Государственное учреждение высшего профессионального образования
«Белорусско-Российский университет»

Кафедра «Автоматизированные системы управления»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по дисциплине «Теория информации»
для специальности 22 02 00
Автоматизированные системы обработки информации и управления

Методические указания содержат описание, порядок выполнения и представления лабораторных работ к защите. Предназначены для студентов 2 курса специальности «Автоматизированные системы обработки информации и управления».

1. Энтропия и ее свойства
2. Количество информации
3. Простейшие алгоритмы сжатия информации
4. Арифметическое кодирование
5. Адаптивные алгоритмы сжатия
6. Адаптивное арифметическое кодирование
7. Подстановочные или словарно- ориентированные алгоритмы сжатия информации. Методы Лемпела-Зива
8. LZ- алгоритмы распаковки данных.
9. Групповое кодирование
10. Полиноминальные коды.
11. Коды Боуза- Чоудхури – Хоккенгема.
12. Циклические избыточные коды

Лабораторная работа № 1

Энтропия и ее свойства

1.Порядок выполнения работы

Ознакомится с методическими указаниями, изложенными в п.3.1 и 3.2;
Решить задачи (по указанию преподавателя);
Подготовить ответы на контрольные вопросы

2. Содержание отчета:

Тема и цель работы
Условия задач
Подробное решение задач
Выводы по работе

Часть 1

3.1 Общие сведения

Энтропия как мера неопределенности

Энтропия является мерой неопределенности опыта, в котором проявляются случайные события, и равна средней неопределенности всех возможных его исходов.

$$H(\alpha) = -\sum_{i=1}^{n} p(A_i) \cdot \log_2 p(A_i)$$

Единица измерения неопределенности называется *бит*

Свойства энтропии

1. $\boldsymbol{H = 0}$ только в двух случаях:
   - какая-либо из $p(A_j) = 1$; однако, при этом все остальные $p(A_i) = 0$ $(i \neq j)$, т.е. реализуется ситуация, когда один из исходов является *достоверным* (и общий итог опыта перестает быть случайным);
   - все $p(A_i) = 0$, т.е. никакие из рассматриваемых исходов опыта невозможны, поскольку нетрудно показать, что

$$\lim_{p \to 0} (p \cdot \log p) = 0$$

   Во всех остальных случаях, очевидно, что $\boldsymbol{H > 0}$.
2. Для двух *независимых* опытов α и β

$$H(\alpha \wedge \beta) = H(\alpha) + H(\beta) \qquad (1)$$

3. Энтропия сложного опыта, состоящего из нескольких независимых, равна сумме энтропий отдельных опытов.

Условная энтропия

Энтропия опыта β при условии, что в опыте α реализовался исход A_i:

$$-\sum_{j=1}^{m} p_{A_i}(B_j) \cdot \log_2 p_{A_i}(B_j) = H_{A_i}(\beta)$$

Средняя условная энтропия *опыта* β *при условии выполнения опыта* α:

$$\sum_{i=1}^{n} p(A_i) \cdot H_{A_i}(\beta) = H_\alpha(\beta)$$

Энтропия сложного опыта:

$$H(\alpha \wedge \beta) = H(\alpha) + H_\alpha(\beta) \qquad (2)$$

Совершенно очевидно, что выражение (1) является частным случаем (2) при условии независимости опытов α и β.

Относительно условной энтропии можно высказать следующие утверждения:

1. Условная энтропия является величиной *неотрицательной*. $H_\alpha(\beta) = 0$ только в том случае, если *любой* исход α полностью определяет исход β, т.е.

$$H_{A_1}(\beta) = H_{A_2}(\beta) = \ldots = H_{A_n}(\beta) = 0.$$

2. Если опыты α и β независимы, то $H_\alpha(\beta) = H(\beta)$, причем это оказывается *наибольшим* значением условной энтропии. Другими словами, опыт α не может повысить неопределенность опыта β; он может либо не оказать никакого влияния (если опыты независимы), либо понизить энтропию β.

$$0 \le H_\alpha(\beta) \le H(\beta)$$

т.е. *условная энтропия не превосходит безусловную*.

3. Из соотношений следует, что $H(\alpha \wedge \beta) \le H(\alpha) + H(\beta)$, причем равенство реализуется только в том случае, если опыты α и β. независимы.

Энтропия и информация

Разность $H(\beta) - H_\alpha(\beta)$ показывает какие новые сведения относительно β мы получаем, производя опыт α. Эта величина называется информацией относительно опыта β, содержащейся в опыте α.

$$I(\alpha, \beta) = H(\beta) - H_\alpha(\beta)$$

Поскольку единицей измерения энтропии является *бит*, то в этих же единицах может быть измерено количество информации.

Энтропия опыта равна той информации, которую мы получаем в результате его осуществления

Формула $I = \log_2 n$ была выведена в 1928 г. американским инженером Р.Хартли и носит его имя. Она связывает *количество равновероятных состояний (n) и количество информации в сообщении (I), что любое из этих состояний реализовалось*. Ее смысл в том, что, если некоторое множество содержит *n* элементов и *x* принадлежит данному множеству, то для его выделения (однозначной идентификации) среди прочих требуется количество информации, равное $\log_2 n$.

Частным случаем применения формулы является ситуация, когда $n = 2^k$; подставляя это значение, получим:

$$I = k \ \ бит$$

Количество информации численно равно числу вопросов с равновероятными бинарными вариантами ответов, которые необходимо задать, чтобы полностью снять неопределенность задачи.

Контрольные вопросы

1. Поясните сущность понятия энтропии.
2. В каких единицах измеряется энтропия?
3. Сформулируйте основные свойства энтропии.

Список задач

1

Имеются два ящика, в каждом из которых по 12 шаров. В первом – 3 белых, 3 черных и 6 красных; во втором – каждого цвета по 4. Опыты состоят в вытаскивании по одному шару из каждого ящика. Что можно сказать относительно неопределенностей исходов этих опытов?

2

В ящике имеются 2 белых шара и 4 черных. Из ящика извлекают последовательно два шара без возврата. Найти энтропию, связанную с первым и вторым извлечениями, а также энтропию обоих извлечений.

3

Имеется три тела с одинаковыми внешними размерами, но с разными массами x_1, x_2 и x_3. Необходимо определить энтропию, связанную с нахождением наиболее тяжелого из них, если сравнивать веса тел можно только попарно.

4

Какое количество информации требуется, чтобы узнать исход броска монеты?

5

Игра «*Угадай-ка–4*». Некто задумал целое число в интервале от 0 до 3. Наш опыт состоит в угадывании этого числа. На наши вопросы Некто может отвечать лишь «*Да*» или «*Нет*». Какое количество информации мы должны получить, чтобы узнать задуманное число, т.е. полностью снять начальную неопределенность? Как правильно построить процесс угадывания?

6

Случайным образом вынимается карта из колоды в 32 карты. Какое количество информации требуется, чтобы угадать, что это за карта? Как построить угадывание?

7

В некоторой местности имеются две близкорасположенные деревни: А или В. Известно, что жители А всегда говорят правду, а жители В – всегда лгут. Известно также, что жители обеих деревень любят ходить друг к другу в гости, поэтому в каждой из деревень можно встретить жителя соседней деревни. Путешественник, сбившись ночью с пути оказался в одной из двух деревень и, заговорив с первым встречным, захотел выяснить, в какой деревне он находится и откуда его собеседник. Какое минимальное количество вопросов с бинарными ответами требуется задать путешественнику?

8

При угадывании результата броска игральной кости задается вопрос «*Выпало 6?*». Какое количество информации содержит ответ?

9

В Петрозаводске 280000 жителей. Какоее минимальное количество вопросов, требующих ответа "да" или "нет", необходимо, чтобы однозначно найти одного жителя.

10

В лотерее N билетов, из них к выигрышных. Студент купил M билетов и после розыгрыша сообщил вам, что выиграл (но, возможно, и не на один билет). Какое количество информации вы получили?

Часть 2

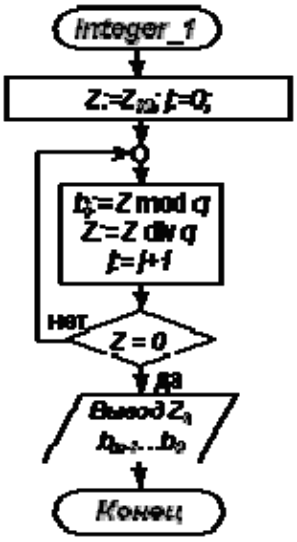
Перевод целых чисел из одной системы счисления в другую

3.2.Общие сведения

Перевод целых чисел из 10-ной системы счисления в систему с произвольным основанием *q*.

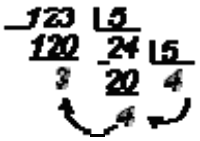
Алгоритм перевода:

1. целочисленно разделить исходное число (Z_{10}) на основание новой системы счисления (q) и найти остаток от деления – это будет цифра 0-го разряда числа Z_q;
2. частное от деления снова целочисленно разделить на q с выделением остатка; процедуру продолжать до тех пор, пока частное от деления не окажется меньше q;
3. образовавшиеся остатки от деления, поставленные в порядке, обратном порядку их получения, и представляют Z_q.



Пример

Выполнить преобразование $123_{10} \rightarrow Z_5$.



Остатки от деления (*3, 4*) и результат последнего целочисленного деления (*4*) образуют обратный порядок цифр нового числа. Следовательно, $\boldsymbol{123_{10} = 443_5}$**.**

Замечание. Полученное число нельзя читать «*четыреста сорок три*», поскольку десятки, сотни, тысячи и прочие подобные обозначения чисел относятся только к десятичной системе счисления. Прочитывать число следует простым перечислением его цифр с указанием системы счисления («*число четыре, четыре, три в пятиричной системе счисления*»).

Алгоритм перевода $\boldsymbol{Z_p \rightarrow Z_{10}}$**:**

необходимо $\boldsymbol{Z_p}$ представить в форме многочлена и выполнить все операции по правилам десятичной арифметики.

Пример

Выполнить преобразование $443_5 \rightarrow Z_{10}$

$443_5 = 4 \cdot 5^2 + 4 \cdot 5^1 + 3 \cdot 5^0 = 4 \cdot 25 + 4 \cdot 5 + 3 \cdot 1 = 123_{10}$

Приведенными алгоритмами удобно пользоваться при переводе числа из десятичной системы в какую-то иную или наоборот. Они работают и для перевода между любыми иными системами счисления, однако, преобразование будет затруднено тем, что все арифметические операции необходимо осуществлять по правилам исходной (в первых алгоритмах) или конечной (в последнем алгоритме) системы счисления. По этой причине переход, например, $Z_3 \rightarrow Z_8$ проще осуществить через промежуточное преобразование к 10-ной системе $Z_3 \rightarrow Z_{10} \rightarrow Z_8$.

Перевод дробных чисел из одной системы счисления в другую

Введем следующие обозначения: правильную дробь в исходной системе счисления p будем записывать в виде $0,Y_p$, дробь в системе q – $0,Y_q$, а преобразование – в виде $0,Y_p \rightarrow 0,Y_q$.

Алгоритмы перевода $0,Y_{10} \rightarrow 0,Y_q$:

1. умножить исходную дробь в 10-ной системе счисления на q, выделить целую часть – она будет первой цифрой новой дроби; отбросить целую часть;
2. для оставшейся дробной части операцию умножения с выделением целой и дробных частей повторять, пока в дробной части не окажется **0** или не будет достигнута желаемая точность конечного числа; появляющиеся при этом целые будут цифрами новой дроби;
3. записать дробь в виде последовательности цифр после нуля с разделителем в порядке их появления в п. (1) и (2).

Пример

Выполнить преобразование $0,375_{10} \rightarrow 0,Y_2$

$$\begin{array}{l} 0,375 \cdot 2 = 0,750 \\ 0,75 \cdot 2 = 1,50 \\ 0,5 \cdot 2 = 1,0 \end{array}$$

Таким образом, $0,375_{10} = 0,011_2$

Перевод $0,Y_p \rightarrow 0,Y_{10}$, как и в случае натуральных чисел, сводится к вычислению значения формы в десятичной системе счисления. Например,

$$0,011_2 = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 0,25 + 0,125 = 0,375_{10}$$

Пример

Выполнить преобразование $5,3(3)_{10} \rightarrow X_3$

Перевод целой части, очевидно, дает: $5_{10} = 12_3$. Перевод дробной части: $0,3(3)_{10} = 0,1_3$. Окончательно: $5,3(3)_{10} = 12,1_3$.

Список заданий

1 Преобразование целых чисел.
Ввод: исходная система счисления, целое число в ней, конечная СС.

Программа 1) проверяет правильность ввода числа в заданной СС;
2) переводит число в 10-ю СС;
3) переводит число в конечную СС.
Вывод: число в 10-й СС и в конечной СС.

2 Преобразование дробных чисел.
Ввод: дробное число в 10-й СС, конечная СС, число знаков в дробной части конечного представления.
Программа переводит дробное число из 10-й СС в указанную СС; количество знаков конечной дроби определяется либо завершением процесса перевода, либо указанным при вводе числом.
Вывод: дробное число в конечной СС.

3 Преобразование вещественных чисел в естественной форме.
Ввод: число в 10-й СС в естественной форме (есть целая и дробная части), конечная СС, число знаков в дробной части конечного представления.
Программа переводит число из 10-й СС в указанную СС; количество знаков конечной дроби определяется либо завершением процесса перевода, либо указанным при вводе числом.
Вывод: вещественное число в конечной СС.

Лабораторная работа № 2

Количество информации

1. Порядок выполнения работы

Ознакомится с методическими указаниями, изложенными в п.3
Решить задачи (по указанию преподавателя)
Подготовить ответы на контрольные вопросы

2. Содержание отчета:

Тема и цель работы
Условия задач
Подробные решения задач
Выводы по работе

3.Общие сведения

В основе теории информации лежит предложенный Шенноном способ измерения количества информации, содержащейся в одной сл. в. относительно другой сл. в. Этот способ приводит к выражению количества информации числом.

Для д.с.в. X и Y, заданных законами распределения $P(X = X_i) = p_i$, $P(Y = Y_j) = q_j$ и совместным распределением $P(X = X_i, Y = Y_j) = p_{ij}$, *количество информации,* содержащейся в X относительно Y, равно

$$I(X,Y) = \sum_{i,j} p_{ij} \log_2 \frac{p_{ij}}{p_i q_j}.$$

Для непрерывных сл. в. X и Y, заданных плотностями распределения вероятностей $p_X(t_1)$, $p_Y(t_2)$ и $P_{XY}(t_1,t_2)$, аналогичная формула имеет вид

$$I(X,Y) = \iint\limits_{\mathbb{R}^2} p_{XY}(t_1,t_2) \log_2 \frac{p_{XY}(t_1,t_2)}{p_X(t_1) p_Y(t_2)} dt_1 dt_2.$$

Очевидно, что

$$P(X = X_i, X = X_j) = \begin{cases} 0, & \text{при } i \neq j \\ P(X = X_i), & \text{при } i = j \end{cases}$$

и, следовательно,

$$I(X,X) = \sum_i p_i \log_2 \frac{p_i}{p_i p_i} = -\sum_i p_i \log_2 p_i.$$

Энтропия д. с. в. X в теории информации определяется формулой

$$H(X) = HX = I(X,X).$$

Свойства меры информации и энтропии:

1) $I(X,Y) \geqslant 0$, $I(X,Y) = 0 \quad \Leftrightarrow \quad X$ и Y независимы;
2) $I(X,Y) = I(Y,X)$;
3) $HX = 0 \quad \Leftrightarrow \quad X$ — константа;
4) $I(X,Y) = HX + HY - H(X,Y)$, где $H(X,Y) = -\sum_{i,j} p_{ij} \log_2 p_{ij}$;
5) $I(X,Y) \leqslant I(X,X)$. Если $I(X,Y) = I(X,X)$, то X — функция от Y.

Контрольные вопросы

1. Как определяется энтропия дискретной системы с равновероятными и неравновероятными состояниями?

2. Чему равна энтропия при неравновероятном и взаимозависимом распределении элементов системы?
3. В чем заключается особенность определения энтропии для непрерывных распределений?
4. Что такое условная и совместная энтропия?
5. Сформулируйте основные свойства энтропии сложных сообщений.
6. Поясните связь между энтропией и информацией.
7. Какие основные требования предъявляется к мере количества информации?
8. Как количественно оценивается информация при полной и неполной достоверности сообщений?

Список задач

1

Найти энтропию д. с. в. X, заданной распределением

X	1	2	3	4	5	6	7	8
p	0.1	0.2	0.1	0.05	0.1	0.05	0.3	0.1

2

Значения д. с. в. X_1 и X_2 определяются подбрасыванием двух идеальных монет, а д.с.в. Y равна сумме количества "гербов", выпавших при подбрасывании этих монет. Сколько информации об X_1 содержится в Y?

3

Сколько информации об X_1 содержится в д. с. в. $Z = (X_1 + 1)^2 — X_2$, где независимые д. с. в. X_1 и X_2 могут с равной вероятностью принимать значение либо 0, либо 1? Найти HX_1 и HZ. Каков характер зависимости между X_1 и Z?

4

Д.св. X_1, X_2 — зависимы и распределены также как и соответствующие д.с.в. из предыдущей задачи. Найти $I(X_1,X_2)$, если совместное распределение вероятностей X_1 и X_2 описывается законом

X_1	0	0	1	1
X_2	0	1	0	1
p	1/3	1/6	1/6	1/3

5

Д. с. в. X_1 и X_2 определяются подбрасыванием двух идеальных тетраэдров, грани которых помечены числами от 1 до 4. Д. с. в. Y равна сумме чисел, выпавших при подбрасывании этих тетраэдров, т. е. $Y = X_1 + X_2$. Вычислить $I(X_1, Y)$, HX_1 и HY.

6

Подсчитать сколько информации об X_1 содержится в д. с. в. $Z = X_1 * X_2$, а также HZ. Д. с. в. X_1 и X_2 берутся из предыдущего упражнения.

7

Д. с. в. X_1 может принимать три значения −1, 0 и 1 с равными вероятностями. Д. с. в. X_2 с равными вероятностями может принимать значения 0, 1 и 2. X_1 и X_2 — независимы. $Y = X_1^2 + X_2$. Найти $I(X_1,Y)$, $I(X_2, Y)$, HX_1, HX_2, HY.

8

Найти энтропии д. с. в. X, Y, Z и количество информации, содержащейся в $Z = X + Y$ относительно Y. X и Y — независимы и задаются распределениями

X	0	1	3	4
P	*1/8*	*1/8*	*1/4*	*1/2*

Y	-2	2
p	*3/8*	*5/8*

9

Сравнить неопределенность, приходящуюся на букву источника информации "Г" (алфавит русского языка), характеризуемую ансамблем, представленным в таблице, с неопределенностью, которая была бы у того же источника при равновероятном использовании букв.

Буква	Вероятность	Буква	Вероятность	Буква	Вероятность
а	0.064	л	0.036	ц	0.040
б	0.015	м	0.026	ч	0.013
в	0.039	н	0.056	ш	0.006
г	0.014	о	0.096	щ	0.003
д	0.026	п	0.024	ь, ъ	0.015
е	0.074	р	0.041	ы	0.016
ж	0.008	с	0.047	э	0.003
з	0.015	т	0.056	ю	0.007
и	0.064	у	0.021	я	0.019
и	0.010	ф	0.020	-	0.143
к	0.029	х	0.090		

10

Заданы ансамбли U и V двух дискретных случайных величин u и v:

U= 4 3 15 10 V= 0.1 0.3 0.5 0.8

0.25 0.25 0.25 0.25 0.3 0.3 0.4 0.1

Сравнить их энтропии.

11

Определить энтропии H(U), H(V), H(U/V), H(U,V), если задана матрица вероятностей состояний системы, объединяющей источники u и v:

$$P(u,v)= \begin{matrix} 0.5 & 0.4 & 0.3 \\ 0.1 & 0.2 & 0.1 \\ 0.2 & 0.3 & 0.4 \end{matrix}$$

Лабораторная работа № 3

Простейшие алгоритмы сжатия информации

1. Порядок выполнения работы

Ознакомится с методическими указаниями, изложенными в п.3;

Выполнить задания (по указанию преподавателя)

2. Содержание отчета:

Тема и цель работы

Условия заданий

Подробное решение, в заданиях, где требуется составление программ, к сдаче представляется программа (исходные файлы и исполняемый модуль), блок-схема программы

Выводы по работе.

3.Общие сведения

Для случая отсутствия статистической взаимосвязи между знаками конструктивные методы построения эффективных кодов были даны впервые американскими учеными Шенноном и Фано. Их методики существенно не различаются и поэтому соответствующий код получил название кода Шеннона — Фано.

Код строят следующим образом: знаки алфавита сообщений выписывают в таблицу в порядке убывания вероятностей. Затем их разделяют на две группы так, чтобы суммы вероятностей в каждой из групп были по возможности одинаковы. Всем знакам верхней половины в качестве первого символа приписывают 0, а всем нижним — 1. Каждую из полученных групп, в свою очередь, разбивают на две подгруппы с одинаковыми суммарными вероятностями и т. д. Процесс повторяется до тех пор, пока в каждой подгруппе останется по одному знаку.

Пример 1.

Знаки	Вероятность	Кодовые комбинации	Ступень разбиения
z_1	1/2	1	I
z_2	1/4	01	II
z_3	1/8	001	III
z_4	1/16	0001	IV
z_5	1/32	00001	V
z_6	1/64	000001	VI
z_7	1/128	0000001	VII
z_8	1/128	0000000	

Пример 2

Знаки	Вероятности	Кодовые комбинации	Ступень разбиения
z_1	0,22	11	I
z_2	0,20	10	IV
z_3	0,16	011	III
z_4	0,16	010	V
z_5	0,10	001	VI
z_6	0,10	0001	VII
z_7	0,04	00001	
z_8	0,02	00000	

Рассмотренная методика Шеннона — Фано не всегда приводит к однозначному построению кода. Ведь при разбиении на подгруппы можно сделать большей по вероятности как верхнюю, так и нижнюю подгруппы.

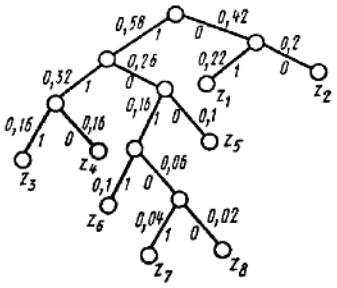
От указанного недостатка свободна методика Хаффмена.

Метод Хаффмена (Huffman) разработан в 1952 г. Он более практичен и никогда по степени сжатия не уступает методу Шеннона-Фэно, более того, он сжимает максимально плотно. Код строится при помощи

двоичного (бинарного) дерева. Вероятности значений д. с. в. приписываются его листьям; все дерево строится, опираясь на листья. Величина, приписанная к узлу дерева, называется весом узла. Два листа с наименьшими весами создают родительский узел с весом, равным сумме их весов; в дальнейшем этот узел учитывается наравне с оставшимися листьями, а образовавшие его узлы от такого рассмотрения устраняются. После постройки корня нужно приписать каждой из ветвей, исходящих из родительских узлов, значения 0 или 1. Код каждого значения д. с. в. — это число, получаемое при обходе ветвей от корня к листу, соответствующему данному значению.

Пример 3

Знаки	Вероятности	Вспомогательные столбцы						
		1	2	3	4	5	6	7
z_1	0,22	0,22	0,22	→0,26	→0,32	→0,42	→0,58	→1
z_2	0,20	0,20	0,20	0,22	0,26	0,32	0,42	
z_3	0,16	0,16	0,16	0,20	0,22	0,26		
z_4	0,16	0,16	0,16	0,16	0,20			
z_5	0,10	0,10	→0,16	0,16				
z_6	0,10	0,10	0,10					
z_7	0,04	→0,06						
z_8	0,02							



Теперь, двигаясь по кодовому дереву сверху вниз, можно записать для каждой буквы соответствующую ей кодовую комбинацию:

Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8
01	00	111	110	100	1011	10101	10100

Для методов Хаффмена и Шеннона-Фэно каждый раз вместе с собственно сообщением нужно передавать и таблицу кодов.

Список заданий

1

Вычислить HX и ML(X) для кодов Хаффмена и Шеннона-Фэно для X.
Д. с. в. X задается следующим распределением вероятностей:

X	1	2	3	4	5
p	7/18	1/6	1/6	1/6	1/9

2

Вычислить среднее количество бит на единицу сжатого сообщения о значении каждой из д. с. в., из заданных следующими распределениями вероятностей, при сжатии методами Шеннона-Фэно и Хаффмена

X_1	1	2	3	4
p	1/3	1/3	1/6	1/6

X_2	1	2	5	6	7
p	0.2	0.1	0.3	0.25	0.15

X_3	1	4	9	16	25	36	49
p	0.1	0.1	0.1	0.3	0.1	0.1	0.2

X_4	-2	-1	0	1	2
p	1/3	1/4	1/5	1/6	1/20

3

Вычислить длину кода Хаффмена для сообщения AAB, полученного от д.св. X со следующим распределением вероятностей P(X = A) = 1/3, P(X = B) = 2/3.

4

Неравномерное алфавитное двоичное кодирование введенного текста по методу Хаффмана.

Работа программы:

Ввод: текст (последовательность символов).

Программа определяет относительные частоты появления знаков в тексте и строит таблицу неравномерных двоичных кодов Хаффмана.

Вывод: (1) таблица кодов; (2) закодированная фраза; (3) средняя информация на знак; избыточность.

5

Неравномерное алфавитное двоичное кодирование (декодирование) по методу Хаффмана

Работа программы:

Режим 1: ввод знаков алфавита и их вероятностей; вывод – таблица кодов (таблица сохраняется в виде файла и выводится на экран);

Режим 2: кодирование: ввод – фраза; вывод – последовательность кодов (вывод на экран) (кодирование производится в соответствии со сформированной ранее таблицы кодов); вывести среднюю информацию на знак; избыточность.

Режим 3: декодирование: ввод – закодированное сообщение; вывод – декодированное сообщение в соответствии со сформированной ранее таблицы кодов.

6

Неравномерное алфавитное двоичное кодирование (декодирование) по методу Шеннона-Фано

Работа программы:

Режим 1: ввод знаков алфавита и их вероятностей; вывод – таблица кодов; таблица сохраняется в виде файла и выводится на экран.

Режим 2: кодирование: ввод – фраза; вывод – последовательность кодов (вывод на экран); средняя информация на знак; избыточность. Кодирование производится в соответствии со сформированной ранее таблицы кодов.

Режим 3: декодирование: ввод – закодированное сообщение; вывод – декодированное сообщение в соответствии со сформированной ранее таблицей кодов.

Лабораторная работа № 4

Арифметическое кодирование

1. Порядок выполнения работы

Ознакомится с методическими указаниями, изложенными в п.3;

Выполнить задания (по указанию преподавателя)

2. Содержание отчета:

Тема и цель работы

Условия заданий

Подробное решение

В заданиях, где требуется составление программ, к сдаче представляется программа (исходные файлы и исполняемый модуль), блок-схема программы

Выводы по работе.

3.Общие сведения

По исходному распределению вероятностей для выбранной для кодирования д. с. в. строится таблица, состоящая из пересекающихся только в граничных точках отрезков для каждого из значений этой д.с.в.; объединение этих отрезков должно образовывать отрезок [0,1], а их длины должны быть пропорциональны вероятностям соответствующих значений д. с. в. Алгоритм кодирования заключается в построении отрезка, однозначно определяющего данную последовательность значений д. с. в. Затем для построенного отрезка находится число, принадлежащее его внутренней части и равное целому числу, деленному на минимально возможную положительную целую степень двойки. Это число и будет кодом для рассматриваемой последовательности. Все возможные конкретные коды — это числа строго большие нуля и строго меньшие одного, поэтому можно отбрасывать лидирующий ноль и десятичную точку, но нужен еще один специальный код-маркер, сигнализирующий о конце сообщения. Отрезки строятся так. Если имеется отрезок для сообщения длины n - 1, то для построения отрезка для сообщения длины n, разбиваем его на столько же частей, сколько значений имеет рассматриваемая д. с. в. Это разбиение делается совершенно также как и самое первое (с сохранением порядка). Затем выбирается из полученных отрезков тот, который соответствует заданной конкретной последовательности длины n.

Принципиальное отличие этого кодирования от рассмотренных ранее методов в его непрерывности, т. е. в ненужности блокирования. Эффективность арифметического кодирования растет с ростом длины сжимаемого сообщения (для кодирования Хаффмена или Шеннона-Фэно этого не происходит).

При сжатии заданных данных, например, из файла все рассмотренные методы требуют двух проходов. Первый для сбора частот символов, используемых как приближенные значения вероятностей символов, и второй для собственно сжатия.

Получение исходного сообщения из его арифметического кода происходит по следующему алгоритму.

Шаг 1. В таблице для кодирования значений д.с.в. определяется интервал, содержащий текущий код, — по этому интервалу однозначно определяется один символ исходного сообщения. Если этот символ — это маркер конца сообщения, то конец.

Шаг 2. Из текущего кода вычитается нижняя граница содержащего его интервала, полученная разность делится на длину этого же интервала. Полученное число считается новым текущим значением кода. Переход к шагу 1.

Список заданий

1

Вычислить среднее количество бит на единицу сжатого сообщения о значении каждой из д. с. в., из заданных следующими распределениями вероятностей, при сжатии арифметическим методом

X_1	1	2	3	4
p	1/3	1/3	1/6	1/6

X_2	1	2	5	6	7
p	0.2	0.1	0.3	0.25	0.15

X_3	1	4	9	16	25	36	49
p	0.1	0.1	0.1	0.3	0.1	0.1	0.2

X_4	-2	-1	0	1	2
p	1/3	1/4	1/5	1/6	1/20

2

Вычислить длину арифметического кода для сообщения AAB, полученного от д.св. X со следующим распределением вероятностей P(X = A) = 1/3, P(X = B) = 2/3.

3

Составить арифметический код для сообщения BAABC, полученного от д. с. в. X со следующим распределением вероятностей P(X = A) = 1/4, P(X = B) = 1/2, P(X = C) = 1/4. Каков будет арифметический код для этого же сообщения, если X распределена по закону P(X = A) = 1/3, P(X = B) = 7/15, P(X = C) = 1/5?

4

Д. с. в. X может принимать три различных значения. При построении блочного кода с длиной блока 4 для X необходимо будет рассмотреть д.св. $\overline{X}$ — выборку четырех значений X. Сколько различных значений может иметь $\overline{X}$? Если считать сложность построения кода пропорциональной количеству различных значений кодируемой д. с. в., то во сколько раз сложнее строить блочный код для X по сравнению с неблочным?

Лабораторная работа № 5

Адаптивные алгоритмы сжатия

1. Порядок выполнения работы

1.1.Ознакомится с методическими указаниями, изложенными в п.3;

1.2.Выполнить задания (по указанию преподавателя)

2. Содержание отчета:

2.1.Тема и цель работы

2.2.Условия заданий

2.3.Подробное решение, в заданиях, где требуется составление программ, к сдаче представляется программа (исходные файлы и исполняемый модуль), блок-схема программы

2.4.Выводы по работе.

3.Общие сведения

Различия статической и динамической моделей

Практически все существующие алгоритмы сжатия данных можно реализовать в двух вариантах - статическом и динамическом. Каждый из них имеет своё право на жизнь и, соответственно, занимает определённую нишу, используясь в тех случаях, когда он наиболее удобен. Алгоритм Хаффмана не является исключением - применяясь во многих и многих архиваторах и форматах он используется иногда в статическом (*JPEG, ARJ*) или в динамическом (*утилита compact для UNIX, ICE*) вариантах.

В чём же заключается различие между этими двумя типами реализаций? Так как алгоритм Хаффмана сжимает данные за счёт вероятностей появления символов в источнике, следовательно для того чтоб успешно что-то сжать и разжать нам потребуется знать эти самые вероятности для каждого символа. Статические алгоритмы справляются с этим делом не мудрствуя лукаво - перед тем как начать сжатие файла программа быстренько пробегается по самому файлу и подсчитывает какой символ сколько раз встречается. Затем, в соответствии с вероятностями появлений символов, строится двоичное дерево Хаффмана - откуда извлекаются соответствующие каждому символу коды разной длины. И на третьем этапе снова осуществляется проход по исходному файлу, когда каждый символ заменяется на свой древесный код. Таким образом статическому алгоритму требуется два прохода по файлу источнику, чтоб закодировать данные.

Динамический алгоритм позволяет реализовать однопроходную модель сжатия. Не зная реальных вероятностей появлений символов в исходном файле - программа постепенно изменяет двоичное дерево с каждым встречаемым символом увеличивая частоту его появления в дереве и перестраивая в связи с этим само дерево. Однако становится очевидным, что, выиграв в количестве проходов по исходному файлу - мы начинаем терять в качестве сжатия, так как в статическом алгоритме частоты встречаемости символов были известны с самого начала и длины кодов этих символов были более близки к оптимальным, в то время как динамическая модель изучая источник постепенно доходит до его реальных частотных характеристик и узнаёт их лишь полностью пройдя исходный файл. Конечно, обидно, но сей недостаток можно компенсировать другим преимуществом динамического алгоритма. Так как динамическое двоичное дерево постоянно модифицируется новыми символами - нам нет необходимости запоминать их частоты заранее - при разархивировании, программа, получив из архива код символа, точно так же восстановит дерево, как она это делала при сжатии и увеличит на единичку частоту его встречаемости. Более того, нам не требуется запоминать какие символы в двоичном дереве не встречаются. Все символы которые будут добавляться в дерево и есть те, которые нам потребуются для восстановления первоначальных данных. В

статическом алгоритме с этим делом немного сложнее - в самом начале сжатого файла требуется хранить информацию о встречаемых в файле источнике символах и их вероятностных частотах. Это обусловлено тем, что ещё до начала разархивирования нам необходимо знать какие символы будут встречаться и каков будет их код.

Адаптивный алгоритм Хаффмана.

В начале работы алгоритма дерево кодирования содержит только один специальный символ, всегда имеющий частоту 0. Он необходим для занесения в дерево новых символов: после него код символа передается непосредственно. Обычно такой символ называют escape-символом ((ESC)). Расширенный ASCII кодируют каждый символ 8-битным числом, т.е. числом от 0 до 255. При построении дерева кодирования необходимо для возможности правильного декодирования как-то упорядочивать структуру дерева. Расположим листья дерева в порядке возрастания частот и затем в порядке возрастания стандартных кодов символов. Узлы собираются слева направо без пропусков. Левые ветви помечаются 0, а правые — 1.

Список заданий

1

Закодировать сообщение BBCBBC, используя адаптивный алгоритм Хаффмена с упорядоченным деревом.

2

Закодировать сообщения "AABCDAACCCCDBB", "КИБЕРНЕТИКИ" и "СИНЯЯ СИНЕВА СИНИ", используя адаптивный алгоритм Хаффмена с упорядоченным деревом. Вычислить длины в битах исходного сообщения в коде ASCII+ и его полученного кода.

3

Распаковать сообщение 'A'0'F'00'X'01111101010111011110100101, полученное по адаптивному алгоритму Хаффмена с упорядоченным деревом, рассчитать длину кода сжатого и несжатого сообщения в битах.

Лабораторная работа № 6

Адаптивное арифметическое кодирование

1. Порядок выполнения работы

1.1.Ознакомится с методическими указаниями, изложенными в п.3;

1.2.Выполнить задания (по указанию преподавателя)

2. Содержание отчета:

2.1.Тема и цель работы

2.2.Условия заданий

2.3.Подробное решение

В заданиях, где требуется составление программ, к сдаче представляется программа (исходные файлы и исполняемый модуль), блок-схема программы

2.4.Выводы по работе.

3.Общие сведения

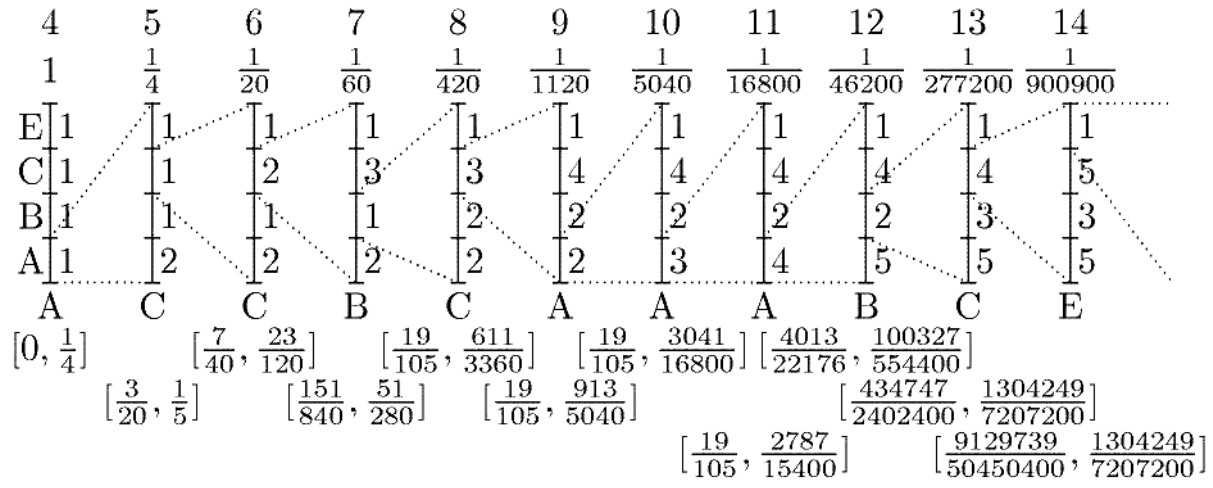
Для арифметического кодирования, как и для кодирования методом Хаффмена, существуют адаптивные алгоритмы. Реализация одного из них запатентована фирмой IBM.

Построение арифметического кода для последовательности символов из заданного множества можно реализовать следующим алгоритмом. Каждому символу сопоставляется его вес: вначале он для всех равен 1. Все символы располагаются в естественном порядке, например, по возрастанию. Вероятность каждого символа устанавливается равной его весу, деленному на суммарный вес всех символов. После получения очередного символа и постройки интервала для него, вес этого символа увеличивается на 1 (можно увеличивать вес любым регулярным способом).

Заданное множество символов — это, как правило, ASCII+. Для того, чтобы обеспечить остановку алгоритма распаковки вначале сжимаемого сообщения надо поставить его длину или ввести дополнительный символ-маркер конца сообщения. Если знать формат файла для сжатия, то вместо начального равномерного распределения весов можно выбрать распределение с учетом этих знаний. Например, в текстовом файле недопустимы ряд управляющих символов и их вес можно занулить.

Пример 1.

Пусть заданное множество — это символы A, B, C. Сжимаемое сообщение — ACCBCAAABC. Введем маркер конца сообщения — E. Кодирование согласно приведенному алгоритму можно провести согласно схеме



Так как

$$\frac{759021}{2^{22}=4194304} = 0.00101110010100111011012 \in \left(\frac{9129739}{50450400}, \frac{1304249}{7207200}\right),$$

$$code(\text{ACCBCAAABC}) = 00101110010100111011012 \text{ и}$$

$$L(\text{ACCBCAAABC}) = 22.$$

Способ распаковки адаптивного арифметического кода почти аналогичен приведенному для неадаптивного. Отличие только в том, что на втором шаге после получения нового кода нужно перестроить разбиение единичного отрезка согласно новому распределению весов символов. Получение маркера конца или заданного началом сообщения числа символов означает окончание работы.

Пример 2.

Распакуем код 001011100101001110112, зная, что множество символов сообщения состоит из A, B, C и E, причем последний — это маркер конца сообщения.

$$0.001011100101001110112 = \frac{759021}{4194304}.$$

Веса				Число-код и его интервал	Символ	Длина интервала
A	B	C	E			
1	1	1	1	$\frac{759021}{4194304} \in (0, \frac{1}{4})$	A	$\frac{1}{4}$
2	1	1	1	$\frac{759021}{1048576} \in (\frac{3}{5}, \frac{4}{5})$	C	$\frac{1}{5}$
2	1	2	1	$\frac{649377}{1048576} \in (\frac{1}{2}, \frac{5}{6})$	C	$\frac{1}{3}$
2	1	3	1	$\frac{375267}{1048576} \in (\frac{2}{7}, \frac{3}{7})$	B	$\frac{1}{7}$
2	2	3	1	$\frac{529717}{1048576} \in (\frac{1}{2}, \frac{7}{8})$	C	$\frac{3}{8}$
2	2	4	1	$\frac{5429}{393216} \in (0, \frac{2}{9})$	A	$\frac{2}{9}$
3	2	4	1	$\frac{16287}{262144} \in (0, 0.3)$	A	0.3
4	2	4	1	$\frac{27145}{131072} \in (0, \frac{4}{11})$	A	$\frac{4}{11}$
5	2	4	1	$\frac{298595}{524288} \in (\frac{5}{12}, \frac{7}{12})$	B	$\frac{1}{6}$
5	3	4	1	$\frac{240425}{262144} \in (\frac{8}{13}, \frac{12}{13})$	C	$\frac{4}{13}$
5	3	5	1	$\frac{1028373}{1048576} \in (\frac{13}{14}, 1)$	E	

Список заданий

1

Составить адаптивный арифметический код с маркером конца для сообщения B A ABC.

Лабораторная работа № 7

Подстановочные или словарно- ориентированные алгоритмы сжатия информации. Методы Лемпела-Зива

1. Порядок выполнения работы

1.1.Ознакомится с методическими указаниями, изложенными в п.3;

1.2.Выполнить задания (по указанию преподавателя)

2. Содержание отчета:

2.1.Тема и цель работы

2.2.Условия заданий

2.3.Подробное решение

В заданиях, где требуется составление программ, к сдаче представляется программа (исходные файлы и исполняемый модуль), блок-схема программы

2.4.Выводы по работе.

3.Общие сведения

Практически все словарные методы кодирования принадлежат семье алгоритмов из работы двух израильских ученых - Зива и Лемпела, опубликованной в 1977 году. *Сущность их состоит в том, что фразы в сжимаемом тексте заменяются указателем на то место, где они в этом тексте уже ранее появлялись*.

Это семейство алгоритмов называется методом Зива-Лемпела и обозначается как *LZ-сжатие*. Этот метод быстро приспосабливается к структуре текста и может кодировать короткие функциональные слова, так как они очень часто в нем появляются. Новые слова и фразы могут также формироваться из частей ранее встреченных слов.

Декодирование сжатого текста осуществляется напрямую - происходит простая замена указателя готовой фразой из словаря, на которую тот указывает. На практике LZ-метод добивается хорошего сжатия, его важным свойством является очень быстрая работа декодера. (Когда мы говорим о тексте, то предполагаем, что кодированию подвергается некоторый вектор данных с конечным дискретным алфавитом, и это не обязательно текст в буквальном смысле этого слова.)

Большинство словарных методов кодирования носят имя авторов идеи метода Зива и Лемпела, и часто считают, что все они используют один и тот же алгоритм кодирования. На самом деле разные представители этого семейства алгоритмов очень сильно различаются в деталях своей работы.

Все словарные методы кодирования можно разбить на две группы.

Методы, принадлежащие к *первой группе,* находя в кодируемой последовательности цепочки символов, которые ранее уже встречались, вместо того, чтобы повторять эти цепочки, заменяют их указателями на предыдущие повторения.

Словарь в этой группе алгоритмов в неявном виде содержится в обрабатываемых данных, сохраняются лишь указатели на встречающиеся цепочки повторяющихся символов.

Все методы этой группы базируются на алгоритме, разработанном и опубликованном, как уже отмечалось, сравнительно недавно - в 1977 году Абрахамом Лемпелем и Якобом Зивом, - LZ77. Наиболее совершенным представителем этой группы, включившим в себя все достижения, полученные в данном направлении, является алгоритм LZSS, опубликованный в 1982 году Сторером и Шимански.

Процедура кодирования в соответствии с алгоритмами этой группы иллюстрируется рис.1.

Алгоритмы второй группы в дополнение к исходному словарю источника создают словарь фраз, представляющих собой повторяющиеся комбинации символов исходного словаря, встречающиеся во входных данных. При этом размер словаря источника возрастает, и для его

кодирования потребуется большее число бит, но значительная часть этого словаря будет представлять собой уже не отдельные буквы, а буквосочетания или целые слова. Когда кодер обнаруживает фразу, которая ранее уже встречалась, он заменяет ее индексом словаря, содержащим эту фразу. При этом длина кода индекса получается меньше или намного меньше длины кода фразы.

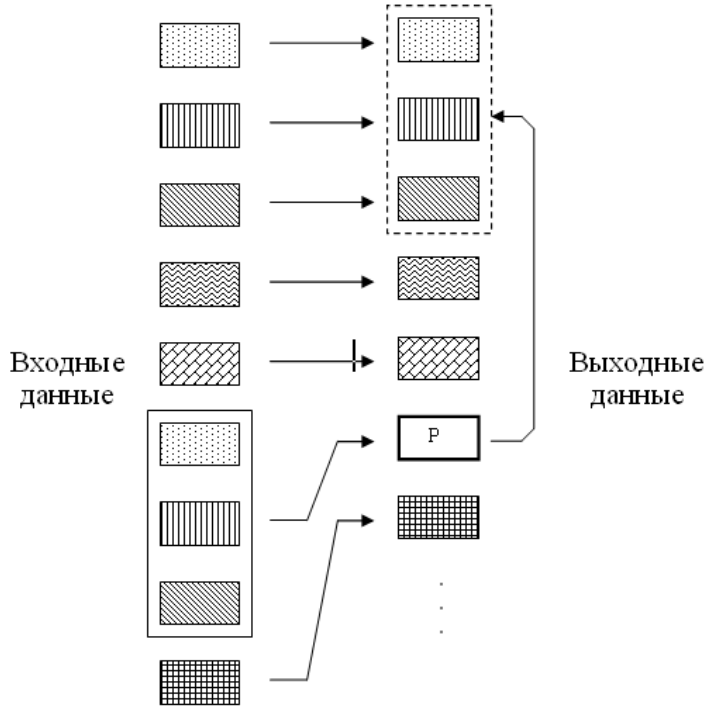


Рис.1

Все методы этой группы базируются на алгоритме, разработанном и опубликованном Лемпелем и Зивом в 1978 году, – LZ78. Наиболее совершенным на данный момент представителем этой группы словарных методов является алгоритм LZW, разработанный в 1984 году Терри Вэлчем.

Идею этой группы алгоритмов можно также пояснить с помощью рис. 2.

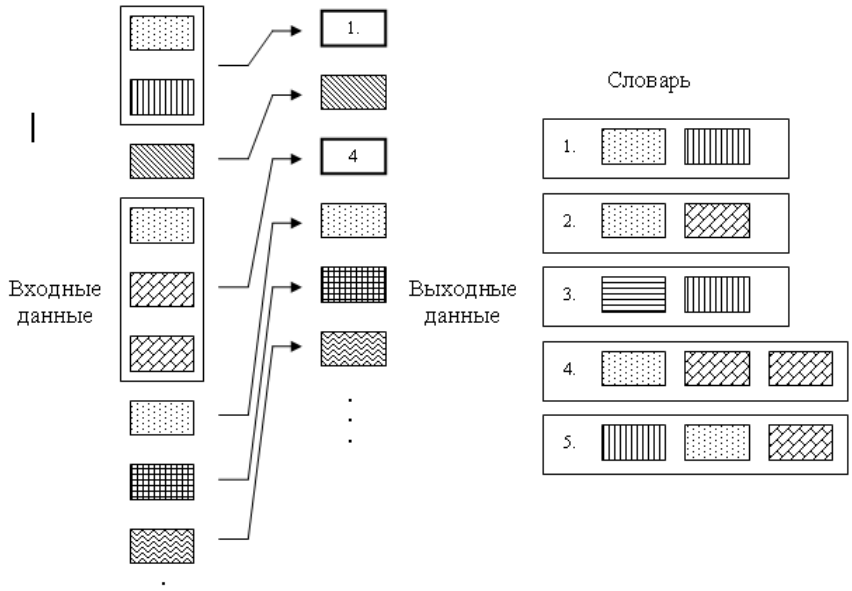


Рис 2.

Алгоритмы второй группы несколько проще в объяснении их работы, поэтому начнем рассмотрение принципа действия LZ-кодеров с алгоритма LZW.

Рассмотрим в самом общем виде работу LZW-кодера.

Процесс сжатия выглядит достаточно просто. Мы последовательно считываем символы входного потока (строку) и проверяем, есть ли в уже созданной нами таблице такая строка.

Если строка есть, то считываем следующий символ, а если такой строки нет, - заносим в выходной поток код для предыдущей найденной строки, заносим ее в таблицу и начинаем поиск снова.

Пусть на вход кодера поступает последовательность символов вида **/ WED / WE / WEE / WEB**, при этом размер алфавита входных символов ***dim A = 255.***

Схема сжатия выглядит следующим образом:

Входные символы	Выходной код	Новые символы словаря
/W	***/***	256 = ***/W***
E	***W***	257 = ***WE***
D	***E***	258 = ***ED***
/	***D***	259 = ***D/***
WE	***256***	260 = ***/WE***
/	***E***	261 = ***E/***
WEE	***260***	262 = ***/WEE***
/W	***261***	263 = ***E/W***
EB	***257***	264 = ***WEB***
<END>	***B***	

В результате получим выходной код

/WED<256>E<260><261><257>B.

Как при этом изменилась длина выходного кода в сравнении с входным ?

Если для двоичного кодирования строки ***/ WED / WE / WEE / WEB*** длиной в 15 букв и размером алфавита ***dim A = 255*** нам понадобилось бы ***15 • $\log_2$ 255 = 15x8 = 120 бит***, то для двоичного кодирования выходной строки кодера ***/ WED <256> E <260> <261> <257> B*** длиной в 10 новых символов с алфавитом в 264 буквы – ***10 • 9 = 90 бит.***

Алгоритм LZW-сжатия выглядит следующим образом:

```
set w = NIL
loop
   read a character K
   if wK exists in the dictionary
      w = wK
   else
      output the code for w
      add wK to the string table
      w = K
endloop
```

LZW-кодер начинает работу со словаря размером в 4K, который содержит по адресам от 0 до 255 ссылки на отдельные буквы и от 256 до 4095 – ссылки на подстроки. В процессе кодирования текст подвергается разбору на подстроки, где каждая новая подстрока есть самая длинная из уже просмотренных плюс один символ. Она кодируется как индекс ее префикса плюс дополнительный символ, после чего новая подстрока добавляется в словарь и на нее в дальнейшем можно будет ссылаться.

Работа кодера семейства LZ77 - первой опубликованной версии LZ-метода - выглядит несколько иначе.

В алгоритме LZ77 указатели обозначают фразы в окне постоянного размера, предшествующие позиции кода. Максимальная длина заменяемых указателями подстрок определяется параметром ***F*** (обычно это от 10 до 20). Эти ограничения позволяют LZ77 использовать "скользящее окно" из ***N*** символов. Из них первые ***N-F*** были уже закодированы, а последние ***F*** составляют упреждающий буфер.

При кодировании символа в первых ***N-F*** символах окна ищут самую длинную, совпадающую с этим буфером строку. Она может частично перекрывать буфер, но не может быть самим буфером.

Найденное наибольшее соответствие затем кодируется триадой [***i, j, a***] где ***i*** есть его смещение от начала буфера, ***j*** - длина соответствия, ***a*** - первый символ, не соответствующий подстроке окна.

Затем окно сдвигается вправо на ***j +1*** символ и готово к новому шагу алгоритма.

Привязка определенного символа к каждому указателю гарантирует, что кодирование будет выполняться даже в том случае, если для первого символа упреждающего буфера не будет найдено соответствие.

Объем памяти, требуемый кодеру и декодеру, ограничивается размером окна. Количество бит, необходимое для представления смещения (***i***) в триаде, составляет [***log(N-F)***]. Количество символов (***j***), заменяемых триадой, может быть закодировано [***logF***] битами.

Алгоритм кодирования для LZ77 приведен ниже.

Кодер:

```
while( lookAheadBuffer not empty )
   {
   get a pointer ( position, match ) to the longest match in the window
      for the lookahead buffer;

   if( length > MINIMUM_MATCH_LENGTH )
      {
      output a ( position, length ) pair;
      shift the window length characters along;
      }
   else
      {
      output the first character in the lookahead buffer;
      shift the window 1 character along;
      }
   }
```

Список заданий

1

Закодировать сообщение "AABCDAACCCCDBB, вычислить длины в битах полученных кодов, используя алгоритмы,
LZ77 (словарь — 12 байт, буфер — 4 байта),

LZ78 (словарь — 16 фраз),
LZSS (словарь — 12 байт, буфер — 4 байта),
LZW (словарь — ASCII+ и 16 фраз).

2

Закодировать сообщение "КИБЕРНЕТИКИ", вычислить длины в битах полученных кодов, используя алгоритмы,
LZ77 (словарь — 12 байт, буфер — 4 байта),
LZ78 (словарь — 16 фраз),
LZSS (словарь — 12 байт, буфер — 4 байта),
LZW (словарь — ASCII+ и 16 фраз).

3

Закодировать сообщение "СИНЯЯ СИНЕВА СИНИ", вычислить длины в битах полученных кодов, используя алгоритмы,
LZ77 (словарь — 12 байт, буфер — 4 байта),
LZ78 (словарь — 16 фраз),
LZSS (словарь — 12 байт, буфер — 4 байта),
LZW (словарь — ASCII+ и 16 фраз).

4

Может ли для первого символа сообщения код LZ78 быть короче кода LZW при одинаковых размерах словарей? Обосновать. Для LZW в размер словаря не включать позиции для ASCII+.

Лабораторная работа № 8

LZ- алгоритмы распаковки данных

1. Порядок выполнения работы

1.1.Ознакомится с методическими указаниями, изложенными в п.3;

1.2.Выполнить задания (по указанию преподавателя)

2. Содержание отчета:

2.1.Тема и цель работы

2.2.Условия заданий

2.3.Подробное решение

В заданиях, где требуется составление программ, к сдаче представляется программа (исходные файлы и исполняемый модуль), блок-схема программы

2.4.Выводы по работе.

3.Общие сведения

Рассмотрим в самом общем виде работу LZW- декодера.

LZW-декодер, обрабатывая входной поток закодированных данных, восстанавливает из него исходные данные. Так же, как и алгоритм сжатия, декодер добавляет новые строки в словарь всякий раз, когда находит во входном потоке новый код. Все, что ему остается сделать, – это преобразовать входной код в выходную строку символов и отдать ее на выход кодера.

Схема работы LZW-декодера выглядит следующим образом:

строка на входе кодера - ***/WED<256>E<260><261><257>B.***

Входные символы	Выходная строка	Новые символы словаря
/	***/***	
W	***W***	256 = ***/W***
E	***E***	257 = ***WE***
D	***D***	258 = ***ED***
256	***/W***	259 = ***D/***
E	***E***	260 = ***/WE***
260	***/WE***	***261*** = ***E/***
261	***E/***	262 = ***/WEE***
257	***WE***	***263*** = ***E/W***
B	***B***	264 = ***WEB***

Самым замечательным качеством этого способа сжатия является то, что *весь словарь новых символов передается декодеру без собственно передачи*. В конце процесса декодирования декодер имеет точно такой же словарь новых символов, какой в процессе кодирования был накоплен кодером, при этом его создание было частью процесса декодирования.

Работа декодера семейства LZ77 - первой опубликованной версии LZ-метода

Декодирование осуществляется очень просто и быстро. При этом поддерживается тот же порядок работы с окном, что и при кодировании, но в отличие от поиска одинаковых строк он, наоборот, копирует их из окна в соответствии с очередной триадой.

Алгоритм декодирования для LZ77 приведен ниже.

Декодер:

Whenever a (position, length) pair is

encountered, go to that (position) in the window and copy (length) bytes to the output.

Список заданий

Распаковать каждое приведенное сообщение и рассчитать длину кода каждого сжатого сообщения в битах.

1

Сообщение, сжатое LZ77 (словарь — 12 байт, буфер — 4 байта), — (0,0,'A') (0,0,'F') (0,0,'X') (9,2,'F') (8,1,'F') (6,2,'X') (4,3,'A').

2

Сообщение, сжатое LZSS (словарь— 12 байт, буфер — 4 байта), — O'A' O'F' O'X' 1(9,2) 1(8,2) 1(6,3) 1(4,4) 1(9,1).

3

Сообщение, сжатое LZ78 (словарь — 16 фраз), — (0,'A') (0,'F') (0,'X') (1,'F') (2,'X') (5,'A') (3,'A') (2,'F') (0,'A').

4

Сообщение, сжатое LZW (словарь — ASCII+ и 16 фраз), — O'A' O'F' O'X' (256) (257) (257) O'A' (258) O'F' O'F' O'A'.

Лабораторная работа № 9

Групповое кодирование

1. Порядок выполнения работы

1.1.Ознакомится с методическими указаниями, изложенными в п.3;
1.2.Выполнить задания (по указанию преподавателя)

2. Содержание отчета:

2.1.Тема и цель работы
2.2.Условия заданий
2.3.Подробное решение
2.4.Выводы по работе.

3.Общие сведения

Группой называют множество элементов, в котором определена одна основная операция и выполняются следующие аксиомы:

1. В результате применения операции к любым двум элементам группы образуется элемент этой же группы (требование замкнутости).
2. Для любых трех элементов группы A, B,C удовлетворяется равенство (A + B) +c = A + (B + C) (если основная операция сложение) и равенство A(BC) = (AB)C (если основная операция – умножение).
3. В любой группе существует однозначно определенный элемент, удовлетворяющий при всех значениях A из группы условию
   A + 0 = 0 + A = A (если основная операция – сложение) или усло
   вию A x 1 = 1 x A = A (если основная операция – умножение)
4. Всякий элемент A группы обладает элементом однозначно определенным уравнением A + (-A) = -A +A = 0 (если основная операция сложение) или уравнением A/A = 1 (если основная операция умножение).

ЗАДАЧА Построить групповой код объемом 15 слов, способный исправить единичные и обнаружить двойные ошибки.

Известно, что для исправления всех ошибок кратности S и одновременного обнаружения всех ошибок кратности R (R ≥ S) минимальное Хэмминговое расстояние можно выбирать из условия D ≥ R + S + 1.

В соответствии с этим соотношением код должен обладать минимальным Хэмминговым расстоянием, равным 4. Такой код можно построить в два этапа. Сначала строится код заданного объема, способный исправить единичные ошибки. Это код Хэмминга (7,4). Затем добавляем еще один проверочный разряд, который обеспечивает четность числа единиц в разрешенных комбинациях.

Таким образом, получается код (8,4). В процессе кодирования реализуются соотношения:

A1 = (A3 + A5 + A7)mod2

A2 = (A3 + A6 + A7)mod2

A4 = (A5 + A6 + A7)mod2

A1 = (A1 + A2 + A3 + A4 + A5 + A6 + A7)mod2

Обозначим синдром кода (7,4) через S1 , результат общей проверки

на четность – через S2 (S2 = $\sum_{i=1}$Ai) и пренебрегая возможностью

возникновения ошибок кратности 3 и выше, запишем алгоритм декодирования:

при S1 = 0 и S2 = 0 ошибок нет

при S1 = 0 и S2 = 1 ошибка в восьмом разряде

при S1 ≠ 0 и S2 = 0 двойная ошибка (коррекция блокируется, посылается запрос повторной передачи)

при S1 ≠ 0 и S2 = 0 одиночная ошибка (осуществляется ее исправление).

ЗАДАЧА Используя таблицу составить правила построения кода (8,2), исправляющего все одиночные и двойные ошибки.

Номер разрядов	Опознаватель	Номер разрядов	Опознаватель	Номер разрядов	Опознаватель
1	0001	7	0111	12	1100
2	0010	8	1000	13	1101
3	0011	9	1001	14	1110
4	0100	10	1010	15	1111
5	0101	11	1011	16	10000
6	0110				

Усекая таблицу на восьмом разряде, найдем следующие проверочные равенства:

(A1 + A5 + A8)mod2 = 0

(A2 + A5 + A8)mod2 = 0

(A3 + A5)mod2 = 0

(A4 + A5)mod2 = 0

(A6 + A8)mod2 = 0

(A7 + Aπ)mod2 = 0

Соответственно правила построения кода выразим соотношениями

A1 = (A5 + A8)mod2 (1)

A2 = (A5 + A8)mod2 (2)

A3 =A5 (3)

A4 = A5 (4)

A6 = A8 (5)

A7 = A8 (6)

Отметим, что для построения кода dmin =5, и, следовательно, он может использоваться для обнаружения ошибок кратности от 1 до 4.

ЗАДАЧА Построить систему разделенных проверок для декодирования информационных символов рассмотренного ранее группового кода (8,2).

Поскольку код рассчитан на исправление любых единичных и двойных ошибок, число проверочных равенств для определения каждого символа должно быть не менее 5. Подставив в равенство (1) и (2) значения A8, полученные из равенств (5) и (6), и записав их относительно A5 совместно с равенствами (3) и (4) и тривиальным равенством A5 = A5, получим следующую систему разделенных проверок для символа A5:

A5 = (A6 + A1)mod2

A5 = (A7 + A2)mod2

A5 = A3

A5 = A4

A5 = A5

Для символа A8 систему разделенных проверок строим аналогично

A8 = (A3 + A1)mod2

A8 = (A4 + A2)mod2

A8 = A6

A8 = A7

A8 = A8

Список заданий

1

Определить, являются ли группами следующие множества кодовых комбинаций

1) 0101, 0011, 1111, 0010
2) 00000, 11010, 11100, 0100
3) 0000, 0001, 0010, 0100

2

Построить групповой код объемом 13 слов, способный исправить единичные и обнаружить двойные ошибки.

3

Используя таблицу составить правила построения кода (13,2), исправляющего все одиночные и двойные ошибки.

Номер разрядов	Опознаватель	Номер разрядов	Опознаватель	Номер разрядов	Опознаватель
1	0001	7	0111	12	1100
2	0010	8	1000	13	1101
3	0011	9	1001	14	1110
4	0100	10	1010	15	1111
5	0101	11	1011	16	10000
6	0110				

4

Построить систему разделенных проверок для декодирования информационных символов рассмотренного ранее группового кода (13,2).

Для кодирующих матриц

$$E_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}, E_2 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}:$$

1. Построить соответственно (2, 5)-код и (3, 4)-код.
2. Найти основные характеристики полученных кодов: минимальное расстояние между словами кода; вероятность необнаружения ошибки; максимальную кратность ошибок, до которой включительно они все исправляются или обнаруживаются.
3. Построить таблицы декодирования.
4. Уточнить характеристики полученных кодов, при использовании их для исправления ошибок, т. е. найти вероятность правильной передачи и описать ошибки, исправляемые этими кодами.
5. Во что будут декодированы слова: 10001, 01110, 10101, 1001, 0110, 1101?

Лабораторная работа № 10

Полиноминальные и циклические коды

1. Порядок выполнения работы

1.Ознакомится с методическими указаниями, изложенными в п.4;
2.Выполнить задания (по указанию преподавателя)
3.Подготовить ответы на контрольные вопросы

2. Содержание отчета:

1.Тема и цель работы
2.Условия заданий
3.Подробное решение.
4.Выводы по работе.

3. Контрольные вопросы

1. Каковы основные свойства циклических кодов?
2. Какие известны способы построения циклических кодов?
3. В чем заключается методика декодирования циклических кодов?

4. Общие сведения

Полиномиальные коды

Представление кодового слова (***n,k***)-кода в виде последовательности $U = (U_0, U_1, ..., U_{n-1})$ длиной ***n*** символов или их задание с помощью системы проверочных уравнений и порождающей матрицы не является единственно возможным. Еще один удобный и широко используемый способ представления того же кодового слова состоит в том, что элементы $U_0, U_1, ..., U_{n-1}$ являются коэффициентами многочлена от ***X***, то есть

$$U(x) = f(x) = U_0 + U_1 \cdot X + U_2 \cdot X^2 + ... + U_{n-1} \cdot X^{n-1} \quad . \qquad (1)$$

Используя это представление, можно определить полиномиальный код как *множество всех многочленов степени, не большей* ***n -1****, содержащих в качестве общего множителя некоторый фиксированный многочлен* ***g(x).***

Многочлен ***g(x)*** *называется порождающим многочленом кода*.

Представление кодовых слов в такой форме позволяет свести действия над комбинациями символов к действию над полиномами.

Определим действия над полиномами в поле двоичных символов ***GF(2)***.

Суммой двух полиномов ***f(x)*** и ***g(x)*** из ***GF(2)*** называется полином из ***GF(2),*** определяемый следующим образом:

$$f(x) + g(x) = \sum_{i=0}^{n-1} \left(f_i + g_i \right) \cdot x^i \quad . \qquad (2)$$

Другими словами, сложению двоичных полиномов соответствует сложение по ***mod2*** коэффициентов при одинаковых степенях ***x.***

Например:

$$\begin{array}{r} X^3+X^2+0 \cdot X+1 \\ \hline X^3+X^2+X+0 \end{array} \quad \begin{array}{l} X+1 \\ = \end{array} \quad X^3+X^2+X \ , \qquad (3)$$

Произведением двух полиномов из ***GF(2)*** называется полином из ***GF(2),*** определяемый следующим образом :

$$\begin{array}{r} X^3 + X^2 + 0 \cdot X + 1 \\ X^2 + X + 1 \\ \hline X^3 + 0 + X + 0 = X^3 + X. \end{array} \qquad (4)$$

$$f(x) \cdot g(x) = \sum_{i=0}^{n-1} \left(\sum_{j=0}^{i} f_j \cdot g_{i-j} \right) \cdot x^i, \qquad (5)$$

то есть произведение получается по обычному правилу перемножения степенных функций, однако получаемые коэффициенты при данной степени X складываются по модулю 2.

Например:

$$\begin{array}{r} X^3 + X^2 + 0 + 1 \\ X + 1 \\ \hline X^3 + X^2 + 0 + 1 \\ X^4 + X^3 + 0 + X \\ \hline X^4 + 0 + X^2 + X + 1 = X^4 + X^2 + X + 1, \end{array} \qquad (6)$$

$$\begin{array}{r} X^3 + X^2 + 0 + 1 \\ X^2 + X \\ \hline X^4 + X^3 + 0 + X \\ X^5 + X^4 + 0 + X^2 \\ \hline X^5 + 0 + X^3 + X^2 + X = X^5 + X^3 + X^2 + X. \end{array} \qquad (7)$$

Наконец, можно сформулировать теорему о делении полиномов: *для каждой пары полиномов* $C(x)$ *и* $d(x)$, $d(x) \neq 0$ *существует единственная пара полиномов* $q(x)$ — *частное и* $\rho(x)$ — *остаток, такие, что*

$$C(x) = q(x) \cdot d(x) + \rho(x), \qquad (8)$$

где степень остатка $\rho(x)$ меньше степени делителя $d(x)$.

Иными словами, *деление полиномов производится по правилам деления степенных функций, при этом операция вычитания заменяется суммированием по* ***mod2***.

Например:

$$\begin{array}{l|l} X^4+0+X^2+X+1 & X+1 \\ \underline{X^4+X^3} & \overline{X^3+X^2+1} \\ \quad X^3+X^2+X+1 & \\ \quad \underline{X^3+X^2} & \\ \qquad\quad X+1 & \\ \qquad\quad \underline{X+1} & \\ \qquad\quad 0 \longleftarrow \text{остаток } \rho(x). & \end{array} \qquad (9)$$

Еще раз напомним, что при сложении по ***mod2*** сумма двух единиц (то есть двух элементов полинома с одинаковыми степенями) будет равна нулю, а не привычным в десятичной системе счисления двум. И, кроме этого, операции вычитания и сложения по ***mod2*** совпадают.

Циклические коды

Частным и наиболее широко распространенным классом полиномиальных кодов являются *циклические коды*.

Линейный (***n,k***)-код *называется циклическим, если в результате циклического сдвига кодового слова получается другое кодовое слово данного кода.* Другими словами, если $U = (U_0, U_1, \ldots U_{n-1})$ является кодовым словом, то и $V = (U_{n-1}, U_0, U_1, \ldots U_{n-2})$, полученное циклическим сдвигом U, является кодовым словом данного кода.

Циклические коды привлекательны по двум причинам.

Во-первых, операции кодирования и вычисления синдрома для них выполняются очень просто с использованием сдвиговых регистров.

Во-вторых, этим кодам присуща алгебраическая структура, и можно найти более простые и эффективные способы их декодирования.

Основные свойства циклических кодов:

*1. В циклическом (**n,k**)-коде каждый ненулевой полином должен иметь степень, по крайней мере (**n-k**), но не более **n-1**.*

*2. Существует только один кодовый полином **g(x)** степени **(n-k)** вида*

$$g(x)=1+g_1 \cdot x+g_2 \cdot x^2+\ldots+g_{n-k-1} \cdot x^{n-k-1}+x^{n-k}, \qquad (10)$$

называемый порождающим полиномом кода.

*3. Каждый кодовый полином **U(x)** является кратным **g(x),** то есть*

$$U(x)= m(x) \cdot g(x). \qquad (11)$$

Кодирование с использованием циклических кодов

Предположим, надо закодировать некоторую информационную последовательность

$$m = (m_0, m_1, m_2 \ldots m_{k-1}). \qquad (12)$$

Соответствующий ей полином выглядит следующим образом:

$$m(x)= m_0 + m_1 \cdot x + m_2 \cdot x^2 +...+m_{k-1} \cdot x^{k-1}. \quad (13)$$

Умножив $m(x)$ на x^{n-k}:

$$x^{n-k} \cdot m(x)= m_0 \cdot x^{n-k} + m_1 \cdot x^{n-k+1} +...+m_{k-1} \cdot x^{n-1}, \quad (14)$$

получим полином степени $n-1$ или меньшей.

Воспользовавшись теоремой о делении полиномов, можно записать

$$x^{n-k} \cdot m(x) = q(x) \cdot g(x) + \rho(x), \quad (15)$$

где $q(x)$ и $\rho(x)$ — частное и остаток от деления полинома $x^{n-k} \cdot m(x)$ на порождающий полином $g(x)$.

Поскольку степень $g(x)$ равна $(n-k)$, то степень $\rho(x)$ должна быть $(n-k-1)$ или меньше, а сам полином $\rho(x)$ будет иметь вид

$$\rho(x)= \rho_0 + \rho_1 \cdot x + \rho_2 \cdot x^2 +...+ \rho_{n-k-1} \cdot x^{n-k-1}. \quad (16)$$

С учетом правил арифметики в $GF(2)$ данное выражение можно переписать следующим образом:

$$\rho(x) + x^{n-k} \cdot m(x) = q(x) \cdot g(x), \quad (17)$$

откуда видно, что полином $\rho(x)+x^{n-k} \cdot m(x)$ является кратным $g(x)$ и имеет степень $n-1$ или меньшую. Следовательно, полином $\rho(x)+x^{n-k} \cdot m(x)$ - это кодовый полином, соответствующий кодируемой информационной последовательности $m(x)$.

Раскрыв последнее выражение, получим

$$\rho(x)+m(x) \cdot x^{n-k} =\rho_0 +\rho_1 x +\rho_2 x^2 ..+\rho_{n-k-1} x^{n-k-1} + m_0 x^{n-k} + m_1 \cdot x^{n-k+1} +..+ m_{k-1} x^{n-1},$$

что соответствует кодовому слову

$U =$	$(\rho_0, \rho_1 ... \rho_{n-k-1}$,	$m_0, m_1 ... m_{k-1})$,
	проверочные символы	*информационные символы*

Таким образом, кодовое слово состоит из неизменной информационной части m, перед которой расположено ($n-k$) проверочных символов. Проверочные символы являются коэффициентами полинома $\rho(x)$, то есть остатка от деления $m(x) \cdot x^{n-k}$ на порождающий полином $g(x)$.

Чтобы полученный результат был понятнее, напомним, что умножению некоторого двоичного полинома на x^{n-k} соответствует сдвиг двоичной последовательности $m = (m_0, m_1 ... m_{k-1})$ на $n-k$ разрядов вправо.

Рассмотрим пример. С использованием кода, задаваемого порождающим полиномом $g(x) = 1 + x + x^3$, закодируем произвольную последовательность, например $m = (0111)$.

Последовательности $m =(0111)$ соответствует полином $m (x)=x+ x^2 + x^3$.

Умножим $m(x)$ на x^{n-k} :

$$m(x) \cdot x^{n-k} = m(x) \cdot x^3 = (x + x^2 + x^3) \cdot x^3 = x^4 + x^5 + x^6. \qquad (18)$$

Разделим $m(x) \cdot x^{n-k}$ на порождающий полином $g(x)$:

$$\begin{array}{l|l} X^6 + X^5 + X^4 & X^3 + X + 1 \\ \underline{X^6 + 0 + X^4 + X^3} & X^3 + X^2 = g(x) \\ X^5 + 0 + X^3 & \\ \underline{X^5 + 0 + X^3 + X^2} & \\ X^2 = \rho(x). & \end{array} \qquad (19)$$

Таким образом, кодовый полином, соответствующий информационной последовательности $m = (0111)$, будет иметь следующий вид:

$$U(x) = 0 * X^0 + 0 * X^1 + 1 * X^2 + 0 * X^3 + 1 * X^4 + 1 * X^5 + 1 * X^6, \qquad (20)$$

а соответствующее кодовое слово $U = (0010111)$.

Итак, циклический (n,k)-код k-разрядной информационной последовательности $m = (m_0, m_1 \ldots m_{k-1})$ получают следующим образом:

- информационную последовательность m умножают на x^{n-k}, то есть сдвигают вправо на $n-k$ разрядов;

- полином полученной последовательности делят на порождающий полином кода $g(x)$;

- полученный остаток от деления $m(x) \cdot x^{n-k}$ на $g(x)$ прибавляют к $m(x) \cdot x^{n-k}$, то есть записывают в младших $n-k$ разрядах кода.

Еще одним важным свойством циклического (n,k)-кода, вытекающим из теоремы о существовании циклических кодов, является то, что его порождающий полином делит без остатка двучлен $x^n + 1$, то есть

$$x^n + 1 = g(x) \cdot h(x) + 0. \qquad (21)$$

Полином $h(x)$ — частное от деления $x^n + 1$ на $g(x)$ - называется проверочным полиномом.

Поскольку $h(x)$ однозначно связан с $g(x)$, он также определяет код. Следовательно, с помощью проверочного полинома $h(x)$ тоже можно производить кодирование.

Вычисление синдрома и исправление ошибок в циклических кодах

Вычисление синдрома для циклических кодов является довольно простой процедурой.

Пусть $U(x)$ и $r(x)$ - полиномы, соответствующие переданному кодовому слову и принятой последовательности.

Разделив $r(x)$ на $g(x)$, получим

$$r(x) = q(x) \cdot g(x) + s(x), \qquad (22)$$

где - $q(x)$ — частное от деления, $s(x)$ — остаток от деления.

Если $r(x)$ является кодовым полиномом, то он делится на $g(x)$ без остатка, то есть $s(x) = 0$.

Следовательно, ***s(x) ≠ 0*** *является условием наличия ошибки в принятой последовательности, то есть синдромом принятой последовательности.*

Синдром ***s(x)*** имеет в общем случае вид

$$S(x) = S_0 + S_1 \cdot x + \ldots + S_{n-k-1} \cdot x^{n-k-1} . \qquad (23)$$

При наличии в принятой последовательности ***r*** хотя бы одной ошибки вектор синдрома ***S*** будет иметь, по крайней мере, один нулевой элемент, при этом факт наличия ошибки легко обнаружить

Покажем, что синдромный многочлен ***S(x)*** однозначно связан с многочленом ошибки ***e(x),*** а значит, с его помощью можно не только обнаруживать, но и локализовать ошибку в принятой последовательности.

Пусть

$$e(x) = e_0 + e_1 \cdot x + e_2 \cdot x^2 + \ldots + e_{n-1} \cdot x^{n-1} \qquad (24)$$

— полином вектора ошибки.

Тогда полином принятой последовательности

$$r(x) = U(x) + e(x). \qquad (25)$$

Прибавим в этом выражении слева и справа ***U(x),*** а также учтем, что

$$r(x) = q(x) \cdot g(x) + S(x),\ U(x) = m(x) \cdot g(x), \qquad (26)$$

тогда

$$e(x) = \left[m(x) + q(x)\right] \cdot g(x) + S(x) = f(x) \cdot g(x) \cdot S(x), \qquad (27)$$

то есть *синдромный полином* ***S(x)*** *есть остаток от деления полинома ошибки* ***e(x)*** *на порождающий полином* ***g(x).***

Отсюда следует, что по синдрому ***S(x)*** можно однозначно определить вектор ошибки ***e(x),*** а следовательно, исправить эту ошибку.

Список заданий

1

По кодирующему многочлену $x^7 + x^5 + x + 1$ построить полиномиальные коды для двоичных сообщений 0100, 10001101, 11110.

2

Принадлежат ли коду Голея кодовые слова 100001010111110100111111 и 110001110111100100111111?

3

Получено сообщение циклическим кодом $F^*(x) = x^6 + x^4 + x^3 + x^2$. Проверить декодированием наличие ошибок в принятой комбинации, если образующий полином $P(x) = x^3 + x^2 + 1$.

4

Получена комбинация $F^*(x) = x^6 + x^4 + x^2 + 1$, закодированная циклическим кодом. Образующий полином $P(x) = x^3 + x^2 + 1$. Проверить наличие ошибок в кодовой комбинации.

Лабораторная работа № 11

Коды Боуза- Чоудхури – Хоккенгема

1. Порядок выполнения работы

1.1.Ознакомится с методическими указаниями, изложенными в п.3;

1.2.Выполнить задания (по указанию преподавателя)

2. Содержание отчета:

2.1.Тема и цель работы

2.2.Условия заданий

2.3.Подробное решение

2.4.Выводы по работе.

3.Общие сведения

Французский ученый А. Хоквингем (1959 г.) и американцы Р. К. Боуз и Д. К. Рой-Чоудхури (1960 г.) нашли большой класс кодов, обеспечивающий произвольное минимальное кодовое расстояние dmin ≥ 5. Они получили название БЧХ (Боуза-Чоудхури-Хоквингема). Порождающие полиномы для таких кодов в зависимости от предъявляемых к ним требований, можно найти в таблице

k	n	m	s	d_{min}	Порождающий полином	
					Символическая запись	Запись в виде полинома
3	7	4	1	3	13	1011
4	15	11	1	3	23	10011
8	15	7	1	3	721	111010001
10	15	5	3	7	2467	10100110111
5	31	26	1	3	45	100101
10	31	21	2	5	3551	11101101001
15	31	16	3	7	107657	1000111110101111
25	31	11	5	11	5423325	10110001001101101010 1

Где n — общее число элементов, m — число информационных элементов, k — число избыточных элементов (n = m + k).

Процедура построения кода БЧХ по заданным M и dmin:

1. по d_{min} найти значение, при котором обеспечивается необходимое число информационных элементов m при минимальной избыточности k_{min};
2. найти в таблице соответствующий порождающий полином;
3. если d_{min} четное, умножить найденный полином на (x + 1);
4. если $m_{табл} >> m_{задан}$, то можно перейти к укороченному циклическому коду, вычеркивая в порождающей матрице исходного кода с параметрами $m_{табл}$, k_{min} ($m_{табл} - m_{задан}$) столбцов слева и столько же строк сверху.

Методика построения БЧХ. Методика построения кодов БЧХ аналогична общей методике построения ц. к. и отличается в основном выбором образующего многочлена.
Последовательность построения P(x) для кодов БЧХ тоже, что и для обычных ц.к., однако образующий полином является произведением t неприводимых полиномов,

$$G(x) = M_1(x)*M_2(x)\ldots M_t(x),$$ где t – кратность ошибки.

Методика выбора (построения) образующего полинома основана на понятии корня двоичного многочлена и теоремы БЧХ.

Понятие корня двоичного многочлена.

1. Элемент α является корнем двоичного полинома f(x), если $f(\alpha)=0$.
2. Количество корней многочлена равно степени полинома.

Если $f(x)=q_0+q_1x+q_2x^2+\ldots+q_nx^n$;
$q_n\neq0$; тогда $\alpha_1, \alpha_2, \alpha_3 \ldots \alpha_n$ при которых $f(\alpha_i)=0$, i=1,2,n.

Пример 1: $f(x)=(x+1)$ – количество корней – 1; $\alpha_1=1$;
$f(\alpha_1)= f(1)=0$.

Пример 2: Пусть требуется определить все корни бинома $x^{15}+1$.

1. Количество корней $\alpha_1, \alpha_2, \alpha_3 \ldots \alpha_{15}$.
2. Представление $x^{15}+1$ в виде произведения неприводимых сомножителей:

$$x^{15}+1 = \underset{f_1}{(x+1)}\underset{f_2}{(x^2+x+1)}\underset{f_3}{(x^4+x+1)}\underset{f_4}{(x^4+x^3+1)}\underset{f_5}{(x^4+x^3+x^2+x+1)}.$$

Корни полинома получены Питерсеном и сведены в специальную таблицу.

Фрагмент таблицы:

Бином	**Неприводимые многочлены**	**Корни**
$x^{15}+1$	$f_1(x)$: $x+1$	α_1
	$f_2(x)$: x^2+x+1	α_6, α_{11}
	$f_3(x)$: x^4+x+1 $f_4(x)$: x^4+x^3+1 $f_5(x)$: $x^4+x^3+x^2+x+1$	$\alpha_2, \alpha_3, \alpha_5, \alpha_9$ $\alpha_8, \alpha_{12}, \alpha_{14}, \alpha_{15}$ $\alpha_4, \alpha_7, \alpha_{10}, \alpha_{13}$

Т.е. каждый корень α_i является корнем $f_j(x)$ и корнем порождающего бинома $x^{15}+1$ и имеет свой порядковый номер.

Для построения полинома кодов БЧХ используется теорема БЧХ: (без доказательств)

Если образующий полином содержит непрерывную цепь из m корней, то данный порождающий полином обладает корректирующими свойствами кода с $d_{min}=m+1$.

При этом ц.к., исправляющие одиночные ошибки являются частным случаем (m=2) из общей теоремы БЧХ.

Пример: 1). Если взять в качестве порождающего

$f_3(x) \longrightarrow \alpha_2, \alpha_3, \alpha_5, \alpha_9 \longrightarrow m=2$

$f_4(x): \longrightarrow \alpha_8, \alpha_{12}, \alpha_{14}, \alpha_{15} \longrightarrow m=2$

$\Big\}\ d_{min} \geq 3.$

2). $F=f_3(x)*f_5(x) \longrightarrow \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_7, \alpha_9, \alpha_{10}, \alpha_{13} \longrightarrow m=4 \qquad d_{min} \geq 5.$

Поскольку (как уже было отмечено выше) методика этапов кодирования и декодирования кодов БЧХ отличается от кодов, исправляющих одиночные ошибки, только выбором образующего многочлена, рассмотрим методику выбора P(x) для БЧХ.

Методика выбора порождающего полинома для кодов БЧХ.

1. Определение количества информационных разрядов:
   $k = [\log_2 N]$.
2. Определение количества проверочных разрядов:
   $n = k + r = k + t * h \le 2^h - 1$;
   t – кратность ошибки.
   Длины кодовой комбинации $\underline{n = k + r}$ и степени бинома $\underline{x^n + 1}$.
3. Разложение (представление) $\underline{x^n + 1}$ в виде произведения неприводимых сомножителей (по таблице Питерсона).
4. Выбор неприводимых многочленов в качестве сомножителей образующего полинома т.о., чтобы
набор корней содержал непрерывную цепь корней длиной не менее чем $\underline{m=d_{min}-1=2t}$;
5. Представление в виде произведения неприводимых сомножителей.
   Этап декодирования аналогичен ц.к. При этом $l>1$.

Пример.

Построить ц.к. для передачи различных символов, исправляющий одну или две ошибки:

1. $k = [\log_2 N] = [\log_2 100] = 7$, $\underline{d_{min} \ge 5}$.
2. $n = k + r = 7 + 2*h \le 2^n - 1$; $h = 4$; $r = l*h = 2*4 = 8 \Rightarrow \underline{n = 15}$.
3.

$$\underset{\alpha_1}{f_1} \quad \underset{\alpha_6,\ \alpha_{11}}{f_2} \quad \underset{\alpha_2,\ \alpha_3,\ \alpha_5,\ \alpha_9}{f_3} \quad \underset{\alpha_8,\ \alpha_{12},\ \alpha_{14},\ \alpha_{15}}{f_4} \quad \underset{\alpha_4,\ \alpha_7,\ \alpha_{10},\ \alpha_{13}}{f_5}$$

$$x^{15}+1 = (x+1)(x^2+x+1)(x^4+x+1)(x^4+x^3+1)(x^4+x^3+x^2+x+1).$$

4. В качестве $F(x)$:

$F_1(x) = f_3 * f_5 = (x^4+x+1)(x^4+x^3+x^2+x+1)$
$\underline{\alpha_2,\ \alpha_3,\ \alpha_4,\ \alpha_5},\ \alpha_7,\ \alpha_9,\ \alpha_{10},\ \alpha_{13}$

$F_2(x) = f_4 * f_5 = (x^4+x^3+1)(x^4+x^3+x^2+x+1)$
$\alpha_4,\ \alpha_7,\ \alpha_8,\ \alpha_{10},\ \underline{\alpha_{12},\ \alpha_{13},\ \alpha_{14},\ \alpha_{15}}$

$m=4$

При выборе в качестве порождающего $F_1(x)$ или $F_2(x)$ – корректирующие возможности полученного ц.к. будут равны.

5. Степень образующего многочлена $F(x) = 8$;
   $F(x) = F_1(x) = (x^4+x+1)(x^4+x^3+x^2+x+1) = x^8 + x^7 + x^6 + x^4 + 1$.

$$C_{15,7} = \begin{array}{|c|c|} \multicolumn{1}{c}{\text{7 разр.}} & \multicolumn{1}{c}{\text{8 разр.}} \\ \hline 0000001 & 11010001 \\ 0000010 & 01110011 \\ 0000100 & 11100110 \\ 0001000 & 00011101 \\ 0010000 & 00111010 \\ 0100000 & 01110100 \\ 1000000 & 11101000 \\ \hline \end{array}$$

$W(E_i) \ge d_{min} = 5$;
$\underline{W(R(x_2) \ge 4}$.

10000,00000,00000 | 111010001

```
        11101,0001 ↓
R1(x)    1101,00010
         1110,10001 ↓
R2(x)     011,10011,0
R3(x)      11,10011,00
           11,10100,01
R4(x)       0,00111,01
R5(x)       0,01110,10
R6(x)       0,11101,00
R7(x)       1,11010,00
```

Необходимо закодировать и передать: $\vec{H}$=1000011

$\underset{k}{\underline{1100001}}\ \underset{r}{\underline{01001101}} \Rightarrow \tilde{E}(x) = \underline{1101011|01001101}$,

1). $R_1(x) = \underline{01101110} \rightarrow \underline{W(R_1(x)) > 2.}$

2). Циклический сдвиг на 4 позиции

$E_2(x) = 0110100|11011101$

$R_2(x) = 01110101 \rightarrow W(R_2(x)) > 2;$

3). Ц.к. влево на 2 позиции

$\tilde{E}_3(x) = 1010011|01110101;$

$R_3(x) = 00000101 \rightarrow W(R_3(x)) = 2;$

$\tilde{E}_3(x) = \tilde{E}_3(x) + R_3(x).$

4). Циклический сдвиг вправо на 4 + 2 = 6 позиций.

После этого получаем направленную кодовую комбинацию.

2. Порядок выполнения работы

2.1.Ознакомится с методическими указаниями, изложенными в п.1;

2.2.Выполнить задания.

3. Содержание отчета:

3.1.Тема и цель работы

3.2.Условия заданий

3.3.Подробное решение

3.5.Выводы по работе.

Список заданий

1

Найти кодирующий многочлен БЧХ-кода $g(x)$ с длиной кодовых слов 15 и минимальным расстоянием между кодовыми словами 7. Использовать примитивный многочлен $m_1(x) = 1 + x + x^4$ с корнем α. Проверить, будут ли α^3 и α^5 корнями соответственно многочленов

$m_3(x) = 1 + x + x^2 + x^3 + x^4$ и

$m_5(x) = 1 + x + x^2$.

Лабораторная работа № 12

Циклические избыточные коды

1. Порядок выполнения работы

1.1.Ознакомится с методическими указаниями, изложенными в п.3;
1.2.Выполнить задания (по указанию преподавателя)

2. Содержание отчета:

2.1.Тема и цель работы
2.2.Условия заданий
2.3.Подробное решение
2.4.Выводы по работе.

3.Общие сведения

Циклический избыточный код (Cyclical Redundancy Check — CRC) имеет фиксированную длину и используется для обнаружения ошибок. Наибольшее распространения получили коды CRC-16 и CRC-32, имеющие длину 16 и 32 бита соответственно. Код CRC строится по исходному сообщению произвольной длины, т.е. этот код не является блочным в строгом смысле этого слова. Но при каждом конкретном применении этот код — блочный, *(m, m* + 16)-код для CRC-16 или *(m, m* + 32)-код для CRC-32.

Вычисление значения кода CRC происходит посредством деления многочлена, соответствующего исходному сообщению (полином-сообщение), на фиксированный многочлен (полином-генератор). Остаток от такого деления и есть код CRC, соответствующий исходному сообщению. Для кода CRC-16 полином-генератор имеет степень 16, а для CRC-32 — 32. Полиномы-генераторы подбираются специальным образом и для кодов CRC-16/32 стандартизированы Международным консультативным комитетом по телеграфной и телефонной связи (CCITT). Для CRC-16, например, стандартным является полином-генератор $x^{16} + x^{12} + x^5 + 1$.

Пример построения CRC-4 кода для сообщения 11010111, используя полином-генератор $x^4+x^3+x^2+1$. Исходному сообщению соответствует полином $x^7 + x^6 + x^4 + x^2 + x + 1$, т. е. нумерация битов здесь начинается справа.

$$\begin{array}{l|l} x^7+x^6+x^4+x^2+x+1 & x^4+x^3+x^2+1 \\ \underline{x^7+x^6+x^5+x^3} & \hline x^3+x \\ x^5+x^4+x^3+x^2+x+1 & \\ \underline{x^5+x^4+x^3+x} & \\ x^2+1 & \end{array}$$

Полиному $x^2 + 1$ соответствуют биты 0101 — это и есть CRC-4 код.

Существуют быстрые алгоритмы для расчета CRC-кодов, использующие специальные таблицы, а не деление многочленов с остатком.

CRC-коды способны обнаруживать одиночную ошибку в любой позиции и, кроме того, многочисленные комбинации кратных ошибок, расположенных близко друг от друга. При реальной передаче или хранении информации ошибки обычно группируются на некотором участке, а не распределяются равномерно по всей длине данных. Таким образом, хотя для идеального случая двоичного

симметричного канала CRC-коды не имеют никаких теоретических преимуществ по сравнению, например, с простыми контрольными суммами, для реальных систем эти коды являются очень полезными.

Коды CRC используются очень широко: модемами, телекоммуникационными программами, программами архивации и проверки целостности данных и многими другими программными и аппаратными компонентами вычислительных систем.

Список заданий

1

Построить CRC-4 код для сообщений 10000000 и 101111001, используя полином-генератор $x^4 + 1$.