Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования

Поволжский государственный университет телекоммуникаций и
информатики

кафедра ТОРС

Задание и методические указания к лабораторной работе
по дисциплинам «**Общая теория связи**»,
«**Теория информации и информационных систем**»,

для студентов 2 курса направлений 210700, 200700
дневной формы обучения

«**Кодирование дискретных источников**»

Составители: д.т.н., проф. Николаев Б. И.
к.т.н. Борисенков А. В.
к.т.н. Чингаева А. М.

Самара, 2014 г.

УДК 621.391

Задание и методические указания к лабораторной работе по дисциплинам «**Общая теория связи**», «**Теория информации и информационных систем**» для студентов 2 курса направлений 210700, 200700 дневной формы обучения «**Кодирование дискретных источников**» / сост. Б. И. Николаев, А. В. Борисенков, А. М. Чингаева — Самара: ПГУТИ, 2014 — 20 с.

Методическая разработка содержит краткую теорию и подробные указания к выполнению лабораторной работы, посвящённой кодированию дискретных источников.

1 Краткие теоретические сведения

Кодированием в теории информации называют преобразование одного алфавита символов в другой.

Различают три основных вида кодов:

1. примитивные — используются для согласования алфавита источника сообщений с алфавитом канала;

2. экономные — для «сжатия» информации (уменьшения или полного устранения избыточности, содержащейся в сообщении);

3. помехоустойчивые — для обнаружения или исправления ошибок, возникающих в канале связи.

Операция кодирования **взаимно однозначна**, т.е. при отсутствии ошибок в канале связи потери информации при кодировании и декодировании отсутствуют.

Примитивные и экономные коды применяются для кодирования источника, помехоустойчивые — для канального кодирования.

1.1 Примитивное кодирование

При примитивном кодировании символы алфавита источника представляются набором чисел от 0 до $(K-1)$ (K — объём алфавита источника), т.е. нумеруются. При этом номер представляется в виде n-разрядного m-позиционного кода, где

$$m^n = K \quad (n = \log_m K).$$

Примитивные коды являются **равномерными**, т.е. каждый символ исходного алфавита содержит фиксированное количество кодовых позиций (кодовая комбинация имеет фиксированную длину).

Примитивный код широко используется в компьютерной технике для представления символов и знаков текста («набор символов», «кодировка»). Примеры:

1. ASCII (American Standard Code for Information Interchange) — американская стандартная кодировочная таблица: 7-битная таблица для представления печатных символов и некоторых специальных кодов.

2. Национальные 8-битные варианты ASCII — для кириллицы это Windows-1251 (CP1251), КОИ-8, IBM code page 866 (CP866).

1.2 Экономное кодирование

Экономное кодирование, называемое также **сжатием без потерь**, используется для уменьшения времени передачи информации или объёма памяти, необходимой для её хранения. Суть экономного кодирования заключается в том, что избыточность на выходе кодера уменьшается по сравнению с избыточностью на его входе (избыточностью первичного источника). На практике это означает, что объём данных, выраженный, например, в битах, на выходе кодера уменьшается по сравнению с объёмом данных на его входе. Отношение выходного объёма к входному, выраженное в процентах, называют **коэффициентом сжатия**:

$$r = \frac{N_{\text{вых}}}{N_{\text{вх}}} \cdot 100\%.$$

Эффективность кодирования в теории определяется как

$$\eta_{\text{и}} = (1 - \rho_{\text{и}}) \cdot 100\%,$$

где $\rho_{\text{и}}$ — избыточность источника. Для 32-символьной модели русского языка $\rho_{\text{и}} \approx 0{,}73 = 73\%$ и $\eta_{\text{и}} \approx 27\%$.

Коэффициент сжатия $r \geqslant \eta_{\text{и}}$.

Экономное кодирование широко используется в компьютерной технике для сжатия данных (в архиваторах) и в цифровых системах передачи информации.

Существует два основных вида экономного кодирования:

1. Энтропийное кодирование — символы исходного алфавита кодируются кодовыми комбинациями переменной длины, таким образом, что энтропия выходных символов кодера оказывается максимально возможной (приближается к максимально возможной).

2. Словарное кодирование — последовательности символов исходного текста различной длины заменяются соответствующими индексами из словаря.

2 Энтропийное кодирование: коды Шеннона-Фано и Хаффмана

Если источник выдаёт сообщения a_i из алфавита K с различными вероятностями $P(a_i)$, оптимальным будет такое кодирование, при котором символам с большей вероятностью ставятся в соответствие кодовые слова меньшей длины n_i. При этом средняя длина кодового слова $\bar{n} = \sum_{i=1}^{K} n_i P(a_i)$ будет минимальной.

Код, комбинации которого имеют различную длину, называют **неравномерным**. Для неравномерного кода встаёт вопрос однозначности декодирования. Однозначность может быть обеспечена при помощи специального символа-разделителя (как в коде Морзе), однако, это не оптимально. Без разделителя можно обойтись, если ни одно кодовое слово не является началом другого. Коды, обладающие таким свойством, называют **префиксными**.

Минимально возможное среднее количество кодовых символов на один символ источника определяется **пределом Шеннона**:

$$\bar{n}_{\min} = \frac{H(A)}{\log_2 m},$$

m — позиционность кода.

Одним из первых экономных кодов, использующих принцип энтропийного кодирования, является код Шеннона-Фано.

Код Шеннона-Фано строится следующим образом:

1. Символы алфавита располагаются в порядке убывания их вероятностей.

2. Полученное множество делится на два подмножества с максимально близкими суммарными вероятностями символов.

3. Первому подмножеству присваивается кодовый символ «0», второму — «1».

4. Каждое подмножество вновь делится на две половины. Процесс повторяется до тех пор, пока в каждом подмножестве не останется по единственному символу.

Процедура кодирования кодом Шеннона-Фано лучше всего описывается с помощью **кодового дерева**. Построение дерева идёт от «корня» (совокупности всех символов алфавита) к «листьям» (отдельным символам).

Пример построения кодового дерева кода Шеннона-Фано для источника с алфавитом «А» (P(А) = 0,35), «Б» (P(Б) = 0,17), «В» (P(В) = 0,17), «Г» (P(Г) = 0,16), «Д» (P(Д) = 0,15) показан на рис. 1.

Для получения кода символа необходимо пройти от «корня» к соответствующему «листу»: «А» — «00», «Б» — «01», «В» — «10», «Г» — «110», «Д» — «111».

Код Шеннона-Фано не является оптимальным, хотя и позволяет получить оптимальное разбиение для некоторых частных случаев распределения вероятностей, и в настоящее время практически не используется.

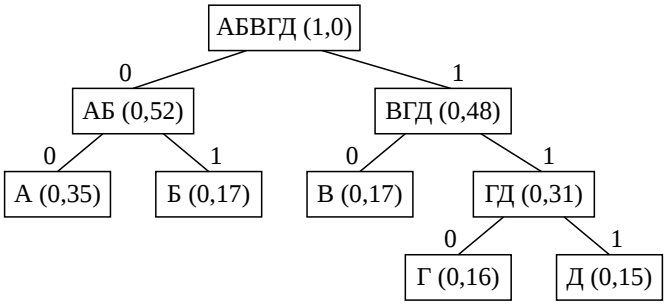

Рис. 1. Кодовое дерево кода Шеннона-Фано
(в скобках указаны суммарные вероятности — «веса» узлов дерева)

В качестве основного метода энтропийного кодирования обычно применяется код Хаффмана.

Отличие **кода Хаффмана** от кода Шеннона-Фано заключается в том, что построение кодового дерева идёт не от «корня» к «листьям», а от «листьев» к «корню», т.е. в обратном порядке:

1. Символы исходного алфавита образуют список свободных узлов («листьев» дерева), вес каждого из которых равен вероятности символа.

2. Два свободных узла дерева с наименьшими весами объединяются в новый узел («родитель») с весом, равным сумме весов объединённых узлов.

3. «Родитель» добавляется в список свободных узлов, а его «потомки» удаляются из списка.

4. Одной ветви, отходящей от «родителя», ставится в соответствие символ «0», другой — символ «1».

5. Процесс продолжается до тех пор, пока в списке не останется единственный узел, который будет являться корнем дерева.

Пример кодового дерева кода Хаффмана показан на рис. 2. Источник и вероятности те же, что и в примере на рис. 1.

Коды символов: «А» — «0», «Б» — «100», «В» — «101», «Г» — «110», «Д» — «111».

Код, построенный по методу Хаффмана, является оптимальным.

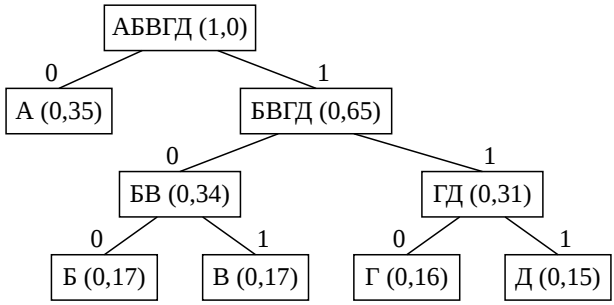

Рис. 2. Кодовое дерево кода Хаффмана

3 Словарное кодирование: семейство кодов Лемпела-Зива

При словарном кодировании исходные данные разбиваются на «слова» — уникальные цепочки входных символов различной длины, — которые затем заменяются соответствующими кодовыми комбинациями фиксированной длины, представляющими собой индексы (номера) входной цепочки в «словаре» кодера.

Словарное кодирование является одним из самых распространённых методов сжатия в современных архиваторах.

Существует множество подходов к реализации словарного кодирования, различающихся способом построения словаря. Однако, независимо от способа, словарь в большинстве случаев является динамическим, т.е. пополняется в процессе кодирования (и декодирования).

Для словарных кодов характерны большие вычислительные затраты при кодировании, при этом затраты на декодирование минимальны.

Основной характеристикой словарных кодов, влияющей на эффективность сжатия, является размер словаря D: чем он больше, тем выше эффективность (но и выше вычислительные затраты).

Базой всех современных словарных кодов являются коды Лемпела-Зива LZ77 и LZ78, опубликованные в 1977 и 1978 гг. соответственно в работах Абрахама Лемпела и Якоба Зива. Оба алгоритма используют механизм кодирования совпадений и отличаются подходом к построению словаря.

Алгоритм LZ77 в качестве неявного словаря использует скользящее окно, содержащее D уже обработанных входных символов. Величина D является эквивалентом размера словаря в других алгоритмах.

Кодовое слово в алгоритме LZ77 состоит из трёх элементов: позиция,

длина, символ. Алгоритм работает следующим образом:

1. В «словаре» (среди D предшествующих символов) ищется наиболее длинная цепочка совпадений.

2. Найденная цепочка кодируется парой «позиция-длина», где «позиция» означает количество символов, на которое нужно вернуться в потоке данных, а «длина» — количество символов, которое нужно извлечь с указанной позиции.

3. После этого на выход выдаётся некодированный символ, следующий сразу за найденной цепочкой.

4. Процедура повторяется до тех пор, пока не будет закодирован весь объём данных.

Рассмотрим работу алгоритма на примере. Закодируем текст «LZ77LZ77.».

1. На вход поступает символ «L». Поскольку словарь ещё пуст, на выход выдаётся пара чисел (0, 0), которая означает «вернуться на 0 символов назад и извлечь 0 символов», и некодированный символ «L».

2. На вход поступает символ «Z». «Словарь» (буфер) содержит лишь один предыдущий символ «L». Символ «Z» в буфере отсутствует, следовательно, на выход снова выдаётся пара (0, 0) и сам символ «Z».

3. На вход поступает символ «7». В буфере содержатся символы «LZ». Символ «7» в буфере отсутствует, на выход выдаётся (0, 0, «7»).

4. На вход поступает символ «7». В буфере содержатся символы «LZ7». Символ «7» найден в буфере на позиции 1 назад от текущей. На выход ничего не выдаётся, продолжается поиск более длинного совпадения: пары, начинающейся с символа «7».

5. На вход поступает символ «L». В буфере содержатся символы «LZ77». Пара «7L» в буфере отсутствует, значит, наиболее длинное совпадение найдено: это «7». На выход выдаётся (1, 1), что означает «вернуться на 1 шаг назад и извлечь 1 символ», и символ «L».

6. На вход поступает символ «Z». В буфере содержатся символы «LZ77L». Символ «Z» найден в буфере на позиции 4 назад от текущей. Продолжается поиск более длинного совпадения: пары, начинающейся с символа «Z».

7. На вход поступает символ «7». В буфере содержатся символы «LZ77LZ». Пара «Z7» найдена в буфере, продолжается поиск более длинного совпадения: тройки символов, начинающейся с «Z7».

8. На вход поступает символ «7». В буфере содержатся символы «LZ77LZ7». Тройка «Z77» найдена в буфере, продолжается поиск более длинного совпадения: четвёрки символов, начинающейся с «Z77».

9. На вход поступает символ «.». В буфере содержатся символы «LZ77LZ77». Четвёрка «Z77.» в буфере отсутствует, значит, наиболее длинное совпадение найдено: это «Z77». На выход выдаётся (4, 3), что означает «вернуться на 4 шага назад и извлечь 3 символа», и символ «.».

Сжатый текст: (0, 0, «L»), (0, 0, «Z»), (0, 0, «7»), (1, 1, «L»), (4, 3, «.»).

Пример проиллюстрирован графически на рис. 3.

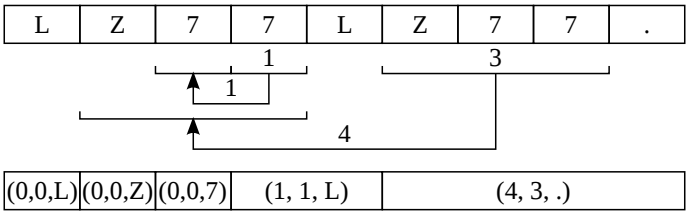

Рис. 3. Иллюстрация алгоритма LZ77

Допустим, символы исходного текста закодированы 8-битным примитивным кодом, размер словаря $D = 8$, а максимально допустимая длина цепочки повторений $L = 4$. Тогда для кодирования 9 символов исходного текста требуется $8 \cdot 9 = 72$ бита, а для сжатого текста $(\log_2 8 + \log_2 4 + 8) \cdot 5 = 65$ бит. Коэффициент сжатия $r = 90\%$.

Декодирование осуществляется просто:

1. (0, 0, «L») — на выход выдаётся символ «L». «Словарь» (буфер) содержит символ «L».

2. (0, 0, «Z») — на выход выдаётся символ «Z». Буфер содержит символы «LZ».

3. (0, 0, «7») — на выход выдаётся символ «7». Буфер содержит символы «LZ7».

4. (1, 1, «L») — декодер возвращается на 1 шаг назад в буфере и выдаёт на выход 1 символ — это «7». Буфер содержит символы «LZ77». Затем на выход выдаётся символ «L». Буфер содержит символы «LZ77L».

5. (4, 3, «.») — декодер возвращается на 4 шага назад в буфере и выдаёт на выход 3 символа — это «Z77». Буфер содержит символы «LZ77LZ77». Затем на выход выдаётся символ «.».

Восстановленный текст: «LZ77LZ77.».

Алгоритм LZ78 использует словарь в явном виде: таблицу нумерованных записей, содержащих максимально короткие ранее не встречавшиеся цепочки символов. Первый символ словаря (с номером 0) — «пустой» символ. Кодовое слово в алгоритме LZ78 состоит из двух элементов: индекс и символ. Словарь здесь также является динамическим и строится в процессе сжатия. Алгоритм работает следующим образом:

1. Входной поток разбивается на максимально короткие ранее не встречавшиеся цепочки.

2. Полученные цепочки заносятся в словарь и нумеруются.

3. Префикс (все символы, кроме последнего) каждой цепочки ищется в словаре и заменяется соответствующим индексом (порядковым номером), последний символ цепочки выдаётся на выход без изменений.

Рассмотрим работу алгоритма на том же примере, что и выше: закодируем текст «LZ77LZ77.».

1. Разобьём текст на максимально короткие ранее не встречавшиеся цепочки: «L», «Z», «7», «7L», «Z7», «7.».

2. Словарь будет содержать следующие записи: 0 — «пусто», 1 — «L», 2 — «Z», 3 — «7», 4 — «7L», 5 — «Z7», 6 — «7.».

3. Закодируем цепочки: цепочка «L» соответствует паре («пусто», «L»), на выход выдаётся (0, «L»).

4. Цепочка «Z» соответствует паре («пусто», «Z»), на выход выдаётся (0, «Z»).

5. Цепочка «7» соответствует паре («пусто», «7»), на выход выдаётся (0, «7»).

6. Цепочка «7L» соответствует паре («7», «L»), на выход выдаётся (3, «L»).

7. Цепочка «Z7» соответствует паре («Z», «7»), на выход выдаётся (2, «7»).

8. Цепочка «7.» соответствует паре («7», «.»), на выход выдаётся (3, «.»).

Сжатый текст: (0, «L»), (0, «Z»), (0, «7»), (3, «L»), (2, «7»), (3, «.»). Если размер словаря $D = 8$, то для представления сжатого текста потребуется $(\log_2 8 + 8) \cdot 6 = 66$ бит. Коэффициент сжатия $r = 92\%$.

Декодирование:

1. (0, «L») — на выход выдаётся «пусто» (т.е. ничего) и «L», в словарь записывается символ «L» под номером 1.

2. (0, «Z») — на выход выдаётся «пусто» (т.е. ничего) и «Z», в словарь записывается символ «Z» под номером 2.

3. (0, «7») — на выход выдаётся «пусто» (т.е. ничего) и «7», в словарь записывается символ «7» под номером 3.

4. (3, «L») — на выход выдаётся «7» и «L», в словарь записывается пара «7L» под номером 4.

5. (2, «7») — на выход выдаётся «Z» и «7», в словарь записывается пара «Z7» под номером 5.

6. (3, «.») — на выход выдаётся «7» и «.», в словарь записывается пара «7.» под номером 6.

Восстановленный текст: «LZ77LZ77.».

Работа алгоритма LZ78 наглядно представляется в виде таблицы:

Кодирование

Вход (словарь)	Индекс	Выход
–	0	–
L	1	(0, «L»)
Z	2	(0, «Z»)
7	3	(0, «7»)
7L	4	(3, «L»)
Z7	5	(2, «7»)
7.	6	(3, «.»)

Декодирование

Вход	Выход (словарь)	Индекс
–	–	0
(0, «L»)	L	1
(0, «Z»)	Z	2
(0, «7»)	7	3
(3, «L»)	7L	4
(2, «7»)	Z7	5
(3, «.»)	7.	6

В 1978 г. Терри Уэлш предложил улучшенную реализацию алгоритма Лемпела-Зива LZ78, которая получила название «**алгоритм Лемпела-Зива-Уэлша (LZW)**».

Отличие LZW от LZ78 заключается в том, что словарь предварительно инициализируется всеми одиночными символами входного алфавита. Для 8-битных данных это будут 256 возможных значений 8-битного байта, для 32-символьной модели русского текста — 32 символа модифицированного алфавита. Во входном потоке ищется максимально короткая цепочка, отсутствующая в словаре. Найденная цепочка заносится в словарь, на выход выдаётся индекс префикса цепочки, а последний символ цепочки становится первым символом новой цепочки, которая далее будет искаться в словаре.

Рассмотрим работу алгоритма на том же примере, что и выше: закодируем текст «LZ77LZ77.». Размер словаря пусть также будет равен 8: $D = 8$.

1. Алфавит источника состоит из 4 символов: «L», «Z», «7», «.». Эти символы заносятся в словарь под номерами 0, 1, 2 и 3 соответственно.

2. На вход поступает символ «L». Такой символ в словаре уже есть. На выход ничего не выдаётся, продолжается поиск уже двухсимвольной цепочки, начинающейся с «L».

3. На вход поступает символ «Z». В словаре ищется пара «LZ». Такой пары нет. «LZ» заносится в словарь под номером 4, на выход выдаётся код префикса «L»: 0. Продолжается поиск двухсимвольной цепочки, начинающейся с «Z».

4. На вход поступает символ «7». В словаре ищется пара «Z7». Такой пары нет. «Z7» заносится в словарь под номером 5, на выход выдаётся код префикса «Z»: 1. Продолжается поиск двухсимвольной цепочки, начинающейся с «7».

5. На вход поступает символ «7». В словаре ищется пара «77». Такой пары нет. «77» заносится в словарь под номером 6, на выход выдаётся код префикса «7»: 2. Продолжается поиск двухсимвольной цепочки, начинающейся с «7».

6. На вход поступает символ «L». В словаре ищется пара «7L». Такой пары нет. «7L» заносится в словарь под номером 7, на выход выдаётся код префикса «7»: 2. Продолжается поиск двухсимвольной цепочки, начинающейся с «L».

7. На вход поступает символ «Z». В словаре ищется пара «LZ». Такая пара уже есть. На выход ничего не выдаётся, продолжается поиск уже трёхсимвольной цепочки, начинающейся с «LZ».

8. На вход поступает символ «7». В словаре ищется цепочка «LZ7». Такой цепочки нет. Достигнут предел размера словаря, поэтому в словарь ничего не заносится, на выход выдаётся код префикса «LZ»: 4. Продолжается поиск двухсимвольной цепочки, начинающейся с «7».

9. На вход поступает символ «7». В словаре ищется цепочка «77». Такая пара уже есть. На выход ничего не выдаётся, продолжается поиск уже трёхсимвольной цепочки, начинающейся с «77».

10. На вход поступает символ «.». В словаре ищется цепочка «77.». Такой цепочки нет. Достигнут предел размера словаря, поэтому в словарь ничего не заносится, на выход выдаётся код префикса «77»: 6. Продолжается поиск двухсимвольной цепочки, начинающейся с «.».

11. Достигнут конец текста, на выход выдаётся код последней искомой цепочки, в данном случае это один символ «.», его индекс в словаре — 3.

Сжатый текст: 0, 1, 2, 2, 4, 6, 3. Для представления сжатого текста потребуется $\log_2 8 \cdot 7 = 21$ бит. Строго говоря, алфавит нашего источника является нестандартным, поэтому его также придётся сохранить (передать), затратив на это $8 \cdot 4 = 32$ бита. В итоге сжатый текст займёт $21 + 32 = 53$ бита. Коэффициент сжатия $r = 74\%$.

В виде таблицы:

Текущая цепочка	Следующий символ	Словарь	Индекс	Выход
		L	0	
		Z	1	
		7	2	
		.	3	
L	Z	LZ	4	0
Z	7	Z7	5	1
7	7	77	6	2
7	L	7L	7	2
LZ	7	LZ7	–	4
77	.	77.	–	6
.	–	–	–	3

Если не переходить к нестандартному словарю, то для кодирования придётся использовать как минимум 9 бит (размер словаря $D = 512$ — первые 256 записей будут содержать все возможные 8-битные символы, оставшиеся 256 — кодируемые цепочки). Индексы символов и цепочек в словаре будут уже другими, но их количество не изменится. Т.о. сжатый текст займёт $9 \cdot 7 = 63$ бита. Коэффициент сжатия $r = 88\%$.

Декодирование:

1. В словарь заносятся символы источника «L», «Z», «7» и «.» под номерами 0, 1, 2 и 3 соответственно.

2. 0 — на выход выдаётся запись словаря с номером 0: «L». Формируется запись для словаря: предыдущая цепочка (пусто) и первый символ новой цепочки: «L». Т.к. «L» есть в словаре, ничего не происходит.

3. 1 — на выход выдаётся запись словаря с номером 1: «Z». Формируется запись для словаря: предыдущая цепочка («L») и первый символ новой цепочки: «Z». Цепочки «LZ» в словаре нет, ей присваивается номер 4.

4. 2 — на выход выдаётся запись словаря с номером 2: «7». Формируется запись для словаря: предыдущая цепочка («Z») и первый символ новой цепочки: «7». Цепочки «Z7» в словаре нет, ей присваивается номер 5.

5. 2 — на выход выдаётся запись словаря с номером 2: «7». Формируется запись для словаря: предыдущая цепочка («7») и первый символ новой цепочки: «7». Цепочки «77» в словаре нет, ей присваивается номер 6.

6. 4 — на выход выдаётся запись словаря с номером 4: «LZ». Формируется запись для словаря: предыдущая цепочка («7») и первый символ новой цепочки: «L». Цепочки «7L» в словаре нет, ей присваивается номер 7.

7. 6 — на выход выдаётся запись словаря с номером 6: «77». Формируется запись для словаря: предыдущая цепочка («LZ») и первый символ новой цепочки: «7». Цепочки «LZ7» в словаре нет, но т.к. достигнут предел размера словаря, в словарь ничего не заносится.

8. 3 — на выход выдаётся запись словаря с номером 3: «.».

Восстановленный текст: «LZ77LZ77.».

В виде таблицы:

Вход	Выход	Гипотеза	Словарь	Индекс
			L	0
			Z	1
			7	2
			.	3
0	L	L?	–	–
1	Z	Z?	LZ	4
2	7	7?	Z7	5
2	7	7?	77	6
4	LZ	LZ?	7L	7
6	77	77?	LZ7	–
3	.	.?	77.	–

Чем длиннее исходное сообщение, тем более длинные записи будут вноситься в словарь, тем больше будет эффективность сжатия словарным кодом.

Код LZ78 проигрывает LZW по эффективности и на практике не используется. LZW широко применяется для сжатия изображений (форматы GIF, TIFF) и данных. С ростом производительности ЭВМ в задачах сжатия произвольных данных на первый план вышел алгоритм LZ77 (и его вариации), который позволяет добиться наилучшего сжатия, но при этом требует больших вычислительных затрат и объёма памяти. Кроме того, в архиваторах код LZ77 используется в паре с энтропийным кодом (например, код Хаффмана в алгоритме Deflate, который используется архиваторами zip, gzip, 7z и др.) для кодирования более часто встречающихся индексов более короткими комбинациями, что позволяет дополнительно улучшить эффективность сжатия.

4 Домашнее задание

1. Выбрать в качестве источника сообщений источник с алфавитом «А», «Б», «В», «Г», «Д» и соответствующими вероятностями букв

$$P(\text{А}) = \frac{b+c+1}{N}, \quad P(\text{Б}) = \frac{a+b+1}{N}, \quad P(\text{В}) = \frac{a+c+1}{N},$$
$$P(\text{Г}) = \frac{c+1}{N}, \quad P(\text{Д}) = \frac{b+1}{N},$$

где abc — три последние цифры номера студенческого билета, $N = 2a+3b+3c+5$. Сумма рассчитанных вероятностей символов должна строго равняться единице!

2. Для ($c \mod 2$) $= 0$[1] выбрать код Шеннона-Фано, для 1 — код Хаффмана. Построить кодовое дерево выбранного энтропийного кода, записать кодовые комбинации для каждого символа исходного алфавита и вычислить среднюю длину кодового слова.

3. Выбрать в качестве исходного сообщения текст «ФИОФИОФИО.», где «ФИО» — инициалы студента.

4. Для ($c \mod 3 = 0$) выбрать код LZ77, для 1 — LZ78, для 2 — LZW. Закодировать исходное сообщение выбранным словарным кодом, при этом размер словаря и максимально допустимую длину цепочки повторений (для LZ77) подобрать так, чтобы все разряды кодового слова использовались наилучшим образом. Подсчитать длину кодового слова. Вычислить коэффициент сжатия при условии, что символы исходного сообщения кодированы примитивным двоичным 8-битным кодом.

5 Указания к выполнению работы

Перед началом работы ознакомьтесь с интерфейсом пользователя: нажмите `F1` или выберите в главном меню «`Помощь` → `Руководство пользователя`».

5.1 Цель работы

Целью работы является изучение методов кодирования дискретных источников на примере сжатия текстов на русском языке.

5.2 Исследование энтропийных методов кодирования

1. Откройте исходный текстовый файл на русском языке.

2. Изучите модифицированное и двоичное отображение текста.

3. Сожмите текст с помощью кода Шеннона-Фано. Запишите полученное значение коэффициента сжатия r.

4. Восстановите текст. Сравните восстановленный текст с исходным.

5. Изучите и скопируйте в отчёт кодовое дерево кода Шеннона-Фано.

[1] $x \mod y$ — операция взятия по модулю, эквивалентна взятию остатка от деления x на y

6. Изучите и скопируйте в отчёт содержимое словаря. Запишите полученное значение средней длины кодового слова $n' = \bar{n}$.

7. Перейдите к модифицированному тексту. Найдите энтропию модифицированного текста $H(A)$. Сравните полученное значение энтропии с $\bar{n}$ и пределом Шеннона. Сделайте выводы.

8. Перейдите к сжатому тексту. Найдите энтропию сжатого текста. Сделайте выводы.

9. Добавьте шум к сжатому тексту. Восстановите текст. Сравните восстановленный текст с исходным. Сделайте выводы.

10. Повторите п. 2–9 для кода Хаффмана.

11. Сравните результаты, полученные для кодов Шеннона-Фано и Хаффмана. Сделайте выводы.

5.3 Исследование словарных методов кодирования

1. Сожмите текст с помощью алгоритма Лемпела-Зива LZ78. Подберите размер словаря («Параметры») таким образом, чтобы получить наилучшее сжатие.

2. Запишите оптимальный размер словаря, длину кодового слова n и коэффициент сжатия r.

3. Изучите полученный сжатый текст.

4. Восстановите текст. Сравните восстановленный текст с исходным.

5. Изучите словарь, полученный в процессе сжатия. Запишите реальный получившийся размер словаря и длину наибольшего фрагмента. Сделайте выводы.

6. Добавьте шум к сжатому тексту. Восстановите текст. Сравните восстановленный текст с исходным. Сделайте выводы.

7. Повторите п. 1–6 для алгоритмов Лемпела-Зива-Уэлша LZW и Лемпела-Зива LZ77. Для LZ77 в п. 1 подбирать следует не только размер словаря, но и максимально допустимую длину фрагмента.

8. Сравните результаты, полученные для всех трёх алгоритмов LZ*. Сделайте выводы.

6 Содержание отчёта

1. Название и цель работы.
2. Выполненное домашнее задание.
3. Результаты, таблицы, кодовые деревья и выводы по всем пунктам работы.
4. Общий вывод по результатам лабораторной работы.

Примечание: общий вывод не должен быть перефразировкой целей работы, а должен содержать обобщение (но не копию!) всех выводов, сделанных по каждому пункту в отдельности.

7 Контрольные вопросы

1. Что в теории информации понимается под кодированием? Какие виды кодов вы знаете?
2. Опишите цели и суть примитивного кодирования. Приведите примеры.
3. Опишите основные принципы экономного кодирования (сжатия). Как оценивается эффективность сжатия?
4. Какие виды экономных кодов вы знаете?
5. В чём заключается суть энтропийного кодирования?
6. Какие коды называют префиксными? Какими свойствами обладают префиксные коды? Приведите примеры.
7. Как строится код Шеннона-Фано? В чём его преимущества? Недостатки?
8. Как строится код Хаффмана? В чём его преимущества? Недостатки?
9. В чём заключается суть словарного кодирования?
10. Опишите работу алгоритма Лемпела-Зива (LZ77). В чём его преимущества? Недостатки?
11. Опишите работу алгоритма Лемпела-Зива (LZ78). В чём его преимущества? Недостатки?

12. Опишите работу алгоритма Лемпела-Зива-Уэлша (LZW). В чём его преимущества? Недостатки?

13. Как влияет наличие ошибок в канале связи на качество декодирования экономных кодов?

Литература

1. Кловский Д. Д. Теория электрической связи. — М.: Радиотехника, 2009. — 648 с.

2. Теория электрической связи: учебник для вузов / А. Г. Зюко, Д. Д. Кловский, В. И. Коржик, М. В. Назаров; Под ред. Д. Д. Кловского. — М.: Радио и связь, 1998. — 432 с.

3. Шеннон К. Работы по теории информации и кибернетике. — М,: Издательство иностранной литературы, 1963.

4. Яглом И. М., Яглом А. М. Вероятность и информация. — М.: Наука, 1973.

Николаев Борис Иванович
Борисенков Алексей Владимирович
Чингаева Анна Михайловна

Задание и методические указания к лабораторной работе
по дисциплинам «**Общая теория связи**»,
«**Теория информации и информационных систем**»,

для студентов 2 курса направлений 210700, 200700
дневной формы обучения

«Кодирование дискретных источников»