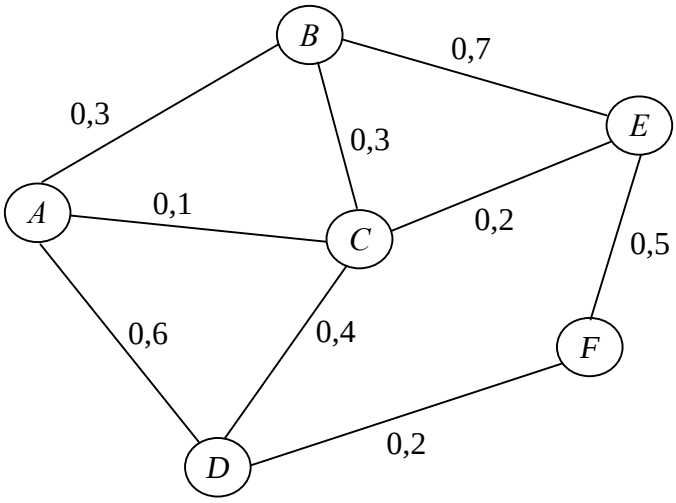


Рис. 17.3

Числа над ребрами указывают на соответствующий коэффициент загрузки канала. Этот коэффициент, по существу, представляет собой вероятность занятости канала. Пусть нужно найти наименее загруженный путь из узла сети A в узел сети F. Если этот путь образуют каналы с вероятностями занятости p_1, p_2, …, p_z, то вероятность занятости хотя бы одного канала можно найти по формуле

$$\rho = p_1 + p_2 + \ldots + p_z - p_1p_2 - p_1p_3 - \ldots - p_1p_z - \ldots -$$
$$- p_2p_3 - \ldots - p_2p_z - \ldots - p_{z-1}p_z + p_1p_2p_3 + p_1p_2p_4 + \ldots +$$
$$+ p_1p_2p_z - \ldots - (-1)^k\, p_1p_2\ldots p_k + \ldots \qquad (17.1)$$

Формулой (17.1) пользоваться крайне неудобно. Однако в методе пометок Дейкстры можно ограничиться только двумя узлами:

$$\rho_{ab} = p_a + p_b - p_{ab} = p_a + p_b - p_a p_b. \qquad (17.2)$$

Итак, воспользуемся формулой (17.2).

В методе Дейкстры каждому узлу, начиная с исходного узла A, последовательно присваиваются пометки. Пометка представляет собой значение, вычисленное с помощью формулы (17.1) для пути с наименьшей нагрузкой, ведущего из узла A в данный узел. Узлу A припишем пометку 0. Смотрим, какие узлы можно пометить из узла A. Это узлы B, C, D. Приписываем узлам B, C, D пометки,

равные соответствующим вероятностям занятости каналов *AB*, *AC*, *AD* (на рис. 17.4 пометки показаны в рамках).

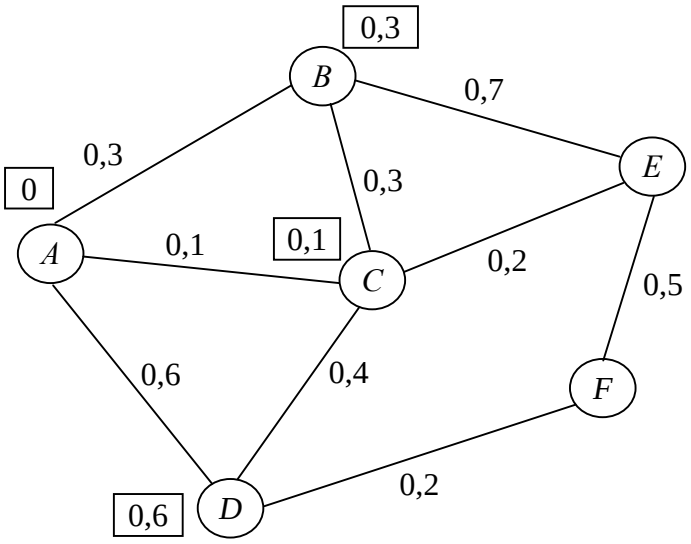


Рис. 17.4

Попытаемся пометить узел *C* из узла *B*:

$$\rho_C = 0{,}3 + 0{,}3 - 0{,}3 \cdot 0{,}3 = 0{,}51,$$

где 0,3 (первое слагаемое) – пометка узла *B*; 0,3 (второе слагаемое) – вероятность занятости канала *BC*.

Но узлу *C* уже приписана пометка 0,1, и $0{,}1 < 0{,}51$. Поэтому помечать узел *C* из узла *B* нецелесообразно. Попытаемся пометить узел *C* из узла *D*:

$$\rho_C = 0{,}1 + 0{,}4 - 0{,}1 \cdot 0{,}4 = 0{,}46.$$

Опять же пометка узла *C* из узла *D* нецелесообразна. Таким образом, узел *C* получает окончательную пометку 0,1. Но целесообразно пометить узел *B* из узла *C*:

$$\rho_D = 0{,}1 + 0{,}4 - 0{,}1 \cdot 0{,}4 = 0{,}46 < 0{,}6.$$

Следовательно, изменяем пометку узла *D* на 0,46. Для узла *B* пометку улучшить не удается. Теперь можно пометить узел *E* из узла *B*:

$$\rho_E = 0{,}3 + 0{,}7 - 0{,}3 \cdot 0{,}7 = 0{,}89$$

и из узла *C*:

$$\rho_E = 0{,}1 + 0{,}2 - 0{,}1 \cdot 0{,}2 = 0{,}28.$$

Ясно, что более выгодно пометить узел *E* из узла *C*. Теперь картина пометок примет новый вид (рис. 17.5).

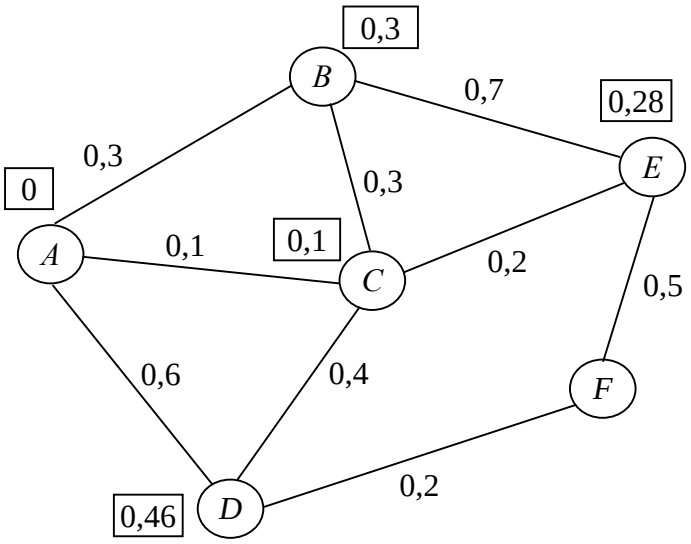


Рис. 17.5

Далее, помечаем узел *F*. Очевидно, что наилучшая пометка *F* получается из узла *D* и составляет

$$\rho_F = 0{,}46 + 0{,}2 - 0{,}46 \cdot 0{,}2 = 0{,}568.$$

Итак, наилучший путь, ведущий из узла *A* в узел *F*, такой:

$$A \to C \to D \to F.$$

На этом пути вероятность занятости хотя бы одного канала наименьшая.

Перейдем теперь к транспортному уровню. Этот уровень нужен, если имеются разнородные, связанные через шлюзы сети передачи данных. Этот уровень обеспечивает, следовательно, согласование представления информации, принятого в той или иной сети, адресацию в рамках каждой сети и межсетевую адресацию.

Остаются еще три уровня взаимодействия открытых систем:

– программный;
– представительный;
– сеансовый.

Данные уровни относятся уже к локальной ЭВМ, ее операционной системе и программным приложениям, из которых выполняется запрос соединения.

Задача сеансового уровня – организовать прием и передачу данных средствами операционной системы. Обычно обмен осуществляется запуском фонового процесса, периодически просматривающего содержимое специальных ячеек памяти – портов, куда помещаются данные, предназначенные для отсылки абоненту. На сеансовом уровне реализуется контакт «клиент – сервер». Программа (процесс) клиента запрашивает данные у процесса сервера, выполняющегося на другом компьютере. Взаимодействие клиента и сервера реализуется в пределах одного сеанса. Представительский уровень предоставляет услуги по конвертированию данных в требуемый формат. Например, таким форматом, используемым в Интернет, является HTTP (для передачи простого текста) или SMTP (для передачи почтовых сообщений) и др.

Наконец, программный уровень определяет набор средств по организации обмена с удаленными источниками в том или ином языке программирования.

Так, язык Java дает возможность реализовать сокетное соединение (socket-порт) на основе технологии вызова удаленных модулей RMI или CORBA, а также вызов по адресной ссылке удаленных объектов – Enterprise Java Beans. Эти средства зависят от языка программирования и сами по себе требуют отдельного рассмотрения.

Лекция 18
ПОМЕХОУСТОЙЧИВОЕ КОДИРОВАНИЕ. КОДЫ ХЭММИНГА

Цель. Рассмотреть математический аппарат помехоустойчивого кодирования по Хэммингу.

Проблема помехоустойчивого кодирования связана с тем, что при передаче или хранении данных возможно искажение отдельных разрядов. Имеется возможность не только обнаруживать искаженные разряды, но и исправлять их. Это делается с помощью дополнительных разрядов, включаемых в информационное сообщение в виде так называемого контрольного кода. Мы

рассмотрим помехоустойчивое кодирование на основе кодов Хэмминга и циклических кодов.

Построение кодов Хэмминга изложим на следующем простом примере. Пусть передаются четырехбитовые данные, например,

$$I = \underset{a_1}{1}\ \underset{a_2}{0}\ \underset{a_3}{1}\ \underset{a_4}{0}$$

Строим дополнительно три разряда $r = 3$. Число дополнительных разрядов для обнаружения и исправления одиночных ошибок выбирают из условия

$$r + n \le 2^r - 1, \qquad (18.1)$$

где n – число информационных разрядов; r – число дополнительных контрольных разрядов.

В примере $n = 4$, $r = 3$, и неравенство (18.1) выполняется.

Дополнительные контрольные разряды строят как функции общего вида:

$$f_i = a_{i1} \oplus a_{i2} \oplus \ldots \oplus a_{iz}, \qquad (18.2)$$

где операция $\oplus$ – сложение по модулю 2. Ее определяют согласно табл. 18.1.

Итак, для каждого дополнительного разряда используют свою функцию f_i. Нам нужны три такие функции.

Таблица 18.1

a_i	a_j	$a_i \oplus a_j$
0	0	0
0	1	1
1	0	1
1	1	0

Заметим, что эти функции, вообще говоря, произвольны, но они должны удовлетворять следующим условиям.

1. Они должны быть взаимно независимы.

2. Каждый из информационных разрядов a_i должен участвовать как минимум в одной функции.

Имеется еще одно, третье условие, которое мы отметим чуть ниже. Итак, пусть

$$f_1 = a_2 \oplus a_3 \oplus a_4;$$

$$f_2 = a_1 \oplus a_3 \oplus a_4; \qquad (18.3)$$

$$f_3 = a_1 \oplus a_2 \oplus a_4.$$

Определение. Функция f_i называется независимой от функций f_{j1}, …, f_{jk}, если имеется входной набор, на котором значения f_i не совпадают ни с одним из значений f_{j1}, …, f_{jk}.

Убедимся, например, что значение f_1 на наборе $a_1 = 0$, $a_2 = 1$, $a_3 = 1$, $a_4 = 1$ отличается от значений f_2 и f_3. Поэтому f_1 не зависит от функций f_2, f_3. Для информационного слова $I = 1010$ определяем дополнительные контрольные разряды:

$$e_1 = f_1(I) = 0 \oplus 1 \oplus 0 = 1;$$

$$e_2 = f_2(I) = 1 \oplus 1 \oplus 0 = 0;$$

$$e_3 = f_3(I) = 1 \oplus 0 \oplus 0 = 1.$$

Итак, передаваемое сообщение имеет такой вид

$$\underset{a_1}{1}\ \underset{a_2}{0}\ \underset{a_3}{1}\ \underset{a_4}{0}\ \ \underset{e_1}{1}\ \underset{e_2}{0}\ \underset{e_3}{1}$$

Теперь посмотрим, как выполняется обнаружение и исправление ошибок.

При использовании помехоустойчивого кодирования Хэмминга строят так называемые синдромные матрицы. Синдромную матрицу M определяют непосредственно из представления кодировочных функций типа (18.3). Обратимся за иллюстрацией к табл. 18.2.

Таблица 18.2

$M =$

a_1	a_2	a_3	a_4	Синдром
0	0	0	1	1 1 1
0	0	1	0	1 1 0
0	1	0	0	1 0 1
1	0	0	0	0 1 1

Синдромную матрицу M заполняют для указанных в табл. 18.2 комбинаций информационных разрядов. К каждой такой комбинации все $a_i = 0$, кроме одного (ошибочного). В итоге, например, строка

$$\underset{0}{a_1}\ \underset{0}{a_2}\ \underset{0}{a_3}\ \underset{1}{a_4}$$

с ошибкой в четвертом разряде дает нам синдром 111, вычисленный с помощью (18.3).

Пусть в переданном сообщении I произошел сбой во втором разряде и было получено сообщение

$$I' = \underset{a_1}{1}\ \underset{\underline{a_2}}{1}\ \underset{a_3}{1}\ \underset{a_4}{0}\ \underset{e_1}{1}\ \underset{e_2}{0}\ \underset{e_3}{1}$$

По формуле (18.3) вычисляем дополнительные контрольные разряды для I':

$$e_1'e_2'e_3' = 000.$$

Итак, выпишем отдельно принятые контрольные разряды и те, которые должны быть, и найдем их поразрядные суммы по модулю 2 ($\oplus$):

$$\begin{array}{c} e_1e_2e_3: 1\ 0\ 1 \\ e_1'e_2'e_3': 0\ 0\ 0 \quad \oplus \\ \hline \text{Синдром: } 1\ 0\ 1 \end{array}$$

Эта сумма и даст нам синдром 101. По синдромной таблице определяем ошибочный разряд (строка 3 в табл. 18.2 соответствует второму разряду a_2, где стоит 1). Итак, ошибочный разряд найден.

Теперь, допустим, сбой произошел в одном из контрольных разрядов. Например, принято сообщение

$$I'' = \underset{a_1'}{1}\ \underset{a_2'}{0}\ \underset{a_3'}{1}\ \underset{a_4'}{0}\ \underset{e_1'}{0}\ \underset{e_2'}{0}\ \underset{e_3'}{1}$$

Для этого сообщения подсчитываем контрольные разряды:

$$e_1''e_2''e_3'' = 101.$$

Находим синдром:

$$\begin{array}{c} e_1'e_2'e_3': 0\ 0\ 1 \\ e_1''e_2''e_3'': 1\ 0\ 0 \quad \oplus \\ \hline \end{array}$$

Синдром: 1 0 0

В табл. 18.2 такого значения нет. Вывод: информационные разряды не содержат ошибки. Аналогичным образом легко проверить, что сбои в других контрольных разрядах дают ту же картину. Обратим внимание читателя на то, что речь идет про одиночные сбои(!).

Теперь нам следует вернуться к третьему условию по выбору кодирующих функций f_i. Напомним, первое условие требует их независимости, второе – чтобы каждый информационный разряд участвовал в записи хотя бы одной функции. Третье условие требует, чтобы синдром, вычисленный для сбоя контрольного разряда,

не совпадал с синдромом, вычисленным для сбоя какого-нибудь информационного разряда. Как проверить это условие? Достаточно взять информационный код ‘0000’, найти для него дополнительные контрольные разряды (получим для этого случая ‘000’), а затем вычислить синдромы для трех вариантов сбоев в контрольных разрядах: 100, 010, 001. Отсюда легко получить некоторую общую рекомендацию. Мы видим, что синдромы, вычисленные в случае сбоя контрольного разряда, содержат ровно одну «1». Поэтому кодирующие функции следует подбирать так, чтобы они давали синдромы с двумя и более единицами.

Лекция 19
ЦИКЛИЧЕСКИЕ КОДЫ

Цель. Рассмотреть математический аппарат помехоустойчивого кодирования на основании циклических полиномов.

Пусть подлежит отправке информационное сообщение

$$I = 1010.$$

С данным сообщением можно сопоставить следующий полином

$$\begin{array}{c} 1\ \ 0\ \ 1\ \ 0 \\ |\ \ |\ \ |\ \ | \\ x^3\ x^2\ x^1\ x^0 \\ \hline 1\cdot x^3 + 0\cdot x^2 + 1\cdot x^1 + 0\cdot x^0 = x^3 + x \end{array}$$

Коэффициенты у полинома, полученного таким способом, равны 0 или 1 (члены полинома с нулевыми коэффициентами просто не выписываем). Далее такой полином будем называть циклическим. Особенностью циклических полиномов является то, что операции вычитания над полем таких полиномов нет – операция вычитания заменяется операцией сложения полиномов. Более того, операция сложения выполняется также специфически. Никаких переносов не существует. Если число слагаемых x^k с одной и той же степенью k при x четное, то результатом будет 0; если нечетное – то результатом будет x^k.

Пример.

$$P_1 = x^3 + x + 1,\ \ P_2 = x^4 + x^2 + x + 1;$$

$$P_1 + P_2 = x^4 + x^3 + x^2 + \underbrace{(x + x)}_{1\ 2\ 3\ \text{четное}} + \underbrace{(1 + 1)}_{1\ 2\ 3\ \text{четное}} = x^4 + x^3 + x^2.$$

Умножение выполняется обычным образом, но сохраняется введенное правило сложения.

Пример.

а) $$P_1 = x^2 + 1,$$

$$P_2 = x + 1,$$

$$P_1P_2 = (x+1)(x^2+1) = x^3 + x + x^2 + 1.$$

b) $$P_1 = x^3 + x + 1,$$

$$P_2 = x + 1,$$

$$P_1P_2 = (x^3 + x^2 + 1)(x+1) = x^4 + x^3 + x^2 + x + x + 1 =$$
$$= x^4 + x^3 + x^2 + 1.$$

Определение. Циклический полином P называется *первичным*, если его нельзя представить как произведение $P = P_1P_2$, где P_1 и P_2 отличны от 1 и самого P.

Первичными полиномами, например, являются $x + 1$, x, $x^3 + x + 1$, $x^3 + x^2 + 1$ и т. д.

Итак, прежде всего определяют дополнительные разряды, присоединяемые к передаваемому информационному сообщению. Для выбора числа дополнительных разрядов будем использовать результат Боуза и Чоудхури, который состоит в следующем. Пусть n – число информационных разрядов, а k – число дополнительных разрядов. Требуется выполнение следующих условий:

a) $n + k = 2^m - 1$ для некоторого целого m (любого подходящего);

b) $k \le mt_u$, где t_u – число обнаруживаемых ошибок.

Для $t_u = 1$ получим $n + k = 2^m - 1$, $k \le m$.

Сложив эти неравенства и полагая $k = m$, имеем

$$2^k \ge n + k - 1.$$

Это соотношение в точности копирует (18.1) для кодов Хэмминга.

В нашем случае возьмем $k = 3$. Значения дополнительных разрядов получают следующим образом. Умножают исходный полином на полином x^k и делят на первичный полином степени k (например, возьмем $x^3 + x + 1$). Остаток от деления и даст нам дополнительные разряды.

Итак, имеем:

1) $(x^3 + x)x^3 = x^6 + x^4$;

2) $$\begin{array}{l|l} x^6 + x^4 & x^3 + x + 1 \\ \hline x^6 + x^4 + x^3 & x^3 + 1 \\ \hline x^3 & \end{array}$$

$$\frac{x^3 + x + 1}{x + 1}$$ – остаток

Еще раз напомним, что операции вычитания над полем циклических полиномов нет. Вычитание заменяется сложением, которое выполняется по сформулированным выше правилам.

Остаток $x + 1$ можно записать так:

$$0 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$$

(поскольку у нас число дополнительных разрядов $k = 3$).

Этому полиному соответствует двоичный код 001.

Итак, мы должны передать число I = 1010. Для него мы определили дополнительные контрольные разряды 011. При этом нами использован первичный полином степени $k = 3$ – $x^3 + x + 1$. Отправляемое сообщение имеет следующий вид:

1010’011

Теперь рассмотрим, как обнаруживаются и исправляются одиночные ошибки.

Допустим, был искажен второй разряд и, следовательно, принято число

$$I' = 1\underline{1}10011$$

Переведем его в полиномиальную форму:

$$I' = x^6 + x^5 + x^4 + x + 1.$$

Теперь надо разделить I' на первичный полином $x^3 + x + 1$. Если деление произойдет нацело без остатка, то ошибки нет. В противном случае сразу фиксируется факт ошибки и дальнейшие действия направлены на исправление ошибки. В нашем случае деление дает

$$\begin{array}{l|l} x^6 + x^5 + x^4 + x + 1 & x^3 + x + 1 \\ \hline x^6 + x^4 + x^3 & x^3 + x^2 \\ \hline x^5 + x^3 + x + 1 & \\ x^5 + x^3 + x^2 & \\ \hline x^2 + x + 1 & \end{array}$$ – остаток

Остаток указывает на ошибку. Производим циклический сдвиг принятой комбинации на один разряд влево:

$$1110011$$

$$I'' = 1100111$$

Полученное таким образом число снова делят на первичный полином. Если в остатке получается число, отличное от 1, то производят очередной циклический сдвиг влево и снова делят и т. д.
до тех пор, пока в остатке, наконец, не получится ровно 1. В нашем случае второе деление дает:

$$\begin{array}{l|l}
x^6+x^5+x^2+x+1 & x^3+x+1 \\
\underline{x^6+x^4+x^3} & \underline{x^3+x^2+1} \\
x^5+x^4+x^3+x^2+x+1 & \\
\underline{x^5+x^3+x^2} & \\
x^4+x+1 & \\
\underline{x^4+x^2+x} & \\
x^2+x & \text{– остаток}
\end{array}$$

Остаток не равен 1. Производим циклический сдвиг:

$$1100111$$

$$I'' = 1001111.$$

Выполняем очередное деление

$$\begin{array}{l|l}
x^6+x^3+x^2+x+1 & x^3+x+1 \\
\underline{x^6+x^4+x^3} & \underline{x^3+x} \\
x^4+x^2+x+1 & \\
\underline{x^4+x^2+x} & \\
1 & \text{– остаток}
\end{array}$$

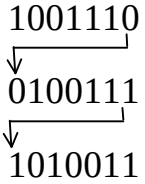
Мы получили в остатке 1. Это означает, что процесс делений и сдвигов прекращается. Для исправления ошибки нужно к последнему делимому добавить 1:

$$\begin{array}{r}
x^6+x^3+x^2+x+1 \\
\underline{+1} \\
x^6+x^3+x^2+x
\end{array}$$

Сложение выполняется по правилам циклических полиномов (без переносов). Теперь необходимо получить двоичное представление этого полинома:

1001110

Наконец, нужно выполнить обратный циклический сдвиг вправо столько раз, сколь раз до этого сдвигали число влево:



Мы получили восстановленное число.

Таким образом, циклические коды позволяют обнаруживать и исправлять многократные ошибки.

Лекция 20
ПРИЕМ И ФИЛЬТРАЦИЯ СИГНАЛОВ

Ц е л ь. Изложить суть задачи фильтрации.

20.1. Сущность фильтрации

Фильтром называется устройство, предназначенное для изменения спектра сигнала с целью получения заданных составляющих спектра. Мы уже видели, что демодуляция связана с устранением высокочастотной составляющей сигнала, которая выполняла роль несущей частоты. Различают следующие типы фильтров:

- фильтры низких частот (ФНЧ);
- фильтры высоких частот (ФВЧ);
- полосовые фильтры (ПФ);
- режекторные фильтры (РФ).

ФНЧ пропускают только низкочастотные составляющие спектра сигнала; ФВЧ, напротив, – только высокочастотные. Полосовой фильтр пропускает ту часть спектра, которая определяется заданным диапазоном (полосой) частот; напротив, РФ подавляет все частоты в пределах заданного диапазона. Основными параметрами любого фильтра являются следующие:

1) частота среза (ω_c). Эта частота задает границу полосы пропускания. У ФНЧ и ФВЧ имеется одна частота среза; у ПФ и РФ – две;

2) центральная частота (ω_0). Определена для ПФ как $\omega_0 = \sqrt{\omega_1 \omega_2}$, ω_1 – нижняя, а ω_2 – верхняя частота среза. Относительно центральной частоты ω_0 амплитуда сигнала уменьшается равномерно;

3) добротность Q. Определяется как отношение

$$Q = \frac{\omega_0}{\omega_2 - \omega_1};$$

4) передаточная функция фильтра

$$W = \frac{U_{вых}}{U_{вх}},$$

где $U_{вых}$ – значение напряжения на выходе фильтра; $U_{вх}$ – значение напряжения на входе фильтра.

Одними из наиболее широко используемых являются фильтры Баттерворта, которые очень просты в изготовлении и представляют обычный контур, содержащий сопротивление (R), индуктивность (L) и емкость (C) (рис. 20.1)

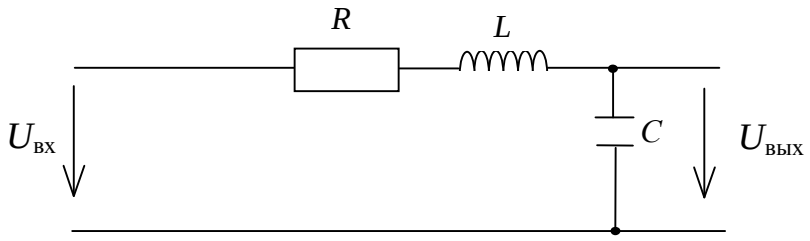
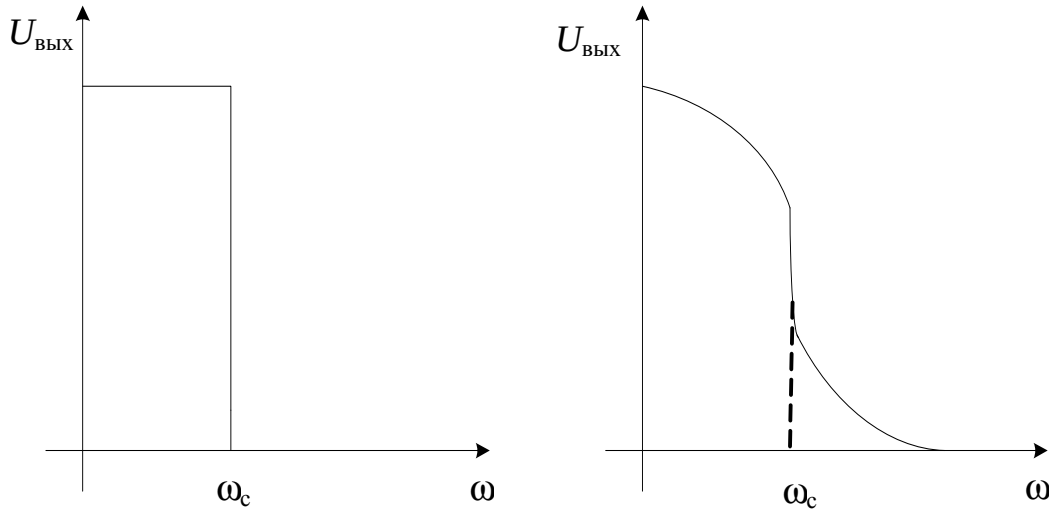


Рис. 20.1

Работу фильтра низких частот Баттерворта иллюстрирует его амплитудно-частотная характеристика (АЧХ), показанная на рис. 20.2.



Идеальная АЧХ АЧХ ФНЧ Баттерворта

Рис. 20.2

Амплитудно-частотная характеристика (АХЧ) представляет зависимость амплитуды выходного сигнала от частоты входного сигнала. При графическом представлении АХЧ на оси абсцисс откладывают частоту входного сигнала, а на оси ординат – амплитуду выходного сигнала. При этом амплитуда входного сигнала фиксируется на одном уровне. Идеальная АХЧ на рис. 20.2 показывает, что постоянный сигнал на входе фильтра преобразуется
в постоянный сигнал на выходе в пределах диапазона частот [0; ω_c]. Реальный фильтр низких частот Баттерворта приводит к искажению формы выходного сигнала (рис. 20.2).

Для системы на рис. 20.1 можно составить уравнение согласно закону Кирхгофа:

$$L\frac{di}{dt} + iR + U_{\text{вых}} = U_{\text{вх}}.$$

$$i = C\frac{dU_{\text{вых}}}{dt}.$$

Получаем следующее дифференциальное уравнение:

$$LC\frac{d^2U_{\text{вых}}}{dt^2} + RC\frac{dU_{\text{вых}}}{dt} + U_{\text{вых}} = U_{\text{вх}}(t). \qquad (20.1)$$

Его в общем виде можно записать так:

$$a_2x''(t) + a_1x'(t) + a_0x(t) = f(t), \qquad (20.2)$$

где $x'(t) = \frac{dx(t)}{dt}$; $x''(t) = \frac{d^2x(t)}{dt^2}$; $f(t)$ – входное воздействие (входное напряжение); $x(t)$ – выходной сигнал.

Решение таких уравнений изучают в курсе теории дифференциальных уравнений. Общее решение ищут в форме

$$x(t) = x_{\text{общ}}(t) + x_{\text{част}}(t), \qquad (20.3)$$

где $x_{\text{общ}}(t)$ – общее решение уравнения (20.2) с нулевой правой частью ($f(t) = 0$); $x_{\text{част}}(t)$ – какое-то частное решение уравнения (20.2) для заданной правой части.

Наша задача состоит в том, чтобы показать фильтрующие свойства схемы на рис. 20.1. Итак, мы хотим убедиться, что

низкочастотные сигналы такая схема пропускает на выход без изменений, а высокочастотные – подавляет. Поэтому рассмотрим решение двух уравнений:

$$a_2x''(t)+a_1x'(t)+a_0x(t)=1; \tag{20.4}$$

$$a_2x''(t)+a_1x'(t)+a_0x(t)=\cos\omega t. \tag{20.5}$$

Уравнение (20.4) соответствует постоянному сигналу на входе, а уравнение (20.5) – колебательному процессу. Рассмотрим сначала общее решение однородного уравнения

$$a_2x''(t)+a_1x'(t)+a_0x(t)=0.$$

Для решения таких дифференциальных уравнений составляют характеристическое алгебраическое уравнение

$$a_2r^2+a_1r+a_0=0. \tag{20.6}$$

Действительно, подобное уравнение получают, если используют замену $x(t)=Ae^{rt}$. Тогда $x'(t)=Are^{rt}$ и $x''(t)=Ar^2e^{rt}$, что дает (20.6) после сокращения на Ae^{rt}. Левая часть уравнения (20.6) – это обычный квадратичный трехчлен. Имеются следующие случаи.

1. Корни r_1 и r_2 – действительные и различные числа. Тогда общее решение записывают так:

$$x_{\text{общ}}(t)=c_1e^{r_1t}+c_2e^{r_2t},$$

где постоянные c_1, c_2 находят из задаваемых начальных условий $(x(0)=\alpha_1,\ x'(0)=\alpha_2)$.

2. Корни r_i совпадают, т. е. $r_1=r_2$. В этом случае говорят, что уравнение содержит действительный корень кратности 2. Ему соответствует решение:

$$x_{\text{общ}}(t)=(c_1+c_2t)e^{r_it}.$$

3. Корни комплексные:

$$r_1=\alpha+i\beta,\ r_2=\alpha-i\beta.$$

Тогда $x_{\text{общ}}(t)=e^{\alpha t}(c_1\cos\beta t+c_2\sin\beta t)$.

Среди корней характеристических алгебраических уравнений произвольных степеней могут присутствовать и действительные, и комплексные различной кратности. В этом случае общее решение записывается как сумма решений, соответствующих каждому из корней [16–18] с учетом его кратности.

Частное решение неоднородного уравнения (20.2) находят методом подбора или вариации произвольных постоянных. В случае когда

$$f(t) = e^{zt}[R_{n_1}(t)\cos\beta t + P_{n_2}(t)\sin\beta t],$$

решение ищут в виде

$$x_{\text{част}}(t) = t^s e^{zt}[H_m(t)\cos\beta t + G_m(t)\sin\beta t], \qquad (20.7)$$

где $R_{n_1}(t)$ – многочлен степени n_1 от t; $P_{n_2}(t)$ – многочлен степени n_2 от t; $H_m(t)$ и $G_m(t)$ – многочлены степени m с неотрицательными коэффициентами, причем

$$m = \max(n_1, n_2).$$

Наконец,

$$S = \begin{cases} 0, \text{ если } z + i\beta \text{ не совпадает ни с одним из корней характеристического уравнения;} \\ k, \text{ если } z + i\beta \text{ совпадает с корнем характеристического уравнения кратности } k. \end{cases}$$

Пусть, например, характеристическое уравнение имеет вид

$$r^2 + 2r + 5 = 0.$$

Находим корни:

$$r_1 = -1 + 2i, \quad r_2 = -1 - 2i.$$

Общее решение соответствующего однородного дифференциального уравнения заменяют следующим образом:

$$x_{\text{общ}}(t) = e^{-t}(c_1 \cos 2t + c_2 \sin 2t).$$

Теперь найдем частное решение (20.4) в виде

$$x_{\text{част}}(t) = A,$$

где A – константа.

Значение A определяем в результате подстановки $x_{\text{част}}(t) = A$ в уравнение (20.4):

$$a_2 \cdot 0 + a_1 \cdot 0 + a_0 \cdot A = 1 \ (a_2 = 1,\ a_1 = 2,\ a_0 = 5).$$

Имеем $A = \frac{1}{5}$ и $x_{\text{част}}(t) = \frac{1}{5}$.

Общее решение получаем в форме

$$x(t) = e^{-t}(c_1 \cos 2t + c_2 \sin 2t) + \frac{1}{5}.$$

Пусть имеем начальные условия:

$$x(0) = 1;\ \ x'(0) = 1.$$

Находим c_1, c_2:

$$1 = 1(c_1 + 0) + \frac{1}{5} \to c_1 = \frac{4}{5};$$

$$x'(t) = -e^{-t}(c_1 \cos 2t + c_2 \sin 2t) + e^{-t}(-2c_1 \sin 2t + 2c_2 \cos 2t);$$

$$0 = 1 \cdot \frac{4}{5} + 1 \cdot 2c_2 \to c_2 = -\frac{2}{5}.$$

В итоге получаем окончательно

$$x(t) = e^{-t}\left(\frac{4}{5}\cos 2t - \frac{2}{5}\sin 2t\right) + \frac{1}{5}. \qquad (20.8)$$

Итак, реакцией на постоянное входное воздействие схемы на рис. 20.1 является низкочастотный сигнал, так как величина $e^{-t}\left(\frac{4}{5}\cos 2t - \frac{2}{5}\sin 2t\right)$ быстро стремится к 0 при росте t. Это означает, что низкочастотная (слабоизменяющаяся) составляющая проходит на выход фильтра. Теперь рассмотрим, что происходит с высокочастотной составляющей.

Ищем частное решение уравнения (20.5) в виде

$$x_{\text{част}}(t) = H \cos \omega t + F \sin \omega t. \qquad (20.9)$$

При этом общее решение запишется в форме

$$x(t) = e^{-t}(c_1 \cos 2t + c_2 \sin 2t) + H \cos \omega t + F \sin \omega t.$$

Полагаем ω достаточно большим для высокочастотных составляющих сигнала. Подставляем (20.9) в (20.5), приняв, как и ранее, $a_2 = 1,\ a_1 = 2,\ a_0 = 5$. Получаем

$$-H\omega^2 \cos\omega t - F\omega^2 \sin\omega t - 2H\omega \sin\omega t + F\omega \cos\omega t + \\ + 5H\cos\omega t + 5F\sin\omega t = \cos\omega t.$$

Отсюда, группируя отдельно члены с $\cos\omega t$ и $\sin\omega t$, имеем:

$$\cos\omega t\left(-H\omega^2 + 2F\omega + 5H\right) + \sin\omega t\left(-F\omega^2 - 2H\omega + 5H\right) = \cos\omega t.$$

Одним из возможных условий является следующее:

$$\begin{cases} -H\omega^2 + 2F\omega + 5H = 1, \\ -F\omega^2 - 2H\omega + 5F = 0. \end{cases}$$

Отсюда

$$H = \frac{5 - \omega^2}{(5 - \omega^2)^2 + 4\omega^2};$$

$$F = \frac{2\omega}{(5 - \omega^2)^2 + 4\omega^2}.$$

Теперь очевидно, что с ростом t составляющая $e^{-t}(c_1 \cos 2t + c_2 \sin 2t)$ общего решения стремится к 0. Кроме того, и H, и F стремятся к 0 с ростом частоты ω. Поэтому высокочастотные составляющие входного сигнала подавляются (близки к 0). Итак, нами обоснованы фильтрующие свойства схемы на рис. 20.1. Разумеется, качество фильтрации связано с подбором параметров R, L, C.

Заметим, что также широко применяются активные фильтры, в состав которых входит схема усиления сигнала (усилитель). Вопросы, связанные с проектированием таких фильтров, мы рассматривать не будем.

20.2. Передаточная функция фильтра

Обобщением уравнения (20.2) является уравнение вида

$$a_n x^{(n)}(t) + a_{n-1} x^{(n-1)}(t) + \ldots + a_1 x^{(1)}(t) + a_0 x(t) = f(t), \qquad (20.10)$$

где $x(t)$ и $f(t)$ – соответственно выход и вход системы.

Рассмотрим функцию

$$h(t) = \frac{x(t)}{f(t)}, \qquad (20.11)$$

представляющую отношение выходного сигнала к входному для произвольного момента времени $t \geq 0$. Если бы $h(t)$ нам была известна, то по сигналу на входе системы в момент t на основании функции входного сигнала $f(t)$ можно было бы определить сигнал на выходе системы в момент t.

Однако по записи (20.10) найти $h(t)$ не удается. В связи с этим используют преобразование Лапласа произвольной непрерывной ограниченной функции $g(t)$:

$$G(p) = \int_0^\infty g(t) e^{-pt} dt. \qquad (20.12)$$

Нам нужно применить это преобразование к (20.10); в частности, нужно найти преобразование от производных $x'(t)$, $x''(t)$ и т. д.

Найдем

$$\int_0^\infty g'(t) e^{-pt} dt = \int_0^\infty e^{-pt} dg(t) = e^{-pt} g(t)\Big|_0^\infty - \int_0^\infty g(t) d(e^{-pt}) =$$

$$= -g(0) + p\int_0^\infty g(t) e^{-pt} dt = pG(p) - g(0).$$

Обычно полагают нулевые начальные условия, т. е. $g(0) = 0$, тогда

$$\int_0^\infty g'(t) e^{-pt} dt = pG(p).$$

Найдем по аналогии

$$\int_0^\infty g''(t)e^{-pt}dt = \int_0^\infty e^{-pt}dg'(t) = e^{-pt}g'(t)\Big|_0^\infty - \int_0^\infty g'(t)de^{-pt} =$$

$$= -g'(0) + p\int_0^\infty g'(t)e^{-pt}dt = p^2G(p) - g'(0) - pg(0)$$

(и снова для нулевых начальных условий это дает нам $p^2G(p)$).

Очевидно, в общем случае

$$\int_0^\infty g^{(n)}(t)e^{-pt}dt = p^{(n)}G(p) - p^{(n-1)}g(0) - p^{(n-2)}g'(0) - \ldots - g^{(n-1)}(0).$$

Теперь преобразованием Лапласа для (20.10) является

$$a_n p^n X(p) + a_{n-1}p^{n-1}X(p) + \ldots + a_1 p^1 X(p) + a_0 X(p) = F(p). \quad (20.13)$$

Под передаточной функцией системы понимается отношение преобразования Лапласа входной функции сигнала к преобразованию Лапласа выходной функции сигнала при нулевых начальных условиях [19]:

$$H(p) = \frac{X(p)}{F(p)} = \frac{1}{a_n p^n + a_{n-1}p^{n-1} + \ldots + a_1 p^1 + a_0}.$$

Итак,

$$X(p) = H(p)F(p). \quad (20.14)$$

Можно показать, что уравнению (20.14) соответствует свертка оригиналов

$$x(t) = \int_0^\infty h(\tau)\, f(t-\tau)d\tau. \quad (20.15)$$

Итак, мы установили следующее: фильтр Баттерворта реализует на самом деле преобразование входного сигнала согласно уравнению свертки (20.15). Зная $h(t)$, можно найти выходной сигнал $x(t)$ по входному сигналу $f(t)$. Передаточную функцию $H(p)$ определяют из общего уравнения (20.10).

Пример. Рассмотрим уравнение

$$x''(t) - 5x'(t) + 6x(t) = 1;$$

$$x(0) = x'(0) = 0.$$

Запишем преобразование Лапласа для левой и правой частей уравнения:

$$p^2 X(p) - 5pX(p) + 6X(p) = \frac{1}{p};$$

$$\left(\int_0^\infty 1 e^{-pt} dt = -\frac{1}{p} \int_0^\infty de^{-pt} = -\frac{1}{p} e^{-pt} \Big|_0^\infty = \frac{1}{p} \right).$$

Далее получим

$$\underbrace{X(p)[p^2 - 5p + 6]}_{W(p)} = \underbrace{\frac{1}{p}}_{F(p)}.$$

Находим:

$$X(p) = \frac{F(p)}{W(p)} = \frac{1}{p(p^2 - 5p + 6)} = \frac{1}{p(p-2)(p-3)} =$$

$$= \frac{A}{p} + \frac{B}{p-2} + \frac{C}{p-3}.$$

Отсюда легко определить: $A = \frac{1}{6};\ B = -\frac{1}{2};\ C = \frac{1}{3}.$

Наша задача – найти $x(t)$ по изображению $X(p)$. Воспользуемся известным результатами (табл. 20.1).

Таблица 20.1

$x(t)$	$X(p)$	$x(t)$	$X(p)$
c	$\frac{c}{p}$	$\cos \alpha t$	$\frac{p}{p^2 + \alpha^2}$
t	$\frac{1}{p^2}$	$\frac{1}{\alpha} e^{-\frac{t}{\alpha}}$	$\frac{1}{1 + \alpha p}$
t^n	$\frac{n!}{p^{n+1}}$	$\frac{1}{\alpha}\left(e^{\alpha t - 1}\right)$	$\frac{1}{p(p - \alpha)}$
$t^n e^{\alpha t}$	$\frac{n!}{(p - \alpha)^{n+1}}$	$t \sin \alpha t$	$\frac{2p\alpha}{\left(p^2 + \alpha^2\right)^2}$

при $n = 0$ $e^{\alpha t}$	$\frac{n!}{p-\alpha}$	$t\cos\alpha t$	$\frac{p^2-\alpha^2}{\left(p^2+\alpha^2\right)^2}$
$(1+\alpha t)e^{\alpha t}$	$\frac{p}{(p-\alpha)^2}$	$\frac{e^{\alpha t}-e^{\beta t}}{\alpha-\beta}$	$\frac{1}{(p-\alpha)(p-\beta)}$
$\sin\alpha t$	$\frac{\alpha}{p^2+\alpha^2}$		

Теперь найдем выходной сигнал в виде функции времени:

$$x(t) = A + Be^{2t} + ce^{3t} = \frac{1}{6} - \frac{1}{2}e^{2t} + \frac{1}{3}e^{3t}.$$

Проверим, выполняются ли для него начальные условия $x(0) = x'(0) = 0$ (так как по определению передаточная функция вычисляется для нулевых начальных условий). Найдем

$$x'(t) = -e^{2t} + e^{3t};\ x'(0) = 0.$$

Приведем основные свойства преобразования Лапласа [20]. Пусть $f_k(t)$, $k = 1, \ldots, n$ – конечный набор исходных функций (называемых функциями-оригиналами); L обозначает преобразование Лапласа, а c_k – числовые константы. Тогда

$$L\left[\sum_{k=1}^{n} c_k f_k(t)\right] = \sum_{k=1}^{n} c_k F_k(p),$$

где

$$L[f_k(t)] = F_k(p);$$

$$L[f'(t)] = pF(p) - f(0);$$

$$L\left[\int_0^t f(x)dx\right] = \frac{1}{p}F(p);$$

$$L\left[\int_0^t f(t-\tau)g(\tau)d\tau\right] = F(p)G(p);$$

$$L[tf(t)] = -\frac{d}{dp}F(p);$$

$$L\left[\frac{1}{t} f(t)\right] = \int_p^\infty F(s)ds;$$

$$L\left[t^n f(t)\right] = (-1)^n \frac{d^n}{(dp)^n} F(p);$$

$$\lim_{p\to 0}\left[pF(p)\right] = \lim_{t\to\infty}\left[f(t)\right].$$

Лекция 21
ЦИФРОВЫЕ ФИЛЬТРЫ И *Z*-ПРЕОБРАЗОВАНИЕ

Цель. Рассмотреть основы дискретных фильтров и *Z*-преобразования как базового способа их расчета.

Дискретным аналогом формулы (20.14) является следующая формула

$$X_i(n) = \sum_{k=0}^{i} h_k f(i-k), \qquad (21.1)$$

где $f(j)$ – значение входного сигнала на такте с номером j.

Иначе говоря, теперь рассматривают не непрерывные входные сигналы, а дискретные последовательности: $f(0)$, $f(1)$, $f(2)$, …, $f(N)$. Теперь уже преобразование Лапласа не помогает, поскольку оно предназначено для непрерывных сигналов. Для дискретных сигналов используют *Z*-преобразование. Сущность *Z*-преобразования передается формулой

$$F(z) = \sum_{n=0}^{\infty} f(n) z^{-n}. \qquad (21.2)$$

Пусть, например, $f(n) = 1$, тогда $F(z) = \sum_{n=0}^{\infty} \left(\frac{1}{z}\right)^n = \frac{1}{1-\frac{1}{z}} = \frac{z}{z-1}$.

Пусть далее $f(n) = n$. Тогда $F(z) = \sum_{n=0}^{\infty} n \left(\frac{1}{z}\right)^n$.

Примем для ясности $w = \frac{1}{z}$. Тогда можно записать

$$\sum_{n=0}^{\infty} n w^n = \left(\sum_{n=0}^{\infty} w^n \right)' - (w + w^2 + w^3 + ...) - 1;$$

$$\left(\sum_{n=0}^{\infty} w^n \right)' = \sum_{n=0}^{\infty} (w^n)' = 1 + 2w + 3w^2 + 4w^3 + ...$$

Отсюда,

$$\sum_{n=0}^{\infty} nw^n = \left(\frac{1}{1-w}\right)' - \frac{w}{1-w} - 1 = \frac{1}{(1-w)^2} - \frac{w}{1-w} - 1 =$$

$$= \frac{1-w+w^2-1-w^2+2w}{(1-w)^2} = \frac{w}{(1-w)^2}.$$

Это дает искомое Z-преобразование

$$F(z) = \sum_{n=0}^{\infty} f(n) z^{-n} = \frac{\frac{1}{z}}{\left(1-\frac{1}{z}\right)^2} = \frac{z}{(1-z)^2}.$$

Другие важные результаты Z-преобразования содержит табл. 21.1.

Таблица 21.1

$f(n)$	$F(z)$	$f(n)$	$F(z)$
$(-1)^n$	$\frac{z}{z+1}$	$\frac{a^n}{n!}$	$e^{\frac{a}{z}}$
n^2	$\frac{z(z+1)}{(z-1)^3}$	$(n+1)a^n$	$\frac{z^2}{(z-a)^2}$
na^{n-1}	$\frac{z}{(z-a)^2}$	$\sin(n\beta)$	$\frac{z\sin\beta}{z^2-2z\cos\beta+1}$
$C_n^m a^m$	$\frac{a^m z}{(z-a)^{m+1}}$	$\cos(n\beta)$	$\frac{z(z-\cos\beta)}{z^2-2z\cos\beta+1}$
a^n	$\frac{z}{z-a}$		

Найдем далее Z-преобразование от функций $f(n+a)$, $f(n-a)$. Имеем для $f(n+a)$:

$$Z(n+a) = \sum_{n=0}^{\infty} f(n+a)\frac{1}{z^n}.$$

Для удобства опять обозначим $w = \frac{1}{z}$, тогда

$$\sum_{n=0}^{\infty} f(n+a)w^n = \frac{1}{w^a}\sum_{n=0}^{\infty} f(n+a)w^{n+a} =$$

$$= \frac{1}{w^a}\left[f(a)w^a + f(a+1)w^{a+1} + ... + f(r)w^r + ... \right] =$$

$$= \frac{1}{w^a}\left[F(w) - f(0)w^0 - f(1)w - ... - f(a-1)w^{a-1} \right].$$

Окончательно получаем

$$f(n+a) \to z^a \left[F(z) - f(0) - \frac{f(1)}{z} - ... - \frac{f(a-1)}{z^{a-1}} \right].$$

Аналогичным образом найдем, что

$$f(n-a) \to z^{-a}F(z), \text{ где } f(n-a) = 0 \text{ при } n < a.$$

Теперь нас будет интересовать *Z*-преобразование от формулы (4.29). Установлено, что

$$Z\left(\sum_{n=0}^{\infty} g(k) f(N-k) \right) = h(z) f(z). \qquad (21.3)$$

Приведенные сведения позволяют провести расчет цифрового фильтра. Прежде всего заметим, что аналогом уравнения (20.10) в цифровом фильтре является уравнение в конечных разностях. Итак, в общем случае имеем следующее уравнение цифрового фильтра:

$$a_n x(n+k) + a_{n-1}x(n+k-1) + ... + a_0 x(k) =$$
$$= b_m f(m+k) + b_{m-1} f(m+k-1) + ... + b_0 f(k) \qquad (21.4)$$

с начальными условиями $x(0) = x_0$, $x(1) = x_1$, …, $x(n-1) = x_{n-1}$,

где $x(n + k)$ – дискретное значение сигнала на выходе на такте $(n + k)$; $f(m + k)$ – дискретное значение сигнала на входе на такте $(m + k)$.

Выполнив *Z*-преобразование в левой и правой части уравнения (21.4), получим

$$(a_n z^n + a_{n-1}z^{n-1} + ... + a_0)X(z) = (b_m z^m + b_m z^{m-1} + ... + b_0)F(z).$$

Это уравнение составлено для нулевых начальных условий $x(0) = 0$, $x(1) = 0$, …

Отсюда записываем

$$X(z) = F(z)W(z),$$

где $W(z)$ – передаточная функция для нулевых начальных условий.

Имеем

$$W(z) = \frac{b_m z^m + b_m z^{m-1} + \ldots + b_0}{a_n z^n + a_{n-1} z^{n-1} + \ldots + a_0}.$$

По выражению $X(z)$ можно найти оригинал $X(n)$, используя таблицу, либо обратное Z-преобразование.

***Пример*.** Рассмотрим уравнение цифрового фильтра:

$$x(k+1) - 4x(k) = 4k.$$

Начальные условия нулевые: $x(0) = 0$.

Применим Z-преобразование:

$$zX(z) - 4X(z) = \frac{4z}{(1-z)^2}.$$

Отсюда

$$X(z) = \frac{4z}{(1-z)^2(z-4)}.$$

Разложим это выражение на простые дроби:

$$\frac{4z}{(1-z)^2(z-4)} = z\left[\frac{A}{z-4} + \frac{B}{z-1} + \frac{C}{(z-1)^2}\right].$$

Получим: $A = \frac{4}{9}$, $B = -\frac{4}{9}$, $C = -\frac{4}{3}$.

Найдем $x(n)$, используя табл. 21.1:

$$x(n) = \frac{4}{9} \cdot 4^n - \frac{4}{9} - \frac{4}{3}n.$$

Остается выяснить, как получают разностные уравнения. Одним из способов является метод инвариантной передаточной функции. Довольно часто передаточная функция непрерывной системы имеет вид

$$H_1(t) = Ae^{\alpha t} \qquad (21.5)$$

либо

$$H_2 = At^n e^{\beta t}. \qquad (21.6)$$

Каждое из уравнений (21.5) или (21.6) может входить в представление передаточной функции. Дискретизацией (21.5) является

$$h_1(k) = Ae^{k\alpha T}, \tag{21.7}$$

а дискретизацией (21.6) является

$$h_2(k) = A(kt)^n e^{k\beta T}, \tag{21.8}$$

где T – период дискретизации [21].

Теперь можно найти Z-преобразование от (21.7) и (21.8).

$$H_1(z) = \sum_{n=0}^{\infty} Ae^{n\alpha T} z^{-n} = \frac{A}{1 - e^{\alpha T} z^{-1}}. \tag{21.9}$$

$$H_2(z) = A\frac{d^n}{d\beta^n}\left(\frac{1}{1 - e^{\beta T} z^{-1}}\right). \tag{21.10}$$

Зная дискретную передаточную функцию и сигнал на входе схемы, легко найти выходной сигнал.

Лекция 22
МАШИННАЯ АРИФМЕТИКА

Цель. Изучить основы выполнения операций машинной арифметики и принципы технической реализации арифметико-логического блока.

22.1. Операции над двоичными числами

Рассмотрим сначала операцию сложения. Ее выполняют над двоичными числами обычным образом (т. е. так же как и над десятичными). Например, пусть $A = 1010$ (10_{10}), $B = 0110$ (6_{10}). Найдем $C = A + B$:

$$\begin{array}{r} 1\,0\,1\,0 \\ 0\,1\,1\,0 \\ \hline 1\,0\,0\,0\,0 \\ \quad\;\; 1\;1 \end{array}$$

$C = 10000$ (16_{10})

При сложении двух единиц происходит перенос в старший разряд.

Операция вычитания заменяется сложением с использованием дополнительного кода вычитаемого числа. Дополнительный код числа получают заменой 1 на 0 (0 на 1) и добавлением в младший разряд 1.

Пример. Пусть $A = 1100$. Формируем его дополнительный код. Сначала инвертируем разряды числа (заменяем их на противоположные): $1100 \rightarrow 0011$, затем добавляем в младший разряд 1:

$$\begin{array}{r} 0\,0\,1\,1 \\ \qquad\;\; 1 \\ \hline 0\,1\,0\,0 \end{array}$$

Итак, дополнительным кодом числа $A = 1100$ является $A' = 0100$.

Пусть нам необходимо найти $C = A - B$, где $A = 1010$, $B = 0110$. Для этого нужно сложить A с дополнительным кодом B. Найдем дополнительный код числа B: 1010. Складываем число A с

дополнительным кодом числа *B*, при этом единицу переноса в пятый разряд игнорируем:

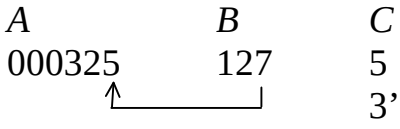
$$\begin{array}{r} 1\,0\,1\,0 \\ 1\,0\,1\,0 \\ \hline 1\,0\,1\,0\,0 \end{array}$$

В ответе мы получили 0100 (4_{10}). Таким образом, вычитание заменяется просто сложением с дополнительным кодом.

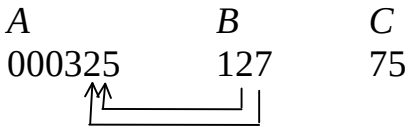
Перейдем к умножению чисел. Рассмотрим несколько модифицированный способ умножения десятичных чисел на следующем примере.

Пусть нужно найти $C = AB$, где $A = 325$ и $B = 127$.

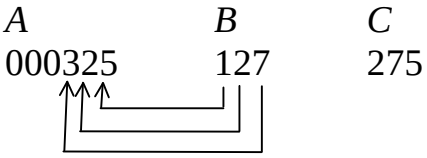
Расположим числа следующим образом:



К числу *A* присоединяем столько нулей слева, сколько цифр в числе *B*. Умножим последнюю цифру числа *B* на последнюю цифру числа *A*. Запишем 5 и 3 – в качестве переноса. Теперь свяжем стрелками разряды *A* и *B*:

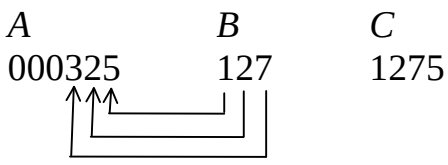


Найдем следующую цифру числа *C*: $2 \cdot 7 + 2 \cdot 5 + 3 = 27$. Этой цифрой является 7. Цифра 2 идет в качестве переноса. Следующее группирование:



Имеем: $3 \cdot 7 + 2 \cdot 2 + 5 \cdot 1 + 2 = 32$.

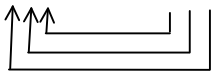
Продолжаем процесс группирования:



$$0 \cdot 7 + 2 \cdot 3 + 1 \cdot 2 + 3 = 11$$

И далее

A	*B*	*C*
000325	127	41275



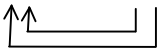
$$1 \cdot 3 + 1 = 4$$

Итак, мы нашли окончательный ответ: 41275.

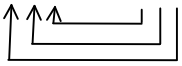
Эту технику легко применить к двоичным числам. Пусть $A = 110$ (6_{10}) и $B = 011$ (3_{10}). Проиллюстрируем процесс умножения.

A	*B*	*C*
000110	011	0

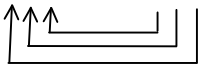
A	*B*	*C*
000110	011	10



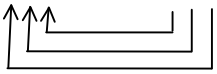
A	*B*	*C*
000110	011	010
		1’



A	*B*	*C*
000110	011	0010
		1’



A	*B*	*C*
000110	011	10010



В ответе мы получаем $C = 10010$ (18_{10}). Преимущество такой схемы умножения состоит в том, что ее легко реализовать на основе комбинационной схемы.

Таким образом, при реализации схемы умножения основная сложность связана с построением логики переносов.

Рассмотрим теперь деление (ограничимся случаем нахождения целого остатка).

Пусть $A = 100100$ (36_{10}) и $B = 100$ (4_{10}).

Присоединяем к B справа k нулей, выбрав k так, чтобы число разрядов в A и B совпало:

A = 100100

B' = 100000.

Если окажется, что $B' \le A$, то получаем, что в частном должны быть $(k + 1)$-я цифра, причем $z_k = 1$ при нумерации цифр с 0. Если $B' > A$, то в частном должно быть k цифр. У нас $k = 3$, $z_3 = 1$, поскольку A = 100100 > B' = 100000.

Находим $A' = A - B' = 100$. Поскольку $A' = B$, то $k = 0$ и $z_0 = 1$. Ясно, что остальные цифры z_2 и z_1 равны 0. Итоговый ответ: 9.

Очевидным образом такое деление соответствует представлению числа

$$A = 2^n a_n + 2^{n-1} a_{n-1} + \ldots + 2^0 a_0.$$

Следовательно, если частное есть $Z = 2^k z_k + 2^{k-1} z_{k-1} + \ldots + 2^0 z_0$, то

$$A = BZ = 2^k z_k B + 2^{k-1} z_{k-1} B + \ldots + 2^0 z_0 B.$$

Умножению B на 2^k соответствует приписывание справа k нулей. Таким образом, сначала сравниваем A и $2^k B$, затем находим $A - 2^k B$ и продолжаем рекурсивно «извлекать» цифры частного z_{k-1}, z_{k-2}, …, z_0.

22.2. Построение двоичного сумматора

Построим двоичный сумматор для сложения четырехразрядных чисел $a_3a_2a_1a_0$ и $b_3b_2b_1b_0$. Для того чтобы это осуществить, нужно построить таблицу истинности (см. таблицу).

Таблица

	a_i	b_i	s_{i-1}	p_i	s_i
	0	0	0	0	0
*	0	0	1	1	0
*	0	1	0	1	0
	0	1	1	0	1
*	1	0	0	1	0
	1	0	1	0	1
	1	1	0	0	1

*	1	1	1	1	1

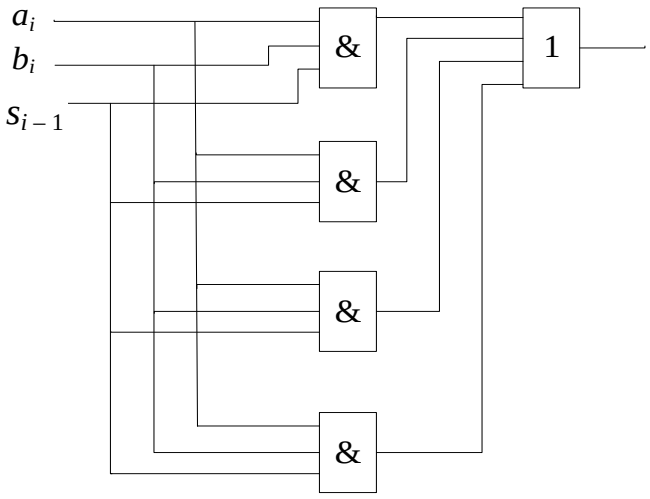
В этой таблице s_{i-1} означает сигнал переноса с $(i-1)$-го разряда; p_i – результирующее значение i-го разряда суммы; s_i – значение сигнала переноса в старший разряд.

В таблице входными данными являются a_i, b_i, s_{i-1}; выходными – p_i и s_i. Рассмотрим общую технику реализации, например, p_i. Для этого пометим знаком «*» в таблице те строки, где $p_i = 1$.

Рассмотрим первую такую строку. Видим, что $p_i = 1$, когда $a_i = 0$, $b_i = 0$, $s_{i-1} = 1$. Однако $p_i = 1$ также и для третьей, пятой и для восьмой строки. Объединяя все эти результаты, получим:

$$p_i = \overline{a}_i \overline{b}_i s_{i-1} \vee \overline{a}_i b_i \overline{s}_{i-1} \vee a_i \overline{b}_i \overline{s}_{i-1} \vee a_i b_i s_{i-1}.$$

Как только любое из слагаемых в правой части окажется равным 1, то и $p_i = 1$. Заметим, что мы пишем в слагаемом $\overline{a}_i$, если в соответствующей строке таблице $a_i = 0$, и a_i, если $a_i = 1$. Технически реализовать эту схему можно, как показано на рисунке.



Рисунок

Лекция 23
ОПЕРАЦИОННЫЕ УСИЛИТЕЛИ И АНАЛОГОВАЯ ОБРАБОТКА СИГНАЛОВ

Ц е л ь . Изложить теоретико-прикладные принципы построения и использования аналоговых операционных усилителей.

23.1. Базовые определения и обозначения

Операционный усилитель (ОУ) – это многокаскадный усилитель с дифференциальным входом (рис. 23.1).

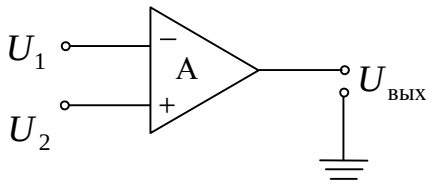


Рис. 23.1

Его особенностью являются высокое входное сопротивление, низкое выходное сопротивление и бесконечная ширина полосы пропускания. Основное соотношение ОУ следующее:

$$U_{вых} = (U_2 - U_1)A. \qquad (23.1)$$

Соотношение (23.1) просто указывает, что происходит усиление разности сигналов (вход «+» является прямым, а вход «–» – инверсным); A является коэффициентом усиления. С помощью операционного усилителя можно получать различные операционные схемы. Простейшей является схема повторителя напряжения (рис. 23.2).

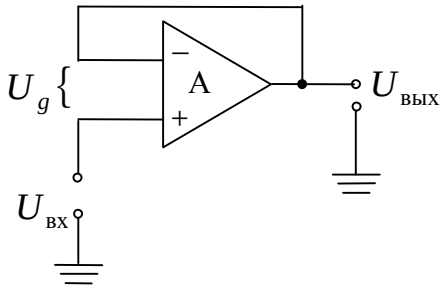


Рис. 23.2

Для схемы на рис. 23.2 заметим, что в силу основного уравнения (23.1)

$$U_{\text{вых}} = AU_g.$$

Далее привлечем закон Кирхгофа, согласно которому сумма падений напряжений вдоль любого замкнутого контура равна 0. Имеем

$$U_{\text{вх}} + U_g = U_{\text{вых}}.$$

Тогда

$$U_{\text{вх}} + \frac{U_{\text{вых}}}{A} = U_{\text{вых}}.$$

Поскольку значение A в ОУ достигает нескольких десятков тысяч, то $\frac{U_{\text{вых}}}{A} \to 0,$ и мы имеем

$$U_{\text{вх}} = U_{\text{вых}}.$$

Следовательно, схема на рис. 23.2 выполняет роль повторителя сигналов.

Рассмотрим теперь схему на рис. 23.3.

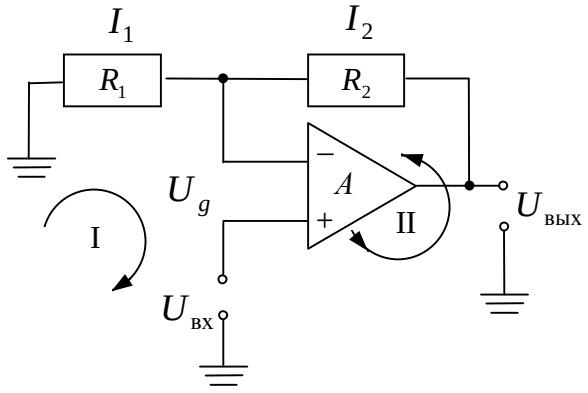


Рис. 23.3

Так, вдоль контура I имеем

$$I_1 R_1 = U_{\text{вх}} + U_g.$$

Вдоль контура II

$$U_{\text{вх}} + U_g + I_2 R_2 = U_{\text{вых}},$$

откуда $I_2 R_2 = U_{\text{вых}} - (U_{\text{вх}} + U_g)$.

Поскольку входное сопротивление ОУ очень высокое, то ток, протекающий через усилитель по контуру I, очень мал. Следовательно, имеем:

$$I_1 \approx I_2;$$

$$U_g = \frac{U_{\text{вых}}}{A} \approx 0.$$

Отсюда находим

$$\frac{U_{\text{вх}}}{R_1} \approx \frac{U_{\text{вых}} - U_{\text{вх}}}{R_2}. \qquad (23.2)$$

Таким образом, из (23.2) найдем

$$K = \frac{U_{\text{вых}}}{U_{\text{вх}}} = 1 + \frac{R_2}{R_1}. \qquad (23.3)$$

Величина K называется *коэффициентом обратной связи*. Эта величина полностью определяется соотношением $\frac{R_2}{R_1}$.

Наконец, рассмотрим схему двухвходового инверсного сумматора (рис. 23.4).

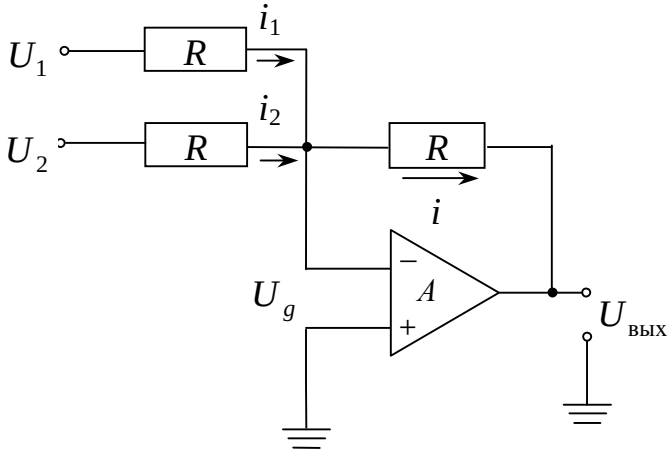


Рис. 23.4

Найдем:

$$i_1 = \frac{U_1}{R};$$

$$i_2 = \frac{U_2}{R};$$

$$i \approx -(i_1 + i_2);$$

$$U_{\text{вых}} \approx -(i_1 + i_2)R = -U_1 - U_2.$$

Разумеется, количество входов такого сумматора может быть каким угодно.

23.2. Интегрирование и дифференцирование на основе операционных усилителей

Операция интегрирования имеет следующий вид:

$$U_S = \int_{t_1}^{t_2} U_{\text{вх}}(t)dt.$$

Схему интегратора можно представить так, как показано на рис. 23.5.

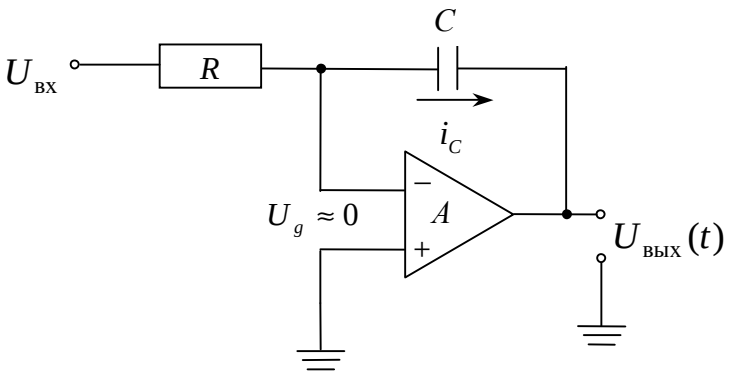


Рис. 23.5

Из теории известно, что

$$i_C = C\frac{dU_C}{dt}. \qquad (23.4)$$

Однако $U_C = -U_{\text{вых}}$, следовательно,

$$i_C = -C\frac{dU_{\text{вых}}}{dt}.$$

С другой стороны, $i_C \approx i_R$, поэтому

$$\frac{U_{\text{вх}}}{R} = -C\frac{dU_{\text{вых}}}{dt}.$$

Окончательно можно записать:

$$U_{\text{вх}}(t) = -RC\frac{dU_{\text{вых}}(t)}{dt}$$

и

$$U_{\text{вых}}(t) = -\frac{1}{RC}\int_0^t U_{\text{вх}}(t)dt. \qquad (23.5)$$

Таким образом, схема на рис. 23.5 выполняет роль интегрирующего элемента. Значение выходного напряжения определяет величину интеграла, вычисленного от $U_{\text{вх}}(t)$ на отрезке [0, t].

Схема дифференцирования приведена на рис. 23.6. Эта схема обеспечивает изменение напряжения на выходе по закону производной функции входного сигнала.

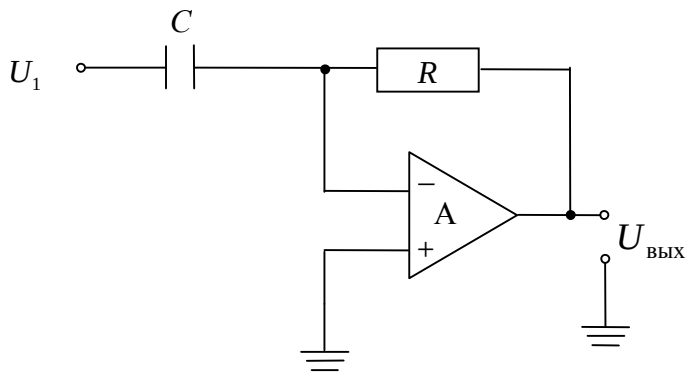


Рис. 23.6

Опять же

$$i_C = C\frac{dU_C}{dt};$$

$$i_C R \approx U_{\text{вых}};$$

$$U_1 = -U_C;$$

$$-RC\frac{dU_1}{dt} = U_{\text{вых}}.$$

Таким образом, напряжение на выходе изменяется по закону $\frac{dU_1}{dt}$.

Лекция 24
СТАТИСТИЧЕСКАЯ ОБРАБОТКА ДАННЫХ

Ц е л ь . Рассмотреть базовые задачи статистической обработки информации.

Прежде всего, мы начнем рассматривать вопрос об устранении сильно отклоняющихся и подозрительных данных. В первом случае у нас имеется некая выборочная совокупность, например данные о росте студентов. Поскольку эти данные вносились вручную, то не исключена возможность ошибки. Нужно выявить и отбросить все подозрительные данные.

Во втором случае совокупность данных образует временной ряд. Проблема состоит в том, что на эти данные может быть наложена помеха. Предварительная обработка данных в этом случае состоит в сглаживании временного ряда.

24.1. Удаление подозрительных и ошибочных значений из выборочной совокупности

Пусть выборочная совокупность состоит из значений

$$S = \{V_0, V_1, ..., V_z\},$$

где V_i – (например) рост i-го студента.

Обозначим через $V_{(n)}$ – максимальное среди чисел V_0, …, V_z и через $V_{(1)}$ – минимальное среди чисел V_0, …, V_z.

Будем рассматривать случайную величину

$$V = \frac{V_{(n)} - \overline{V}}{\sqrt{\frac{z-1}{z}}} \frac{1}{\Delta S_z} \qquad (24.1a)$$

или

$$V = \frac{\overline{V} - V_{(1)}}{\sqrt{\frac{z-1}{z}}} \frac{1}{\Delta S_z}, \qquad (24.1b)$$

где

$$\Delta S_z = \sqrt{\frac{1}{z-1}\sum_{i=0}^{z}(\Delta V_i)^2}, \qquad (24.2)$$

$$\Delta V_i = V_i - \overline{V}, \qquad (24.3)$$

$$\overline{V} = \frac{1}{z}\sum_{i=0}^{z} V_i. \qquad (24.4)$$

В табл. 24.1 приведены значения $V_{\max}$ – максимально возможные значения $V_{(n)}$, возникающие вследствие статистического разброса для соответствующего уровня надежности α (n – число измерений).

Таблица 24.1

n	$\alpha = 0,95$	$\alpha = 0,99$	n	$\alpha = 0,95$	$\alpha = 0,99$	n	$\alpha = 0,95$	$\alpha = 0,99$
3	1,41	1,41	19	2,69	2,93	35	2,85	3,22
4	1,69	1,72	20	2,62	2,96	36	2,86	3,24
5	1,87	1,96	21	2,64	2,98	37	2,87	3,25
6	2,00	2,13	22	2,66	3,01	38	2,88	3,26
7	2,09	2,26	23	2,68	3,03	39	2,89	3,27
8	2,17	2,37	24	2,70	3,05	40	2,90	3,28
9	2,24	2,46	25	2,72	3,07	41	2,91	3,29
10	2,29	2,54	26	2,73	3,09	42	2,92	3,30
11	2,34	2,61	27	2,75	3,11	43	2,93	3,31
12	2,39	2,66	28	2,76	3,12	44	2,94	3,32
13	2,43	2,71	29	2,78	3,14	45	2,95	3,33
14	2,46	2,76	30	2,79	3,16	46	2,96	3,34
15	2,48	2,80	31	2,80	3,17	47	2,96	3,35
16	2,52	2,84	32	2,82	3,18	48	2,97	3,35
17	2,55	2,87	33	2,83	3,20	49	2,98	3,36
18	2,58	2,90	34	2,84	3,21	50	2,98	3,37

Пример. Пусть получены оценки «годности» V_i по росту для шести студентов: 0,4; 0,5; 0,53; 0,7; 0,65; 0,6.

По формулам находим:

$$\overline{V} = \frac{0,4 + 0,5 + 0,53 + 0,7 + 0,65 + 0,6}{6} = 0,56;$$

$$\Delta S_z = \sqrt{\frac{1}{6-1}\left(0,16^2 + 0,06^2 + 0,03^2 + 0,14^2 + 0,09^2 + 0,04^2\right)} \approx 0,13;$$

$$V = \frac{0{,}7 - 0{,}56}{0{,}13\sqrt{\frac{5}{6}}} \approx \sqrt{1{,}2} \approx 1{,}1.$$

По таблице для уровня надежности $\alpha = 0{,}95$ и $n = 6$ найдем $V_{\max} = 2{,}00$.

Поскольку $V = 1{,}1 < 2{,}00$, нет оснований считать, что среди данных V_i максимальная оценка 0,7 ошибочна.

Таким же образом можно проверить на ошибочность и минимальную оценку, использовав формулу (24.1b).

24.2. Аномальные отклонения временного ряда

Временным рядом является последовательность значений измеряемой величины (или случайной величины) в моменты t_0, t_1, …, t_N, которые, как правило, отстают друг от друга на одно и то же значений Δt. Говорить, что некоторые значения такого ряда сильно отклоняются от ожидаемых значений можно лишь применительно к этим самым ожидаемым значениям.

Далее будем говорить, что упорядоченная по временным отсчетам последовательность ожидаемых значений изменяемой величины называется *временным трендом*. Итак, будем считать, что заранее известен общий вид линии тренда. Например, в простейшем случае таковой линией будет прямая

$$Y(t) = a_1 t + a_0. \qquad (24.6)$$

Этим случаем мы и ограничим наше рассмотрение. Пусть данные измерений представлены табл. 24.2.

Таблица 24.2

t_i	$Y(t_i)$
0	10
1	14
2	21
3	26
4	32
5	39
6	48
7	62

Прежде всего, нужно подобрать наилучшую эмпирическую линию тренда типа (24.6), т. е. найти по данным табл. 24.2 коэффициенты a_0, a_1. После того как мы получим такую линию тренда, для значений t_i из табл. 24.2 определим соответствующие значения $a_1 t_i + a_0$ и затем устраним линию тренда, перейдя от значений $Y(t_i)$ из табл. 24.2 к значениям, вычисленным по формуле

$$V_i = Y(t_i) - a_1 t_i - a_0. \qquad (24.7)$$

Теперь уже для множества значений V_i из (24.7) можно применить технику устранения подозрительных значений или выявления таковых, описанную в подразд. 24.1.

Нахождение коэффициентов a_0, a_1 по эмпирическим данным из табл. 24.2 является основной задачей регрессионного анализа. Техника такова. Составим уравнение

$$L = \sum_{i=1}^{n} \left(Y(t_i) - a_1 t_i - a_0\right)^2 \to \min, \qquad (24.8)$$

где L – невязка; n – число строк в табл. 24.2.

Ясно, что величина V_i характеризует отклонение реальных данных от линии тренда. Для отыскания коэффициентов a_0, a_1 составим систему уравнений

$$\begin{cases} \dfrac{\partial L}{\partial a_0} = 0, \\ \dfrac{\partial L}{\partial a_1} = 0. \end{cases} \qquad (24.9)$$

Система уравнений (24.9) определяет равенство нулю производных в точке минимума. Найдем

$$\frac{\partial L}{\partial a_0} = -2\sum_{i=1}^{n} \left[Y(t_i) - a_1 t_i - a_0\right] = 0,$$

отсюда

$$\sum_{i=1}^{n} Y(t_i) = a_1 \sum_{i=1}^{n} t_i + n a_0.$$

Далее,

$$\frac{\partial L}{\partial a_1} = -2\sum_{i=1}^{n} t_i \left[Y(t_i) - a_1 t_i - a_0 \right] = 0,$$

отсюда

$$\sum_{i=1}^{n} t_i Y(t_i) = a_1 \sum_{i=1}^{n} t_i^2 + a_0 \sum_{i=1}^{n} t_i.$$

Итак, мы получаем алгебраическую систему уравнений:

$$\begin{cases} \sum_{i=1}^{n} Y(t_i) = a_1 \sum_{i=1}^{n} t_i + n a_0, \\ \sum_{i=1}^{n} t_i Y(t_i) = a_1 \sum_{i=1}^{n} t_i^2 + a_0 \sum_{i=1}^{n} t_i. \end{cases} \qquad (24.10)$$

Решив систему (24.10) найдем коэффициенты a_0 и a_1. Дальнейшие действия определены нами выше.

Иногда возникает необходимость выяснить наличие тренда в эмпирических данных. Для этой цели следует использовать статистический критерий серий. Например, пусть дана таблица со следующими данными (табл. 24.3).

Таблица 24.3

t_i	Y_i
0	1
1	1,2
2	1,8
3	2,6
4	2,3
5	2,4
6	3,2
7	2,6
8	2,5
9	4,0
10	5,3
11	5,2
12	6,1

Априори будем полагать, что переход, если и существует, связан с увеличением Y_i при переходе к большему значению i. Сопоставим таблице 24.3 последовательность из знаков «+» и «–»

такую, что «+» пишем тогда, когда следующее наблюдение Y_i больше предыдущего, и пишем «–», если это не так. Имеем следующую последовательность серий:

$$S = + + + – + + – – + + + – +. \qquad (24.11)$$

Наличию тренда соответствует ситуация, когда чередование «+» и «–» не случайно (не равномерно). Серией называется последовательность из идущих подряд одних и тех же знаков. Есть серии из знаков «+» и серии из знаков «–».

Обозначим число серий через R. В нашем случае $R = 7$. Далее, пусть n_1 – число «+» и n_2 – число «–». У нас $n_1 = 8,\ n_2 = 4.$

Если тренда нет, то «+» и «–» появляются независимо друг от друга. Вводится в рассмотрение величина

$$Z = \frac{R - \left(\dfrac{2n_1 n_2}{n_1 + n_2} + 1 \right)}{\sqrt{\dfrac{2n_1 n_2 \left(2n_1 n_2 - n_1 - n_2\right)}{\left(n_1 + n_2\right)^2 \left(n_1 + n_2 - 1\right)}}}. \qquad (24.12)$$

Величина Z является случайной величиной, имеющей нормальное распределение со средним значением, равным 0, и дисперсию, равную 1.

Для нашего случая найдем:

$$Z = \frac{7 - \left(\dfrac{2 \cdot 8 \cdot 4}{8 + 4} + 1 \right)}{\sqrt{\dfrac{2 \cdot 8 \cdot 4 \cdot \left(2 \cdot 8 \cdot 4 - 8 - 4\right)}{12^2 \cdot 11}}} = 0{,}459.$$

Вероятность того, что случайная величина превзойдет 0,458, составляет 0,32. Это достаточно большое значение, так что мы вполне имеем основание считать, что в последовательности (24.7) «+» и «–» выпадают независимо и никакого тренда нет. Если мы сомневаемся в правильности этого результата, следует проверить, правильно ли мы сформировали последовательность (24.7).

Например, если бы последовательность имела такой вид:

$$+ + + + + + + – – + + + – +, \qquad (24.13)$$

то $R = 5,\ n_1 = 9,\ n_2 = 3$ и

$$Z = \frac{9 - \left(\frac{2 \cdot 9 \cdot 3}{9+3} + 1 \right)}{\sqrt{\frac{2 \cdot 9 \cdot 3 \cdot (2 \cdot 9 \cdot 3 - 9 - 3)}{12^2 \cdot 11}}} = \frac{\left[9 - (18+1) \right] \cdot 12 \cdot \sqrt{11}}{\sqrt{54 \cdot 42}} =$$

$$= -\frac{120\sqrt{11}}{47 \cdot 62} = -8,357.$$

Вероятность того, что случайная нормальная нормированная величина меньше –8,35 равна 0,0001. Следовательно, в (24.13) тренд есть, а ведь в (24.7) мы заменили только один минус на плюс. Поэтому если мы считаем, что динамика такова, что Y_i растет в целом с ростом i, то нужен более строгий критерий для формирования серии.

24.3. Оценка параметров распределения по результатам выборки

Важной задачей обработки данных является оценка параметров распределения по результатам выборки. Мы знаем, что случайные данные подчиняются, вообще говоря, разным законам распределения вероятности F. Одним из очень распространенных законов распределения является нормальный (гауссовский) закон. Этот закон характеризуется плотностью вероятности $f(x)$, определяемой формулой (1.14). Данная формула содержит два параметра (две константы), характеризующие распределение: математическое ожидание $M[\xi]$ и среднеквадратическое отклонение σ. Поэтому, чтобы задать нормальный закон распределения, нужно указать эти два параметра. Другие законы распределения вероятности могут содержать иное число параметров. Знание закона распределения очень важно, так как позволяет решить практически любую задачу, сформулированную как статистическая задача на множестве данных, подчиняющихся этому закону.

Задача оценки параметров распределения распадается на две подзадачи. Первая состоит в определении размера выборки; вторая – в получении самих оценок.

А. Оценка объема выборки

Сначала рассмотрим критерий А. Н. Колмогорова, который оценивает, насколько то или иное теоретическое распределение согласуется с эмпирическим распределением.

Определение. Эмпирическая функция распределения определяется следующим образом [12]:

$$F_э(x) = \begin{cases} 0, & x < x_{(1)}, \\ k/n, & x_{(k)} \le x \le x_{(k+1)}, \\ 1, & x > x_{(n)}, \end{cases} \qquad (24.14)$$

где $x_{(i)}$ – i-е значение вариационного ряда чисел $x_{(1)}$, $x_{(2)}$, …, $x_{(n)}$.

Пример. Пусть дано множество чисел выборки: 2,1; –0,6; 0,2; 3; –1; 1,3.

Вариационный ряд строится как упорядоченная по неубыванию последовательность:

$$-1;\ -0{,}6;\ 0{,}2;\ 1{,}3;\ 2{,}1;\ 3. \qquad (24.15)$$

График эмпирической функции распределения показан на рис. 24.1.

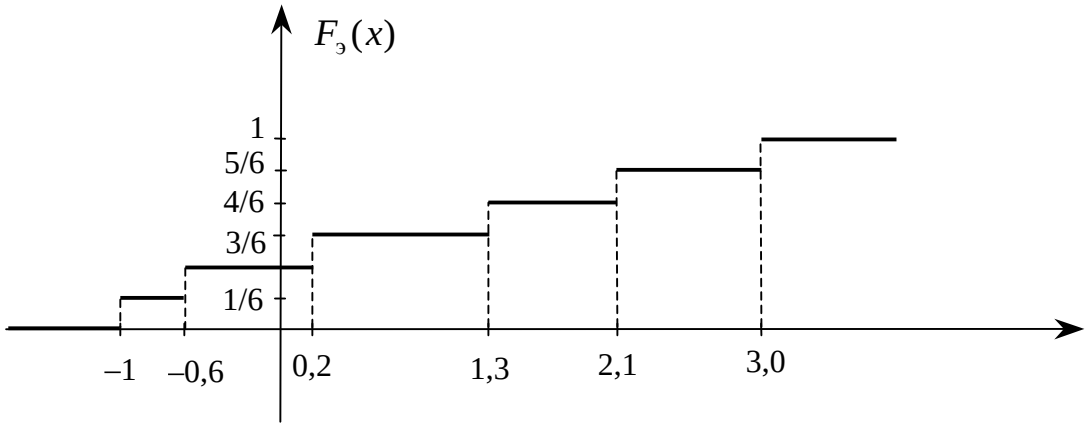


Рис. 24.1

Формула (24.14) соотносится с рис. 24.1 следующим образом. Возьмем, например, число 1,3 из вариационного ряда (24.15). Этому числу предшествуют слева три числа: –1; –0,6 и 0,2. Следовательно, эмпирическая вероятность того, что случайное число не превзойдет 1,3, составляет 3/6 (всех чисел $n = 6$). Этот

факт и отображен на рис. 24.1 в виде отрезка на уровне 3/6 в интервале от 0,2 до 1,3. Остальные отрезки эмпирической функции распределения получены из аналогичных соображений.

В критерии Колмогорова [12] определяется величина

$$D_\alpha = \sup_{x_i} \left| F_э(x_i) - F_т(x_i) \right|,$$

где $F_т(x_i)$ – значение теоретической функции распределения для числа x_i. Далее, рассматривается, критическая величина:

$$\begin{array}{cccc} C_\alpha = \dfrac{1{,}07}{\sqrt{n}}; & C_\alpha = \dfrac{1{,}22}{\sqrt{n}}; & C_\alpha = \dfrac{1{,}36}{\sqrt{n}}; & C_\alpha = \dfrac{1{,}36}{\sqrt{n}}; \\ \alpha = 0{,}2; & \alpha = 0{,}1; & \alpha = 0{,}05; & \alpha = 0{,}01, \end{array} \qquad (24.16)$$

где α – допустимый уровень ошибки; n – объем выборки.

Критерий Колмогорова: Если $D_\alpha(x) > C_\alpha$, то теоретическое распределение не соответствует эмпирическому.

Применим критерий Колмогорова к оценке объема выборки. Так, в [12] дается следующая оценка:

$$D_\alpha^2 \approx \frac{0{,}5 \ln \dfrac{\alpha}{2}}{n}, \qquad (24.17)$$

Из (24.17) получим

$$n \approx \frac{0{,}5 \ln \dfrac{\alpha}{2}}{D_\alpha^2}. \qquad (24.18)$$

В. Оценка параметров распределения

Мы рассмотрим только один метод оценки параметров – метод максимального правдоподобия.

Пусть оценивается единственный параметр θ и выборка представлена числами x_1, x_2, …, x_z. Составляем функцию

$$L(\theta) = p(x_1 \mid \theta) \cdot p(x_2 \mid \theta) \cdot \ldots \cdot p(x_z \mid \theta), \qquad (24.19)$$

где $p(x_i \mid \theta)$ – вероятность получения значения x_i при условии, что параметр есть θ.

Найдем логарифм левой и правой части (24.19):

$$\ln L(\theta) = \sum_{i=1}^{z} \ln p(x_i \mid \theta). \qquad (24.20)$$

Значение θ определяют из условия максимума ln$L$(θ) [7], т. е.

$$\frac{\partial \ln L(\theta)}{\partial \theta} = \sum_{i=1}^{z} \frac{\partial \ln p(x_i \mid \theta)}{\partial \theta} = 0. \qquad (24.21)$$

Приведем сравнительно простой пример. Пусть дана серия вида

$$+ - - - + - + - - - + - + - - + - - - - - \qquad (24.22)$$

Эта серия состоит из 21 опыта. В каждом опыте событие может наступить (+) или не наступить (–). Вероятность наступления события в одном опыте равно p. Итак, мы имеем здесь известный закон Бернулли. Для этого закона

$$P(n, m, p) = C_n^m \theta^m (1-\theta)^{n-m},$$

где $P(n, m, p)$ – вероятность того, что событие наступит в n опытах ровно m раз при условии, что вероятность наступления события в каждом опыте неизменна и равна p.

У нас n = 21, m = 6. Возьмем z = 1. Имеем

$$\frac{\partial \ln C_{21}^6 \cdot \theta^6 (1-\theta)^{15}}{\partial \theta} = 0;$$

$$\frac{\partial}{\partial \theta}\left[\ln C_{21}^6 + 6\ln\theta + 15\ln(1-\theta)\right] = \frac{6}{\theta} + (-1)\cdot\frac{15}{1-\theta} = 0.$$

Отсюда

$$6 - 6\theta = 15\theta;$$

$$6 = 21\theta;$$

$$\theta = \frac{6}{21} = \frac{2}{7}.$$

Нетрудно заметить, что в общем случае (24.21) для формулы Бернулли даст

$$\frac{\partial}{\partial \theta}\left[\ln C_n^m + m\ln\theta + (n-m)\ln(1-\theta)\right] = m\frac{1}{\theta} + (-1)\frac{n-m}{1-\theta} = 0.$$

Отсюда $\theta = \frac{m}{n}$.

Лекция 25
СЖАТИЕ ПО МЕТОДУ Д. ХАФФМАНА. АРИФМЕТИЧЕСКОЕ КОДИРОВАНИЕ

Цель. Рассмотреть и изучить метод сжатия информации Д. Хаффмана, а также альтернативные алгоритмы сжатия.

Метод сжатия Хаффмана является одним из лучших способов сжатия информации. Данный метод мы поясним на следующем примере. Пусть дана подлежащая сжатию последовательность из «0» и «1»:

$$010000101000101111001011. \qquad (25.1)$$

Разобьем эту последовательность на тройки разрядов и для каждой тройки подсчитаем, сколько раз она встречается в последовательности:

010’000’101’000’101’111’001’011

010 – 1	000 – 2
000 – 2	101 – 2
101 – 2	010 – 1
111 – 1	111 – 1
001 – 1	001 – 1
011 – 1	011 – 1

Упорядочим комбинации по частоте встречаемости (второй столбец). Теперь «объединим» две последние комбинации во втором столбце в одну и все комбинации снова переупорядочим по убыванию частоты встречаемости:

‘001-011’ – 2
000 – 2
101 – 2
010 – 1
111 – 1

Снова объединим две последние комбинации в одну и проведем очередное упорядочение:

‘010-111’ – 2
‘001-011’ – 2

000 – 2
101 – 2

Дальнейший процесс объединения комбинаций выполним по аналогии:

‘000-101’ – 2
‘010-111’ – 2
‘001-011’ – 2

‘010-111-001-011’ – 4
‘000-101’ – 4

Теперь нарисуем дерево, схематически передающее реализованный нами способ объединения комбинаций (рис. 25.1).

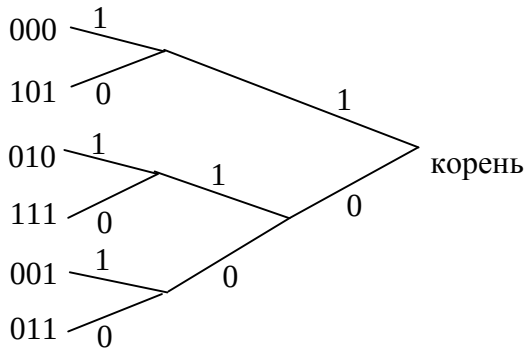


Рис. 25.1

Для полученного дерева выполним движение с корневой вершины к исходным тройкам. При выходе из любого узла помечаем одно из двух ребер «1», а другое – «0» (рис. 25.1). Итак, запомним, что разметка выполняется при движении справа налево по построенному дереву. Теперь новый код для любой из исходных троек получаем как комбинацию, сопоставляемую с путем из корня в данную вершину. Получаем следующее соответствие исходных троек и новых комбинаций:

000 – 11
101 – 10
010 – 001
111 – 010
001 – 001
011 – 000

С учетом нового способа кодировки исходная последовательность (25.1) может быть переписана таким образом:

$$01111101110010001000. \qquad (25.2)$$

Длина последовательности (25.2) сократилась в сравнении с длиной (25.1) с 24 до 20 бит. Основной принцип кодирования Хаффмана состоит в том, что часто встречаемые комбинации кодируются более короткими последовательностями. Таким образом, общая длина последовательности оценивается как

$$L = \sum_i s_i P_i, \qquad (25.3)$$

где s_i – число бит в i-й комбинации; P_i – относительная частота встречаемости i-й комбинации.

Было замечено, хотя это и не обязательно верно во всех случаях, что длина Хаффмана комбинации s_i, как правило, не превосходит величину – $\log_2 P_i$. Так, в нашем примере относительная частота комбинации 000 составляет $\frac{2}{8} = \frac{1}{4}$. Ее код Хаффмана – 11 (длина этого кода равна 2). Имеем

$$2 \le -\log_2 \frac{1}{4} = 2 \text{ (что справедливо).}$$

Таким образом, при кодировании Хаффмана результирующую длину кода можно ориентировочно записать в виде

$$L \approx -\sum_{i=1}^{\mu} \log_2 P_i. \qquad (25.4)$$

Кодирование Хаффмана можно применять повторно.

Арифметическое кодирование. Пусть у нас имеется последовательность

$$a_1 a_2 a_2 a_3. \qquad (25.5)$$

Относительная частота символов:

$$P_1 = \frac{1}{4},\ P_2 = \frac{1}{2},\ P_3 = \frac{1}{4}.$$

При арифметическом кодировании последовательность (25.5) заменяется одним единственным числом. Для получения этого числа разобьем интервал [0, 1] на подинтервалы:

$$[0; 0{,}25], (0{,}25; 0{,}75], (0{,}75; 1]. \qquad (25.6)$$

Заметим, что длина каждого подинтервала соответствует относительной частоте соответствующего символа. Нетрудно сообразить, что первый подинтервал [0; 0,25] соответствует символу a_1; подинтервал (0,25; 0,75] – a_2 и (0,75; 1] – a_3. В последовательности (25.6) первый символ – a_1. Ему соответствует интервал [0; 0,25]. Поэтому выбираем этот подинтервал и делим его опять согласно относительным частотам символов:

$$[0; 0{,}0625), [0{,}0625; 0{,}1875), [0{,}1875; 0{,}25]. \qquad (25.7)$$

Следующий символ – a_2. Ему соответствует интервал [0,0625; 0,1875). Делим его на подинтервалы:

$$[0{,}0625; 0{,}09375), [0{,}09375; 0{,}15625), [0{,}15625; 0{,}1875). \qquad (25.8)$$

Следующий символ – a_2. Выбираем второй подинтервал [0,09375; 0,15625). Делим его на три подинтервала соответственно частотам символов:

$$[0{,}09375; 0{,}109375), [0{,}109375; 0{,}140625), [0{,}140625; 0{,}15625). \qquad (25.9)$$

Наконец, последний символ – a_3. Ему соответствует третий интервал. Поэтому в качестве окончательного кода для последовательности (25.5) можно указать любое число из интервала [0,140625; 0,15625). Например, возьмем 0,15. Итак, последовательность (25.5) кодируется числом 0,15.

Чтобы восстановить исходную последовательность, нужно действовать таким образом. Согласно частотам символов составляем исходное разбиение (25.6). Видим, что наш код 0,15 попадает
в первый подинтервал [0; 0,25]. Значит первый символ – a_1. Далее разбиваем интервал [0; 0,25] на три подинтервала (25.7) и смотрим, к какому из них принадлежит наш код 0,15. Теперь это второй подинтервал, поэтому следующий символ a_2. Далее из представления (25.8) снова выбираем второй подинтервал и символ a_2. Наконец, из (25.9) выбираем символ a_3.

Арифметическое сжатие может давать большие последовательности цифр и поэтому его применение ограничивается небольшими последовательностями символов.

Лекция 26
СЖАТИЕ ГРАФИЧЕСКИХ ФАЙЛОВ

Цель. Изучить основы JPEG-алгоритмов для сжатия графической информации. Дать определение дискретного косинус-преобразования.

Сжатие графической информации основано на частичной потере информации. В самом деле, в изображении соседние пикселы (точки) мало различаются по яркости (светимости) и цвету. Особенностью является то обстоятельство, что глаз человека меньше различает именно светимость двух соседних точек. Поэтому модель YCbCr в большей степени ориентирована на сжатие, чем модель RGB. Для получения сжатого изображения применяют ортогональные преобразования данных. Ортогональные преобразования выполняют таким образом, чтобы большая часть данных при преобразовании получила маленькие (близкие к нулю значения)
и лишь небольшая часть данных оказалась значимой. Затем выполняется квантование (округление данных) так, что малозначимые данные становятся равными 0. В результате этих операций в итоговой, преобразованной матрице будет большое число нулевых элементов. Такая матрица эффективно сжимается с помощью алгоритма Хаффмана. Поясним сказанное на примере. Пусть исходные данные представлены следующей матрицей:

$$D = \begin{pmatrix} 3 & 2 & 5 & 1 \\ 4 & 4 & 3 & 6 \\ 2 & 0 & 5 & 2 \\ 0 & 1 & 0 & 3 \end{pmatrix}.$$

Возьмем матрицу преобразования:

$$W = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix}. \qquad (26.1)$$

Сначала найдем по правилам матричной алгебры произведение

$$C = \frac{1}{2} WD,$$

затем

$$C' = \frac{1}{2} CW. \qquad (26.2)$$

Получим

$$C' = \begin{pmatrix} 10{,}25 & -2{,}25 & 0{,}25 & 0{,}75 \\ 3{,}75 & 1{,}25 & -0{,}25 & 0{,}25 \\ -2{,}75 & 0{,}75 & -0{,}75 & -0{,}25 \\ -0{,}25 & -0{,}75 & -2.25 & 4{,}25 \end{pmatrix}.$$

В этой матрице доминирует лишь небольшое число элементов. Можно выполнить квантование, например, следующим образом:

$$C' = \begin{pmatrix} 10 & -2 & 0 & 0 \\ 4 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 \\ 0 & 0 & -2 & 4 \end{pmatrix}.$$

Именно эта операция и приводит к потере данных, хотя эта потеря мало отражается на исходных данных. В самом деле, легко восстановить из (26.2) матрицу C:

$$2C'W^{-1} = C. \qquad (26.3)$$

и аналогично

$$2W^{-1}C = D. \qquad (26.4)$$

С помощью (26.3), (26.4) получим восстановленную приближенную матрицу исходных данных:

$$D \approx \begin{pmatrix} 2{,}75 & 1{,}75 & 4{,}75 & 1{,}75 \\ 3{,}25 & 4{,}25 & 3{,}25 & 6{,}25 \\ 2{,}25 & 1{,}25 & 4{,}25 & 1{,}25 \\ -0{,}25 & 0{,}75 & -0{,}25 & 2{,}75 \end{pmatrix}.$$

После квантования получим

$$D \approx \begin{pmatrix} 3 & 2 & 5 & 2 \\ 3 & 4 & 3 & 6 \\ 2 & 1 & 4 & 1 \\ 0 & 1 & 0 & 3 \end{pmatrix}.$$

Эта последняя матрица очень близка к исходной матрице *D* (!).

Прежде всего, заметим, что матрица преобразования *W* должна строиться специальным образом. Во-первых, векторное произведение любых ее двух строк или столбцов равно 0 (а это означает, что строки и столбцы матрицы независимы и, следовательно, определитель матрицы отличен от 0). Во-вторых, матрица *W* подбирается так, что сумма элементов только в первой строке
и в первом столбце максимальны; в остальных строках и столбцах суммы элементов равны или близки к 0.

Из соображений, близких к рассмотренным, строится матрица дискретного косинусного преобразования (DCT-матрица), используемая в алгоритме JPEG. Матрица двумерного DCT-преобразования определяется из следующей формулы:

$$G_{ij} = \frac{1}{\sqrt{2n}} C_i C_j \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} p_{xy} \cos\left[\frac{(2y+1)\, j\pi}{2n}\right] \cos\left[\frac{(2x+1) i\pi}{2n}\right], \quad (26.5)$$

где p_{xy} – значение пиксела в строке *x* и столбце *y* квадратной матрицы пикселов размером $n \times n$;

$$C_i = \begin{cases} \dfrac{1}{2}, & \text{если } i = 0; \\ 1, & \text{если } i = 1, \ldots, n-1. \end{cases}$$

Матрица одномерного DCT-преобразования использует расчетную формулу

$$G_f = \frac{1}{2} C_f \sum_{t=0}^{n-1} p_t \cos\left[\frac{(2t+1)\, f\pi}{2n}\right]. \quad (26.6)$$

Заметим, что величины

$$\cos\left[\frac{(2t+1)f\pi}{2n}\right]$$

изменяются для $f = 0, 1, 2, ..., n-1$ и $t = 0, 1, 2, ..., n-1$ так, что в результате из них можно построить следующую матрицу преобразования для $n = 8$ (см. таблицу).

Таблица

θ	$\frac{\pi}{16}$	$\frac{3\pi}{16}$	$\frac{5\pi}{16}$	$\frac{7\pi}{16}$	$\frac{9\pi}{16}$	$\frac{11\pi}{16}$	$\frac{13\pi}{16}$	$\frac{15\pi}{16}$
cos(0θ)	1	1	1	1	1	1	1	1
cos(θ)	0,981	0,831	0,556	0,195	–0,195	–0,556	–0,831	–0,981
cos(2θ)	0,924	0,383	–0,383	–0,924	–0,924	–0,383	0,383	0,924
cos(3θ)	0,831	–0,195	–0,981	–0,556	0,556	0,981	0,195	–0,831
cos(4θ)	0,707	–0,707	–0,707	0,707	0,707	–0,707	–0,707	0,707
cos(5θ)	0,556	–0,981	0,195	0,831	–0,831	–0,195	0,981	–0,556
cos(6θ)	0,383	–0,924	0,924	–0,383	–0,383	0,924	–0,924	0,383
cos(7θ)	0,195	–0,556	0,831	–0,981	0,981	–0,831	0,556	–0,195

Эта матрица является ортогональной и построена по тем же принципам, что и матрица *W*, которую мы рассмотрели выше. Нам остается коротко охарактеризовать алгоритм сжатия JPEG, основу которого составляет DCT-преобразование.

В JPEG используется цветовая модель YCrCb, в которой Y передает светимость пиксела. Преобразование DCT выполняется отдельно к светимости Y и отдельно к матрице, кодирующей хроматические числа Cr и Cb. К светимости Y применяется одномерное DCT-преобразование. Для компоненты <Cr, Cb> выполняется разбиение изображения на матрицы пикселов 8 × 8. К каждой из таких матриц применяется двумерное DCT-преобразование. Таким образом, выполняется сжатие с потерей информации.

Сокращение JPEG происходит от слов Joint Photographic Expert Group – «Совместная группа по фотографии». Проект JPEG стал стандартом в 1991 г. – принят международной организацией стандартов ISO.

Лекция 27
ОСНОВЫ ЗАЩИТЫ ИНФОРМАЦИИ

Цель. Рассмотреть основные подходы к проблеме защиты информации. Изучить современные алгоритмы защиты.

Современные способы защиты информации используют в первую очередь различные методы шифрования. Мы рассмотрим здесь два криптографических метода: RSA и DES. Основные принципы криптографии можно сформулировать следующим образом.

1. В шифровании основную роль играет не алгоритм, а ключ.
2. Алгоритм шифрования должен быть таким, чтобы шифрование выполнялось легко и эффективно с вычислительной точки зрения; наоборот, дешифрование должно представлять собой сложнейшую математическую задачу (например, переборного типа).

27.1. Алгоритм RSA

Пусть необходимо передать по линии связи число x (рассмотрим здесь только целые положительные числа). Вместо числа x передают число y, вычисляемое по формуле

$$y = x^e \bmod m, \qquad (27.1)$$

где e и m – открытые числа (известны всем абонентам сети).

Требуем, чтобы e и m были взаимно простыми числами (т. е. не имели общих делителей, кроме 1, причем $m > e$).

Оказывается, зная y, e и m, найти x – сложнейшая математическая задача. Пока же продемонстрируем, как найти y по x, e, m.

Операция

$$a = b \bmod m \qquad (27.2)$$

дает целочисленный остаток a от деления b на m. Например,

$$2 = 17 \bmod 5$$

или

$$1 = 41 \bmod 8.$$

Но пусть требуется найти

$$630 \bmod 18 = ?$$

Это сделать посложнее. Но можно записать

$$630 = 2 \cdot 315 = 2 \cdot 5 \cdot 63 = 2 \cdot 5 \cdot 7 \cdot 9 = 63 \cdot 10.$$

Теперь можно использовать правило разложения на множители

$$a \cdot b \bmod m = [(a \bmod m)(b \bmod m)] \bmod m.$$

В самом деле, пусть

$$c = c_1 + r_1 m;$$

$$a = a_1 + r_2 m;$$

$$b = b_1 + b_3 m.$$

тогда

$$c_1 + r_1 m = (a_1 + r_2 m)(b_1 + r_3 m) = a_1 b_1 + r_2 m b_1 + r_3 m a_1 + r_2 r_3 m^2.$$

Последняя сумма дает остаток от деления на m, равный $a_1 b_1 \bmod m$. Однако $a_1 = a \bmod m$, $b_1 = b \bmod m$. Поэтому $63 \cdot 10 \bmod 18 = 9 \cdot 10 \bmod 18 = 0$. Теперь нетрудно это правило применить, скажем, к

$$7^{13} \bmod 8 = ?$$

Запишем $7^{13} = 7^2 \cdot 7^2 \cdot 7^2 \cdot 7^2 \cdot 7^2 \cdot 7^2 \cdot 7$. Имеем $7^2 \bmod 8 = 1$. Поэтому $7^{13} \bmod 8 = 1^6 \cdot 7 \bmod 8 = 7$.

Обратимся теперь к формуле (6.16).

Пусть $x = 4$, $e = 11$, $m = 14$.

Найдем

$$4^{11} \bmod 14 = 16^5 \cdot 4 \bmod 14 = 4 \cdot 2^5 \bmod 14 = 4 \cdot 32 \bmod 14 = 16 \bmod 14 = 2.$$

Итак, $y = 2$. Это значение и будет передано по сети вместо x.

Теперь рассмотрим, как восстановить x по y, m, e. Для этой цели нужно найти число d, удовлетворяющее условию

$$de = 1 \bmod \varphi(m), \qquad (27.3)$$

где $\varphi(m)$ – значение функции Эйлера от числа m.

Функция Эйлера вычисляется следующим образом

$$\varphi(ab) = \varphi(a)\varphi(b). \qquad (27.4)$$

Если p – простое число и r – целое, то

$$\varphi(p^r) = p^{r-1}(p-1). \qquad (27.5)$$

Формул (27.4) и (27.5) достаточно для того, чтобы найти функцию Эйлера для любого целого положительного числа. Однако для больших значений m вычисление функции Эйлера $\varphi(m)$ становится крайне сложной задачей. В нашем случае получаем:

$$\varphi(14) = \varphi(2 \cdot 7) = \varphi(2)\varphi(7) = 2^0(2-1)7^0(7-1) = 6.$$

Отметим, что значение $\varphi(m)$ равно числу целых чисел на отрезке 1..m, взаимно простых с m.

Пример. $\varphi(5) = 5^0(5-1) = 4$. Все четыре числа: 1, 2, 3, 4 взаимно просты с m.

Теперь обратимся к уравнению (27.3). В этом уравнении d играет роль секретного ключа. Решить уравнение (27.3) путем перебора значений d можно, но если в числе m, например, 100 цифр, то на вычисление d уйдет значительное время работы ПЭВМ типа Pentium II. Для небольших значений, таких как в нашем примере, можно воспользоваться алгоритмом решения уравнений в целых числах, рассмотренным ниже.

Итак, в нашем примере уравнение имеет следующий вид:

$$11d = 1 \bmod 6. \qquad (27.6)$$

Уравнение (27.6) можно переписать уже известным нам образом:

$$11d = 1 + 6r. \qquad (27.7)$$

где r и d – неизвестные целые числа.

Представим (27.7) в виде системы двух линейных неравенств:

$$11d - 6r \geq 1,$$
$$11d - 6r \leq 1,$$

или, что эквивалентно,

$$\begin{aligned} 11d - 6r &\geq 1, \\ -11d + 6r &\geq 1. \end{aligned} \qquad (27.8)$$

В неравенстве с положительной правой частью выделим член с минимальным по модулю коэффициентом и разрешим нера-

венство относительно этого члена, как мы делали это при решении систем линейных неравенств на основе стратегии устранения невязок (СУН):

$$-6r \geq 1 - 11d,$$

$$-r \geq \frac{1}{6} - 1\frac{5}{6}d.$$

Отсюда легко получить отсекающее неравенство:

$$-r \geq \frac{1}{6} - 2d + z; \tag{27.9a}$$

$$-r \geq 1 - 2d + z; \tag{27.9b}$$

$$r = -1 + 2d - z; \tag{27.9c}$$

где z – новая целочисленная переменная.

Заметим, что переход от (27.9a) к (27.9b) правомерен, так как r, d, z – целочисленны.

Выполним подстановку (27.9c) в систему (27.8). Получим новую систему:

$$\begin{gathered} -d + 6z \geq -5; \\ d - 6z \geq 5. \end{gathered} \tag{27.10}$$

Обратим внимание на следующее принципиальное обстоятельство. В сравнении с (27.8) значение минимального коэффициента понизилось. Этот факт можно строго обосновать. Следовательно, весь процесс должен закончиться рано или поздно одним из двух результатов:

1) минимальный коэффициент по модулю станет равным 1 (как в (27.10));

2) будет получена система вида

$$ax_i \geq b;$$

$$-ax_i \geq -b,$$

где a и b – взаимно просты (в этом случае нет решения в целых числах).

В первом случае процесс решения завершен. Получаем из (27.10) подстановку для d:

$$d = 5 + 6z. \tag{27.11}$$

Тогда из (27.9c) найдем:

$$r = 9 + 11z. \qquad (27.12)$$

Следовательно, формулы (27.11) и (27.12) дают нам итоговые подстановки для d и r из (27.7). Например, пусть $z = 0$. Тогда $d = 5$, $r = 9$. Возьмем именно это значение для минимального d.

Итак, мы подошли к решающему моменту: наш секретный ключ $d = 5$. Получили число $y = 2$, $m = 14$, $e = 11$. Восстанавливаем x по формуле

$$x = y^d \bmod m. \qquad (27.13)$$

Все сошлось. Возьмем, например, $z = 1$. Тогда $d = 11$, $r = 20$:

$$x = 2^{11} \bmod 14 = 2^5 \cdot 2^5 \cdot 2 \bmod 14 = 4 \cdot 4 \cdot 2 \bmod 14 = 4$$

(ответ тот же).

Таким образом, не имеет значения, какое z брать для получения d.

Метод RSA, вообще говоря, требует разложения на простые множители числа m, которое должно быть большим целым положительным числом. Для установления факта, что m – простое, можно использовать малую теорему Ферма:

$$a^{p-1} \bmod p = 1. \qquad (27.14)$$

В (27.14) a и p должны быть взаимно просты, а p – простое число.

Пример. $a = 6$, $p = 7$; $6^6 \bmod 7 = 36^3 \bmod 7 = 1$.

$a = 8$, $p = 11$:

$$8^{10} \bmod 11 = 64^5 \bmod 11 = 9^5 \bmod 11 = 81^2 \cdot 9 \bmod 11 = 16 \cdot 9 \bmod 11 =$$

$$= 16 \cdot 9 \bmod 11 = 45 \bmod 11 = 1.$$

К сожалению, формула (27.14) справедлива также для некоторых составных чисел. Поэтому чтобы применить ее на практике, нужно выполнить проверку для некоторых различных a. Вычисление функции Эйлера для произвольного большого целого положительного числа является очень сложной математической задачей. Поэтому заранее подбирают простые множители из существующих таблиц простых чисел, на основании которых и строят функцию Эйлера.

27.2. Электронная подпись

Принцип электронной подписи легко выводится из алгоритма RSA. Пусть нужно подписать текст T. Этот текст нужно «свернуть» в некое число y. Для свертки используют алгоритм хэширования. Теперь из уравнения (27.13) на основании секретного ключа получают подпись X. Число X и возвращается вместе с T и y в качестве подписи в документу T. Не зная секретного ключа d (заменяющего фамилию), нельзя найти X. Проверяющий легко проверит (27.1), чтобы удостовериться, что X и y соответствуют друг другу. Заметим, что кто-либо может перехватить документ и присвоить ему свою подпись, если ему известно преобразование (27.13). Поэтому принятый «подделанный» документ должен быть «застрахован».

Для этого используется число e – открытый ключ подписавшего документ. Теперь очень легко проверить соотношение:

$$y = x^e \bmod m.$$

Подобрать секретный ключ d к открытому ключу e практически невозможно.

27.3. Метод Диффи – Хеллмана

Метод Диффи и Хеллмана является двухключевым. Каждый абонент в сети имеет два ключа: один секретный – d_i, второй открытый, вычисляемый по формуле

$$y_i = \alpha^{d_i} \bmod p,$$

где p – большое простое число (взаимно простое с α); $\alpha < p$.

Число α выбирается так, чтобы числа α^1, α^2, …, α^{p-1} по модулю p давали некоторую перестановку чисел $\{1, 2, \ldots, p-1\}$. Число α называется первообразным корнем p. Все абоненты публикуют свои открытые ключи y_i. Допустим, абоненты A и B хотят передать друг другу информацию. Тогда первый из них, например A, берет открытый ключ y_B абонента B и вычисляет

$$(y_A)^{d_B} = (\alpha^{d_B d_\alpha}) \bmod p.$$

Таким образом, оба абонента находят одно и то же число, которое они и используют в качестве ключа для обмена сообщениями. При этом секретные ключи абонентов остаются известными только им одним.

27.4. Алгоритм DES

Алгоритм DES (Description Encryption System) состоит в последовательности чередующихся операций подстановки и перемешивания. Алгоритм DES использует секретный ключ *K*. Пусть, например, блок данных *B* = 10010011, а ключ *K* = 11001011.

Подстановка есть сложение по модулю 2:

$$B \oplus K = \begin{array}{r} 10010011 \\ \underline{11001011} \\ 01011000 \end{array}$$

Таким образом, операция подстановки непредсказуемым образом искажает исходные данные. Для восстановления исходных данных достаточно выполнить подстановку вторично:

$$B \oplus K = \begin{array}{r} 01011000 \\ \underline{11001011} \\ 10010011 \end{array}$$

Операция перемешивания заключается в обмене местами двух блоков в соответствии с заранее составленными таблицами. Длина обмениваемых блоков одинакова, но варьирует от одного прохода к другому. Операция подстановки выполняется не над целым ключом, а над случайно выбираемой его частью. Эта случайно выбираемая часть сама определяется значением ключа. Одним из способов получения псевдослучайных чисел является следующий:

$$a_n = k a_{n-1} + C (\mathrm{mod}\, M), \qquad (27.15)$$

где a_i – псевдослучайное число, порожденное на шаге *i*.

Первоначально полагают, например, $a_0 = 0$. В качестве константы *C* примем

$$C = K + \underbrace{9999\ldots9}_{n-1}, \qquad (27.16)$$

где *K* – секретный ключ.

Величина *M* как правило равна $10^n - 1$; величина *n* равна числу разрядов генерируемого псевдослучайного числа. Например, пусть *n* = 2; *K* = 31; a_0 = 0; *C* = 31 + 9 = 40; *M* = 99:

$$a_1 = (31 \cdot 0 + 40) \bmod 99 = 40;$$
$$a_2 = (31 \cdot 40 + 40) \bmod 99 = 93;$$
$$a_3 = (31 \cdot 93 + 40) \bmod 99 = 12;$$
$$a_4 = (31 \cdot 12 + 40) \bmod 99 = 75;$$
$$a_5 = (31 \cdot 75 + 40) \bmod 99 = 9 \text{ и т. д.}$$

Порожденные псевдослучайные числа могут определять номера разрядов, выбираемых из ключа в формируемую часть для выполнения операции подстановки.

Для дешифрации принятой комбинации нужно знать ключ *K* и выполнять операции подстановки и перемешивания в обратном порядке.

Часть 2. ИНФОРМАЦИОННЫЕ СИСТЕМЫ И СПЕЦИАЛЬНЫЕ ВОПРОСЫ

Лекция 28
ТИПОВАЯ АРХИТЕКТУРА АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Цель. Рассмотреть архитектуру и функциональное назначение типовых звеньев автоматизированных информационных систем.

Информационные системы – это программно-технические комплексы, которые характеризуются:

1) целью (целями) функционирования. Цели функционирования информационных систем могут быть самые разные. Это могут быть хранилища данных, системы цифровой обработки, сети передачи, системы принятия решений, мультимедийные системы и т. д.

2) основными процессами обработки информации. Если мы возьмем базу данных, то таковыми являются: хранение, поиск и сортировка информации. Если возьмем систему для работы с текстом, то таковыми будут представление текста, поиск и классификация текстовых документов, шифрование текста, интерпретация текста и др.

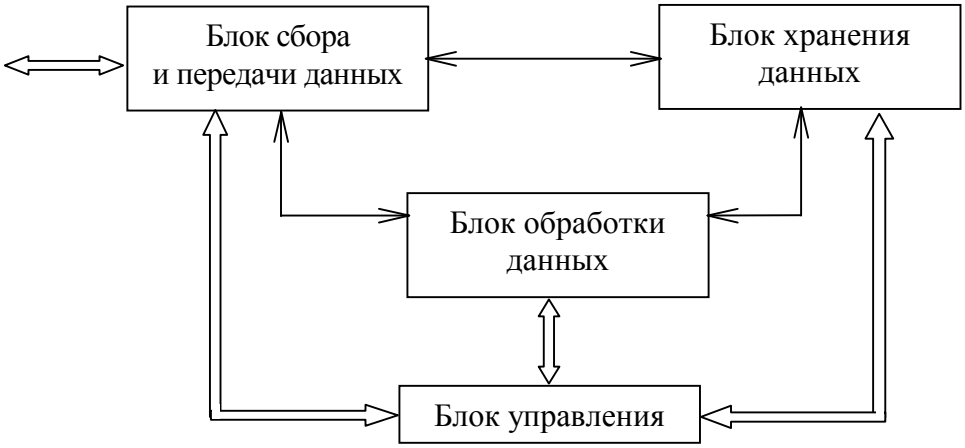
По способу реализации информационные системы делятся на программные, аппаратные и программно-аппаратные.

По степени участия в их работе человека информационные системы бывают автоматическими и автоматизированными.

Аппаратные системы по типу обрабатываемых данных делятся на цифровые и аналоговые системы (дискретные и непрерывные).

По территориальному разнесению системы бывают сосредоточенными и распределенными. В свою очередь распределенные
системы могут быть локально-распределенными и глобально-распределенными системами.

Типовую АИС можно представить как систему, состоящую из следующих блоков (рисунок).



Рисунок

Блок сбора и передачи данных предназначен для ввода/вывода данных в систему. Данные могут вводиться от датчиков (например, в системах технологического контроля), поступать по линии связи (например, через антенное устройство или модем). Принятые данные могут подвергаться предварительной обработке, в частности фильтрации. Затем сохраняются в блоке хранения данных.

Блок хранения данных может быть реализован как база данных или запоминающее устройство на магнитной основе или Flash-памяти (или других носителях данных).

Блок обработки данных выполняет самые разнообразные задачи обработки информации. Например, одним из типов обработки данных является статистическая обработка. Процедуры сжатия
и восстановления данных также являются примерами процедур обработки информации.

Наконец, блок управления осуществляет общее управление автоматизированной информационной системой. Он может быть

реализован как устройство автоматического типа или устройство микропрограммного управления либо как микроЭВМ.

Даже при отсутствии тех или иных блоков в системе на рисунке мы получаем разновидность автоматизированной информационной системы.

Лекция 29
ПРИМЕРЫ АВТОМАТИЗИРОВАННЫХ СИСТЕМ

29.1. Базы данных и знаний

Цель. Рассмотреть вопросы организации и поиска информации в современных базах данных и знаний, реализованных на основе языка XML.

Базы данных и знаний получили в настоящее время очень широкое распространение. Из современных баз данных отметим прежде всего следующие: MS SQL Server 2000/2005, Oracle, Visual FoxPro, Access, My SQL, Dbase, Symantec и др.

Базы данных делятся на сетевые и локальные. В качестве примера локальной БД можно привести Access.

Основным типом базы данных в настоящее время является реляционная база данных. Данные в реляционной БД хранятся в таблицах в виде записей таблицы. Запись состоит из отдельных полей. Каждое поле хранит определенный тип информации. Например, рассмотрим таблицу «Футболисты» (табл. 29.1).

Таблица 29.1

Фамилия	Возраст	Амплуа
Зеедорф	27	Полузащитник
Тони	22	Нападающий
Каннаваро	28	Защитник
Дидо	23	Вратарь
…	…	…

Видим, что запись состоит из трех полей: <Фамилия>, <Возраст>, <Амплуа>. В каждом поле хранится свой тип данных. Обычно в БД хранится много таблиц и эти таблицы взаимосвязаны. Связь обеспечивается через общие поля таблиц. Например, могла бы быть представлена еще таблица «Трансферт», в которой указывалось бы, какие игроки и по какой цене выставлены на продажу (табл. 29.2).

Таблица 29.2

Фамилия	Выставляемая цена
Зеедорф	6 млн. 500 тыс. $
Дидо	10 млн. 200 тыс. $
…	…

Связь между таблицами позволяет выбирать информацию одновременно из нескольких таблиц. Поиск информации играет ключевую роль в современных базах данных. Этот процесс основан на наличии информационной зависимости между данными. Говорим, что поле *A* является ключом, если по значению в этом поле можно уникальным образом определить запись в таблице.

Теперь вспомним описанные в этом пособии методы поиска информации. Как правило, в БД для поиска используются *B*-деревья. Узлам *B*-дерева соответствуют значения ключей. Ключи могут входить в более сложные выражения, которые называются индексными выражениями.

Для работы с БД широкое распространение получил язык SQL (Structured Query Language). Основными операторами данного языка являются следующие: CREATE, UPDATE, DELETE, SELECT, ALTER, INSERT и др.

Приведем примеры операторов.

CREATE TABLE Футболисты (фамилия char(40), возраст int, амплуа char(20))

Эта команда позволяет создать таблицу «Футболисты» с полями <Фамилия>, <Возраст> и <Амплуа>. Указываются типы полей. Поле <Фамилия> имеет строковый тип char длинной в 40 символов. Поле <Возраст> имеет целочисленный тип; поле <Амплуа> имеет строковый тип (char) длиной 20 символов. Созданная по команде CREATE TABLE таблица будет пустой. Для добавления записи в существующую таблицу можно применить команду

INSERT INTO Футболисты VALUES (“Зеедорф”, 27, “полузащитник”)

В команде INSERT указывается название таблицы и значения полей вставляемой записи. Если поле не имеет значения, то указывается NULL:

INSERT INTO Футболисты VALUES (NULL, NULL, NULL)

Команда UPDATE позволяет изменить одну или несколько записей.

Пример.

UPDATE Футболисты SET возраст = 28 WHERE фамилия = “Зеедорф”

Здесь изменяется поле <Возраст>:

SET возраст = 28

для каждой записи, в которой поле <Фамилия> имеет значение «Зеедорф».

Для выборки записей из одной или нескольких таблиц используется команда SELECT.

Пример.

SELECT фамилия FROM Футболисты WHERE амплуа = “полузащитник”

После ключевого слова SELECT перечисляются через запятую названия выбираемых полей. В примере указано единственное поле – <Фамилия>. После ключевого слова FROM указывается одна или несколько таблиц, из которых осуществляется выборка информации. Наконец, после слова WHERE указывается одно или несколько условий, которым должны удовлетворять выбираемые записи.

Пример.

SELECT фамилия, возраст FROM Футболисты WHERE амплуа = “полузащитник” AND возраст < 30 and возраст > 21

В этом примере указаны сразу три простых условия.

Команда DELETE используется для удаления записей.

Пример.

DELETE FROM Футболисты WHERE Фамилия = “Зеедорф”

Эта команда удаляет все записи, в которых фамилия имеет значение «Зеедорф».

При выборе информации из БД можно осуществлять вычисление полей. Например, если футболисту больше 30 лет, то увеличивается трансфертный налог:

SELECT фамилия, возраст, (IF (возраст > 30; цена * 0.13; цена * 0.1)) AS налог FROM Футболисты, Трансферт WHERE Футболисты**.**фамилия = Трансферт**.**фамилия

Эта команда выбирает поля <Фамилия>, <Возраст> и вычисляет поле налог как функцию IF (возраст > 30; цена * 0.13; цена * 0.1).

Функция IF имеет три аргумента. Первый аргумент задает условие: возраст > 30. Если в записи возраст больше 30, то функция IF возвращает второй аргумент (цена * 0.13); иначе – третий аргумент (цена * 0.1)).

Широкое распространение получили XML-базы данных (особенно и главным образом в Интернете). XML-базу данных можно рассматривать как базу знаний фреймового типа. Познакомимся
с XML-базами данных более подробно.

29.2. Базы данных XML

Простейшим примером XML-документа может служить следующий:

```
<? xml version=”1.0” ?>
<root>
        <book>
                <title> War and Peace </title>
                <author> L. Tolstoy </author>
        </book>
        <book>
                <title> Three men in the boat </title>
                <author> Jerom K. Jerom </author>
        </book>
</root>
```

Здесь <root>, <book>, <title>, <author> суть элементы (теги). Каждый элемент имеет спаренный с ним закрывающий элемент, в записи которого присутствует наклонная черта. Область, ограниченная местом появления элемента, называется областью действия элемента. Выделяется один главный элемент (<root>), область действия которого содержит области действия всех других элементов. Этот элемент называется корневым. XML-документ представляет собой текстовый файл, имеющий расширение .xml. Этот документ просматривает браузер Internet Explorer. Для просмотра в среде Windows достаточно дважды щелкнуть мышью по имени XML-файла.

Элементы могут иметь атрибуты. Атрибуты указываются непосредственно в тегах элементов.

```
<? xml version=”1.0” ?>
<root>
    <book>
        <title> War and Peace </title>
        <author country=“Russia”> L. Tolstoy </author>
    </book>
    <book>
        <title> Three men in the boat </title>
        <author country=“England”> Jerom K. Jerom </author>
    </book>
</root>
```

Иногда в XML-файл помещают схему описания документа (DTD). Такая схема определяет структуру документа и типы элементов.

```
<? xml version=”1.0” ?>
    <DOCTYPE root [
    <! Element root(book)>
    <! Element book(title, author )>
    <! Element title(#PCDATA)>
    <! Element author(#PCDATA)>
    <! ATTLIST author
        country CDATA #REQUIRED>
                ]>
<root>
    <book>
        <title> War and Peace </title>
…
```

Здесь схема DTD начинается тегом

<DOCTYPE root […,

в котором указывается имя корневого тега <root>. Затем следует перечисление элементов. Каждый элемент определяется в теге <! Element имя(…)>.

Например,

<! Element book(title, author)>

определяет элемент book, причем в строках записываются имена включенных в него элементов. Если элемент не содержит других элементов, то в скобках указывают тип элемента, например:

#PCDATA – последовательность алфавитно-цифровых символов;

EMPTY – никаких данных нет.

Тег <! ATTRIBUTE …> используется для указания атрибутов элемента. Опция #REQUIRED означает, что атрибут всегда должен иметь значение. Альтернативной является опция #IMPLIED (атрибут не обязателен).

Далее в этой части мы будем рассматривать работу с XML-документами.

XML-документы можно рассматривать в качестве базы знаний фреймового типа. Более подробные сведения можно найти в [22].

Лекция 30
КЛИЕНТ-СЕРВЕРНЫЕ СИСТЕМЫ

Цель. Изучить принципы построения и работы клиент-серверных систем.

Клиент-серверные системы связаны с развитием сетевых технологий. Сервер – это компьютер, обрабатывающий программные запросы клиентов. Познакомимся с этой технологией на примере сети Интернет. Клиентские приложения (сайты) создаются на базе таких языков, как Java, C#, JavaScript, VBScript, Perl и др. Обычно сайт – это оконный документ, содержащий необходимые средства для ввода информации, ее визуализации и отправки серверу. Сайты могут размещаться также непосредственно на стороне сервера. Связь между клиентом и сервером выполняет посредник – программа Internet Explorer (браузер). Имеются и другие интернет-посредники, например, Opera или NetScape Navigator. Браузер передает данные сайта программе веб-сервера. Широкое понятие сервера с этого момента следует детализировать. Во-первых, мы различаем веб-сервер, во-вторых, серверы базы данных и, в-третьих, серверы приложений. Все три типа серверов объединяются под общим термином. В действительности веб-сервер «общается» именно с браузером. Его задача – получить данные от клиента и запустить приложение-обработчик, которое эти данные обработает. Такие приложения обработчики пишут на языках Java, C#, JSP, ASP, ASP.NET и др. В качестве веб-сервера можно отметить IIS (входит в стандартную поставку Windows 2000/NT/XP), Apache, Tom Cat, Web Sphere и пр. Таким образом, на стороне сервера должен быть запущен Web.Server. Веб-сервер самостоятельно связывается с браузером, получает от него данные сайта и другую информацию и вызывает так называемую программу скрипта. Простая программа скрипта на языке ASP может иметь, например, следующий вид:

```
<HTML>
        <body>
                <%
                set db = Server.CreateObject(“ADODB.Connection”)
```

```
            db.open (“xmldb”)
            set rs = db.execute (“Select * from stud”)
            do until rs.EOF
                  Response.write (“Found!”)
                  Response.write (“name:” & rs(“name”))
                  Response.write (“group:” & rs(“group”))
                  Response.write (“<br>”)
                  rs. MoveNext
            loop
            db.close
            set rs=nothing
            %>
      </body>
</HTML>
```

Программа скрипта встраивается в текст HTML-документа, который и представляет собой сайт, открываемый пользователю веб-браузером. Сайты обычно создают на языке HTML. HTML-документ помещается между тегами <HTML> <body> … </body> </HTML>. Вместо многоточия вставляется последовательность различных тегов для управления содержимым. Документ HTML имеет расширение .html (или .htm). Он открывается браузером (для этого достаточно дважды щелкнуть мышью на имени HTML-файла). ASP-скрипты имеют расширение APS. Они представляют собой также HTML-файлы, как в примере выше, но открываются не браузером, а веб-сервером. ASP-скрипт содержит теги <% ... %>, между которыми помещается текст программы скрипта. В нашем случае текст программы скрипта содержит следующие команды языка Visual Basic

```
set db = Server.CreateObject(“ADODB.Connection”)
```

Эта команда создает объектную переменную по имени db, которая используется для создания соединения (connection) с базой данных. Открытие базы данных выполняет команда

```
db.open (“xmldb”),
```

где xmldb – это имя базы данных.

Следующая команда

```
set rs = db.execute (“Select * from stud”)
```

осуществляет выполнение SQL-запроса

Select * from stud,

т. е. выбирает все записи из таблицы с именем stud из базы данных xmldb. Переменная rs будет содержать все выбранные записи.

Далее, выполняется цикл по числу выбранных записей:

do until rs.EOF

…

loop

В цикле происходит многократный вывод на экран информации командой Response.write (…). Выбираемые из таблицы значения полей содержатся в переменных rs(“name”) и rs(“group”), где “name” и “group” – названия полей в записях базы данных.

Итак, обращение к базе данных реализуется через инструкцию

db.execute (“Select * from stud”)

Обычно в качестве СУБД при работе в Интернете используется серверная СУБД типа MySQL или SQL Server 2000/2005. Программа сервера базы данных в этих случаях должна быть предварительно запущена и активна. Разобранное нами взаимодействие клиента и сервера выполняется через HTML-сайт клиента и ASP-документ на сервере. Посредником между этими сайтами выступает Internet Explorer.

30.1. Средства для создания клиентских сайтов

Структура HTML-документа определяется упорядоченным набором тегов следующего вида:

<HTML>
<HEAD> текст
</HEAD>
<BODY> управляющие теги и текст
</BODY>
</HTML>

Программа-клиент браузер при просмотре файлов с этими тегами выполняет отображение документа в окне. Подобный файл можно набрать в любом текстовом редакторе. Теги играют роль команд и заставляют браузер выполнить предписываемые

ими действия. Область действия тега определяется тем местом, в котором он указан, и тем местом, в котором он закрыт (помещают в угловые скобки с предшествующей косой чертой). В общем случае теги имеют параметры. Рассмотрим некоторые из них.

```
<BODY alink = color background = url
        bgcolor = color
        scroll = yes/no
        bgproperties = fixed>
        …
</BODY>
```

Параметр alink задает цвет гиперссылок. Параметр background задает адрес рисунка, на фоне которого отображается документ. Параметр bgcolor задает цвет фона документа. Параметр scroll задает возможность прокрутки документа (yes). Параметр bgproperties = fixed устанавливает невозможность прокрутки фонового рисунка документа при прокрутке содержимого документа. В качестве возможных для использования цветов можно указывать: aliceblue – синий, aqua – голубой, azure – лазурный, gold – золотой, gray – серый, green – зеленый, crimson – кремовый, darkmagenta – малиновый, ivory – цвет слоновой кости, purple – розовый, sienna – фиолетовый и т. д.

Для форматирования текста в документе следует использовать следующие теги:

от <H1> до <H6> – изображает текст от наименьшего до максимального размера. Этот тег использует параметр aling = center/left/right, который устанавливает смещение текста в окне (по центру/влево/вправо).

Пример.

<H1 align = center> Учиться, учиться и еще раз учиться <H1>

<P> – задает начало и конец параграфа.

<CENTER> – размещает текст и рисунки по центру.

 – переводит каретку на следующую строку.

<HR> – горизонтальная линия. Длину линии в пикселах можно задать с помощью параметра width = число; цвет линии –

параметром color = цвет; толщину линии – параметром size = число.

<HTML>
<BODY bgcolor = floralwhite>
<H1> Simple document <HR align=left size=4 width=200 color=red> </HR> </H1>
</BODY>
</HTML>

Можно использовать теги:

<B> – задает жирный вариант текста;

<I> – задает курсивный вариант текста;

<U> – задает подчеркивание текста;

<Strike> – устанавливает перечеркивание текста;

<CITE> – используется для выделения цитат;

<PRE> – используется для учета пробелов;

<Font color=цвет face=шрифт size=24> OK </Font> – задает цвет, название и размер шрифта;

<! -- комментарий --> – используется для вставки комментария в тело HTML-документа.

Для размещения в документе рисунка наберите:

<IMG src=url alt=text border=n height=n1 width=n2>

Здесь:

src – адрес файла с изображением или видеоклипом;

alt – текст, выводимый на месте изображения, если оно не загружено браузером;

border – толщина рамки;

height и width – соответственно, высота и ширина изображения.

Пример.

<IMG src=“globe.jpeg” border=2 alt=“Not available” width=200 height=400>

Рассмотрим, как в документе размещаются гиперссылки. Гиперссылка – это строка, при щелчке мышью на которой выполняется переход к другому документу, связанному с этой гиперссылкой. Задание гиперссылки реализуется так:

 press here

В данном примере в качестве гиперссылки используется строка press here. Щелчок кнопкой мыши в области гиперссылки приведет к загрузке нового документа из файла w.html.

Для вызова из клиентского сайта северного скрипта следует задать параметр action тега <Form>.

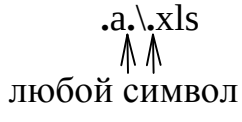
```
<html>
<body>
<form method="post" action="C:\Program Files\ActivePerl\Perl\
bin\first.cgi">
<input type="text" value="abc">
</form>
</body>
</html>
```

Обратим внимание на следующие два момента. Первый: в теге <Form> заданы два важных параметра: method и action. Параметр action указывает сетевой адрес программы обработки для сайта, находящийся на стороне сервера (отметим, что клиент и сервер могут находиться на одном и том же компьютере). Второй: параметр method указывает, каким образом будут передаваться на сервер данные скрипта. Соединение с сервером будет установлено при нажатии кнопки SEND. Такая кнопка имеет специальный тип – Submit. В методе Get (method="get") строка параметров в виде value=значение, …, value=значение передается вместе с заголовком запроса; в методе post (method="post") она передается отдельным файлом.

30.2. Средства обработки текста в клиентских HTML-документах

В настоящее время достаточно активно используются механизмы обработки так называемых регулярных выражений. Регулярное выражение представляет собой шаблон, который задает набор символьных фраз, ему удовлетворяющих. Символ «**.**» (точка) соответствует любому символу. Пара «**\.**» соответствует самой точке.

Пример.



Данному регулярному выражению могут соответствовать названия файлов типа rad.xls, mar.xls и др. В квадратных скобках указывают список любых символов, из которых выбирается один (любой).

Пример.

[0-9]a.\.xls

Теперь подходящими являются имена типа 2az.xls и 0aa.xls и т. п.

Выражение [^0-9] соответствует любому символу, отличному от цифры.

Комбинация «\[« соответствует квадратной скобке.

Пример. MyArray\[[0-9]\] соответствует myArray[1] и т. п.

Используются следующие специальные символы:

\d – любая цифра;

\D – любой нецифровой символ;

\w – любой алфавитно-цифровой символ;

+ – соответствует вхождению одного или нескольких символов.

Пример. [\w.]+ может соответствовать zzz.aaa или z.az.ww и т. д.

Знак вопроса (?) соответствует вхождению только одного или ни одного символа.

Пример. a?a порождает aba, aza, aa и пр.

В фигурных скобках указывают число вхождений:

#[0-9 A-F] {4}

соответствует, например, #0000 (четыре вхождения цифры 0).

Шаблон

\b cap

соответствует любому слову, начинающемуся с cap, а шаблон cap\b – любому слову, заканчивающемуся на cap.

Рассмотрим, как использовать этот язык регулярных выражений в клиентских сайтах. Основное его назначение – выполнить поиск вхождений любой последовательности, удовлетворяющей шаблону. Следующий HTML-документ является иллюстрацией.

```
<HTML>
<Script language=VBScript>
<!--
```

```
Sub parser
        StrMain="The website ASP.com is to be find"
        Set RegularExpressionObject=New RegExp
        With RegularExpressionObject
                .Pattern="ASP"
                .IgnoreCase=False
                .Global=True
        End With
        Set arr=RegularExpressionObject.Execute(strMain)
                s=""
        If arr.Count>0 Then
                For each x in arr
                        S=s&x.Value&"::"&x.FirstIndex
                Next
        Else
                S="Not found"
        End If
        MsgBox s
        Set RegularExpressionObject=nothing
End Sub
-->
</Script>
<Form><Input Type="Button" Value="Parser" onClick="parser">
</Form>
</HTML>
```

Разберем этот документ. К числу ранее рассмотренных тегов здесь добавлен тег <Script language=VBScript>. Этот тег используется для вставки программы или программ, написанных на языке VBScript (Visual Basic Script). Все программы дополнительно помещаются в область комментариев, ограниченную тегами <! -- … -->. Если комментарии не указывать, то содержимое скрипта выполнится один раз при загрузке документа браузером.

В нашем примере имеется единственная программа с именем parser:

```
Sub parser
…
End Sub
```

Собственно, изучение этой программы требует знания языка VBScript. Поскольку это не входит в наши задачи, то ограничимся только тем фрагментом, который связан с регулярными выражениями.

Создаем объектную переменную типа регулярного выражения командой

Set RegularExpressionObject=New RegExp

Переменная с именем RegularExpressionObject будет хранить шаблон, к тому же она обладает рядом методов для работы с регулярными выражениями. Сам шаблон задается в свойстве Pattern командой

.Pattern=“ASP”

Эта сокращенная запись команды. Полная запись должна иметь такой вид:

RegularExpressionObject.Pattern=“ASP”

Мы видим, что сначала следует указать имя объектной переменной (RegularExpressionObject), а затем через точку – имя свойства Pattern. Аналогичным образом указываются и методы. Для сокращения используется конструкция:

```
With RegularExpressionObject
…
End With
```

Эта конструкция отменяет необходимость все время записывать имя RegularExpressionObject – указывается только точка.

Свойство .IgnoreCase=False означает, что учитывается регистр записи символов, т. е. шаблоны aSP и ASP суть два разных шаблона.

Свойство .Global означает, что после создания объект доступен и в других программах.

Выполнение регулярного выражения реализуется командой

Set arr=RegularExpressionObject.Execute(strMain)

Метод Execute находит все слова в строке StrMain, содержащие слово ASP и формирует из всех найденных слов массив arr.

Далее, проверяется, найдено ли хотя бы одно такое слово (If arr.Count>0). При положительном результате проверки организуется цикл для вывода этих слов и позиций, в которых они встречались. Каждое слово получают как обращение x.Value, x – элемент массива arr. Доступ к словам массива arr реализуется в цикле

```
For each x in arr
…
Next
```

Позиция вхождения слова, содержащего ASP, определяется ссылкой x.FirstIndex.

Лекция 31
СИСТЕМЫ ТЕХНИЧЕСКОЙ ДИАГНОСТИКИ

Цель. Изложить принципы построения систем технической диагностики. Проиллюстрировать материал на примерах.

Методы теории информационных процессов, которые мы рассмотрели в первой части, находят широкое применение в различных областях деятельности. Одной из таких областей является техническая диагностики. В основе диагностики лежит тест, который представляет собой совокупность вопросов (y_1, y_2, …, y_m). Для простоты будем считать, что вопросы являются двоичными (т. е. требуют ответа «Да» (1) или «НЕТ» (0)). Имеется набор результатов диагностирования: r_1, r_2, …, r_n. Поскольку число различных комбинаций вопросов и ответов составляет 3^m (имеется три варианта: не задавать вопрос; получить ответ «НЕТ» или получить ответ «ДА»), то это число может быть огромным. Поэтому не всегда возможно хранить комбинации и их результаты в виде правил типа:

$$\text{Если } y_1 = *,\ y_2 = *,\ y_3 = 0, \ldots, y_m = 1, \text{ то } r_1.$$

Это правило читается так: если вопросы y_1, y_2 не задавались, на y_3 получен ответ «НЕТ», …, на y_m получен ответ «ДА», то результат диагностирования r_1.

Более реально – учитывать свойство «различения». Говорим, что результаты (исходы) r_i и r_j различает вопрос y_t, если при $y_t = 1$ не может иметь места r_i, а при $y_t = 0$ не может иметь места r_j. Здесь запись $y_t = 1$ означает, что на вопрос y_t получен положительный ответ, а $y_t = 0$ означает получение отрицательного ответа. Ясно, что r_i и r_j в общем случае могут различать не единственный вопрос.

Минимальным тестом называется тест, содержащий наименьшее возможное число вопросов, задав которые, можно получить точные значения об исходе (результате). Иначе говоря, результат будет определен однозначно.

Следует заметить, что результаты могут характеризоваться вероятностями: p_1, p_2, …, p_n. В самом деле, какие-то исходы

встречаются более часто, а какие-то – менее часто. Наконец, вопросы
могут иметь цены c_i, характеризующие затраты на проведение вопроса (эксперимента). Поэтому разумно говорить также о тесте минимальной стоимости.

31.1. Определение минимального теста при равных вероятностях исходов и одинаковых стоимостях вопросов

Сформулируем эту задачу как задачу о минимальном покрытии. Составим матрицу покрытия M вида

	s_1	s_1	s_1	…	s_1	$(t = C_m^2)$
y_1	0	1	0	…	1	
y_2	1	0	0	…	0	
…	…	…	…	…	…	
y_m	0	1	0	…	0	

Здесь y_1, …, y_m – вопросы; s_i – состояние, соответствующее некоторой паре исходов $s_i = (r_\alpha, r_\beta)$.

Количество всех таких пар равняется C_m^2. Каждая пара обязательно должна быть представлена каким-то состоянием (столбцом) матрицы M. Например, пусть исходы могут быть такими: r_1, r_2, r_3, r_4. Тогда в M будут следующие состояния:

$$s_1 = (r_1, r_2);\ s_2 = (r_1, r_3);\ s_3 = (r_1, r_4);\ s_4 = (r_2, r_3);$$
$$s_5 = (r_2, r_4);\ s_6 = (r_3, r_4).$$

Всего $\dfrac{m(m-1)}{2} = \dfrac{4 \cdot 3}{2} = 6.$

Задача формулируется следующим образом. Для матрицы покрытия M найти минимальное по числу строк множество $\pi = \{y_{\pi 1}, y_{\pi 2}, ..., y_{\pi k}\}$ (вопросов), такое, что в каждом столбце M хотя бы одна строка из π содержала бы «1». При этом заметим, что строка y_t содержит в столбце $s_k = (r_{k1}, r_{k2})$ «1», если она различает исходы r_{k1} и r_{k2}, сопоставленные с этим столбцом. В противном случае строка y_t содержит в этом столбце «0».

Эта известная задача дискретной математики может быть решена методами типа ветвей и границ [23], групповых резольвент [24] и др.

31.2. Определение минимального теста при равных вероятностях исходов и разных стоимостях вопросов

По существу, мы опять имеем дело с задачей о покрытии. На этот раз каждой строке матрицы M приписан неотрицательный целый вес c_i (цена вопроса – y_i).

Нужно найти покрытие матрицы M с минимальной суммарной ценой. Эта задача известна как задача о минимальном взвешенном покрытии.

Лекция 32
ОПРЕДЕЛЕНИЕ МИНИМАЛЬНОГО ТЕСТА ПРИ РАЗНЫХ ВЕРОЯТНОСТЯХ И ОДИНАКОВЫХ СТОИМОСТЯХ ВОПРОСОВ

Цель. Рассмотреть приближенный алгоритм построения минимального диагностического дерева для указанного случая.

Рассмотрим приближенное решение этой задачи. Изложение проведем на примере следующей матрицы:

$M =$

	(1, 2)	(1, 3)	(1, 4)	(2, 3)	(2, 4)	(3, 4)
y_1	1		1			
y_2	1	1				1
y_3				1	1	
y_4					1	
y_5		1		1		
y_6				1		1
y_7			1		1	

$p_1 = 0,2;\ \ p_2 = 0,4;\ \ p_3 = 0,25;\ \ p_4 = 0,15$, где p_i – вероятность исхода r_i.

Ясно, что $\sum_{i=1}^{4} p_i = 1$. Используемый нами эвристический принцип таков: для выявления более вероятного исхода следует задать вопросов меньше, чем для выявления менее вероятного. Этот принцип реализован в алгоритме Хаффмана. Применим алгоритм Хаффмана таким образом. Упорядочим вероятность исходов по убыванию и будем последовательно объединять последние два исхода и при этом каждый раз производить пересчет вероятностей. Получим дерево диагностирования, показанное
на рис. 32.1.

Нетрудно сообразить, что «разделить» исходы r_1 и r_2 можно либо вопросом y_1, либо вопросом y_2 (нами произволь-

но выбран y_1). Но какой вопрос задать, чтобы отделить r_3 от r_1 и r_4?

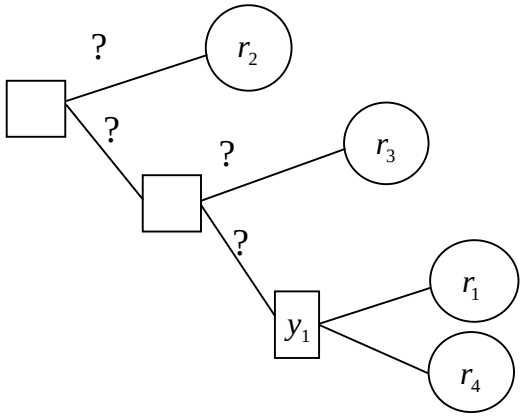


Рис. 32.1

Чтобы решить эту задачу оставим в матрице M только столбцы (r_1, r_3), (r_3, r_4):

$M' =$

	(1, 3)	(3, 4)
y_1		
y_2	1	1
y_3		
y_4		
y_5	1	
y_6		1
y_7		

Нас будет, как и ранее, интересовать минимальное покрытие M'. Оно единственно: $\pi = \{y_2\}$.

Таким образом, нам нужен вопрос y_2. Именно он отделит r_3 от r_1 и r_4. С учетом этого факта рис. 32.1. преобразуем в рис. 32.2.

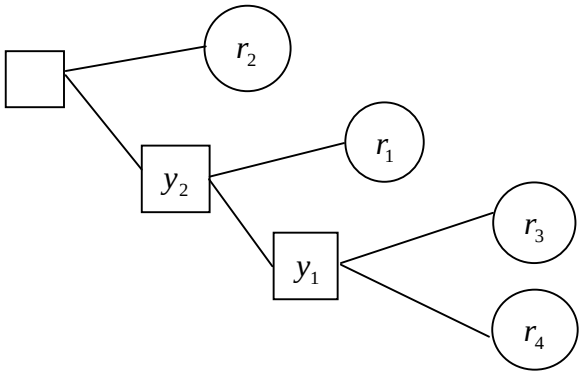


Рис. 32.2

Теперь остается только одна неопределенная вершина: она отделяет исход r_2 от всех остальных. Поступим аналогично: выпишем из исходной матрицы столбцы (r_2, r_1), (r_2, r_3), (r_2, r_4) и найдем минимальное покрытие следующей матрицы:

$M'' =$

	(1, 2)	(2, 3)	(2, 4)
y_1	1		
y_2	1		
y_3		1	1
y_4			1
y_5		1	
y_6		1	
y_7			1

Возьмем, например, минимальное покрытие $\pi = \{y_1, y_3\}$. Эти два вопроса позволяют отделить исход r_2 от всех остальных (рис. 32.3)

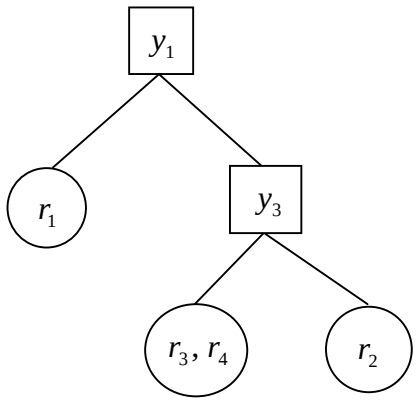


Рис. 32.3

Теперь объединим рис. 32.2 и 32.3.

Результат наших действий следует пояснить. Напомним, что вопрос y_i разделяет исходы r_α и r_β, если ответ «Да» не допускает r_β, а ответ «НЕТ» не допускает r_α.

Однако при этом мы ничего не можем сказать относительно исходов, отличных от r_α и r_β. Поэтому обратимся к рис. 32.3. Вопрос y_1 отделяет r_1 и r_2, r_1 и r_4 (как видно из матрицы M). Следовательно, он допускает только такой исход (рис. 32.4).

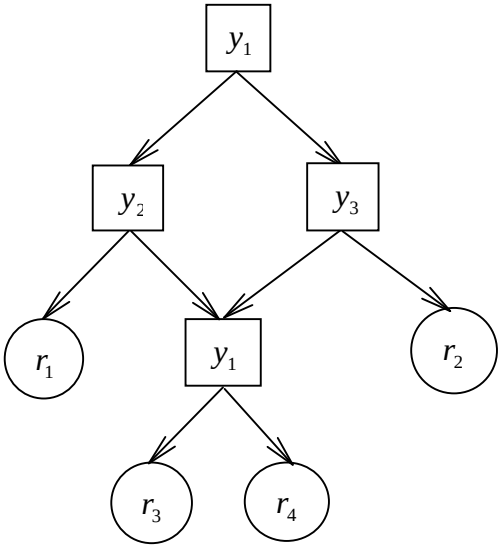


Рис. 32.4

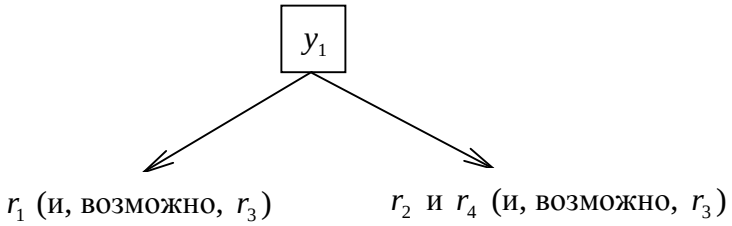


Рис. 32.4

Теперь уже вопрос y_3 к множеству r_2, r_3, r_4 выделяет r_2 по одной ветви и r_2, r_3 – по другой (рис. 32.3).

Итак, суть описанного эвристического метода заключается в следующем. Сначала алгоритм Хаффмана строит минимальное или близкое к нему дерево, используя только вероятности. При этом не все вопросы, задаваемые в узлах дерева, известны. Затем производится доопределение вопросов. Для этого решается следующая задача: какое минимальное число вопросов задать, чтобы «отделить» единственный исход от других? Эта задача решается методом минимального покрытия. Полученное множество

вопросов «дообъединяется» с известной частью построенного дерева Хаффмана.

Лекция 33
ОПРЕДЕЛЕНИЕ МИНИМАЛЬНОГО ТЕСТА ПРИ РАЗНЫХ ЦЕНАХ ВОПРОСОВ И РАЗНЫХ ВЕРОЯТНОСТЯХ ИСХОДОВ

Цель. Рассмотреть приближенный алгоритм построения минимального диагностического дерева для указанного случая.

Эту задачу будем решать методом ветвей и границ (МВГ). Прежде всего от нас потребуется указать способ оценки какого-либо дерева (графа) вопросов. Обратимся к рис. 33.1.

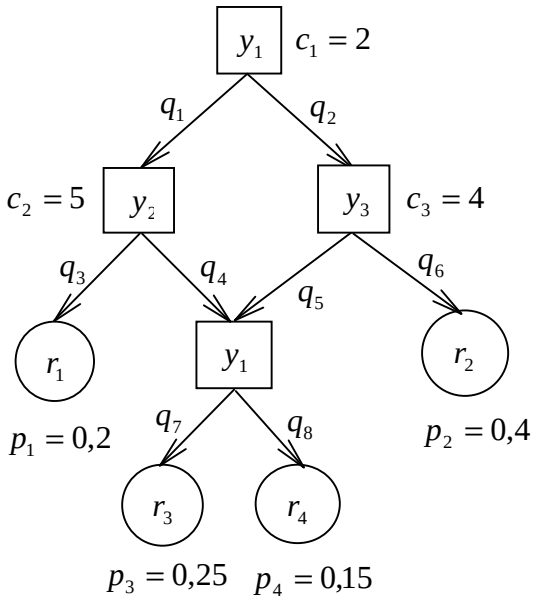


Рис. 33.1

Цена этого графа складывается из величин

$$D_j = p_j \sum_{i \in w_j} c_i,$$

где p_i – вероятность j-го исхода; $\sum_{i \in w_j} c_i$ – сумма цен вопросов, ведущих к исходу r_j.

Так, $D_2 = (c_1 + c_3) \cdot p_2 = 2,4$; $D_1 = (c_1 + c_2) \cdot p_1 = 1,4$ и т. д. Поэтому ценой графа на рис. 33.1 будет сумма $D_1 + D_2 + D_3 + D_4$. Может оказаться так, что имеется несколько различных путей к какому-то исходу. Например, к исходу r_3 ведут пути $y_1 \to y_2 \to y_1$ и $y_1 \to y_3 \to y_1$. Однако всегда можно найти вероятность того или иного пути, зная вероятности исходов r_i.

Для этого, вообще говоря, нам потребуется составить алгебраическую систему уравнений. Покажем, как это сделать. Каждой ветви графа припишем неизвестную вероятность q_k (рис. 33.1). Теперь можно записать:

$$q_1 + q_2 = 1; \tag{33.1a}$$

$$q_3 + q_4 = 1; \tag{33.1b}$$

$$q_5 + q_6 = 1; \tag{33.1c}$$

$$q_7 + q_8 = 1; \tag{33.1d}$$

$$q_1 q_2 q_8 + q_2 q_5 q_8 = 0,15; \tag{33.1e}$$

$$q_2 q_6 = 0,4; \tag{33.1f}$$

$$q_1 q_4 q_7 + q_2 q_5 q_7 = 0,25; \tag{33.1g}$$

$$q_1 q_3 = 0,2. \tag{33.1h}$$

Из (33.1d), (33.1e), (33.1g):

$$q_7 + q_8 = 1;$$

$$(q_1 q_4 + q_2 q_5) q_8 = 0,15;$$

$$(q_1 q_4 + q_2 q_5) q_7 = 0,25.$$

Обозначим $q_1 q_4 + q_2 q_5 = a$. Тогда

$$a q_8 = 0,15;$$

$$a(1 - q_8) = 0,25,$$

откуда $a = \frac{2}{5}$, $q_8 = \frac{3}{8}$, $q_7 = \frac{5}{8}$.

Далее найдем приближенные значения:

$$q_3 \approx 0,405;$$

$$q_4 \approx 0,595;$$

$$q_2 \approx 0,51;$$

$$q_1 \approx 0,49;$$

$$q_5 \approx 0,22;$$

$$q_6 \approx 0,78.$$

Заметим, что полученная нами система уравнений является нелинейной, так что для ее решения следует использовать вспомогательный пакет, например, MATHCAD.

Таким образом, теперь легко найти

$$D_3 = (c_1 + c_2 + c_1)(q_1 q_4 q_7) + (c_1 + c_3 + c_1)(q_2 q_5 q_7) =$$
$$= 9 \cdot 0,49 \cdot 0,595 \cdot 0,627 + 8 \cdot 0,51 \cdot 0,22 \cdot 0,78 \approx 2,45.$$

$$D_4 = (c_1 + c_2 + c_1)(q_1 q_4 q_8) + (c_1 + c_3 + c_1)(q_2 q_5 q_8) =$$
$$= 9 \cdot 0,49 \cdot 0,595 \cdot 0,375 + 8 \cdot 0,51 \cdot 0,22 \cdot 0,375 \approx 1,32.$$

Следовательно, общая цена дерева на рис. 33.1 составляет $\sum_{i=1}^{4} D_i = 7,57.$

Помимо представленного способа оценки всего дерева вопросов, нам нужно научиться оценивать нижнюю границу цены неполностью сформированного дерева.

Возьмем, например, следующий фрагмент дерева (рис. 33.2).

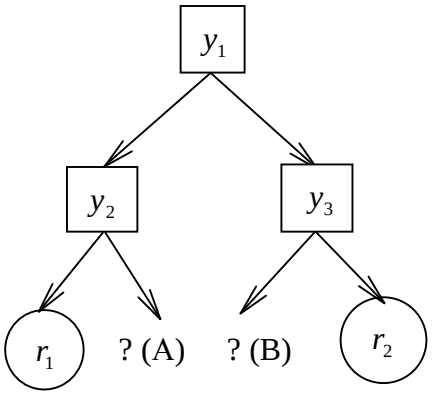


Рис. 33.2

Для того чтобы получить интересующую нас наилучшую оценку дерева на рис. 33.2 его нужно некоторым оптимальным образом достроить до конца. Чтобы это сделать, нужно вместо вопросов

определить, какие исходы должны быть определены по оставшимся ветвям. Например, возьмем ветвь A. После задания вопросов y_1, y_2 остается разделить вопросы r_3 и r_4. Для этого нужно найти матрицу покрытия для этих вопросов и определить на ней стоимость минимального взвешенного покрытия. Эта величина сразу даст нам значения c_A. Аналогичным образом получим значения c_B (c_A определит стоимость оставшихся вопросов для ветви A, а c_B – стоимость оставшихся вопросов для ветви B). Теперь нужно оценить вероятность ветви A и вероятность ветви B. Поскольку они представляют события $r_3 \vee r_4$, то эти вероятности находят по формуле

$$P(A \vee B) = P(A) + P(B) - P(A)P(B). \qquad (33.2)$$

В нашем примере минимальное взвешенное покрытие ищем по матрице (в столбце c_i указаны веса вопросов):

$M =$

	(3, 4)	c_i
y_1		2
y_2	1	5
y_3		4
y_4		2
y_5		3
y_6	1	8
y_7		6

Ясно, что наилучшее покрытие есть $\{y_2\}$ с весом 5. Поэтому рис. 33.2 достраивается следующим образом (рис. 33.3).

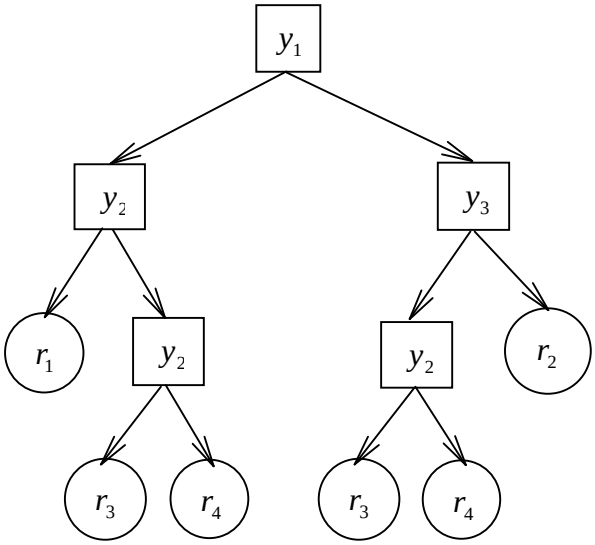


Рис. 33.3

Теперь нам остается объединить одноименные исходы, чтобы можно было записать вероятность этого исхода (рис. 33.4).

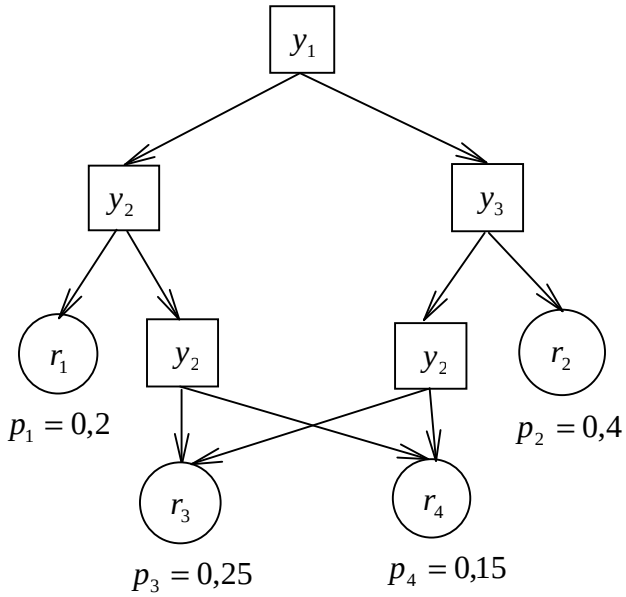


Рис. 33.4

Данный граф можно оценить точно так же, как мы делали ранее. С другой стороны, если мы начинаем с «0» и ищем минимальное взвешенное покрытие матрицы *M*, например пусть это будет

$$\pi = \{y_1, y_2, y_3\},$$

то исходный граф примет следующий вид (рис. 33.5).

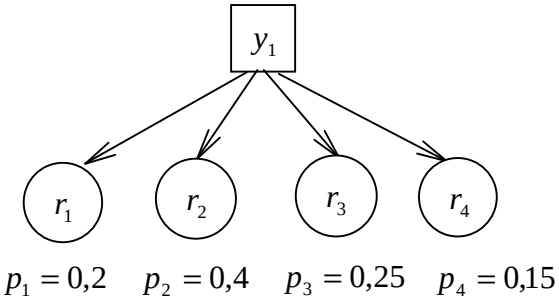


Рис. 33.5

Оценка графа на рис. 33.5 следующая:

$$D = c_1 + c_2 + c_3.$$

Таким образом, способ вычисления оценки для МВГ нами разобран. Перейдем к описанию сути МВГ для рассмотренной задачи.

Под состоянием задачи S_i понимается текущий граф вопросов. Например, граф на рис. 33.2 вполне может соответствовать некоторому состоянию задачи. Исходное состояние – это пустой граф. Пусть мы находимся в состоянии S_i. Тогда из этого состояния всегда ясно, какое множество исходов следует различить. Значит, для S_i можно записать все те вопросы, которые различают любую пару исходов r_α и r_β из множества исходов, приписанных S_i. Пусть одним из таких вопросов будет y_k. Тогда мы можем создать новое дерево состояний, как показано на рис. 33.6.

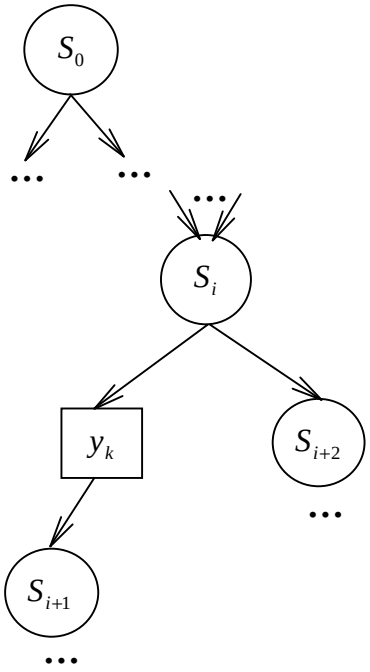


Рис. 33.6

Итак, мы попали в вершину S_i. Мы можем задать вопрос y_k и перейти в состояние S_{i+1} либо не задавать его и перейти в состояние S_{i+2}. Тем самым метод ветвей и границ создает дерево состояний. Каждая вершина этого дерева и есть состояние. Мы теперь знаем, что каждому состоянию соответствует некий фрагмент дерева вопросов. Для каждого такого фрагмента мы показали, как оценивать значение нижней границы. Для работы с деревом состояний оценка нижней границы каждой вершины-состояния является важнейшей операцией, реализуемой МВГ. Запомним следующее. Текущей будем считать ту вершину дерева состояний, в которой минимальна оценка нижней границы. Из текущей вершины мы всегда пытаемся перейти в новое состояние (породить новую вершину в дереве состояний). Такую вершину можно породить либо нет.
В последнем случае текущая вершина закрывается и далее не рассматривается. Новая текущая вершина выбирается только из числа открытых. В конце концов, применяя данную технику, нам удастся построить вершину-состояние, которому соответствует законченное дерево вопросов. Такое дерево вопросов дает значение

рекорда *R* для МВГ. Как только мы найдем рекорд *R*, МВГ будет выполняться несколько иначе, а именно, если оценка нижней границы текущей вершины больше либо равна *R*, то такая вершина закрывается и далее не рассматривается. Если окажется, что все вершины закрыты, то конец всего алгоритма – конечная вершина-состояние, давшая наилучшее значение рекорда *R*, – и определяет результиру-ющее дерево вопросов. В противном случае МВГ требует выбрать новую текущую вершину и строить для нее все возможные
продолжения.

Трудоемкость МВГ может оказаться весьма высокой. Это связано с порождением огромного по числу вершин дерева состояний.

Лекция 34
ПРОБЛЕМА ИНФОРМАЦИОННОЙ СЛОЖНОСТИ

Цель. Изложить вариант оценки вычислительной сложности для задач типа распознавания свойств. Сформулировать и разъяснить основные концепции.

Для изучения проблемы информационной сложности задач нужно, прежде всего, оговорить, какие задачи мы будем рассматривать. По очевидным причинам такие задачи должны быть в некотором смысле универсальными, т. е. к ним можно было бы свести многие другие задачи из некоторого широкого класса задач. Довольно часто в качестве такого класса задач выбирают задачи распознавания свойств, требующие только одного из двух ответов: «ДА» или «НЕТ». Кроме того, требуют, чтобы решение задачи можно было эффективно проверить с помощью детерминированного алгоритма. Ранее мы рассматривали задачи сортировки и отмечали, что вычислительная сложность алгоритма сортировки определяется числом сравнений. Если это число растет как некоторый полином от размера исходного массива, то такой алгоритм называется эффективным. Обобщим это определение.

Детерминированный алгоритм называется *эффективным*, если число выполняемых им операций (не только сравнения) растет как некоторый полином от размера задачи. Под размером задачи понимают, вообще говоря, длину записи задачи при условии, что эта запись неизбыточна.

Итак, найти хорошее решение задачи – значит, построить для нее эффективный детерминированный алгоритм.

Назовем алгоритм недетерминированным, если по крайней мере на одном шаге он допускает различные варианты продолжения работы и нет указаний, какой вариант выбрать.

Недетерминированный алгоритм называется эффективным, если его вычислительная сложность по любой из возможных трасс вычислений не превышает некоторого полинома от размера входной записи.

Таким образом, под задачей распознавания свойств будем понимать следующую задачу:

а) она требует ответа «ДА» или «НЕТ»;

b) решение может быть проверено (если задача его действительно имеет) с помощью полиномиального (эффективного) недетерминированного алгоритма.

Рассмотрим задачу о выполнимости системы логических формул в форме дизъюнктов.

Дизъюнкт – это формула вида

$$x_1^{\alpha_1} \vee x_2^{\alpha_2} \vee \ldots \vee x_k^{\alpha_k}, \qquad (34.1)$$

где $x_i^{\alpha_i} = \begin{cases} x_i, \text{ если } \alpha_i = 1, \\ \bar{x}_i, \text{ если } \alpha_i = 0. \end{cases}$

Примером дизъюнктов служат, например, следующие:

$$x_1 \vee x_2 \vee \bar{x}_3,$$

$$\bar{x}_1 \vee \bar{x}_2,$$

$$x_1 \vee x_2 \vee x_3$$

и т. д.

Говорим, что дизъюнкт выполним в интерпретации $I = \{x_1 = \beta_1, x_2 = \beta_2, \ldots, x_n = \beta_n\}$, $\beta_i = \{0, 1\}$, если его значением в I служит «1».

Пример. Дизъюнкт $x_1 \vee x_2 \vee \bar{x}_3$ выполним в интерпретации $I = \{x_1 = 1, x_2 = 0, x_3 = 1\}$, так как $1 \vee 0 \vee \bar{1} = 1$.

Ясно, что число выполняющих интерпретаций для дизъюнкта, содержащего n переменных, составляет 2^{n-1} и в одной интерпретации он не выполняется.

Задача «Выполнимость» требует дать ответ на вопрос: имеется ли интерпретация I, в которой выполняется каждый из дизъюнктов исходной системы.

Система дизъюнктов задачи «Выполнимость» называется также *конъюнктивной нормальной формой*. Среди всех конъюнктивных нормальных форм имеются формы с наименьшим числом дизъюнктов. Такие формы называются минимальными КНФ, эквивалентными данному множеству дизъюнктов.

Эта задача, очевидно, относится к числу задач распознавания. По смыслу вопроса она дает ответ – «ДА» или «НЕТ». Если I – некоторая выполняющая интерпретация, то легко проверить (т. е. реали-зовать эффективный алгоритм проверки), что I выполняет

каждый дизъюнкт системы. Эту проверку вполне можно осуществить с помощью детерминированного полиномиального алгоритма;

С. Кук доказал знаменитую теорему о том, что задача «Выполнимость» является универсальной в классе задач распознавания (этот класс получил короткое название NP – Nondeterministic Polynomial.

Итак, «Выполнимость» универсальна в NP. Однако, несмотря на более чем 100-летние усилия, не удалось найти для задачи «Выполнимость» эффективного детерминированного решающего алгоритма. Такая ситуация оказалась неслучайной. Сложность этой задачи оказывается большей, чем «пропускная способность» любого алгоритма, который был бы эффективным или детерминированным.

Остановимся на понятии пропускной способности канала связи, которое легко применить к вычислительной машине, решающей задачи.

Под пропускной способностью канала связи понимают величину

$$TH = \lim_{T\to\infty} \frac{I(T)}{T}, \qquad (34.2)$$

где $I(T)$ – количество информации, переданное по каналу связи за время T; T – время передачи.

Проще говоря, TH определяет, какое максимальное число бит информации в единицу времени может быть передано через канал связи. Очевидным образом следует признать, что эта величина имеет предел в силу конечности скорости света (во всяком случае, нам не известны в природе взаимодействия, протекающие с большей скоростью).

Количество информации, содержащейся в системе дизъюнктов. Пусть A и B – два дизъюнкта. В силу законов теории вероятностей можно записать

$$P(A \vee B) = P(A) + P(B) - P(AB) =$$
$$= P(A) + P(AB) + P(\bar{A}B) - P(AB) = P(A) + P(\bar{A}B), \qquad (34.3)$$

где $P(A)$ – вероятность того, что произвольная интерпретация I выполняет формулу A.

Далее, если A & B = false (ложь), то

$$P(A \vee B) = P(A) + P(B). \qquad (34.4)$$

Обобщением (34.3) для произвольного числа формул (дизъюнктов) служит

$$P(A \vee B \vee C \vee ... \vee W) = P(A) + P(\bar{A}B) + P(\bar{A}\bar{B}C) + ... + \\ + P(\bar{A}\bar{B}\bar{C}...W). \qquad (34.5)$$

Наконец, заметим, что вероятность выполнения формулы

$$A = x_{r_1}^{\alpha_1} \vee x_{r_2}^{\alpha_2} \vee ... \vee x_{r_k}^{\alpha_k}$$

есть

$$P(A) = 1 - 2^{-k}, \qquad (34.6)$$

а вероятность выполнения формулы

$$A = x_{r_1}^{\alpha_1} \,\&\, x_{r_2}^{\alpha_2} \,\&\, ... \,\&\, x_{r_k}^{\alpha_k}$$

есть

$$P(A) = 2^{-k}. \qquad (34.7)$$

Пример. Пусть дана формула $A = x_1 \vee x_2 \vee \bar{x}_3$. Тогда в силу (34.6) вероятность ее выполнения

$$P(A) = 1 - 2^{-3} = \frac{7}{8}.$$

Теорема. Пусть даны две формулы A и B и пусть $A \to B$ (B следует из A). Тогда

$$P(A) \le P(B).$$

Д о к а з а т е л ь с т в о. По условию $A \to B$ = true = 1, $P(true) = 1$ и true & $z = z$ для любой формулы z. Тогда, учтя, что $A \to B \equiv \bar{A} \vee B,$ получим

$$P(A \,\&\, A \to B) = P\left[A \,\&\, (\bar{A} \vee B)\right] = P(AB) \le P(B).$$

Но $P(A \,\&\, A \to B) = P(A),$ так как $A \to B$ – true. Поэтому $P(A) \le P(B)$.

Определение. Энтропией формулы A с n переменными называется величина

$$H(A) = -P(A)\log_2 P(A) - [1 - P(A)]\log_2[1 - P(A)]. \qquad (34.8)$$

При решении задач, связанных с угадыванием ответа, а задачи распознавания свойств относятся к таковым, естественно считать, что если $P(A) < P(B)$, то задача A сложнее задачи B, поскольку в случае задачи A придется сделать большее число проб, чтобы угадать решение. В то же время не всегда имеет место соотношение

$$H(A) > H(B),$$

когда

$$P(A) < P(B).$$

Например,

$$P(A) = 0{,}2 < P(B) = 0{,}4,\ \ H(A) = 0{,}72 < H(B) = 0{,}9709.$$

Таким образом, энтропия, определяемая стандартным образом по формуле (34.8), не отражает сложность задачи угадывания (энтропия больше, а перебор меньше). Это говорит о необходимости иного подхода к оценке сложности решения задач типа «Выполнимость».

При решении задач решатель работает с условиями задачи, часть которых задана явным образом, а часть скрыта в ее содержании.

Условие «принимается» во внимание, если решатель затрагивает один или более тактов своей работы на проверку истинности (ложности) условия.

Условие, логически не следующее из других условий, называется независимым от этих последних.

Для данного множества условий y_1, y_2, …, y_n условие α является простейшим, если его по форме можно представить как дизъюнкт, никакой собственный поддизъюнкт которого не содержится среди дизъюнктов формулы, логически эквивалентной y_1 & y_2 & … & y_n.

Пример. Рассмотрим формулу $A = x_1 x_2$. Ей эквивалентна формула $B = (x_1 \vee \bar{x}_2)\ \&\ x_2$. Дизъюнкт $x_1 \vee \bar{x}_2$ не является простейшим, так как его поддизъюнкт – x_1, входящий в эквивалентную формулу $A = x_1x_2$, состоящую из дизъюнктов x_1 и x_2.

Минимально необходимое и достаточное по числу условий множество Inf_A^{set} простейших независимых условий, записанных с помощью переменных задачи A, которое должен принять во

внимание решатель этой задачи, называется *инфологической сложностью задачи A*, а элементы Inf_A^{set} *– инфами*.

Итак, *инф* – это условие, в записи которого участвуют переменные задачи. Инфологическая сложность – мощность (число элементов) множества Inf_A^{set}, обозначаемая далее Inf_A.

Мы рассматриваем только алгоритмы, принимающие во внимание инфы, т. е. не игнорирующие инфы, так как при игнорировании инфа условие не берется в расчет, как если бы его не было, что нелегко, ибо инф есть необходимое и независимое от других условие. Игнорирование инфов наделяет алгоритм качеством оракула, что нами не допускается.

Алгоритм, не игнорирующий инфы, назовем «реалистическим». Не оговаривается, впрочем, как алгоритм получает инф и как он представлен.

Условие, эквивалентное множеству не менее двух инфов, называется *формой*. Принять форму во внимание – значит, заменить индивидуальный анализ инфов задачи распознаванием истинности некоторого предиката (условия срабатывания формы), который определяется способом записи (спецификации) задачи. Очевидным образом предполагается, что между формальным представлением задачи (спецификацией) и ее содержанием (множеством инфов) существует связь.

Рассматриваем только такие задачи, которые формулируются в терминах $n \geq 1$ однородных булевых переменных x_1, x_2, …, x_n и решением (выполняющим набором, интерпретацией) которых является множество индивидуальных значений для переменных x_i ($i = 1, \ldots, n$), удовлетворяющих условиям задачи, представленным в ее спецификации. Это отличает такого рода задачи от «ДА-НЕТ»-задач, хотя и не принципиально. Кроме того, в качестве решения принимается любой подходящий набор из числа выполняющих. Согласно теореме С. Кука, будем иметь дело только с задачами «Выполнимость», если противное не оговорено явно. Таким образом, слово «задача» понимается как «задача “Выполнимость”».

Далее мы будем использовать следующие предположения. Во-первых, никакой реалистический алгоритм не использует никаких иных способов работы с инфами, кроме как индивидуального анализа инфов и/или анализа форм. Во-вторых, для реалистических алгоритмов исключена возможность игнорирования инфов.

Подведем краткий итог. Мы показали, что энтропия не всегда правильно передает сложность задачи. Поэтому нами введено понятие инфа и инфологической сложности. Далее мы представим некую теорию, которая объясняет, почему задача «Выполнимость» не имеет эффективных решающих алгоритмов.

Лекция 35
ПОСТУЛАТЫ ТЕОРИИ ИНФОЛОГИЧЕСКОЙ СЛОЖНОСТИ

Цель. Изложить постулаты рассматриваемой теории вычислительной сложности.

Определение. Две задачи $З_1(x_1, x_2, ..., x_n)$ и $З_2(x_1, x_2, ..., x_n)$, записанные на одном и том же множестве переменных, эквивалентны, если выполняющие их наборы суть одно и то же множество либо обе задачи не выполнимы.

Определение. Выполнимая задача *B* называется ослаблением выполнимой задачи *A*, если часть (пустую или нет) или все условия задачи *A* заменить их следствиями (хотя бы одним следствием).

Замечание. Замена пустой части условий следствиями означает присоединение следствий.

ПОСТУЛАТ 1:

1a. Инфологическая сложность любой задачи, сформулированной как задача отыскания произвольного выполняющего набора на множестве булевых переменных, есть целое неотрицательное число.

1b. Инфологическая сложность любой задачи $A(x)$, формализованной на основе единственной переменной $x \in \{0, 1\}$, не выше 2.

1c. Всякая выполнимая ослабленная задача имеет инфологическую сложность, не превосходящую сложность исходной выполнимой задачи.

Замечание. Невыполнимая задача $З(x_1, x_2, ..., x_n)$ в общем случае может содержать невыполнимую часть в качестве собственного подмножества. Указать априори, какая эта часть, не решая задачу, нельзя. Однако невыполнимость этой части достаточна для того, чтобы игнорировать оставшуюся часть задачи и не принимать

во внимание содержащиеся в ней инфы. Для выполнимых задач это невозможно ввиду того, что инфы представляют не зависимые от других простейшие условия.

Следствие. Любая выполнимая задача содержит число инфов не меньшее, чем любая ее ослабленная задача.

Обоснование п. 1с постулата 1, видеть можно в том, что вероятность выполнимости ослабленной задачи не меньше вероятности выполнимости исходной задачи, а отбрасывание части дизъюнктов только увеличивает вероятность выполнимости оставшихся. Мы стоим на позиции, что чем больше вероятность выполнимости задачи, тем легче эта задача (менее сложна!).

Операция устранения литеры. Рассмотрим следующую систему дизъюнктов:

$$S = \begin{cases} x_1 \vee x_2, & \text{(35.1a)} \\ \overline{x}_1 \vee x_2 \vee \overline{x}_3, & \text{(35.1b)} \\ x_1 \vee x_3 \vee x_4, & \text{(35.1c)} \\ x_2 \vee \overline{x}_3 \vee \overline{x}_4, & \text{(35.1d)} \\ \overline{x}_3 \vee \overline{x}_4, & \text{(35.1e)} \\ \overline{x}_2 \vee \overline{x}_4. & \text{(35.1f)} \end{cases}$$

Операция *резолюции* (резолюционирования) двух дизъюнктов, содержащих контрарную пару литер, скажем, x_i и $\overline{x}_i$ состоит в порождении *резольвенты* – дизъюнкта, в котором присутствуют все литеры, входящие в родительские дизъюнкты, кроме x_i и $\overline{x}_i$.

Пример. Пусть $D_1 = x_1 \vee x_2$, $D_2 = \overline{x}_1 \vee x_2 \vee \overline{x}_3$. Их резольвентой будет дизъюнкт $R_{1,2} = x_2 \vee \overline{x}_3$ (x_1 и $\overline{x}_1$ устранены).

Можно показать, что резольвента $R_{1,2}$ есть логическое следствие из D_1 и D_2, т. е.

$$D_1 \,\&\, D_2 \to R_{1,2}.$$

Следовательно, в силу п. 1с постулата 1, присоединение резольвент к системе не увеличивает ее сложность (т. е. не приводит к увеличению числа инфов). Более того, отбрасывание части дизъюнктов при этом еще более упрощает задачу. Оказывается, можно полностью избавиться от какой-нибудь

литеры x_i ($\bar{x}_i$), перейдя от системы S к новой системе S', которая не содержит ни x_i, ни $\bar{x}_i$, но эквивалентна S в следующем смысле (в смысле H-эквивалентности): S и S' H-эквивалентны, если и только если:

а) S' записана с помощью собственного подмножества литер S;

б) любая выполняющая интерпретация для S содержит выполняющую интерпретацию для S', а любая выполняющая интерпретация для S' может быть достроена до выполняющей интерпретации для S.

Продемонстрируем все сказанное на системе S (35.1). Перейдем от S к S', устранив литеру x_1 ($\bar{x}_1$). Для этого включим сразу в S' все дизъюнкты из S, не содержащие x_1 ($\bar{x}_1$), т. е. (35.1d)–(35.1f). Для оставшихся дизъюнктов (35.1a)–(35.1c) найдем все возможные резольвенты по x_1 ($\bar{x}_1$). Так, $R_{a,b} = x_2 \vee \bar{x}_3$; $R_{b,c} = x_2 \vee x_3 \vee \bar{x}_3 \vee x_4$.

Присоединим в S' только не тавтологические резольвенты, т. е. не содержащие в своей записи фрагменты $\ldots x_k \vee \bar{x}_k \ldots$. Следовательно, $R_{b,c}$ не присоединяется. Другие дизъюнкты в S' не присоединяются. Итак, S' есть следующая система.

$$S' = \begin{array}{l} x_2 \vee \bar{x}_3, \\ x_2 \vee \bar{x}_3 \vee \bar{x}_4, \\ \bar{x}_3 \vee \bar{x}_4, \\ \bar{x}_2 \vee \bar{x}_4. \end{array} \qquad (35.2)$$

Можно доказать, что S к S' H-эквивалентны. Из наших рассмотрений следует, что сложность S' не выше сложности S.

ПОСТУЛАТ 2. Пусть задачи A и B H-эквивалентны и выполнимы, и Inf_{AB} – минимально необходимое и достаточное число инфов, принимаемых во внимание или игнорируемых при сведении задачи A к задаче B. Тогда

$$Inf_A = Inf_B + Inf_{AB}.$$

Следствие.

1. $Inf_{AC} = Inf_{AB} + Inf_{BC}$.

2. $Inf_{AB} = -Inf_{BA}$.

$$Inf_A = Inf_B + Inf_{AB},\ Inf_B = Inf_C + Inf_{BC},$$

откуда

$$Inf_A = Inf_C + Inf_{BC} + Inf_{AB} = Inf_C + Inf_{AC}.$$

Заметим, что отрицательное значение числа инфов для процедуры сведения означает, что инфы игнорируются. Ясно также, что эквивалентные преобразования одной задачи в другую не изменяют ее инфологическую сложность.

Смысл постулата 2 для наших рассмотрений таков. Пусть дана выполнимая задача $З(x_1, x_2, ..., x_n)$. Постулат 2 позволяет ограничить весь спектр возможных способов решения единственным: $З_1(x_1, x_2, ..., x_n)$ решается как последовательность переходов к *H*-эквивалентным задачам:

$$З_1(x_1, x_2, ..., x_n) \to З_2(x_1, x_2, ..., x_{n-1}) \to ... \to З_{n-1}(x_1). \qquad (35.3)$$

Последовательность (35.3) позволяет получить эффективное решение при условии, если исходная задача содержит полиномиальное (от n) число инфов. Если же задача содержит экспоненциальное (от n) число инфов, то никакие трансформации к *H*-эквивалентным задачам не позволяют принять во внимание только полиномиальное число инфов при допущении о реалистичности алгоритма. Именно этот факт и позволяет говорить о том, что при экспоненциальном нарастании числа инфов невозможен полиномиальный алгоритм, принимающий во внимание индивидуальные инфы, при условии, которое нами будет установлено на пропускную способность (алгоритма/ЭВМ).

Обозначим: TH – пропускная способность алгоритма, N – число выполняемых им шагов при решении задачи, I – число принятых во внимание при этом инфов.

$$TH = I / N. \qquad (35.4)$$

ПОСТУЛАТ 3. Какова бы ни была задача A на входе решающего алгоритма, имеет место ограничение пропускной способности алгоритма:

$$TH \le C_A < \infty, \qquad (35.5)$$

где C_A – фиксированная константа.

Смысл постулата 3 состоит в том, что никакой физический вычислитель не может иметь бесконечную скорость обработки информации. Данный постулат связан с физическими ограничениями теории относительности. Этот постулат дает нам следующее: нельзя переработать экспоненциально нарастающее число инфов при ограниченной пропускной способности за полиномиальное число тактов. В самом деле, при допущении обратного следовало бы, что

$$\frac{\exp(\beta n)}{\Phi(n)} = c \text{ (для любого } n),$$

где $\exp(n)$ – экспонента от n; $\Phi(n)$ – значение некоторого фиксированного полинома; β и c – положительные константы.

Однако используя элементарно доказываемое свойство

$$\lim_{n\to\infty} \frac{e^{\beta n}}{\alpha_k n^k + \alpha_{k-1} n^{k-1} + ... + \alpha_1 n + \alpha_0} = \infty$$

(для этого достаточно взять k производных от числителя и знаменателя), получим противоречие. Следовательно, при индивидуальном анализе инфов и экспоненциально нарастающем числе инфов алгоритм должен тратить экспоненциально растущее число тактов!

В дальнейшем мы вынуждены будем ввести в рассмотрение формы – вычисляемые условия, используемые для замены анализа индивидуальных инфов анализом группы инфов.

ПОСТУЛАТ 4. Инфологическая сложность выполнимой задачи «Выполнимость» $(x_1, x_2, ..., x_n)$ не ниже числа дизъюнктов в минимальной по числу дизъюнктов конъюнктивной нормальной форме, эквивалентной этой задаче «Выполнимость».

В самом деле, все дизъюнкты минимальной КНФ суть инфы по определению. Формулы с меньшим числом инфов для задачи «Выполнимость» не существует.

Лекция 36
ОБОСНОВАНИЕ НЕСУЩЕСТВОВАНИЯ ЭФФЕКТИВНОГО АЛГОРИТМА В КЛАССЕ ЗАДАЧ РАСПОЗНАВАНИЯ СВОЙСТВ

Цель. Обоснование экспоненциальной сложности задачи «Выполнимость» на основе рассматриваемой теории инфологической сложности.

36.1. Инфы

Перейдем к рассмотрению важного технического результата. Некоторые универсальные в классе NP задачи формализуются через условие вида

$$a_1x_1 + a_2x_2 + ... + a_nx_n = c, \qquad (36.1)$$

где a_i, c – целые и положительные; $x_i \in \{0,1\}$.

В частности, такова задача об упаковки в контейнер (ЗУК). Частный вид (36.1) входит в задачу о минимальном покрытии 0,1 – матрицы. Сформулируем проблемы: «Можно ли ЗУК свести к эквивалентной задаче «Выполнимость», записанной на том же множестве переменных, так, чтобы для любого n размеры задачи «Выполнимость» (коротко ВЫП) были ограничены некоторым фиксированным полиномом от n?»

Ответ на поставленную проблему дает теорема.

Теорема. Невозможно сведение $\text{ЗУК}(x_1, x_2, ..., x_n)$ к $\text{ВЫП}(x_1, x_2, ..., x_n)$ с сохранением свойства эквивалентности множества решений, при котором для каждого n размеры $\text{ВЫП}(x_1, x_2, ..., x_n)$ были бы ограничены некоторым фиксированным полиномом от n [26].

В частности, эквивалентная ВЫП для условия

$$x_1 + x_2 + ... + x_n = (n+1)/2, \qquad (36.2)$$

где n – четное, требует минимальной КНФ с $O(2^{n/2})$ дизъюнктами, т. е. размеры ВЫП растут экспоненциально.

Этот технический результат дает нам следующее.

Покажем, как для (36.2) построить *H*-эквивалентную КНФ полиномиальных от n размеров. Этот общий способ продемонстрируем для уравнения

$$\sum_{i=1}^{n} x_i = k.$$

Будем использовать дополнительные булевские переменные y_s, z_r, …, w_t ($s, r, t \in k+1, \ldots, n$), имея в виду $k \le n$:

$$x_1 + y_{k+1} + y_{k+2} + \ldots + y_n = 1;$$

$$x_2 + z_{k+1} + z_{k+2} + \ldots + z_n = 1;$$

$$\ldots$$

$$x_k + w_{k+1} + w_{k+2} + \ldots + w_n = 1;$$

$$y_{k+1} + z_{k+1} + \ldots + w_{k+1} \le 1;$$

$$y_{k+2} + z_{k+2} + \ldots + w_{k+2} \le 1;$$

$$\ldots$$

$$y_n + z_n + \ldots + w_n \le 1.$$

В приведенной системе нет основных переменных x_{k+1}, x_{k+2}, …, x_n. Они выражаются через дополнительные переменные следующим образом:

$$\begin{gathered} x_{k+1} = y_{k+1} \vee z_{k+1} \vee \ldots \vee w_{k+1}; \\ x_{k+2} = y_{k+2} \vee z_{k+2} \vee \ldots \vee w_{k+2}; \\ \ldots \\ x_n = y_n \vee z_n \vee \ldots \vee w_n. \end{gathered} \qquad (36.3)$$

Нетрудно показать, что полученная расширенная система потребует $k(1+C_n^2)+(n-k)C_{n-k}^2$ дизъюнктов. В самом деле, условие

$$x + y + z + \ldots + w = 1$$

передается системой

$$x \vee y \vee z \vee \ldots \vee w;$$

$$\overline{x} \vee \overline{y};$$

$$\overline{x} \vee \overline{z};$$

$$\dots$$

$$\overline{z} \vee \overline{w}.$$

Условие

$$x + y + z + \dots + w \leq 1$$

передается такой же системой, но без дизъюнкта $x \vee y \vee z \vee \dots \vee w$.

Итак, полученную полиномиальную по размерам КНФ для (36.2) назовем *H*-КНФ, а первоначальную минимальную КНФ экспоненциальных размеров, назовем Э-минКНФ.

Имеет место утверждение: Э-минКНФ и *H*-КНФ суть *H*-эквивалентные задачи и

$$Inf_{\text{Н-КНФ}} \geq Inf_{\text{Э-минКНФ}}.$$

Д о к а з а т е л ь с т в о. *H*-эквивалентность следует непосредственно из определения: любое решение *H*-КНФ содержит решение Э-минКНФ. Наоборот, любое решение Э-минКНФ может быть достроено до решения Н-КНФ в силу (36.2).

Мы видим, что полиномиальная по размеру записи КНФ может содержать экспоненциальное число инфов. Поэтому любой реалистический алгоритм, который выполняет учет индивидуальных инфов, потратит экспоненциально растущее число шагов на решение задачи. Таким образом, только использование форм может дать надежду на преодоление экспоненциальной сложности. Однако мы убедимся ниже, что это невозможно.

Итак, сформулируем основные идеи проведенного нами рассмотрения. Мы показали, что имеются полиномиальные от n (n – число переменных) задачи ВЫП, локализующие в себе экспоненциальное число инфов. Имеется *H*-эквивалентная КНФ для таких задач ВЫП, которая содержит экспоненциальное число дизъюнктов. Таким образом, в силу постулата об ограниченной пропускной способности алгоритма нужно затратить экспоненциальное число тактов, чтобы принять во внимание каждый инф индивидуально. Следовательно, чтобы число тактов вычислителя было полиномиальным, нужно либо игнорировать инфы (чего не допускают реалистические алгоритмы), либо принимать во внимание целые блоки инфов, т. е. формы. Анализ этой возможности мы проведем в следующем параграфе.

36.2. Формы

Форма F суть конструкция «если …, то» или $F \equiv P \to f$, где P – условие срабатывания формы; f – алгоритм, конструирующий любые наборы, представляющие решения задачи, и удовлетворяющий P и только их. Тогда можно утверждать, что $P \to$ ВЫП (т. е. задача «Выполнимость» есть следствие P) и, следовательно, по п. 1с постулата 1, ВЫП есть ослабление P. Однако тогда P содержит экспоненциальное число инфов, если ВЫП содержит экспоненциальное число инфов. Поэтому вычисление P потребует экспоненциального времени. С другой стороны, остается возможность не вычислять P, а запомнить P, т. е. не решать задачу, а «вспомнить», что при таком-то P решение ВЫП было таким-то. Этот вариант не проходит, поскольку можно зашифровать любую ВЫП с помощью двоичного ключа, так что число ключей окажется значительно больше 2^n. Следовательно, алгоритм проверки P потребует запомнить все эти ключи; ясно, что можно удержать в памяти лишь фиксированное количество ключей. Остановимся более подробно на этом вопросе.

Любая задача ВЫП записывается с помощью знаков: «(», «)», «~», «∨» и цифр 0, 1, …, 9. Всех значков 14, поэтому каждый значок можно закодировать четырьмя разрядами. Выбор системы кодировки для нас не принципиален. Таким образом, задача ВЫП, состоящая из n символов будет закодирована двоичным числом N_1. Пусть секретный ключ не известен заранее и равен N_2. Тогда с числами N_1 и N_2 можно сопоставить число

$$Z = \frac{1}{2}(N_1^2 + 2N_1N_2 + N_2^2 + 3N_1 + N_2).$$

Это число Z уникально, и оно уникальным образом определяет N_1 и N_2. Значит, если секретный ключ составляет также $4n$ разрядов, то всех ключей будет экспонента: 2^{4n}. Следовательно, даже для одной и той же задачи необходимо держать экспоненту констант. Числа Z распределяются хаотически. Если мы захотим найти закон их появления для некоторой конкретной ВЫП, то число инфов, необходимых для описания такого закона, будет не меньше, чем число инфов в ВЫП, так как найти этот закон значит найти решение ВЫП.

Итак, формы не могут решить проблемы эффективности алгоритма для решения задачи «Выполнимость». Это и есть главный результат. Введенные нами постулаты заимствованы так или иначе из теории систем. Они характеризуют то понимание сложности, которое интуитивно используется разработчиками алгоритмов. Представляется, что проблема сложности вычислений лишена практического смысла вне физики. Точные алгоритмы, не игнорирующие инфы, могут быть абсолютно неприемлемыми для практики. Следовательно, далее инфы должны быть игнорированы, чтобы преодолеть ограничения размерности. Одним из вариантов на этом пути является угадывание решения, базирующееся, в частности,
на механизме эвристических методов.

ЛИТЕРАТУРА

1. Жевняк, Р. М. Высшая математика / Р. М. Жевняк, А. А. Карпук. – Минск: Выш. шк., 1988. – Ч. V. – 284 с.

2. Пугачев, В. С. Теория вероятностей и математическая статистика / В. С. Пугачев. 2002. – 400 с.

3. Прохоров, Ю. В. Теория вероятностей. Основные понятия. Предельные теоремы. Случайные процессы / Ю. В. Прохоров, Ю. А. Розанов. – М.: Наука, 1987. – 397 с.

4. Гмурман, В. Е. Теория вероятностей и математическая статистика / В. Е. Гмурман. – М.: Высш. шк., 1998. – 238 с.

5. Вентцель, Е. С. Теория вероятностей / Е. С. Вентцель. – М.: Высш. шк., 2006. – 576 с.

6. Микулик, Н. А. Решение технических задач по теории вероятностей и математической статистике / Н. А. Микулик, Г. Н. Рейзина. – Минск: Выш. шк., 1991. – 148 с.

7. Колемаев, В. А. Теория вероятностей и математическая статистика / В. А. Колемаев, О. В. Староверов, В. Б. Турундаевский. – М.: Высш. шк., 1991. – 398 с.

8. Стратонович, Р. Л. Теория информации / Р. Л. Стратонович. – М.: Сов. радио, 1975. – 424 с.

9. Роджерс, Х. Теория рекурсивных функций и эффективная вычислимость / Х. Роджерс. – М.: Мир, 1972. – 625 с.

10. Колесников, В. Л. Математические основы компьютерного моделирования химико-технологических систем / В. Л. Колесников. – Минск: БГТУ, 2003. – 308 с.

11. Львовский, Е. Н. Статистические методы построения эмпирических формул / Е. Н. Львовский. – М.: Высш. шк., 1988. – 238 с.

12. Ллойд, Э. Справочник по прикладной статистике: в 2 т. / под ред. Э. Ллойда. – М.: Финансы и статистика, 1990.

13. Зыков, А. А. Основы теории графов / А. А. Зыков. – М.: Наука, 1987. – 380 с.

14. Кофман, А. Введение в прикладную комбинаторику / А. Кофман. – М.: Радио и связь, 1982. – 431 с.

15. Арженeнко, А. Ю. Оптимальные бинарные вопросники / А. Ю. Арженeнко, Б. Н. Чугаев. – М.: Энергоатомидат, 1989. – 128 с.

16. Пантелеев, А. В. Теория управления в примерах и задачах / А. В. Пантелеев, А. С. Бортаковский. – М.: Высш. шк., 2003. – 584 с.

17. Пискунов, Н. С. Дифференциальное и интегральное исчисления для втузов: в 2 т. / Н. С. Пискунов. – М.: Интеграл-Пресс, 2001. – 960 с.

18. Жевняк, Р. М. Высшая математика / Р. М. Жевняк, А. А. Карпук. – Минск: Выш. шк., 1985. – Ч. III. – 210 с.

19. Иващенко, Н. Н. Автоматическое регулирование. Теория и элементы систем / Н. Н. Иващенко. – М.: Машиностроение, 1985. – 736 с.

20. Математические основы теории автоматического регулирования / В. А. Иванов [и др.]. – М.: Высш. шк., 1977. – 822 с.

21. Сергиенко, А. Б. Цифровая обработка сигналов / А. Б. Сергиенко. – СПб.: Питер, 2005. – 604 с.

22. Шеперд, Деван. Освой самостоятельно XML / Деван Шеперд. – М.: Вильямс, 2002. – 432 с.

23. Пападимитриу, Х. Комбинаторная оптимизация / Х. Пападимитриу, К. Стайглиц. – М.: Мир, 1985. – 510 с.

24. Герман, О. В. Введение в теорию экспертных систем и обработку знаний / О. В. Герман. – Минск: ДизайнПро, 1995. – 256 с.

25. Гэри, М. Вычислительные машины и труднорешаемые задачи / М. Гэри, Д. Джонсон. – М.: Мир, 1982. – 356 с.

26. Герман, О. В. Об одной концепции вычислительной сложности / О. В. Герман, Гончарова Е. Н. // Вестник ставропольского государственного университета. – 2006. – Вып. 40. – С. 16–24.

Дополнительная литература

1. Душин, В. К. Теоретические основы информационных процессов и систем: учебник / В. К. Душин. – М.: Дашков и Ко, 2004. – 348 с.

2. Сэломон, Д. Сжатие данных, изображения и звука / Д. Сэломон. – М.: ТехноСфера, 2004. – 364 с.

3. Крисевич, Р. Е. Сжатие и поиск информации / Р. Е. Крисевич. – М.: Радио и связь, 1989. – 89 с.

4. Цымбал, В. П. Задачник по теории информации и кодированию / В. П. Цымбал. – М.: Высш. шк. 1976. – 275 с.

5. Трауб, Дж. Информация, неопределенность, сложность / Дж. Трауб. – М.: Мир, 1988. – 178 с.

6. Ященко, В. В. Введение в криптографию / В. В. Ященко. – М.: МЦНМО, 2001. – 286 с.

7. Миркин, Б. Г. Анализ качественных признаков и структур / Б. Г. Миркин. – М.: Статистика, 1980. – 308 с.

8. Железнов, И. Г. Сложные технические системы / И. Г. Железнов. – М.: Высш. шк., 1984. – 118 с.

9. Ульман, Дж. Основы систем баз данных / Дж. Ульман. – М.: Финансы и статистика, 1983. – 332 с.

10. Горелова, В. Л. Основы прогнозирования систем / В. Л. Горелова, Е. Н. Мельникова. – М.: Высш. шк. 1986. – 287 с.

ОГЛАВЛЕНИЕ