Лекции по теории информации

Н.К. Верещагин

Оглавление

1 Информация по Хартли **7**
1.1 Игра в 10 вопросов 8
1.2 Упорядочивание n чисел: верхняя и нижняя оценки 9
1.3 Упорядочение 5 различных чисел с помощью 7 сравнений . 10
1.4 Поиск фальшивой монетки из 81 за 4 взвешивания 12
1.5 Поиск фальшивой монетки из 12 за 3 взвешивания 13
1.6 Цена информации 15
1.7 Задачи для самостоятельной работы 18

2 Логика знания **23**
2.1 Сообщения и увеличение знаний 24
2.2 Карточки с цифрами 26
2.3 Задача о шляпах. 27
2.4 Задачи для самостоятельной работы 27
2.5 Отступление: парадокс неожиданной контрольной 29

3 Коммуникационная сложность **33**
3.1 Среднее арифметическое и медиана мультимножества . . . 35
3.2 Предикат равенства 36
3.3 Метод трудных множеств и метод размера прямоугольников 38
3.4 Метод ранга матрицы 40
3.5 Вероятностные протоколы 40

4 Энтропия Шеннона **43**
4.1 Определение 43
4.2 Коды 44
4.2.1 Некоторые известные коды 49

4.3 Коммуникационная сложность в среднем и энтропия Шеннона . . . 54
4.4 Неравенство Макмиллана . . . 55
4.5 Энтропия пары случайных величин . . . 56
4.6 Условная энтропия . . . 58
4.7 Независимость и энтропия . . . 61
4.8 «Релятивизация» и информационные неравенства . . . 63
4.9 Задачи для самостоятельной работы . . . 68

5 Кодирование текстов 71
5.1 Учет частот букв . . . 72
5.1.1 Кодирование слова его номером в алфавитном порядке . . . 73
5.1.2 Использование кодов Шеннона–Фано, арифметического и кода Хаффмана . . . 73
5.2 Сбалансированные слова . . . 75
5.3 Учёт частот пар, троек и т.д. . . . 77
5.3.1 Разбиение на блоки . . . 82
5.3.2 Стационарные источники. . . . 83
5.4 Неинъективные кодирования . . . 84
5.4.1 Теорема Шеннона о бесшумном канале. . . . 84
5.4.2 Марковские цепи . . . 87
5.5 Теорема Вольфа–Слепяна . . . 90
5.6 Каналы с помехами . . . 95
5.6.1 Пропускная способность канала с помехами . . . 95
5.6.2 Использование каналов с помехами . . . 96

6 Предсказания и игры 105

7 Колмогоровская сложность 113
7.1 Что такое колмогоровская сложность? . . . 113
7.2 Оптимальные способы описания . . . 115
7.3 Сложность и случайность . . . 118
7.4 Невычислимость KS и парадокс Берри . . . 120
7.5 Перечислимость сверху колмогоровской сложности . . . 121
7.6 Сложность и информация . . . 123
7.7 Сложность пары слов . . . 125
7.8 Условная колмогоровская сложность . . . 129

7.8.1 Количество информации 138
7.9 Сложность и энтропия 139
7.9.1 Колмогоровская сложность и энтропия частот . . . 139
7.9.2 Математическое ожидание сложности 140
7.10 Применения колмогоровской сложности 142
7.10.1 Бесконечность множества простых чисел 142
7.10.2 Перенос информации по ленте 143
7.10.3 Бритва Оккама 146

8 Дополнительные задачи **149**

Понятие информации не является строго определенным в математическом смысле. То же самое относится и к часто употребляемым выражениям типа «преобразование информации», «количество информации» и так далее. Скорее у нас имеется некоторая интуиция о том, какие вопросы относятся к теории информации, а какие — нет. В настоящих лекциях исследуются некоторые задачи, которые по мнению автора имеют явный информационный «привкус».

Что касается строгих определений, то больше повезло понятию количества информации — для его уточнения имеется несколько математических моделей. Наиболее известными из них являются «информация по Хартли», энтропии Шеннона и колмогоровская сложность. В своей работе «Три подхода к определению понятия количества информации», в которой сравниваются эти три понятия (и впервые определяется последнее из них) Колмогоров называл подход Хартли к определению количества информации «комбинаторным». В двух словах он состоит в следующем: для того, чтобы идентифицировать любой неизвестный элемент из известного конечного множества A необходимо и достаточно $\log_2 |A|$ бит информации (игнорируя ошибки округления). Энтропия Шеннона является более сложным понятием. Она определена для случайных величин и отражает «неопределенность» случайной величины. Наконец, колмогоровская сложность измеряет количество информации в произвольном тексте (слове). В наших лекциях мы расскажем обо всех трех понятиях (и не только), и начнем мы с самого простого — информации по Хартли.

Глава 1

Информация по Хартли

Пусть имеется (известное нам) конечное множество A, элементы которого мы называем *исходами*. Чтобы идентифицировать любой элемент множества A нам необходимо и достаточно $\log_2 |A|$ бит информации (игнорируя ошибку округления). По этой причине говорят, что исходы из множества A имеют $\log_2 |A|$ бит информации (по Хартли). В дальнейшем двойку в основании логарифма мы писать не будем — по умолчанию все логарифмы берутся по основанию 2.

Пусть нам было известно, что $x \in A$, а затем нам дополнительно сообщили, что $x \in B$, где B некоторое подмножество A. Теперь нам необходимо и достаточно $\log |B|$ бит, чтобы идентифицировать x. Таким образом, мы можем представлять себе, что нам сообщили

$$\log |A| - \log |B| = \log |A|/|B|$$

бит информации.

Пример 1. Пусть, нам нужно узнать неизвестное упорядочение пятиэлементного множества $\{1, 2, 3, 4, 5\}$. Множество A состоит из всех перестановок этого множества и его мощность равна 5!. Допустим нам сообщили, что $1 > 2$ или $3 > 4$. Сколько бит в этой информации? Множество B состоит из всех перестановок, в которых $1 > 2$ или $3 > 4$. Нетрудно понять, что количество таких перестановок равно трём четвертям от общего количества перестановок, а значит в этой информации $\log(4/3)$ бит.

Решения обсуждаемых ниже задач можно излагать как на теоретико-информационном языке, так и на на чисто комбинаторном языке. Реше-

ния первого вида более интуитивно ясные (хотя и менее строгие). Поэтому мы, как правило, будем предпочитать теоретико-информационный язык.

Количество информации аддитивно в следующем смысле. Пусть даны три множества $C \subset B \subset A$ и мы сначала знаем только, что неизвестный исход x принадлежит A. Допустим, что нам сначала сказали, что $x \in B$, а затем дополнительно сказали, что $x \in C$. В первый раз нам сообщили $\log|A|/|B|$ бит информации, а во второй — $\log|B|/|C|$ бит. Сумма этих двух чисел равна $\log|A|/|C|$, то есть как раз столько, сколько мы в целом узнали.

То же самое справедливо и для любого количества сообщений: Пусть изначально множество возможных исходов равно A. Пусть нам сделали k сообщений о неизвестном исходе из A, сообщив в i-ый раз n_i бит информации. Обозначим через B множество всех исходов, согласованных с полученными сообщениями. Тогда $n_1 + \cdots + n_k = \log|A|/|B|$.

В предлагаемых задачах нам нужно найти неизвестный объект x из данного множества A, при этом разрешается проводить тесты определенного вида с возможными результатами ДА/НЕТ. Любой такой тест задается некоторым подмножеством B множества A и имеет два результата: положительный, $x \in B$, и отрицательный, $x \notin B$. Положительный результат приносит нам $\log|A|/|B|$ бит информации, а отрицательный приносит $\log|A|/|\bar{B}|$ бит (через $\bar{B}$ мы обозначаем дополнение множества B, то есть, разность A и B).

1.1 Игра в 10 вопросов

Задача 1. Имеется неизвестное число от 1 до 1000. Разрешается задавать любый вопросы с ответами ДА/НЕТ. Сколько необходимо и достаточно вопросов для отгадывания числа?

Читатель, привыкший к математической строгости, может законно возмутиться — в задаче используются неопределенные понятия «задать вопрос», «неизвестное число», «отгадать число» и так далее. Тем не менее, мы предпочитаем не определять их формально, опираясь на интуицию.

Во-первых, очевидно, что достаточно 10 вопросов, поскольку неизвестное число можно записать 10 двоичными цифрами и спросить про каждую из них. Осталось доказать, что 9 вопросов не хватит.

Сначала мы изложим совсем простое комбинаторное доказательство, а потом (немного более сложное, но зато интуитивно более ясное) теоретико-информационное доказательство.

Комбинаторное доказательство. Любой алгоритм, который решает задачу, задав 9 вопросов, определяет некоторую сюръективную функцию из множества всех двоичных последовательностей длины 9 в множество чисел от 1 до 1000 (ее значение на данной последовательности $a_1 \dots a_9$ равно результату, выданному алгоритмом, если на заданные вопросы он получил ответы $a_1 \dots a_9$). Поскольку такой функции не существует (по соображениям мощности), то и такого алгоритма не существует.

Теоретико-информационное доказательство. Любой вопрос с двоичным ответом в худшем случае приносит нам не более 1 бита информации. В самом деле, пусть перед тем, как мы задали вопрос, нам было известно, что $x \in A$, и мы спросили принадлежит ли x некоторому подмножеству $B \subset A$. Очевидно, что либо $\log |A|/|B| \leqslant 1$, либо $\log |A|/|\bar{B}| \leqslant 1$ (иными словами, либо $|B| \geqslant |A|/2$, либо $|\bar{B}| \geqslant |A|/2$.) Предположим теперь, что на каждый вопрос нам дают наихудший для нас ответ, сообщая не более 1 бита информации. Из аддитивности следует, что после девяти вопросов и ответов нам в сообщат не более 9 бит, а нужно нужно $\log 1000 > 9$ бит, поскольку в конце мы должны узнать, чему равно задуманное число.

1.2 Упорядочивание n чисел: верхняя и нижняя оценки

Задача 2. Дано n различных по весу камешков $a_1, \dots, a_n$. Надо их упорядочить по весу, используя наименьшее количество взвешиваний (за одно взвешивание можно про любые два камешка a_i, a_j выяснить, какой из них легче).

Решение. Мы докажем, что необходимо и достаточно произвести $\log n! + O(n)$ взвешиваний. Точнее, мы установим, что любой алгоритм в худшем случае выполняет $\lceil \log n! \rceil$ взвешиваний и существует алгоритм, который упорядочивает камешки за $\lfloor \log n! \rfloor + n - 1$ взвешиваний. Точное значение наименьшего количества сравнений при больших n неизвестно.

Нижняя оценка. Всего существует $n!$ разных упорядочиваний n камешков. Поэтому необходимо $\lceil \log n! \rceil$ взвешиваний.

Верхняя оценка. Чтобы упорядочить камешки за $\log n! + n - 1$ взвеши-

ваний, можно применить алгоритм последовательной вставки. А именно, пусть у нас уже имеется упорядоченный массив из k камешков. Чтобы вставить в него k+1-ый камешек, применим бинарный поиск: сравним его со средним камешком из массива, затем сравним со средним камешком из верхней или нижней половины массива (в зависимости от исхода первого взвешивания) и так далее. Для этого нам понадобится $\lceil \log(k+1) \rceil$ взвешиваний (например, при $k = 2, 3$ будет 2 взвешивания, а при $k = 4, 5, 6, 7$ — 3 взвешивания).

Начнем с упорядоченного массива из одного камешка и будем последовательно вставлять в него второй, третий, и так далее. Общее количество сравнений будет

$$\lceil \log 2 \rceil + \lceil \log 3 \rceil + \cdots + \lceil \log n \rceil \leqslant \log n! + n - 2.$$

Замечание 1. Формула Стирлинга учит, что $n! \sim \sqrt{2\pi n} \cdot (n/e)^n$, а значит $\log n! = n \log n + O(n)$. Таким образом, необходимо и достаточно сделать $n \log n + O(n)$ взвешиваний.

При всех n, меньших 5, полученные верхняя и нижняя оценки совпадают. Действительно:

$$\begin{aligned} \lceil \log 2! \rceil = 1, &\qquad \lceil \log 2 \rceil = 1, \\ \lceil \log 3! \rceil = 3, &\qquad \lceil \log 2 \rceil + \lceil \log 3 \rceil = 3, \\ \lceil \log 4! \rceil = 5, &\qquad \lceil \log 2 \rceil + \lceil \log 3 \rceil + \lceil \log 4 \rceil = 5. \end{aligned}$$

Однако, уже для $n = 5$ они различаются на единицу:

$$\lceil \log 5! \rceil = 7, \qquad \lceil \log 2 \rceil + \lceil \log 3 \rceil + \lceil \log 4 \rceil + \lceil \log 5 \rceil = 8.$$

Возникает естественное желание узнать, можно ли упорядочить 5 камешков с помощью 7 взвешиваний.

1.3 Упорядочение 5 различных чисел с помощью 7 сравнений

Задача 3. Дано 5 различных по весу камешков $a_1, \ldots, a_5$. Можно ли их упорядочить по весу с помощью 7 взвешиваний?

Решение. Общее количество исходов в этой задаче равно $5! = 120$. Каждое сравнение дает два возможных результата. Поэтому 7 сравнений дают $2^7 = 128$ возможных результатов, что хоть и больше, чем 120, но совсем ненамного. Поэтому каждое взвешивание должно делить множество всех исходов на почти равные части (в информационных терминах: при обоих исходах взвешивание должно давать примерно 1 бит информации о неизвестном порядке). Мы будем пользоваться этим обстоятельством для отсечения заведомо неподходящих вариантов сравнений. В результате мы либо установим, что искомого алгоритма нет, либо найдем его.

При первом сравнении у нас в сущности нет выбора, поскольку все варианты симметричны. Сравниваем, скажем a_1 и a_2. Без ограничения общности, пусть оказалось $a_1 < a_2$. Остается 60 возможных исходов и $2^6 = 64$ возможных результатов оставшихся 6 взвешиваний.

Теперь у нас имеется 2 принципиально разных варианта продолжения: сравнить два новых камешка или один новый с одним старым. Покажем, что второй вариант не годится. Пусть, например, мы сравнили a_1 с a_3. Такое сравнение разделит множество из оставшихся возможными 60 порядков на 20 и 40. Действительно, если $a_3 < a_1 < a_2$, то имеется ровно 20 возможных исходов (четырьмя способами можно вставить a_4 в этот упорядоченный массив и для каждого из этих способов можно 5 способами вставить a_5 в полученный упорядоченный массив). Отсюда следует, что при другом результате сравнения ($a_1 < a_3$) остается $60 - 20 = 40$ исходов. Поскольку $40 > 2^5$, с помощью оставшихся 5 взвешиваний невозможно будет найти порядок.

Итак, во втором взвешивании мы должны сравнить два новых камешка, например, a_3 и a_4. Такое сравнение разделит оставшиеся исходы поровну на 30 и 30. Без ограничения общности, пусть оказалось $a_3 < a_4$.

Каким должно быть третье взвешивание? Докажем, что пятый камешек не может в нем участвовать. Допустим, что третьим взвешиванием мы сравним новый камешек a_5 с одним из предыдущих. По симметрии, неважно, с чем его сравнить. Пусть, скажем, мы сравнили a_1, a_5. Такое сравнение разделит оставшиеся 30 исходов на 10 и 20. Действительно, существует 10 порядков, для которых $a_5 < a_1 < a_2$ и $a_3 < a_4$ (слить два упорядоченных массива из соответственно k, l элементов в один упорядоченный массив из $k+l$ элементов можно $(k+1)+(k+2)+\cdots+(k+l) = C_{k+l}^{l}$ способами). Поэтому слить массивы $a_5 < a_1 < a_2$ и $a_3 < a_4$ в один можно $C_{2+3}^{2} = 10$ способами. Поскольку $20 > 16$, такое сравнение нам не подходит.

Таким образом, при третьем взвешивании мы должны сравнить между собой какие-то из a_1, a_2, a_3, a_4. Есть два принципиально разных варианта: сравнить между собой два тяжелых (или легких) камешка и сравнить между собой тяжелый и легкий камешки. Второй вариант очевидно плох, поскольку разделяет множества из оставшихся 30 исходов на 5 и 25. Действительно, если оказалось, что тяжелый камешек легче легкого, то мы получаем полностью упорядоченный массив из $a_1, \ldots, a_4$, в который можно вставить a_5 пятью способами.

Итак, в третьем взвешивании мы должны сравнить между собой два тяжелых или два лёгких камешка. Пусть, например, мы сравниваем a_1, a_3. Без ограничения общности, пусть $a_1 < a_3$. Осталось 15 возможных исходов и 4 взвешивания.

Докажем, что в четвертом взвешивании необходимо сравнивать a_5. Действительно, при сравнении a_2 с a_3 или a_4 нам может повезти и мы узнаем полностью, как упорядочены $a_1, \ldots a_4$ (при сравнении a_2 с a_3 хороший для нас результат $a_2 < a_3$, а при сравнении a_2 с a_4 хороший для нас результат $a_2 > a_4$). В этом случае останется 5 возможных исходов. Значит в случае невезения их останется $10 > 8$.

С чем нужно сравнивать a_5? Ясно, что не с самым лёгким a_1 (такое сравнение разделит оставшиеся 15 исходов на 3 и 12) и не с тяжёлыми a_2, a_4 (поскольку в случае везения останется соответственно 6 и 4 возможных исхода, а при невезении — 9 и 11). Итак, методом исключения мы убедились, что в четвертом взвешивании мы должны сравнить a_3 и a_5. Нетрудно убедиться, что это сравнение разделит оставшиеся 15 исходов на 8 и 7 и в каждом из двух результатов взвешивания можно с помощью оставшихся 3 сравнений найти порядок на $a_1, \ldots, a_5$ целиком. Поэтому ответ в задаче положительный.

1.4 Поиск фальшивой монетки из 81 за 4 взвешивания

Задача 4. Дано 81 монеток, из которых ровно одна фальшивая, которая легче всех настоящих (настоящие монеты имеют одинаковый вес). Сколько нужно взвешиваний, чтобы на чашечных весах найти фальшивую монетку. (На чашечных весах можно сравнивать по весу любые непересекающиеся группы монеток. Результатов сравнения 3 — равен-

ство, больше и меньше.)

Решение. Нижняя оценка: необходимо не менее 4 взвешиваний. Действительно, на этот раз каждый тест имеет 3 исхода (больше, меньше и равно). Поэтому в худшем случае один тест приносит не более $\log 3$ бит информации, а нам нужно $\log 81 = 4 \log 3$ бит. Рассуждая, так же, как и в первой задаче (при худших для нас ответах текущее множество X после каждого теста уменьшается не более, чем втрое) убеждаемся, что необходимо 4 взвешивания.

Верхняя оценка: 4 взвешиваний достаточно. Делим монетки на 3 равные группы (по 27 монеток) и сравниваем первые 2 группы. В случае равенства их весов, фальшивая монета находится в третьей группе. В случае неравенства — она в более легкой группе. Далее мы применяем эту стратегию рекурсивно еще 3 раза.

1.5 Поиск фальшивой монетки из 12 за 3 взвешивания

Задача 5. Дано 12 монеток, из которых ровно одна фальшивая, которая может быть как легче, так и тяжелее настоящих. Надо за 3 взвешивания на чашечных весах найти фальшивую монетку и узнать, легче или тяжелее она остальных.

Решение. Общее количество исходов в этой задаче равно 24 (любая из 12 монеток может быть фальшивой, причем она может быть легче или тяжелее). Каждое взвешивание дает три возможных результата (равенство, больше или меньше), поэтому 3 взвешивания дают 27 возможных результатов, что хоть и больше, чем 24, но совсем ненамного. Поэтому каждое взвешивание должно делить множество всех исходов на три почти равных части (в информационных терминах: каждое взвешивание при любом его результате должно давать примерно $\log 3$ битов информации). Мы будем пользоваться этим обстоятельством для отсечения заведомо неподходящих вариантов взвешивания.

Прежде всего заметим, что глупо сравнивать разные по количеству группы монеток. Действительно, можно предполагать, что вес фальшивой монетки примерно равен весу настоящей. При этом предположении группа в которой больше монеток всегда тяжелее группы, в которой их меньше, поэтому такие взвешивания вообще не дают информации.

Итак, первое взвешивание должно быть сравнением двух групп из k монеток (для некоторого k). Нетрудно понять, что k должно быть равно 4. В самом деле при равенстве весов групп остается $2(12-2k)$ возможных исходов (остается $12-2k$ невзвешенных монеток, любая из них может оказаться фальшивой, причем она может быть, как легче, так и тяжелее остальных). А при неравенстве весов групп остается $2k$ возможных исходов (любая из взвешенных монет может быть фальшивой). Поэтому оба числа $2(12-2k)$ и $2k$ не должны превосходить 9, что возможно только при $k=4$.

Итак, в первом взвешивании мы должны сравнить две группы из 4 монеток. Из предыдущего анализа следует, что при результате взвешивания остается 8 возможных исходов.

Разберем три результата первого взвешивания по отдельности. Пусть оказалось, что веса двух групп равны. Тогда все взвешенные монеты настоящие. Остальные монеты будем называть *подозрительными*. Сколько подозрительных монет должны участвовать во втором взвешивании? Мы утверждаем, что ровно три. Действительно, пусть во втором взвешивании всего участвуют k подозрительных монет. Тогда, если фальшивая среди них, то при любом из двух результатов взвешивания остаётся k вариантов (фальшивой может оказаться любая из них). Поэтому $k \leqslant 3$. С другой стороны, если фальшивая монета не попала в число взвешиваемых, то весы покажут равенство и у нас остаётся $2(4-k)$ вариантов (фальшивой может быть любая из оставшихся и она может быть легче или тяжелее). Поэтому $2(4-k) \leqslant 3$ и $k \geqslant 3$.

Итак, во втором взвешивании должны участвовать ровно 3 подозрительных монеты. Можно проверить, что любое взвешивание двух равночисленных групп, содержащих вместе ровно 3 подозрительных монетки, годится. Самый простой способ — сравнить три подозрительных монетки с тремя настоящими. Если весы покажут неравенство, то фальшивая монетка находится в группе из трех отобранных для взвешивания подозрительных монет, и мы знаем, легче или тяжелее. Если весы покажут равенство, то мы знаем фальшивую монетку (но не знаем, легче она или тяжелее). В первом случае надо сравнить любые две подозрительных монетки, а во втором — сравнить фальшивую и настоящую монетки.

Осталось разобрать случай, когда первое взвешивание обнаружило, что одна группа из четырех монеток тяжелее другой группы из четырех монеток. Теперь у нас имеется 8 подозрительных монеток, 4 *легких* и 4 *тяжелых*. Максимум 3 из них могут не участвовать во втором взве-

шивании (поскольку любая из них может быть фальшивой). Есть ли верхняя оценка на количество подозрительных монеток, участвующих во втором взвешивании? Такая оценка получается из следующего соображения. Пусть во втором взвешивании на одной чашке весов ровно k легких подозрительных монеток, а на другой чашке ровно l тяжелых. Тогда $k+l$ должно не может превосходить 3. Действительно, если первая чашка перевесит вторую, то любая из этих $k+l$ монеток может быть фальшивой, и у нас остается $k+l$ возможных исходов.

Любое второе взвешивание, удовлетворяющее обоим этим требованиям, годится. Например, положим во втором взвешивании на каждую из чашек по две тяжелых и по одной легкой подозрительной монетке. При равенстве весов остается два варианта (фальшивая монета легче и она находится среди не участвовавших во втором взвешивании 2 легких подозрительных монет; ее можно найти, сравнив одну из этих 2 монет с настоящей). По симметрии, при каждом из двух других результатов остается 3 возможных исхода и их легко различить, сравнив между собой две тяжелых монеты с той чашки, которая перевесит.

1.6 Цена информации

Задача 6. Имеется неизвестное число от 1 до n (где $n \geqslant 2$). Разрешается задавать любый вопросы с ответами ДА/НЕТ. При этом при ответе ДА мы платим 1 рубль, а при ответе НЕТ — 2 рубля. Сколько необходимо и достаточно заплатить для отгадывания числа?

Решение. Верхняя оценка. На этот раз информация стоит денег и представляется разумным задавать вопросы так, чтобы дорогие отрицательные ответы приносили вдвое больше информации, чем положительные. Таким образом один бит информации будет нам обходиться в некоторое постоянное число c рублей (независимо от неизвестного числа) и при любом задуманном числе мы заплатим $c \log n$ рублей.

Это означает, что каждый раз разумно задавать такой вопрос, чтобы ответы ДАи НЕТделили его в пропорции $\alpha : (1-\alpha)$, где

$$2 \log 1/\alpha = \log 1/(1-\alpha).$$

То есть, α должно быть корнем уравнения $\alpha^2 = 1-\alpha$, $\alpha = (\sqrt{5}-1)/2$, а ответы ДА/НЕТдолжны делить множество исходов в пропорции «золотого сечения».

Обозначим $1/\alpha = (\sqrt{5}+1)/2$ через γ. При ответе ДА мы получим $\log\gamma$ бит информации, а при ответе НЕТ— $2\log\gamma$ бит. В любом случае мы заплатим $c = 1/\log\gamma \approx 1.44$ рублей за один бит, а значит в целом $\log_\gamma n$ рублей.

В этом рассуждении мы игнорировали проблему округления. На самом деле мы не можем разделять текущее множество исходов в отношении золотого сечения, поскольку оно иррационально, а количество исходов — целое число. Поэтому вместо множества исходов X, согласованных с полученными ответами, мы будем хранить некоторый отрезок $Y \subset [1,n]$ с действительными концами. Текущее множество исходов X будет равно множеству всех целых чисел, принадлежащих хранимому отрезку Y. На каждом шаге мы будем делить текущий отрезок Y в пропорции золотого сечения и спрашивать, какой из двух частей принадлежит задуманное число. Тогда при любом ответе на заданный вопрос отношение количества заплаченных денег к величине, на которую уменьшился $\log|Y|$, будет в точности равно c. Мы будем задавать вопросы до тех пор, пока длина отрезка Y не станет меньше 1 (как только она станет меньшей 1, мы узнаем задуманное число). В начальный момент длина отрезка Y равна $n-1$. Поэтому к моменту задания последнего вопроса $\log|Y|$ уменьшится не более, чем на $\log(n-1)$, а значит мы заплатим не более, чем $c\log(n-1) = \log_\gamma(n-1)$ рублей (к этому моменту). Значит всего мы заплатим не более $\log_\gamma(n-1)+2$ рублей. Поскольку количество заплаченных рублей целое, это число можно округлить до целого вниз и получить окончательный ответ в $\lfloor\log_\gamma(n-1)\rfloor + 2$ рублей.

Нижняя оценка. Попробуем «обратить» это рассуждение. Пусть имеется некоторый алгоритм и надо указать такое $x = 1,\ldots,n$, для которого алгоритм заплатит не меньше $\log_\gamma n$ рублей (где $\gamma = (\sqrt{5}+1)/2$). Будем на каждый вопрос давать наименее выгодный для алгоритма ответ. А именно, пусть X есть множество исходов, согласованных с полученными к данному моменту ответами. Пусть ответы ДА/НЕТ на очередной вопрос делят X в пропорции $\beta : (1-\beta)$. Выбираем ответ ДА/НЕТ в зависимости от того, какое из двух чисел

$$\frac{1}{\log 1/\beta}, \quad \frac{2}{\log 1/(1-\beta)}$$

больше. Эти числа равны количеству рублей, заплаченных за один бит полученной информации в случае ответов ДА и НЕТ, соответственно.

При увеличении β первое из этих чисел увеличивается, а второе — уменьшается. Мы уже знаем, что при $\beta = \frac{\sqrt{5}-1}{2}$ оба числа равны $c = 1/\log\gamma$. Поэтому при любом β хотя бы одно из них не меньше c.

Таким образом, мы заставляем алгоритм платить не менее c рублей за бит. Поскольку в целом мы сообщаем алгоритму $\log n$ бит информации, он в худшем случае заплатит $c\log n = \log_\gamma n$ рублей. Поскольку количество заплаченных рублей является целым числом, это можно округлить вверх и получить окончательную нижнюю оценку в $\lceil\log_\gamma n\rceil$ рублей.

Легко увидеть, что доказанные верхняя и нижняя оценки отличаются максимум на 1. Можно ли найти точное значение искомой величины, то есть, наименьшее количество рублей k, достаточное для нахождения любого неизвестного числа среди данных n чисел? Оказывается, что его нетрудно выразить через числа Фибоначчи. Числа Фибоначчи определяются следующим образом: $n_0 = n_1 = 1$ и $n_k = n_{k-1} + n_{k-2}$ для всех $k > 1$. Нетрудно доказать следующее: *наибольшее число n, для которого можно найти неизвестное число x среди данных n чисел, заплатив не более k рублей, равно k-ому числу Фибоначчи n_k*. Для $k = 0, 1$ это очевидно. Для остальных k это доказывается по индукции. В самом деле, найти неизвестное число x от среди n_k чисел, истратив не более k рублей можно так. Разделим данное множество из n_k чисел на части мощности n_{k-1} и n_{k-2}. Затем поинтересуемся, лежит ли x первой части. Если да, то по индуктивному предположению, истратив оставшиеся $k-1$ рублей, мы можем найти его среди чисел первой части. Если нет, то опять же индуктивному предположению, истратив оставшиеся $k-2$ рубля, мы можем найти его среди n_{k-2} чисел второй части. С другой стороны, допустим некоторый алгоритм может найти неизвестное число среди $n > n_k$ чисел, истратив не более k рублей. Рассмотрим две части, на которые делит ответ на первый вопрос эти n чисел. Либо часть, соответствующая ответу ДА, содержит более n_{k-1} элементов, либо часть, соответствующая ответу НЕТ, содержит более n_{k-2} элементов (либо и то, и другое). В первом случае ответим ДА, и по индуктивному предположению алгоритму не хватит оставшихся $k-1$ рублей, чтобы найти x среди элементов первой части. В противном случае ответим НЕТ, и по индуктивному предположению алгоритму не хватит оставшихся $k-2$ рублей, чтобы найти x среди элементов второй части.

Известно, что логарифм k-ого числа Фибоначчи по основанию $(1+\sqrt{5})/2$ примерно равен k, так что новое решение согласуется со старым.

Что получится, если в условиях задачи изменить суммы, выплачива-

емые за положительный и отрицательный ответы? Пусть, скажем, ответ ДА стоит b, а ответ НЕТ стоит c (мы предполагаем, что b, c положительны). Тогда множество возможных исходов надо делить в отношении $\alpha^b : \alpha^c$, где α есть положительный корень уравнения

$$\alpha^b + \alpha^c = 1.$$

(Поскольку функция $\alpha^b + \alpha^c$ возрастает от нуля до бесконечности на положительных числах, это уравнение имеет единственный положительных корень.) Применяя в точности те же самые рассуждения, мы получим, что для нахождения неизвестного числа среди n необходимо заплатить $\lceil \log_{1/\alpha} n \rceil$ и достаточно заплатить $\lfloor \log_{1/\alpha}(n-1) \rfloor + \max\{b, c\}$.

Рассуждение же с числами Фибоначчи можно обобщить только, если отношение b/c является рациональным числом. Пусть скажем, $b/c = m/l$, где m, l целые положительные числа. Без ограничения общности можно считать, что $b = m$ и $c = l$ (этого можно добиться, умножив b и c на подходящее положительное число). Тогда наибольшее n, для которого среди n чисел можно найти неизвестное число, заплатив не более k, удовлетворяет рекуррентному соотношению $n_k = n_{k-m} + n_{k-l}$ (для всех $k \geqslant m, l$). Очевидно, что $n_k = 1$ для всех $k < \max\{m, l\}$. С помощью рекуррентного соотношения можно быстро вычислить n_k для любого данного k.

1.7 Задачи для самостоятельной работы

Задача 7. Среди одинаковых на вид n камешков один радиоактивен. Имеется счетчик Гейгера, позволяющий выяснить, есть ли радиоактивный камешек в любой группе камешков. Докажите, что необходимо и достаточно $\log n$ применений счетчика Гейгера для нахождения радиоактивного камня.

Задача 8. Дано пять монет разного веса и известно, что первая монета легче второй, а третья легче второй и четвертой. Имеются весы, позволяющие сравнить по весу любые две монеты. Доказать, что для того, чтобы упорядочить по весу все монеты, необходимо и достаточно 5 взвешиваний.

Задача 9. Имеются 24 монеты, среди которых ровно одна фальшивая (неизвестно какая). Все настоящие монеты одного веса, а фальшивая легче или тяжелее. На чашечных весах можно сравнивать по весу любые две группы монет. Нужно найти фальшивую монету и выяснить,

легче она или тяжелее. Докажите, что необходимо и достаточно сделать 4 взвешивания.

Задача 10. Даны две группы камешков, причем камешки в каждой группе упорядочены по весу. В первой группе n камешков, а во второй — m. Требуется упорядочить все камешки по весу. Докажите, что для этого необходимо сделать $\log C_{m+n}^m$ взвешиваний. [Указание. Количество возможных способов соединить два упорядоченных массива из n и m элементов в один в точности равно количеству последовательностей из n черных и m белых шаров.]

Задача 11. Докажите, что в предыдущей задаче
(а) достаточно сделать $m+n-1$ взвешиваний,
(б) достаточно сделать $m\lceil\log(n+1)\rceil$ взвешиваний,
(в) достаточно сделать $m(\log n - \log m + O(1))$ взвешиваний (в предположении $n \geqslant m$).

[Указание. (а) Можно применить алгоритм соединения двух упорядоченных массивов, используемый в алгоритме «сортировки слиянием». (б) Можно по очереди вставлять каждый из m элементов второго массива в первый массив. При этом очередной элемент можно не сравнивать с уже вставленными элементами. (в) Можно усовершенствовать алгоритм из предыдущего пункта так. Сначала вставляем в первый массив средний из элементов второго массива. После этого вставляем в первый массив средний из нижней половины и средний из верхней половины второго массива. При этом место для среднего из нижней половины ищем только среди элементов первого массива, меньших среднего элемента второго массива (и аналогично для среднего из верхней половины). На третьем шаге вставляем средние четвертинок второго массива и так далее до тех пор, пока все элементы второго массива не будут вставлены. Вставка одного элемента в массив из k элементов обходится примерно в $\log k$ сравнений. Из вогнутости логарифмической функции следует, что наихудшим для нас будет случай, когда все элементы второго массива попадают в середину той части первого массива, место в которой мы для них ищем. А в этом случае количество сравнений как раз и равно $m(\log n - \log m + O(1))$.]

Задача 12. Имеется неизвестное число от 1 до n (где $n > 1$). Разрешается задавать любые вопросы с ответами ДА/НЕТ. При этом при ответе ДА мы платим 2 рубля, а при ответе НЕТ — 3 рубля. Сколько необходимо и достаточно заплатить для отгадывания числа?

{p66}

Задача 13. Первый игрок задумал число x от 1 до n. На своем ходу второй игрок задает ему вопросы вида: «верно ли, что $x \leqslant k$», выбирая конкретное k по своему усмотрению. На такой вопрос первый игрок обязан ответить НЕТ, если это неравенство неверно, и может дать любой ответ (ДА/НЕТ), если неравенство верно. (То есть, мы можем доверять ответам ДА, но не можем доверять ответам НЕТ.) Докажите, что первый игрок может, задав $6\log n$ вопросов, найти неизвестное x при условии, что количество неправильных ответов не превысило $\log n$.

[Указание. Нужно запустить алгоритм бинарного поиска, модифицированный следующим образом. Каждая вершина дерева поиска соответствует некоторой гипотезе $x \in (a, b]$ о неизвестном исходе x. Эта гипотеза возникла вследствие ответов второго игрока — в некоторый момент он ответил отрицательно на вопрос «$x \leqslant a$?» и утвердительно на вопрос «$x \leqslant b$?» Правила игры гарантируют истинность неравенства $x \leqslant b$, но не гарантируют истинность неравенства $x > a$. Поэтому на каждом шаге алгоритма мы сначала еще раз спрашиваем, верно ли, что $x \leqslant a$. Если нам отвечают ДА, то мы возвращаемся в дереве поиска к отцу текущей вершины. А иначе делим отрезок пополам и переходим к одному из сыновей текущей вершины, как в обычном бинарном поиске (если мы уже находимся в некотором листе, то деление отрезка пополам пропускаем). Рассмотрим следующую «потенциальную» функцию: (расстояние в дереве поиска от текущей вершины до цели)$-2\cdot$(количество неправильных ответов, полученных к текущему моменту). Нетрудно увидеть, что если мы находимся не в целевом листе дерева поиска, то значение потенциальной убывает на 1 (или даже более) на каждом шаге алгоритма. В начале значение этой функции равно $\log n$, а в конце оно не меньше $-2\log n$. Поэтому через $3\log n$ шагов (или раньше) мы окажемся в целевом листе. Осталось заметить, что раз попав в него, мы никогда из него не выйдем. Действительно, при $x = b = a + 1$ на вопрос $x \leqslant a$ первый игрок обязан отвечать отрицательно.]

{p67}

Задача 14. Докажите, что если первый игрок может лгать независимо от того, какой из двух ответов ДА/НЕТ правильный (но общее количество неверных ответов по-прежнему не больше $\log n$), второй игрок может узнать x, задав не более $18\log n$ вопросов.

[Указание. Алгоритм из предыдущей задачи теперь не годится по двум причинам: во-первых, теперь мы не можем доверять ни левой, ни

правой границе текущего отрезка, во-вторых, противник может привести нас в целевой лист, затем увести из него. Модифицируем алгоритм так: перед делением пополам проверим обе границы текущего отрезка и, если первый игрок не подтвердит хотя бы одной из них, возвращаемся в дереве поиска к отцу текущей вершины. А иначе делим отрезок пополам, как при обычном бинарном поиске. Рассмотрим ту же самую потенциальную функцию. По-прежнему она убывает на 1 (или больше) на каждом шаге алгоритма, выполненном не в целевом листе, и не возрастает на шагах, выполненных в целевом листе. Поэтому на первых $6 \log n$ шагах алгоритм не менее $3 \log n$ раз должен пробыть в целевом листе. Поскольку только один лист может обладать этим свойством, мы можем его найти как тот лист, в котором мы были чаще всего на первых $6 \log n$ шагах.]

Глава 2

Логика знания

Пусть A — некоторое множество. Как и в предыдущем параграфе, элементы A — это неизвестные нам исходы, о которых имеется некоторая информация. В логике знаний его элементы принято называть *мира́ми*. Пусть f — некоторая функция из A в некоторое множество I (значение $f(x)$ надо понимать, как известную нам информацию о мире x, в котором мы живём). Для логики знания неважно, какие значения принимает f, важно лишь, на какие классы эквивалентности значения f разбивают A (каждый класс эквивалентности состоит из всех исходов с одинаковым значением $f(x)$). Итак, задано множество A и некоторое отношение эквивалентности $\sim$ на множестве A.

{ex1}

Пример 2. Пусть мирами являются целые числа от 1 до 5, а известная нам информация есть остаток от деления на 3. Тогда классами эквивалентности будут $\{1, 4\}$, $\{2, 5\}$, $\{3\}$.

Пусть φ — это некоторое утверждение о мирах. Все миры делятся на те, в которых утверждение истинно, и те, в которых оно ложно. Например, в примере 2 утверждение «наш мир больше 2» истинно в мирах $3, 4, 5$ и ложно в мирах $1, 2$. Говорят, что *в мире x мы знаем, что φ*, если утверждение φ истинно во всех мирах, эквивалентных x. Утверждение «мы знаем, что φ» будет записываться, как $\Box\varphi$. Множество миров, в которых $\Box\varphi$ истинно, состоит из объединения всех классов эквивалентности, включённых в множество миров, в которых истинно утверждение φ. Поэтому, если в данном мире мы знаем, что φ, то и в самом деле φ истинно в этом мире. Возвращаясь к примеру 2, в мирах $1, 4$ и 3 мы знаем, что наш мир меньше 5, а в мирах 2, 5 мы этого не знаем.

К утверждениям о мирах можно применять обычные логические связки. А именно, утверждение «φ и ψ» истинно во всех мирах, в которых истинно и φ и ψ. Утверждение «φ или ψ» истинно во всех мирах, в которых истинно φ или истинно ψ. Утверждение «не-φ», то есть, «неверно, что φ», истинно во всех мирах, в которых φ ложно. Утверждение «мы не знаем, что φ» по определению означает «неверно, что мы знаем, что φ». В указанном выше примере мы не знаем, что наш мир меньше пяти в мирах 2 и 5. Заметим, что иногда, говоря, что некий человек не знает, что φ, мы подразумеваем, что φ истинно (но человек этого не знает). Мы этого никогда не будем подразумевать.

Из определения непосредственно следует, что если в данном мире мы знаем, что φ, то в этом мире мы знаем, что мы это знаем. Точно так же, если в данном мире мы не знаем, что φ, то в этом мире мы знаем, что мы этого не знаем.

Теперь немного усложним сценарий. Пусть опять имеется некоторое множество миров A и k человек, называемых *жителями*. Все жители живут в одном и том же мире и каждого жителя имеется своя информация о нём. Это означает, что заданы k отношений эквивалентности на A, которые мы будем обозначать, через $\sim_1, \ldots, \sim_k$. Так же, как и раньше, определяются понятие «в мире x житель i знает, что φ». Это утверждение обозначается через $\Box_i\varphi$. Можно использовать и более сложные утверждения, например, Петя знает, что Вася не знает, что φ: $\Box_{\text{Петя}}\Box_{\text{Вася}}\varphi$.

Пример 3. Пусть мирами являются целые числа от 1 до 5, а жителей двое — Алёна и Боря. Известная Алёне информация о мире есть остаток от деления на 3, а известная Боре информация — остаток от деления на 2. Тогда Алёниными классами эквивалентности будут $\{1,4\}$, $\{2,5\}$, $\{3\}$, а Бориными — $\{1,3,5\}$, $\{2,4\}$. В мире 1 Алёна знает, что их мир меньше 5, а вот Боря этого не знает. А в мире 4 оба это знают. В мире 1 Алёна не знает, что Боря не знает, что их мир меньше 5. (Действительно, в мире 4, эквивалентном с точки зрения Алёны миру 1, Боря это знает.)

2.1 Сообщения и увеличение знаний

Пусть кто-то из жителей высказал вслух некоторое истинное суждение о мире (так чтобы все жители слышали это суждение). Такие действия мы будем называть *сообщениями*. Мы будем предполагать, что все жители верят в высказанное суждение и тем самым их информация о мире

увеличивается. Точнее, множество возможных миров сужается путем выбрасывания всех миров, где высказанное суждение ложно. В частности, и некоторые классы эквивалентности из-за этого уменьшаются.

Мы будем рассматривать только такие ситуации, когда сделавший сообщение житель знает, что сообщение истинно. Поэтому класс эквивалентности этого жителя остаётся прежним. Однако это не означает, что его знания не изменились. А именно, сделавший сообщение теперь знает, что все остальные жители знают, что высказанное им суждение истинно (а раньше он мог этого не знать). Это можно наглядно показать на следующем примере. Пусть жителей два, Алёна и Боря, а мирами являются числа 1,2,3,4,5,6. Пусть Алёнины классы эквивалентности суть $\{1,2\}$, $\{3,4\}$ и $\{5,6\}$, а Борины — $\{1,3\}$, $\{2,5\}$, $\{4\}$ и $\{6\}$. И пусть их миром является 1. Предположим, что Алёна сообщает Боре, что мир не равен 5. Боря и сам это знал, но Алёна раньше не знала (а теперь знает), что он это знает. Практическую ценность этого знания Алёны можно увидеть следующим образом. Предположим, что после этого Боря сообщит Алёне, что он не знает, чему равен их мир. Тогда их мир не равен 2 (после исключения мира 5 в мире 2 Боря знает, чему равен их мир). Алёна теперь знает, что их мир равен 1. А если бы Боря сделал то же самое сообщение до того, как Алёна сделала первое сообщение, то Алёна все ещё этого не знала бы.

Это относится не только к жителю, сделавшему сообщение, но и к другим жителям. А именно, знания любого жителя i могут увеличиться после сообщения другим жителем j некоторого суждения, истинность которого жителю i была и так известна. Пусть, скажем, в примере из предыдущего абзаца, Боря сообщил, что ему известно, что мир не равен 6. Алёна и так это знала (поскольку возможные с ее точки зрения миры — это 1 и 2). Но теперь она знает, что Боря знает, что она это знает! Практическую ценность для Алёны этого знания можно увидеть, например, в такой ситуации. Представим себе, что после этого Алёна сообщила Боре, что она не знает, чему равен их мир, а Боря ответил, что и он всё еще не знает. После этого Алёна уже знает, что их мир равен 1, поскольку в мире 2 Боря знал бы после её сообщения, что их мир равен 2 (так как после первого сообщения Бори в мире 5, эквивалентном миру 2, Алёна знает, чему равен их мир). Однако, без первоначального сообщения Бори о том, что мир не равен 6, Алёна этого не знала бы.

Задачи из следующих двух разделов посвящены исследованию роста знаний при высказывании жителями вслух различных (истинных)

утверждений.

2.2 Карточки с цифрами

{p-card}

Задача 15. Алёны и Боря держат в руках некоторую карточку. Карточка имеет две стороны, на одной стороне написано некоторое целое неотрицательное число n, а на другой — $n+1$. Алёна видит одну из двух сторон карточки, а Боря — другую. Происходит следующий диалог. Алёна сообщает Боре, что она не знает, что написано на другой стороне. После этого Боря сообщает то же самое Алёне (что он не знает, что написано на другой стороне). Этот диалог повторяется 10 раз — всего они делают 10 пар сообщений. После этого Алёна говорит, что теперь она знает Борино число. Какая у них карточка?

Решение. Множество A возможных миров в этой задаче состоит из всех пар натуральных чисел вида $(n, n+1)$ и $(n+1, n)$ (первое число — Алёнино, а второе — Борино). Утверждение о том, что Алёна знает Борино число, истинно в данном мире, только если класс эквивалентности этого мира (с точки зрения Алёны) одно-элементен.

Изначально, одно-элементный класс (с точки зрения Алёны) единствен — это $\{(0,1)\}$. То есть, первым сообщением Алёна говорит Боре, что она видит не 0. Из-за чего могут возникнуть новые одно-элементные классы? Из-за того, что сообщения сокращают множество возможных миров (нужно удалить те миры, в которых сообщения ложны).

Итак, после первого сообщения Алёны остаются возможными все миры, кроме мира (0,1). Теперь уже существует два мира, (1,0) и (2,1), в которых Боря знает Алёнино число. Поэтому после первого сообщения Бори остаются возможными все миры, кроме (0,1), (1,0) и (2,1). Рассуждая по индукции, нетрудно установить, что после i-ого сообщения Алёны, возможными мирами являются все миры квадранта $\{(m,n) \in A \mid m \geqslant 2i-1,\ n \geqslant 2i-2\}$ (и только они), а после i-ого сообщения Бори, возможными мирами являются все миры квадранта $\{(m,n) \in A \mid m \geqslant 2i-1,\ n \geqslant 2i\}$ (и только они).

Таким образом, после десятого сообщения Бори остаются возможными все миры $(m,n) \in A$ для которых $m \geqslant 19$ и $n \geqslant 20$, и Алена знает Борино число в мирах $(19,20)$ и $(20,21)$.

2.3 Задача о шляпах.

{p-hats}

Задача 16. Три джентльмена заходят в комнату, в которой имеется три красных и две белых шляпы (которые они видят). Внезапно выключается свет, и в темноте каждый джентльмен надевает одну шляпу, цвета которой он не видит. После включения света каждый видит, какая шляпа на других, но не видит своей шляпы. Происходит следующий диалог. Первый сообщает, что не знает, какая на нём шляпа. Второй сообщает, что и он не знает, какая на нём шляпа. После этого третий сообщает, что теперь он знает, какая на нём шляпа. Какая шляпа на третьем джентльмене?

Решение. Множество A возможных миров в этой задаче состоит из всех слов длины три в алфавите $\{,\}$, кроме слова (поскольку белых шляп всего две). Классы эквивалентности с точки зрения первого суть $\{\}$, $\{,\}$, $\{,\}$ и $\{,\}$. Аналогичным образом определяются классы эквивалентности с точки зрения второго и третьего. (Можно себе представлять миры как вершины куба. Тогда эквивалентные с точки зрения первого миры соединяются ребрами куба, параллельными оси абсцисс. Эквивалентность второго задается ребрами, параллельными оси ординат, а третьего — оси апликат.)

В мире x джентльмен знает цвет своей шляпы, если класс мира x одно-элементен. Поэтому единственным миром, в котором первый знает цвет своей шляпы, является мир , и после сообщения первого джентльмена этот мир удаляется из множества возможных миров. Теперь второй джентльмен знает цвет своей шляпы в двух мирах, и . Удалив и эти миры, мы получим четыре мира, , , и , в каждом из которых третий знает цвет своей шляпы (красный).

2.4 Задачи для самостоятельной работы

Задача 17. Вова загадал число $x = 0, 1, 2, 3, 4, 5$ и сообщил Алене $\lfloor x/2 \rfloor \pmod 3$, а Боре — $\lceil x/2 \rceil \pmod 3$. При каких значениях x Боря знает что $x > 1$? При каких значениях x Алена знает, что он это знает? При каких значениях x Алена знает, что он этого не знает?

Задача 18. Вова загадал число $x = 0, 1, 2, 3, 4, 5$ и сообщил Алене $\lfloor x/2 \rfloor \pmod 3$, а Боре — $\lceil x/2 \rceil \pmod 3$. При каких значениях x Алена знает,

что $x < 3$? При каких значениях x Боря знает, что она это знает? При каких значениях x Боря знает, что она этого не знает?

Задача 19. Вова загадал число $x = 0, 1, 2, 3, \ldots, 10$ и сообщил Алене $\lfloor x/2 \rfloor$, а Боре — $\lceil x/2 \rceil$. При каких значениях x Алена знает, что x не делится на 3? При каких значениях x Боря знает, что она это знает? При каких значениях x Боря знает, что она этого не знает?

Задача 20. В условиях задачи 15 происходит следующие диалоги.
(а) Алёна: «Я знаю, что ты не знаешь моего числа». Боря: «Теперь я знаю, твое число».
(б) Алёна: «Я не знаю твоего числа». Боря: «А я это и без тебя знал». Алёна: «Ага, теперь я знаю твоё число».
В какой ситуации возможен каждый из диалогов (какая карточка и Алёны и Бори)?

Задача 21. В условиях задачи 16 происходит следующие диалоги.
(а) Первый: «Я знаю, что никто не знает, какая на нём шляпа». Второй: «А я этого не знал».
(б) Первый: «Я знаю, что третий не знает цвета своей шляпы». Второй: «Даже теперь я не знаю цвета моей шляпы».
В какой ситуации возможен каждый из диалогов?

Задача 22. (Задача о неверных женах.) В некотором городе имеется k неверных жен (и какое-то количество верных жен). Муж каждой из неверных жен не знает, что его жена ему неверна, но знает, что все остальные неверные жены изменяют своим мужьям. В некоторый день в 12 часов дня объявляется во всеуслышание, что в городе имеются неверные жены, и что любой муж, который знает, что его жена неверна, обязан на следующий день в 12 часов дня ее прилюдно наказать (по любой другой причине прилюдное наказание запрещено). Докажите, что через k дней, но не раньше, все неверные жены будут наказаны. [Указание. Если $k > 1$, то каждый и без сделанного объявления знает, что в городе есть неверные жены. Однако, объявление и в такой ситуации не бессмысленно, поскольку оно гарантирует, что все знают, что $k > 0$. Например, при $k = 2$, муж любой из двух неверных жен без такого объявления не был бы уверен, что муж другой неверной жены знает, что $k > 0$.]

2.5 Отступление: парадокс неожиданной контрольной

В задачах из предыдущих разделов мы всегда считали, что сообщения, принимаемые жителями на веру, истинны в реальном мире. Что может получиться, если принять на веру ложное сообщение, можно увидеть на примере «парадокса неожиданной контрольной» (в другой формулировке — «парадокса неожиданной казни»).

Формулировка парадокса. Учитель сообщил своим ученикам, что на следующей неделе даст контрольную, но она будет неожиданной: накануне дня контрольной ученики не будут знать, что она состоится на следующий день. Ученики рассуждают следующим образом. «В субботу контрольной не будет, поскольку иначе накануне мы знали бы об этом. Раз в субботу контрольной не будет, то и в пятницу ее не будет, поскольку иначе мы знали бы об этом накануне.» Рассуждая подобным образом, они пришли к выводу, что учитель их обманул. Однако учитель дал контрольную в пятницу, тем самым выполнив своё обещание: контрольная была и оказалась неожиданной.

Хочется понять, правильно ли рассуждали ученики и в самом ли деле учитель выполнил свое обещание. Сначала разберёмся, какие в этой задаче возможные миры и как определяется знание. Имеется семь возможных миров, соответствующих семи дням недели. В мире «D» контрольная состоялась в день недели D за исключением случая, когда D есть воскресенье, а в мире «Воскресенье» контрольная вообще не состоялась. Накануне каждого дня недели D у нас имеется свое отношение эквивалентности $\sim_D$, соответствующее знаниям учеников накануне дня D. По отношению эквивалентности $\sim_D$ все миры, предшествующие D, образуют одно-элементные классы, а все миры после D (включая D) эквивалентны друг другу.

Трудность в анализе парадокса состоит в том, что сообщение учителя ссылается само на себя. (В этом смысле этот парадокс напоминает Парадокс лжеца[1].) А именно, в утверждении учителя под знаниями учеников понимаются их знания после того, как им сообщено это утверждение. Но после сообщения утверждения знания изменяются, поскольку классы эквивалентности уменьшаются (из них выбрасываются все миры, в которых это утверждение ложно). В результате этого сужения утвер-

[1]Некто говорит: «Я сейчас лгу». Сказал ли он правду?

ждение о незнании становится более сильным, классы опять сужаются и так далее. Однако, в отличие от Парадокса лжеца, имеется монотонность: при очередном сужении классов утверждение учителя становится более сильным (множество миров, в которых оно истинно, уменьшается, а точнее, не увеличивается). Поскольку количество миров конечно, рано или поздно произойдёт стабилизация — множество миров, в которых истинно утверждение учителя, перестанет изменяться, достигнув своего предела. Утверждение учителя надо понимать, как утверждение, истинное в мирах этого предельного множества (и только в них).

Собственно, ученики и вычислили это предельное множество. Сначала знания учеников задаются исходными классами эквивалентности, а утверждение учителя истинно во всех мирах, кроме «Субботы» и «Воскресенья». После сужения классов эквивалентности путём выбрасывания миров «Суббота» и «Воскресенье», утверждение учителя изменяет свой смысл, и к мирам, в которых оно ложно, добавляется еще и «Пятница». На третьей итерации получаем суждение, истинное только в мирах «Понедельник», «Вторник» и «Среда», и так далее, пока не дойдём до утверждения, ложного во всех мирах.

Итак, при естественном уточнении парадокса сообщение учителя ложно во всех мирах, в частности, оно ложно и в реальном мире «Пятница». Осталось понять, почему же нам кажется, что оно истинно в реальном мире. Это происходит потому, что утверждение учителя истинно при более «бытовом» понимании знания. В бытовом смысле знание означает умение сделать гарантированно верное предсказание. Соотношение между бытовым и логическим знанием следующее. Если принятые нами аксиомы истинны в реальном мире, то всё, что мы знаем в логическом смысле, мы знаем и в бытовом смысле: любое выводимое суждение истинно в реальном мире. Напротив, если хотя бы одна из аксиом ложна в реальном мире, то в логическом смысле мы «знаем» нечто, чего в бытовом смысле не знаем (например, ложную аксиому). После принятия на веру ложного суждения наше умение правильно предсказывать уменьшается, в то время как способность к дедуктивному выводу увеличивается.

Парадокс возник из-за смешения двух разных понятий — логического и бытового пониманий знания. Утверждение учителя истинно при втором понимании, а рассуждения учеников правильны при первом понимании. Оговоримся, что мы не пытаемся уточнить бытовое значение понятия знания. Поэтому рассуждения из предыдущего абзаца не пре-

тендуют на строгость.[2]

[2]Разные комментаторы парадокса уточняют бытовое понятие знания разными способами. Приведём одно из разумных уточнений: если ученики считают, что они знают день контрольной, они могут выбрать один из дней (но только один) и, скажем, не приходить в школу в этот день. Если учитель наметил контрольную в точности на этот день, высказанное им суждение признаётся ложным. При таком уточнении понятия знания, рассуждения учеников ошибочны, начиная с первого шага. В самом деле, нет гарантии, что накануне субботы ученики знают, что на следующий день будет контрольная: они уже могли использовать свое право пропустить один из прошедших дней.

Глава 3

Коммуникационная сложность

{communication-complex

Пусть имеются три конечных множества X, Y, Z и некоторая функция $f : X \times Y \to Z$. В задачах о коммуникационной сложности Алисе дается некоторое $x \in X$, а Бобу — некоторое $y \in Y$. При этом Алиса не знает y, а Боб не знает x, но оба знают X, Y, Z и f. Обмениваясь информацией, они должны совместно найти $f(x, y)$. Точнее, каждый из них должен узнать значение $f(x, y)$. Мы интересуемся количеством битов, которое необходимо и достаточно для этого передать. Эта величина обозначается через $D(f)$.

Формально, $D(f)$ определяется с помощью понятия *коммуникационного протокола* (в коммуникационной сложности алгоритмы общения называются протоколами). *Коммуникационным протоколом вычисления* $f : X \times Y \to Z$ называется конечное дерево с корнем и двоичным ветвлением. Вершины этого дерева означают текущее состояние вычисления. Внутренние вершины этого дерева помечены буквами А (Алиса) и Б (Боб), означающими очередь хода. Каждая вершина v с пометкой А помечена еще некоторой функцией $g_v : X \to \{0, 1\}$, которая говорит Алисе, какой бит (0 или 1) нужно посылать, если вычисление находится в состоянии v. Аналогично, каждая вершина v с пометкой Б помечена некоторой функцией $h_v : Y \to \{0, 1\}$. Листы дерева соответствуют состояниям вычисления, в которых Алиса и Боб уже знают значение функции. Каждый лист v дерева помечен двумя функциями $g_v : X \to Z$ и $h_v : Y \to Z$, сообщающими Алисе и Бобу значение функции.

Вычисление в соответствии с протоколом P устроено так. Поставим фишку в корень дерева и будем двигать ее вдоль дерева к листьям следующим образом. Если фишка в данный момент находится во внутренней

вершине v, помеченной функцией g_v и буквой А, то в следующий момент она передвигается в левого сына v, если $g_v(x) = 0$, и в правого сына v иначе. Если в данный момент фишка находится вершине, помеченной буквой Б, то действуем аналогичным образом: применяем h_v к y. Двигаем фишку в соответствии с этими правилами до тех пор, пока не окажемся в некотором листе дерева, скажем v. Этот лист зависит от пары x, y и будет обозначаться через $v(x, y)$. Результат вычисления равен $g_v(x)$ с точки зрения Алисы и $h_v(y)$ с точки зрения Боба. Мы говорим, что протокол вычисляет f, если для любой пары $(x, y) \in X \times Y$ для листа $v = v(x, y)$, выполнено $g_v(x) = h_v(y) = f(x, y)$. Чуть позже мы увидим, что из этого следует, что для всех v функция f принимает одно и то же значение на всех парах x, y, для которых $v(x, y) = v$. (Поэтому без ограничения общности можно считать, что обе функции, которыми помечен каждый лист, являются константами.)

Длина пройденного нами пути будет количеством переданных битов (для данных x, y). Высота дерева является количеством переданных битов в худшем случае, и называется коммуникационной сложностью протокола. В этих терминах $D(f)$ есть наименьшая высота протокола для вычисления f. Величина $D(f)$ называется *коммуникационной сложностью* f.

Например, пусть $X = Y = \{0, 1\}^n$ и $Z = \{0, 1\}$. Тогда $D(f) \leqslant n + 1$ для любой функции $f : X \times Y \to Z$. Действительно, Алиса передает Бобу свой вход, затем Боб передает Алисе $f(x, y)$. Это протокол задается следующим сбалансированным деревом высоты $n+1$. Все вершины первых n уровней помечены буквой А и $g_v(x) = x_i$ (i-ый бит x) для любой вершины v уровня i (мы считаем, что корень находится на уровне 1). Вершины уровня $n + 1$ соответствуют всевозможным бинарным словам x длины n, причем вычисление на входе x, y как раз и приходит в вершину, соответствующую x. Каждая вершина x уровня $n + 1$ помечена буквой Б и функцией $g_x(y) = f(x, y)$. Листы находятся на уровне $n+2$ и помечены константными функциями, равными последнему переданному биту.

3.1 Среднее арифметическое и медиана мультимножества

Пусть $X = Y$ есть множество непустых подмножеств множества $\{1, \ldots, n\}$. Функция MED_n выдает средний элемент в упорядоченном мультимножестве $x \cup y$. (Средний элемент в мультимножестве определяется следующим образом. Пусть $x = \{x_1, \ldots, x_k\}$, $y = \{y_1, \ldots, y_m\}$. Отсортируем массив $x_1, \ldots, x_k, y_1, \ldots, y_m$ и обозначим через $z_1, \ldots, z_{k+m}$ полученный массив. Тогда $\mathrm{MED}_n(x, y) = z_{\lfloor (k+m)/2 \rfloor}$. Функция AVE_n выдает среднее арифметическое всех элементов из мультимножества $x \cup y$.

Задача 23. Докажите, что $\log n \leqslant D(\mathrm{AVE}_n) \leqslant c \log n$ для некоторой константы c.

Решение. Нижняя оценка доказывается с помощью информационного подхода. Пусть у Боба одно-элементное множество $\{n\}$, а Алисы одноэлементное множество $\{i\}$, где i меняется от 1 до n. Очевидно, что при разных значениях i функция AVE_n принимает разные значения. Поэтому Алиса обязана послать Бобу не менее $\log n$ битов.

Верхняя оценка: Алиса и Боб передают друг другу общее количество и сумму своих элементов.

Задача 24. Докажите, что $\log n \leqslant D(\mathrm{MED}_n) \leqslant c \log^2 n$ для некоторой константы c. (На самом деле верхняя оценка может быть улучшена до $c \log n$, но это сложнее доказать.)

Решение. Нижняя оценка доказывается точно так же, как и в предыдущей задаче. Докажем верхнюю оценку. Это делается с помощью алгоритма бинарного поиска. Чтобы узнать, верно ли неравенство

$$\mathrm{MED}_n(x, y) < k,$$

достаточно знать общее число элементов в мультимножестве $x \cup y$ и сколько из них меньше k. Для этого Алиса и Боб посылают друг другу общее количество элементов в своем множестве, а также количество тех из них, которые меньше k. Поэтому для выполнения одного шага алгоритма бинарного поиска достаточно передать $O(\log n)$ бит, а для выполнения $\log n$ шагов — $O(\log^2 n)$ бит.

3.2 Предикат равенства

Пример 4. Пусть $X = Y = \{0,1\}^n$, а $f(x,y) = 1$, если $x = y$, и $f(x,y) = 0$ иначе. Эта функция называется предикатом равенства на n-битовых строках и обозначается через EQ_n. Для вычисления $f(x,y)$ достаточно передать $n+1$ битов — Алиса передает свою строку Бобу, а затем Боб передает Алисе $f(x,y)$.

Попробуем доказать, что $D(\mathrm{EQ}_n) = n+1$, то есть, для любого алгоритма вычисления EQ_n найдутся входы x, y Алисы и Боба, на которых алгоритм передаст более n битов. Попробуем применить для этого уже известный подход. Представим себе постороннего наблюдателя (назовём его Виктором), который знает протокол и видит все переданные Алисой и Бобом биты, но не знает пары x, y. Будем выбирать переданные биты так, чтобы Виктор получил как можно меньше информации о паре (x,y). Пусть скажем корень протокола помечен буквой А. Множество X делится в корне на два множества: X_0 и X_1, в зависимости от первого переданного Алисой бита. Пойдем налево, если X_0 больше X_1, и направо, иначе. То есть, мы фиксируем первый посланный Алисой бит так, чтобы Виктор получил поменьше информации о неизвестном ему исходе. Теперь множество возможных исходов есть наибольшее из множеств $X_0 \times Y$, $X_1 \times Y$. Точно также поступим в следующей вершине.

На каждом шаге мы находимся в некоторой вершине v протокола. Множество возможных исходов, то есть множество тех пар (x,y), для которых вычисление придет в вершину v, имеет вид $A \times B$. Множества этого вида называют *прямоугольниками*. По построению мощность прямоугольника, соответствующего выбранной нами вершине после k шагов, не меньше 2^{2n-k}. Если дерево имеет высоту не больше n, то в конечном итоге мы окажемся в листе, для которого множество возможных исходов $A \times B$ довольно велико— оно имеет не менее 2^n элементов. При этом для любой пары (x,y) из $A \times B$, и Алиса и Боб знают, равны ли x, y. К сожалению, в этом еще нет противоречия, поскольку бывают большие прямоугольники с таким свойством. Например, в качестве A можно взять множество всех x, у которых 1-ый бит равен 0, а в качестве B — множество всех y, у которых 1-ый бит равен 1. Размер прямоугольника $A \times B$ есть $2^{2n-2} \geqslant 2^n$ (если $n \geqslant 2$). И при этом $x \neq y$ для любой пары (x,y) из $A \times B$.

Однако в этом неудавшемся рассуждении на самом деле много полезного. А именно, по существу мы доказали следующую лемму.

Лемма 1. *Для любого коммуникационного протокола и любой вершины v в этом протоколе множество R_v, состоящее из тех пар (x, y), для которых вычисление приходит в вершину v, является прямоугольником.*

Доказательство. Это легко доказывается индукцией по высоте вершины. Прямоугольники для сыновей вершины v получаются из прямоугольника $A \times B$ для вершины v следующим образом: если v помечена буквой А, то прямоугольники, соответствующие левому и правому сыну v, есть $A_0 \times B$ и $A_1 \times B$, где $A_i = \{x \in A \mid g_v(x) = i\}$. Аналогично, если v помечена буквой Б, то прямоугольники, соответствующие левому и правому сыну v, есть $A \times B_0$ и $A \times B_1$. □

Из этого следует утверждение, о котором мы уже упоминали:

Лемма 2. *Для любого коммуникационного протокола вычисления f и любого листа v функция f принимает одно и то же значение на всех парах из прямоугольника R_v (такие прямоугольники называются одноцветными для f, а значение функции f в его точках— цветом прямоугольника).*

Доказательство. Пусть $R_v = A \times B$. Тогда $g_v(x) = f(x, y) = h_v(y)$, для всех $(x, y) \in A \times B$, где g_v, h_v — функции, которыми помечен лист v. В частности $g_v(x) = h_v(y)$ для всех $(x, y) \in A \times B$, что означает, что g_v постоянна на A, а h_v постоянна на B и значит f постоянна на $A \times B$.[1] □

Теперь мы уже можем доказать неравенство $D(\mathrm{EQ}_n) > n$. Множество $X \times Y$ разбивается на одноцветные прямоугольники, соответствующие листам протокола. Если глубина протокола меньше $n+1$, то листьев в нем не больше 2^n. Ни один одноцветный прямоугольник не может содержать две разных диагональных пары (так называются пары вида (x, x)). Действительно, если прямоугольник содержит пары (x, x) и (y, y), где $x \neq y$, то он содержит и пару (x, y), а значит не является одноцветным. Диагональных пар ровно 2^n штук, поэтому для их покрытия необходимо по крайней мере 2^n прямоугольников цвета 1. Кроме того, нужен хотя бы один прямоугольник цвета 0. Поэтому 2^n одноцветных прямоугольников не хватит даже, чтобы покрыть $X \times Y$ с пересечениями.

[1] В этом рассуждении нам важно, что R_v есть прямоугольник. Иначе утверждение было бы неверным, например, для $R_v = \{(0, 0), (1, 1)\}$ можно положить $g_v(x) = x$, $h_v(y) = y$ и $f(x, y) = x = y$ для всех $(x, y) \in R_v$.

Эти рассуждения можно суммировать в следующей теореме. Матрицей функции $f : X \times Y \to Z$. называется матрица, строки и столбцы которой нумеруются элементами X и Y, соответственно, а элементы равны $f(x, y)$.

Теорема 1. *Пусть f произвольная функция из $X \times Y$ в Z. Обозначим через $C^R(f)$ наименьшее количество одноцветных для f прямоугольников, на которые можно разбить матрицу функции f. Тогда $D(f) \geqslant \log C^R(f)$.*

Доказательство. Действительно, в протоколе наименьшей высоты для вычисления f не больше $2^{D(f)}$ листьев и каждому листу соответствует одноцветный прямоугольник. Эти прямоугольники образуют разбиение матрицы функции. □

3.3 Метод трудных множеств и метод размера прямоугольников

Будем множество $H \subset X \times Y$ называть *трудным для f*, если не существует одноцветного для f прямоугольника, содержащего две различных пары из H. Очевидно, $C^R(f)$ не меньше размера любого трудного множества. Поэтому можно оценивать снизу $C^R(f)$, а значит и коммуникационную сложность, указывая большие трудные множества. В сущности мы это и сделали для оценки $C^R(EQ_n)$, предъявив трудное множество, состоящее из всех диагональных пар и еще одной не-диагональной пары.

Задача 25. Функция GT_n определена на парах натуральных чисел от 1 до 2^n и принимает значение 1, если $x > y$. Докажите, что $D(\mathrm{GT}_n) = n+1$.

Решение. Верхняя оценка очевидна. Для доказательства нижней оценки достаточно рассмотреть трудное множество, состоящее из всех диагональных пар и еще любой одной пары (x, y), для которой $x > y$.

Задача 26. Предикат DISJ_n определен на парах подмножеств $x, y \subseteq \{1, \ldots, n\}$ и принимает значение 1, если x, y не пересекаются. Доказать, что коммуникационная сложность этой функции не меньше $n + 1$.

Решение. Рассмотрим множество H состоящее из всех пар вида $(x, \bar{x})$. Любая такая пара состоит из непересекающихся множеств, поэтому может покрываться только прямоугольником цвета 1. При этом никакой

прямоугольник цвета 1 не может содержать двух таких пар, поскольку для любых разных x, y, либо x пересекает $\bar{y}$, либо $\bar{x}$ пересекает y (либо и то, и другое). Таких пар ровно 2^n и добавив в H еще любую одну пару пересекающихся множеств, мы получим трудное множество размера $2^n + 1$.

Обобщением метода трудных множеств является метод размера прямоугольников. Пусть каждой паре из $X \times Y$ сопоставлено некоторое неотрицательное действительное число, называемое её *весом*. Тогда $C^R(f)$ не меньше отношения суммарного веса всех пар из $X \times Y$ к максимальному весу одноцветного прямоугольника (весом прямоугольника называется сумма весов всех пар из него). Действительно, для любого разбиения матрицы f на одноцветные прямоугольники сумма весов всех прямоугольников из разбиения равна суммарному весу всех пар из $X \times Y$. Поэтому утверждение следует из того, что сумма не превосходит количества слагаемых, умноженного на максимальное слагаемое.

Объясним, почему метод трудных множеств является частным случаем метода размера прямоугольников. Пусть дано трудное множество H. Сопоставим всем парам из H единичный вес, а всем остальным — нулевой вес. Тогда вес любого одноцветного прямоугольника не больше 1, а суммарный вес всех пар равен $|H|$. Поэтому отношение суммарного веса всех пар к максимальному весу одноцветного прямоугольника как раз равно $|H|$.

Задача 27. Рассмотрим функцию $\mathrm{SUM}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{n+1}$, возвращающую сумму чисел, записанных в двоичной системе счисления. Докажите, $D(\mathrm{SUM}_n) = 2n$. [Указание. Множество, состоящее из всех входных пар, является трудным.]

{p-ip}

Задача 28. Функция IP_n определена на парах двоичных слов длины n и принимает значение 1, если $\sum x_i y_i \equiv 1 \bmod 2$. Докажите, что любой коммуникационный протокол, вычисляющий функцию IP_n имеет глубину не менее n. [Указание. Докажите, что любой одноцветный прямоугольник содержит не более 2^n пар (используйте соображения размерности линейных пространств).]

3.4 Метод ранга матрицы

Пусть множество Z состоит из элементов некоторого поля (например, из действительных чисел). Тогда $C^R(f)$ не меньше ранга матрицы функции f над этим полем. Действительно, фиксируем произвольное разбиение матрицы f на одноцветные прямоугольники. Для каждого прямоугольника разбиения рассмотрим матрицу, у которой в этом прямоугольнике значение функции, а вне его всюду нули. Такая матрица имеет ранг не больше 1 (поскольку все ненулевые ее столбцы одинаковы). А вся матрица функции f равна сумме таких матриц. Поскольку ранг суммы не превосходит суммы рангов, ранг матрицы f не превосходит количества прямоугольников разбиения.

Задача 29. Докажите, что ранг матрицы отрицания предиката DISJ_n над полем действительных чисел равен 2^n.

Задача 30. Докажите, что ранг матрицы предиката IP_n над полем действительных чисел не меньше $\varepsilon 2^n$ для некоторого положительного ε.

3.5 Вероятностные протоколы

В вероятностном протоколе у Алисы и Боба есть собственные независимые источники случайности. Биты, пересылаемые Алисой могут зависеть не только от ее входа и текущей вершины дерева, но и от ее случайного входа. Аналогично для Боба. Формально, функции g_v, которыми помечены вершины Алисы, в качестве аргументов получают пары из $X \times R$, а функции h_v, которыми помечены вершины Боба, — пары из $Y \times S$. Аналогично и для функций, которыми помечены листья. Пометки листьев по-прежнему являются элементами множества значений функции, которую должен вычислить протокол.

Перед началом работы Алисе выдается пара $(x, r) \in X \times R$, а Бобу — пара $(y, s) \in Y \times S$. Множества R, S могут быть произвольными (они входят в спецификацию протокола). Результат работы протокола теперь зависит от четверки x, y, r, s.

Мы говорим, что вероятностный протокол вычисляет с ошибкой не более ε (или, кратко, ε-вычисляет) функцию f, если для любой пары x, y с вероятностью не менее $1 - \varepsilon$ (при случайном выборе пары $(r, s) \in R \times S$) результат работы протокола равен $f(x, y)$ и с точки зрения Алисы,

и с точки зрения Боба. Через $R^\varepsilon(f)$ обозначается минимальная высота вероятностного протокола, ε-вычисляющего f.

Оказывается $R^\varepsilon(f)$ может быть значительно меньше $D(f)$ даже для очень малых ε. Другими словами, использование случайности может значительно сократить количество передаваемых битов.

Задача 31. Докажите, что $R^\varepsilon(EQ_n) = O(\log n + \log(1/\varepsilon))$. [Указание. Алиса передает Бобу значение некоторой хэш-функции на x, а Боб сравнивает полученное значение со значением этой же хэш-функции на своем входе и сообщает результат проверки Алисе. Оба считают, что $x = y$, если хэш-значения совпали. Хэш-функция выбирается Алисой случайно из некоторого семейства функций, которое зависит от n и ε и содержит не более $(n/\varepsilon)^c$ функций (поскольку Алиса должна передать Бобу номер выбранной функции) и принимают значения в некотором множестве мощности $(n/\varepsilon)^c$.

В качестве семейства хэш-функции годятся следующие два семейства: $x \mapsto x \bmod p$, где p — простое число подходящей величины, задающее конкретную функцию из семейства, и $x \mapsto (x_0 + x_1 a + \cdots + x_{n-1}a^{n-1}) \bmod p$, где p — простое число подходящей величины и зависит только от n, а a — натуральное число меньшее p, задающее конкретную функцию из семейства.]

Задача 32. Докажите, что $R^\varepsilon(GE_n) = O(\log n(\log\log n + \log(1/\varepsilon)))$. [Указание. Алиса и Боб используют бинарный поиск первого слева бита, в котором различаются их входы. Для этого они применяют вероятностный протокол из предыдущей задачи для сравнения своих первых половин: если они одинаковы, то первое различие левее середины, а иначе правее, и так далее. Вероятность ошибки на каждом шаге не должна превышать $\varepsilon/\log n$, тогда суммарная вероятность ошибки не превысит ε.]

Задача 33. Докажите, что если Алиса и Боб имеют доступ к общему источнику случайности (то есть, им выдается одинаковая случайная строка), то они могут ε-вычислить предикат EQ_n, передав $O(\log(1/\varepsilon))$ бит. [Указание. В этом случае они могут использовать хэш-функции из значительно большего семейства функций, прочитав номер хэш-функции из общего источника случайности. Хэш-значение должно иметь не более $O(\log(1/\varepsilon))$ бит. Например, годится семейство $x \mapsto Ax$, где A — 0-1-матрица размера $k \times n$. Умножение матрицы на столбец x проводятся в поле вычетов по модулю 2. В качестве k можно взять целую часть от $\log(1/\varepsilon)$.]

{p33}

Задача 34. Докажите, что если Алиса и Боб имеют доступ к общему источнику случайности (то есть, им выдается одинаковая случайная строка), то для любого фиксированного положительного ε они могут ε-вычислить предикат GE_n с ошибкой не более ε, передав $O(\log n)$ бит. [Указание. Используйте задачу 13 (или 14).]

Можно доказать, что любой протокол ε-вычисления функции, использующий общий источник случайности, можно переделать в вероятностный протокол её 2ε-вычисления, который использует только частные случайные биты, ценой увеличения количества переданных бит на $O(\log\log|X\times Y|+\log(1/\varepsilon))$. (Правда, алгоритм преобразования требует полиномиального от $|X\times Y|$ времени.) Поэтому из задачи 34 следует, что $R^{\varepsilon}(GT_n)=O(\log n)$ для любого фиксированного положительного ε.

Можно доказать, что для предикатов DISJ_n и IP_n случайность мало помогает: для них R^{ε} не меньше δn для некоторого положительного δ для всех достаточно малых ε. Об этом можно прочитать в превосходном учебнике по коммуникационной сложности [8].

Глава 4

Энтропия Шеннона

4.1 Определение

{entropy-code-length}

Пусть фиксированы неотрицательные числа $p_1, \ldots, p_k$, в сумме равные единице. Их энтропия Шеннона определяется как

$$H = p_1(-\log p_1) + p_2(-\log p_2) + \ldots + p_k(-\log p_k)$$

(при $p = 0$ мы полагаем $p \log p = 0$, до-определяя тем самым функцию $p \mapsto p \log p$ по непрерывности).

Мотивировка этой формулы такова: пусть имеется алфавит из k букв $a_1, \ldots, a_k$, причем буква a_i появляется с частотой p_i, а каждое её появление несёт $-\log p_i$ битов информации; в среднем получается H битов на букву. Нужно только объяснить, почему мы считаем, что появление буквы с вероятностью p несёт $-\log p$ битов информации. Это можно сделать так. Пусть частоты всех букв совпадают (и равны $1/k$). Тогда $H = \log k$, то есть в этом частном случае энтропия совпадает с информацией по Хартли в исходах из множества $\{a_1, \ldots, a_k\}$. Таким образом, энтропия Шеннона обобщает количество информации по Хартли — последнее есть шенноновская энтропия для равновероятных исходов.

Задача 35. Докажите, что энтропия Шеннона не меньше *минимальной энтропии*, определяемой как $H_{\min} = \min_i(-\log p_i)$.

Энтропия Шеннона возникает в самых разных контекстах. Первым таким примером является задача бинарного кодирования букв данного алфавита.

4.2 Коды

{prefix-codes}

Пусть нам нужно закодировать буквы некоторого алфавита A, состоящего из k букв $a_1, \ldots, a_k$, двоичными словами (наподобие азбуки Морзе, только вместо точки и тире используются нуль и единица). Пусть буква a_i кодируется некоторым словом c_i. Естественно требовать, чтобы все слова c_i были различны. Но этого мало, если мы хотим записывать коды подряд. Скажем, если алфавит состоит из букв А, Б и В, имеющих коды 0, 1 и 01, то мы не сможем отличить слово АБАБ от слова АБВ: в обоих случаях будет последовательность 0101. (В азбуке Морзе, кстати, такой проблемы нет, поскольку между отдельными буквами делается перерыв — больший, чем между точками и тире внутри буквы). Поэтому надо отдельно позаботиться об однозначности декодирования.

Другой предмет заботы при построении кода — его экономность. Полезно выбрать кодовые слова c_i по возможности более короткими (насколько это возможно при сохранении однозначности декодирования). Более того, если не удаётся сделать короткими все кодовые слова, разумно в первую очередь позаботиться о наиболее часто встречающихся буквах. (Это обстоятельство учитывалось и при составлении азбуки Морзе.)

Перейдём к формальным определениям. *Кодом* для алфавита A, состоящего из k букв $a_1, \ldots, a_k$, называется набор из k двоичных слов $c_1, \ldots, c_k$. Они называются *кодовыми словами* данного кода; слово c_i называется *кодом* буквы a_i; всякое слово в алфавите A кодируется двоичным словом, получаемым соединением кодов соответствующих букв.

Будем называть код *инъективным*, если коды различных букв различны, и *однозначно декодируемым*, если коды любых двух различных слов различны. Код называется *префиксным*, если ни одно из кодовых слов (соответствующих буквам алфавита A) не является началом (префиксом) другого. (Это название стало традиционным, хотя более логичное— *бес-префиксный* код— также используется.)

Теорема 2. *Всякий префиксный код является однозначно декодируемым.*

Доказательство. Первое кодовое слово (код первой буквы) отщепляется однозначно (в силу префиксности), затем отщепляется код второй буквы и т.п. □

Задача 36. Покажите, что не всякий однозначно декодируемый код является префиксным. [Указание. Он может быть, например, «суффиксным».]

Задача 37. Укажите явно взаимно однозначное соответствие между множеством бесконечных последовательностей цифр $0, 1, 2$ и множеством бесконечных последовательностей нулей и единиц. [Указание. Используйте префиксный код $0 \mapsto 00$, $1 \mapsto 01$, $2 \mapsto 1$.]

Задача 38. Пусть слова $c_1, \ldots, c_k$ и $d_1, \ldots, d_k$ образуют префиксный код (по отдельности). Покажите, что kl слов $c_i d_j$ (приписываем одно слово к другому без разделителя) также образуют префиксный код.

Чтобы сравнивать коды по их экономности, нужно фиксировать частоты букв. Пусть даны неотрицательные числа $p_1, \ldots, p_k$, в сумме равные единице; число p_i будем называть *частотой* (или *вероятностью*) буквы a_i. Для каждого кода $c_1, \ldots, c_k$ (для букв $a_1, \ldots, a_k$) определим *среднюю длину* кода как сумму

$$\sum_i p_i l(c_i).$$

Возникает задача: для данных $p_1, \ldots, p_k$ найти код по возможности меньшей средней длины (в том или ином классе кодов). Что можно сказать о минимально возможной длине префиксного кода для данных частот $p_1, \ldots, p_k$? Оказывается она близка к энтропии Шеннона (отличается от нее менее, чем на 1).

{prefix-code-length}

Теорема 3. (**а**) *Для любого префиксного кода с кодовыми словами $c_1, \ldots, c_k$ выполняется неравенство*

$$\sum_i p_i l(c_i) \geqslant H$$

(средняя длина кода не меньше энтропии).

(**б**) *Существует префиксный код, для которого*

$$\sum_i p_i l(c_i) < H + 1$$

Доказательство. Заметим, что в этой теореме реально фигурируют не сами кодовые слова, а их длины. Поэтому важно знать, какие наборы

чисел могут быть длинами кодовых слов префиксного кода. Ответ даёт такая лемма:

Лемма (неравенство Крафта). *Пусть фиксированы целые неотрицательные числа $n_1, \ldots, n_k$ и требуется найти двоичные слова $c_1, \ldots, c_k$ указанных длин ($l(c_i) = n_i$), причём так, чтобы ни одно из этих слов не было началом другого. Это возможно тогда и только тогда, когда $\sum_i 2^{-n_i} \leqslant 1$.* {kraft-lemma}

Доказательство. Сопоставим каждому двоичному слову v некоторый под-отрезок I_v отрезка действительной прямой $[0, 1]$ по следующему рекурсивному правилу. Пустому слову сопоставляется весь отрезок $[0, 1]$, и для любого слова v словам $v0$ и $v1$ сопоставляется левая и правая половина отрезка I_v. (Например, $I_{01} = [1/4, 1/2]$.) Будем называть отрезок *стандартным*, если он сопоставлен некоторому слову. Ясно, что отрезок стандартен тогда и только тогда, когда его длина есть целая степень двойки и начало отрезка (а значит и конец) кратно его длине.

По построению это соответствие имеет следующие свойства:

- v есть начало u тогда и только тогда, когда $I_u \subset I_v$,
- Если v и u не согласованы (ни одно из них не является началом другого), то I_u и I_v не перекрываются (общая граничная точка перекрытием не считается),
- Длина I_v равна $2^{-l(v)}$.

Теперь легко убедиться в первом утверждении: если слова $c_1, \ldots, c_k$ попарно не согласованы, то $\sum_i 2^{-l(i)} \leqslant 1$. Действительно, $2^{-l(i)}$ есть длина отрезка, сопоставленного c_i. Поскольку отрезки $I_{c_1}, \ldots, I_{c_k}$ не перекрываются, сумма их длин не превосходит длины всего отрезка $[0, 1]$.

Обратное утверждение: без ограничения общности можно считать, что $n_1 \geqslant \ldots \geqslant n_k$. Отложим на луче $[0, \infty)$, начиная с начала без пропусков не перекрывающиеся отрезки длин $2^{-n_1}, \ldots, 2^{-n_k}$. По условию сумма этих чисел не больше 1, поэтому все отложенные отрезки являются подотрезками отрезка $[0, 1]$. Докажем, что каждый из отложенных отрезков является стандартным — этого достаточно, поскольку из того, что отрезки не перекрываются следует, что соответствующие слова $c_1, \ldots, c_k$ попарно не согласованы; кроме того они имеют нужные длины по построению. Фиксируем любое $i \leqslant k$. Поскольку длина i-ого отрезка равна

2^{-n_i} нам достаточно доказать, что начало i-ого отрезка кратно 2^{-n_i}. Это начало есть сумма длин всех предыдущих отрезков, то есть

$$2^{-n_1} + 2^{-n_2} + \cdots + 2^{-n_{i-1}}.$$

Поскольку $n_1 \geqslant \ldots \geqslant n_{i-1} \geqslant n_i$, это число в самом деле кратно 2^{-n_i}. $\square$

Вернёмся к доказательству теоремы. Можно считать, что все p_i положительны, поскольку нулевые p_i не дают вклада ни в среднюю длину кода, ни в энтропию. В пункте (a) нам надо доказать, что если n_i— неотрицательные целые числа и $\sum_i 2^{-n_i} \leqslant 1$, то сумма $\sum p_i n_i$ не меньше шенноновской энтропии H. Это удобнее доказывать сразу для произвольных n_i (не обязательно целых) и перейдя к другим координатам. Обозначим через q_i величину 2^{-n_i}. В этих координатах утверждение таково: если $q_i > 0$ и $\sum q_i \leqslant 1$, то

$$\sum p_i(-\log q_i) \geqslant \sum p_i(-\log p_i)$$

Это неравенство иногда называют *неравенством Гиббса*. Чтобы доказать его, заметим, что разница между правой и левой частью равна {gibbs-inequality}

$$\sum_i p_i \log \frac{q_i}{p_i}$$

и в силу выпуклости логарифма (взвешенная сумма логарифмов не превосходит логарифма взвешенной суммы: $\sum p_i \log u_i \leqslant \log(\sum_i p_i u_i)$; это верно для любых положительных u_i) не превосходит

$$\log\left(\sum_i p_i \frac{q_i}{p_i}\right) = \log\left(\sum q_i\right) \leqslant \log 1 = 0.$$

Утверждение (a) доказано.

Отметим кстати, что неотрицательную величину

$$\sum_i p_i \log \frac{p_i}{q_i}$$

называют *расстоянием Кульбака–Лейблера* (Kullback – Leibler distance) между распределениями вероятностей p_i и q_i (при этом предполагается, что $\sum q_i = 1$), хотя это «расстояние» и не симметрично. Выпуклость

логарифма (отрицательность второй производной) гарантирует, что это расстояние неотрицательно и обращается в нуль, лишь если $p_i = q_i$ при всех i. Величину

$$\sum_i p_i \log \frac{1}{q_i}$$

называют *перекрёстной энтропией* между распределениями вероятностей p_i и q_i. Перекрёстная энтропия не меньше энтропии распределения p_i, причем разница между ними равна расстоянию Кульбака-Лейблера.

Чтобы доказать утверждение (б), рассмотрим числа $n_i = \lceil -\log_2 p_i \rceil$ (где $\lceil u \rceil$ обозначает наименьшее целое число, большее или равное u). Тогда

$$\frac{p_i}{2} < 2^{-n_i} \leqslant p_i$$

Неравенство $2^{-n_i} \leqslant p_i$ гарантирует, что выполнены условия леммы (и потому можно найти кодовые слова соответствующих длин). Неравенство $p_i/2 < 2^{-n_i}$ означает, что n_i превосходит $(-\log p_i)$ менее чем на 1, что сохраняется и после усреднения: средняя длина кода $(\sum p_i n_i)$ превосходит $H = \sum p_i(-\log p_i)$ менее чем на 1. □

Кратко доказательство теоремы можно резюмировать так: если забыть, что длины кодовых слов должны быть целыми, и разрешать любые числа n_i, только бы сумма 2^{-n_i} не превышала единицы, то выгоднее всего взять $n_i = -\log p_i$ (следует из выпуклости логарифма). Требование же целочисленности приводит к увеличению n_i, но не более чем на единицу.

Замечание 2. Кодирование, примененное в доказательстве пункта (б), обладает следующим свойством: $l(c_i) < -\log p_i + 1$. Будем кодирования с таким свойством (с произвольной константой вместо 1) называть *сбалансированными* (поскольку они сохраняют баланс между $l(c_i)$ и $-\log p_i$).

Теорема 4. *Энтропия распределения $p_1, \ldots, p_n$ с n значениями не превосходит $\log n$ и равна $\log n$ в единственном случае, когда все p_i равны.*

Доказательство. Если n есть степень двойки, то неравенство $H \leqslant \log n$ прямо следует из теоремы 3, поскольку можно рассмотреть префиксный код, в котором n кодовых слов имеют длину $\log n$. В общем случае надо применить неравенство Гиббса с $q_i = 1/n$ при всех i и вспомнить, что это неравенство обращается в равенство при $p_i = q_i$. □

Задача 39. Пусть целое число x выбирается случайным образом в интервале от 1 до 1000 (все возможные значения x равновероятны). Докажите, что любой алгоритм, который с помощью вопросов с ответами ДА/НЕТ находит x, задает в среднем не меньше $\log 1000$ вопросов. [Указание. Любой такой алгоритм задает префиксное кодирование чисел от 1 до 1000.]

{p74}

Задача 40. Докажите, что любое инъективное кодирование можно преобразовать в префиксное ценой небольшого увеличения средней длины кода: если у исходного кода средняя длина была l, то у нового она будет не больше $l + 2\log l + 2$. [Указание. Используйте композицию исходного кода и префиксного кодирования $x \mapsto \hat{x}$ со стр. 127, а также вогнутость логарифмической функции.]

{p75}

Задача 41. Пусть букв $\{a_1, \ldots, a_n\}$ произвольный алфавит и $p_1, \ldots, p_n$ — вероятности букв этого алфавита. Докажите, что для любого инъективного кодирования букв этого алфавита средняя длина кода не меньше $H - 2\log H - 2$. [Указание. Используйте предыдущую задачу и теорему 3.]

4.2.1 Некоторые известные коды

Арифметический код. Пусть все вероятности букв $p_1, \ldots, p_n$ положительны. Отложим на отрезке $[0, 1]$, начиная с начала, без пропусков не перекрывающиеся отрезки длин $p_1, \ldots, p_n$. Для каждого i рассмотрим отрезок вида I_c наибольшей длины, целиком включенный в i-ый из полученных отрезков (отрезки I_v определены в доказательстве неравенства Крафта на стр. 46). Одно из таких c (их может быть несколько) и возьмем в качестве кодового слова c_i для i-ой буквы. Заметим, что арифметический код зависит от исходного порядка на буквах, поскольку этог порядок определяет, в каком порядке мы расположим отрезки длин $p_1, \ldots, p_n$ на отрезке $[0, 1]$.

{arithmetical-balanced

Задача 42. Докажите, что $c_1, \ldots, c_n$ есть беспрефиксный код, причем $l(c_i) < -\log p_i + 2$. (То есть, арифметическое кодирование сбалансировано.)

Задача 43. Докажите, что константу 2 в неравенстве $l(c_i) < -\log p_i + 2$ для длин слов арифметического кода нельзя понизить, даже в предположении, что $p_1, \ldots, p_n$ упорядочены по величине.

Обычно арифметический код используют для кодирования букв алфавита, который состоит из слов некоторой длины m над некоторым другим алфавитом $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$, а распределение вероятностей на этих

словах бернуллиевское. Чтобы не путать эти два алфавита, мы будем называть первый алфавит «большим», а алфавит Σ — «малым». Бернуллиевское распределение задаётся вероятностями букв малого алфавита, эти вероятности мы будем обозначать через $q_1, \ldots, q_k$. По определению вероятность слова $\sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_m}$ равна $q_{i_1} \cdot q_{i_2} \cdot \cdots \cdot q_{i_m}$. Это распределение вероятностей соответствует процессу независимого выбора каждой из m букв слова, причем каждая буква выбирается по распределению $q_1, \ldots, q_k$.

Оказывается, в этом случае, несмотря на большое количество ($n = k^m$) букв можно быстро (за $O(mk)$ шагов) найти арифметический код любой данной буквы, если при построении кода используется лексикографический порядок на словах длины m над малым алавитом (= буквах большого алфавита). Это делается так. Пусть нам дано слово $\sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_m}$. Отложим на отрезке $[0, 1]$, начиная с начала, без пропусков не перекрывающиеся отрезки длин $q_1, \ldots, q_k$. Тогда отрезки, соответствующие словам, начинающемуся на букву σ_i, лежат внутри i-ого из полученных отрезков. В частности, отрезок, соответствующий данному нам слову, лежит на i_1-ом из полученных отрезков. Выберем этот отрезок и отложим на нём не перекрывающиеся отрезки длин $q_{i_1}q_1, \ldots, q_{i_1}q_k$. Выберем из них i_2-ой отрезок. Повторив эту процедуру m раз, мы найдем отрезок, соответствующий данному слову, а затем и его арифметический код.

Код Шеннона–Фано. Алгоритм Шеннона–Фано рекурсивный. На вход он получает частоты $p_1 \geqslant \ldots \geqslant p_n$ букв и двоичную строку u, а выдаёт n кодовых слов, каждое из которых начинается на u. Алгоритм работает следующим образом: если n равно 1, то выдаём в качестве кодового слова само исходное слово u и останавливаемся. Иначе укладываем на прямой без пропусков не перекрывающиеся отрезки длин $p_1, \ldots, p_n$ и обозначаем i-ый из полученных отрезков через S_i, а их объединение — через S. Буквы a_i, для которых отрезок S_i попал на левую половину отрезка S, образуют *левую* группу, их коды будут начинаться на $u0$, а те буквы, для отрезок S_i попал на правую половину отрезка S, образуют *правую* группу, их коды будут начинаться на $u1$. Один из отрезков (центральный) может не попасть целиком ни на левую, ни на правую половину. С ним поступим так. Если этот отрезок первый или последний, то отнесём его к левой или правой группе, соответственно. Иначе отнесём его к любой группе. Для определения остальных букв кодовых слов делаем два рекурсивных вызова — первый раз на вход даём набор частот букв левой

группы и слово $u0$, а второй раз — набор частот букв правой группы и слово $u1$.

Чтобы получить код Шеннона–Фано надо запустить этот алгоритм на пустом слове u и исходных частотах букв.

{p-fano-code}

Задача 44. Докажите, что код Шеннона–Фано является префиксным.

Как мы увидим позже (см. задачу 62), средняя длина кода Шеннона–Фано не превосходит $H + O(1)$. Что касается сбалансированности, то это зависит от того, куда относить центральный отрезок.

Задача 45. Докажите, что если центральный отрезок всегда относить туда, куда попала его большая часть или всегда к правой группе, то кодирование Шеннона–Фано не является сбалансированным (то есть не существует константы d, для которой выполнено $l(c_i) < -\log p_i + d$ для любых n и любых исходных вероятностей $p_1, \ldots, p_n$).

{shannon-fano-balanced

Теорема 5. *Если центральный отрезок всегда относить к левой группе, то кодирование Шеннона–Фано является сбалансированным.*

Доказательство. Входом очередного вызова алгоритма Шеннона–Фано является набор частот $p_1, ..., p_k$ и двоичная строка u. Сумму $p_1 + \cdots + p_k$ будем называть *весом* входа, а величину $2^{l(u)}(p_1 + \cdots + p_k)$ — *дисбалансом* входа. Дисбаланс отражает, во сколько раз размер кодового пространства $2^{-l(u)}$, которое мы можем использовать, меньше суммы частот букв, которые нужно закодировать. В этих терминах нам нужно доказать, что дисбаланс входов самых последних вызовов алгоритма (когда на входе только одна буква) ограничен некоторой константой. Для этого рассмотрим любую последовательность рекурсивных вызов алгоритма, начиная с исходного входа и кончая вызовом для буквы. Пусть $w_0, w_1, \ldots, w_k$ веса входов этих вызовов. Их дисбалансы равны $d_0 = w_0$, $d_1 = 2w_1$, $\ldots$, $d_k = 2^k w_k$. Мы знаем, что $d_0 = 1$ и нам нужно доказать, что дисбаланс d_k последнего вызова ограничен константой. Иными словами, нам нужно доказать, что произведение отношений d_{i+1}/d_i ограничено константой. Если при i-ом вызове центрального отрезка нет, то $d_i = d_{i+1}$. В самом деле, в этом случае при i-ом вызове сумма частот в левой и правой группах равна половине исходной суммы (и то же самое верно для длин кодовых пространств). В противном случае $d_{i+1} > d_i$, если i-ый вызов делался для левой группы частот и $d_{i+1} < d_i$, если для правой. Сумма частот в левой группе меньше длины q_i центрального отрезка плюс половина веса

входа p_i. Поэтому

$$\frac{d_{i+1}}{d_i} < \frac{(p_i/2 + q_i)2^{i+1}}{p_i 2^i},$$

или $d_{i+1}/d_i < 1 + 2q_i/p_i$. Поэтому достаточно доказать, что произведение чисел $1 + 2q_i/p_i$ по всем i, для которых $(i+1)$-ый вызов относится к левой группе частот, ограничено константой. Заметим, что из условия упорядоченности частот следует, что в этом произведении числа q_i возрастают. Числа p_i, наоборот, убывают, причём в геометрической прогрессии. Точнее, за исключением, возможно последних двух, каждый вес не менее чем в полтора раза больше следующего за ним веса. В самом деле, если q_i не больше одной шестой части от p_i, то сумма левых частот в i-ом вызове не больше $p_i/2 + q_i \leqslant 2p_i/3$. Но если центральный отрезок стал больше шестой части от суммы всех вероятностей, слева от центрального отрезка не может уместиться больше двух отрезков, поэтому левая группа состоит не более, чем из трёх букв, а значит все они получат кодовые слова в ходе не более чем двух рекурсивных вызовов. Итак, в произведении чисел $1 + 2q_i/p_i$, которое нам нужно оценить сверху, при уменьшении i числа $2q_i/p_i$ убывают в геометрической прогрессии, начиная с пред-предпоследнего члена. Причем последние три числа не больше 2, поскольку $p_i \geqslant q_i$. Отсюда следует, что произведение чисел d_{i+1}/d_i ограничено константой. □

{huffman-encoding}

Код Хаффмана.

Мы доказали, что средняя длина оптимального префиксного кода (для данных $p_1, \ldots, p_k$) заключена между H и $H + 1$. Попробуем разобраться, как найти этот код.

Пусть $n_1, \ldots, n_k$— длины кодовых слов оптимального кода для данных $p_1, \ldots, p_k$. Будем предполагать (переставив буквы), что

$$p_1 \leqslant p_2 \leqslant \ldots \leqslant p_k.$$

В этом случае

$$n_1 \geqslant n_2 \geqslant \ldots \geqslant n_k$$

(если бы более частая буква кодировалась длиннее, чем более редкая, то обмен кодов уменьшил бы среднюю длину).

Заметим, что для оптимального кода $n_1 = n_2$ (две наиболее редкие буквы всегда имеют одну и ту же длину кода). В самом деле, если $n_1 > n_2$, то n_1 больше всех остальных n_i. Поэтому в сумме $\sum_i 2^{-n_i}$

первое слагаемое меньше всех других, неравенство $\sum_i 2^{-n_i} \leqslant 1$ не может обратиться в равенство по соображениям чётности, и левая часть его меньше правой по крайней мере на 2^{-n_1}. А значит, n_1 можно уменьшить на единицу, не нарушив неравенства $\sum_i 2^{-n_i} \leqslant 1$, и код не является оптимальным.

Поэтому при выборе оптимального кода достаточно ограничиться кодами с $n_1 = n_2$, и оптимальный среди них соответствует минимуму выражения

$$p_1 n_1 + p_2 n_2 + p_3 n_3 + \ldots + p_k n_k = (p_1 + p_2)n + p_3 n_3 + \ldots + p_k n_k$$

(если через n обозначить общее значение n_1 и n_2) по всем $n, n_3, \ldots, n_k$, для которых

$$2^{-n} + 2^{-n} + 2^{-n_3} + \ldots + 2^{-n_k} \leqslant 1.$$

Перепишем это неравенство как

$$2^{-(n-1)} + 2^{-n_3} + \ldots + 2^{-n_k} \leqslant 1,$$

а выражение, подлежащее минимизации, как

$$(p_1 + p_2) + (p_1 + p_2)(n-1) + p_3 n_3 + \ldots + p_k n_k.$$

Член $(p_1 + p_2)$ постоянен и не влияет на поиск минимума, так что задача сводится к поиску оптимального префиксного кода для $k-1$ букв с вероятностями $p_1 + p_2, p_3, \ldots, p_k$.

Получаем рекурсивный алгоритм: соединить две наиболее редкие буквы в одну (сложив вероятности), найти оптимальный префиксный код для этого случая (рекурсивный вызов), а потом вместо одного кодового слова x для соединённой буквы взять два кодовых слова на единицу длиннее ($x0$ и $x1$); ясно, что префиксность кода при этом не нарушится.

Построенный с помощью такого алгоритма оптимальный префиксный код называется *кодом Хаффмана* для данного набора вероятностей p_i.

Задача 46. Докажите, что кодирование Хаффмана не является сбалансированным.

4.3 Коммуникационная сложность в среднем и энтропия Шеннона

Раньше мы интересовались, сколько битов требуется передать в худшем случае, чтобы вычислить данную функцию $f(x, y)$. То есть, сложность коммуникационного протокола определялась, как его высота (длина максимального пути от корня к листу). Теперь же будем интересоваться количеством битов, передаваемым протоколом в среднем, где усреднение происходит по некоторой вероятностной мере на парах (x, y).

Итак, зафиксируем некоторое распределение μ вероятностей на на парах (x, y). Средним количеством переданных битов для данного протокола P называется сумма $\sum_{(x,y)\in X\times Y} \mu(x, y)c_P(x, y)$, где $c_P(x, y)$ обозначает количество переданных протоколом P битов на входах x, y.

Мерой множества пар будем называть сумму всех вероятностей пар из R. Обозначение: $\mu(R)$. Через α_μ будем обозначать максимальную меру одноцветного для f прямоугольника.

Лемма 3. *Для любого распределения вероятностей μ среднее количество битов переданных любым протоколом P вычисления f не меньше $-\log \alpha_\mu$.*

Доказательство. Рассмотрим случайную величину исходы которой, равны одноцветным прямоугольникам, соответствующим листьям протокола, а вероятность прямоугольника R есть $\mu(R)$. Энтропия Шеннона этой величины не меньше ее минимальной энтропии, которая не меньше $-\log \alpha_\mu$. Протокол P задает беспрефиксный код для своих листьев (код листа есть путь к этому листу). Средняя длина этого кода равна среднему количеству переданных протоколом битов. Осталось воспользоваться неравенством, связывающим среднюю длину беспрефиксного кода и энтропию. □

Задача 47. Пусть f — один из предикатов EQ_n, DISJ_n, GE_n (см. раздел 3). Построить распределение вероятностей на парах x, y, обладающее следующим свойством. Любой коммуникационный протокол, вычисляющий функцию f, в среднем передает не менее n битов. [Указание. Рассмотрите равномерное распределение на трудном множестве.]

Задача 48. Пусть IP_n — есть предикат скалярного произведения (см. раздел 3). Докажите, что при случайном выборе пары x, y (все пары равновероятны) любой коммуникационный протокол, вычисляющий функцию

IP_n, в среднем передает не менее n битов. [Указание. Коммуникационный протокол задает префиксный код для некоторого разбиения матрицы предиката IP_n на одноцветные прямоугольники. Вероятность попадания случайной точки в любой прямоугольник в таком разбиении не больше 2^{-n} (задача 28). Поскольку энтропия Шеннона не меньше минимальной энтропии, средняя длина такого кода не меньше n.]

4.4 Неравенство Макмиллана

{kraft-mcmillan}

До сих пор мы изучали в основном префиксные коды. Оказывается, что переход к произвольным однозначно декодируемым кодам ничего не даёт (с точки зрения сокращения кода):

{mcmillan-inequality}

Теорема 6 (неравенство Макмиллана)**.** *Пусть $c_1, \ldots, c_k$ — кодовые слова однозначно декодируемого кода, а $n_i = l(c_i)$ — их длины. Тогда*

$$\sum_i 2^{-n_i} \leqslant 1.$$

Тем самым (лемма Крафта) можно построить и префиксный код с теми же длинами слов.

Доказательство. Будем считать, что в кодовых словах вместо цифр 0 и 1 используются буквы u и v. (Скажем, кодовые слова 0, 01 и 11 мы запишем как u, uv и vv.) Напишем формальную сумму $(c_1 + \ldots + c_k)$ всех кодовых слов, возведём её в N-ую степень (число N мы потом выберем) и раскроем скобки, не переставляя u и v (как если бы они не коммутировали). Например, для $N = 2$ и для приведённого выше примера получится

$$(u{+}uv{+}vv)(u{+}uv{+}vv) = uu{+}uuv{+}uvv{+}uvu{+}uvuv{+}uvvv{+}vvu{+}vvuv{+}vvvv.$$

Каждое слагаемое в правой части есть соединение некоторых кодовых слов, причём все слагаемые различны (свойство однозначности декодирования). Теперь подставим вместо u и v число $1/2$. В левой части $(c_1 + \ldots + c_k)^N$ превратится при этом в $(2^{-n_1} + \ldots + 2^{-n_k})^N$. Правую часть оценим сверху: если бы в неё входили все возможные слова данной длины t, то получилось бы 2^t членов, каждый из которых равен 2^{-t}, и

сумма равнялась бы 1 (для каждой длины). Поэтому сумма в правой части не превосходит максимальной длины слагаемых, то есть, не больше $N \max(n_i)$.

Теперь видно, что если $\sum 2^{-n_i} > 1$, то при больших N левая часть (растущая экспоненциально) становится больше правой (растущей линейно). □

4.5 Энтропия пары случайных величин

tropy-pair-definition}

При обсуждении энтропии удобно использовать стандартную для теории вероятностей терминологию. Пусть ξ— случайная величина, принимающая конечное число значений $\xi_1, \ldots, \xi_k$ с вероятностями $p_1, \ldots, p_k$. Тогда её *шенноновская энтропия* определяется формулой

$$H(\xi) = p_1(-\log p_1) + \ldots + p_k(-\log p_k)$$

Это определение позволяет говорить об энтропии пары случайных величин ξ и η (определённых на одном и том же вероятностном пространстве), поскольку такая пара сама образует случайную величину. Следующая теорема утверждает, что энтропия пары не превосходит суммы энтропий:

{entropy-pair-bound}

Теорема 7.

$$H(\langle \xi, \eta \rangle) \leqslant H(\xi) + H(\eta)$$

Мы предполагаем, что величины ξ и η принимают конечное число значений, поэтому эта теорема представляет собой некоторое неравенство с суммами логарифмов. Именно, пусть ξ принимает k значений $\xi_1, \ldots, \xi_k$, а η принимает l значений $\eta_1, \ldots, \eta_l$. Тогда величина $\langle \xi, \eta \rangle$ может принимать, вообще говоря, kl значений $\langle \xi_i, \eta_j \rangle$ (некоторые из значений могут не встречаться или встречаться с вероятностью нуль). Распределение вероятностей для пары $\langle \xi, \eta \rangle$, таким образом, задаётся таблицей из k строк и l столбцов: число p_{ij}, стоящее в i-ой строке и j-ом столбце, представляет собой вероятность события «$(\xi = \xi_i)$ и $(\eta = \eta_j)$» (здесь $i = 1, \ldots, k$ и $j = 1, \ldots, l$). Все числа p_{ij} неотрицательны и в сумме равны единице (некоторые из них могут равняться нулю).

Сложив числа в строках, мы получим распределение вероятностей для величины ξ: она принимает значение ξ_i с вероятностью $\sum_j p_{ij}$; эту

сумму удобно обозначить p_{i*}; аналогичным образом η принимает значение η_j с вероятностью p_{*j}, которая есть сумма чисел в j-ом столбце.

Таким образом, сформулированная теорема представляет собой неравенство, справедливое для любой прямоугольной таблицы с неотрицательными числами, в сумме равными единице:

$$\sum_{i,j} p_{ij}(-\log p_{ij}) \leqslant \sum_i p_{i*}(-\log p_{i*}) + \sum_j p_{*j}(-\log p_{*j})$$

(где p_{i*} и p_{*j} определяются как суммы по строкам и столбцам).

Это неравенство в конечном счёте сводится к выпуклости логарифма, но полезно понимать его интуитивный смысл. Если отождествить (забыв про разницу порядка единицы) энтропию с длиной кратчайшего префиксного кода, то теорему можно доказать так: пусть имеются короткие префиксные коды для ξ и η (с кодовыми словами $c_1, \ldots, c_k$ и $d_1, \ldots, d_l$). Тогда можно рассмотреть код для пары $\langle \xi, \eta \rangle$, кодируя значения $\langle \xi_i, \eta_j \rangle$ словом $c_i d_j$ (приписываем d_j справа к c_i без разделителя). Это будет, как легко проверить, префиксный код (чтобы отщепить кодовое слово от бесконечной последовательности, надо сначала отщепить c_i, а потом d_j; в обоих случаях это делается однозначно). Средняя длина этого кода будет равна сумме средних длин кодов для ξ и η. Он не обязан быть оптимальным (ведь и неравенство может быть строгим), но даёт оценку сверху для оптимального кода.

Доказательство. Это рассуждение можно превратить в строгое доказательство, если вспомнить, что при доказательстве теоремы 3 (с. 45) мы установили, что энтропия равна минимуму величины $\sum_i p_i(-\log_2 q_i)$ по всем наборам неотрицательных чисел q_i с единичной суммой. В частности, энтропия пары (левая часть неравенства) есть минимум сумм

$$\sum_{i,j} p_{ij}(-\log q_{ij})$$

по всем наборам q_{ij} неотрицательных чисел с суммой единица. Будем рассматривать не все наборы, а лишь наборы «ранга 1», которые получаются как произведения

$$q_{ij} = q_{i*} \cdot q_{*j}$$

для некоторых наборов неотрицательных чисел q_{i*} и q_{*j}, каждый из которых имеет сумму 1. Тогда $(-\log q_{ij})$ распадётся в сумму $(-\log q_{i*}) +$

$(-\log q_{*j})$, а вся сумма— в две суммы, которые (после суммирования по одному из индексов) окажутся равными

$$\sum_i p_{i*}(-\log q_{i*})$$

и

$$\sum_j p_{*j}(-\log q_{*j})$$

соответственно. Минимумы этих сумм равны $H(\xi)$ и $H(\eta)$.

Таким образом, левая часть неравенства есть минимум некоторой величины по всем наборам, а правая— по наборам ранга 1, откуда и вытекает требуемое неравенство. □

4.6 Условная энтропия

al-entropy-definition}

Условной вероятностью некоторого события B при условии события A называют отношение вероятности события «A и B» к вероятности события A. Это определение имеет смысл, если вероятность события A отлична от нуля. Мотивировка понятна: мы рассматриваем долю исходов, где произошло B, не среди всех исходов, а только среди тех, где произошло A.

Если A— событие, а ξ— случайная величина с конечным числом значений $\xi_1, \ldots, \xi_k$, то можно рассмотреть (помимо вероятностей $\Pr[\xi = \xi_i]$) и условные вероятности $\Pr[(\xi = \xi_i)|A]$. Их сумма тоже равна единице, и получается некоторое новое распределение вероятностей. Его энтропия называется *условной энтропией величины ξ при условии A* и обозначается $H(\xi|A)$, а само это распределение вероятностей можно обозначить $(\xi|A)$.

Задача 49. Покажите, что величина $H(\xi|A)$ может быть и больше, и меньше величины $H(\xi)$. [Указание: распределение $(\xi|A)$ (особенно при малой вероятности события A) мало связано с распределением вероятностей для ξ.]

Неформально говоря, $H(\xi|A)$— это минимально возможная средняя длина кода, если нас интересуют лишь случаи, когда произошло событие A.

Пусть теперь (как и в прошлом разделе) даны две случайные величины ξ и η. Будем предполагать, что для каждой из них все значения

имеют ненулевую вероятность (нулевые можно выбросить). Для каждого значения η_j величины η рассмотрим событие $\eta = \eta_j$ (его вероятность мы обозначали p_{*j}). Рассмотрим условную энтропию величины ξ при условии этого события. Она соответствует распределению вероятностей $i \mapsto p_{ij}/p_{*j}$. Далее усредним эти энтропии с весами, равными вероятностям событий $\eta = \eta_j$. Полученное среднее называют *условной энтропией ξ при известном η* и обозначают $H(\xi|\eta)$. Формально говоря,

$$H(\xi|\eta) = \sum_j \Pr[\eta = \eta_j] H(\xi|\eta = \eta_j)$$

или, в наших обозначениях,

$$H(\xi|\eta) = \sum_j p_{*j} \sum_i \frac{p_{ij}}{p_{*j}} \left(-\log \frac{p_{ij}}{p_{*j}} \right)$$

Основные свойства условной энтропии перечислены в следующей теореме, справедливой для любых случайных величин ξ, η:

{conditional-entropy}

Теорема 8. (**а**) $H(\xi|\eta) \geqslant 0$;

(**б**) $H(\xi|\eta) = 0$ *тогда и только тогда, когда* $\xi = f(\eta)$ *с вероятностью* 1 *для некоторой функции* f (*оговорка про «вероятность* 1*» означает, что мы пренебрегаем значениями, которые имеют нулевую вероятность*).

(**в**) $H(\xi|\eta) \leqslant H(\xi)$

(**г**) $H(\langle \xi, \eta \rangle) = H(\eta) + H(\xi|\eta)$

Доказательство. Первое из этих утверждений очевидно: все $H(\xi|\eta = \eta_j)$ неотрицательны, потому неотрицательна и их взвешенная сумма.

(б) Если взвешенная сумма равна нулю, то все слагаемые с ненулевыми коэффициентами равны нулю, то есть при каждом значении η_j величина $(\xi|\eta = \eta_j)$ имеет нулевую энтропию (и потому принимает лишь одно значение с точностью до событий нулевой вероятности).

Утверждение (в) можно объяснить так: $H(\xi|\eta)$ будет средней длиной оптимального кода для ξ, если разрешить кодировать значения ξ поразному, в зависимости от значения величины η (в каждом случае свой код, который оптимизируется с учётом условных вероятностей). Ясно, что это облегчает построение оптимального кода, поэтому средняя длина получается меньше $H(\xi)$.

Более формально: при каждом j величина $H(\xi|\eta=\eta_j)$ равна минимуму суммы

$$\sum_i \frac{p_{ij}}{p_{*j}}(-\log q_{ij})$$

по всем неотрицательным $q_{1j}+q_{2j}+\ldots+q_{kj}=1$ (мы используем свой набор переменных для каждого j) и потому $H(\xi|\eta)$ равна минимуму суммы

$$\sum_j p_{*j}\sum_i \frac{p_{ij}}{p_{*j}}(-\log q_{ij})$$

по всем таблицам, составленным из неотрицательных чисел q_{ij}, у которых сумма каждого столбца равна единице. Если мы теперь ограничимся таблицами, у которых все столбцы одинаковы, $q_{ij}=q_i$, то сумма превратится в

$$\sum_j p_{*j}\sum_i \frac{p_{ij}}{p_{*j}}(-\log q_i)=\sum_j\sum_i p_{ij}(-\log q_i)=\sum_i p_{i*}(-\log q_i)$$

и её минимум станет равным $H(\xi)$. Поэтому $H(\xi|\eta)\leqslant H(\xi)$.

Наконец, пункт (г) представляет собой равенство, которое непосредственно следует из определений:

$$\begin{aligned}\sum_{i,j} p_{ij}(-\log p_{ij}) &= \sum_j p_{*j}\sum_i \frac{p_{ij}}{p_{*j}}(-\log\frac{p_{ij}}{p_{*j}}-\log p_{*j}) = \\ &\sum_j p_{*j}\sum_i \frac{p_{ij}}{p_{*j}}(-\log\frac{p_{ij}}{p_{*j}})+\sum_j p_{*j}\sum_i\frac{p_{ij}}{p_{*j}}(-\log p_{*j}) = \\ &\sum_j p_{*j}H(\xi|\eta=\eta_j)+\sum_j p_{*j}(-\log p_{*j}) = H(\xi|\eta)+H(\eta).\end{aligned}$$

Теорема доказана. □

Из этой теоремы немедленно вытекает теорема 7 (с. 56). Кроме того, из неё видно, что энтропия пары не меньше энтропии любого из её членов (поскольку условная энтропия неотрицательна). Отсюда легко вытекает такое утверждение:

{no-new-entropy}

Теорема 9. *Пусть ξ — случайная величина с конечным числом значений, а f — функция, определённая на множестве значений величины ξ. Тогда*

$$H(f(\xi))\leqslant H(\xi),$$

где $f(\xi)$— случайная величина, получающаяся применением f к ξ (формально говоря, композиция f и ξ).

С точки зрения наборов чисел переход от ξ к $f(\xi)$ означает, что мы объединяем некоторые значения (складывая соответствующие вероятности).

Доказательство. В самом деле, величина $\langle \xi, f(\xi)\rangle$ имеет в точности то же распределение, что и ξ, поэтому $H(\xi) = H(\xi, f(\xi))$, а энтропия пары не меньше энтропии второго члена пары. □

Задача 50. Укажите прямое доказательство в терминах кодирования и поиска минимума.

Задача 51. В каких случаях неравенство теоремы 9 обращается в равенство?

Величину $I(\alpha : \beta) \rightleftharpoons H(\beta) - H(\beta|\alpha)$ называют *количеством информации в α о β*. Как видно из формулы оно показывает, на сколько уменьшится энтропия β, если становится известным α. Используя равенство $H(\beta|\alpha) = H(\beta, \alpha) - H(\alpha)$, можно заметить, что $I(\alpha : \beta) = H(\beta) + H(\alpha) - H(\beta, \alpha)$. Отсюда следует, что количество информации коммутативно: количество информации в α о β равно количеству информации в β о α. Поэтому часто величину $I(\alpha : \beta)$ называют *взаимной информацией* α и β.

Задача 52. Докажите, что взаимная информация обладает следующими свойствами:
$I(\alpha : \beta) \leqslant H(\alpha)$,
$I(\alpha : \beta) \leqslant H(\beta)$,
$I(f(\alpha) : \beta) \leqslant I(\alpha : \beta)$ для любой функции f.

4.7 Независимость и энтропия

{independence-and-entr

Понятие независимости случайных величин легко выражается в терминах энтропии. Напомним, что величины ξ и η *независимы*, если вероятность события «$\xi = \xi_i$ и $\eta = \eta_j$» равна произведению вероятностей событий $\xi = \xi_i$ и $\eta = \eta_j$ по отдельности. (Переформулировка: если распределение вероятностей ξ относительно условия $\eta = \eta_j$ совпадает с исходным; аналогично и для η относительно $\xi = \xi_i$.) В наших обозначени-

ях независимость означает, что $p_{ij} = p_{i*}p_{*j}$ (матрица вероятностей имеет ранг 1).

{independence}

Теорема 10. *Величины ξ и η независимы тогда и только тогда, когда*

$$H(\langle \xi, \eta \rangle) = H(\xi) + H(\eta),$$

то есть, у α и β нулевая взаимная информация.

Другими словами, критерий независимости состоит в том, что неравенство теоремы 7 обращается в равенство. Используя теорему 8, можно переписать это равенство в виде $H(\xi) = H(\xi|\eta)$ или $H(\eta) = H(\eta|\xi)$.

Доказательство. Логарифм является строго выпуклой функцией: неравенство

$$\log\left(\sum p_i x_i\right) \geqslant \sum p_i \log x_i,$$

справедливое для неотрицательных p_i с единичной суммой и произвольных положительных x_i, обращается в равенство, лишь если все x_i равны (за исключением тех, которые входят с нулевыми коэффициентами p_i).

Отсюда следует, что для любых неотрицательных p_i, в сумме равных единице, минимум выражения

$$\sum p_i(-\log q_i)$$

который берётся по всем неотрицательным q_i, в сумме равным 1, достигается в единственной точке, когда $q_i = p_i$. (Надо уточнить, что при $p_i = 0$ мы полагаем $p_i(-\log q_i) = 0$ и разрешаем q_i быть нулевым.)

Теперь вспомним, что делалось при доказательстве теоремы 7. Минимум по матрицам ранга 1 (при котором правая часть равна сумме энтропий) достигался при

$$q_{ij} = p_{i*} \cdot p_{*j}$$

Если он совпадает с минимумом по всем наборам q_{ij}, который достигается при $q_{ij} = p_{ij}$, то это значит, что имеет место равенство

$$p_{ij} = p_{i*} \cdot p_{*j}$$

и величины ξ и η независимы. □

Задача 53. Докажите, что величины α, β, γ независимы в совокупности (вероятность события $(\alpha = \alpha_i, \beta = \beta_j, \gamma = \gamma_k)$ равна произведению трёх отдельных вероятностей) тогда и только тогда, когда

$$H(\langle \alpha, \beta, \gamma \rangle) = H(\alpha) + H(\beta) + H(\gamma).$$

4.8 «Релятивизация» и информационные неравенства

{elativization}

Доказанные нами утверждения имеют свои «условные» варианты. Например, неравенство

$$H(\langle\xi,\eta\rangle) \leqslant H(\xi) + H(\eta)$$

при добавлении случайной величины α в качестве условия превращается в

$$H(\langle\xi,\eta\rangle|\alpha) \leqslant H(\xi|\alpha) + H(\eta|\alpha)$$

Нового доказательства по существу не требуется, так как при каждом значении α_i случайной величины α выполнено неравенство

$$H(\langle\xi,\eta\rangle|\alpha=\alpha_i) \leqslant H(\xi|\alpha=\alpha_i) + H(\eta|\alpha=\alpha_i)$$

(применяем неравенство для пары к условным распределениям вероятностей случайных величин ξ и η), и остаётся сложить эти неравенства с весами $\Pr[\alpha=\alpha_i]$.

Теперь можно выразить условные энтропии через безусловные, используя формулу $H(\beta|\gamma) = H(\langle\beta,\gamma\rangle) - H(\gamma)$, и привести подобные члены. Получится такое утверждение:

{basic-shannon}

Теорема 11 (базисное неравенство)**.**

$$H(\xi,\eta,\alpha) + H(\alpha) \leqslant H(\xi,\alpha) + H(\eta,\alpha)$$

Для краткости мы опускаем угловые скобки и пишем $H(\xi,\eta,\alpha)$ вместо $H(\langle\xi,\eta,\alpha\rangle)$ или ещё более подробного $H(\langle\langle\xi,\eta\rangle,\alpha\rangle)$.

Аналогичную «релятивизацию» (добавление случайных величин как условий) можно применить и к понятию взаимной информации и определить, скажем, $I(\alpha:\beta|\gamma)$ как

$$H(\alpha|\gamma) + H(\beta|\gamma) - H(\langle\alpha,\beta\rangle|\gamma).$$

Базисное неравенство (теорема 11) утверждает, что $I(\alpha:\beta|\gamma) \geqslant 0$ для любых случайных величин α,β,γ.

Задача 54. Докажите, что $I(\langle\alpha,\beta\rangle:\gamma) \geqslant I(\alpha:\gamma)$

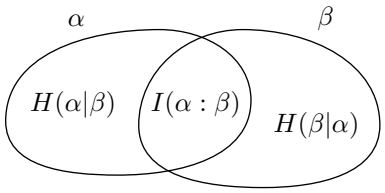
Задача 55. Докажите, что

$$I(\langle\alpha,\beta\rangle : \gamma) = I(\alpha : \gamma) + I(\beta : \gamma|\alpha).$$

Если $I(\alpha : \gamma|\beta) = 0$, говорят, что α и γ *независимы при известном* β, а также что α, β, γ образуют *марковскую цепь* (« прошлое» α связано с «будущим» γ лишь через « настоящее» β).

Задача 56. Докажите, что в этом случае $I(\alpha : \gamma) \leqslant I(\alpha : \beta)$, и потому $I(\alpha : \gamma) \leqslant H(\beta)$.

При решении этих задач полезно использовать диаграммы, похожие на диаграммы Венна. Диаграмма для двух величин состоит из трёх областей, каждой из которых соответствует неотрицательное значение; суммы значений в двух областях слева равна $H(\alpha)$, а в двух областях справа— $H(\beta)$ (рис.4.1).



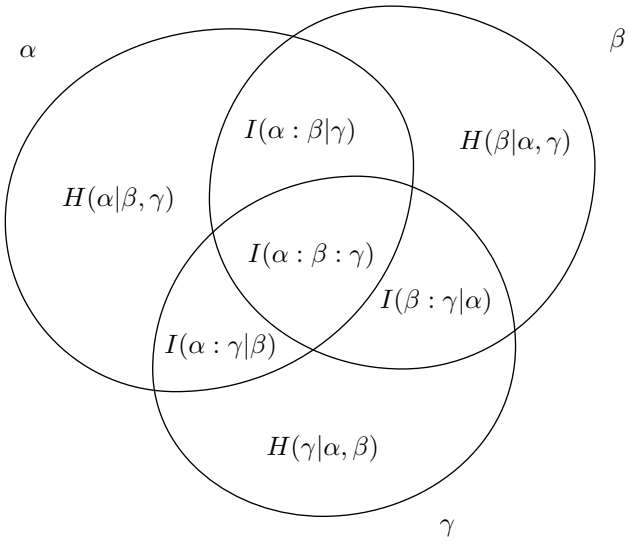
{entropy.1} Рис. 4.1: Энтропии двух случайных величин.

Диаграмма для трёх величин α, β, γ показана на рис. 4.2. Центральной области соответствует значение, которое мы обозначили через $I(\alpha : \beta : \gamma)$; его можно определить как $I(\alpha : \beta) - I(\alpha : \beta|\gamma)$, а также как $I(\alpha : \gamma) - I(\alpha : \gamma|\beta)$ и т.п. — при переходе к безусловным энтропиям получается выражение

$$I(\alpha : \beta : \gamma) = H(\alpha)+H(\beta)+H(\gamma)-H(\alpha,\beta)-H(\alpha,\gamma)-H(\beta,\gamma)+H(\alpha,\beta,\gamma)$$

В отличие от шести других величин на рисунке, $I(\alpha : \beta : \gamma)$ может быть отрицательной. Так будет, например, если величины α и β независимы, но зависимы при известном γ.

Задача 57. Приведите пример таких величин α, β, γ. [Указание. Можно рассмотреть независимые величины α и β, равномерно распределённые на $\{0, 1\}$, и положить $\gamma = \alpha + \beta \bmod 2$.]



{entropy.2} Рис. 4.2: Энтропии трёх случайных величин.

Задача 58. Придумайте алгоритм, который по любому линейному неравенству, содержащему энтропии α, β, γ и кортежей, составленных из них, определяет истинно ли данное неравенство для всех α, β, γ. [Указание. Составьте диаграмму 4.2 и для каждой части диаграммы найдите коэффициент, с которым соответствующая часть появляется в левой части исходного неравенства (слева от знака $\geqslant$). Если все части появляются с неотрицательными коэффициентами, причем коэффициент в центральной части не превосходит суммы коэффициентов своих трех соседей, то неравенство всегда истинно, а иначе нет.]

{th1}

Теорема 12. *Если величины α и β различаются с вероятностью ε, и число различных значений величины α равно a, то выполняется* неравенство Фано*:*

$$H(\alpha|\beta) \leqslant \varepsilon \log a + h(\varepsilon, 1-\varepsilon),$$

где $h(\varepsilon, 1-\varepsilon)$ — энтропия случайной величины с двумя значениями, имеющими вероятности ε и $1-\varepsilon$.

Доказательство. Введём величину γ, которая принимает два значения 0 и 1 при $\alpha \neq \beta$ и $\alpha = \beta$ соответственно. Тогда $H(\alpha|\beta) \leqslant H(\gamma) + H(\alpha|\beta, \gamma)$. Первое слагаемое равно $h(\varepsilon, 1-\varepsilon)$, а второе надо записать как

$$\Pr[\gamma = 0] H((\alpha|\beta)|\gamma = 0) + \Pr[\gamma = 1] H((\alpha|\beta)|\gamma = 1),$$

то есть

$$\Pr[\alpha \neq \beta] H((\alpha|\beta)|\alpha \neq \beta) + \Pr[\alpha = \beta] H((\alpha|\beta)|\alpha = \beta),$$

что не превосходит $\varepsilon \log a + 0$. □

Если случайная величина β не может принимать значения вне множества возможных значений случайной величины α, то неравенство Фано можно усилить, заменив в нем $\log a$ на $\log(a-1)$. В самом деле, условная энтропия $H((\alpha|\beta)|\alpha \neq \beta)$ не превосходит $\log(a-1)$, так как тот исход, который принимает случайная величина β, запрещен для α. Это наблюдение позволяет усилить неравенство Фано и в общем случае.

{th2}

Теорема 13. *Пусть случайная величина α принимает значения в некотором a-элементном множестве A. Пусть значение случайной величины β принадлежит A с вероятностью p, причём условная вероятность $\alpha = \beta$ при условии $\beta \in A$ равна $1 - \varepsilon$. Тогда выполняется* уточнённое неравенство Фано

$$H(\alpha|\beta) \leqslant (1-p)\log a + p\varepsilon \log(a-1) + ph(\varepsilon, 1-\varepsilon),$$

Доказательство. В самом деле, при любом фиксированном исходе $b \in A$ случайной величины β выполнено неравенство

$$H(\alpha|\beta = b) \leqslant \varepsilon(b)\log(a-1) + h(\varepsilon(b), 1-\varepsilon(b)),$$

где $\varepsilon(b)$ означает условную вероятность события $\alpha \neq \beta$ при условии $\beta = b$. Кроме того, имеет место неравенство

$$H(\alpha|\beta \notin A) \leqslant \log a.$$

Складывая эти неравенства с весами $\mathrm{P}[\beta = b]$, мы получаем неравенство

$$H(\alpha|\beta) \leqslant (1-p)\log a + p\varepsilon \log(a-1) + py,$$

где через y обозначено среднее значение функции $h(\varepsilon(b), 1-\varepsilon(b))$ по всем $b \notin A$. Поскольку h является вогнутой функцией (что лекго установить дифференцированием), y не превосходит $h(\varepsilon, 1-\varepsilon)$. □

Если случайная величина α равномерно распределена, то в уточнённом неравенстве Фано достижимо равенство: существует случайная величина β совместно распределённая с α, для которой выполнены условия теоремы 13 и при этом неравенство Фано обращается в равенство

$$H(\alpha|\beta) = (1-p)\log a + p\varepsilon\log(a-1) + ph(\varepsilon, 1-\varepsilon).$$

В самом деле, рассмотрим случайную величину β, условное распределение которой при известном α определено так. Пусть $\alpha = i$. Тогда β принимает некоторое особое значение $\perp$ (не входящее в множество A) с вероятностью $1-p$, значение i с вероятностью $p(1-\varepsilon)$, и равна каждому из остальных $a-1$ элементов A с вероятностью $p\varepsilon/(a-1)$. Тогда все неравенства, использованные в доказательстве теоремы 13, обращаются в равенства. Можно и непосредственно вычислить $H(\alpha|\beta)$ как среднее $H(\alpha|\beta = j)$ по всем исходам j и увидеть, что уточнённое неравенство Фано обращается в равенство. В самом деле, $H(\alpha|\beta = \perp) = \log a$ и

$$H(\alpha|\beta = j) = -(1-\varepsilon)\log(1-\varepsilon) - \varepsilon\log(\varepsilon/(a-1)) = h(\varepsilon, 1-\varepsilon) + \varepsilon\log(a-1)$$

при любом $j \neq \perp$.

Если же α не равномерно распределено, то такого β не существует. В самом деле, для того, чтобы все неравенства, использованные в доказательстве теоремы 13, обратились в равенства, нужно, чтобы все числа $\varepsilon(b)$ были равны друг другу (и значит равны ε), а все условные распределения $\alpha|\beta = j$ при $j \neq \perp$ были симметричными ($\alpha = j$ с вероятностью $1-\varepsilon$, а все остальные значения равновероятны). Кроме того, условные распределения $\alpha|\beta = \perp$ должно быть равномерным. Отсюда следует, что в матрице совместного распределения α, β (значениями α нумеруются строки, а значениями β — столбцы) все строки получаются друг из друга перестановками. А значит сумма в любой строке одна и та же.

Следующая теорема дает еще один пример применения информационных неравенств.

Теорема 14. *Пусть $H(\alpha|\beta, \gamma) = 0$ и $I(\beta : \alpha) = 0$. Тогда $H(\gamma) \geqslant H(\alpha)$. Более того, для любых α, β, γ выполнено $H(\alpha) \leqslant H(\gamma) + I(\beta : \alpha) + H(\alpha|\beta, \gamma)$.*

Это утверждение называют иногда *теоремой Шеннона об идеальном шифре*. (Если агент хочет передать в Центр секретное сообщение α в

виде открытого текста β с помощью заранее согласованного с Центром ключа γ, причём так, чтобы враги, не знающие γ, не получили никакой информации об α, то энтропия ключа должна быть не меньше энтропии сообщения.)

Доказательство. Нарисовав диаграмму случайных величин α, β, γ, нетрудно убедиться, что разница между правой и левой частями последнего неравенства равна $I(\beta : \gamma) + H(\gamma|\alpha, \beta)$, а значит неотрицательна. Можно и не привлекать диаграммы, а вместо этого перенести $I(\beta : \alpha)$ в левую часть. Получится неравенство $H(\alpha|\beta) \leqslant H(\gamma) + H(\alpha|\beta, \gamma)$, которое следует из неравенства $H(\alpha|\beta) \leqslant H(\gamma|\beta) + H(\alpha|\beta, \gamma)$. Последнее неравенство является релятивизацией очевидного неравенства $H(\alpha) \leqslant H(\gamma) + H(\alpha|\gamma)$. □

4.9 Задачи для самостоятельной работы

Задача 59. Пусть вероятности исходов случайной величины есть $1/2$, $1/4$, $1/8$, …, $1/2^n$, $1/2^n$. К чему стремится ее энтропия, когда $n \to \infty$? Тот же вопрос для случайной величины с вероятностями исходов $1/3$, $1/3$, $1/9$, $1/9$, …, $1/3^n$, $1/3^n$, $1/3^n$.

Задача 60. Нарисуйте график функции $p \mapsto h(p, 1-p)$, задающей энтропию случайной величины с двумя исходами, имеющими вероятности $p, 1-p$. Найдите первые три члена ряда Тейлора этой функции в точке $1/2$.

Задача 61. Докажите формулу

$$h(p_1, \ldots, p_n, q_1, \ldots, q_m) = h(p_1 + \cdots + p_n, q_1 + \cdots + q_m) + (p_1 + \cdots + p_n)h(p'_1, \ldots, p'_n) + (q_1 + \cdots + q_m)h(q'_1, \ldots, q'_m),$$

где $p'_i = p_i/(p_1 + \cdots + p_n)$ и $q'_i = q_i/(q_1 + \cdots + q_m)$.

{p-shannon-code}

Задача 62. Докажите, что средняя длина кода Шеннона–Фано (см. с. 51) не превосходит $H + O(1)$. [Указание. Можно по индукции доказывать, что $\sum_i p_i(l(c_i) + \log p_i)$ не превосходит некоторой константы, умноженной на сумму p_i по всем таким i, что i-ый отрезок при каком-то из рекурсивных вызовов оказался в середине.]

Статистическим расстоянием между двумя распределениями вероятностей μ и ν называется максимум по всем событиям A величины $|\mu(A) - \nu(A)|$.

Задача 63. Докажите, что указанный максимум достигается и равен сумме половине суммы $\sum_x |\mu(x)-\nu(x)|$ (суммирование производится по всем исходам x).

Статистическим расстоянием между случайными величинами называется статистическое расстояние между их распределениями.

{statistical-distance}

Задача 64. Пусть даны случайные величины α, β с n возможными исходами. Докажите, что $|H(\alpha)-H(\beta)| \leqslant d\log n+h(d,1-d)$, где d обозначает статистическое расстояние между α, β, а $h(d, 1-d)$ есть энтропия Шеннона случайной величины с вероятностями исходов d и $1-d$. [Указание. Это следует из неравенства Фано. Для его применения надо «спарить» случайные величины α, β, то есть, построить их совместное распределение так, чтобы с вероятностью $1-d$ было выполнено $\alpha = \beta$.]

Задача 65. Докажите, что $I(\langle\alpha,\beta\rangle : \gamma) \geqslant I(\alpha : \gamma)$ и что разность между левой и правой частями равна $I(\beta : \gamma|\alpha)$.

Задача 66. Доказать неравенство $2H(\alpha,\beta,\gamma) \leqslant H(\alpha,\beta)+H(\alpha,\gamma)+H(\beta,\gamma)$.

Задача 67. Докажите следующее обобщение предыдущего неравенства. Пусть $T_1, \ldots, T_k$ — произвольные кортежи, составленные из переменных $\alpha_1, \ldots, \alpha_n$, причем каждая переменная входит ровно в r кортежей. Тогда $rH(\alpha_1, \alpha_2, \ldots, \alpha_n) \leqslant H(T_1)+\cdots+H(T_k)$ (неравенство Шерера (Shearer)).

Глава 5

Кодирование текстов с учетом частотных закономерностей

В разделе 4.2 мы изучили задачу префиксного кодирования букв данного алфавита Σ, минимизирующую среднюю длину кода одной буквы. При этом мы предполагали, что на буквах алфавита задано некоторое распределение вероятностей. Теорема 3 утверждала, что минимальная возможная средняя длина находится в интервале $[H, H+1)$, где H обозначает энтропию этого распределения.

В этом разделе мы изучим следующую задачу. Нам даётся слово (последовательность букв) данной длины m в данном алфавите Σ. Нам нужно передать это слово по каналу, способному за один такт работы передать один бит. На другом конце канала известен алфавит Σ и длина последовательности m. Сколько битов придётся передать в худшем случае?

Точнее, пусть имеется некоторая кодирующая функция E, отображающая слова длины m над n-буквенным алфавитом в двоичные слова длины k и задающая инъективное кодирование множества всех слов длины m (инъективность нужна, чтобы на приёмном конце канала было известно, какое слово передано). Нас интересует наименьшее k, для которого такая функция существует. Ясно, что k должно быть не меньше $\log_2 n^m = m \log_2 n$. В самом деле, дополним все кодовые слова нулями до длины k. Все полученные кодовые слова различны (в силу условия инъективности), и их n^m штук. Поэтому $2^k \geqslant n^m$, что и требовалось установить. С другой стороны, несложно построить инъективное (и даже префиксное) кодирование, для которого коды всех слов имеют длину

$k = \lceil \log_2 n^m \rceil$. Для этого сопоставим каждому слову его номер в лексикографическом порядке, записав его ровно k битами (добавив старшие нули при необходимости).

Таким образом, в рассматриваемой задаче необходимо и достаточно передать $k = \lceil \log_2 n^m \rceil$ бит, столько же, сколько информации по Хартли в каждом слове.

5.1 Кодирование текста с известными частотами букв

Для реальных текстов более экономный код можно получить, учитывая частоты букв в нём. Пусть на приёмном конце известно, что известно, сколько раз встречается каждая буква алфавита в передаваемом слове. Это небольшой объем информации и мы будем далее им пренебрегать. Тогда количество бит, которое необходимо передать, равно двоичному логарифму количества слов той же длины и с теми же частотами букв алфавита. Сколько же таких слов? Оказывается, энтропия Шеннона даёт довольно точную оценку логарифма этого числа.

Теорема 15. *Пусть заданы положительные числа $p_1, \ldots, p_n$ в сумме дающие 1 и число m (длина слова) такое, что все числа $mp_1, \ldots, mp_n$ целые. Обозначим через H энтропию набора $p_1, \ldots, p_n$. Тогда двоичный логарифм количества слов длины m над n-буквенным алфавитом $\{\sigma_1, \ldots, \sigma_n\}$, в которых для всех i количество букв σ_i равно mp_i, равно $mH + O(n \log m)$.*

Доказательство. Количество таких слов задаётся биномиальным коэф-
{page1} фициентом

$$C_m^{k_1,\ldots,k_n} = \frac{m!}{(k_1)! \cdots (k_n)!}$$

По формуле Стирлинга $m! \sim \sqrt{2\pi m}(m/e)^m$. Поэтому

$$\begin{aligned}\log \frac{m!}{(p_1 m)! \cdots (p_n m)!} &= m \log m - p_1 m \log p_1 m - \cdots - p_n m \log p_n m + O(n \log m) \\ &= -p_1 m \log p_1 - \cdots - p_n m \log p_n + O(n \log m) \\ &= Hm + O(n \log m).\end{aligned}$$

□

5.1.1 Кодирование слова его номером в алфавитном порядке

В соответствии с этой теоремой слова с известными частотами букв $p_1, \ldots, p_m$ и данной длиной m можно кодировать Hm битами, где $H = -p_1 \log p_1 - \cdots - p_n \log p_n$ обозначает энтропию Шеннона. Насколько эффективно можно это делать. Оказывается, если кодировать слово его номером в алфавитном порядке на словах той же длины и с теми же частотами букв, то кодирование и декодирование можно выполнить за $O(mn)$ шагов (мы здесь считаем за один шаг мы можем выполнить одну арифметическую операцию над сколь угодно большим целым числом).

Задача 68. Пусть дано слово x длины m над n-буквенным алфавитом. Докажите, что за $O(nm)$ шагов можно найти номер x в лексикографическом порядке на словах той же, что и x, длины и с теми же, что и у x, частотами букв. Докажите, что и, обратно, что за $O(nm)$ шагов можно найти по номеру само слово.

5.1.2 Использование кодов Шеннона–Фано, арифметического и кода Хаффмана

Кодирование с известными частотами букв можно осуществить и с помощью арифметического кода, кода Шеннона–Фано или кода Хаффмана. Это можно делать двумя способами.

Первый способ

Можно вспомнить, что числа $p_1, \ldots, p_n$, на основе которых строится каждый из этих кодов, мы понимали, как частоты букв. Основываясь на этом, вычислим в слове, которое надо закодировать, частоты букв $p_1, \ldots, p_n$. Затем составим префиксный код для букв $\sigma_1, \ldots, \sigma_n$ нашего алфавита и заменим в каждую букву слова на его код. Поскольку префиксный код однозначно декодируем, по так полученному коду можно восстановить исходное слово.

К сожалению этот простой способ не дает той же оценки длины слова, что и кодирование из предыдущего параграфа. В самом деле, длина кода слова при таком подходе будет равна

$$(p_1 l_1 + \cdots + p_n l_n)m,$$

где $l_1, \ldots, l_n$ — длины кодов букв алфавита, а m — длина слова. Величина $p_1 l_1 + \cdots + p_n l_n$ есть средняя длина кода одной буквы, она отличается от энтропии не более, чем на 1. Но при умножении на m эта разница может быть довольно большой, значительно больше, чем разница $O(n \log m)$ из предыдущего раздела. Например, для слова над трехбуквенным алфавитом с равными (по одной трети) частотами всех трех букв, коды букв будут 0, 10, 11, а средняя длина кода одной буквы равна $1/3(1+2+2) = 5/3 \approx 1.67$. При этом энтропия равна $\log_2 3 \approx 1.58$. Разница в примерно $0.09m$ становится существенной при больших значениях m.

Второй способ

Рассмотрим бернуллиевское распределение вероятностей на словах длины m с вероятностями $p_1, \ldots, p_n$ букв $\sigma_1, \ldots, \sigma_n$ нашего алфавита. А именно, в каждой позиции в слове мы выбираем букву σ_i с вероятностью p_i, причем разные буквы независимы. Вероятность любого слова равна произведению $\Pi_{i=1}^n p_i^{k_i}$, где k_i обозначает количество вхождений буквы σ_i в это слово. Теперь применим к полученному распределению вероятностей (слова длины m мы рассматриваем как буквы над новым большим алфавитом) арифметический код или код Шеннона–Фано. Вероятность любого слова длины m с частотами букв $p_1, \ldots, p_n$ равна

$$\Pi_{i=1}^n p_i^{p_i m} = 2^{-Hm}.$$

По задаче 42 длина арифметического кода кода любого такого слова не больше минус логарифма его вероятности плюс константа, а значит не больше $Hm + O(1)$. Для кода Шеннона–Фано то же самое верно по
{p57} теореме 5.

Задача 69. Пусть рациональные числа $p_1, \ldots, p_n$ фиксированы. Докажите, что за полиномиальное от m число шагов по данному слову длины m можно найти его код для арифметического кодирования, соответствующего бернуллиевскому распределению вероятностей на словах длины m. Докажите, что и обратно, по коду за полиномиальное время можно найти сообщений.

5.2 Сбалансированные слова

{frequency}

На практике частоты вхождений букв в передаваемое слово известны лишь приблизительно. В этом разделе изучается, насколько это увеличивает длину кода.

Итак, пусть заданы положительные числа $p_1, \ldots, p_n$ в сумме дающие 1 (приблизительные частоты букв). Пусть дано еще число δ — максимальное допустимое отклонение количества вхождений i-ой буквы от mp_i. Будем называть слово (длины m) *δ-сбалансированным*, если для всех букв a_i из алфавита Σ количество вхождений a_i в слово отличается от mp_i не более, чем на δ.

Чему равно наименьшее такое k, для которого имеется инъективное кодирование δ-сбалансированных слов с максимальной длиной кода k. Ясно, что наименьшее такое k равно двоичному логарифму количества δ-сбалансированных слов. Оценим сверху это количество.

{frequency-code}

Теорема 16. *Двоичный логарифм количества δ-сбалансированных слов длины m не превосходит $mH + O(\delta)$, где H обозначает энтропию случайной величины с вероятностями исходов $p_1, \ldots, p_n$. Константа в $O(\delta)$ зависит от $p_1, \ldots, p_n$ (которые предполагаются фиксированными).*

Доказательство. Рассмотрим следующее распределение вероятностей на словах длины m над Σ. В каждой позиции в слове мы выбираем букву a_i с вероятностью p_i, причем разные буквы независимы. Вероятность любого слова равна произведению $\Pi_{i=1}^n p_i^{k_i}$, где k_i обозначает количество вхождений буквы a_i в это слово. Для δ-сбалансированных слов $k_i = mp_i \pm \delta$, поэтому вероятность равна

$$\Pi_{i=1}^n p_i^{mp_i \pm \delta} = \Pi_{i=1}^n 2^{mp_i \log p_i \pm \delta \log p_i} = 2^{-Hm + O(\delta)}.$$

Сумма вероятностей всех сбалансированных слов не может превышать единицы, поэтому их количество не может быть больше $2^{Hm+O(\delta)}$. □

По этой теореме δ-сбалансированные слова можно закодировать $mH + O(\delta)$ битами. За сколько шагов можно кодировать и декодировать сообщения при таком кодировании? Эта задача сводится к нахождению по слову x его номера в лексикографическом порядке на словах с теми же, что и у x частотами букв и, обратно, самого слова по его номеру.

Инъективное кодирование δ-сбалансированных слов с максимальной длиной кода $mH + O(\delta)$ можно осуществить и с помощью арифметического кода или кода Шеннона–Фано (но не кода Хаффмана). А именно,

рассмотрим распределение вероятностей, использованное в доказательстве теоремы 16. Будем рассматривать слова длины m в алфавите Σ как буквы одного алфавита и рассмотрим арифметическое кодирование (или код Шеннона–Фано) букв этого алфавита. Вероятность любого δ-сбалансированного слова равна $2^{-Hm+O(\delta)}$. По задаче 42 и теореме 5, длина кода любого сбалансированного слова не превосходит $Hm + O(\delta)$.

Теперь обсудим вопрос, насколько оценка минимального k в теореме 16 точная. Оказывается, что минимальное k меньше $Hm - O(\delta)$, то есть оценка точна с точностью $O(\delta)$.

Теорема 17. *Пусть $\delta \geqslant 1$. Тогда двоичный логарифм количества δ-сбалансированных слов длины m не меньше $mH - O(\log m)$, где H обозначает энтропию случайной величины с вероятностями исходов $p_1, \ldots, p_n$. Константа в $O(\log m)$ зависит от $p_1, \ldots, p_n$ (которые предполагаются фиксированными).*

Доказательство. Сначала докажем, что найдутся целые числа $k_1, \ldots, k_n$ в сумме равные m и такие что k_i отличается от $p_i m$ менее, чем на 1. Для этого индукцией по i подберём такие целые $k_1, \ldots, k_i$, что разность сумм $k_1 + \cdots + k_i$ и $p_1 m + \cdots + p_i m$ строго меньше 1. При выборе k_{i+1} округлим $p_{i+1} m$ вверх, если текущая разность отрицательна и вниз, иначе. Разность сумм $k_1 + \cdots + k_n$ и $p_1 m + \cdots + p_n m = m$ будет строго меньше 1, а поскольку оба числа целые, она будет равна нулю.

Теперь докажем, что логарифм количества слов длины m, содержащих $k_1, \ldots, k_n$ букв $a_1, \ldots, a_n$, не меньше $mH - O(\log m)$ (поскольку $\delta \geqslant 1$, все они δ-сбалансированы). Это количество задаётся биномиальным коэффициентом

$$C_m^{k_1,\ldots,k_n} = \frac{m!}{(k_1)! \cdots (k_n)!}$$

Как мы уже видели, логарифм этого числа равен $k_1 \log(m/k_1) + \cdots + k_n \log(m/k_n) + O(\log m)$, то есть, $mH(\beta) + O(\log m)$, где β обозначает случайную величину с m исходами и вероятностями исходов $k_1/m, \ldots, k_n/m$. Нам остаётся показать, что разница между $H(\alpha)$ и $H(\alpha)$ есть $O(\log m/m)$.

По построению распределение случайной величины β близко к распределению случайной величины α. А именно, статистическое расстояние между ними не больше n/m. Вспомним (задача 64), что в этом случае их энтропии отличаются не более, чем на $(n/m) \log_2 n + h(n/m, 1 - n/m)$

(если $n > m$, то второе слагаемое надо заменить на 1). Поскольку n считается константой, первое слагаемое здесь $O(\log m/m)$. Второе же слагаемое можно оценить, воспользовались тем, что $\log(1-z) = O(z)$ при $z \to 0$. Из этого следует, что

$$h(n/m, 1-n/m) \leqslant \frac{n}{m}\log\frac{m}{n} - \log(1-\frac{n}{m}) = O\Big(\frac{\log m}{m}\Big) + O\Big(\frac{n}{m}\Big).$$

□

На практике описанные выше кодирования применяются для фиксированного достаточно большого m. Если длина исходной последовательности больше m, то ее разрезают на блоки длины m и кодируют блоки по отдельности. Такое кодирование называют *блочным*.

Задача 70. Пусть $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_n$ и $i \leqslant m$.
(а) Докажите, что

$$h(p_1/(p_1+\cdots+p_i), \ldots, p_i/(p_1+\cdots+p_i)) \leqslant h(p_1, \ldots, p_n).$$

(Постарайтесь сделать это без использования явных формул с логарифмами.)

(б) Докажите, что

$$h(p_1, \ldots, p_i, \varepsilon) \leqslant h(p_1, \ldots, p_n)(1-\varepsilon) + h(\varepsilon, 1-\varepsilon),$$

где $\varepsilon = p_{i+1} + \cdots + p_n$.

Задача 71. Пусть $p_1, p_2, \ldots, p_n$ — вероятности букв $a_1, a_2, \ldots, a_n$, а p — действительное число от 0 до 1. Докажите, что существует кодирующая и декодирующая функции E, D такие, что с вероятностью не менее p выполнено $D(E(a_i)) = a_i$ и при этом средняя длина $E(a_i)$ не превосходит $ph(p_1, p_2, \ldots, p_n) + 2$.

5.3 Учёт частот пар, троек и т.д.

Для реальных текстов кодирование на основе частот букв далеко от оптимального, поскольку не учитывает зависимостей между соседними буквами. Например, частота вхождения слога `ла` в «среднестатистический» русский текст больше, чем произведение частот вхождений букв `л` и `а` по отдельности. Аналогичное справедливо для троек букв и так

далее. Поэтому представляется разумным обобщить теорему Шеннона и коды, использованные в ней, в следующем направлении.

Пусть дано слово x длины m над n-буквенным алфавитом. Для каждой пары букв a, b обозначим через $k(ab)$ количество вхождений биграммы ab в x и обозначим через $k(a)$ сумму всех чисел $k(ab)$. Рассмотрим совместно распределённые случайные величины α, β, получаемые следующим образом: выбираем случайным образом в слове x две подряд идущих буквы и обозначаем первую через α, а вторую через β. Иными словами,

$$\mathsf{P}[\alpha = a, \beta = b] = p(ab) = k(ab)/(m-1).$$

Обозначим через N количество *правильных* слов x' длины m — таких, у которых те же частоты всех биграмм, что у исходного слова x.

{th529}

Теорема 18. *(1) Число N не превосходит $2^{(m-1)H(\beta|\alpha)+\log n}$. (2) Если для всех биграмм ab, входящих в x, количество вхождений ab в x не меньше n, то N не меньше $2^{(m-1)H(\beta|\alpha)-O(n^2 \log m)}$, где константа в O-обозначении абсолютная.*

Доказательство. (1) Для оценки N сверху применим тот же метод, что и в предыдущем разделе. А именно, для каждой буквы a обозначим через $p(a)$ частоту вхождений буквы a в слово x (без учета последней буквы):

$$p(a) = \sum_b p(ab).$$

Обозначим также частоту вхождений в x буквы b после буквы a через

$$p(b|a) \rightleftharpoons \frac{p(ab)}{p(a)}.$$

Далее рассмотрим следующее распределение вероятностей μ на словах длины m:

{f-distr-1}

$$\mu(y_1 \dots y_m) = \frac{1}{n} \prod_{i=1}^{m-1} p(y_{i+1}|y_i). \tag{5.1}$$

Нетрудно убедиться, что это в самом деле распределение вероятностей (сумма указанных чисел равна 1); оно соответствует следующему процессу порождения случайного слова. Первая буква выбирается случайно с равномерным распределением (множитель $1/n$), а затем каждая следующая буква берется в зависимости от предыдущей по правилам марковской цепи: вероятность появления b после равна $p(b|a)$.

Если слово x' правильно, то

$$\mu(x') = \frac{1}{n}\prod_{a,b} p(b|a)^{(m-1)p(ab)} = 2^{-\log n+(m-1)\sum_{a,b} p(a)\log p(b|a)} = 2^{-\log n-(m-1)H(\beta|\alpha)}.$$

Количество правильных слов не превосходит обратного к этому числу, а значит не больше $2^{(m-1)H(\beta|\alpha)+\log n}$.

(2) Для оценки N снизу рассмотрим ориентированный граф, с кратными ребрами и петлями, вершины которого суть буквы исходного алфавита Σ, и между вершинами a и b имеется $k(ab)$ ребер. (Такого вида графы называются графами де Бройна (de Bruijn).) Правильные слова соответствуют эйлеровым путям в этом графе (путям, которые ровно по одному разу проходят через все ребра). Нам нужно доказать, что количество эйлеровых путей в этом графе велико.

Сначала приведем, простое, но неправильное рассуждение. Выберем в качестве начала эйлерова пути начальную букву $x[1]$ исходного слова x и будем рассматривать в каждой вершине все возможные способы продолжения этого пути. (Продолжения пути с помощью добавлений вершин σ и τ считаются разными, если $\sigma \neq \tau$.) Сколько путей можно построить таким образом? Зафиксируем произвольную вершину a в графе. Количество способов продолжения пути в вершине a равно

$$C_{k(a)}^{k(ab_1),\ldots,k(ab_n)}.$$

Если мы перемножим эти числа для всех зафиксированных вершин, то мы и получим общее количество путей. В самом деле, идя вдоль пути, можно увидеть, какое направление выбирает путь в каждой вершине при каждом её проходе. Поэтому разным вариантам продолжения соответствуют разные эйлеровы пути.

Для оценки величины

$$C_{k(a)}^{k(ab_1),\ldots,k(ab_n)}.$$

воспользуемся формулой Стирлинга, в соответствии с которой

$$d! \sim (d/e)^d\sqrt{2\pi d}.$$

Из этой оценки следует, что

$$\begin{aligned} C_{k(a)}^{k(ab_1),\ldots,k(ab_n)} &\geqslant 2^{k(a)\cdot H(p(b_1|a),\ldots,p(b_n|a))-(1/2)\log(k(ad_1)\cdots k(ad_n))-O(1)} \\ &\geqslant 2^{k(a)\cdot H(p(b_1|a),\ldots,p(b_n|a))-(n/2)\log m-O(1)}. \end{aligned}$$

Поэтому общее количество эйлеровых путей в графе не меньше произведения правых частей этих неравенств по всем вершинам a, равного

$$2^{(m-1)H(\beta|\alpha)-(n^2/2)\log m-O(n)}.$$

Что неправильно в этом рассуждении? Нельзя произвольным образом выбирать способ прохождения каждой вершины, поскольку алгоритм, действующий таким образом, может прийти в вершину, из которой уже не сможет выйти, до того, как пройдены все ребра. (Нетрудно видеть, что этой вершиной может быть только последняя буква слова x.) Другими словами, не каждому способу прохождения вершин соответствует некоторый эйлеров путь. Нам нужно объяснить, почему количество способов, которым всё-таки соответствует путь, составляет существенную долю от общего числа.

Для этого докажем, что в нашем графе имеется цикл, проходящий через каждую вершину от одного до $n-1$ раз. Наш ориентированный граф слабо связен (любая пара различных вершин a, b связана неориентированным путем — в качестве этого пути можно взять часть эйлерова пути, задаваемого исходным словом). Выбросим из нашего графа ребра любого простого пути, соединяющие первую букву x с последней. Граф останется связным, поскольку между любыми соседними вершинами не менее n ребер. В полученном графе входная степень любой вершины равна выходной. Любой слабо связный граф, в котором входная степень любой вершины равна ее выходной степени, является и сильно связным. Действительно, иначе отношение «a достижимо из b ориентированным путем» задавало бы пред-порядок, в котором более одного класса эквивалентности (a эквивалентно b, если они достижимы друг из друга ориентированным путем). Тогда в вершины любого максимльного класса эквивалентности входит в совокупности больше ребер, чем выходит.

Поскольку наш граф сильно связен, в нем существует цикл C, проходящий через все вершины. Рассмотрим самый короткий такой цикл и докажем, что каждая вершина встречается в нем менее n раз. Допустим некоторая вершина встречается n (или более) раз. Её вхождения разбивают цикл на n участков. Каждая из остальных $n-1$ вершин встречается на одном из участков — выберем любой один такой участок. Хотя бы один участок окажется не выбран, его можно без ущерба выбросить из цикла.

Зафиксируем любой такой цикл и модифицируем рассмотренный выше жадный алгоритм следующим образом. Сначала мы проходим по лю-

бому простому пути, соединяющему первую букву x с последней. Затем удаляем пройденные ребра из графа, а также все ребра зафиксированного цикла. Всего из каждой вершины окажется удаленными не более n ребер. В оставшемся графе степени всех вершин четны. После мы продолжаем путь жадным образом (из последней буквы слова x) до тех пор, пока это возможно. Рано или поздно мы вернемся в исходную вершину, исчерпав все ребра, исходящие из нее. После этого мы, двигаясь вдоль цикла, перемещаемся в первую вершину, из которой есть непройденные ребра. Затем продолжаем движение жадным образом по непройденным ребрам. И так далее, пока все ребра не окажутся пройденными. Если после этого цикл еще не будет пройден, то проходим его до конца.

Общее количество путей, которые можно простроить таким образом равно произведению по всем вершинам v графа чисел

$$C_{k'(a)}^{k'(ab_1),\ldots,k'(ab_n)},$$

где через $k'(ab_i)$ обозначено количество ребер графа из a в b_i, которые не входят в пройденный сначала простой путь из первой буквы x в последнюю и в зафиксированный цикл. По построению $k'(ab_i) \geqslant k(ab_i) - n$. При уменьшении d_i на единицу число

$$C_{d_1+\cdots+d_n}^{d_1,\ldots,d_n}$$

уменьшается не более, чем в $d_1 + \cdots + d_n \leqslant m$ раз. Поэтому число эйлеровых циклов не более чем в m^{n^2} раз меньше того, которое мы получили неправильным рассуждением. Поэтому общее количетсво эйлеровых путей не меньше $2^{(m-1)H(\beta|\alpha)-(n^2/2+n^2)\log m-O(n)}$. □

В условии доказанной теоремы предполагается, что числа $k(ab)$ являются количеством диграмм в данном нам слове x. Возникает естественный вопрос: для каких наборов чисел $k(ab)$ такое слово x существует? Имеется очевидное необходимое условие в терминах ориентированного граф, вершинами которого являются буквы алфавита, и из вершины a в вершину b ведет $k(ab)$ рёбер. В этом графе входная степень каждой вершины, кроме быть может двух, должна быть равной её выходной степени, а для двух исключительных вершин они должны различаться на 1. Кроме того, граф должен быть слабо связен. Можно убедиться, что это условие является и достаточным: любой такой граф имеет эйлеров путь. В самом деле, начнем этот путь в той вершине, для которой

выходная степень на 1 больше входной (и в любой вершине, если таких вершин нет). Будем любым способом продолжать этот цикл до тех пор, пока не придем в вершину, из которой не выходит непройденных ребер. Эта вершина обязана иметь входную степень на единицу больше входной в первом случае и исходной вершиной во втором. Если все ребра оказались пройденными, то эйлеров путь построен. Иначе в силу слабой связности графа на построенном пути найдется вершина, из которой ведет хотя бы одно непройденное ребро. В этой вершине построенный цикл можно разорвать и добавить к нему еще один цикл и так далее, пока не включим в него все рёбра.

Напомним, что длина кода на основе частот букв примерно равна $mH(\beta) \approx mH(\alpha)$. Поскольку $H(\beta|\alpha) \leqslant H(\beta)$, построенный код не длиннее кода на основе частот букв, а при больших m и при наличии зависимости между соседними буквами он и строго короче. Например, новый код значительно лучше старого для слова $x = 01010101\ldots$. Частоты обеих букв в этом слове равны $1/2$, поэтому кодирование на основе частот букв дает кодовое слово длины m. С другой стороны $H(\beta|\alpha)$ равно нулю, поскольку каждая позиция в этом слове однозначно определяется предыдущей. Поэтому новый код будет иметь длину, пренебрежимо малую по сравнению с m.

Как и раньше, если частоты диграмм неизвестны, то их можно написать в начало кода — длина кодового слова увеличится незначительно, если m достаточно велико. Аналогичным образом можно учитывать зависимости между тройками букв. Для троек получаемый код имеет длину примерно $mH(\gamma|\beta\alpha) \leqslant mH(\beta|\alpha)$, где $\alpha\beta\gamma$ обозначает случайно выбранное подслово длины 3 в исходном слове x. Точно так же можно учитывать зависимости между l подряд идущими буквами для произвольного l. Чем больше l, тем более экономичный код мы получим. Только надо иметь в виду, что если частоты комбинаций заранее не известны, то при большом l дополнительная информация о них в размере $O(n^l \log n)$ бит может стать больше получаемой экономии. Чтобы в этом случае в самом деле получалась экономия, m должно быть значительно больше n^l.

5.3.1 Разбиение на блоки

Частоты l-буквенных сочетаний можно учитывать и другим образом: разбить слово x на блоки длины l, рассмотреть каждый из них как бук-

ву расширенного алфавита и применить кодирование на основе частот букв к полученному слову $\bar{x}$. Предположим, что для любого l-буквенного сочетания w частота «буквы» w в слове $\bar{x}$ будет примерно та же, что и частота блока w в слове x, если рассматривать все его подслова длины l (а не только те, начальные позиции которых кратны l). Тогда длина кода, который мы получим кодируя слово $\bar{x}$, будет примерно равна длине слова $\bar{x}$, умноженной на $H(X)$, где через $X = X_1 \dots X_l$ обозначена энтропия случайной величины, равной случайно выбранному подслову длины l в нашем слове. То есть, одна буква исходного слова x кодируется в среднем $H(X_1 \dots X_l)/l$ битами.

Возникает вопрос, что более выгодно, учитывать частоты блоков длины l так, как это было сделано в предыдущем разделе, или разбивать слово на блоки длины l и использовать кодирование на основе частот для алфавита, состоящего из всевозможных блоков длины l? В первом случае мы потратим примерно $H(X_l|X_1 \dots X_{l-1})$ бит на одну букву исходного слова, а во втором — $H(X_1 \dots X_l)/l$. Интуитивно кажется, что первое более выгодно. В следующем разделе мы обсуждаем, при каких предположениях это в самом деле так.

5.3.2 Стационарные источники.

Пусть дана бесконечная последовательность случайных величин $X_1, X_2, \dots$ со значениями в данном алфавите такая, что распределение случайной величины $X_i, X_{i+1}, \dots, X_{i+l}$ зависит только от l (но не от i). Такие последовательности называются *стационарными источниками* букв данного алфавита.

Задача 72. Пусть $X_0, X_1, X_2, \dots$ состояния марковской цепи из определения 1, в моменты времени $0, 1, 2, \dots$. Докажите, что $X_0, X_1, X_2, \dots$ является стационарным источником.

Итак, пусть дан стационарный источник $X_0, X_1, X_2, \dots$. Что можно сказать о соотношениях между величинами

$$H(X_l|X_1 \dots X_{l-1}) \quad \text{и} \quad \frac{H(X_1 \dots X_l)}{l} \quad ?$$

Ответ дается в следующих задачах.

Задача 73. Докажите, что

$$H(X_l|X_1, X_2, \dots, X_{l-1}) \geqslant H(X_{l+1}|X_1, X_2, \dots, X_l).$$

Это означает, что с ростом l экономность первого способа улучшается (если частоты известны).

Задача 74. Докажите, что

$$\frac{H(X_1, X_2, \ldots, X_l)}{l} \geqslant H(X_l | X_1, X_2, \ldots, X_{l-1}).$$

Это означает, что первый способ учета частот l-буквенных сочетаний не хуже второго.

Задача 75. Докажите, что

$$\frac{H(X_1, X_2, \ldots, X_l)}{l} \leqslant \frac{H(X_1, X_2, \ldots, X_{l+1})}{l+1}.$$

Это означает, что с ростом l экономность второго способа улучшается (если частоты известны).

Задача 76. Докажите, что

$$\lim_{l\to\infty} \frac{H(X_1, X_2, \ldots, X_l)}{l} = \lim_{l\to\infty} H(X_l | X_1, X_2, \ldots, X_{l-1}).$$

То есть, при очень больших l оба способа дают код примерно одной длины.

5.4 Неинъективные кодирования

Допустим сообщение, которое надо передать, получено в результате вероятностного процесса. Нам нужно придумать кодер и декодер, обеспечивающие передачу сообщения, при этом разрешается небольшая вероятность ошибки ε. С этой вероятностью раскодированное сообщение может отличаться от исходного. Поэтому кодирование может быть неинъективным. Мы хотим, чтобы длина кода была поменьше.

Сначала рассмотрим простейший случай — бернуллиевские источники.

5.4.1 Теорема Шеннона о бесшумном канале.

Пусть опять фиксированы числа $p_1, \ldots, p_n$ и кодируемая последовательность x длины m выбирается случайно: в каждой позиции в слове x мы

выбираем букву a_i с вероятностью p_i, причем разные буквы независимы. Множество кодовых слов опять должно быть префиксным (чтобы было ясно, когда передача кодового слова закончена), но разные слова могут иметь один код (так что однозначное декодирование невозможно). Но при декодировании нам разрешается давать неправильный ответ в вероятностью, не превосходящей некоторого ε. Оказывается, что и в этой ситуации минимально возможная длина кода примерно равна mH и слабо зависит от ε.

Уточним постановку задачи. Пару функций E, D будем называть *ε-корректным* m, k-кодированием, если E отображает слова длины m над n-буквенным алфавитом в двоичные слова длины не больше k, а D, наоборот, двоичные слова длины не больше k в слова длины m над n-буквенным алфавитом и при этом вероятность события $D(E(x)) \neq x$ не превосходит ε, и кроме того, множество значений E есть префиксное множество.

Теорема 19. *Для всех $p_1, \ldots, p_n$ и всех положительных ε найдется такое d, что для почти всех m существует ε-корректное m, k-кодирование с $k \leqslant mH(\alpha) + d\sqrt{m}$. Обратно, для всех $p_1, \ldots, p_n$ и всех $\varepsilon < 1$ найдётся константа d такая, что для всех достаточно больших m и всех ε-корректных m, k-кодирований выполнено $k \geqslant mH(\alpha) - d\sqrt{m}$.*

Как видно из формулировки, главный член $mH(\alpha)$ не зависит от допустимой вероятности ошибки, от нее зависит лишь добавочный член меньшего порядка. Это утверждение называют *Shannon noiseless coding theorem* (теорема Шеннона о кодировании без помех). Имеется в виду, что в теореме изучается передача информации по каналу, не вносящему помех.

Доказательство. Зафиксируем $p_1, \ldots, p_n, \varepsilon$. Сначала докажем первое утверждение. Наиболее естественным было бы использовать теорему 3 для алфавита, состоящего из всех возможных сообщений (слов длины m над Σ). Энтропия распределения вероятностей на сообщениях равна mH, поскольку сообщение состоит из независимых одинаково распределённых букв, и энтропия каждой буквы равна H. Поэтому существует префиксный код со средней длиной меньше $mH + 1$. Но, к сожалению, из того, что средняя длина не больше $mH + 1$ еще не следует, что длина кода меньше $k \leqslant mH(\alpha) + d\sqrt{m}$ с близкой к 1 вероятностью.

Поэтому мы пойдём другим путём. Укажем сразу, какие именно слова будут правильно декодироваться. Это будут $O(\sqrt{m})$-сбалансированные слова. Константа c в $O(\sqrt{m})$ выбирается так, чтобы суммарная вероятность всех не $c\sqrt{m}$-сбалансированных слов не превосходила ε.

Докажем, что такая константа c существует. По неравенству Чебышёва вероятность того, что количество вхождений i-ой буквы в случайном слове отличается от $p_i m$ не более чем на δ, не превосходит $\frac{m}{\delta^2}$. При $\delta = c\sqrt{m}$ эта вероятность не превосходит $1/c^2$, поэтому можно выбрать настолько большое c, чтобы она была меньше ε/n (для всех i). При таком c вероятность несбалансированности будет меньше ε для всех m.

По теореме 16 логарифм количества $O(\sqrt{m})$-сбалансированных слов есть $mH + O(\sqrt{m})$. Поэтому можно определить значение E на i-ом сбалансированном слове (в лексикографическом порядке) как i-ое двоичное слово длины $k = mH + O(\sqrt{m})$. А затем определить D, как обратное к E отображение. Как будет определено E на несбалансированных словах, неважно.

Теперь докажем второе утверждение. Оно доказывается «обратными рассуждениями». Пусть имеется ε-корректное m, k-кодирование E, D. Опять рассмотрим $c\sqrt{m}$-сбалансированные слова. Но на этот раз выберем c таким большим, чтобы суммарная вероятность несбалансированных слов была меньше $(1 - \varepsilon)/2$. Тогда суммарная вероятность $c\sqrt{m}$-сбалансированных слов x, для которых $D(E(x)) = x$, будет не меньше $(1-\varepsilon)/2$. Действительно, по условию доля слов, для которых $D(E(x)) = x$, не меньше $1 - \varepsilon$. Причем, не более, чем $(1 - \varepsilon)/2$ из них не являются $c\sqrt{m}$-сбалансированными.

Теперь оценим сверху суммарную вероятность сбалансированных слов x, для которых $D(E(x)) = x$. Как мы видели в доказательстве теоремы 16, вероятность каждого сбалансированного слова не больше $2^{-mH+O(\sqrt{m})}$. По условию слов, для которых $D(E(x)) = x$, не больше 2^k (действительно, для каждого двоичного слова u длины k существует не более одного x, для которого $D(u) = x$, $E(x) = u$). Поэтому суммарная вероятность сбалансированных слов x, для которых $D(E(x)) = x$, не превосходит $2^{k-mH+O(\sqrt{m})}$.

Соединив верхнюю и нижнюю оценки, мы получаем

$$(1 - \varepsilon)/2 \leqslant 2^{k-mH+O(\sqrt{m})},$$

откуда и следует утверждение теоремы. □

Первое утверждение теоремы можно доказать и с помощью арифметического кода (см. с. 62). А именно, рассмотрим слова длины m в n-буквенном алфавите, как новые буквы, и рассмотрим на буквах нового алфавита то же самое распределение вероятностей, что и в теореме. Напомним, что при арифметическом кодировании длина l кодового слова буквы связана с её вероятностью p неравенством $l < -\log p + 2$. Поэтому длины кодовых слов всех сбалансированных слов не превосходят $mH + O(\sqrt{m})$. Остальные слова кодирующая функция может отображать куда угодно. Поэтому будет достаточно $mH + O(\sqrt{m})$ бит.

5.4.2 Марковские цепи

Пусть фиксировано неотрицательное целое l и пусть для каждого слова w длины $l+1$ в n-буквенном алфавите задано неотрицательное число $\pi(w)$, сумма которых (по всем словам длины $l+1$) равна 1. Пусть еще эти числа удовлетворяют условию *стационарности*:

$$\sum_a \pi(ua) = \sum_b \pi(bu).$$

Число, задаваемое этой формулой мы будем обозначать через $\pi(u)$. Заметим, что частоты вхождения $l+1$-буквенных блоков в любую фиксированную последовательность «почти» удовлетворяют этому равенству. Точнее, левая часть равенства есть частота вхождения слова w, во всех позициях, кроме последней, а правая часть — частота вхождения u во всех позициях, кроме первой. Поэтому левая и правая часть могут отличаться максимум на $1/(m-l-2)$.

{def-shannon-general}

Определение 1. Рассмотрим марковскую цепь, состояниями которой являются слова w длины $l+1$ с положительным $\pi(w)$, а переходы имеют вид $au \to ub$, где a, b — произвольные буквы, а u — l-буквенное слово. Вероятность такого перехода по определению есть

$$\frac{\pi(ub)}{\pi(u)}$$

(она не зависит от a).

Как нетрудно видеть, распределение π является стационарным для этой марковской цепи:

$$\sum_a \pi(au)\frac{\pi(ub)}{\pi(u)} = \pi(ub)$$

для всех $l+1$-буквенных блоков ub. Эта марковская цепь задает следующий процесс порождения случайного слова длины $m \geqslant l+1$. Первые $l+1$ букв берутся случайным образом с распределением π. А затем мы применяем марковскую цепь для получения $l+2$-ой буквы, потом опять применяем марковскую цепь и так $m-l-1$ раз. Обозначим так определенное распределение на словах длины m через μ. Будем предполагать еще, что марковская цепь *эргодична*. Это означает, что найдется такое N, что для всех состояний u, v марковская цепь за N шагов с положительной вероятностью переходит из u в v.

Будем называть пару функций E, D m, k, ε-кодированием, если E отображает слова длины m над n-буквенным алфавитом в двоичные слова длины k, а D, наоборот, двоичные слова длины k в слова длины m над n-буквенным алфавитом и при этом μ-вероятность события $D(E(x)) \neq x$ не превосходит ε. Рассмотрим случайные величины α, β, получаемые в результате следующего процесса. Генерируем случайное слово длины $l+1$ с распределением π. Последняя его буква есть β, а слово, полученное отрезанием последней буквы есть α.

{th-shannon-general}

Теорема 20. *Для всех положительных ε найдется такое c, что для почти всех m существует m, k, ε-кодирование с $k \leqslant mH(\beta|\alpha) + c\sqrt{m}$. Обратно, для всех $\varepsilon < 1$ найдётся c такое, что для всех достаточно больших m и всех m, k, ε-кодирований выполнено $k \geqslant mH(\beta|\alpha) - c\sqrt{m}$.*

Эта теорема обобщает теорему Шеннона (последняя получится, если положить $l = 0$) и ее доказательство есть прямое обобщение доказательства теоремы Шеннона.

Доказательство. Утверждение теоремы не изменится, если в качестве случайной величины β взять взять целиком случайное слово длины $l+1$ (а α оставить прежним). В такой форме мы и будем доказывать теорему.

Первое утверждение. Назовем слово x длины m над n-буквенным алфавитом *сбалансированным*, если для всех слов w длины $l+1$ частота вхождения w в x отличается от $\pi(w)$ не более чем на $d/\sqrt{m}$. Величину d выберем из условия, чтобы вероятность несбалансированности была менее ε (при всех достаточно больших m). Почему найдется такое d? Нам понадобится центральная предельная теорема для эргодических марковских цепей [7]. В простейшем варианте теорема гласит, что для любой эргодической марковской цепи, любого ее начального состояния и любого состояния q, распределение случайной величины ((количество

посещений состояния q в первые m моментов времени) $-\pi(q))/\sqrt{m}$ стремится с ростом m к нормальному распределению. Здесь π обозначает стационарное распределение марковской цепи. То есть, для любого b вероятность того, что частота вхождений диграммы w отличается от $\pi(w)$ более чем на $d/\sqrt{m}$ стремится к интегралу

$$\frac{1}{\sigma\sqrt{2\pi}}\int_{-d}^{d} e^{-u^2/2\sigma^2}\,du,$$

где σ не зависит от m. Выберем настолько большое d, чтобы этот интеграл был больше $1-\varepsilon/2n^2$ (для всех диграмм). Для такого d при всех достаточно больших m вероятность того, что отклонение частоты каждой диграммы w от $\pi(w)$ превысит $d/\sqrt{m}$, будет меньше ε/n^2. Поэтому, вероятность несбалансированности меньше ε.

В любом сбалансированном слове частота вхождения слова w длины $l+1$ равна $\pi(w)+O(1/\sqrt{m})$. Значит, для любого слова u длины l и любой буквы b, частота вхождения u равна

$$\pi(u)+O(1/\sqrt{m}),$$

а частота вхождения b после u равна

$$\frac{\pi(ub)}{\pi(u)}+O(1/\sqrt{m}).$$

Следовательно, вероятность каждого сбалансированного слова равна

$$\prod_{u,b}\Big(\frac{\pi(ub)}{\pi(u)}\Big)^{m\pi(ub)+O(\sqrt{m})}=2^{-mH(\beta|\alpha)+O(\sqrt{m})}.$$

Поэтому количество сбалансированных слов не больше $2^{mH(\beta|\alpha)+O(\sqrt{m})}$, а значит их можно закодировать словами длины $k=mH(\beta|\alpha)+O(\sqrt{m})$.

Второе утверждение доказывается так: Опять рассмотрим сбалансированные слова. Но на этот раз выберем b таким большим, чтобы суммарная вероятность несбалансированных слов была меньше $(1-\varepsilon)/2$ при всех достаточно больших m. Тогда суммарная вероятность сбалансированных слов x, для которых $D(E(x))=x$, будет не меньше $(1-\varepsilon)/2$. Действительно, по условию доля слов, для которых $D(E(x))=x$, не меньше $1-\varepsilon$. Причем, не более, чем $(1-\varepsilon)/2$ из них являются несбалансированными.

Теперь оценим сверху суммарную вероятность сбалансированных слов x, для которых $D(E(x)) = x$. Как мы видели, вероятность каждого сбалансированного слова не больше $2^{-mH+O(\sqrt{m})}$. По условию слов, для которых $D(E(x)) = x$, не больше 2^k (действительно, для каждого двоичного слова u длины k существует не более одного x, для которого $D(u) = x$, $E(x) = u$). Поэтому суммарная вероятность сбалансированных слов x, для которых $D(E(x)) = x$, не превосходит $2^{k-mH+O(\sqrt{m})}$.

Соединив верхнюю и нижнюю оценки, мы получаем

$$(1 - \varepsilon)/2 \leqslant 2^{k-mH+O(\sqrt{m})},$$

откуда и следует утверждение теоремы. □

5.5 Передача информации при наличии дополнительной информации у принимающей стороны. Теорема Вольфа–Слепяна.

Пусть имеются две совместно распределенные случайные величины α, β, принимающие значения в алфавитах Γ, Σ. Рассмотрим следующую ситуацию. У передающей стороны имеется слово α^k длины k в алфавите Γ, а у принимающей — слово β^k длины k в алфавите Σ. Эти слова получены в результате k независимых испытаний пары (α, β). Сколько битов необходимо передать, чтобы принимающая сторона смогла найти α^k, если допустить небольшую вероятность ошибки ε?

Сначала рассмотрим упрощенную задачу, в которой передающая сторона знает не только α^k, но и β^k (при этом обе стороны знают Γ, Σ, k и совместное распределение α, β).

Теорема 21. *Для любого положительного ε и случайных величин α, β найдется такое c, что для всех k существует пара функций*

$$E : \Gamma^k \times \Sigma^k \to \{0,1\}^n, \qquad D : \{0,1\}^n \times \Sigma^k \to \Gamma^k$$

(кодер и декодер), где $n = H(\alpha|\beta)k + c\sqrt{k}$, для которых

$$D(E(\alpha^k, \beta^k), \beta^k) = \alpha^k$$

с вероятностью не менее $1 - \varepsilon$.

Доказательство. Будем отождествлять пару слов $(a, b) \in \Gamma^k \times \Sigma^k$ со словом длины k в алфавите $\Gamma \times \Sigma$ приписыванием i-ой буквы a к i-ой букве b для всех $i \leqslant k$.

Пусть дана разрешенная вероятность ошибки ε. По закону больших чисел найдётся такая константа c, что с вероятностью не менее $1-\varepsilon$ случайно взятая пара слов α^k, β^k является $c\sqrt{k}$-сбалансированной. Последнее означает, что если рассматривать эту пару слов как слово длины k в алфавите $\Gamma \times \Sigma$, то количество вхождению любой буквы (γ, σ) в это слово отличается $k\,\mathsf{P}(\alpha = \gamma, \beta = \sigma)$ не более, чем на $c\sqrt{k}$.

Схема кодирования-декодирования будет ошибаться только в случае, когда пара слов a, b не является $c\sqrt{k}$-сбалансированной. Поэтому вероятность ошибки не будет превосходить ε. Получив пару a, b, кодер будет кодировать a его номером среди всех слов a', для которых пара (a', b), является $c\sqrt{k}$-сбалансированной. Для доказательства теоремы нам осталось установить, что для любой $c\sqrt{k}$-сбалансированной пары (a, b) количество $c\sqrt{k}$-сбалансированных пар вида (a', b) есть $2^{H(\alpha|\beta)k+O(\sqrt{k})}$.

Пусть a, b — $c\sqrt{k}$-сбалансированная пара слов. Очевидно, что в этом случае a является $|\Gamma|c\sqrt{k}$-сбалансированным относительно распределения α. Как мы знаем, из этого следует, что вероятности появления a и пары (a, b) равны, соответственно,

$$2^{-H(\alpha)k+O(\sqrt{k})}, \quad 2^{-H(\alpha,\beta)k+O(\sqrt{k})}.$$

Отсюда следует, что вероятность появления a при известном b есть $2^{-H(\alpha|\beta)k+O(\sqrt{k})}$. Поэтому количество возможных первых компонент пар сбалансированных слов при фиксированной второй компоненте не больше $2^{H(\alpha|\beta)k+O(\sqrt{k})}$.

□

Задача 77. Докажите, что значение n в этом утверждении нельзя существенно уменьшить: если существуют функции с указанными свойствами, то $n \geqslant H(\alpha|\beta)k - c\sqrt{k}$, где c не зависит от k, но зависит от $\varepsilon, \alpha, \beta$. [Указание. Используйте рассуждения из доказательства теоремы Шеннона о бесшумном канале.]

Оказывается, аналогичное утверждение справедливо и в том случае, когда передающий не знает исхода β^k. Это утверждение называется теоремой Вольфа — Слепяна.

Теорема 22. *Для любого положительного ε и случайных величины α, β*

найдется такое c, что для всех k существует пара функций

$$E : \Gamma^k \to \{0,1\}^n, \qquad D : \{0,1\}^n \times \Sigma^k \to \Gamma^k$$

(кодер и декодер), где $n = H(\alpha|\beta)k + c\sqrt{k}$, для которых

$$D(E(\alpha^k), \beta^k) = \alpha^k$$

с вероятностью не менее $1 - \varepsilon$.

Отличие от предыдущего утверждения в том, что теперь E имеет только один аргумент.

Доказательство. Будем рассматривать $c\sqrt{k}$-сбалансированные пары слов длины k, где c выбрано таким образом, чтобы случайная пара (α^k, β^k) была $c\sqrt{k}$-сбалансированной с близкой к единице вероятностью. Будем такие пары называть просто сбалансированными. Насколько близкой, мы поймём позднее (это будет зависеть от ε). Через T будем обозначать множество всех сбалансированных пар. Для любого $a \in \Gamma^k$ через $T(a)$ будем обозначать множество всех слов b, для которых пара (a, b) сбалансирована. Аналогично определим $T(b)$ для $b \in \Sigma^k$. Множество $T(a)$ будем называть *тенью* a. Через $T(X)$ (для некоторого множества слов $X \subset \Sigma^k$) будем обозначать объединение теней всех слов из X.

Наш план таков. Мы выделим в множестве всех возможных сообщений Γ^k попарно непересекающихся части $X_1, \ldots, X_N$ с такими свойствами. Во-первых, $\mathsf{P}[\alpha^k \in X_1 \cup \cdots \cup X_N] \geqslant 1 - \varepsilon/2$. Во-вторых, для любого $i \leqslant N$ тень любого слова $a \in X_i$ слабо пересекается с объединением теней всех предшествующих a слов из X_i (относительно упорядочения X_i, которое мы тоже определим). Точнее, при условии $\alpha^k = a$, условная вероятность того, что β^k принадлежит тени a и не принадлежит тени ни одного из предшествующих a слов из X_i, не меньше $1 - \varepsilon/2$. Кодирующая функция E будет для данного слова a выдавать то i, для которого $a \in X_i$ (если таких i нет, то значение E равно любому i). Декодирующая функция D на данной паре входов (b, i) будет выдавать первый элемент a из X_i, для которого b принадлежит тени a. Если b не принадлежит тени никакого слова из X_i, то D выдает что угодно. Из свойств $X_1, \ldots, X_N$ следует, что вероятность неправильного декодирования не больше ε. Действительно, эта вероятность складывается из двух вероятностей: вероятности того, что α^k не принадлежит ни одному из

$X_1, \ldots, X_N$ (по условию эта вероятность меньше $\varepsilon/2$) и суммы по всем a из объединения $X_1, \ldots, X_N$ вероятности того, что $\alpha_k = a$ и β^k не принадлежит тени a или принадлежит тени одного из предшествующих a слов. По условию при любом фиксированном a эта вероятность не превосходит $\varepsilon/2$, поэтому и второе слагаемое не больше $\varepsilon/2$. Количество частей N не будет превосходить $2^{H(\alpha|\beta)k+O(\sqrt{k})}$, что и обеспечит требуемую верхнюю оценку длины кода. (Конец плана.)

Итак, осталось построить множества сообщений $X_1, \ldots, X_N$ с требуемыми свойствами. Мы их будем строить по очереди. Сначала построим X_1, начав с пустого множества и добавляя в него слова по очереди так, чтобы тень очередного элемента мало пересекалась с объединением теней предыдущих элементов. Добавлять новые элементы с сохранением этого свойства мы сможем до тех пор, пока $\mathsf{P}[\beta^k \in T(X_1)]$ не станет достаточно большим. После этого мы перейдем к X_2 и будем формировать его таким же образом. Но на этот раз добавляемые к X_2 сообщения будем выбирать из дополнения к X_1. Опять же, процесс добавления мы сможем продолжать до тех пор, пока $\mathsf{P}[\beta^k \in T(X_2)]$ не станет достаточно большим. И так далее. Мы закончим в тот момент, когда станет выполнено неравенство $\mathsf{P}[\alpha^k \in X_1 \cup \cdots \cup X_N] \geqslant 1 - \varepsilon/2$. Поскольку по построению $X_1, \ldots, X_N$ дизъюнктны и каждое из них довольно большое, общее их количество будет небольшим.

Для реализации этого плана нам понадобится следующая лемма. В ее формулировке под X следует понимать уже построенную часть очередного множества X_i, а под Y — дополнение к $X_1 \cup \cdots \cup X_{i-1}$.

Лемма. Для любой пары множеств $X \subset Y \subset \Gamma^k$, если

$$\mathsf{P}[\alpha^k \in Y] \geqslant \varepsilon/2$$

(Y не слишком мало) и

$$\mathsf{P}[\beta^k \in T(X)] \leqslant \varepsilon^2/8$$

(тень X мала), то существует $a \in Y$ такое, что

$$\mathsf{P}[\beta^k \in T(a) \setminus T(X) | \alpha^k = a] \geqslant 1 - \varepsilon/2$$

(тень a мало пересекается с объединением теней слов из X, а значит его можно добавить к X).

Доказательство леммы. Выберем случайным образом пару слов (a, b), используя для этого случайную величину $(\alpha^k, \beta^k)|(\alpha^k \in Y)$. Выбор будем

считать удачным, если оказалось, что пара (a, b) сбалансирована и b не принадлежит $T(X)$. Оценим сверху вероятность неудачи.

Выберем константу c в определении сбалансированности так, чтобы вероятность несбалансированности случайной пары (α^k, β^k) была меньшей $\varepsilon^2/8$. Тогда безусловная вероятность неудачи (вероятность события «$(\alpha^k, \beta^k) \notin T$ или $\beta^k \in T(X)$») не превосходит $\varepsilon^2/8 + \varepsilon^2/8 = \varepsilon^2/4$. Условная вероятность неудачи не больше безусловной вероятности, деленной на вероятность условия, то есть не больше $(\varepsilon^2/4)/(\varepsilon/2) = \varepsilon/2$. Отсюда следует, что для некоторого фиксированного $a \in Y$ условная вероятность неудачи не больше этого числа. □

Сначала построим X_1. Для этого положим $Y = \Gamma^k$ и применим лемму к $X = \emptyset$. Мы найдем элемент a_1 со свойством

$$\mathsf{P}[\beta^k \in T(a_1) | \alpha^k = a_1] \geqslant 1 - \varepsilon/2$$

и добавим его в X. Вторично применив лемму, мы найдем a_2 со свойством

$$\mathsf{P}[\beta^k \in T(a_2) \setminus T(a_1) | \alpha^k = a_2] \geqslant 1 - \varepsilon/2$$

и опять добавим его в X. И так будем добавлять элементы в X до тех пор, пока $\mathsf{P}[\beta^k \in T(X)] \leqslant \varepsilon^2/8$. Как только это условие станет ложным, закончим добавлять слова в X и положим $X_1 = X$.

Теперь положим $Y = \Gamma^k \setminus X_1$ и таким же образом построим X_2. Затем положим $Y = \Gamma^k \setminus (X_1 \cup X_2)$ и построим X_3. И так далее. Мы остановимся, когда очередное $Y = \Gamma^k \setminus (X_1 \cup X_2 \cup \cdots \cup X_N)$ перестанет удовлетворять условию леммы, то есть, $\mathsf{P}[\alpha^k \in Y] < \varepsilon/2$. В этот момент построенные множества обладают требуемыми свойствами и нам осталось только оценить сверху N.

Верхняя оценка для N получается из следующего соображения. *Весом* пары (a, b) будем называть $\mathsf{P}(\beta^k = b)$. По построению, для каждого i сумма весов всех пар, у которых первая компонента принадлежит X_i, не меньше $\varepsilon^2/8$. Поэтому N не превосходит общего веса всех сбалансированных пар, деленного на $\varepsilon^2/8$. Итак, нам нужно оценить сумму весов всех сбалансированных пар:

$$\sum_{(a,b) \in T} \mathsf{P}(\beta^k = b) = \sum_{b \in B} \mathsf{P}(\beta^k = b) \times |T(b)|.$$

Здесь B обозначает множество все вторых компонент сбалансированных пар. В доказательстве предыдущей теоремы мы уже оценивали $|T(b)|$

(для произвольного $b \in B$) величиной $2^{H(\alpha|\beta)k+O(\sqrt{k})}$. Итак, мы получаем оценку

$$\sum_{b\in B} \mathsf{P}(\beta^k = b) \times |T(b)| \leqslant 2^{H(\alpha|\beta)k+O(\sqrt{k})} \sum_{b\in B} \mathsf{P}(\beta^k = b) \leqslant 2^{H(\alpha|\beta)k+O(\sqrt{k})}.$$

Следовательно, N не превосходит

$$2^{H(\alpha|\beta)k+O(\sqrt{k})}/(\varepsilon^2/8) = 2^{H(\alpha|\beta)k+O(\sqrt{k})},$$

что и требовалось доказать. □

5.6 Каналы с помехами

5.6.1 Пропускная способность канала с помехами

Канал с помехами задается своим входным и выходным алфавитами $A = \{a_1, \ldots, a_l\}$, $B = \{b_1, \ldots, b_m\}$ и стохастической матрицей $W \in \mathbb{R}^{m\times l}$ (все элементы неотрицательны и сумма элементов в каждой строке равна единице). Строки матрицы соответствуют символам входного алфавита, а столбцы — символам выходного алфавита. Элемент W_{ab} матрицы есть вероятность того, что на выходе канала будет буква b, при условии, что на вход канала подается буква a. Случайную величину со значениями в выходном алфавите, задаваемую строчкой номер a матрицы W мы будем обозначать через $W(a)$.

Никакого распределения на входных буквах в определении канала нет. Однако пропускная способность канала определяется с помощью таких распределений. Пусть α — произвольная случайная величина со значениями во входном алфавите канала. «Применив канал» к случайной величине α, мы получим некоторую новую случайную величину, которую мы будем обозначать через $\beta = W(\alpha)$. По определению совместное распределение α, β задается равенством $\mathsf{P}[\alpha = i, \beta = j] = W_{ij} \cdot \mathsf{P}[\alpha = i]$.

Определение 2. Пропускная способность $I(W)$ канала с матрицей W определяется как максимум по всем α общей информации α и $W(\alpha)$:

$$I(W) = \max_{\alpha} I(\alpha : W(\alpha)).$$

Примеры. Канал с матрицей $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ имеет пропускную способность 1 (в качестве α надо взять равномерно распределенную случайную величину). Этот канал не имеет помех и просто копирует вход на выход. Канал с матрицей $\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$ имеет пропускную способность 0. Выходной символ в этом канале никак не зависит от входа. Канал с матрицей $\begin{pmatrix} 1/2 & 1/2 \\ 1/3 & 2/3 \end{pmatrix}$ имеет пропускную способность между нулем и единицей. Выходной символ в этом канале зависит от входа, но не сохраняет полную информацию о входе.

5.6.2 Использование каналов с помехами

Если передать один бит по третьему каналу из предыдущего примера, то выходной бит не даст почти никакой информации о входном бите. Тем не менее канал можно использовать для надежной передачи одного бита информации, если сначала закодировать каждый из двух битов достаточно длинной последовательностью нулей и единиц и передавать по каналу коды вместо самих битов. Например, в качестве кода нуля можно использовать из одних нулей, а в качестве кода единицы — из одних единиц. В самом деле, если передать много нулей подряд, то на выходе с большой вероятностью будет примерно поровну единиц и нулей, а если передавать много единиц подряд, то на выходе с большой вероятностью единиц будет примерно вдвое больше, чем нулей. Таким образом по этому каналу можно довольно надежно передать один бит информации.

Если нам нужно передать не один бит, а достаточно длинную последовательность битов, то можно разбить ее на блоки и кодировать каждый блок по отдельности. Требование состоит в том, чтобы после внесения помех каналом каждый блок можно было декодировать с ничтожной вероятностью ошибки. Теорема о канале с помехами оценивает сколько символов нужно передать по каналу в расчете на один бит исходной последовательности.

Итак, будем передавать исходную последовательность по каналу с помехами следующим образом. Сначала разрезаем ее на блоки некоторой длины k, затем кодируем каждый блок $x = b_1 \dots b_k$ несколькими буквами входного алфавита канала, затем каждую из кодирующих букв передаем по каналу и полученную последовательность выходных букв канала де-

кодируем, восстанавливая исходный блок. То есть передача одного блока x осуществляется следующим образом

$$x \to \boxed{\text{E}} \to \text{канал} \to \boxed{\text{D}} \to x.$$

Здесь E — отображение из $\{0,1\}^k$ в A^n, а D — отображение из B^n в $\{0,1\}^k$.

Такой способ использования канала называется блочным k,n-кодированием (k — длина кодируемого блока, а n — длина его кода), а отношение k/n называется скоростью передачи информации. Мы предполагаем, что каждую из n букв канал передает независимо, то есть при подаче на вход n-буквенного слова $x = (a_1 \dots a_n)$, на выходе появится слово $y = (b_1 \dots b_n)$ с вероятностью, равной произведению чисел $W_{a_1 b_1}, \dots, W_{a_n b_n}$. Будем случайную величину с этим распределением обозначать через $W^n(a)$.

Пусть δ — действительное число на интервале $(0,1)$. Будем блочное кодирование, задаваемое функциями E, D, называть δ-надежным, если для любого входного слова $x \in \{0,1\}^k$ вероятность события

$$D(W^n(E(x))) = x$$

не меньше δ. При данных n, δ мы хотим найти наибольшее такое k, для которого существует δ-надежное k,n-кодирование, то есть, существует δ-надежная пара отображений $E : \{0,1\}^k \to A^n$ и $D : B^n \to \{0,1\}^k$. Отношение k/n для максимального такого k называется максимальной скоростью δ-надежной передачи информации при n-кратном использовании канала. Оказывается, при достаточно больших n максимально возможная скорость передачи информации примерно равна пропускной способности канала и слабо зависит от δ. Точнее, при любом фиксированном $0 < \delta < 1$ максимально возможная скорость при n-кратном использовании канала стремится к пропускной способности канала, когда n стремится к бесконечности. Более точные оценки даются в следующей теореме.

Теорема 23. *Пусть дан канал с матрицей W, алфавитами A, B и пропускной способностью I. Для некоторого c, зависящего только от матрицы W, выполнено следующее.*

(1) Для любого положительного δ и любого δ-надежного k,n-кодирования выполнено неравенство

$$k \leqslant In + c\sqrt{n/\delta} + |A| \log(n+1) + \log(2/\delta).$$

(2) Для любого $\varepsilon > 0$ и всех всех $n > 0$ существует $(1-\varepsilon)$-надежное k, n-кодирование с $k \geqslant In - c\sqrt{n/\varepsilon} - \log(2/\varepsilon)$.

Доказательство. (1) Пусть дано некоторое δ-надежное k, n-кодирование E, D. Нам нужно оценить сверху k функцией от n, δ, W.

Будем называть *типом* слова $x \in \{0,1\}^k$ кортеж из $|A|$ натуральных чисел, в котором i-ый элемент равен количеству вхождений i-ой буквы из A в слово $E(x)$ (код x). Каждый элемент из типа есть натуральное число от 0 до n, поэтому общее число типов не превосходит $(n+1)^{|A|}$ (а на самом деле даже меньше, поскольку сумма всех компонент любого типа должна быть равна n).

Мы оценим сверху количество $x \in \{0,1\}^k$, имеющих данный тип P, некоторым числом, не зависящим от P, а затем умножим эту оценку на общее количество типов. Таким образом мы оценим сверху 2^k, а значит и k.

Итак, пусть фиксирован тип $P = (l_1, \ldots, l_{|A|})$. Рассмотрим случайную величину α со значениями в A, которая равна i-ой букве из алфавита A с вероятностью l_i/n. Обозначим через β результат передачи этой случайной величины по каналу: $\beta = W\alpha$, а через β^n — результат n независимых испытаний β. Наша цель — доказать, что двоичный логарифм количества сообщений типа P не превосходит $I(\alpha : \beta)n$ (с некоторой точностью). Поскольку $I(\alpha : \beta)$ не превосходит пропускной способности канала I, мы и докажем искомую оценку для k.

Пусть дано сообщение x типа P. Тогда результат передачи слова $E(x)$ по каналу W есть случайная величина со значениями в B^n, распределённая по следующей вероятностной мере $\mu_{E(x)}$: все буквы независимы друг от друга, причём распределение буквы с номером i зависит от того, какова i-ая буква слова $E(x)$ и определяется соответствующей строкой матрицы W. Пусть $N_a(E(x))$ обозначает множество позиций в которых в $E(x)$ стоит буква $a \in A$.

Слово y длины n над алфавитом B назовём *γ-сбалансированным относительно $\mu_{E(x)}$*, если для любой буквы $a \in A$, входящей в $E(x)$, и для любой буквы $b \in B$ выполнено следующее:

> количество вхождений буквы b в слово y, номера которых принадлежат $N_a(E(x))$, отличается от $W_{ab}|N_a(E(x))|$ не более, чем на γ, а если $W_{ab} = 0$, то таких вхождений нет вовсе.

По неравенству Чебышёва случайное слово, распределённое по мере $\mu_{E(x)}$ является γ-сбалансированным с вероятностью не менее $1 - |A \times B|n/\gamma^2$.

Выберем такое $\gamma = O(\sqrt{n/\delta})$, чтобы эта вероятность была не меньше $1 - \delta/2$.

Итак, для каждого слова x типа P с вероятностью не менее $1 - \delta/2$ при передаче его кода $E(x)$ по каналу на выходе канала появляется γ-сбалансированное слово относительно $\mu_{E(x)}$. Вспомним, что вероятность того, что переданное слово декодируется правильно (то есть принадлежит $D^{-1}(x)$), не меньше δ. Поэтому с вероятностью не менее $\delta/2$ переданное слово обладает обоими свойствами: оно принадлежит $D^{-1}(x)$ и является γ-сбалансированным относительно распределения $\mu_{E(x)}$.

Докажем теперь, что слов, обладающих этими двумя свойствами довольно много. Это следует из двух фактов: каждое такое слово имеет небольшую вероятность (потому что оно сбалансировано) и суммарная их вероятность большая (не меньше $\delta/2$).

Оценим вероятность γ-сбалансированных слов через условную энтропию $H(\beta|\alpha)$. Здесь мы будем использовать наше предположение, что тип слова x равен P. В самом деле, вероятность по мере $\mu_{E(x)}$ любого слова y есть произведение по всем парам букв $(a, b) \in A \times B$ таким, что $P(a) > 0$ (так мы обозначаем член типа P, соответствующий букве a) величины W_{ab} в степени, равной количеству позиций в y, в которых написано b, а в той же позиции в x написано a. По определению γ-сбалансированности количество таких позиций равно

$$W_{ab}|N_a(E(x))| = W_{ab}P(a) = W_{ab}\mathrm{P}[\alpha = a]n$$

с точностью плюс минус γ. Первое равенство здесь выполнено, поскольку x есть слово типа P, а второе равенство выполнено по определению α. Таким образом, вероятность любого γ-сбалансированного слова по мере $\mu_{E(x)}$ равна

$$\prod_{ab} W_{ab}^{W_{ab}\mathrm{P}[\alpha=a]n\pm\gamma} = 2^{-H(\beta|\alpha)n\pm c\gamma},$$

где c равно минус логарифму минимального ненулевого члена матрицы W. Поскольку вероятность любого γ-сбалансированного слова не больше $2^{-H(\beta|\alpha)n+O(\gamma)}$, количество γ-сбалансированных слов в $D^{-1}(x)$ не меньше $(\delta/2)2^{H(\beta|\alpha)n-O(\gamma)}$.

Теперь мы докажем, что для каждого γ-сбалансированного относительно $\mu_{E(x)}$ слова y вероятность события $[\beta^n = y]$ равна $2^{-H(\beta)n+O(\gamma)}$. В самом деле количество вхождений любой буквы b в y равно сумме по всем a количества вхождений буквы b в y в тех позициях, где в $E(x)$

входит a. Последнее равно

$$P(a)W_{ab} \pm \gamma = n\mathrm{P}[\alpha = a]W_{ab} \pm \gamma$$

по определению γ-сбалансированности и по определению α. Сумма по a величин $\mathrm{P}[\alpha = a]W_{ab}$ равна $\mathrm{P}[\beta = b]$. Таким образом количество вхождений буквы b в y равно $n\mathrm{P}[\beta = b] \pm |B|\gamma$. Тем самым вероятность события $[\beta^n = y]$ есть $2^{-H(\beta)n+O(\gamma)}$, что и требовалось доказать.

Отсюда следует, что сумма чисел $\mathrm{P}[\beta^n = y]$ по всем γ-сбалансированным относительно $\mu_{E(x)}$ словам из $D^{-1}(x)$ не меньше

$$(\delta/2)2^{H(\beta|\alpha)n-H(\beta)n-O(\gamma)} = (\delta/2)2^{-I(\beta:\alpha)n-O(\gamma)}.$$

Для разных x множества $D^{-1}(x)$ не пересекаются, а сумма вероятностей событий $[\beta^n = y]$ по всем y равна 1. Поэтому количество слов x типа P не может быть больше

$$(2/\delta)2^{I(\beta:\alpha)n+O(\gamma)}.$$

(2) Зафиксируем случайную величину α такую, что общая информация α и $\beta = W(\alpha)$ равна пропускной способности канала I. Зафиксируем также $0 < \varepsilon$ и $n > 0$. Через (α^n, β^n) мы будем обозначать n независимых копий случайной величины (α, β).

Нам нужно построить множество X из $2^k = 2^{H(\alpha:\beta)n-O(\sqrt{n})}$ слов $x_1, \ldots, x_{2^k} \in A^n$, которые будут использованы как коды исходных 2^k сообщений. Кодирующая функция E будет отображать i-ое в лексикографическом порядке бинарное слово длины k в слово x_i. Что нам нужно от слов $x_1, \ldots, x_{2^k}$? Нам требуется хорошая отличимость случайных величин $W^n(x_i)$ друг от друга. Точнее, нам нужно, чтобы существовала декодирующая функция D', которая по $W^n(x_i)$ с вероятностью не менее $1 - \varepsilon$ восстанавливает x_i (декодирующая функция D будет композицией D' и E^{-1}).

Для построения X и D' мы будем использовать сбалансированные слова (как и в теореме Вольфа-Слепяна). Точнее, для каждой пары букв (a, b) обозначим через $p(a, b)$ вероятность этой пары, то есть, вероятность события $\alpha = a, \beta = b$. Среднее количество вхождений пары (a, b) в случайное слово $(\alpha_1, \beta_1) \ldots (\alpha_n, \beta_n)$ равно $np(a, b)$. Зафиксируем некоторое неотрицательное действительное число γ и назовем пару слов $(x, y) \in A^n \times B^n$ *γ-сбалансированной*, если выполнено следующее

> для любой пары букв (a, b) с $p(a, b) > 0$ количество вхождений этой пары букв в слово $(a_1, b_1) \ldots (a_n, b_n)$ отличается от ожидаемого количества $np(a, b)$ не более чем на γ, а для любой пары букв (a, b) с $p(a, b) = 0$ таких вхождений нет.

Используя неравенство Чебышёва, мы можем такое $\gamma = O(\sqrt{n/\varepsilon})$, что с вероятностью не менее $1-\varepsilon/2$ пара слов (α^n, β^n) является γ-сбалансированной. Зафиксируем такое γ.

Пусть T обозначает множество всех сбалансированных пар. Для $x \in A^n$ обозначим через $T(x)$ *тень* x, определеяемую как множество всех слов y, для которых пара $(x, y) \in T$. Мы будем использовать оценку

$$|T(x)| \leqslant 2^{H(\beta|\alpha)n+O(\gamma)}.$$

Это неравенство выполнено, поскольку для каждой сбалансированной пары (x, y) условная вероятность $\mathsf{P}(\beta^n = y|\alpha^n = x)$ равна $2^{-H(\beta|\alpha)n+O(\gamma)}$. В самом деле, эта вероятность равна $\prod W_{ab}^{p(a,b)n\pm\gamma}$, где произведение берется по всем парам букв (a, b), присутствующих в x, y. По определению сбалансированности $p(a, b) > 0$ для любой такой пары букв. Поскольку $p(a, b) = W_{ab}\mathrm{P}[\alpha = a] = \mathrm{P}[\beta = b|\alpha = a]\mathrm{P}[\alpha = a]$, эта вероятность равна $2^{-H(\alpha|\beta)n\pm c\gamma}$, где c есть абсолютная величина логарифма наименьшего ненулевого элемента матрицы W. Поскольку сумма вероятностей слов из тени x при известном x не превосходит единицы, мы получаем желаемую оценку.

Значение декодирующей функции D' на слове $y \in B^n$ будет равно первому такому x_i, для которого $y \in T(x_i)$. Если y не принадлежит объединению теней слов $x_1, \ldots, x_{2^n}$, то определим $D'(y)$ произвольным образом. Нам нужно подобрать слова $x_1, x_2, \ldots$ так, чтобы для любого i вероятность события $W(x_i) \in T(x_i) \setminus (T(x_1) \cup \cdots \cup T(x_{i-1}))$ была не меньше $1 - \varepsilon$. Будем подбирать их по очереди. Очередное x_i будем искать, используя следующую лемму.

Лемма. Пусть $X \subset A^n$ любое множество такое, что

$$\mathsf{P}[\beta^n \in T(X)] \leqslant \varepsilon/2.$$

Тогда существует $x \notin X$ такое, что

$$W(x) \in T(x) \setminus T(X)$$

с вероятностью не менее $1 - \varepsilon$.

Доказательство леммы. Выберем случайным образом пару слов (x, y), используя случайную величину (α^n, β^n). Выбор будем считать удачным, если оказалось, что пара (x, y) сбалансирована и y не принадлежит $T(X)$. Эксперимент может быть неудачным по двум причинам: (x, y) не сбалансирована, вероятность этого (в силу выбора константы в определении

сбалансированности) не превосходит $\varepsilon/2$, и $y \in T(X)$, вероятность этого по условию не больше $\mathsf{P}[\beta^n \in T(X)] \leqslant \varepsilon/2$. Таким образом, выбор удачен с вероятностью не менее $1 - \varepsilon/2 - \varepsilon/2 = 1 - \varepsilon$. Отсюда следует, что для некоторого фиксированного x условная вероятность удачи не меньше этого числа. (Конец доказательства леммы.)

Будем по этой лемме увеличивать X до тех пор, пока выполнено условие леммы, то есть $\mathsf{P}[\beta^n \in T(X)] \leqslant \varepsilon/2$. В результате мы построим множество X для которого $\mathsf{P}[\beta^n \in T(X)] > \varepsilon/2$. Докажем, что отсюда следует, что X настолько большое, как нам нужно, то есть $|X| \geqslant 2^{I(\beta:\alpha)n - O(\gamma)}$. Действительно, вероятность события $\beta^n \in T(X)$ не превосходит суммы $\mathsf{P}(\beta^n = b)$ по всем $y \in T(X)$.

Суммируемые величины не превосходят $2^{-H(\beta)n + O(\gamma)}$. В самом деле, если пара слов (x, y) является γ-сбалансированной, то количество вхождений любой буквы b в слово y равно

$$\sum_a (np(a,b) \pm \gamma) = \mathrm{P}[\beta = b]n + O(\gamma),$$

а логарифм вероятности y поэтому равен

$$\sum_b (\mathrm{P}[\beta = b]n + O(\gamma)) \cdot \log \mathrm{P}[\beta = b] = -H(\beta)n + O(c\gamma),$$

где c равно абсолютной величине минимального из ненулевых чисел $\mathrm{P}[\beta = b]$.

Как мы видели, мощность $T(X)$ не больше $|X| 2^{H(\beta|\alpha)n + O(\gamma)}$. Следовательно,

$$\varepsilon/2 < \mathsf{P}[\beta^n \in T(X)] \leqslant |X| 2^{H(\beta|\alpha)n + O(\gamma)} 2^{-H(\beta)n + O(\gamma)} = |X| 2^{-I(\alpha:\beta)n + O(\gamma)},$$

откуда $|X| \geqslant 2^{I(\beta:\alpha)n - O(\gamma)}(\varepsilon/2)$. Осталось вспомнить, что $\gamma = O(\sqrt{n/\varepsilon})$. □

Задача 78. Параллельным соединением двух каналов с матрицами W^1, W^2 и алфавитами A_1, A_2, B_1, B_2 называется канал с входным алфавитом $A_1 \times A_2$, выходным алфавитом $B_1 \times B_2$, матрица которого есть тензорное произведение W^1, W^2 (то есть, вероятность преобразования пары (σ_1, σ_2) в пару (δ_1, δ_2) равна произведению $W^1_{\sigma_1\delta_1} W^2_{\sigma_2\delta_2}$). Докажите, что пропускная способность параллельного соединения равна сумме пропускных способностей исходных каналов.

Задача 79. Используя неравенство Фано и предыдущую задачу, докажите, что для любого δ-надежного n, k-кодирования выполнено неравенство $n \leqslant (Ik + 1)/\delta$. [Указание. Рассмотрите случайную величину α со значениями в множестве A^k равную $E(x)$ для случайно выбранного слова x длины n с равномерным распределением. С помощью неравенства Фано докажите, что общая информация α и $W^k\alpha$ не меньше $n\delta - 1$. Затем с помощью предыдущей задачи докажите, что общая информация α и $W^k\alpha$ не больше Ik.]

Задача 80. Доказать, что пропускная способность канала равна нулю тогда и только тогда, когда все строки матрицы канала одинаковы.

Задача 81. Пусть первый канал имеет матрицу W и алфавиты A, B, а второй канал — матрицу V и алфавиты B, Γ. Их последовательным соединением называется канал с алфавитами A, Γ и матрицей VW (произведение матриц). Докажите, что пропускная способность последовательного соединения каналов не больше пропускной способности каждого из исходных каналов.

Задача 82. Докажите, что пропускная способность последовательного соединения каналов не определяется однозначно пропускными способностями исходных каналов.

Задача 83. Пусть даны два канала с матрицами W^1, W^2 и алфавитами A_1, A_2, B_1, B_2, причем A_1 не пересекается с A_2, а B_1 не пересекается с B_2. Их прямой суммой называется канал с входным алфавитом $A_1 \cup A_2$, выходным алфавитом $B_1 \cup B_2$, матрица которого получится, если разместить матрицы W^1, W^2 по диагонали, а в незаполненные места записать нули (то есть, вероятность преобразования букв из A_1 в буквы из B_1 такая же, как у первого канала, вероятность преобразования букв из A_2 в буквы из B_2 такая же, как у второго канала, а вероятность преобразования букв из A_1 в буквы из B_2 и букв из A_2 в буквы из B_1 нулевая). Обозначим через c_1, c_2, c пропускные способности первого, второго и полученного каналов, соответственно. Докажите формулу $2^c = 2^{c_1} + 2^{c_2}$.

Глава 6

Предсказания и игры

{predictions}

В этом разделе изучается следующий сценарий. Игрок в казино должен предсказывать результаты бросаний, выполненных крупье. В простейшей ситуации (которой мы и ограничимся) каждое бросание имеет два исхода, обозначаемых -1 и $+1$, а предсказанием является некоторое действительное число на отрезке $[-1, 1]$, понимаемое, как оценка шансов появления $+1$ в результате очередного бросания. Игроку известны результаты всех выполненных к текущему моменту бросаний. Таким образом, возможными стратегиями действий игрока являются функции, которые сопоставляют каждой последовательности из -1 и $+1$ (составленной из исходов уже совершённых бросаний) некоторое число на отрезке $[-1, 1]$.

Пусть $x_1, \ldots, x_k$ обозначает произвольную последовательность исходов бросаний, сделанных крупье, а $p_1, \ldots, p_k$ обозначает произвольную последовательность предсказаний, сделанных игроком. Как оценить качество этих предсказаний для этой последовательности исходов? Один из наиболее естественных способов состоит в том, что за предсказания игрок отвечает своими деньгами, как и происходит в реальном казино. А именно, будем считать, что у игрока в каждый момент имеется капитал (начальный капитал примем за 1), и предсказание p означает, что $\frac{1+p}{2}$-ая часть текущего капитала ставится на 1, а остаток (то есть, $\frac{1-p}{2}$-ая часть текущего капитала) ставится на 0. Ставка, сделанная на выпавший исход возвращается игроку в удвоенном размере, а ставка, сделанная на противоположный исход, отбирается в пользу казино. Качество предсказаний будем оценивать текущим капиталом игрока (чем он больше, тем предсказания лучше). Нетрудно убедиться, что после выпадения исходов $x_1, \ldots, x_k$ и сделанных предсказаний $p_1, \ldots, p_k$ капитал игрока выража-

ется формулой

$$(1 + x_1p_1) \cdot (1 + x_2p_2) \cdot \dots \cdot (1 + x_kp_k). \qquad (6.1)$$ {capital}

Пусть игрок руководствуется некоторой стратегией игры S (которая сообщает, какую надо сделать ставку, исходя из последовательности исходов, выпавших к данному моменту). И пусть x — последовательность плюс/минус единиц длины k. Обозначим через $C_S(x)$ текущий капитал игрока, руководствующегося стратегией S, после k бросаний с исходами $x_1, \dots, x_k$. В соответствии с формулой (6.1),

$$C_S(x) = (1 + x_1S(\Lambda)) \cdot (1 + x_2S(x_1)) \cdot \dots \cdot (1 + x_kS(x_1 \dots x_{k-1})).$$

(Через Λ мы обозначаем пустую последовательность.) Что можно сказать о функции $x \mapsto C_S(x)$?

{th17}

Теорема 24. *Для любой стратегии S функция $C_S(x)$ принимает неотрицательные значения, причем*

$$C_S(\Lambda) = 1, \qquad \text{и} \qquad C_S(x) = \frac{C_S(x(-1)) + C_S(x1)}{2}$$

для всех x. Обратно, для любой функции $C(x)$, удовлетворяющей этим условиям (такие функции называются мартингалами*) существует стратегия игры S, для которой $C = C_S$.*

Доказательство. Равенство $C_S(\Lambda) = 1$ выполнено по условию. Неотрицательность функции C_S очевидна.

Равенство $C_S(x) = \frac{C_S(x(-1))+C_S(x1)}{2}$ доказывается так. Пусть в текущий момент капитал игрока равен $C_S(x)$, а предсказание равно p. По правилам выплаты,

$$C(x(-1)) = 2 \cdot (C(x)(1 - p)/2) = C(x)(1 - p),$$
$$C(x1) = 2 \cdot (C(x)(1 + p)/2) = C(x)(1 + p).$$

Отсюда сразу следует доказываемое равенство.

Обратно, пусть имеется произвольный мартингал $C(x)$. Тогда

$$C(x1) - C(x) = C(x) - C(x(-1))$$

для всех x. Рассмотрим следующую стратегию игры S: после исходов $x = x_1 \dots x_k$ делаем предсказание

$$\frac{C(x1)}{C(x)} - 1 = 1 - \frac{C(x(-1))}{C(x)}.$$

(Эта величина находится на отрезке $[-1, 1]$, поскольку, с одной стороны, она получена добавлением неотрицательного числа к -1, а, с другой стороны, она получена вычитанием неотрицательного числа из 1.) В случае исхода 1 капитал станет равным

$$C(x)\Big(1 + \Big(\frac{C(x1)}{C(x)} - 1\Big)\Big) = C(x1),$$

как мы и хотели. Аналогичным образом, в случае исхода -1 капитал будет равен

$$C(x)\Big(1 - \Big(1 - \frac{C(x(-1))}{C(x)}\Big)\Big) = C(x(-1)),$$

как и требуется. □

Пусть всего казино делает n бросаний. Какие значения может принимать функция $C_S(x)$ на словах длины n (то есть, в конце игры)?

{th18}

Теорема 25. *Среднее значение функции $C_S(x)$ по всем словам из ± 1 длины n равно 1. Обратно, для любой неотрицательной функции $C(x)$, определенной на словах длины n, среднее значение которой равно 1, существует стратегия игры S, для которой $C(x) = C_S(x)$ для всех строк x из ± 1 длины n.*

Доказательство. По предыдущей теореме, функция $C(x)$ является мартингалом. Среднее значение мартингала на словах любой длины n равно 1, что легко доказать индукцией по n.

Обратно, пусть $C(x)$ удовлетворяет условию. Продолжим ее на все слова длины не более n, положив $C(y)$ равным среднему арифметическому $C(x)$ по всем продолжениям x длины n слова y. Очевидно, C является мартингалом и по предыдущей теореме есть C_S для некоторой стратегии S. □

Задача 84. Пусть казино делает n бросаний так, что последовательность исходов x заведомо принадлежит некоторому множеству $A \subset \{-1, 1\}^n$.

Пусть игроку стало известно множество A. Докажите, что у игрока есть стратегия, гарантирующая заключительный капитал не менее $2^n/|A|$, какая бы последовательность исходов $x \in A$ ни выпала.

Напомним, что мы оцениваем качество предсказаний с помощью формулы (6.1), отражающей игру в казино со ставками на два равновероятных исхода в пределах текущего капитала. На практике используются и другие формулы. Говорят, например, что американское бюро погоды ($+1$ соответствует наличию дождя, -1 — его отсутствию) за неточные предсказания подвергается штрафу, пропорциональному величине

$$(x_1 - p_1)^2 + (x_2 - p_2)^2 + \cdots + (x_k - p_k)^2.$$

(Чем выше эта величина, тем худшим считается предсказание.) Легко видеть, что при идеально точном предсказании штраф, вычисленный по этой формуле, равен нулю, а иначе положителен. Можно вместо возведения в квадрат в этой формуле использовать взятие абсолютной величины:

$$|x_1 - p_1| + |x_2 - p_2| + \cdots + |x_k - p_k|.$$

Наконец, для вычисления штрафа можно использовать минус логарифм формулы (6.1):

{log}
$$-\log(1 + x_1 p_1) - \log(1 + x_2 p_2) - \cdots - \log(1 + x_k p_k) \tag{6.2}$$

(минус нужен, чтобы сохранить монотонность — чем хуже предсказание, тем больше штраф, а логарифм нужен, чтобы общий штраф стал равен сумме отдельных штрафов).

Все три рассмотренных формулы для вычисления штрафа имеют следующий общий вид. Зафиксирована некоторая функция $\rho : [-1, 1] \times \{-1, 1\} \to \mathbb{R}$ для вычисления штрафа в одном акте предсказания: если очередное предсказание равно $p \in [-1, 1]$, а исход равен $q \in \{-1, 1\}$, то начисляется штраф в размере $\rho(p, q)$, который добавляется к уже имеющемуся штрафу. То есть, общий штраф при предсказаниях $p_1, \ldots, p_k$ и исходах $x_1, \ldots, x_k$ вычисляется по формуле

$$\rho(p_1, x_1) + \rho(p_2, x_2) + \cdots + \rho(p_k, x_k).$$

Возникает естественный вопрос. Пусть фиксирована функция штрафа ρ. Можно ли построить какую-нибудь хорошую стратегию предсказаний, то есть, стратегию, которая в том или ином смысле хорошо предсказывает *все* последовательности? В идеале мы хотели бы построить

стратегию, которая *любую* последовательность предсказывает не хуже, чем *любая* другая стратегия. К сожалению, этот идеал недостижим (для любой из трёх рассмотренных выше функций штрафа). Действительно, для любой конкретной последовательности $x_1, \ldots, x_k$ оптимальной будет стратегия, выдающая в качестве предсказаний символы этой конкретной последовательности, а все другие стратегии предсказывают эту стратегию строго хуже этой. Поскольку при любом $k > 0$ существует более одной последовательности длины k, то и оптимальной стратегии не существует.

Поэтому умерим наши запросы и поставим вопрос следующим образом: пусть дано конечное или даже счетное семейство стратегий $S_1, S_2, \ldots$, с которыми нам нужно конкурировать. Существует ли стратегия S со следующим свойством: для любой стратегии-конкурента S_i для некоторой константы c_i выполнено неравенство

$$\text{штраф}_S(x) \leqslant \text{штраф}_{S_i}(x) + c_i$$

для *всех* последовательностей x?

Оказывается, что для многих функций ρ ответ на этот вопрос положительный. Более того, можно явно описать класс функций ρ, для которых это верно. Это было сделано в работах Вовка и его учеников. Более подробно о способах построения хороших стратегий предсказания можно прочитать в статье [11] и в монографии [5]. Мы ограничимся только одной конструкцией, изложенной в работе Р. Соломонова [10], работающей для функции штрафа $\rho(p, q) = -\log(1 + pq)$ (общий штраф, следовательно, задаётся формулой (6.2)).

Теорема 26. *Пусть задано произвольное счётное семейство стратегий $S_1, S_2, \ldots$. Существует стратегия S такая, что для любого i найдется константа c_i, для которой*

$$\textit{штраф}_S(x) \leqslant \textit{штраф}_{S_i}(x) + c_i$$

для всех x.

Доказательство. Обозначим через $C_i(x)$ капитал игрока, полученный после исходов x при игре в соответствии со стратегией S_i. По теореме 24 (первая часть) функция $x \mapsto C_i(x)$ является мартингалом. Рассмотрим

функцию $x \mapsto \sum_{i=1}^{\infty} 2^{-i} C_i(x)$. Нетрудно проверить, что она также является мартингалом. По второй части теоремы 24 найдётся стратегия S, для которой

$$C_S(x) = \sum_i 2^{-i} C_i(x)$$

для всех x. Эта стратегия и есть искомая. В самом деле, для любого i и всех x выполнено неравенство

$$C_S(x) \geqslant 2^{-i} C_i(x),$$

а значит

$$\text{штраф}_S(x) = -\log C_S(x) \leqslant i - \log C_i(x) = \text{штраф}_{S_i}(x) + i.$$

Теорема доказана.

Можно пере-изложить доказательство в игровых терминах: стратегия S мысленно делит свой начальный капитал 1 между стратегиями $S_1, S_2, \ldots$ так, что i-ая стратегия получает капитал 2^{-i}. Затем стратегия S мысленно запускает все стратегии параллельно и вычисляет предсказание каждой из них. Выдаваемое ей предсказание есть взвешенная сумма всех вычисленных предсказаний, при этом мнение каждой стратегии учитывается с весом, пропорциональным текущему капиталу этой стратегии. Это довольно естественная линия поведения — если у Вас есть несколько советчиков того, куда вложить деньги, то мнение каждого из них естественно учитывать с весом, пропорциональным количеству уже заработанных им денег. □

В следующих задачах мы считаем, что казино генерирует исходы бросаний случайным образом, в соответствии с некоторым распределением вероятностей (необязательно равномерным).

Задача 85. Пусть казино делает n бросаний, используя некоторое распределение вероятностей $x \mapsto \mu(x)$ на строках длины n (известное игроку).
(а) Докажите, что у игрока есть стратегия со средним капиталом $2^{n - H_{\min}}$ в конце игры, где $H_{\min}$ обозначает минимальную энтропию распределения p: $H_{\min} = \min_x(-\log \mu(x))$.
(б) Докажите, что у игрока есть стратегия, средний логарифм капитала которой (в конце игры) равен $n - H$, где H обозначает энтропию Шеннона распределения μ.

До сих пор мы рассматривали случай, когда выигранная ставка удваивается. При такой выплате по теореме 25 средний капитал игрока равен начальному. Это можно интерпретировать следующим образом: действия казино являются «честными», если выбор исходов осуществляется с помощью симметричной монетки, гарантирующей равномерное распределение на исходах.

Представим теперь, что выбор исходов в казино осуществляется, с помощью некоторого (возможно неравномерного) распределения $x \mapsto \mu(x)$ на словах длины n. Чтобы избежать чисто технических проблем, будем считать, что $\mu(x)$ положительно для всех x. При какой выплате на сделанную ставку игра останется честной в этом случае? Ответ: если после исходов $x = (x_1, \ldots, x_k)$ игрок сделал ставку -1 и угадал, то ему нужно возвратить ставку, умноженную на $\mu(x)/\mu(x(-1))$. Аналогично, если игрок сделал ставку на 1 и угадал, то ему должна возвращаться ставка с коэффициентом $\mu(x)/\mu(x1)$.

Правильность такой стратегии выплат подтверждается тем, что выполняются аналоги теорем 24 и 25.

{th17a}

Теорема 27. *Для любой стратегии S игрока функция $C_S(x) = C(x)$ принимает неотрицательные значения, причем*

$$C(\Lambda) = 1, \qquad \text{и} \qquad C(x) = \frac{\mu(x(-1))}{\mu(x)} C(x(-1)) + \frac{\mu(x1)}{\mu(x)} C(x1).$$

Обратно, для любой функции $C(x)$, удовлетворяющей этим условиям (такие функции называются мартингалами *относительно распределения μ) существует стратегия игры S, для которой $C = C_S$.*

{th18a}

Теорема 28. *Для любой стратегии S среднее значение функции $C_S(x) = C(x)$, на словах длины n по распределению μ равно 1. Обратно, для любой неотрицательной функции $C(x)$, определенной на словах длины n, среднее которой по распределению μ равно 1, существует стратегия игры S, для которой $C(x) = C_S(x)$ для всех строк длины n.*

Доказательство этих теорем совершенно аналогично доказательству теорем 24 и 25.

Задача 86. Пусть казино делает n бросаний, используя некоторое распределение вероятностей $x \mapsto \nu(x)$ на строках длины n (известное игроку). При этом казино производит выплаты так, как если бы оно использовало

распределение μ (то есть, выигранная ставка на -1, сделанная после исходов x, увеличивается в $\mu(x)/\mu(x(-1))$ раз, а ставка на 1 — в $\mu(x)/\mu(x1)$ раз.

(а) Докажите, что у игрока есть стратегия со средним капиталом $\max_x\{\nu(x)/\mu(x)\}$ в конце игры.

(б) Докажите, что у игрока есть стратегия, средний логарифм капитала которой равен расстоянию Кульбака—Лейблера $\sum_x \nu(x)\log\frac{\nu(x)}{\mu(x)}$ между распределениями ν, μ.

Глава 7

Колмогоровская сложность

7.1 Что такое колмогоровская сложность?

«На пальцах» это проще всего объяснить так. Существуют программы, которые сжимают файлы (для экономии места в архиве): наиболее распространённые называются `zip`, `gzip`, `compress`, `rar`, `arj`.

Применив такую программу к некоторому файлу (с текстом, данными, программой), мы получаем его сжатую версию (которая, как правило, короче исходного файла). По ней можно восстановить исходный файл (с помощью парной программы «декомпрессии»; часто сжатие и восстановление объединены в одну программу).

Так вот, в первом приближении колмогоровскую сложность файла можно описать как длину его сжатой версии. Тем самым файл, имеющий регулярную структуру и хорошо сжимаемый, имеет малую колмогоровскую сложность (в сравнении с его длиной). Напротив, плохо сжимаемый файл имеет сложность, близкую к длине.

Это описание весьма приблизительно и нуждается в уточнениях— как технических, так и принципиальных. Начнём с технического: вместо файлов (последовательностей байтов) мы будем рассматривать двоичные слова. Более существенны другие отличия:

- Программы сжатия нет— мы рассматриваем лишь программу восстановления. Точнее, назовём *декомпрессором* произвольный алгоритм (программу), который получает на вход двоичные слова и выдаёт на выход также двоичные слова. Если декомпрессор D на входе x даёт y, мы пишем $D(x) = y$ и говорим, что x является *опи-*

санием y при данном D (относительно данного D). Декомпрессоры мы также будем называть *способами описания*.

- Мы не требуем, чтобы декомпрессор был определён на всех словах. При некоторых x вычисление $D(x)$ может не останавливаться и не давать результата. Мы также не накладываем никаких ограничений на время работы D: на некоторых входах программа D может дать ответ лишь после очень долгой работы.

Пусть фиксирован некоторый способ описания (декомпрессор) D. Для данного слова x рассмотрим все его описания, то есть все слова y, для которых $D(y)$ определено и равно x. Длину самого короткого из них и называют *колмогоровской сложностью* слова x при данном способе описания D:

$$KS_D(x) = \min\{\, l(y) \mid D(y) = x\}.$$

Здесь и далее $l(y)$ обозначает длину слова y. Индекс D подчёркивает, что определение зависит от выбора способа D. Минимум пустого множества, как обычно, считается равным $+\infty$, так что $KS_D(x)$ бесконечно для тех x, которые не входят в область значений функции D (не могут быть результатами декомпрессии, не имеют описаний).

Это определение кажется бессодержательным, поскольку для разных D получаются разные определения, в том числе абсурдные. Например, если D нигде не определён, то KS_D всегда бесконечно. Если $D(y) = \Lambda$ (пустое слово) при всех y, то сложность пустого слова равна нулю (поскольку $D(\Lambda) = \Lambda$ и $l(\Lambda) = 0$), а сложность всех остальных слов бесконечна.

Более осмысленный пример: если программа–декомпрессор копирует вход на выход и $D(x) = x$ при всех x, то каждое слово имеет единственное описание (самого себя) и $KS_D(x) = l(x)$.

Естественно, для любого слова x можно подобрать способ описания D, при котором x имеет малую сложность. Достаточно положить $D(\Lambda) = x$, и тогда $KS_D(x)$ будет равно нулю. Можно также подобрать способ описания, благоприятствующий целому классу слов: например, для слов из одних нулей хорош такой способ описания:

$$D(bin(n)) = 000\ldots000 \quad (n \text{ нулей}),$$

где $bin(n)$— двоичная запись числа n. Легко проверить, что длина слова $bin(n)$ примерно равна $\log_2 n$ (не превосходит $\log_2 n + 1$). Мы получаем,

что для построенного способа описания сложность слова из n нулей близка к $\log_2 n$ (и много меньше длины слова, то есть n). Зато все другие слова (содержащие хотя бы одну единицу) имеют бесконечную сложность.

На первый взгляд кажется, что определение сложности настолько сильно зависит от выбора способа описания, что никакая общая теория невозможна.

7.2 Оптимальные способы описания

Способ описания тем лучше, чем короче описания. Поэтому естественно дать такое определение: способ D_1 *не хуже* способа D_2, если

$$KS_{D_1}(x) \leqslant KS_{D_2}(x) + c$$

при некотором c и при всех x.

В этом определении требует пояснения константа c. Мы соглашаемся пренебрегать увеличением сложности не более чем на константу. Конечно, можно сказать, что это лишает определение практического смысла, так как константа c может быть сколь угодно велика. Однако без такого «огрубления» обойтись не удаётся.

Пример. Пусть даны два произвольных способа описания D_1 и D_2. Покажем, что существует способ описания D, который не хуже их обоих.

В самом деле, положим

$$D(0y) = D_1(y)$$
$$D(1y) = D_2(y)$$

Другими словами, первый бит описания мы воспринимаем как номер способа описания, а остальную часть— как собственно описание. Ясно, что если y является описанием x при D_1 (или D_2), то $0y$ (соответственно $1y$) является описанием x при D, и это описание всего лишь на один бит длиннее. Поэтому

$$KS_D(x) \leqslant KS_{D_1}(x) + 1$$
$$KS_D(x) \leqslant KS_{D_2}(x) + 1$$

при всех x, так что способ D не хуже обоих способов D_1 и D_2.

Этот приём используется на практике. Например, формат `zip`–архива предусматривает заголовок, в котором указывается номер используемого метода сжатия, а затем идёт сам сжатый файл. При этом использование N различных способов описания приводит к тому, что начальные $\log_2 N$ битов (или около того) приходится зарезервировать для указания используемого способа.

{intro-universal}

Несколько обобщив эту же идею, можно доказать такую теорему:

Теорема 29 (Соломонова–Колмогорова)**.** *Существует способ описания D, который не хуже любого другого: для всякого способа описания D' найдётся такая константа c, что*

$$KS_D(x) \leqslant KS_{D'}(x) + c$$

для любого слова x.

Способ описания, обладающий указанным в теореме свойством, называют *оптимальным*.

Доказательство. Напомним, что по нашему определению способами описания являются вычислимые функции. Программы, их вычисляющие, можно считать двоичными словами. Будем при этом предполагать, что по программе можно определить, где она кончается (такой способ записи по-английски называется self-delimiting; подходящего русского слова, пожалуй, нет, и мы будем называть такие программы само-ограниченными). Если выбранный способ записи программ не таков, то его можно модифицировать. Например, можно продублировать каждый знак в записи программы (заменив 0 на 00 и 1 на 11), а в конце программы добавить группу 01.

Теперь определим новый способ описания D, положив

$$D(py) = p(y),$$

где p— произвольная программа (при выбранном способе записи), а y— любое двоичное слово. Другими словами, D читает слово слева направо, выделяя из него начало–программу. (Если таковой не оказывается, D делает что угодно, например, зацикливается.) Далее D применяет найденную программу (p) к остатку входа (y) и выдаёт полученный результат.

Покажем, что D не хуже любого другого способа описания P. Пусть p— программа, вычисляющая функцию P, причём записанная в выбранной нами форме. Если слово y является кратчайшим описанием слова x

относительно P, то py будет описанием x относительно D (не обязательно кратчайшим). Поэтому при переходе от P к D длина описания увеличится не более чем на $l(p)$, и

$$KS_D(x) \leqslant KS_P(x) + l(p).$$

Видно, что константа (то есть $l(p)$) зависит лишь от выбора способа описания P, но не от x. □

По существу здесь используется тот же приём, что и в предыдущем примере, только вместо двух способов описания соединяются все мыслимые способы— каждый со своим префиксом. Именно так устроены так называемые «само-разархивирующиеся» архивы (self-extracting archives). Такой архив представляет собой исполняемый файл, в котором, однако, собственно программа занимает лишь небольшой начальный кусок. Эта программа загружается в память, после чего читает и разархивирует остаток архива.

Заметим, что построенный нами оптимальный способ описания на некоторых входах работает очень долго (поскольку бывают долго работающие программы), а на некоторых входах и вовсе не определён.

Фиксируем некоторый оптимальный способ описания D и будем называть *колмогоровской сложностью* слова x значение $KS_D(x)$. В обозначении $KS_D(x)$ будем опускать индекс D и писать просто $KS(x)$.

Замена оптимального способа на другой оптимальный способ приводит к изменению сложности не более чем на константу: для любых оптимальных способов D_1 и D_2 найдётся такая константа $c(D_1, D_2)$, что

$$|KS_{D_1}(x) - KS_{D_2}(x)| \leqslant c(D_1, D_2)$$

при всех x. Это неравенство записывают как

$$KS_{D_1}(x) = KS_{D_2}(x) + O(1),$$

понимая под $O(1)$ произвольную ограниченную функцию от x.

Имеет ли смысл говорить о колмогоровской сложности конкретного слова x согласно этому определению, не указывая, какой оптимальный способ мы используем? Нет— легко понять, что подбором подходящего оптимального способа можно сделать сложность данного слова любой. Точно так же нельзя говорить, что слово x проще слова y (имеет меньшую сложность): выбирая тот или иной оптимальный способ, можно сделать любое из двух слов более простым.

Каков же тогда смысл колмогоровской сложности, если ни про какое конкретное слово ничего нельзя сказать?

Свойства оптимального способа, построенного при доказательстве теоремы Соломонова–Колмогорова, зависят от использованного способа записи программ (то есть от выбора языка программирования). Два таких способа приводят к сложностям, отличающимся не более чем на константу, которая есть длина интерпретатора одного языка программирования, написанного на другом языке. Можно надеяться, что при естественном выборе языков эта константа будет измеряться тысячами или даже сотнями. Тем самым, если мы говорим о сложностях порядка сотен тысяч (скажем, для текста романа) или миллионов (скажем, для ДНК), то уже не так важно, какой именно язык программирования мы выбрали.

Но тем не менее надо всегда помнить, что все утверждения о колмогоровской сложности носят асимптотический характер. Чтобы утверждение имело смысл, в нём должна идти речь о сложности не изолированного слова, а последовательности слов. Для теории сложности вычислений такого рода ситуация типична: скажем, оценки на сложность решения какой-либо задачи обычно также имеют асимптотический характер.

7.3 Сложность и случайность

Вернёмся к неравенству $KS(x) \leqslant l(x) + O(1)$ (теорема 33). Для большинства слов данной длины это неравенство близко к равенству. В самом деле, справедливо такое утверждение:

{intro-cardinality}

Теорема 30. *Пусть n— произвольное число. Тогда существует менее 2^n слов x, для которых $KS(x) < n$.*

Доказательство. Пусть D— оптимальный способ описания, фиксированный при определении колмогоровской сложности. Тогда все слова вида $D(y)$ при $l(y) < n$ (и только они) имеют сложность менее n. А таких слов не больше, чем различных слов y, имеющих длину меньше n, которых имеется

$$1 + 2 + 4 + 8 + \ldots + 2^{n-1} = 2^n - 1$$

штук (слов длины k ровно 2^k). □

Отсюда легко заключить, что доля слов сложности меньше $n-c$ среди всех слов длины n меньше $2^{n-c}/2^n = 2^{-c}$. Например, доля слов сложности менее 90 среди всех слов длины 100 меньше 2^{-10}.

Таким образом, большинство слов несжимаемы или почти несжимаемы. Это можно выразить и так: почти наверняка случайно взятое слово данной длины окажется (почти) несжимаемым. Рассмотрим следующий мысленный (или даже реальный) эксперимент. Бросим монету, скажем, 80000 раз и сделаем из результатов бросаний файл длиной в 10000 байтов (8 битов образуют один байт). Можно смело держать пари, что ни один существовавший до момента бросания архиватор не сумеет сжать этот файл более чем на 10 байтов. (В самом деле, вероятность этого меньше 2^{-80} для каждого конкретного архиватора, а число различных архиваторов не так уж велико.)

Вообще естественно считать случайными несжимаемые слова: случайность есть отсутствие закономерностей, которые позволяют сжать слово. Конечно, чёткой границы между случайными и неслучайными объектами провести нельзя. Глупо интересоваться, какие именно из восьми слов длины 3 (то есть из слов $000, 001, 010, 011, 100, 101, 110, 111$) случайны, а какие нет. Другой пример: начав со «случайного» слова длиной 10000, будем заменять в нём по очереди единицы на нули. В конце получится явно неслучайное слово (из одних нулей), но имеет ли смысл спрашивать, в какой именно момент слово перестало быть случайным? Вряд ли.

Естественно определить *дефект случайности* двоичного слова x как разность $l(x) - KS(x)$. Используя эту терминологию, можно сказать так: теорема 33 утверждает, что дефект случайности почти что неотрицателен (не меньше некоторой константы), а теорема 30 гарантирует, что для случайно взятого слова данной длины n (мы считаем все слова длины n равновероятными) вероятность иметь дефект больше d оценивается сверху числом $1/2^d$.

Теперь закон больших чисел (утверждающий, что у большинства двоичных слов данной длины n доля единиц близка к $1/2$) можно попытаться перевести на язык колмогоровской сложности так: у любого слова с малым дефектом случайности частота единиц близка к $1/2$. Из этого «перевода» вытекает исходная формулировка закона (поскольку мы уже знаем, что слова с малым дефектом случайности составляют большинство); как мы впоследствии убедимся, в некотором смысле эти формулировки равносильны.

Если мы хотим провести чёткую границу между случайными и неслучайными объектами, мы должны от конечных объектов перейти к бесконечным. Определение случайности для бесконечных последовательностей нулей и единиц было дано учеником Колмогорова шведским математиком Мартин-Лёфом. Впоследствии Левин и Шнорр нашли критерий случайности в терминах сложности: последовательность случайна тогда и только тогда, когда дефект случайности её начальных отрезков ограничен. (Правда, нужно использовать другой вариант колмогоровской сложности, так называемую *монотонную* сложность.)

Задача 87. Докажите, что среднее арифметическое $KS(x)$ по всех словам длины n есть $n+O(1)$. [Указание. Доля слов x, для которых $KS(x) < n-l$, меньше 2^{-l}, и ряд $\sum l2^{-l}$ сходится. Это доказывает нижнюю оценку для среднего арифметического.]

7.4 Невычислимость KS и парадокс Берри

Прежде чем говорить о применениях колмогоровской сложности, скажем о препятствии, с которым сталкивается любое применение. Увы, функция KS невычислима: не существует алгоритма, который по данному слову x определяет его колмогоровскую сложность. Более того, не существует никакой нетривиальной (неограниченной) вычислимой нижней оценки для KS, как показывает следующая теорема.

{intro-nobound}

Теорема 31. *Пусть k — произвольная (не обязательно всюду определённая) вычислимая функция из Ξ в $\mathbb{N}$. (Другими словами, вычисляющий её алгоритм применим к некоторым двоичным словам и даёт в качестве результатов натуральные числа.) Если k является нижней оценкой для колмогоровской сложности (то есть $k(x) \leqslant KS(x)$ для тех x, для которых $k(x)$ определено), то k ограничена: все её значения не превосходят некоторой константы.*

Доказательство. Доказательство этой теоремы повторяет так называемый « парадокс Берри». Этот парадокс состоит в предложении рассмотреть

> *наименьшее число, которое нельзя определить фразой из не более чем тринадцати русских слов.*

Эта фраза как раз содержит тринадцать слов и определяет то самое число, которое нельзя определить, так что получается противоречие.

Следуя этой идее, мы можем искать *первое попавшееся двоичное слово, колмогоровская сложность которого больше некоторого* N. С одной стороны, это слово по построению будет иметь сложность больше N, с другой стороны, приведённое его описание будет коротким (оно включает в себя число N, но двоичная запись числа N гораздо короче самого́ N). Но как искать это слово? Для этого-то и нужна вычислимая нижняя оценка колмогоровской сложности.

Перейдём к формальному изложению доказательства. По условию функция k является вычислимой нижней оценкой колмогоровской сложности. Рассмотрим функцию $B(N)$, аргументом которой является натуральное число N, вычисляемую следующим алгоритмом: «Развернуть параллельно вычисления $k(0), k(1), k(2)\ldots$ и проводить их до тех пор, пока не обнаружится некоторое x, для которого $k(x) > N$. Первое из таких x и будет результатом».

Если функция k ограничена, то теорема доказана. В противном случае функция B определена для всех N, и $k(B(N)) > N$ по построению. По предположению k является нижней оценкой сложности, так что и $KS(B(N)) > N$. С другой стороны, $B(N)$ эффективно получается по двоичной записи $bin(N)$ числа N, поэтому

$$KS(B(N)) \leqslant KS(bin(N)) + O(1) \leqslant l(bin(N)) + O(1) \leqslant \log_2 N + O(1)$$

(первое неравенство следует из теоремы 34, а второе— из теоремы 33; $O(1)$ обозначает ограниченное слагаемое). Получается, что

$$N < KS(B(N)) \leqslant \log_2 N + O(1),$$

что при больши́х N приводит к противоречию. □

7.5 Перечислимость сверху колмогоровской сложности

Множество натуральных чисел называется *перечислимым*, если существует алгоритм без входа, который печатает список всех элементов множества в некотором порядке, разделяя элементы, например, запятыми.

Временны́е интервалы между печатью последовательных элементов могут быть сколь угодно большими: после печати очередного элемента мы не знаем, появится ли когда-нибудь следующий. Если множество бесконечно, то алгоритм, разумеется, не останавливается. Но он может не останавливаться даже в случае конечного множества.

Перечислимыми являются

- множество четных чисел: алгоритм делает счетное число шагов, на шаге i он печатает $2i$,
- множество простых чисел: на шаге i алгоритм проверяет, является ли i простым, и печатает его, если оно простое, и просто переходит к $i+1$ иначе,
- множество n, для которых в десятичной записи числа π найдется n троек подряд, окруженных не-тройками (алгоритм вычисляет по очереди все цифры десятичной записи π и, как только, находит очередной массив троек, окруженных не-тройками, печатает его длину.

Аналогично определяется перечислимость множества двоичных слов, множества пар натуральных чисел и так далее.

Пусть функция $f(x)$ определена на всех двоичных словах и принимает в качестве значений натуральные числа, а также специальный символ $+\infty$. Функция f называется *перечислимой сверху*, если множество

$$G_f = \{\langle x, n\rangle \mid f(x) < n\},$$

называемое иногда «над-графиком» функции f, перечислимо. Это объясняет несколько странное название «перечислимая сверху».

le-upper-reformulated}

Теорема 32. (а) *Функция KS перечислима сверху, причём* $|\{x \mid KS(x) < n\}| < 2^n$ *при всех* n.

(б) *Если функция* KS' *перечислима сверху и* $|\{x \mid KS'(x) < n\}| < 2^n$ *при всех* n, *то найдётся такое* c, *что* $KS(x) < KS'(x) + c$ *для всех* x.

Доказательство. (а) Мы уже проводили оценку количества слов сложности меньше n, поэтому нам нужно доказать только перечислимость сверху. Зафиксируем оптимальный декомпрессор D из определения KS.

Алгоритм перечисления над-графика функции KS: Запустим D параллельно на всех входах: на n-ом этапе дадим D проработать n шагов

на первых n двоичных словах. Когда в ходе этого процесса мы обнаружим, что $D(p) = x$ для некоторых p, x, ставим в очередь на печать пары $\langle x, n\rangle$ для всех $n > l(p)$. При этом оставляем в очереди счетное количество свободных мест для будущих пар. Кроме того, на каждом этапе выводим на печать первую пару из очереди.

(б) Пусть имеется функция KS' с указанными в теореме свойствами. Определим отображение $x \mapsto D'(x)$ следующим образом. Запустим перечислитель для над-графика KS'. При появлении очередной пары $\langle x, n\rangle$ берем первое слово p длины n, на котором D' не было определено ранее, и определяем $D'(p) = x$. Неравенство $|\{x \mid KS'(x) < n\}| < 2^n$ гарантирует, что такое p обязательно найдется.

Определенное нами отображение вычисляется следующим алгоритмом: на входе p запускаем описанный выше процесс и дожидаемся, когда $D'(p)$ станет определенным (если этого никогда не произойдет, то алгоритм не завершит свою работу на входе p). Будем этот алгоритм обозначать той же буквой. По построению $KS_{D'}(x) = KS'(x) + 1$. Отсюда следует, что $KS(x) \leqslant KS'(x) + O(1)$. □

Теорему 32 можно воспринимать в качестве альтернативного определения колмогоровской сложности: $KS(x)$ есть наименьшая с точностью до постоянного слагаемого перечислимая сверху функция, удовлетворяющая неравенству $|\{x \mid KS(x) < n\}| < 2^n$ при всех n. Обычно это альтернативное определение используется в следующем виде. Пусть для некоторой перечислимой сверху функции $f(x)$ нам нужно выяснить, верно ли неравенство $KS(x) < f(x) + O(1)$ (то есть, существует ли константа c такая, что для $KS(x) < f(x) + c$ для всех x). По теореме 32 это неравенство имеет место тогда и только тогда, когда $|\{x \mid f(x) < n\}| = O(2^n)$.

7.6 Сложность и информация

Колмогоровскую сложность слова x можно назвать также *количеством информации* в слове x. В самом деле, слово из одних нулей, которое может быть описано коротко, содержит мало информации, а какое-то сложное слово, которое не поддаётся сжатию, содержит много информации (пусть даже и бесполезной— мы не пытаемся отделить осмысленную информацию от бессмысленной, так что любая абракадабра для нас содержит много информации, если её нельзя задать коротко).

Если слово x имеет сложность k, мы говорим, что x содержит k битов информации. Естественно ожидать, что число битов информации в слове не превосходит его длины, то есть что $KS(x) \leqslant l(x)$. Так и оказывается (но только надо добавить константу, о чём мы уже говорили).

{intro-length}

Теорема 33. *Существует такая константа c, что*

$$KS(x) \leqslant l(x) + c$$

для любого слова x.

Доказательство. Мы уже говорили, что если $P(y) = y$ при всех y, то $KS_P(x) = l(x)$. В силу оптимальности найдётся такое c, что

$$KS(x) \leqslant KS_P(x) + c = l(x) + c$$

при всех x. □

Утверждение этой теоремы обычно записывают так: $KS(x) \leqslant l(x) + O(1)$. Из него вытекает, в частности, что колмогоровская сложность любого слова конечна (то есть что любое слово имеет описание).

Вот ещё один пример свойства, которого естественно ожидать от понятия «количество информации»: при алгоритмических преобразованиях количество информации не увеличивается (точнее, увеличивается не более чем на константу, зависящую от алгоритма).

{intro-transform}

Теорема 34. *Для любого алгоритма A существует такая константа c, что*

$$KS(A(x)) \leqslant KS(x) + c$$

для всех x, при которых $A(x)$ определено.

Доказательство. Пусть D— оптимальный декомпрессор, используемый при определении колмогоровской сложности. Рассмотрим другой декомпрессор D':

$$D'(p) = A(D(p))$$

(D' применяет сначала D, а затем A). Если p является описанием некоторого x относительно D, причём $A(x)$ определено, то p является описанием $A(x)$ относительно D'. Взяв в качестве p кратчайшее описание x относительно D, находим, что

$$KS_{D'}(A(x)) \leqslant l(p) = KS_D(x) = KS(x),$$

а в силу оптимальности

$$KS(A(x)) \leqslant KS_{D'}(A(x)) + c \leqslant KS(x) + c$$

при некотором c и при всех x, что и требовалось доказать. □

Эта теорема гарантирует, что количество информации «не зависит от кодировки»— если мы, скажем, заменим в каком-то слове все нули на единицы и наоборот (или разбавим это слово нулями, добавив после каждой цифры по нулю), то полученное слово будет иметь ту же сложность, что и исходное (с точностью до константы), поскольку преобразования в ту и другую сторону выполняются некоторым алгоритмом.

По этой причине колмогоровская сложность определена не только для двоичных слов, но и для других «конструктивных объектов». Напрмер, сложность натурального числа n определяется, как $KS(n) = KS(f(n))$, где f — произвольная вычислимая инъективная функция, отображающая натуральные числа в двоичные слова (кодирование). По теореме 34 при смене f на другое $KS(n)$ изменится не более чем на константу. Аналогичным образом определяется сложность пары слов $KS(\langle x, y\rangle)$, как $KS([x, y])$ для некоторого кодирования $\langle x, y\rangle \mapsto [x, y]$, троек слов и так далее.

7.7 Сложность пары слов

Пусть x и y— два слова. Сколько битов информации содержится в паре, состоящей из x, y? Естественно ожидать, что количество информации в ней не превосходит суммы количеств информации в x и в y. И действительно это так, правда, с некоторой поправкой.

{intro-pair}

Теорема 35. *Существует такая константа c, что при любых x и y выполнено неравенство*

$$KS(\langle x, y\rangle) \leqslant KS(x) + 2\log KS(x) + KS(y) + c$$

Доказательство. Попробуем для начала доказать теорему без добавочного члена $2\log KS(x)$, её ослабляющего. Это естественно делать так. Пусть D— оптимальный способ описания, используемый при определении колмогоровской сложности. Рассмотрим новый способ описания D'. Именно, если $D(p) = x$ и $D(q) = y$, будем считать pq описанием

кода $[x,y]$ пары $\langle x,y\rangle$, то есть, положим $D'(pq) = [x,y]$. Тогда сложность $[x,y]$ относительно D' не превосходит длины слова pq, то есть $l(p)+l(q)$. Если в качестве p и q взять кратчайшие описания, то получится $KS_{D'}([x,y]) \leqslant KS_D(x) + KS_D(y)$, и остаётся воспользоваться оптимальностью D и перейти от D' к D (при этом возникает константа c).

Что неверно в этом рассуждении? Дело в том, что определение D' некорректно: мы полагаем $D'(pq) = [D(p), D(q)]$, но D' не имеет средств разделить p и q. Вполне может оказаться, что есть два разбиения слова на части, дающие разные результаты: $p_1q_1 = p_2q_2$, но $D(p_1) \ne D(p_2)$ или $D(q_1) \ne D(q_2)$.

Для исправления этой ошибки напишем перед словом pq длину слова p в двоичной записи. При этом в этой двоичной записи мы удвоим каждый бит, а после неё напишем 01, чтобы алгоритм мог понять, где кончается двоичная запись и начинается само p. Более точно, обозначим через $bin(k)$ двоичную запись числа k, а через $\overline{x}$ результат удвоения каждого бита в слове x. (Например, $bin(5) = 101$, а $\overline{bin(5)} = 110011$.) Теперь положим

$$D'(\,\overline{bin(l(p))}\,01pq) = D(p)D(q).$$

Это определение корректно: алгоритм D' сначала читает $\overline{bin(l(p))}$, пока цифры идут парами. Когда появляется группа 01, он определяет $l(p)$, затем отсчитывает $l(p)$ цифр и получает p. Остаток входа есть q, после чего уже можно вычислить $[D(p), D(q)]$.

Величина $KS_{D'}(\langle x,y\rangle)$ оценивается теперь числом $2l(bin(l(p))) + 2 + l(p)+l(q)$; двоичная запись числа $l(p)$ имеет длину не больше $\log_2 l(p)+1$, поэтому D'-описание слова xy имеет длину не более $2\log_2 l(p)+4+l(p)+l(q)$, откуда и вытекает утверждение теоремы. □

Заметим, что в теореме можно поменять местами p и q и доказать, что $KS(\langle x,y\rangle) \leqslant KS(x) + KS(y) + 2\log_2 KS(y) + c$. Будем в дальнейшем писать $KS(x,y)$ вместо $KS(\langle x,y\rangle)$. Константу 2 при логарифме в этой теореме можно понизить:

{condit-pair1}

Теорема 36.

$KS(x,y) \leqslant KS(x) + \log KS(x) + 2\log\log KS(x) + KS(y) + O(1)$;
$KS(x,y) \leqslant KS(x) + \log KS(x) + \log\log KS(x) + 2\log\log\log KS(x) + KS(y) + O(1)$;
. . .

(Последовательность утверждений теоремы можно продолжать неограниченно. Кроме того, можно поменять местами x и y.)

Доказательство. Назовём вычислимое отображение $x \mapsto \hat{x}$ множества двоичных слов в себя *бес-префиксным кодированием*, если ни для каких двух различных слов x и y слово $\hat{x}$ не является началом слова $\hat{y}$. (Отсюда, в частности, следует, что $\hat{x} \neq \hat{y}$ при $x \neq y$.) Смысл этого определения таков: любое слово вида $\hat{x}z$ можно однозначно разрезать на части и найти слова x и z.

Пример беспрефиксного кодирования: $x \mapsto \overline{x}01$, где $\overline{x}$ означает строку x с удвоенными битами. Здесь признаком окончания слова являются биты 01. Это кодирование не очень экономно (увеличивает длину вдвое). Более экономно такое кодирование:

$$x \mapsto \hat{x} = \overline{bin(l(x))}\,01x$$

(здесь $bin(l(x))$— двоичная запись длины слова x). Для него {prefix}

$$l(\hat{x}) = l(x) + 2\log l(x) + O(1).$$

Этот приём можно «итерировать»: начав с произвольного беспрефиксного кодирования $x \mapsto \hat{x}$, можно построить новое (также бес-префиксное) кодирование

$$x \mapsto \widehat{bin(l(x))}x.$$

В самом деле, если слово $\widehat{bin(l(x))}x$ является началом слова $\widehat{bin(l(y))}y$, то одно из слов $\widehat{bin(l(x))}$ и $\widehat{bin(l(y))}$ является началом другого, и потому $bin(l(x)) = bin(l(y))$. Отсюда мы заключаем, что x есть начало y, а потом— что $x = y$. (Другими словами, сначала мы однозначно декодируем длину слова, пользуясь тем, что она записана в бес-префиксном коде, а потом уже однозначно определяем само слово.)

Такая итерация даёт бес-префиксное кодирование, при котором

$$l(\hat{x}) = l(x) + \log l(x) + 2\log\log l(x) + O(1),$$

затем

$$l(\hat{x}) = l(x) + \log l(x) + \log\log l(x) + 2\log\log\log l(x) + O(1),$$

и так далее.

Вернёмся к доказательству теоремы. Пусть D— оптимальный способ описания, используемый при определении сложности. Рассмотрим способ описания D', задаваемый так:

$$D'(\hat{p}q) = [D(p), D(q)],$$

где $\hat{p}$— беспрефиксный код слова p, а квадратные скобки означают кодирование пар, использованное при определении сложности пары. Беспрефиксность кодирования $p \mapsto \hat{p}$ гарантирует корректность этого определения ($\hat{p}$ однозначно вычленяется из $\hat{p}q$).

Если p и q— кратчайшие описания слов x и y, то слово $\hat{p}q$ будет описанием слова $[x, y]$, и его длина как раз даёт оценку, указанную в теореме. (Чем более экономно бес-префиксное кодирование, тем лучше получается оценка; описанный выше итеративный метод построения беспрефиксных кодов даёт необходимые оценки.) □

Из теоремы 36 следует, что

$$KS(x, y) \leqslant KS(x) + KS(y) + O(\log n)$$

для слов x и y длины не более n. Как говорят, сложность пары не превосходит суммы сложностей её членов с точностью до логарифма (длин).

Возникает естественный вопрос: нельзя ли усилить оценку и доказать, что $KS(x, y) \leqslant KS(x) + KS(y) + O(1)$?

{condit-pair1-comment}

Следующее простое рассуждение показывает, что это невозможно. В самом деле, из этой оценки вытекало бы, что $KS(x, y) \leqslant l(x) + l(y) + O(1)$. Рассмотрим какое-то N и всевозможные $n = 0, 1, 2, \ldots, N$. Для каждого такого n имеется 2^n слов x длины n, а также 2^{N-n} слов y длины $N-n$. Комбинируя такие x и y, мы для данного n получаем 2^N пар $\langle x, y\rangle$, а всего (для всех n от 0 до N) получаем $(N+1)2^N$ пар. Если бы сложность всех этих пар $\langle x, y\rangle$ не превосходила $l(x) + l(y) + O(1) = N + O(1)$, то получилось бы $(N + 1)2^N$ различных слов $[x, y]$, имеющих сложность не более $N + O(1)$, а таких слов, как мы знаем (теорема 30, с. 118), лишь $O(2^N)$.

onditp-no-improvement}

Задача 88. Докажите, что не найдётся такого c, что

$$KS(x, y) \leqslant KS(x) + \log KS(x) + KS(y) + c$$

при всех x и y. [Указание. Замените в правой части неравенства сложности на длины и подсчитайте количество пар.]

Задача 89. Дайте естественное определение сложности тройки слов. Покажите, что $KS(x,y,z) \leqslant KS(x) + KS(y) + KS(z) + O(\log n)$, если слова x, y, z имеют длину не больше n.

Задача 90. (а) Покажите, что если $x \mapsto \hat{x}$— бес-префиксное кодирование, то

$$\sum_{x\in\Xi} 2^{-l(\hat{x})} \leqslant 1$$

(здесь Ξ— множество всех двоичных слов).

(б) Покажите, что если бес-префиксное кодирование увеличивает длину слова не более чем на $f(n)$ (где n— исходная длина), то есть $l(\hat{x}) \leqslant l(x) + f(l(x))$, то ряд $\sum_n 2^{-f(n)}$ сходится.

Насколько неравенство теоремы 35 близко к равенству? Может ли $KS(x,y)$ быть существенно меньше суммы $KS(x) + KS(y)$? В полном согласии с нашей интуицией это возможно, если x и y содержат много общего. Например, при $x = y$ мы имеем $KS(x,x) = KS(x) + O(1)$, поскольку $[x,x]$ алгоритмически получается из x и обратно (теорема 34).

Чтобы уточнить это наблюдение, мы определим понятие количества информации, которая содержится в x, но не содержится в y (для произвольных слов x и y). Эту величину называют *условной колмогоровской сложностью x при условии y* и обозначают через $KS(x|y)$ (говорят также «при известном y», «относительно y»). Её определение аналогично определению обычной (безусловной) сложности, но декомпрессор D имеет доступ не только к сжатому описанию, но и к слову y.

Если провести аналогию между энтропией Шеннона и колмогоровской сложностью, то $KS(x|y)$ аналогична условной энтропии $H(\xi|\eta)$ (а $KS(x)$ аналогична обычной безусловной энтропии $H(\xi)$).

7.8 Условная колмогоровская сложность

{condit-c}

Посылая файл по электронной почте, можно сэкономить, если послать не сам файл, а его сжатый вариант (кратчайшее описание). Экономия может быть ещё больше, если получатель уже имеет старую версию файла— в таком случае достаточно описать внесённые изменения. Эти соображения приводят к следующему определению *условной сложности слова x при известном слове y*.

Назовём *способом условного описания* произвольную вычислимую функцию D двух аргументов (аргументы и значения функции D являются

двоичными словами). Первый аргумент мы будем называть *описанием*, второй— *условием*. Если $D(y,z) = x$, мы говорим, что слово y является *описанием* слова x *при известном* z (ещё говорят «при условии z», «относительно z»). Сложность $KS_D(x|z)$ определяется как длина кратчайшего описания слова x при известном слове z:

$$KS_D(x|z) = \min\{l(y) \mid D(y,z) = x\}.$$

Говорят, что способ (условного) описания D_1 *не хуже* способа D_2, если найдётся такая константа c, что

$$KS_{D_1}(x|z) \leqslant KS_{D_2}(x|z) + c$$

для любых слов x и z. Способ (условного) описания D называется *оптимальным*, если он не хуже любого другого способа (условного) описания.

{condit-universal}

Теорема 37. *Существует оптимальный способ условного описания.*

Доказательство. Этот вариант теоремы Колмогорова–Соломонова доказывается точно так же, как и безусловный (см. теорему 29, с. 116).

Именно, фиксируем некоторый способ программирования для функций двух аргументов, при котором программы записываются в виде двоичных слов, и положим

$$D(\hat{p}y, z) = p(y,z),$$

где $p(y,z)$ обозначает результат применения программы p ко входам y и z, а $\hat{p}$ есть беспрефиксный код слова p. Теперь легко проверить, что если D'— произвольный способ условного описания, а p— соответствующая ему программа, то

$$KS_D(x|z) \leqslant KS_{D'}(x|z) + l(\hat{p}).$$

□

Как и прежде, мы фиксируем некоторый оптимальный способ D условного описания и опускаем индекс D в KS_D. Соответствующую функцию мы называем *условной колмогоровской сложностью*; как и безусловная, она определена с точностью до ограниченного слагаемого.

Вот несколько простых свойств условной колмогоровской сложности:

{condit-basic}

Теорема 38.

$$\begin{aligned} KS(x|y) &\leqslant KS(x) + O(1);\\ KS(x|x) &= O(1);\\ KS(f(x,y)|y) &\leqslant KS(x|y) + O(1);\\ KS(x|y) &\leqslant KS(x|g(y)) + O(1). \end{aligned}$$

Здесь g и f— произвольные вычислимые функции (одного и двух аргументов); имеется в виду, что указанные в теореме неравенства выполнены, если $f(x,y)$ (соответственно $g(y)$) определено.

Доказательство. Безусловный способ описания можно рассматривать и как условный (не зависящий от второго аргумента), отсюда получаем первое неравенство. Второе утверждение получается, если рассмотреть способ описания $D(p,z) = z$. Третье неравенство доказывается так: пусть D— оптимальный способ условного описания; рассмотрим другой способ

$$D'(p,y) = f(D(p,y),y)$$

и сравним его с оптимальным. Доказательство последнего неравенства аналогично, только нужно определить способ описания так:

$$D'(p,y) = D(p,g(y)).$$

□

Задача 91. Покажите, что условная сложность «непрерывна по второму аргументу»: $KS(x|y0) = KS(x|y) + O(1)$; $KS(x|y1) = KS(x|y) + O(1)$.

Задача 92. Покажите, что при фиксированном y функция $x \mapsto KS(x|y)$ отличается от KS не более чем на константу, зависящую от y (и не превосходящую $2KS(y) + O(1)$).

Задача 93. Докажите, что $KS([x,z]|[y,z]) \leqslant KS(x|y) + O(1)$ для любых x, y, z (квадратные скобки означают вычислимое кодирование пар).

{conditional-as-proble

Задача 94. Пусть фиксирован некоторый разумный язык программирования. (Формально говоря, нужно, чтобы соответствующая ему нумерация была главной, то есть чтобы была возможна эффективная трансляция программ с других языков [1].) Покажите, что условная сложность

$KS(x|y)$ равна (с точностью до $O(1)$) минимальной сложности программы, которая даёт x на входе y. [Указание. Если D— оптимальный способ условного описания, то сложность программы, которая получается из D фиксацией первого аргумента p, не превосходит $l(p) + O(1)$. С другой стороны, если программа p переводит y в x, то $KS(x|y) = KS(p(y)|y) \leqslant KS(p) + O(1)$.]

Многие свойства безусловной сложности легко переносятся и на условную. Вот некоторые из них (доказательства повторяют соответствующие рассуждения для безусловной сложности):

- Функция $KS(x|y)$ перечислима сверху (это означает, что множество троек $\langle x, y, n\rangle$, для которых $KS(x|y) < n$, перечислимо).
- При данных y и n множество тех слов x, для которых $KS(x|y) < n$, содержит менее 2^n элементов. Отсюда следствие:
- Для всякого y и для всякого n найдётся слово x длины n, сложность которого при известном y не меньше n.

Задача 95. Докажите, что для любых слов y и z и для любого n найдётся слово x длины n, у которого $KS(x|y) \geqslant n-1$ и $KS(x|z) \geqslant n-1$. [Указание: плохие слова каждого типа образуют менее половины.]

{condit-ub}

Теорема 39. *Если $\langle x, y\rangle \mapsto K(x|y)$— произвольная перечислимая сверху функция, причём для любых y и n множество*

$$\{x \mid K(x|y) < n\}$$

содержит менее 2^n элементов, то $KS(x|y) \leqslant K(x|y) + c$ при некотором c и при любых x и y.

Доказательство этой теоремы почти дословно повторяет доказательство теоремы 32, поэтому мы его опускаем.

Понятие условной сложности позволяет придать смысл вопросу о том, сколько новой информации в ДНК одного организма по сравнению с ДНК другого: если d_1— двоичное слово, кодирующее ДНК первого организма, а d_2— двоичное слово, кодирующее ДНК второго, то искомая

величина есть $KS(d_1|d_2)$. Аналогичным образом можно спрашивать, какой процент информации был потерян при переводе романа на другой язык: в этом случае нас интересует отношение

$$KS(\text{оригинал}|\text{перевод})/KS(\text{оригинал}).$$

Разность $KS(x) - KS(x|y)$ естественно назвать количеством информации об x в y: эта разность показывает, насколько знание слова y упрощает описание слова x.

В теореме о сложности пары (теорема 36, с. 126) также можно заменить сложность на условную:

{condit-pair2}

Теорема 40.

$$KS(x, y) \leqslant KS(x) + 2\log KS(x) + KS(y|x) + O(1)$$

Доказательство. Пусть D_1— оптимальный способ безусловного описания, а D_2— оптимальный способ условного описания. Построим способ описания D', положив

$$D'(\hat{p}q) = [D_1(p), D_2(q, D_1(p))].$$

Здесь $\hat{p}$— беспрефиксный код слова p, а квадратные скобки обозначают вычислимое кодирование пар, используемое при определении сложности пары. Если p— кратчайшее описание слова x, а q— кратчайшее описание слова y при известном x, то слово $\hat{p}q$ будет описанием слова $[x, y]$ относительно D', откуда

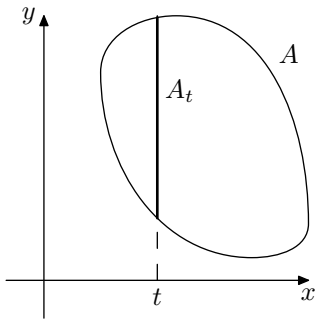
$$KS(x, y) \leqslant KS_{D'}(x, y) + O(1) \leqslant l(\hat{p}) + l(q) + O(1).$$

Осталось заметить, что можно выбрать беспрефиксное кодирование так, чтобы $l(\hat{p})$ не превосходило $l(p) + 2\log l(p) + O(1)$ (см. доказательство теоремы 36 о сложности пары, с. 126). □

В этой теореме, как и раньше, можно заменить $2\log KS(x)$ на $\log KS(x) + 2\log\log KS(x)$ и так далее. Можно также добавочный член перенести на условную сложность, написав

$$KS(x, y) \leqslant KS(x) + KS(y|x) + 2\log KS(y|x) + O(1).$$

В доказательстве при этом следует заменить $D'(\hat{p}q)$ на $D'(\hat{q}p)$.



{condit-c.1} Рис. 7.1: Сечение A_t множества A простых пар.

Задача 96. Докажите «неравенство треугольника»:

$$KS(x|z) \leqslant KS(x|y) + 2\log KS(x|y) + KS(y|z) + O(1)$$

для любых трёх слов x, y, z.

Если не вдаваться в тонкости различных вариантов оценки добавочного члена с логарифмом, результат теоремы 40 можно сформулировать так: для слов x и y длины не более n имеет место неравенство

$$KS(x, y) \leqslant KS(x) + KS(y|x) + O(\log n).$$

Оказывается, что это неравенство на самом деле представляет собой равенство (с той же логарифмической точностью).

{condit-pair3}

Теорема 41 (Колмогорова–Левина)**.**

$$KS(x, y) = KS(x) + KS(y|x) + O(\log n).$$

для слов x, y длины не больше n.

Доказательство. В одну сторону неравенство уже доказано. Осталось доказать, что $KS(x, y) \geqslant KS(x) + KS(y|x) + O(\log n)$, если x и y— слова длины не более n.

Пусть x, y— произвольные слова длины не более n. Обозначим сложность $KS(x, y)$ пары $\langle x, y\rangle$ через a. Пусть A— множество всех пар слов, у которых сложность не больше a. Число элементов в множестве A не больше $O(2^a)$ (точнее, меньше 2^{a+1}), и пара $\langle x, y\rangle$— один из них.

Для каждого слова t рассмотрим сечение A_t множества A:

$$A_t = \{u \mid \langle t, u\rangle \in A\}$$

(рис. 7.1). Сумма мощностей всех множеств A_t при всех t равна мощности множества A, то есть не превосходит $O(2^a)$. Поэтому больши́х сечений у множества A немного, что мы сейчас и используем.

Пусть $m = \lfloor \log_2 |A_x| \rfloor$, где x— первая компонента исходной пары. Другими словами, пусть число элементов в A_x заключено между 2^m и 2^{m+1}. Докажем, что $KS(y|x)$ не сильно превосходит m, а $KS(x)$ не сильно превосходит $a - m$. Начнём с первого.

Зная a, можно перечислять множество A. Если мы к тому же знаем x, то можно оставлять от A только пары, у которых первая координата равна x, и получить перечисление множества A_x. Чтобы задать y, помимо a и условия x, достаточно указать порядковый номер элемента y в этом перечислении. Для этого достаточно $m + O(1)$ битов, и вместе с a получается $m + O(\log n)$ битов. Заметим, что $a = KS(x, y)$ не превосходит $O(n)$, если x и y— слова длины не более n, а потому для задания a достаточно $O(\log n)$ битов. Итак,

$$KS(y|x) \leqslant m + O(\log n).$$

Перейдём ко второй оценке. Множество B всех t, для которых $|A_t| \geqslant 2^m$, содержит не более $2^{a+1}/2^m$ элементов (иначе сумма $|A| = \sum |A_t|$ была бы больше 2^{a+1}). Множество B можно перечислять, зная a и m. (В самом деле, надо перечислять пары в множестве A; как только найдётся 2^m пар с одинаковой первой координатой, эта координата помещается в перечисление множества B.) Тем самым исходное слово x (как и любой другой элемент множества B) можно задать, указав $(a - m) + O(\log n)$ битов ($a - m$ битов уходят на порядковый номер слова x в B, а $O(\log n)$ битов позволяют дополнительно указать a и m). Отсюда

$$KS(x) \leqslant (a - m) + O(\log n),$$

и остаётся сложить два полученных неравенства. □

Доказанную только что теорему можно рассматривать как перевод на язык колмогоровской сложности такого комбинаторного факта. Пусть дано множество A пар слов. Его мощность не превосходит произведения

мощности проекции на первую координату и мощности наибольшего сечения A_t (первая координата равна t, вторая произвольна). Это соответствует неравенству $KS(x,y) \leqslant KS(x) + KS(y|x) + O(\log n)$. Обратное неравенство интерпретируется сложнее. Пусть дано множество A пар слов, а также числа p и q, для которых $|A| \leqslant pq$. Тогда можно разбить A на две части P и Q с такими свойствами: проекция P на первую координату содержит не более p элементов, а все сечения Q_x множества Q (первая координата равна x, вторая произвольна) содержат не более q элементов. (В самом деле, если отнести к P те сечения, которые содержат больше q элементов, то их число не превосходит p.)

Заметим, что на самом деле для доказательства важны не длины слов x и y, а их сложности. По существу мы доказали такой факт:

{condit-pair3a}

Теорема 42 (Колмогорова–Левина, вариант со сложностью)**.**

$$KS(x,y) = KS(x) + KS(y|x) + O(\log KS(x,y))$$

для любых слов x, y.

Задача 97. Оцените более точно константы в проведённом доказательстве и покажите, что

$$KS(x) + KS(y|x) \leqslant KS(x,y) + 2\log KS(x,y) + O(\log\log KS(x,y)).$$

[Указание. Для задания y при известном x, кроме номера y в A_x нужно знать $a = KS(x,y)$, для чего достаточно $\log a + O(\log\log a$ бит, включая информацию о длине записи a в префиксном виде. Для задания x достаточно знать его номер среди слов t с толстыми сечениями, записанный ровно $a+1-m$ битами, и число a, поскольку m можно найти по $a+1-m$ и a.]

Задача 98. Покажите, что в теореме Колмогорова–Левина члены порядка $O(\log n)$ неизбежны (причём в обе стороны): при любом n найдутся слова x и y длины не более n, для которых

$$KS(x,y) \geqslant KS(x) + KS(y|x) + \log n - O(1),$$

а также слова x и y длины не более n, для которых

$$KS(x,y) \leqslant KS(x) + KS(y|x) - \log n + O(1).$$

[Указание. В первом случае можно воспользоваться замечанием после теоремы 36 (с. 128). Во втором случае можно взять в качестве x число между $n/2$ и n, для которого $KS(x) = \log n + O(1)$, а затем в качестве y взять слово длины x, для которого $KS(y|x) = x + O(1)$.]

Задача 99. Покажите, что изменение одного бита в слове длины n меняет его сложность не более чем на $\log n + O(\log\log n)$.

{number-of-description

Задача 100. Фиксируем некоторый (безусловный) способ описания D. Докажите, что для некоторой константы c и для всех n и k выполнено такое свойство: если какое-то слово x имеет 2^k описаний длины не более n, то $KS(x|k) \leqslant n - k + c$. [Указание. Пусть k фиксировано. Для каждого n рассмотрим слова x, имеющие не менее 2^k описаний длины не более n. Число таких слов (при данном k) не превосходит 2^{n-k}, и можно воспользоваться теоремой 39, с. 132.]

С помощью этой задачи можно доказать такое утверждение о безусловной сложности:

Задача 101. Пусть D— фиксированный (безусловный) способ описания. Тогда найдётся такое число c, что для любого слова x число кратчайших D-описаний слова x не превосходит c. [Указание. В условиях предыдущей задачи $KS(x) \leqslant n - k + 2\log k + O(1)$, поэтому при $KS(x) = n$ значение k ограничено.]

Задача 102. Докажите, что найдётся константа c с таким свойством: если для данных x и n вероятность события $KS(x|y) \leqslant k$ (для случайно взятого слова y длины n; все такие слова считаем равновероятными) не меньше 2^{-l}, то $KS(x|n, l) \leqslant k + l + c$. [Указание. Соединим каждое слово y длины n со всеми словами x, сложность которых относительно y не превосходит k, получим двудольный граф с $O(2^{n+k})$ рёбрами, и в нём число вершин x, из которых выходит не менее 2^{n-l} рёбер, есть $O(2^{k+l})$. Обратите внимание, что в $KS(x|n, l)$ не входит k— это не опечатка!]

Эта задача позволяет ответить на такой вопрос: чему в среднем равна сложность $KS(x|y)$ для данного x и случайно выбранного слова y данной длины n? Ясно, что $KS(x|y) \leqslant K(x|n) + O(1)$ (поскольку $n = l(y)$ восстанавливается по y). Оказывается, что для большинства слов y данной длины эта оценка точная:

Задача 103. Докажите, что найдётся такая константа c, что для любого слова x и для любых натуральных чисел n и d доля тех слов y длины n, у которых $KS(x|y) < KS(x|n) - d$ (среди всех слов длины n), не превосходит

$cd^2/2^d$. Выведите отсюда, что среднее арифметическое $KS(x|y)$ по всем словам y данной длины n равно $KS(x|n) + O(1)$ (константа в $O(1)$ не зависит от x и n).

Задача 104. Докажите, что $KS(x) = KS(x|KS(x)) + O(1)$. [Указание. Пусть x имеет короткое описание q при известном $KS(x)$. Тогда для восстановления x достаточно задать q и разницу длин $KS(x) - l(q)$, и получается более короткое описание слова x, чем это возможно.]

7.8.1 Количество информации

{conditi}

Мы знаем (теорема 38), что условная сложность $KS(y|x)$ не превосходит безусловной сложности $KS(y)$ (с точностью до константы). Разность $KS(y) - KS(y|x)$ показывает, насколько знание слова x упрощает описание слова y. Поэтому её называют *количеством информации в слове x о слове y*. Обозначение: $I(x : y)$.

Теорема 38 показывает, что информация $I(x : y)$ неотрицательна (с точностью до константы): существует такое c, что $I(x : y) \geqslant c$ при всех x и y.

Вспомнив, что

$$KS(x, y) = KS(x) + KS(y|x) + O(\log KS(x, y)),$$

(теорема 42, с. 136), можно выразить условную сложность через безусловные: $KS(y|x) = KS(x, y) - KS(x) + O(\log KS(x, y))$. Тогда для информации получается выражение:

$$I(x : y) = KS(y) - KS(y|x) = KS(x) + KS(y) - KS(x, y) + O(\log KS(x, y)).$$

Отсюда сразу вытекает

{condit-symm}

Теорема 43 (симметрия информации)**.**

$$I(x : y) = I(y : x) + O(\log KS(x, y))$$

Эта теорема гарантирует, что разница между $I(x : y)$ и $I(y : x)$ логарифмически мала по сравнению с $KS(x, y)$. Как показывает следующая задача, эта разница может быть сравнима с самими значениями $I(x : y)$ и $I(y : x)$, если те много меньше $KS(x, y)$.

Задача 105. Покажите, что если x— слово длины n, для которого $KS(x|n) \geqslant n$, то $I(x : n) = KS(n) + O(1)$, в то время как $I(n : x) = O(1)$.

Симметрия (пусть и не полная, а лишь с точностью до логарифмического слагаемого) позволяет называть $I(x:y)$ (или $I(y:x)$) *взаимной информацией* слов x и y. Соотношения между взаимной информацией, условными сложностями и сложностью пары можно изобразить на символической картинке (рис. 7.2).

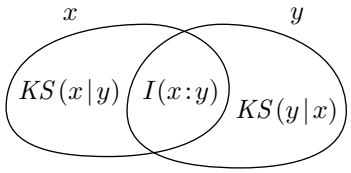


Рис. 7.2: Взаимная информация и условная сложность {condit-i.1}

На ней показано, что слова x и y имеют $I(x:y) \approx I(y:x)$ битов общей информации. Добавив $KS(x|y)$ битов (информация, которая есть в x, но не в y, левая область), мы получаем

$$I(y:x) + KS(x|y) \approx (KS(x) - KS(x|y)) + KS(x|y) \approx KS(x)$$

битов (как и должно быть в слове x). Аналогичным образом центральная часть вместе с $KS(y|x)$ справа дают $KS(y)$. Наконец, все три области вместе складываются в

$$KS(x|y)+I(x:y)+KS(y|x) = KS(x)+KS(y|x) = KS(x|y)+KS(y) = KS(x,y)$$

(все равенства верны с точностью $O(\log n)$, если слова x и y имеют длину не больше n).

7.9 Сложность и энтропия {complexity-and-entrop

Рассмотренные нами свойства шенноновской энтропии, условной энтропии и взаимной информации (для случайных величин) сходны с аналогичными утверждениями о колмогоровской сложности. Можно ли каким-либо образом уточнить и формализовать эти аналогии?

7.9.1 Колмогоровская сложность и энтропия частот {frequencies-entropy}

Будем рассматривать произвольный (не обязательно двух-буквенный) конечный алфавит A и слова в этом алфавите. Колмогоровская сложность для них определяется обычным образом.

Пусть фиксирован алфавит A, содержащий k букв. Рассмотрим произвольное слово x некоторой длины N в этом алфавите. Пусть $p_1, \ldots, p_k$— частоты появления букв в слове x. Все они являются дробями со знаменателем N; сумма частот равна единице. Пусть $h(p_1, \ldots, p_k)$— шенноновская энтропия соответствующего распределения.

complexity-le-entropy}

Теорема 44.

$$KS(x) \leqslant h(p_1, \ldots, p_k) \cdot N + O(\log N).$$

Имеется в виду, что $O(\log N)$ не больше $c \log N$, причём константа c не зависит ни от N, ни от слова x, ни от частот $p_1, \ldots, p_k$, но может зависеть от k, так что в этой теореме размер алфавита считается фиксированным.

Доказательство. Применим к x кодирование на основе частот букв (с. 74). Длина полученного кода есть $Nh(p_1, \ldots, p_k) + O(\log N)$. Длина задания N и частот $p_1, \ldots, p_n$ входят в $O(\log N)$. Поскольку для кодирования на основе частот букв кодирующая и декодирующая функции вычислимы, мы и получаем неравенство теоремы. □

Заметим, что неравенство теоремы 44 может быть как угодно далеко от равенства: если, скажем, алфавит состоит из двух букв, и в слове x они чередуются, то правая часть равна 1, а левая имеет порядок $(\log N)/N$. Что и не удивительно— правая часть учитывает только частоты, а не другие (не-частотные) закономерности. В следующем разделе мы покажем, что при случайном порождении слова сложность близка к энтропии.

7.9.2 Математическое ожидание сложности

Пусть фиксирован k-буквенный алфавит A и k неотрицательных чисел $p_1, \ldots, p_k$ с суммой 1 (которые мы для простоты считаем рациональными).

Рассмотрим случайную величину ξ, значениями которой являются буквы алфавита A, принимаемые с вероятностями $p_1, \ldots, p_k$. Для каждого N рассмотрим случайную величину ξ^N, соответствующую N независимым копиям случайной величины ξ. Её значениями являются слова длины N в алфавите A (каждая буква порождается независимо, причём вероятность появления i-ой буквы равна p_i). Нас будет интересовать математическое ожидание сложности значений случайной величины ξ^N (взвешенное среднее с весами, равными вероятностям).

ed-complexity}

Теорема 45. *Математическое ожидание величины $KS(\xi^N)$ равно $NH(\xi)+O(\log N)$ (константа в $O(\log N)$ может зависеть от ξ, но не зависит от N).*

Заметим, что (при положительных p_i) среди значений величины ξ^N будут встречаться все слова длины N в алфавите A; некоторые из них имеют сложность много больше NH (если только распределение вероятностей не равномерное, поскольку в последнем случае таких слов нет), а другие имеют сложность много меньше NH.

Доказательство. Для каждого слова длины N (то есть для каждого значения случайной величины ξ^N) рассмотрим его кратчайшее описание (относительно оптимального способа описания). Полученные слова образуют инъективный код. Средняя длина этого кода как раз равна математическому ожиданию сложности значений величины ξ^N. Поэтому по задаче 41 она не может быть меньше $H(\xi^N) - 2\log H(\xi^N) - 2$. Поскольку $H(\xi^N) = NH(\xi) = O(N)$, нижняя оценка доказана.

А теорема 3 позволяет получить верхнюю оценку. В самом деле, она утверждает существование префиксного кода, имеющего среднюю длину кодового слова не выше $H + 1$. Такой код можно алгоритмически построить, зная число N (и числа p_i, предполагаемые фиксированными). Например, можно использовать конструкцию из доказательства теоремы 3, или применить код Хаффмана, или даже просто перебирать все коды, пока не найдётся подходящий.

Так или иначе построенный код можно рассматривать как некоторый способ условного описания (при известном N), для которого средняя префиксная сложность значения величины ξ^N не превосходит $H(\xi^N) + 1 = NH(\xi) + 1$. Переход к оптимальному способу описания увеличит среднюю сложность не более чем на константу. □

Доказанная теорема показывает, что *в среднем* сложность равна энтропии, хотя она бывает и больше, и меньше её. На самом деле верно и более сильное утверждение: почти что с единичной вероятностью сложность близка к энтропии, и вероятность того, что случайно выбранное значение величины ξ^N будет иметь сложность, сильно отличающуюся от энтропии, мала. Это утверждение является алгоритмическим аналогом теоремы Шеннона о пропускной способности канала без шума и следует из нее.

С одной стороны, в качестве декодирующей функции в теореме Шеннона можно рассмотреть оптимальный способ описания, а в качестве кодирующей функции — (невычислимый) компрессор, находящий кратчайшее описание. Поэтому для каждого $\varepsilon > 0$ существует c_ε такое, что вероятность события $KS(\xi^N) < NH(\xi) - c_\varepsilon\sqrt{N}$ меньше ε. С другой стороны, декодирующая функция, примененная в доказательстве теоремы Шеннона, вычислима, поэтому с вероятностью не менее $1-\varepsilon$ выполнено $KS(\xi^N|N, p_1, \ldots, p_k, \varepsilon) \leqslant NH(\xi) + c_\varepsilon\sqrt{N}$, следовательно, $KS(\xi^N) \leqslant NH(\xi) + c_\varepsilon\sqrt{N} + O(\log N)$.

7.10 Применения колмогоровской сложности

Прежде всего оговоримся: речь пойдёт не о практических применениях, а о связи колмогоровской сложность с другими вопросами.

7.10.1 Бесконечность множества простых чисел

{appl-primes}

Начнём с совсем «игрушечного» применения: докажем, что множество простых чисел бесконечно.

Пусть это не так и существует всего m различных простых чисел $p_1, \ldots, p_m$. Тогда любое целое число x разлагаясь на простые множители, представляется в виде

$$x = p_1^{k_1} p_2^{k_2} \ldots p_m^{k_m}$$

и тем самым может быть задано набором степеней $k_1, \ldots, k_m$. Каждое из чисел k_i не превосходит $\log x$ (основания степеней не меньше 2) и потому имеет сложность не более $O(\log\log x)$ (двоичная запись содержит $O(\log\log x)$ битов). Поскольку m фиксировано, то сложность набора $\langle k_1, k_2, \ldots, k_m\rangle$ есть $O(\log\log x)$ и потому сложность любого числа x (которое алгоритмически получается из этого набора) есть $O(\log\log x)$. А для «случайного» n-битового числа x сложность примерно равна n, а не $O(\log n)$, как получается по этой формуле (логарифм n-битового числа не превосходит n).

Можно ли считать это «честным» применением колмогоровской сложности? Скептик скажет, что здесь всего лишь производится обычный подсчёт числа возможностей (counting argument, как говорят): если существует лишь m различных простых чисел, то различных чисел от 1

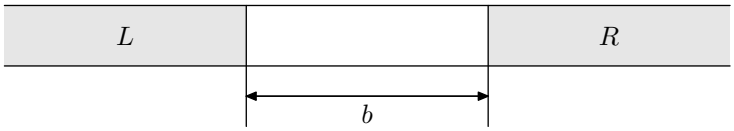


Рис. 7.3: Буферная зона размера b. {tape-buffer}

до x существует не более $(\log x)^m$, поскольку каждое такое число задаётся показателями степеней при простых множителях, и каждый показатель меньше $\log x$. После чего мы приходим к противоречию, поскольку $x > (\log x)^m$ при больших x.

Возразить на это нечего— именно это рассуждение и пересказано с помощью колмогоровской сложности (и стало немного более громоздким за счёт различных асимптотических обозначений). Тем не менее в подобном переводе может быть некоторый смысл, поскольку новый язык формирует новую интуицию, и она может быть полезна, даже если потом можно всё пересказать на старом языке.

7.10.2 Перенос информации по ленте

{appl-tape}

Следующий пример тоже модельный— мы докажем, что копирование слова длины n на одноленточной машине Тьюринга требует (в худшем случае) не менее εn^2 шагов. Этот классический результат был получен в 1960-ые годы с помощью так называемого «метода следов»; наше доказательство является переводом классического на язык колмогоровской сложности. (Мы предполагаем известными основные понятия, связанные с машинами Тьюринга; см., например, [1]).

Пусть на ленте машины Тьюринга выделена «буферная зона» некоторого размера b; нас интересует скорость переноса информации через эту буферную зону, скажем, слева направо, из области L в область R (рис. 7.3)

В некоторый момент ленты буферная область и R пусты, а область L содержит произвольную информацию. Нас интересует, какова может быть сложность слова R через t шагов после этого. Мы докажем, что она не больше $(t \log m)/b + O(\log t)$, где m— число состояний машины Тьюринга, а b— ширина буферной зоны. В самом деле, каждое из m состояний машины несёт $\log m$ битов информации, и за шаг информация

переносится на одну клетку, а перенос её на b клеток требует в b раз больше времени.

Осталось лишь уточнить формулировку и доказательство.

Теорема 46. *Пусть фиксирована машина Тьюринга с m состояниями. Тогда существует такая константа c, что для любого b и для любого вычисления с буферной зоной b (вначале эта зона и лента справа от неё пусты, головка машины слева от зоны) сложность правой части ленты $R(t)$ после t шагов вычисления не превосходит*

$$\frac{t \log m}{b} + 4 \log t + c.$$

Доказательство. Проведём границу где-нибудь посередине буферной зоны, и при каждом пересечении границы головкой машины Тьюринга слева направо (при «выезде за границу») будем записывать, в каком состоянии она её пересекла. Записанная последовательность состояний называется *следом* машины. Зная след, можно восстановить работу машины справа от границы (без использования информации о ленте слева от границы). В самом деле, надо искусственно поместить машину в первое состояние, указанное в следе, и выпустить её за границу; по возвращении перевести её во второе состояние следа и снова выпустить и так далее. При этом за границей машина будет вести себя так же, как и раньше (ведь ничего, кроме состояния, при пересечении границы у ней нет). В частности, в некоторый момент t' состояние ленты будет содержать $R(t)$. При этом t' не превосходит t, хотя может быть и меньше (поскольку время, проведённое машиной слева от границы, теперь не учитывается). Таким образом, можно алгоритмически восстановить $R(t)$, зная след, t', а также расстояние b' от границы до начала зоны R. Поэтому найдётся такая константа c (зависящая от машины, но не от b и t), что для любого следа S

$$KS(R(t)) \leqslant l(S) \log m + 4 \log t + c$$

(мы умножили длину $l(S)$ следа S на $\log m$, поскольку след является словом в m-буквенном алфавите и на каждую букву приходится $\log m$ битов; приписывание к нему чисел b' и t' (в само-ограниченной записи) требует не более $2 \log b + 2 \log t$ битов, причём можно считать, что $t > b$ (иначе R пуста, головка туда не дошла); константа c возникает при переходе к оптимальному способу описания).

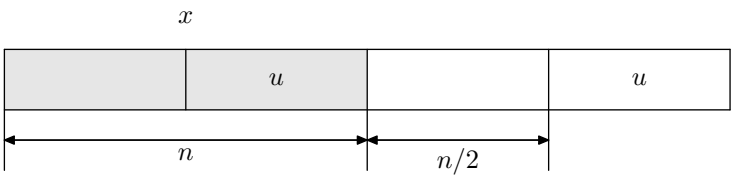


Рис. 7.4: Буферная зона при копировании. {tape-copy-buffer}

Это неравенство верно для любого начального содержимого части L и для любого положения границы: если теперь для данного L взять самый короткий из следов, то его длина меньше t/b (всего разных положений границы $b+1$, и в каждый момент пересечение происходит лишь в одном месте, так что сумма длин следов не больше t). Получаем требуемое неравенство. □

Задача 106. Покажите, что оценку теоремы можно улучшить, заменив b в знаменателе на $2b$. [Указание. Въезд суммарно потребует почти столько же шагов, сколько и выезд (не более чем на b меньше)]

Теперь сразу же получается квадратичная оценка для задачи копирования.

Пусть одноленточная машина M копирует любое входное слово: если вначале на ленте написано слово x из нулей и единиц, а дальше лента пуста, то в конце работы справа от x появляется его копия, и на ленте написано xx.

Теорема 47. *Существует такая константа $\varepsilon > 0$, что для любого n существует слово длины n, копирование которого с помощью M занимает не менее εn^2 шагов.*

Доказательство. Будем для простоты считать, что n чётно, и возьмём в качестве x слово, у которого вторая половина u имеет сложность не меньше длины (то есть $n/2$). Применим теорему о скорости переноса информации, считая буферной зоной участок длины $n/2$ справа от x (рис. 7.4).

Пусть копирование заняло t шагов, тогда сложность зоны R после этого не меньше $n/2$; с другой стороны, она не превосходит $t \log m/b + 4 \log t +$ по доказанной теореме, где b— ширина буферной зоны, то есть

$n/2$. Получаем, что

$$\frac{n}{2} \leqslant \frac{t \log m}{n/2} + 4 \log t + c.$$

Без ограничения общности считаем, что $t < n^2$ (иначе всё и так хорошо) и заменяем $4 \log t$ на $8 \log n$. Получаем, что

$$t \geqslant \frac{n^2}{4 \log m} - O(n \log n).$$

Второй член мал по сравнению с первым при больших n (и формально можно распространить неравенство на все n за счёт уменьшения коэффициента при n^2). □

Насколько существенна в этом доказательстве колмогоровская сложность? Скептик может вновь сказать, что по существу мы просто оценивали число слов, которые можно скопировать за данное время, используя тот факт, что разные слова должны давать разные следы (иначе справа от границы поведение машины было бы одно и то же). Действительно, именно так и выглядело первоначальное доказательство этой оценки (уточним, что в первоначальных работах рассматривалось не копирование, а распознавание симметрии, но вся техника та же самая). Насколько идея этого доказательства становится понятнее при переводе его на язык колмогоровской сложности— вопрос вкуса.

Многие оценки в теории сложности вычислений могут быть доказаны аналогичным образом, когда в качестве «трудного случая» рассматривается слово максимальной сложности в некотором классе и затем показывается, что если бы оценка не соблюдалась, то это слово имело бы меньшую сложность. Много таких приложений (со ссылками на оригинальные работы) приведено в книге Ли и Витаньи [9], которые сами сыграли большую роль в развитии этого метода (называемого *Incompressibility method*). Отметим, что во многих случаях оценка впервые была доказана как раз с использованием колмогоровской сложности.

7.10.3 Бритва Оккама

Мы закончим следующим философским вопросом: что значит, что теория хорошо объясняет результаты эксперимента? Пусть имеется какой-то набор экспериментальных данных и разные теории, его объясняющие. Например, экспериментальными данными могут быть положения

планет на небесной сфере. Их можно объяснять в духе Птолемея, рассматривая эпициклы и дифференты (и внося дополнительные поправки при необходимости), а можно ссылаться на законы современной механики. Как объяснить, чем современная теория лучше? Можно сказать так: современная теория позволяет вычислять положения планет с той же (и даже лучшей) точностью, имея меньше параметров. Условно говоря, достижение Кеплера состоит в том, что он нашёл более короткое описание для экспериментальных данных. Совсем грубо можно сказать, что экспериментаторы получают двоичные слова, после чего теоретики ищут для этих слов короткие описания (и тем самым верхние оценки на сложность), и лучше тот теоретик, у которого описание короче (оценка меньше).

Этот подход называют иногда «бритвой Оккама» по имени философа, который говорил, что не следует множить сущности без необходимости. Насколько такое толкование одобрил бы сам Оккам, сказать трудно.

Можно использовать ту же идею в более практической ситуации. Представим себе, что мы собираемся автоматически читать надписи на конвертах и ищем правило, отделяющее изображения нулей от изображений единиц. (Будем считать, что изображение дано в виде матрицы битов, записанной как двоичное слово.) У нас есть несколько тысяч образцов— картинок, про которые мы знаем, нуль это или единица. По этим образцам мы хотим сформулировать какое-то разумное разделяющее правило. Что означает в этом контексте слово «разумное»? Если мы просто перечислим все образцы того или другого типа, получится вполне действующее правило— действующее до тех пор, пока нам не принесут новый образец, — но оно будет слишком длинным. Естественно считать, что разумное правило должно иметь короткое описание, то есть по возможности меньшую колмогоровскую сложность.

Глава 8

Дополнительные задачи

Задача 107. Сколькими способами можно упорядочить по весу 6 различных камешков?

Задача 108. Сколькими способами можно упорядочить 6 различных камешков так, чтобы первый камешек был больше второго?

Задача 109. Сколькими способами можно упорядочить 7 различных камешков так, чтобы первый камешек был больше второго, а второй больше третьего?

Задача 110. Сколькими способами можно упорядочить 7 различных камешков так, чтобы первый камешек был больше второго, а третий больше четвертого?

Задача 111. Дано шесть монет разного веса и известно, что первая монета легче второй, а вторая легче третьей. Имеются весы, позволяющие сравнить по весу любые две монеты. Доказать, что для того, чтобы упорядочить по весу все монеты, необходимо 7 взвешиваний.

Задача 112. Доказать, что, более того, в предыдущей задаче необходимо 8 взвешиваний и выполнить упорядочение за 8 взвешиваний.

Задача 113. Дано шесть монет разного веса и известно, что первая монета легче второй, третья легче четвертой, а четвертая легче пятой. Имеются весы, позволяющие сравнить по весу любые две монеты. Доказать, что для того, чтобы упорядочить по весу все монеты, необходимо 6 взвешиваний.

Задача 114. Доказать, что, более того, в предыдущей задаче необходимо 7 взвешиваний и выполнить упорядочение за 7 взвешиваний.

Задача 115. В клетках шахматной доски написали в каком-то порядке числа от 1 до 64, каждое по одному разу. Про любое множество клеток доски мы можем спросить, какие числа на них стоят, и нам выдают полный список. За какое наименьшее количество вопросов можно понять, где какие числа стоят?

Задача 116. Имеются две карточки, на каждой из которых написано целое число от 1 до 9. Алёне сообщена сумма этих чисел, а Боре — разность первого и второго числа. Что написано на карточках, если в этой ситуации Боря знает число на первой карточке? При каких числах на карточках Алёна знает, что Боря знает число на первой карточке?

Задача 117. Имеются две карточки, на каждой из которых написано целое число от 1 до 9. Алёне сообщена сумма этих чисел. При каких числах на карточках Алёне необходимо и достаточно задать два вопроса с ответами ДА/НЕТ, чтобы узнать оба числа?

Задача 118. У Алисы имеется последовательность x из n битов, а у Боба последовательность y из n битов. Они хотят вычислить $f(x,y) = \sum_{i=1}^{n}(x_i + y_i) \pmod 2$. Докажите, что для этого достаточно передать 2 бита, то есть коммуникационная сложность функции f не больше 2.

Задача 119. Докажите, что коммуникационная сложность функции f из предыдущей задачи в точности равна 2.

Задача 120. У Алисы имеется последовательность x из n битов, а у Боба последовательность y из n битов. Они хотят вычислить $f(x,y) = \sum_{i=1}^{n}(x_i + y_i)$. Докажите, что для этого достаточно передать $2\lceil \log_2 n \rceil$ бит.

Задача 121. Докажите, что коммуникационная сложность функции f из предыдущей задачи не меньше $\lceil \log_2 n \rceil$.

Задача 122. Пусть вероятности букв a,b,c,d,e,f,g равны, соответственно, 0.05, 0.1, 0.15, 0.15, 0.15, 0.2, 0.2. Найдите среднюю длину оптимального префиксного хода (построив код Хаффмана) и сравните ее с энтропией Шеннона.

Задача 123. Пусть вероятности букв a,b,c,d,e,f,g равны, соответственно, 0.05, 0.1, 0.1, 0.15, 0.2, 0.2, 0.2. Найдите среднюю длину оптимального префиксного хода (построив код Хаффмана) и сравните ее с энтропией Шеннона.

Задача 124. Пусть случайная величина α имеет распределение $p(0) = 1/9$, $p(1) = 1/6$, $p(2) = 1/6$, $p(3) = 5/9$. Положим $\beta = \alpha \pmod 2$. Найти $H(\alpha)$, $H(\beta)$, $I(\beta : \alpha)$, $H(\alpha|\beta)$, $H(\beta|\alpha)$, $H(\alpha, \beta)$.

Задача 125. Веса 2 монеток выбираются случайно и независимо среди чисел $1, \ldots, 4$. Какова энтропия Шеннона случайной величины, равной результату сравнения на чашечных весах весов первой и второй монетки? Какова энтропия случайной величины, равной разности весов первой и второй монетки? Какова взаимная информация этих случайных величин?

Задача 126. Доказать, что функция $I(\alpha : \beta) - I(\alpha : \beta|\gamma)$ является симметрической функцией случайных величин α, β, γ.

Задача 127. Алена выбирает число $x = 1, \ldots, n$ случайным образом так, что вероятность выбора числа i пропорциональна $1/i^2$ (коэффициент пропорциональности подобран так, чтобы сумма всех вероятностей была равна 1). Докажите, что Боря может, задав Алене в среднем $O(1)$ вопросов с ответами да/нет, узнать выбранное число.

Задача 128. Алена выбирает число $x = 1, \ldots, n$ случайным образом так, что вероятность выбора числа i пропорциональна $1/i$ (коэффициент пропорциональности подобран так, чтобы сумма всех вероятностей была равна 1). Докажите, что Боря может, задав Алене в среднем $(\log_2 n)/2 + o(\log n)$ вопросов с ответами да/нет, узнать выбранное число. Докажите, что любой алгоритм Бори в среднем задает не менее $(\log_2 n)/2 + o(\log n)$ вопросов.

Задача 129. Алена выбирает число $x = 1, \ldots, n$ случайным образом так, что вероятность выбора числа i пропорциональна $1/(i \ln i)$ (коэффициент пропорциональности подобран так, чтобы сумма всех вероятностей была равна 1). Докажите, что Боря может, задав Алене в среднем $o(\log n)$ вопросов с ответами да/нет, узнать выбранное число.

Задача 130. Пусть случайная величина α имеет распределение $1/3, 2/3$, а случайная величина β имеет распределение $1/2, 1/2$. В каких пределах могут изменяться $I(\beta : \alpha)$, $H(\alpha|\beta)$, $H(\beta|\alpha)$, $H(\alpha, \beta)$?

Задача 131. Алисе сообщено значение случайной величины α, а Бобу — значение некоторой функции f от α. Придумайте алгоритм, который позволит Алисе сообщить Бобу значение α, передав в среднем не более $H(\alpha|f(\alpha)) + 1$ битов.

Задача 132. Алисе сообщено значение случайной величины α, а Бобу — значение некоторой функции f от α. Докажите, что не существует алгоритма, который позволит Алисе сообщить Бобу значение α, передав в среднем менее $H(\alpha|f(\alpha))$ битов.

Задача 133. Доказать, что для любых случайных величин $\alpha_1, \ldots, \alpha_n$ выполнено следующее неравенство: $(n-1)H(\alpha_1, \ldots, \alpha_n)$ не превосходит суммы энтропий n кортежей, получающихся из кортежа $\langle \alpha_1, \ldots, \alpha_n \rangle$ вычеркиванием его компонент.

Задача 134. Доказать, что для любых случайных величин α, β, γ выполнено неравенство:

$$H(\alpha) \leqslant H(\alpha|\beta) + H(\alpha|\gamma) + I(\beta : \gamma).$$

Задача 135. Построить совместно распределённые случайные величины ξ, η, β, γ, для которых не выполнено неравенство

$$I(\xi : \eta) \leqslant I(\xi : \eta|\beta) + I(\xi : \eta|\gamma) + I(\beta : \gamma).$$

Задача 136. Пусть $\alpha_1, \alpha_2, \alpha_3, \ldots$ произвольные совместно распределенные случайные величины с одним и тем же конечным множеством исходов. Верно ли, что они образуют марковскую цепь тогда и только тогда, когда $I(\alpha_{i-1} : \alpha_{i+1}|\alpha_i) = 0$ для всех $i > 1$. (Последовательность случайных величин $\alpha_1, \alpha_2, \alpha_3, \ldots$ образует марковскую цепь, если для некоторой марковской цепи, α_i есть состояние этой цепи в момент i).

Задача 137. Алена выбирает число $x = 1, \ldots, n$ случайным образом по известному Боре распределению вероятностей. Известно, что Боря может, задав Алене в среднем q вопросов с ответами да/нет, узнать выбранное число. Докажите, что тогда некоторое число x выбирается Аленой с вероятностью не меньше 2^{-q}.

Задача 138. Приведите пример совместно распределенных случайных величин α, β, γ для которых неравенство

$$H(\alpha, \gamma) + H(\beta, \gamma) \leqslant H(\alpha, \beta, \gamma) + H(\gamma)$$

неверно.

Задача 139. Пусть дано линейное неравенство от $H(\alpha)$, $H(\beta)$, $H(\gamma)$, $H(\alpha, \gamma)$, $H(\beta, \gamma)$, которое истинно для всех троек α, β, γ таких, что $I(\alpha : \beta|\gamma) = 0$. Докажите, что тогда неравенство верно и для всех вообще троек совместно распределённых случайных величин α, β, γ.

Задача 140. Алёна выбирает число $x = 1, \ldots, n$ случайным образом. Задача Бори — задавая Алёне вопросы с ответами да/нет, узнать выбранное число. Боре разрешается ошибаться с вероятностью не более $1/2$. Постройте алгоритм Бори, который задает в худшем случае не более $\log n - 1$ вопросов (n — степень двойки). Докажите, любой алгоритм Бори задает в худшем случае не менее $\log n - 1$ вопросов.

Задача 141. Игрок в казино может поставить на любой из двух исходов (0,1) бросания симметричной монетки любую ставку в пределах его текущего капитала. Сделанная ставка удваивается в случае выигрыша и теряется иначе. Существует ли стратегия игрока, которая гарантированно утраивает начальный капитал после шести игр при условии, что в этих шести играх выпало чётное количество 1?

Задача 142. Игрок в казино может поставить на любой из двух исходов (0,1) бросания монетки любую ставку в пределах его текущего капитала. Сделанная ставка удваивается в случае выигрыша и теряется иначе. Пусть игроку стало известно, что в первых шести бросаниях будет ровно 3 единицы. Придумайте стратегию игрока, гарантирующую возрастание начального капитала в 3.2 раза. Какую ставку эта стратегия делает после того, как в первом бросании выпал 0? Какую ставку эта стратегия делает после того, как в первых двух бросаниях выпали нули? Докажите, что число 3.2 в этой задаче нельзя увеличить.

Задача 143. Игрок в казино может ставить на любой из шести исходов бросания кубика в пределах своего текущего капитала (можно ставить одновременно на несколько исходов различные ставки). Различные бросания кубика независимы и все грани равновероятны. При выпадении исхода i все сделанные ставки отбираются, но выплачивается выигрыш, в $_i$ раз больший ставки, сделанной на этот исход, где $c_1 = c_2 = c_3 = 4$, $c_4 = 8$, $c_5 = c_6 = 16$. Придумайте стратегию игрока, которая гарантирует с вероятностью не менее $7/8$ увеличение начального капитала в полтора раза в течение 3 игр.

Задача 144. Игрок в казино может ставить на любой из шести исходов бросания кубика в пределах своего текущего капитала (можно ставить сразу на несколько исходов различные ставки). Различные бросания кубика независимы и вероятности выпадения исходов $1, \ldots, 6$ равны $1/2, 1/4, 1/16, 1/16, 1/16, 1/16$ (они известны игроку). При выпадении любого исхода i все сделанные ставки отбираются, но выплачивается ставка, сделанная на этот исход, умноженная на 6. Придумайте стратегию

игрока, которая гарантирует с вероятностью не менее $7/8$ увеличение капитала вдвое в течение трех игр.

Задача 145. (а) Найти пропускную способность следующего двоичного канала с шумом. Если входная буква равна 0, то на выходе канала появляются 0 и 1 с вероятностями $1/3$ и $2/3$, соответственно. Если входная буква равна 1, то на выходе канала появляются 0 и 1 с вероятностями $1/6$ и $5/6$, соответственно. (б) Найти наименьшую возможную вероятность ошибки при передаче по этому каналу одного бита при кодировании его двумя буквами. (Вероятность ошибки при данной схеме кодирования/декодирования определяется, как максимум по всем исходным сообщениям вероятности неправильного декодирования данного сообщения.)

Задача 146. На множестве слов длины 4 в двухбуквенном алфавите 0,1 задана бернуллиева мера с вероятностями 0 и 1 равными $1/3$ и $2/3$, соответственно. Какова минимально возможная вероятность ошибки при кодировании элементов этого множества трех-битовыми последовательностями. (Вероятность ошибки при данной схеме кодирования/декодирования определяется, как максимум по всем исходным сообщениям вероятности неправильного декодирования данного сообщения.)

Задача 147. Постройте кодовые таблицы кодов Хаффмана, Шеннона–Фано и арифметического кода для последовательности `ebdecaefbcffcbcaddbaacbccedcffbb`.

Задача 148. Доказать, что колмогоровская сложность слова, составленного из n знаков после запятой числа $\sqrt{2}$, равна $KS(n) + O(1)$.

Задача 149. Зафиксируем некоторое вычислимое однозначное кодирование конечных множеств (двоичных) слов (двоичными) словами. Колмогоровской сложностью конечного множества слов назовем колмогоровскую сложность его кода. Положим $A_n = \{x : l(x) = n,\ KS(x) < n\}$. Докажите, что $KS(A_n) \leqslant n + O(\log n)$.

Задача 150. Зафиксируем некоторый оптимальный декомпрессор D. Для каждого слова x будем рассматривать кратчайшие описания x^* слова x (слово p называется описанием x, если $D(p) = x$). Докажите, что $KS(x^*) = KS(x, KS(x))$ для некоторого такого x^*.

Задача 151. Доказать, что для в любой строке длины n из нулей и единиц можно так изменить один бит, что колмогоровская сложность полученной строки станет не меньше $\log_2 n$.

Задача 152. Известно, что колмогоровская сложность неориентированного графа (без петель) с вершинами $1, 2, \ldots, n$ не меньше $n(n-1)/2$. (а) Докажите, что отсюда следует, что граф связен, если n достаточно велико. (б) Докажите, что отсюда следует, что граф содержит цикл, если n достаточно велико. (в) Докажите, что отсюда следует, что входная и выходная степени любой вершины не меньше εn для некоторого $\varepsilon > 0$ и всех достаточно больших n.

Задача 153. Известно, что сложность строки x длины n из нулей и единиц не меньше n. (а) Докажите, что отсюда следует, что x содержит не меньше, чем $n/2 - O(\sqrt{n})$ нулей. (б) Докажите, что для всех достаточно больших n в x не встречается $5 \log_2 n$ нулей подряд. (в) Докажите, что для всех достаточно больших n в строке x встречается $(\log n)/2$ нулей подряд. (г) Докажите, что $KS(x|n) \geqslant n - O(1)$. (д) Сотрём в x все биты с нечётными номерами и обозначим полученное слово через x'. Докажите, что $KS(x') \geqslant n/2 - O(1)$.

Задача 154. Зафиксируем некоторый оптимальный декомпрессор D. Для каждого n рассмотрим самое «долгоиграющее» описание длины не более n, то есть, такое p_n, для которого $D(p_n)$ определено и время вычисления $D(p_n)$ максимально (среди всех $p \in \mathrm{Dom}D$ длины не более n). Докажите, что $KS(p_n) \geqslant n - O(1)$.

Задача 155. Зафиксируем некоторый оптимальный декомпрессор D. Для каждого n рассмотрим максимальное время работы D на входах p длины не более n, на которых D останавливается. Обозначим полученное число через t_n. (а) Докажите, что функция $n \mapsto t_n$ растет быстрее любой вычислимой функции. (б) Докажите, что $KS(t_n) = n + O(\log n)$.

Задача 156. Пусть $m(n)$ обозначает минимальную колмогоровская сложность натуральных чисел в интервале $[n, +\infty)$. Докажите, что функция $m(n)$ невычислима.

Задача 157. Известно, что всюду определенная вычислимая функция f по любому натуральному n дает некоторое такое натуральное k, что колмогоровская сложность всех натуральных чисел $n, n+1, \ldots$ больше k. Докажите, что f ограничена сверху.

Задача 158. Известно, что колмогоровская сложность целого числа x в интервале от 1 до n не меньше $\lfloor \log_2 n \rfloor$. Докажите, что число x составное, если n достаточно велико.

Задача 159. Докажите, что не существует константы c, для которой для всех x, y выполнено $KS(x, y) \leqslant KS(x) + KS(y) + \log_2 KS(y) + c$.

Задача 160. Докажите, что количество слов длины n, для которых $KS(x|n) < KS(x) - c$ меньше $2^{n-c+O(1)}$.

Задача 161. Докажите неравенство $KS(x, y|x, z) \leqslant KS(y|z) + O(1)$. Докажите, что это неравенство является строгим (то, есть обратное неравенство $KS(x, y|x, z) \geqslant KS(y|z) - O(1)$ не является верным для всех x, y, z).

Задача 162. Пусть бесконечная последовательность $x_1, x_2, \ldots$ обладает следующим свойством: для всех n выполнено $KS(x_1 \ldots x_n|n) \leqslant c$. (а) Докажите, что количество таких последовательностей не превосходит $O(2^c)$. (б) Докажите, что то же самое верно и для последовательностей, удовлетворяющих более слабому неравенству $KS(x_1 \ldots x_n) \leqslant \log n + c$.

Задача 163. Докажите, что для любых x, y, n найдется слово z длины n, для которого $KS(z|x) \geqslant n - 1$ и $KS(z|y) \geqslant n - 1$.

Литература

[1] Верещагин Н. К., Шень А. Лекции по математической логике и теории алгоритмов. Часть 3. Вычислимые функции. М.:МЦНМО, 2008. (ftp://ftp.mccme.ru/users/shen/logic/comput)

[2] Успенский В.А., Верещагин Н. К., Шень А. Колмогоровская сложность. (Рукопись незаконченной книги.) (ftp://ftp.mccme.ru/users/shen/kolmbook)

[3] И. Чисар, Я. Кёрнер. Теория информации. Москва, Изд-во «Мир», 1985. 397 с.

[4] А.М. Яглом, И.М. Яглом. Вероятность и информация. Москва, Изд-во «Наука», 1973. 512 с.

[5] Nicolo Cesa-Bianchi and Gabor Lugosi. Prediction, Learning, and Games. Cambridge University Press, Cambridge, England, 2006.

[6] M. Cover, J.A. Thomas. "Elements of information theory John Wiley & Sons 1991.

[7] Galin L. Jones. On the Markov chain central limit theorem. Probab. Surveys 1 (2004) 299-320.

[8] E. Kushilevitz, N. Nisan. Communication Complexity. Camridge UP. 1997.

[9] Li M., Vitányi P., *An Introduction to Kolmogorov Complexity and Its Applications*, Second Edition, Springer, 1997. 638 pp.

[10] Solomonoff R. J., A formal theory of inductive inference, part 1, part 2. *Information and Control* (now *Information and Computation*), v. 7 (1964), p. 1–22, p. 224–254.

[11] V.Vovk. A game of prediction with expert advice. Journal of Computer and System Sciences 56, 153-173 (1998).