ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Б1.Б.11 Теория информации

Направление подготовки (специальность) 10.03.01 Информационная безопасность

Профиль образовательной программы Безопасность автоматизированных систем

Форма обучения очная

СОДЕРЖАНИЕ

2.2 Практическое занятие № 3-4 Виды информации. Хранение информации. Свойства информации. Емкость канала связи. Способы измерения информации

2.3 Практическое занятие № 5-6 Понятие энтропии и семантическая информация. Смысл энтропии Шеннона. Понятие семантической информации. Семантическая мера информации. Семантическая информация в системах

2.4 Практическое занятие № 7-8 Необратимое сжатие. Обратимое сжатие. Кодирование Хаффмана. Упорядоченное дерево Хаффмана. Избыточность сообщений. Код Фано

2.5 Практическое занятие № 9-10 Словарно-ориентированные алгоритмы сжатия. Метод Лемпела-Зива LZ77. Метод LZSS. Метод LZ78. Метод LZW. LZ-алгоритмы распаковки данных

2.6 Практическое занятие № 11-12 Программы сжатия. Особенности программ-архиваторов. Сжатие информации с потерями

2.7 Практическое занятие № 13-14 Алгоритм арифметического кодирования. Адаптивное арифметическое кодирование

2.8 Практическое занятие № 15-16 Помехозащитное кодирование. Матричное кодирование и групповые коды. Информационный канал. Помехозащитное кодирование в двоичном симметричном канале. Матричное кодирование. Групповые коды. Совершенные и квазисовершенные коды

2.9 Практическое занятие № 17-18 Полиномиальные коды. Принцип построения полиномиальных кодов. Линейные коды. Циклические коды. Исправление ошибок при построении кодов

2.10 Практическое занятие № 19-21 Криптография. Криптосистема без передачи ключей. Криптосистема с открытым ключом. Электронная подпись. Стандарт шифрования данных. Кодировка текстовой информации

1. КОНСПЕКТ ЛЕКЦИЙ

1. 1 Лекция №1 (2 часа).

Тема: «Предмет теории информации. Хранение, свойства, и измерение информации»

1.1.1 Вопросы лекции:

1.Понятие информации.

2.Передача информации.

3.Измерение информации.

4. Формальное представление информации.

5. Виды информации.

6. Хранение информации.

7. Свойство информации.

8. Емкость канала связи.

9. Способы измерения информации.

1.1.2 Краткое содержание вопросов:

1. Понятие информации.

Слово «информация» происходит от латинского слова informatio, что в переводе означает сведение, разъяснение, ознакомление. Понятие «информация» является базовым в курсе информатики, однако невозможно дать его определение через другие, более «простые» понятия.

Понятие «информация» используется в различных науках, при этом в каждой науке понятие «информация» связано с различными системами понятий. Информация в биологии: Биология изучает живую природу и понятие «информация» связывается с целесообразным поведением живых организмов. В живых организмах информация передается и храниться с помощью объектов различной физической природы (состояние ДНК), которые рассматриваются как знаки биологических алфавитов. Генетическая информация передается по наследству и хранится во всех клетках живых организмов. Философский подход: Информация – это взаимодействие, отражение, познание. Кибернетический подход: Информация – это характеристики управляющего сигнала, передаваемого по линии связи.

2. Передача информации

Обмен информацией производится по каналам передачи информации. Каналы передачи информации могут использовать различные физические принципы. Так, при непосредственном общении людей информация передается с помощью звуковых волн, а при разговоре по телефону - с помощью электрических сигналов, которые распространяются по линиям связи. Компьютеры могут обмениваться информацией с использованием каналов связи различной физической природы: кабельных, оптоволоконных, радиоканалов и др.

Общая схема передачи информации включает в себя отправителя информации, канал передачи информации и получателя информации. Если производится двусторонний обмен информацией, то отправитель и получатель информации могут меняться ролями.

Основной характеристикой каналов передачи информации является их пропускная способность (скорость передачи информации). Пропускная способность канала равна количеству информации, которое может передаваться по нему в единицу времени.

3. Измерение информации.

Обычно пропускная способность измеряется в битах в секунду (бит/с) и кратных единицах Кбит/с и Мбит/с. Однако иногда в качестве единицы измерения используется байт в секунду (байт/с) и кратные ему единицы Кбайт/с и Мбайт/с.

Соотношения между единицами пропускной способности канала передачи информации такие же, как между единицами измерения количества информации:

1 байт/с = 23 бит/с = 8 бит/с;

1 Кбит/с = 210 бит/с = 1024 бит/с;

1 Мбит/с = 210 Кбит/с = 1024 Кбит/с;

1 Гбит/с = 210 Мбит/с = 1024 Мбит/с.

4. Формальное представление знаний.

Представление знаний — вопрос, возникающий в когнитологии (науке о мышлении), в информатике и в исследованиях искусственного интеллекта. В когнитологии он связан с тем, как люди хранят и обрабатывают информацию. В информатике — с подбором представления конкретных и обобщённых знаний, сведений и фактов для накопления и обработки информации в ЭВМ. Главная задача в искусственном интеллекте (ИИ) — научиться хранить знания таким образом, чтобы программы могли осмысленно обрабатывать их и достигнуть тем подобия человеческого интеллекта.

Под термином «представление знаний» чаще всего подразумеваются способы представления знаний, ориентированные на автоматическую обработку современными компьютерами, и, в частности, представления, состоящие из явных объектов ('класс всех слонов', 'Клайд — индивид') и из суждений или утверждений о них ('Клайд — слон', 'все слоны серые'). Представление знаний в подобной явной форме позволяет компьютерам делать дедуктивные выводы из ранее сохранённого знания ('Клайд — серый').

5. Виды информации.

Основные виды информации по ее форме представления, способам ее кодирования и хранения, что имеет наибольшее значение для информатики, это:

графическая или изобразительная — первый вид, для которого был реализован способ хранения информации об окружающем мире в виде наскальных рисунков, а позднее в виде картин, фотографий, схем, чертежей на бумаге, холсте, мраморе и др. материалах, изображающих картины реального мира;

звуковая (акустическая) — мир вокруг нас полон звуков и задача их хранения и тиражирования была решена с изобретением звукозаписывающих устройств в 1877 г; ее разновидностью является музыкальная информация — для этого вида был изобретен способ кодирования с использованием специальных символов, что делает возможным хранение ее аналогично графической информации;

текстовая — способ кодирования речи человека специальными символами — буквами, причем разные народы имеют разные языки и используют различные наборы букв для отображения речи; особенно большое значение этот способ приобрел после изобретения бумаги и книгопечатания;

числовая — количественная мера объектов и их свойств в окружающем мире; особенно большое значение приобрела с развитием торговли, экономики и денежного обмена; аналогично текстовой информации для ее отображения используется метод кодирования специальными символами — цифрами, причем системы кодирования (счисления) могут быть разными;

видеоинформация — способ сохранения «живых» картин окружающего мира, появившийся с изобретением кино.

Существуют также виды информации, для которых до сих пор не изобретено способов их кодирования и хранения — это тактильная информация, передаваемая ощущениями, органолептическая, передаваемая запахами и вкусами и др.

6.Хранение информации.

Хранение информации – это ее запись во вспомогательные запоминающие устройства на различных носителях для последующего использования.

Хранение является одной из основных операций, осуществляемых над информацией, и главным способом обеспечения ее доступности в течение определенного промежутка времени.

Основное содержание процесса хранения и накопления информации состоит в создании, записи, пополнении и поддержании информационных массивов и баз данных в активном состоянии.

В результате реализации такого алгоритма, документ, независимо от формы представления, поступивший в информационную систему, подвергается обработке и после этого отправляется в хранилище (базу данных), где он помещается на соответствующую "полку" в зависимости от принятой системы хранения. Результаты обработки передаются в каталог.

Этап хранения информации может быть представлен на следующих уровнях:

- внешнем;
- концептуальном, (логическом);
- внутреннем;
- физическом.

Внешний уровень отражает содержательность информации и представляет способы (виды) представления данных пользователю в ходе реализации их хранения

Концептуальный уровень определяет порядок организации информационных массивов и способы хранения информации (файлы, массивы, распределенное хранение, сосредоточенное и др.).

Внутренний уровень представляет организацию хранения информационных массивов в системе ее обработки и определяется разработчиком.

Физический уровень хранения означает реализацию хранения информации на конкретных физических носителях.

Хранение и поиск информации являются не только операциями над ней, но и предполагают использование методов осуществления этих операций. Информация запоминается так, чтобы ее можно было отыскать для дальнейшего использования. Возможность поиска закладывается во время организации процесса запоминания. Для этого используют методы маркирования запоминаемой информации, обеспечивающие

поиск и последующий доступ к ней. Эти методы применяются для работы с файлами, графическими базами данных и т.д.

7. Свойства информации.

Информация обладает следующими свойствами:

достоверность

полнота

точность

ценность

своевременность

понятность

доступность

краткость и т. д.

Информация достоверна, если она отражает истинное положение дел. Недостоверная информация может привести к неправильному пониманию или принятию неправильных решений. Достоверная информация со временем может стать недостоверной, так как она обладает свойством устаревать, т. е. переста-ет отражать истинное положение дел.

Информация полна, если ее достаточно для понимания и принятия ре-шений. Как неполная, так и избыточная информация сдерживает принятие ре-шений или может повлечь ошибки.

Точность информации определяется степенью ее близости к реальному состоянию объекта, процесса, явления и т. п.

Ценность информации зависит от того, насколько она важна для реше-ния задачи, а также от того, насколько в дальнейшем она найдет применение в каких-либо видах деятельности человека.

Только своевременно полученная информация может принести ожидае-мую пользу. Одинаково нежелательны как преждевременная подача информации (когда она еще не может быть усвоена), так и ее задержка.

Если ценная и своевременная информация выражена непонятным обра-зом, она может стать бесполезной. Информация становится понятной, если она выражена языком, на котором говорят те, кому предназначена эта информация.

Информация должна преподноситься в доступной (по уровню восприя-тия) форме. Поэтому одни и те же вопросы по-разному излагаются в школь-ных учебниках и научных изданиях.

Информацию по одному и тому же вопросу можно изложить кратко (сжато, без несущественных деталей) или пространно (подробно, многословно). Краткость информации необходима в справочниках, энциклопедиях, всевозмож-ных инструкциях.

8. Емкость канала связи.

Объём канала ~V[1] определяется по формуле: ~V_k=\Delta F_k\cdot T_k\cdot D_k, где ~T_k — время, в течение которого канал занят передаваемым сигналом;

Для передачи сигнала по каналу без искажений объём канала ~V_k должен быть больше либо равен объёму сигнала ~V_s, то есть ~V_k \geqslant ~V_s. Простейший случай вписывания объёма сигнала в объём канала — это достижение выполнения неравенств ~\Delta F_k\geqslant~\Delta F_s, ~T_k \geqslant~T_s> и ~\Delta D_k \geqslant~\Delta D_s. Тем не менее, ~V_k \geqslant ~V_s может выполняться и в других случаях, что даёт возможность добиться требуемых характеристик канала изменением других параметров. Например, с уменьшением диапазона частот можно увеличить полосу пропускания.

9.Способы измерения информации.

Наиболее часто используются следующие два способа измерения информации: объёмный и вероятностный.

Объёмный подход. В двоичной системе счисления знаки 0 и 1 будем называть битами (от английско-го выражения BInary digiTs - двоичные цифры). Отдают предпочтение именно двоичной системе счисления потому, что в техническом устройстве наиболее просто реализовать два противоположных физических состояния: намагничено / не намагничено, вкл./выкл., заряжено / не заряжено и др.

Объём информации, записанной двоичными знаками в памяти компьютера или на внешнем носителе информации, подсчитывается просто по количеству требуемых для такой записи двоичных символов. При этом невозмож-но нецелое число битов.

Для удобства использования введены и более крупные, чем бит, единицы коли-чества информации. Так, двоичное слово из восьми знаков содержит один байт информации, 1024 байта образуют килобайт (кбайт), 1024 килобайта – мегабайт (Мбайт), а 1024 мегабайта - гигабайт (Гбайт).

Энтропийный (вероятностный) подход. Этот подход принят в теории информации и кодирования. Данный способ измерения исходит из следующей модели: получатель сообщения имеет определённое представление о возможных наступлениях некоторых событий. Эти представления в общем случае недостоверны и выражаются вероятностями, с которыми он ожидает то или иное событие. Общая мера неопределённостей называется энтропией. Энтропия характеризуется некоторой математической зависимостью от совокупности вероятности наступления этих событий.

Количество информации в сообщении определяется тем, насколько уменьшилась эта мера после получения сообщения: чем больше энтропия системы, тем больше степень её неопределённости. Поступающее сообщение полностью или частично снимает эту неопределённость, следовательно, количество информации можно измерять тем, насколько понизилась энтропия системы после получения сообщения. За меру количества информации принимается та же энтропия, но с обратным знаком.

1. 2 Лекция №2 (2 часа).

Тема: «Понятие энтропии и семантическая информация. (Интерактивная форма)»

1.2.1 Вопросы лекции:

1. Смысл энтропии Шеннона.
2. Понятие семантической информации.
3. Семантическая мера информации.
4. Семантическая информация в системах.

1.2.2 Краткое содержание вопросов:

1. Смысл энтропии Шеннона.

Наиболее широкое распространение при определении среднего количества информации, которое содержится в сообщениях от источников самой разной природы, получил подход. К Шеннона. Рассмотрим следующую ситуацию.

Источник передает элементарные сигналы k различных типов. Проследим за достаточно длинным отрезком сообщения. Пусть в нем имеется N1 сигналов первого типа, N2 сигналов второго типа, ..., Nk сигналов k-го типа, причем N1 + N2 + ... + Nk = N – общее число сигналов в наблюдаемом отрезке, f1, f2, ..., fk – частоты соответствующих сигналов. При возрастании длины отрезка сообщения каждая из частот стремится к фиксированному пределу, т.е.

lim fi = pi, (i = 1, 2, ..., k),

где pi можно считать вероятностью сигнала. Предположим, получен сигнал i-го типа с вероятностью pi, содержащий – log pi единиц информации. В рассматриваемом отрезке i-й сигнал встретится примерно Npi раз (будем считать, что N достаточно велико), и общая информация, доставленная сигналами этого типа, будет равна произведению Npi log pi. То же относится к сигналам любого другого типа, поэтому полное количество информации, доставленное отрезком из N сигналов, будет примерно равно

Чтобы определить среднее количество информации, приходящееся на один сигнал, т.е. удельную информативность источника, нужно это число разделить на N. При неограниченном росте приблизительное равенство перейдет в точное. В результате будет получено асимптотическое соотношение – формула Шеннона

В последнее время она стала не менее распространенной, чем знаменитая формула Эйнштейна E = mc2. Оказалось, что формула, предложенная Хартли, представляет собой частный случай более общей формулы Шеннона. Если в формуле Шеннона принять, что p1 = p2 = ... = pi = ... =pN = 1/N, то

Знак минус в формуле Шеннона не означает, что количество информации в сообщении – отрицательная величина. Объясняется это тем, что вероятность p, согласно определению, меньше единицы, но больше нуля. Так как логарифм числа, меньшего единицы, т.е. log pi – величина отрицательная, то произведение вероятности на логарифм числа будет положительным.

Кроме этой формулы, Шенноном была предложена абстрактная схема связи, состоящая из пяти элементов (источника информации, передатчика, линии связи, приемника и адресата), и сформулированы теоремы о пропускной способности, помехоустойчивости, кодировании и т.д.

В результате развития теории информации и ее приложений идеи Шеннона быстро распространяли свое влияние на самые различные области знаний. Было замечено, что формула Шеннона очень похожа на используемую в физике формулу энтропии, выведенную Больцманом. Энтропия обозначает степень неупорядоченности статистических форм движения молекул. Энтропия максимальна при равновероятном распределении параметров движения молекул (направлении, скорости и пространственном положении). Значение энтропии уменьшается, если движение молекул упорядочить. По мере увеличения упорядоченности движения энтропия стремится к нулю (например, когда возможно только одно значение и направление скорости). При

составлении какого-либо сообщения (текста) с помощью энтропии можно характеризовать степень неупорядоченности движения (чередования) символов. Текст с максимальной энтропией – это текст с равновероятным распределением всех букв алфавита, т.е. с бессмысленным чередованием букв, например: ЙХЗЦЗЦЩУЩУШК ШГЕНЕЭФЖЫЫДВЛВЛОАРАПАЯЕЯЮЧБ СБСЬМ. Если при составлении текста учтена реальная вероятность букв, то в получаемых таким образом «фразах» будет наблюдаться определенная упорядоченность движения букв, регламентируемая частотой их появления: ЕЫТ ЦИЯЬА ОКРВ ОДНТ ЬЧЕ МЛОЦК ЗЬЯ ЕНВ ТША.

При учете вероятностей четырехбуквенных сочетаний текст становится настолько упорядоченным, что по некоторым формальным признакам приближается к осмысленному: ВЕСЕЛ ВРАТЬСЯ НЕ СУХОМ И НЕПО И КОРКО. Причиной такой упорядоченности в данном случае является информация о статистических закономерностях текстов. В осмысленных текстах упорядоченность, естественно, еще выше. Так, в фразе ПРИШЛ... ВЕСНА мы имеем еще больше информации о движении (чередовании) букв. Таким образом, от текста к тексту увеличиваются упорядоченность и информация, которой мы располагаем о тексте, а энтропия (мера неупорядоченности) уменьшается.

Используя различие формул количества информации Шеннона и энтропии Больцмана (разные знаки), Л. Бриллюэн охарактеризовал информацию как отрицательную энтропию, или негэнтропию. Так как энтропия является мерой неупорядоченности, то информация может быть определена как мера упорядоченности материальных систем.

В связи с тем, что внешний вид формул совпадает, можно предположить, что понятие информация ничего не добавляет к понятию энтропии. Однако это не так. Если понятие энтропии применялось ранее только для систем, стремящихся к термодинамическому равновесию, т.е. к максимальному беспорядку в движении ее составляющих, к увеличению энтропии, то понятие информации обратило внимание и на те системы, которые не увеличивают энтропию, а наоборот, находясь в состоянии с небольшими значениями энтропии, стремятся к ее дальнейшему уменьшению.

2. Смысл семантической информации.

Выделяют следующие аспекты информации:

- прагматический,
- семантический,
- синтаксический.

Прагматический аспект связан с возможностью достижения поставленной цели с использованием получаемой информации. Этот аспект информации влияет на поведение потребителя. Если информация была эффективной, то поведение потребителя меняется в желаемом направлении, т. е. информация имеет прагматическое содержание. Таким образом, этот аспект характеризует поведенческую сторону проблемы.

Семантический аспект позволяет оценить смысл передаваемой информации, соотнося ее с информацией, хранящейся до появления данной. Семантические связи между словами или другими смысловыми элементами языка отражают словарь-тезаурус. Он состоит из двух частей: списка слов и устойчивых словосочетаний, которые сгруппированы по смыслу, и некоторого ключа, т е. алфавитного словаря, позволяющего расположить слова и словосочетания в определенном порядке. Тезаурус имеет особое значение в системах хранения информации, в которые могут вводиться семантические отношения, в основном подчинения, что позволяет на логическом уровне осуществлять организацию информации в виде отдельных записей, массивов и их комплексов. Существуют развитые тезаурусы, в которые включаются сложные высказывания и семантические связи между ними. Это позволяет хранить более сложную информацию и детально оценивать семантическое содержание вновь поступающей информации. Наличие тезауруса позволяет переводить поступающую семантическую информацию на некоторый стандартизированный семантический язык в соответствии с выбранным тезаурусом. Таким образом, при возникновении информации можно изменить исходный тезаурус. Степень изменения тезауруса может быть принята как характеристика количества информации.

Синтаксический аспект информации связан со способом ее представления. В зависимости от реального процесса, в котором участвует информация, т.е. осуществляется ее сбор, передача, преобразование, отображение, представление, ввод или вывод, она представляется в виде специальных знаков, символов. Характерным носителем информации является сообщение, под которым обычно понимают все то, что подлежит передаче. Сообщения представляют в виде электрического сигнала, передаваемого по выбранной физической среде. Для этого сообщение подвергают преобразованию, т. е. придают ему электрический характер, далее кодированию, при котором сообщение превращается в некоторую. последовательность символов, однозначно его отображающих, и модуляции, при которой каждый элемент кода (либо код в целом) переводится в электрический сигнал, способный передаваться на заданное расстояние по выбранному каналу связи. Процессы преобразования, кодирования и модуляции исключительно многообразны, а синтаксический аспект информации при ее передаче в

настоящее время хорошо развит. Иной характер синтаксический аспект имеет, например, при хранении информации. В этом случае могут быть предложены такие формы, при которых удается осуществить быстрый поиск, введение новой информации, вывод требуемой информации из информационной базы и в целом обновления базы данных. Требуемому представлению информации при ее хранении отвечают разработанные к настоящему времени типовые структуры баз и банков данных, которые позволяют наилучшим образом реализовать информационное обслуживание пользователей в системе управления. Таким образом, развитие общества привело к тому, что оказалось необходимым хранить, обрабатывать, передавать, преобразовывать огромные объемы данных.

3. Семантическая мера информации.

Синтаксическая мера информации оперирует с обезличенной информацией, не выражающей смыслового отношения к объекту. На синтаксическом уровне учитываются тип носителя и способ представления информации, скорость передачи и обработки, размеры кодов представления информации.

Существуют два основных подхода в определении количества информации. Исторически они возникли почти одновременно. В конце 40-х г. XX века один из основоположников кибернетики, американский математик Клод Шеннон развил вероятностный подход к измерению количества информации, а работы по созданию ЭВМ привели к «объемному» подходу.

Объём данных (VД) понимается в техническом смысле этого слова как информационный объём сообщения или как объём памяти, необходимый для хранения сообщения без каких-либо изменений.

Информационный объём сообщения измеряется в битах и равен количеству двоичных цифр (“0” и “1”), которыми закодировано сообщение.

В компьютерной практике слово “бит” используется также как единица измерения объёма памяти. Ячейка памяти размером в 1 бит может находиться в двух состояниях (“включено” и “выключено”) и в неё может быть записана одна двоичная цифра (0 или 1). Понятно, что бит – слишком маленькая единица измерения информации, поэтому пользуются кратными ей величинами. Основной единицей измерения информации является байт. 1 байт равен 8 битам. В ячейку размером в 1 байт можно поместить 8 двоичных цифр, то есть в одном байте можно хранить 256 = 28 различных чисел. Для измерения ещё больших объёмов информации используются следующие величины:

1 Кбайт (один килобайт) = 210 байт = 1024 байта (1 kB);

1 Мбайт (один мегабайт) = 210 Кбайт = 1024 Кбайта (1 MB);

1 Гбайт (один гигабайт) = 210 Мбайт = 1024 Мбайта (1 GB);

1 Тбайт (один терабайт) = 210 Гбайт = 1024 Гбайта (1 TB);

1 Пбайт (один петабайт) = 210 Тбайт = 1024 Тбайта (1PB);

1 Эбайт (один эксабайт) = 210 Пбайт = 1024 Пбайта (1 EB);

1 Збайт (один зеттабайт) = 210 Эбайт = 1024 Эбайта (1 ZB);

1 Йбайт (один йоттабайт) = 210 Збайт = 1024 Збайта (1 YB).

Пример 1. При двоичном кодировании текста каждая буква, знак препинания, пробел занимают 1 байт. На странице книги среднего формата примерно 50 строк, в каждой строке около 60 символов, таким образом, полностью заполненная страница имеет объём 50х60= = 3000 байт ≈3 Килобайта. Вся книга среднего формата занимает ≈0,5 Мегабайт. Один номер четырёхстраничной газеты – 150 Килобайт. Если человек говорит по 8 часов в день без перерыва, то за 70 лет он наговорит около 10 Гигабайт информации. Один чёрно-белый кадр (при 32 градациях яркости каждой точки) содержит примерно 300Кб информации, цветной кадр содержит уже около 1Мб информации. Телевизионный фильм продолжительностью 1,5часа с частотой 25 кадров в секунду - 135 Гб.

При вероятностном подходе количество информации I на синтаксическом уровне определяется через понятие энтропии системы.

Пусть до получения информации потребитель имеет некоторые предварительные (априорные) сведения о системе α. Мерой его неосведомленности о системе является функция H(α), которая в то же время служит и мерой неопределенности состояния системы.

После получения некоторого сообщения β получатель приобрел некоторую дополнительную информацию Iβ(α), уменьшившую его априорную неосведомленность так, что неопределенность состояния системы после получения сообщения β стала Hβ(α). Тогда количество информации Iβ(α) о системе, полученной в сообщении β, определится как

$$I_\beta(\alpha) = H(\alpha) - H_\beta(\alpha).$$

т.е. количество информации измеряется изменением (уменьшением) неопределенности состояния системы. Если конечная неопределенность Hβ(α) обратится в нуль, то первоначальное неполное знание заменится полным знанием и количество информации будет определяться как Iβ(α) = H(α). Иными словами, энтропия системы H(a) может рассматриваться как мера недостающей информации.

Энтропия системы H(α), имеющая N возможных состояний, согласно формуле Шеннона, равна:

где – вероятность того, что система находится в i-м состоянии. Для случая, когда все состояния системы равновероятны, т.е. их вероятности равны , ее энтропия определяется соотношением:

Пример 2. Часто информация кодируется числовыми кодами в той или иной системе счисления, особенно это актуально при представлении информации в компьютере. Естественно, что одно и то же количество разрядов в разных системах счисления может передавать разное число состояний отображаемого объекта, что можно представить в виде соотношения

N = mn,где N – число всевозможных отображаемых состояний;

m – основание системы счисления (разнообразие символов, применяемых в алфавите);

n – число разрядов (символов) в сообщении.

Допустим, что по каналу связи передается n-разрядное сообщение, использующее m различных символов. Так как количество всевозможных кодовых комбинаций будет N = mn, то при равновероятности появления любой из них количество информации, приобретенной абонентом в результате получения сообщения, будет определяться по формуле Хартли:

I = log N = n log m

Если в качестве основания логарифма принять m, то I = n. В данном случае количество информации (при условии полного априорного незнания абонентом содержания сообщения) будет равно объему данных I = VД, полученных по каналу связи.

Наиболее часто используются двоичные и десятичные логарифмы. Единицами измерения в этих случаях будут соответственно бит и дит.

Семантическая мера информации

Для измерения смыслового содержания информации, т.е. ее количества на семантическом уровне, наибольшее признание получила тезаурусная мера, которая связывает семантические свойства информации со способностью пользователя принимать поступившее сообщение. Для этого используется понятие «тезаурус пользователя».

Тезаурус – это совокупность сведений, которыми располагает пользователь или система.

4. Семантическая информация в системах.

Семантическая (смысловая) адекватность определяет степень соответствия информации об объекте самому объекту.

1. 3 Лекция №3 (2 часа).

Тема: «Сжатие информации. Адаптивные алгоритмы сжатия.(Интерактивная форма»

1.3.1 Вопросы лекции:

1. Необратимое сжатие.
2. Обратимое сжатие.
3. Кодирование Хаффмана.
4. Упорядоченное дерево Хаффмана.
5. Избыточность сообщений.
6. Код Фано

1.3.2 Краткое содержание вопросов:

1. Необратимое сжатие.

Под необратимым сжатием подразумевают такое преобразование входного потока данных, при котором выходной поток, основанный на определенном формате информации, представляет, с некоторой точки зрения, достаточно похожий по внешним характеристикам на входной поток объект, однако отличается от него объемом. Степень сходства входного и выходного потоков определяется степенью соответствия некоторых свойств объекта (т.е. сжатой и несжатой информации, в соответствии с некоторым определенным форматом данных), представляемого данным потоком информации. Такие подходы и алгоритмы используются для сжатия, например, данных растровых графических файлов с низкой степенью повторяемости байтов в потоке. При таком подходе используется свойство структуры формата графического файла и возможность представить графическую картинку приблизительно схожую по качеству отображения (для восприятия человеческим глазом) несколькими (а точнее n) способами. Поэтому, кроме степени или величины сжатия, в таких алгоритмах возникает понятие качества, т.к. исходное изображение в процессе сжатия изменяется, то под качеством можно понимать степень соответствия исходного и результирующего изображения, оцениваемая субъективно, исходя из формата информации. Для графических файлов такое соответствие определяется визуально, хотя имеются и соответствующие интеллектуальные алгоритмы и программы. Необратимое сжатие невозможно применять в областях, в которых необходимо иметь точное соответствие информационной структуры

входного и выходного потоков. Данный подход реализован в популярных форматах представления видео и фото информации, известных как JPEG и JFIF алгоритмы и JPG и JIF форматы файлов.

2. Обратимое сжатие.

Обратимое сжатие всегда приводит к снижению объема выходного потока информации без изменения его информативности, т.е. - без потери информационной структуры. Более того, из выходного потока, при помощи восстанавливающего или декомпрессирующего алгоритма, можно получить входной, а процесс восстановления называется декомпрессией или распаковкой, и только после процесса распаковки данные пригодны для обработки в соответствии с их внутренним форматом.

3. Кодирование Хаффмана.

Определение 1: Пусть A={a1,a2,...,an} - алфавит из n различных символов, W={w1,w2,...,wn} - соответствующий ему набор положительных целых весов. Тогда набор бинарных кодов C={c1,c2,...,cn}, такой что:

(1) ci не является префиксом для cj, при i!=j

(2) $\sum_{i=1}^n{w_i|c_i|}$ минимальна (|ci| длина кода ci)

называется минимально-избыточным префиксным кодом или иначе кодом Хаффмана.

Замечания:

Свойство (1) называется свойством префиксности. Оно позволяет однозначно декодировать коды переменной длины.

Сумму в свойстве (2) можно трактовать как размер закодированных данных в битах. На практике это очень удобно, т.к. позволяет оценить степень сжатия не прибегая непосредственно к кодированию.

В дальнейшем, чтобы избежать недоразумений, под кодом будем понимать битовую строку определенной длины, а под минимально-избыточным кодом или кодом Хаффмана - множество кодов (битовых строк), соответствующих определенным символам и обладающих определенными свойствами.

Известно, что любому бинарному префиксному коду соответствует определенное бинарное дерево.

Определение 2: Бинарное дерево, соответствующее коду Хаффмана, будем называть деревом Хаффмана

Задача построения кода Хаффмана равносильна задаче построения соответствующего ему дерева. Приведем общую схему построения дерева Хаффмана:

Составим список кодируемых символов (при этом будем рассматривать каждый символ как одноэлементное бинарное дерево, вес которого равен весу символа).

Из списка выберем 2 узла с наименьшим весом.

Сформируем новый узел и присоединим к нему, в качестве дочерних, два узла выбранных из списка. При этом вес сформированного узла положим равным сумме весов дочерних узлов.

Добавим сформированный узел к списку.

Если в списке больше одного узла, то повторить 2-5.

Приведем пример: построим дерево Хаффмана для сообщения S="A H F B H C E H E H C E A H D C E E H H H C H H H D E G H G G E H C H H".

Для начала введем несколько обозначений:

Символы кодируемого алфавита будем выделять жирным шрифтом: A, B, C.

Веса узлов будем обозначать нижними индексами: A5, B3, C7.

Составные узлы будем заключать в скобки: ((A5+B3)8+C7)15.

Итак, в нашем случае A={A, B, C, D, E, F, G, H}, W={2, 1, 5, 2, 7, 1, 3, 15}.

A2B1C5D2E7F1G3H15

A2C5D2E7G3H15 (F1+B1)2

C5E7G3H15 (F1+B1)2 (A2+D2)4

C5E7H15 (A2+D2)4 ((F1+B1)2+G3)5

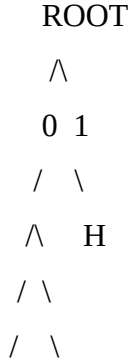
E7H15 ((F1+B1)2+G3)5 (C5+(A2+D2)4)9

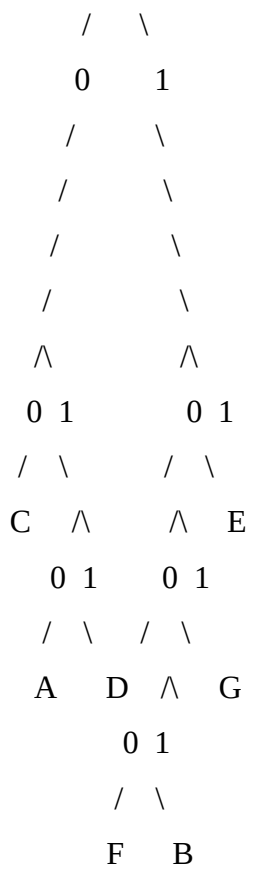
H15 (C5+(A2+D2)4)9 (((F1+B1)2+G3) 5+E7)12

H15 ((C5+(A2+D2)4) 9+(((F1+B1)2+G3) 5+E7)12)21

(((C5+(A2+D2)4) 9+(((F1+B1)2+G3) 5+E7)12)21+H15)36

В списке, как и требовалось, остался всего один узел. Дерево Хаффмана построено. Теперь запишем его в более привычном для нас виде.





Листовые узлы дерева Хаффмана соответствуют символам кодируемого алфавита. Глубина листовых узлов равна длине кода соответствующих символов.

Путь от корня дерева к листовому узлу можно представить в виде битовой строки, в которой "0" соответствует выбору левого поддерева, а "1" - правого. Используя этот механизм, мы без труда можем присвоить коды всем символам кодируемого алфавита. Выпишем, к примеру, коды для всех символов в нашем примере:

A=0010bin C=000bin E=011bin G=0101bin

B=01001bin D=0011bin F=01000bin H=1bin

Теперь у нас есть все необходимое для того чтобы закодировать сообщение S. Достаточно просто заменить каждый символ соответствующим ему кодом:

S/="0010 1 01000 01001 1 000 011 1 011 1 000 011 0010 1 0011 000 011 011 1 1 1 000 1 1 1 0011 011 0101 1 0101 0101 011 1 000 1 1".

Оценим теперь степень сжатия. В исходном сообщении S было 36 символов, на каждый из которых отводилось по [log2|A|]=3 бита (здесь и далее будем понимать квадратные скобки [] как целую часть, округленную в положительную сторону, т.е. [3,018]=4). Таким образом, размер S равен 36*3=108 бит

Размер закодированного сообщения S/ можно получить воспользовавшись замечанием 2 к определению 1, или непосредственно, подсчитав количество бит в S/. И в том и другом случае мы получим 89 бит.

Итак, нам удалось сжать 108 в 89 бит.

Теперь декодируем сообщение S/. Начиная с корня дерева будем двигаться вниз, выбирая левое поддерево, если очередной бит в потоке равен "0", и правое - если "1". Дойдя до листового узла мы декодируем соответствующий ему символ.

Ясно, что следуя этому алгоритму мы в точности получим исходное сообщение S.

4. Упорядоченное дерево Хаффмана.

Дерево Хаффмана – двоичное дерево, листьями которого являются символы алфавита, в каждой вершине которого хранится частота символа (для листа) или сумма частот ее двух детей (будем называть это число весом вершины). При этом выполняется следующее свойство: если расстояние от корня до вершины X больше, чем до вершины Y, то вес X не превосходит веса Y.

5. Избыточность сообщений.

Избыточность — термин из теории информации, означающий превышение количества информации, используемой для передачи или хранения сообщения, над его информационной энтропией. Для уменьшения избыточности применяется сжатие данных без потерь, в то же время контрольная сумма применяется для внесения дополнительной избыточности в поток, что позволяет производить исправление ошибок при передаче информации по каналам, вносящим искажения (спутниковая трансляция, беспроводная передача и т. д.).

Количественное определение:

Информационное содержание одного сообщения в потоке, в наиболее общем случае, определяется как:

r=\mathbb E H(M_t|M_{t-1},M_{t-2},M_{t-3}, \dots)

Обозначим как R логарифм числа символов в алфавите сообщений:

R = \log |M|

Абсолютная избыточность может быть определена как разность этих двух величин:

D = R - r

Соотношение \frac D R называется относительной избыточностью и дает математическую оценку максимальной степени сжатия, на которую может быть уменьшен размер файла.

6. Код Фано

Принцип построения оптимального кода Шеннона-Фано следующий.

Сообщения, входящие в ансамбль, располагаются в строку (в столбец) по мере убывания вероятностей.

Выбирается основание кода K.

Все сообщения ансамбля разбиваются на K групп с равными суммарными вероятностями внутри каждой группы. Всем сообщениям первой группы в качестве первого символа присваивается 0, сообщениям второй группы – символ 1, а сообщениям K-й группы – символ (K – 1); тем самым обеспечивается равная вероятность появления всех символов 0, 1,…, K на первой позиции в кодовых словах.

Каждая из групп делится на K подгрупп с равной суммарной вероятностью в каждой подгруппе. Всем сообщениям первых подгрупп в качестве второго символа присваивается 0, всем сообщениям вторых подгрупп – 1, а сообщениям K -х подгрупп – символ (K – 1).

Процесс продолжается до тех пор, пока в каждой подгруппе не окажется по одной комбинации.

1.4 Лекция №4 (2 часа)

Тема: «Арифметическое кодирование»

1.4.1 Вопросы лекции:

1.Алгоритм арифметического кодирования

2. Адаптивное арифметическое кодирование.

1.4.3 Краткое содержание вопросов.

1.Алгоритм арифметического кодирования

Сейчас существует множество алгоритмов сжатия информации. Большинство из них широко известны, но есть и некоторые весьма эффективные, но, тем не менее, малоизвестные алгоритмы. Эта статья рассказывает о методе арифметического

кодирования, который является лучшим из энтропийных, но тем не менее мало кто о нём знает.

Прежде чем рассказывать об арифметическом кодировании, надо сказать пару слов об алгоритме Хаффмана. Этот метод эффективен, когда частоты появления символов пропорциональны 1/2n (где n – натуральное положительное число). Это утверждение становится очевидным, если вспомнить, что коды Хаффмана для каждого символа всегда состоят из целого числа бит. Рассмотрим ситуацию, когда частота появление символа равна 0,2, тогда оптимальный код для кодирования это символа должен иметь длину – log2(0,2)=2,3 бита. Понятно, что префиксный код Хаффмана не может иметь такую длину, т.е. в конечном итоге это приводит к ухудшению сжатия данных.

Арифметическое кодирование предназначено для того, чтобы решить эту проблему. Основная идея заключается в том, чтобы присваивать коды не отдельным символам, а их последовательностям.

Вначале рассмотрим идею, лежащую в основе алгоритма, затем рассмотрим небольшой практический пример.

Как и во всех энтропийных алгоритмах мы обладаем информацией о частоте использования каждого символа алфавита. Эта информация является исходной для рассматриваемого метода. Теперь введём понятие рабочего отрезка. Рабочим называется полуинтервал [a;b) с расположенными на нём точками. Причём точки расположены т.о., что длины образованных ими отрезков равны частоте использования символов. При инициализации алгоритма a=0 b=1.

Один шаг кодирования заключается в простой операции: берётся кодируемый символ, для него ищется соответствующий участок на рабочем отрезке. Найденный участок становится новым рабочим отрезком (т.е. его тоже необходимо разбить с помощью точек).

Эта операция выполняется для некоторого количества символов исходного потока. Результатом кодирования цепочки символов является любое число (а также длина его битовой записи) с итогового рабочего отрезка (чаще всего берётся левая граница отрезка). Звучит это довольно сложно. Давайте попробуем разобраться с помощью небольшого примера. Закодируем сообщение «ЭТОТ_МЕТОД_ЛУЧШЕ_ХАФФМАНА» с помощью описанного метода.

Составив таблицу частоты появления символов, мы можем приступать к кодированию. На первом этапе составим рабочий отрезок. Выглядеть он будет так:

Берём первый символ из потока, это символ «Э». Соответствующий ему отрезок – отрезок [0,96;1). Если бы мы хотели закодировать один символ, то результатом кодирования было бы любое число из этого отрезка. Но мы не остановимся на одном символе, а добавим ещё один. Символ «Т». Для этого составим новый рабочий отрезок с a=0,96 и b=1. Разбиваем этот отрезок точками точно так же как мы сделали это для исходного отрезка и считываем новый символ «Т». Символу «Т» соответствует диапазон [0,24;0,36), но наш рабочий отрезок уже сократился до отрезка [0,96;1). Т.е. границы нам необходимо пересчитать. Сделать это можно с помощью двух следующих формул: High=Lowold+(Highold-Lowold)*RangeHigh(x), Low=Lowold+(Highold-Lowold)*RangeLow(x), где Lowold – нижняя граница интервала, Highold – верхняя граница интервала RangeHigh и RangeLow – верхняя и нижняя границы кодируемого символа.

Пользуясь этими формулами, закодируем первое слово сообщения целиком:

Результатом кодирования будет любое число из полуинтервала [0,97218816; 0,97223424).

Предположим, что результатом кодирования была выбрана левая граница полуинтервала, т.е. число 0,97218816.

Рассмотрим процесс декодирования. Код лежит в полуинтервале [0,96;1). Т.е. первый символ сообщения «Э». Чтобы декодировать второй символ (который кодировался в полуинтервале [0,96;1)) полуинтервал нужно нормализовать, т.е. привести к виду [0;1). Это делается с помощью следующей формулы: code=(code-RangeLow(x))/(RangeHigh(x)-RangeLow(x)), где code – текущее значение кода.

Применяя эту формулу, получаем новое значение code=(0,97218816-0,96)/(1-0,96)= 0,304704. Т.е. второй символ последовательности «Т».

Снова применим формулу: code=(0,304704-0,24)/(0,36-0,24)= 0,5392. Третий символ последовательности – «О».

Продолжая декодирование, по описанной схеме мы можем полностью восстановить исходный текст.

Т.о. мы с вами рассмотрели алгоритм кодирования и декодирования с помощью самого эффективного из всех энтропийных алгоритмов. Надеюсь, из этой статьи вы узнали что-то новое и поняли основные идеи, лежащие в алгоритме арифметического кодирования.

2. Адаптивное арифметическое кодирование.

Каждому символу сопоставляется его вес, вначале вес 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. Для того чтобы обеспечить остановку алгоритма распаковки вначале сжимаемого сообщения, надо поставить его длину или ввести дополнительный символ-маркер. Затем, аналогичным простому арифметическому кодированию методом, выбирается число, описывающее кодирование.

Декодирование происходит следующим образом: на каждом шаге определяется интервал, содержащий данный код (выбранное число) – по этому интервалу однозначно задается исходный символ сообщения. Затем из текущего кода вычитается нижняя граница содержащего интервала, полученная разность делится на длину этого же интервала. Полученное число считается новым текущим значением кода. Получение маркера конца или заданного перед началом работы алгоритма числа символов означает окончание работы.

1.5Лекция № 5(2 часа)

Тема: «Словарно- ориентированные алгоритмы сжатия»(Интерактивная форма)

1.5.1 Вопросы лекции:

1. Метод Лемпела-Зива LZ77.
2. Метод LZSS.
3. Метод LZ78.
4. Метод LZW.
5. LZ-алгоритмы распаковки данных.

1.5.2 Краткое содержание вопросов:

1. Метод Лемпела-Зива LZ77.

А — — (Lempel-Ziv-Welch, LZW) — это универсальный алгоритм сжатия данных без потерь, созданный Авраамом Лемпелем (англ. Abraham Lempel), Яаковом Зивом (англ. Jacob Ziv) и Терри Велчем (англ. Terry Welch). Он был опубликован Велчем в 1984 году, в качестве улучшенной реализации алгоритма LZ78, опубликованного Лемпелем и Зивом в 1978 году. Алгоритм разработан

так, чтобы его можно было быстро реализовать, но он не обязательно оптимален, поскольку он не проводит никакого анализа входных данных.

Акроним «LZW» указывает на фамилии изобретателей алгоритма: Лемпель, Зив и Велч, но многие[кто?] утверждают, что, поскольку патент принадлежал Зиву, то метод должен называться алгоритмом Зива — Лемпеля — Велча.

Данный алгоритм при сжатии (кодировании) динамически создаёт таблицу преобразования строк: определённым последовательностям символов (словам) ставятся в соответствие группы бит фиксированной длины (обычно 12-битные). Таблица инициализируется всеми 1-символьными строками (в случае 8-битных символов — это 256 записей). По мере кодирования алгоритм просматривает текст символ за символом, и сохраняет каждую новую, уникальную 2-символьную строку в таблицу в виде пары код/символ, где код ссылается на соответствующий первый символ. После того как новая 2-символьная строка сохранена в таблице, на выход передаётся код первого символа. Когда на входе читается очередной символ, для него по таблице находится уже встречавшаяся строка максимальной длины, после чего в таблице сохраняется код этой строки со следующим символом на входе; на выход выдаётся код этой строки, а следующий символ используется в качестве начала следующей строки.

Алгоритму декодирования на входе требуется только закодированный текст, поскольку он может воссоздать соответствующую таблицу преобразования непосредственно по закодированному тексту.

2. Метод LZSS.

Алгоритм LZSS получил свое название по именам своих разработчиков: Lempel, Ziv, Storer, Szimansky. Существенный вклад в развитие алгоритма внес T. C. Bell. LZSS является развитием LZ77 и стремится избавиться от недостатков своего предшественника. LZSS отличается от LZ77 тем, как поддерживается скользящее окно и какие коды выдает компрессор.

LZSS, помимо собственно окна с содержимым сообщения, поддерживает двоичное дерево для ускорения поиска совпадения. Каждая подстрока, покидающая буфер, добавляется в дерево поиска, а подстрока, покидающая словарь, удаляется из дерева. Такая организация модели данных позволяет добиться существенного увеличения скорости поиска совпадения, которая, в отличие от LZ77, становится пропорциональна не произведению размеров окна и подстроки, а его двоичному логарифму. Это позволяет экспериментировать с большими окнами, не теряя в скорости сжатия.

Кодер LZSS использует однобитовый префикс, для того чтобы отличать незакодированные символы от пар <смещение, длина>. Такие коды позволяют получить существенный выигрыш в размере сжатого сообщения

Как и в LZ77, в этом алгоритме используется обычный символьный буфер для хранения содержимого окна. В целях повышения эффективности “скольжения” окна по содержимому сообщения доступ к нему организован как к кольцевому, и размер окна кратен степени 2.

Дерево поиска представляет собой двоичное лексикографически упорядоченное дерево. Каждый узел в дереве соответствует одной подстроке словаря и содержит ссылки на родителя и двух потомков: “большего” и “меньшего” в смысле лексикографического сравнения символьных строк.

Для того чтобы кодер мог начать работать, необходимо загрузить буфер очередными символами сообщения и проинициализировать дерево. Для этого в дерево вставляется содержимое буфера.

Алгоритм последовательно выполняет следующие действия:

кодирует содержимое буфера;

считывает очередные символы в буфер, удаляя при необходимости наиболее “старые” строки из словаря;

вставляет в дерево новые строки, соответствующие считанным символам.

Завершение работы

Для того чтобы декодер смог вовремя остановиться, декодируя сжатое сообщение, кодер помещает в сжатый файл специальный символ “КОНЕЦ ФАЙЛА” после того, как он обработал все символы сообщения.

Вся работа по поиску расположения и установлению длины максимального совпадения содержимого словаря с буфером происходит в процессе добавления в дерево новой строки.

При добавлении строки в дерево алгоритм последовательно перемещается от корня дерева к его листу, каждый раз осуществляя лексикографическое сравнение новой строки и узла дерева. В зависимости от результата сравнения выбирается больший или меньший потомок этого узла и запоминается длина совпадения строк и положение текущего узла.

Если в результате сравнения оказывается, что содержимое буфера и строка, на которую ссылается текущий узел, в точности совпадают, то ссылки в текущем узле обновляются так, чтобы они указывали на буфер, и процедура добавления строки в дерево завершается.

Если потомок текущего узла, выбранный для очередного шага, отмечен как несуществующий, остается заполнить его соответствующей ссылкой и завершить работу.

Нужно отметить, что при окончании работы процедуре добавления строки известны и смещение, и длина наибольшего совпадения, и для этого не потребовался полный перебор всех возможных вариантов совпадения.

Декодер читает один бит сжатой информации и решает - это символ или пара <смещение, длина>. Если это символ, то следующие 8 битов выдаются как раскодированный символ и помещаются в скользящее окно. Иначе, если это не закодированный конец файла, то соответствующее количество символов словаря помещается в окно и выдается в раскодированном виде. Поскольку это все, что делает декодер, понятно, почему процедура декодирования работает так быстро.

3. Метод LZ78.

В отличие от LZ77, работающего с уже полученными данными, LZ78 ориентируется на данные, которые только будут получены (LZ78 не использует скользящее окно, он хранит словарь из уже просмотренных фраз). Алгоритм считывает символы сообщения до тех пор, пока накапливаемая подстрока входит целиком в одну из фраз словаря. Как только эта строка перестанет соответствовать хотя бы одной фразе словаря, алгоритм генерирует код, состоящий из индекса строки в словаре, которая до последнего введенного символа содержала входную строку, и символа, нарушившего совпадение. Затем в словарь добавляется введенная подстрока. Если словарь уже заполнен, то из него предварительно удаляют менее всех используемую в сравнениях фразу. Если в конце алгоритма мы не находим символ, нарушивший совпадения, то тогда мы выдаем код в виде (индекс строки в словаре без последнего символа, последний символ).

4. Метод LZW.

Непосредственным предшественником LZW является алгоритм LZ78, опубликованный Абрахамом Лемпелем (Abraham Lempel) и Якобом Зивом (Jacob Ziv) в 1978 г. Этот алгоритм воспринимался как математическая абстракция до 1984 г., когда Терри Уэлч (Terry A. Welch) опубликовал свою работу с модифицированным алгоритмом, получившим в дальнейшем название LZW (Lempel—Ziv—Welch).

Опубликование алгоритма LZW произвело большое впечатление на всех специалистов по сжатию информации. За этим последовало большое количество программ и приложений с различными вариантами этого метода.

Этот метод позволяет достичь одну из наилучших степеней сжатия среди других существующих методов сжатия графических данных, при полном отсутствии потерь или искажений в исходных файлах. В настоящее время используется в файлах формата TIFF, PDF, GIF, PostScript и других, а также отчасти во многих популярных программах сжатия данных (ZIP, ARJ, LHA).

Процесс сжатия выглядит следующим образом: последовательно считываются символы входного потока и происходит проверка, существует ли в созданной таблице строк такая строка. Если такая строка существует, считывается следующий символ, а если строка не существует, в поток заносится код для предыдущей найденной строки, строка заносится в таблицу, а поиск начинается снова.

Например, если сжимают байтовые данные (текст), то строк в таблице окажется 256 (от "0" до "255"). Если используется 10-битный код, то под коды для строк остаются значения в диапазоне от 256 до 1023. Новые строки формируют таблицу последовательно, т. е. можно считать индекс строки ее кодом.

Для декодирования на вход подается только закодированный текст, поскольку алгоритм LZW может воссоздать соответствующую таблицу преобразования непосредственно по закодированному тексту. Алгоритм генерирует однозначно декодируемый код за счет того, что каждый раз, когда генерируется новый код, новая строка добавляется в таблицу строк. LZW постоянно проверяет, является ли строка уже известной, и, если так, выводит существующий код без генерации нового. Таким образом, каждая строка будет храниться в единственном экземпляре и иметь свой уникальный номер. Следовательно, при декодировании во время получения нового кода генерируется новая строка, а при получении уже известного, строка извлекается из словаря.

Кодирование

Начало.

Шаг 1. Все возможные символы заносятся в словарь. Во входную фразу X заносится первый символ сообщения.

Шаг 2. Считать очередной символ Y из сообщения.

Шаг 3. Если Y — это символ конца сообщения, то выдать код для X, иначе:

Если фраза XY уже имеется в словаре, то присвоить входной фразе значение XY и перейти к Шагу 2 ,

Иначе выдать код для входной фразы X, добавить XY в словарь и присвоить входной фразе значение Y. Перейти к Шагу 2.

Конец.

Декодирование

Начало.

Шаг 1. Все возможные символы заносятся в словарь. Во входную фразу X заносится первый код декодируемого сообщения.

Шаг 2. Считать очередной код Y из сообщения.

Шаг 3. Если Y — это конец сообщения, то выдать символ, соответствующий коду X, иначе:

Если фразы под кодом XY нет в словаре, вывести фразу, соответствующую коду X, а фразу с кодом XY занести в словарь.

Иначе присвоить входной фразе код XY и перейти к Шагу 2 .

Конец.

5. LZ-алгоритмы распаковки данных.

1. LZ77, длина словаря - 8 байт (символов). Коды сжатого сообщения - $\langle 0,0,'К'\rangle\langle 0,0,'Р'\rangle\langle 0,0,'А'\rangle\langle 0,0,'С'\rangle\langle 0,0,'Н'\rangle\langle 5,1,'Я'\rangle\langle 0,0,''\rangle\langle 0,4,'К'\rangle$.

ВХОДНОЙ КОД	ПЕЧАТЬ	СЛОВАРЬ
<0,0,'К'>	"К"	".....К"
<0,0,'Р'>	"Р"	"....КР"
<0,0,'А'>	"А"	"...КРА"
<0,0,'С'>	"С"	"..КРАС"
<0,0,'Н'>	"Н"	".КРАСН"
<5,1,'Я'>	"АЯ"	"КРАСНАЯ"
<0,0,' '>		"КРАСНАЯ "
<0,4,'К'>	"КРАСК"	"АЯ КРАСК"
<0,0,'А'>	"А"	"Я КРАСКА"

2. LZSS, длина словаря - 8 байт (символов). Коды сжатого сообщения - $0'К'0'Р'0'А'0'С'0'Н'1\langle 5,1\rangle 0'Я'0''1\langle 0,4\rangle 1\langle 4,1\rangle 1\langle 0,1\rangle$.

ВХОДНОЙ КОД	ПЕЧАТЬ	СЛОВАРЬ
0 'К'	"К"	".....К"
0 'Р'	"Р"	"....КР"
0 'А'	"А"	"...КРА"
0 'С'	"С"	"..КРАС"
0 'Н'	"Н"	".КРАСН"
1 <5,1>	"А"	"КРАСНА"
0 'Я'	"Я"	"РАСНАЯ"
0 ' '		"АСНАЯ "
1 <0,4>	"КРАС"	"НАЯ КРАС"
1 <4,1>	"К"	"АЯ КРАСК"
1 <0,1>	"А"	"Я КРАСКА"

3. LZ78, длина словаря - 16 фраз. Коды сжатого сообщения - $\langle 0,'К'\rangle\langle 0,'Р'\rangle\langle 0,'А'\rangle\langle 0,'С'\rangle\langle 0,'Н'\rangle\langle 3,'Я'\rangle\langle 0,' '\rangle\langle 1,'Р'\rangle\langle 3,'С'\rangle\langle 1,'А'\rangle$.

ВХОДНОЙ КОД	ПЕЧАТЬ (СЛОВАРЬ)	ПОЗИЦИЯ СЛОВАРЯ
		0
<0,'К'>	"К"	1
<0,'Р'>	"Р"	2
<0,'А'>	"А"	3
<0,'С'>	"С"	4
<0,'Н'>	"Н"	5
<3,'Я'>	"АЯ"	6
<0,' '>		7
<1,'Р'>	"КР"	8
<3,'С'>	"АС"	9
<1,'А'>	"КА"	10

4. LZW, длина словаря - 500 фраз. Коды сжатого сообщения - $0'К'0'Р'0'А'0'С'0'Н'0'А'0'Я'0''\langle 256\rangle\langle 258\rangle 0'К'0'А'$.

При распаковке нужно придерживаться следующего правила. Словарь пополняется после считывания первого символа идущего за текущим кода, т.е. из фразы, соответствующей

следующему после раскодированного коду, берется первый символ. Это правило позволяет избежать бесконечного цикла при раскодировании сообщений вида wKwK, где w - фраза, а K - символ. Конкретным примером такого сообщения является любая последовательность трех одинаковых символов, пары которых ранее не встречались.

ВХОДНОЙ КОД	ПЕЧАТЬ	СЛОВАРЬ	ПОЗИЦИЯ СЛОВАРЯ
		ASCII+	0-255
0'К'	"К"	"КР"	256
0'Р'	"Р"	"РА"	257
0'А'	"А"	"АС"	258
0'С'	"С"	"СН"	259
0'Н'	"Н"	"НА"	260
0'А'	"А"	"АЯ"	261
0'Я'	"Я"	"Я "	262
0' '		"К"	263
<256>	"КР"	"КРА"	264
<258>	"АС"	"АСК"	265
0'К'	"К"	"КА"	266
0'А'	"А"		

Упражнение 32 Распаковать каждое приведенное сообщение и рассчитать длину кода каждого сжатого сообщения в битах. Сообщение, сжатое LZ77 (словарь - 12 байт, буфер - 4 байта), - $\langle 0,0,'A'\rangle\langle 0,0,'F'\rangle\langle 0,0,'X'\rangle\langle 9,2,'F'\rangle\langle 8,1,'F'\rangle\langle 6,2,'X'\rangle\langle 4,3,'A'\rangle$. Сообщение, сжатое LZSS (словарь - 12 байт, буфер - 4 байта), - $0'A'0'F'0'X'1\langle 9,2\rangle 1\langle 8,2\rangle 1\langle 6,3\rangle 1\langle 4,4\rangle 1\langle 9,1\rangle$. Сообщеие, сжатое LZ78 (словарь - 16 фраз), - $\langle 0,'A'\rangle\langle 0,'F'\rangle\langle 0,'X'\rangle\langle 1,'F'\rangle\langle 2,'X'\rangle\langle 5,'A'\rangle\langle 3,'A'\rangle\langle 2,'F'\rangle\langle 0,'A'\rangle$. Сообщение, сжатое LZW(словарь - ASCII+ и 16 фраз), - $0'A'0'F'0'X'\langle 256\rangle\langle 257\rangle\langle 257\rangle 0'A'\langle 258\rangle 0'F'0'F'0'A'$

1.6 Лекция №6(2 часа)

Тема: «Программы сжатия» (Интерактивная форма)

1.6.1 Вопросы лекции:

1. Особенности программ- архиваторов
2. Сжатие информации с потерями.

1.6.2 Краткое содержание вопросов:

1. Особенности программ- архиваторов

Архиваторы – это специальные программы, разработанные для создания различных архивов. Как правило, сами архивы используются для хранения разнообразных данных в компактном удобном виде. В качестве таких данных могут выступать папки и файлы. В основном все данные подвергаются предварительной процедуре упаковки или сжатия. Именно поэтому практически любой архиватор также является специальной программой для сжатия всех данных.

Сжатие файлов и данных

Каждая программа, предназначенная для сжатия различных данных, может быть рассмотрена в качестве архиватора. Эффективность сжатия является очень важной характеристикой архиватора. Именно от этого зависит размер любых создаваемых архивов. При этом, чем меньше будет размер архива, тем менее необходимо места для его хранения. Необходима небольшая пропускная способность каналов для передачи или, к примеру, затрачивается меньше времени на архивацию. Все преимущества таких архивов очевидны, особенно если учитывать тот факт, что файлы уменьшаются в размере от 2 до 5 раз. Сжатие различных данных сегодня используются очень широко, практически везде. В качестве примера можно привести документы PDF, которые чаще всего содержат исключительно сжатую информацию. Также большое количество исполняемых файлов EXE сжимаются специальными упаковщиками, а различные мультимедийные файлы (MPG, MP3, JPG, GIF) – это своеобразные архивы.

Упаковка/распаковка файлов

Все программы-архиваторы имеют один определенный недостаток – это отсутствие прямого доступа к различным данным. Для начала файлы необходимо извлекать из архива или же распаковывать. Операция упаковки, как и распаковки, требует определенных системных ресурсов. Такая операция не мгновенная. Именно поэтому все архивы практически всегда применяют с достаточно редко применяемыми данными, для хранения установочных файлов или, к примеру, резервных копий. Сегодня существует большое количество архиваторов, обладающих разной

эффективностью и распространенностью. При этом некоторые очень интересные программы данного типа просто неизвестны большинству потенциальных пользователей.

Кроме различий в большой функциональности программы такого типа делятся на две большие группы: симметричные и асимметричные. Последние для операции распаковки требуют немного оперативной памяти и времени в отличие от операции упаковки. Это позволяет в короткие сроки получать содержимое архивов на довольно маломощных компьютерах. При этом симметричные программы требуют для распаковки и упаковки одинаковый объем памяти, а также одинаковое время. Использование данных программ на разнообразных компьютерах и оперативного доступа к содержимому любых архивов является ограниченным. Самый популярный на сегодня архиватор RAR в своей основе использует именно словарный асимметричный метод сжатия, а вот для текстов можно использовать PPM симметричный метод. В результате распаковка различных RAR архивов, которые сжаты с максимальной степенью, иногда невозможна на компьютерах с довольно ограниченной оперативной памятью.

2. Сжатие информации с потерями.

Говоря о кодах сжатия, различают понятия «сжатие без потерь» и «сжатие с потерями». Очевидно, что когда мы имеем дело с информацией типа «номер телефона», то сжатие такой записи за счет потери части символов не ведет ни к чему хорошему. Тем не менее можно представить целый ряд ситуаций, когда потеря части информации не приводит к потери полезности оставшейся. Сжатие с потерями применяется в основном для графики (JPEG), звука (MP3), видео (MPEG), то есть там, где в силу огромных размеров файлов степень сжатия очень важна, и можно пожертвовать деталями, не существенными для восприятия этой информации человеком. Особые возможности для сжатия информации имеются при компрессии видео. В ряде случаев большая часть изображения передается из кадра в кадр без изменений, что позволяет строить алгоритмы сжатия на основе выборочного отслеживания только части «картинки». В частном случае изображение говорящего человека, не меняющего своего положения, может обновляться только в области лица или даже только рта — то есть в той части, где происходят наиболее быстрые изменения от кадра к кадру.

В целом ряде случаев сжатие графики с потерями, обеспечивая очень высокие степени компрессии, практически незаметно для человека. Так, из трех фотографий,

показанных ниже, первая представлена в TIFF-формате (формат без потерь), вторая сохранена в формате JPEG с минимальным параметром сжатия, а третья с максимальным. При этом можно видеть, что последнее изображение занимает почти на два порядка меньший объем, чем первая.Однако методы сжатия с потерями обладают и рядом недостатков.

Первый заключается в том, что компрессия с потерями применима не для всех случаев анализа графической информации. Например, если в результате сжатия изображения на лице изменится форма родинки (но лицо при этом останется полностью узнаваемо), то эта фотография окажется вполне приемлемой, чтобы послать ее по почте знакомым, однако если пересылается фотоснимок легких на медэкспертизу для анализа формы затемнения — это уже совсем другое дело. Кроме того, в случае машинных методов анализа графической информации результаты кодирования с потерей (незаметные для глаз) могут быть «заметны» для машинного анализатора.

Вторая причина заключается в том, что повторная компрессия и декомпрессия с потерями приводят к эффекту накопления погрешностей. Если говорить о степени применимости формата JPEG, то, очевидно, он полезен там, где важен большой коэффициент сжатия при сохранении исходной цветовой глубины. Именно это свойство обусловило широкое применение данного формата в представлении графической информации в Интернете, где скорость отображения файла (его размер) имеет первостепенное значение. Отрицательное свойство формата JPEG — ухудшение качества изображения, что делает практически невозможным его применение в полиграфии, где этот параметр является определяющим.

Теперь перейдем к разговору о сжатии информации без потерь и рассмотрим, какие алгоритмы и программы позволяют осуществлять эту операцию.

1.7 Лекция № 7 (2 часа)

Тема: «Помехозащитное кодирование. Матричное кодирование и групповые коды»

1.7.1 Вопросы лекции:

1. Информационный канал.

2. Помехозащитное кодирование в двоичном симметричном канале.

3. Матричное кодирование.

4. Групповые коды.

5. Совершенные и квазисовершенные коды.

1.7.2 Краткое содержание вопросов:

1. Информационный канал.

Информационные каналы различаются по своей пропускной способности.

Пропускная способность – это количество информации, передаваемое каналом в единицу времени. Измеряется пропускная способность в бит/с. В честь изобретателя телеграфа этой единице было дано имя Бод:

1 Бод = 1 бит/с.

Пропускная способность информационного канала определяется двумя параметрами: разрядностью и частотой. Она пропорциональна их произведению.

Разрядностью называют максимальное количество информации, которое может быть одновременно помещено в канал.

Частота показывает, сколько раз информация может быть помещена в канал в течение единицы времени.

Разрядность почтового канала огромна. Так, пересылая по почте, например, лазерный диск, можно поместить одновременно в канал более 600 Мб информации. В то же время частота почтового канала очень низкая – выемка почты из ящиков происходит не чаще пяти раз в сутки.

Телефонный канал информации однобитный: одновременно по телефонному проводу можно послать или единицу (ток, импульс), или ноль. Частота этого канала может достигать десятки и сотни тысяч циклов в секунду. Это свойство телефонной сети позволяет использовать ее для связи между компьютерами.

2. Помехозащитное кодирование в двоичном симметричном канале.

Простейший код для борьбы с шумом - это контроль четности, он, в частности, широко используется в модемах. Кодированиезаключается в добавлении к каждому байту девятого бита таким образом, чтобы дополнить количество единиц в байте до заранее выбранного для кода четного (even) или нечетного (odd) значения. Используя этот код, можно лишь обнаруживать большинство ошибок.

Простейший код, исправляющий ошибки, - это тройное повторение каждого бита. Если с ошибкой произойдет передача одного бита из трех, то ошибка будет исправлена, но если случится двойная или тройная ошибка, то будут получены неправильные данные. Часто коды для исправления ошибок используют совместно с кодами для обнаружения ошибок.

При тройном повторении для повышения надежности три бита располагают не подряд, а на фиксированном расстоянии друг от друга. Использование тройного повторения естественно значительно снижает скорость передачи данных.

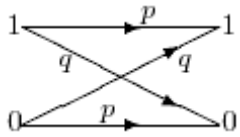


Рис. 7.6.

Двоичный симметричный канал изображен на рис. 7.6, где p - это вероятность безошибочной передачи бита, а q -вероятность передачи бита с ошибкой. Предполагается, что в таком канале ошибки происходят независимо. Далее рассматриваются только такие каналы.

Двоичный симметричный канал реализует схему Бернулли, поэтому вероятность передачи n бит по двоичному симметричному каналу с k ошибками равна

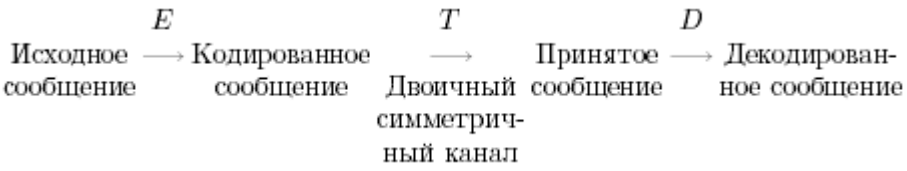
$$P_n(k) = C_n^k p^{n-k} q^k.$$

Пример. Вероятность передачи одного бита информации с ошибкой равна $q = 0.01$ и нас интересует вероятностьбезошибочной передачи 1000 бит (125 байт). Искомую вероятность можно подсчитать по формуле $P_{1000}(0) = C_{1000}^0 p^{1000} q^0 = 0.99^{1000} \approx 4.32 * 10^{-5}$, т.е. она ничтожно мала.

Добиться минимальности вероятности ошибки при передаче данных можно используя специальные коды. Обычно используют помехозащитные коды. Идея систематических кодов состоит в добавлении к символам исходных кодов, предназначенных для передачи в канале, нескольких контрольных символов по определенной схеме кодирования. Принятая такая удлиненная последовательность кодов декодируется по схеме декодирования в первоначально переданную. Приемник способен распознавать и/или исправлять ошибки, вызванные шумом, анализируя дополнительную информацию, содержащуюся в удлиненных кодах.

Двоичным (m,n)-кодом называется пара, состоящая из схемы кодирования $E\colon \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ и схемы декодирования $D\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^m$, где $\mathbb{Z}_2^n$ - множество всех двоичных последовательностей длины n, $m < n$ (случай $m = n$ рассматривается в криптографии).

Функции D и E выбираются так, чтобы функция $H = D \circ T \circ E$, где T - **функция ошибок**, с вероятностью, близкой к единице, была тождественной. Функции D и E считаются безошибочными, т.е. функция $D \circ E$ - тождественная (См. рис. 7.7).



3. Матричное кодирование.

Ранее каждая схема кодирования описывалась таблицами, задающими кодовое слово длины n для каждого исходного слова длины m. Для блоков большой длины этот способ требует большого объема памяти и поэтому непрактичен. Например, для $(16, 33)$-кода потребуется $33 * 2^{16} = 2\,162\,688$ бит.

Гораздо меньшего объема памяти требует **матричное кодирование**. Пусть E матрица размерности $m \times n$, состоящая из элементов e_{ij}, где i - это номер строки, а j - номер столбца. Каждый из элементов матрицы e_{ij} может быть либо 0, либо 1.Кодирование реализуется операцией $b = aE$ или $b_j = a_1e_{1j} + a_2e_{2j} + \cdots + a_me_{mj}$, где кодовые слова рассматриваются как векторы, т.е как матрицы-строки размера $1 \times n$.

Пример. Рассмотрим следующую 3×6-матрицу:

$$E = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Тогда кодирование задается такими отображениями: $000 \to 000000$, $001 \to 001111$, $010 \to 010011$, $011 \to 011100$, $100 \to 100110$, $101 \to 101001$, $110 \to 110101$, $111 \to 111010$.

Рассмотренный пример показывает преимущества матричного кодирования: достаточно запомнить m кодовых слов вместо 2^m слов. Это общий факт.

Кодирование не должно приписывать одно и то же кодовое слово разным исходным сообщениям. Простой способ добиться этого состоит в том, чтобы m столбцов (в предыдущем примере - первых) матрицы E образовывали единичную матрицу. При умножении любого вектора на единичную матрицу получается этот же самый вектор, следовательно, разным векторам-сообщениям будут соответствовать разные вектора систематического кода.

Матричные коды называют также линейными кодами. Для линейных $(n-r,n)$ -кодов с минимальным расстоянием Хэмминга d существует **нижняя граница Плоткина** (Plotkin)[3] для минимального количества контрольных разрядов r при $n \geqslant 2d-1$,
$r \geqslant 2d-2-\log_2 d.$

4. Групповые коды.

Наиболее широкое распространение в настоящее время получили систематические коды, называемые также линейными блочными кодами. В общем случае линейные коды определяются как пространство в поле GF(q). Двоичные линейные коды носят название групповых.

Групповым кодом называют такой код, множество кодовых комбинаций которого образует группу относительно операции сложения по модулю 2. Групповая структура кода обеспечивает ему ряд важных свойств.

Свойство 1. Минимальное кодовое расстояние группового кода равно минимальному весу его ненулевых комбинаций, т.е. если {vi} множество разрешенных комбинаций, а wi вес комбинации vi, то dmin=wi.

Кодовое расстояние между комбинациями определяется как вес двух комбинаций. В силу свойства замкнутости сумма двух комбинаций также является кодовой комбинацией, поэтому таблица кодовых расстояний группового кода однозначно соответствует таблице весов разрешенных комбинаций. Следовательно, минимальному расстоянию Хэмминга соответствует ненулевая комбинация с минимальным весом.

Свойство 2 (граница Синглтона). Минимальное кодовое расстояние группового кода связано с количеством проверочных разрядов неравенством dmin < nk+1. Имеется комбинация группового кода с одним ненулевым информационным и nk проверочными элементами. Такая комбинация не может иметь вес, больший nk+1, что соответствует неравенству. Концепция защищенной ВС Курс лекций по информатике

Граница Синглтона показывает, что для исправления t ошибок код должен иметь не менее 2t проверочных элементов (два проверочных разряда на ошибку).

Для групповых кодов введено специальное обозначение (n, k) код.

Способы задания групповых кодов.

Существует три эквивалентных способа задания групповых кодов.

1) Перечисление кодовых комбинаций, т.е. составление списка всех разрешенных комбинаций. В этом случае, зная k информационных разрядов, из списка выбирается соответствующее n элементное слово.

2) С помощью системы проверочных соотношений определяются правила формирования проверочных элементов по известным информационным.

Пусть информационная последовательность имеет вид (ak1, ak2, ..., a0), тогда система линейных уравнений имеет вид:

$$
\begin{cases}
ak = \sum_{i=0}^{k1} h0, r+i * ai \\
ak+1 = \sum_{i=0}^{k1} h1, r+i * ai \\
an1 = \sum_{i=0}^{k1} hrj, r+i * ai
\end{cases} \quad (1)
$$

где

h(i=0...k1, j=r...r+k1) двоичные символы 0 или 1, значения которых определяются правилами образования конкретных групповых кодов, а ak, ..., an1, проверочные элементы.

Система уравнений (1) задает правила кодирования и обнаружения ошибок для групповых кодов, на ее основе можно построить кодирующее устройство и устройство

обнаружения ошибок. При программной реализации на основе (1) разрабатываются программы процедур кодирования и декодирования.

Прикладной протокол (в стандартах STEP) - информационная модель определенного приложения, которая описывает с высокой степенью полноты множество сущностей, имеющихся в приложении, вместе с их атрибутами, и выражена средствами языка Express

5. Совершенные и квазисовершенные коды.

Групповой (m, n) -код, исправляющий все ошибки веса, не большего k , и никаких других, называется совершенным.

Свойства совершенного кода2:

Для совершенного (m, n) -кода, исправляющего все ошибки веса, не большего k , выполняется соотношение $\sum_{i=0}^{k} C_n^i = 2^{n-m}$. Верно и обратное утверждение;

Совершенный код, исправляющий все ошибки веса, не большего k , в столбцах таблицы декодирования содержит все слова, отстоящие от кодовых на расстоянии, не большем k . Верно и обратное утверждение;

Таблица декодирования совершенного кода, исправляющего все ошибки в не более чем k позициях, имеет в качестве лидеров все строки, содержащие не более k единиц. Верно и обратное утверждение.

Совершенный код - это лучший код, обеспечивающий максимум минимального расстояния между кодовыми словами при минимуме длины кодовых слов. Совершенный код легко декодировать: каждому полученному слову однозначно ставится в соответствие ближайшее кодовое. Чисел m , n и k $(1 < k < \frac{n-1}{2})$, удовлетворяющих условию совершенности кода очень мало. Но и при подобранных m , n и k совершенный код можно построить только в исключительных случаях.

Если m , n и k не удовлетворяют условию совершенности, то лучший групповой код, который им соответствует называется квазисовершенным, если он исправляет все ошибки кратности, не большей k , и некоторые ошибки кратности $k+1$. Квазисовершенных кодов также очень мало.

Двоичный блочный (m, n) -код называется оптимальным, если он минимизирует вероятность ошибочного декодирования. Совершенный или квазисовершенный код - оптимален. Общий способ построения оптимальных кодов пока неизвестен.

Для любого целого положительного числа r существует совершенный (m, n) -код, исправляющий одну ошибку, называемый кодом Хэмминга (Hamming), в котором $m = 2^r - r - 1$ и $n = 2^r - 1$.

Действительно, $\sum_{i=0}^{1} C_n^i = 1 + 2^r - 1 = 2^r = 2^{n-m}$.

Порядок построения кода Хэмминга следующий:

Выбираем целое положительное число r . Сообщения будут словами длины $m = 2^r - r - 1$, а кодовые слова - длины $n = 2^r - 1$;

В каждом кодовом слове $b = b_1 b_2 \dots b_n$ r бит с индексами-степенями двойки $(2^0, 2^1, \dots, 2^{r-1})$ - являются контрольными, остальные - в естественном порядке - битами сообщения. Например, если $r = 4$, то биты b_1, b_2, b_4, b_8 - контрольные, а $b_3, b_5, b_6, b_7, b_9, b_{10}, b_{11}, b_{12}, b_{13}, b_{14}, b_{15}$ - из исходного сообщения;

Строится матрица M из $2^r - 1$ строк и r столбцов. В i -ой строке стоят цифры двоичного представления числа i . Матрицы для r=2, 3 и 4 таковы:

$$M_{3\times 2} = \begin{bmatrix} 01 \\ 10 \\ 11 \end{bmatrix} \qquad M_{7\times 3} = \begin{bmatrix} 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{bmatrix} \qquad M_{15\times 4} = \begin{bmatrix} 0001 \\ 0010 \\ 0011 \\ 0100 \\ 0101 \\ 0110 \\ 0111 \\ 1000 \\ 1001 \\ 1010 \\ 1011 \\ 1100 \\ 1101 \\ 1110 \\ 1111 \end{bmatrix} ;$$

Записывается система уравнений $bM = 0$, где M - матрица из предыдущего пункта. Система состоит из r уравнений. Например, для $r = 3$:

$$\begin{cases} b_4 + b_5 + b_6 + b_7 = 0 \\ b_2 + b_3 + b_6 + b_7 = 0 \\ b_1 + b_3 + b_5 + b_7 = 0 \end{cases};$$

Чтобы закодировать сообщение a, берутся в качестве b_j, j не равно степени двойки, соответствующие биты сообщения и отыскиваются, используя полученную систему уравнений, те b_j, для которых j - степень двойки. В каждое уравнение входит только одно b_j, $j = 2^i$. В выписанной системе b_4 входит в 1-е уравнение, b_2 - во второе и b_1 - в третье. В рассмотренном примере сообщение $a = 0111$ будет закодировано кодовым словом $b = 0001111$.

Декодирование кода Хэмминга проходит по следующей схеме. Пусть принято слово $b + e$, где b - переданное кодовое слово, а e - строка ошибок. Так как $bM = 0$, то $(b + e)M = bM + eM = eM$. Если результат нулевой, как происходит при правильной передаче, считается, что ошибок не было. Если строка ошибок имеет единицу в i -й позиции, то результатом произведения eM будет i -я строка матрицы M или двоичное представление числа i. В этом случае следует изменить символ в i -й позиции слова $b + e$, считая позиции слева, с единицы.

Пример. $(4, 7)$ -код Хэмминга имеет в качестве одного из кодовых слов $b = 0001111$. Матрица $M_{7\times3}$ приведена на шаге 3 хода построения кода Хэмминга. Ясно, что $bM = 0$. Добавим к b строку ошибок $e = 0010000$. Тогда $b + e = 0011111$ и $(b + e)M = 011 = 3_{10}$, т.е. ошибка находится в третьей позиции. Если $e = 0000001$, то $b + e = 0001110$ и позиция ошибки - $(b + e)M = 111 = 7_{10}$ и т.п. Если ошибка допущена в более чем в одной позиции, то декодирование даст неверный результат.

1.8 Лекция №8(2 часа)

Тема : «Полиномиальные коды »

1.8.1 Вопросы лекции:

1. Принцип построения полиномиальных кодов.

2. Линейные коды.

3. Циклические коды.

4. Исправление ошибок при построении кодов.

1.8.2 Краткое содержание вопросов:

1. Принцип построения полиномиальных кодов.

Код, в котором кодовая комбинация, полученная путем циклического сдвига разрешенной кодовой комбинации является также разрешенной кодовой комбинацией, называется циклическим (полиномиальным, кодом с циклическими избыточными проверками-ЦИП).

Сдвиг осуществляется справа налево, при этом крайний левый символ переносится в конец комбинации.

Циклический код относится к линейным, блочным, корректирующим, равномерным кодам.

В циклических кодах кодовые комбинации представляются в виде многочленов, что позволяет свести действия над кодовыми комбинациями к действием над многочленами (используя аппарат полиномиальной алгебры).

Циклические коды являются разновидностью систематических кодов и поэтому обладают всеми их свойствами. Первоначально они были созданы для упрощения схем кодирования и декодирования. Их эффективность при обнаружении и исправлении ошибок обеспечила им широкое применение на практике.

Принцип построения циклических кодов

Идея построения циклических кодов базируется на использовании неприводимых многочленов. Неприводимым называется многочлен, который не может быть представлен в виде произведения многочленов низших степеней, т.е. такой многочлен делиться только на самого себя или на единицу и не делиться ни на какой другой многочлен. На такой многочлен делиться без остатка двучлен xn+1.Неприводимые многочлены в теории циклических кодов играют роль образующих полиномов.

Чтобы понять принцип построения циклического когда, умножаем комбинацию простого k-значного кода Q(x) на одночлен xr ,а затем делим на образующий полином P(x) , степень которого равна r. В результате умножения Q(x) на xr степень каждого одночлена, входящего в Q(x), повышается на r. При делении произведения xrQ(x) на

образующий полином получается частное C(x) такой же степени, как и Q(x).Результат можно представить в вид

$$\frac{Q(x)\,x^r}{P(x)} = C(x) + \frac{R(x)}{P(x)}, \qquad (1)$$

где R(x) - остаток от деления Q(x) xr на P(x).

Частное C(x) имеет такую же степень, как и кодовая комбинация Q(x) простого кода, поэтому C(x) является кодовой комбинацией этого же простого k-значного кода. Следует заметить,что степень остатка не может быть больше степени образующего полинома, т.е. его наивысшая степень может быть равна (r-1). Следовательно, наибольшее число разрядов остатка R(x) не превышает числа r.

Умножая обе части равенства (1) на P(x) и произведя некоторые перестановки получаем :

$$F(x) = C(x)\,P(x) = Q(x)\,x^r + R(x) \qquad (2)$$

Таким образом, кодовая комбинация циклического n-значного кода может быть получена двумя способами:

1) умножение кодовой комбинации Q(x) простого кода на одночлен xr

и добавление к этому произведению остатка R(x) , полученного в результате деления произведения Q(x) xr на образующий полином P(x);

2) умножения кодовой комбинации C(x) простого k-значного на образующий полином P(x).

При построении циклических кодов первым способом расположение информационных символов во всех комбинациях строго упорядочено - они занимают k старших разрядов комбинации, а остальные (n-k) разрядов отводятся под контрольные.

При втором способе образования циклических кодов информационные и контрольные символы в комбинациях циклического кода не отделены друг от друга, что затрудняет процесс декодирования.

2. Линейные коды.

В области математики и теории информации линейный код — это важный тип блокового кода, использующийся в схемах определения и коррекции ошибок. Линейные коды, по сравнению с другими кодами, позволяют реализовывать более эффективные алгоритмы кодирования и декодирования информации.

В процессе хранения данных и передачи информации по сетям связи неизбежно возникают ошибки. Контроль целостности данных и исправление ошибок — важные задачи на многих уровнях работы с информацией (в частности, физическом, канальном, транспортном уровнях модели OSI).

В системах связи возможны несколько стратегий борьбы с ошибками:

обнаружение ошибок в блоках данных и автоматический запрос повторной передачи поврежденных блоков — этот подход применяется в основном на канальном и транспортном уровнях;

обнаружение ошибок в блоках данных и отбрасывание поврежденных блоков — такой подход иногда применяется в системах потокового мультимедиа, где важна задержка передачи и нет времени на повторную передачу;

исправление ошибок (англ. forward error correction) применяется на физическом уровне.

3. Циклические коды.

Циклический код — линейный код, обладающий свойством цикличности, то есть каждая циклическая перестановка кодового слова также является кодовым словом. Используется для преобразования информации для защиты её от ошибок (см. Обнаружение и исправление ошибок).

Пусть $\overline{y} = (y_0, y_1, \ldots, y_{n-1}) \in L^n$ слово длины n над алфавитом из элементов конечного поля $L = \mathbb{GF}(q)$ и $y(x) = y_0 + y_1 x + y_2 x^2 + \ldots + y_{n-1} x^{n-1}$ полином, соответствующий этому слову, от формальной переменной x. Видно, что это соответствие является изоморфизмом линейных пространств. Так как «слова» состоят из букв из поля, то их можно складывать и умножать (поэлементно), причём результат будет в том же поле. Полином, соответствующий линейной комбинации $\overline{y} = m_1\overline{y_1} + m_2\overline{y_2}$ пары слов $\overline{y_1} = (y_{1,0}, \ldots, y_{1,n-1})$ и $\overline{y_2} = (y_{2,0}, \ldots, y_{2,n-1})$, равен линейной комбинации

$$\overline{y}(x) = \sum_{k=0}^{n-1} (m_1 y_{1k} + m_2 y_{2k}) x^k = m_1\overline{y_1}(x) + m_2\overline{y_2}(x)$$

полиномов этих слов

Это позволяет рассматривать множество слов длины n над конечным полем как линейное пространство полиномов со степенью не выше n-1 над полем $\mathbb{GF}(q)$

4. Исправление ошибок при построении кодов.

В процессе хранения данных и передачи информации по сетям связи неизбежно возникают ошибки. Контроль целостности данных и исправление ошибок — важные

задачи на многих уровнях работы с информацией (в частности, физическом, канальном, транспортном уровнях сетевой модели OSI)

В системах связи возможны несколько стратегий борьбы с ошибками:

обнаружение ошибок в блоках данных и автоматический запрос повторной передачи повреждённых блоков — этот подход применяется, в основном, на канальном и транспортном уровнях;

обнаружение ошибок в блоках данных и отбрасывание повреждённых блоков — такой подход иногда применяется в системах потокового мультимедиа, где важна задержка передачи и нет времени на повторную передачу;

исправление ошибок (англ. forward error correction) применяется на физическом уровне.

Корректирующие коды — коды, служащие для обнаружения или исправления ошибок, возникающих при передаче информации под влиянием помех, а также при её хранении.

Для этого при записи (передаче) в полезные данные добавляют специальным образом структурированную избыточную информацию (контрольное число), а при чтении (приёме) её используют для того, чтобы обнаружить или исправить ошибки. Естественно, что число ошибок, которое можно исправить, ограничено и зависит от конкретного применяемого кода.

С кодами, исправляющими ошибки, тесно связаны коды обнаружения ошибок. В отличие от первых, последние могут только установить факт наличия ошибки в переданных данных, но не исправить её.

В действительности, используемые коды обнаружения ошибок принадлежат к тем же классам кодов, что и коды, исправляющие ошибки. Фактически любой код, исправляющий ошибки, может быть также использован для обнаружения ошибок (при этом он будет способен обнаружить большее число ошибок, чем был способен исправить).

По способу работы с данными коды, исправляющие ошибки, делятся на блоковые, делящие информацию на фрагменты постоянной длины и обрабатывающие каждый из них в отдельности, и свёрточные, работающие с данными как с непрерывным потоком.

1.9 Лекция №9 (2 часа)

Тема: «Основы теории защиты информации»

1.9.1 Вопросы лекции

1. Криптография.

2. Криптосистема без передачи ключей.

3. Криптосистема с открытым ключом.

4. Электронная подпись.

5. Стандарт шифрования данных.

6. Кодировка текстовой информации.

1.9.2 Краткое содержание вопросов

1. Криптография.

) — наука о методах обеспечения конфиденциальности (невозможности прочтения информации посторонним), целостности данных (невозможности незаметного изменения информации), аутентификации (проверки подлинности авторства или иных свойств объекта), а также невозможности отказа от авторства.

Изначально криптография изучала методы шифрования информации — обратимого преобразования открытого (исходного) текста на основе секретного алгоритма или ключа в шифрованный текст (шифротекст). Традиционная криптография образует раздел симметричных криптосистем, в которых зашифрование и расшифрование проводится с использованием одного и того же секретного ключа. Помимо этого раздела современная криптография включает в себя асимметричные криптосистемы, системы электронной цифровой подписи (ЭЦП), хеш-функции, управление ключами, получение скрытой информации, квантовую криптографию.

Криптография не занимается защитой от обмана, подкупа или шантажа законных абонентов, кражи ключей и других угроз информации, возникающих в защищённых системах передачи данных.

2. Криптосистема без передачи ключей.

Основу данного метода криптографии составляют теоремы Эйлера и Ферма, а также алгоритмы решения сравнений первой степени, базирующиеся, в свою очередь, на алгоритме Евклида.

Сущность алгоритма Евклида состоит в отыскании нод(a,b) и заключается в следующем. Пусть a и bположительные целые числа и $a > b$. Тогда для указанных значений будут справедливы соотношения:

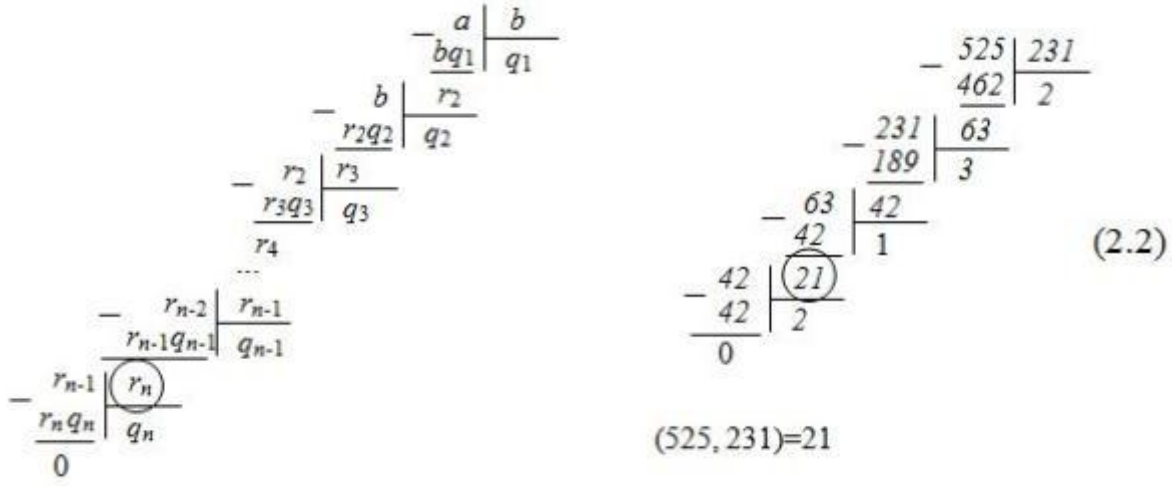
$$a = bq_1 + r_2, \qquad 0 < r_2 < b,$$

$$b = r_2 q_2 + r_3, \qquad 0 < r_3 < r_2,$$

$$r_2 = r_3 q_3 + r_4, \qquad 0 < r_4 < r_3,$$
$$\cdots\cdots\cdots\cdots \qquad (2.1)$$
$$r_{n-2} = r_{n-1} q_{n-1} + r_n, \qquad 0 < r_n < r_{n-1},$$
$$r_{n-1} = r_n q_n.$$

Алгоритм заканчивается при получении некоторого остатка $r_{n+1} = 0.$ Наибольшим общим делителем тогда будет последний не равный нулю остаток r_n.

Алгоритм Евклида можно представить также в виде следующих арифметических действий:



(2.2)

В теории существует математическая связь данного алгоритма и представления частного от деления двух чисел в виде полиномиальных дробей:

$$P_i = q_i P_{i-1} + P_{i-2},$$
$$Q_i = q_i Q_{i-1} + Q_{i-2}, \qquad P_0 = 1,\ Q_0 = 0,\ Q_1 = 1. \qquad (2.3)$$

Из данных равенств следует:

$$\frac{P_1}{Q_1} = q_1, \qquad \frac{P_2}{Q_2} = q_1 + \frac{1}{q_2}, \qquad \frac{P_3}{Q_3} = q_1 + \frac{1}{q_2 + \frac{1}{q_3}}, \qquad (2.4)$$

$$\frac{P_4}{Q_4} = q_1 + \frac{1}{q_2 + \frac{1}{q_3 + \frac{1}{q_4}}}, \quad \cdots \quad \frac{P_n}{Q_n} = q_1 + \frac{1}{q_2 + \frac{1}{\cdots + \frac{1}{q_{n-1} + \frac{1}{q_n}}}} = \frac{a}{b}.$$

3. Криптосистема с открытым ключом.

Криптографическая система с открытым ключом (или асимметричное шифрование, асимметричный шифр) — система шифрования и/или электронной подписи (ЭП), при которой открытый ключ передаётся по открытому (то есть незащищённому, доступному для наблюдения) каналу и используется для проверки ЭП и для шифрования сообщения. Для генерации ЭП и для расшифровки сообщения используется закрытый ключ[1]. Криптографические системы с открытым ключом в настоящее время широко применяются в различных сетевых протоколах, в частности, в протоколах TLS и его предшественнике SSL (лежащих в основе HTTPS), в SSH. Также используется в PGP, S/MIME.

Пусть K — пространство ключей, а e и d — ключи шифрования и расшифрования соответственно. E_e — функция шифрования для произвольного ключа e\inK, такая что:

E_e(m)=c

Здесь c\inC, где C — пространство шифротекстов, а m\inM, где M — пространство сообщений.

D_d — функция расшифрования, с помощью которой можно найти исходное сообщение m, зная шифротекст c :

D_d(c)=m

{E_e: e\inK} — набор шифрования, а {D_d: d\inK} — соответствующий набор для расшифрования. Каждая пара (E, D) имеет свойство: зная E_e, невозможно решить уравнение E_e(m)=c, то есть для данного произвольного шифротекста c\inC, невозможно найти сообщение m\inM. Это значит, что по данному e невозможно определить соответствующий ключ расшифрования d. E_e является односторонней функцией, а d — лазейкой.[3]

4. Электронная подпись.

(ЭЦП) — реквизит электронного документа, полученный в результате криптографического преобразования информации с использованием закрытого ключа подписи и позволяющий проверить отсутствие искажения информации в электронном документе с момента формирования подписи (целостность), принадлежность подписи владельцу сертификата ключа подписи (авторство), а в случае успешной проверки подтвердить факт подписания электронного документа (неотказуемость).

Электронная подпись предназначена для определения лица, подписавшего электронный документ, и является аналогом собственноручной подписи в случаях, предусмотренных законом.

Электронная подпись применяется при совершении гражданско-правовых сделок, оказании государственных и муниципальных услуг, исполнении государственных и муниципальных функций, при совершении иных юридически значимых действий.

5. Стандарт шифрования данных.

Стандарты шифрования данных — совокупность правил, используемых в мощной алгоритмической технике кодировании информации.

Алгоритм Data Encryption Standart использует множество подстановок и перестановок, производит шифрование 64-битовых блоков с помощью ключа размером 64 бит. Основными являются 56 бит, остальные 8 бит — контрольные. Дешифрование в Data Encryption Standart — процедура, обратная шифрованию, производится повтором операций в обратном порядке. Шифрование заключается в перестановке бит 64-битового блока, 16 циклах, и в конечной перестановке битов. Расшифрование является обратным процессу шифрования.

Data Encryption Standart подходит для шифрования и аутентификации данных. Он позволяет преобразовывать открытый текст в 64- битовый выходной зашифрованный текст.

Для того чтобы пользоваться алгоритмом Data Encryption Standart для решения задач разработаны 4 режима: электронная кодовая книга ECB (Electronic Code Book), сцепление блоков шифра CFB (Cipher Feed Back) и обратная связь по выходу OFB (Output Feed Back). Обратная связь по зашифрованному тексту и по выходу используются в качестве поточных шрифтов. Это происходит так: каждый бит из последующего потока шифруется отдельно с использованием ключа и ранее закодированной информации.

6. Кодировка текстовой информации.

Символы на экране вашего компьютера формируются на основе двух вещей — наборов векторных форм (представлений) всевозможных символов (они находятся в файлах со шрифтами, которые установлены на вашем компьютере) и кода, который позволяет выдернуть из этого набора векторных форм (файла шрифта) именно тот символ, который нужно будет вставить в нужное место.

Кодировка ASCII

Для начала немного посчитаем. Помните, что такое бит? Это минимальный носитель информации, ноль или один. А байт содержит восемь битов. Сколько может быть комбинаций из нулей и единиц длины 8? Ответ – 2*2*2*2*2*2*2*2=256. Именно столько значений может принимать один байт. Иногда еще байт называют символом – потому что как раз для кодировки символа и стали использовать один байт. Даже меньше, изначально была придумана кодировка ASCII, которая использовала 7 битов – в первые 128 значений можно было вольготно разместить английский алфавит в обоих регистрах, диакритические знаки, цифры и набор спец-символов. И эта кодировка действительно стала универсальной, поэтому англоязычные пользователи крайне редко могут испытывать проблемы с кодировкой.

Кодировка ASCII (American Standard Code for Information Interchange, которая по русски обычно произносится как «аски») описывает первые 128 символов из наиболее часто используемых англоязычными пользователями — латинские буквы, арабские цифры и знаки препинания. Так же еще в эти 128 символов кодировки ASCII попадали некоторые служебные символы, навроде скобок, решеток, звездочек и т.п. Именно эти 128 символов из первоначального вариант ASCII стали стандартом, и в любой другой кодировке текста вы их обязательно встретите и стоять они будут именно в таком порядке. Но дело в том, что с помощью одного байта информации можно закодировать не 128, а целых 256 различных значений (двойка в степени восемь равняется 256), поэтому вслед за базовой версией ASCII появился целый ряд расширенных кодировок ASCII, в которых можно было кроме 128 основных символов закодировать еще и символы национальной кодировки (например, русской).

Кодировка КОИ-8R

Принцип работы кодировки KOI-8R такой— каждый символ текста кодируется одним единственным байтом.

Среди особенностей кодировки KOI-8R можно отметить то, что русские буквы в ее таблице идут не в алфавитном порядке. В кодировке KOI-8R русские буквы расположены в тех же ячейках таблицы, что и созвучные им буквы латинского алфавита из первой части таблицы ASCII. Это было сделано для удобства перехода с русских символов на латинские путем отбрасывания всего одного бита (два в седьмой степени или 128).

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРОВЕДЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

2.1 Практическое занятие №1-2 (4 часа).

Тема: «Понятие информации. Передача информации. Измерение информации. Формальное представление знаний.»

2.1.1 Задание для работы:

1. Понятие информации
2. Передача информации
3. Измерение информации
4. Формальное представление знаний

2.1.2 Краткое описание проводимого занятия:

Общее число неповторяющихся сообщений, которое может быть составлено из алфавита m путем комбинирования по n символов в сообщении,

$$N = m^n. \quad (1)$$

Неопределенность, приходящаяся на символ первичного (кодируемого)[1] алфавита, составленного из равновероятных и взаимонезависимых символов,

$$H = \log m. \quad (2)$$

Основание логарифма влияет лишь на удобство вычисления. В случае оценки энтропии:

а) в двоичных единицах

$$H = \log_2 m \ бит/символ;$$

б) в десятичных единицах

$$H = \lg m \ дит/символ,$$

где $\log_2 m = 3{,}32 \lg m$, $1 \ бит \approx 0{,}3 \ дит$;

в) в натуральных единицах

$$H = \ln m \ нат/символ,$$

где $\log_2 m = 1{,}443 \ln m$, $1 \ бит \approx 0{,}693 \ нат$.

Так как информация есть неопределенность, снимаемая при получении сообщения, то количество информации может быть представлено как произведение общего числа сообщений k на среднюю энтропию H, приходящуюся на одно сообщение:

$$I = kH \ бит. \quad (3)$$

Для случаев равновероятных и взаимонезависимых символов первичного алфавита количество информации в k сообщениях алфавита m равно

$$I = k \log_2 m \ бит.$$

Для неравновероятных алфавитов энтропия на символ алфавита

$$H = \sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i} = -\sum_{i=1}^{m} p_i \log_2 p_i \ \textit{бит/символ}, \qquad (4)$$

а количество информации в сообщении, составленном из k неравновероятных символов,

$$I = -k \sum_{i=1}^{m} p_i \log_2 p_i \ \textit{бит}. \qquad (5)$$

При решении задач, в которых энтропия вычисляется как сумма произведений вероятностей на их логарифм, вероятности всегда должны представлять группу полных событий, независимо от того, являются ли они безусловными $p(a_i)$, условными $p(a_i/b_j)$ или вероятностями совместных событий $p(a_i, b_j)$.

Количество информации определяется исключительно характеристиками первичного алфавита, объем — характеристиками вторичного алфавита. Объем[1] информации

$$Q = kl_{ср}, \qquad (6)$$

где $l_{ср}$ — средняя длина кодовых слов вторичного алфавита. Для равномерных кодов (все комбинации кода содержат одинаковое количество разрядов)

$$Q = kn,$$

где n — длина кода (число элементарных посылок в коде). Согласно (3), объем равен количеству информации, если $l_{ср} = H$, т. е. в случае максимальной информационной нагрузки на символ сообщения. Во всех остальных случаях $I < Q$.

Например, если кодировать в коде Бодо (см. приложение 3) некоторый равновероятный алфавит, состоящий из 32 символов, то

$$I = kH = k \log_2 m = k \log_2 32 = k \cdot 5 \ \textit{бит};$$

$$Q = kl_{ср} = k \cdot 5.$$

Если кодировать в коде Бодо русский 32-буквенный алфавит, то без учета корреляции между буквами количество информации

$$I = kH = k \cdot 4{,}358 \ \textit{бит}; \ Q = k \cdot 5; \ Q > I,$$

т. е. если в коде существует избыточность и $H \neq H_{макс}$, то объем в битах всегда больше количества информации в тех же единицах.

Задача 1.1. Известно, что одно из k возможных сообщений, передаваемых равномерным двоичным кодом, несет 3 бита информации. Чему равно k?

Задача 1.2. Как определить количество информации в одном сообщении, если известно максимально возможное количество сообщений? Как определить количество информации, если известно количество качественных признаков, из которых составлены сообщения, и известно количество символов в каждом сообщении? Привести примеры.

2.1.3 Результаты и выводы:

(По данной форме необходимо представить все практические занятия)

2.2 Практическое занятие №3-4 (4 часа).

Тема: «Виды информации. Хранение информации. Свойства информации. Емкость канала связи. Способы измерения информации.»

2.2.1 Задание для работы:

1. Хранение информации

2. Свойства информации

2.2.2 Краткое описание проводимого занятия:

Потери информации в каналах связи с шумами обычно описывают при помощи условной энтропии и энтропии объединения.

Если помех нет или их уровень настолько низок, что они не в состоянии уничтожить сигнал или имитировать полезный сигнал в отсутствие передачи, то при передаче a_i мы будем твердо уверены, что получим b_j — сигнал, соответствующий переданному a_i-му сигналу. События A и B статистически жестко связаны, условная вероятность максимальна $p(b_j/a_i) = 1$, а условная энтропия

$$H(A/B) = -\sum_{i=1}^{m} p(b_j/a_i) \log p(b_j/a_i) = 0, \qquad (23)$$

так как $\log p(b_j/a_i) = \log 1 = 0$. В этом случае количество информации, содержащееся в принятом ансамбле сообщений B, равно энтропии передаваемых сообщений ансамбля A, т. е. $I(B, A) = H(A)$[1].

При высоком уровне помех любой из принятых сигналов b_j может соответствовать любому переданному сигналу a_i, статистическая связь между переданными и принятыми сигналами отсутствует.

В этом случае вероятности $p(a_i)$ и $p(b_j)$ есть вероятности независимых событий и $p(b_j/a_i) = p(b_j)$; $p(a_i/b_j) = p(a_i)$.

$$H(A/B) = -\sum_i \sum_j p(b_j)\, p(a_i/b_j) \log p(a_i/b_j) =$$

$$= -\sum_i \sum_j p(b_j)\, p(a_i) \log p(a_i) = \sum_j p(b_j)\, H(A) = H(A),$$

так как $\sum_j p(b_j) = 1$, т. е. условная энтропия равна безусловной, а количество информации, содержащееся в B, относительно A равно нулю:

$$I(A, B) = H(A) - H(A/B) = 0.$$

Информационные характеристики реальных каналов связи лежат между этими двумя предельными случаями. При этом потери информации при передаче k символов по данному каналу связи

$$\Delta I = kH(A/B).$$

Неемотря на то, что часть информации поражается помехами, между принятыми и переданными сообщениями существует статистическая взаимосвязь. Это позволяет описывать информационные характеристики реальных каналов связи при помощи энтропии объединения статистически зависимых событий. Так как

$$H(A, B) = H(A) + H(B/A) = H(B) + H(A/B), \qquad (24)$$

то потери в канале связи могут быть учтены при помощи энтропии объединения следующим образом:

$$I(B, A) = H(A) + H(B) - H(B, A), \qquad (25)$$

а с использованием условной энтропии

$$I(B, A) = H(A) - H(A/B) = H(B) - H(B/A).$$

Для вычисления среднего количества информации, содержащегося в принятом ансамбле сообщений B относительно переданного ансамбля сообщений A в условиях действия помех, пользуются следующими выражениями, выведенными непосредственно из выражения (25):

$$I(B, A) = \sum_i \sum_j p(a_i)\, p(b_j/a_i) \log \frac{p(b_j/a_i)}{p(b_j)}, \qquad (26)$$

$$I(A, B) = \sum_i \sum_j p(b_j)\, p(a_i/b_j) \log \frac{p(a_i/b_j)}{p(a_i)}, \qquad (27)$$

$$I(B,\ A)=I(A,\ B)=\sum_i\sum_j p(a_i,\ b_j)\log\frac{p(b_j/a_i)}{p(b_j)}=$$

$$=\sum_i\sum_j p(b_j,\ a_i)\log\frac{p(a_i/b_j)}{p(a_i)}=\sum_i\sum_j p(a_i,\ b_j)\log\frac{p(a_i,\ b_j)}{p(a_i)\,p(b_j)}. \quad (28)$$

Для вычислений часто удобно применять выражения (26—28) в виде

$$I(A,\ B)=\sum_j p(b_j)\sum_i[p(a_i/b_j)\log p(a_i/b_j)-p(a_i/b_j)\log p(a_i)],$$

$$I(B,\ A)=\sum_i p(a_i)\sum_j[p(b_j/a_i)\log p(b_j/a_i)-p(b_j/a_i)\log p(b_j)],$$

$$I(A,\ B)=I(B,\ A)=\sum_i\sum_j p(a_i,\ b_j)\log p(a_i,\ b_j)-$$

$$-\sum_i\sum_j p(a_i,\ b_j)\log p(a_i)\,p(b_j).$$

Для полного и всестороннего описания канала связи необходимо задать: канальную матрицу вида $p(a_i/b_j)$ и безусловные вероятности вида $p(b_j)$ или канальную матрицу вида $p(b_j/a_i)$ и безусловные вероятности вида $p(a_i)$, или канальную матрицу вида $p(a_i,\ b_j)$. В последнем случае сумма значений матрицы по столбцам дает безусловные вероятности вида $p(b_j)\left(\sum_j p(b_j)=1\right)$, а сумма по строкам дает безусловные вероятности вида $p(a_i)\left(\sum_i p(a_i)=1\right)$. Условные вероятности могут быть найдены из выражений:

$$p(a_i/b_j)=\frac{p(b_j,\ a_i)}{p(b_j)};\quad p(b_j/a_i)=\frac{p(a_i,\ b_j)}{p(a_i)}.$$

Зная условные и безусловные вероятности, можно найти $H(A)$, $H(B)$, $H(A/B)$ и $H(B/A)$.

Если уровень помех настолько высок, что с равной вероятностью можно ожидать переход любого символа источника сообщения в произвольный символ первичного алфавита, то энтропия канала связи будет равна $\log_2 m$, а количество информации $I=H(A)-\log_2 m\leqslant 0$, при этом значение I может быть отрицательной величиной, что означает, что канал связи вносит дезинформацию.

Задача 3.1. Канал связи описан следующей канальной матрицей:

$$p(b/a)=\begin{vmatrix}0{,}98 & 0{,}01 & 0{,}01\\ 0{,}1 & 0{,}75 & 0{,}15\\ 0{,}2 & 0{,}3 & 0{,}5\end{vmatrix}.$$

Вычислить среднее количество информации, которое переносится одним символом сообщения, если вероятности появления символов источника сообщений равны $p(a_1)=0{,}7$; $p(a_2)=0{,}2$; $p(a_3)=0{,}1$.

Чему равны информационные потери при передаче сообщения из 400 символов алфавита a_1, a_2, a_3? Чему равно количество принятой информации?

Задача 3.2. Передан русский текст по телетайпу. Какие сведения необходимо иметь для того, чтобы определить количество принятой информации, если известно, что в канале связи часть информации поражается шумами?

2.3 Практическое занятие №5-6 (4 часа).

Тема: «Понятие энтропии и семантическая информация. Смысл энтропии Шеннона. Понятие семантической информации. Семантическая мера информации. Семантическая информация в системах.»

2.3.1 Задание для работы:

1. Емкость канала связи

2. Способы измерения информации

2.3.2 Краткое описание проводимого занятия:

Потери информации в каналах связи с шумами обычно описывают при помощи условной энтропии и энтропии объединения.

Если помех нет или их уровень настолько низок, что они не в состоянии уничтожить сигнал или имитировать полезный сигнал в отсутствие передачи, то при передаче a_i мы будем твердо уверены, что получим b_j — сигнал, соответствующий переданному a_i-му сигналу. События A и B статистически жестко связаны, условная вероятность максимальна $p(b_j/a_i) = 1$, а условная энтропия

$$H(A/B) = -\sum_{i=1}^{m} p(b_j/a_i) \log p(b_j/a_i) = 0, \qquad (23)$$

так как $\log p(b_j/a_i) = \log 1 = 0$. В этом случае количество информации, содержащееся в принятом ансамбле сообщений B, равно энтропии передаваемых сообщений ансамбля A, т. е. $I(B, A) = H(A)$[1].

При высоком уровне помех любой из принятых сигналов b_j может соответствовать любому переданному сигналу a_i, статистическая связь между переданными и принятыми сигналами отсутствует.

В этом случае вероятности $p(a_i)$ и $p(b_j)$ есть вероятности независимых событий и $p(b_j/a_i) = p(b_j)$; $p(a_i/b_j) = p(a_i)$.

$$H(A/B) = -\sum_i \sum_j p(b_j)\, p(a_i/b_j) \log p(a_i/b_j) =$$

$$= -\sum_i \sum_j p(b_j)\, p(a_i) \log p(a_i) = \sum_j p(b_j)\, H(A) = H(A),$$

так как $\sum_j p(b_j) = 1$, т. е. условная энтропия равна безусловной, а количество информации, содержащееся в B, относительно A равно нулю:

$$I(A, B) = H(A) - H(A/B) = 0.$$

Задача 3.1. Канал связи описан следующей канальной матрицей:

$$p(b/a) = \begin{vmatrix} 0{,}98 & 0{,}01 & 0{,}01 \\ 0{,}1 & 0{,}75 & 0{,}15 \\ 0{,}2 & 0{,}3 & 0{,}5 \end{vmatrix}.$$

Вычислить среднее количество информации, которое переносится одним символом сообщения, если вероятности появления символов источника сообщений равны $p(a_1) = 0{,}7$; $p(a_2) = 0{,}2$; $p(a_3) = 0{,}1$.

Чему равны информационные потери при передаче сообщения из 400 символов алфавита a_1, a_2, a_3? Чему равно количество принятой информации?

Решение. Энтропия источника сообщений

$$H(A) = -\sum_{i=1}^{m} p_i \log_2 p_i = -(0{,}7 \log_2 0{,}7 + 0{,}2 \log_2 0{,}2 +$$

$$+ 0{,}1 \log_2 0{,}1) = 0{,}3602 + 0{,}4644 + 0{,}3322 = 1{,}1568 \text{ } бит/символ.$$

Общая условная энтропия

$$H(B/A) = -\sum_i p(a_i) \sum_j p(b_j/a_i) \log_2 p(b_j/a_i) = -[0{,}7 (0{,}98 \log_2 0{,}98 +$$

$$+ 2 \cdot 0{,}01 \log_2 0{,}01) + 0{,}2 (0{,}75 \log_2 0{,}75 + 0{,}1 \log_2 0{,}1 +$$

$$+ 0{,}15 \log_2 0{,}15) + 0{,}1 (0{,}2 \log_2 0{,}2 + 0{,}3 \log_2 0{,}3 +$$

$$+ 0{,}5 \log_2 0{,}5)] = 0{,}7 (0{,}0285 + 2 \cdot 0{,}0664) + 0{,}2 (0{,}3113 + 0{,}322 +$$

$$+ 0{,}4105) + 0{,}1 (0{,}4644 + 0{,}5211 + 0{,}5) = 0{,}473 \text{ } бит/символ.$$

Потери в канале связи

$$\Delta I = kH(B/A) = 400 \cdot 0{,}473 = 189{,}5 \text{ } бит.$$

Энтропия приемника

$$H(B) = -\sum_{j=1}^{m} p(b_j) \log_2 p(b_j);$$

$$p(b_1) = \sum_i p(a_i)\, p(b_1/a_i) = p(a_1)\, p(b_1/a_1) + p(a_2)\, p(b_1/a_2) +$$

$$+ p(a_3)\, p(b_1/a_3) = 0{,}7 \cdot 0{,}98 + 0{,}2 \cdot 0{,}1 + 0{,}1 \cdot 0{,}2 = 0{,}726;$$

$$p(b_2) = p(a_1)\, p(b_2/a_1) + p(a_2)\, p(b_2/a_2) + p(a_3)\, p(b_2/a_3) =$$

$$= 0{,}7 \cdot 0{,}01 + 0{,}2 \cdot 0{,}75 + 0{,}1 \cdot 0{,}3 = 0{,}187;$$

$$p(b_3) = p(a_1)\, p(b_3/a_1) + p(a_2)\, p(b_3/a_2) + p(a_3)\, p(b_3/a_3) =$$

$$= 0{,}7 \cdot 0{,}01 + 0{,}2 \cdot 0{,}15 + 0{,}1 \cdot 0{,}5 = 0{,}087;$$

$$p(b_1) + p(b_2) + p(b_3) = 0{,}726 + 0{,}187 + 0{,}087 = 1,$$

т. е. $\sum_j p(b_j) = 1$.

$$H(B) = -(0{,}726 \log_2 0{,}726 + 0{,}187 \log_2 0{,}187 +$$

$$+ 0{,}087 \log_2 0{,}087) = 1{,}094 \text{ } бит/символ.$$

Среднее количество полученной информации

$$I = k[H(B) - H(B/A)] = kH(B) - \Delta I = 248{,}1 \text{ } бит.$$

Задача 3.2. Передан русский текст по телетайпу. Какие сведения необходимо иметь для того, чтобы определить количество принятой информации, если известно, что в канале связи часть информации поражается шумами?

Задача 3.3. Чему равны информационные потери в канале связи, описанном при помощи следующей канальной матрицы:

$$p(a/b) = \begin{vmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{vmatrix}?$$

Задача 3.4. Определить информационные потери в канале связи, описанном следующей матрицей:

$$p(a/b) = \begin{vmatrix} 0{,}99 & 0{,}01 & 0 \\ 0{,}01 & 0{,}98 & 0 \\ 0 & 0{,}01 & 1 \end{vmatrix},$$

если символы алфавита встречаются в сообщениях с равной вероятностью.

2.4 Практическое занятие №7-8 (4 часа).

Тема: «Необратимое сжатие. Обратимое сжатие. Кодирование Хаффмана. Упорядоченное дерево Хаффмана. Избыточность сообщений. Код Фано.
»

2.4.1 Задание для работы:

1. Смысл энтропии Шеннона

2. Понятие семантической сети

2.4.2 Краткое описание проводимого занятия:

Понятие условной энтропии в теории информации используется при определении взаимозависимости[1] между символами кодируемого алфавита, для определения потерь при передаче информации по каналам связи, при вычислении энтропии объединения.

Во всех случаях при вычислении условной энтропии в том или ином виде используются условные вероятности.

Если при передаче n сообщений символ A появился m раз, символ B появился l раз, а символ A вместе с символом B — k раз, то вероятность появления символа A $p(A) = \frac{m}{n}$; вероятность появления символа B $p(B) = \frac{l}{n}$; вероятность совместного появления символов A и B $p(AB) = \frac{k}{n}$; условная вероятность появления символа A относительно символа B и условная вероятность появления символа B относительно символа A

$$p(A/B) = \frac{p(AB)}{p(B)} = \frac{k}{l};\ p(B/A) = \frac{p(AB)}{p(A)} = \frac{k/n}{m/n} = \frac{k}{m}. \quad (7)$$

Если известна условная вероятность, то можно легко определить и вероятность совместного появления символов A и B, используя выражение (7)

$$p(AB) = p(B)\,p(A/B) = p(A)\,p(B/A). \quad (8)$$

От классического выражения (4) формула условной энтропии отличается тем, что в ней вероятности — условные:

$$H(b_j/a_i) = -\sum_i p(b_j/a_i) \log p(b_j/a_i); \quad (9)$$

$$H(a_i/b_j) = -\sum_i p(a_i/b_j) \log p(a_i/b_j), \qquad (10)$$

где индекс i выбран для характеристики произвольного состояния источника сообщений A, индекс j выбран для характеристики произвольного состояния адресата B.

Различают понятия частной и общей условной энтропии. Выражения (9) и (10) представляют собой частные условные энтропии.

Общая условная энтропия сообщения B относительно сообщения A характеризует количество информации, содержащееся в любом символе алфавита, и определяется усреднением по всем символам, т. е. по всем состояниям с учетом вероятности появления каждого из состояний, и равна сумме вероятностей появления символов алфавита на неопределенность, которая остается после того, как адресат принял сигнал

$$H(B/A) = -\sum_i p(a_i) H(b_j/a_i) = -\sum_i \sum_j p(a_i)\, p(b_j/a_i) \log p(b_j/a_i). \qquad (11)$$

Выражение (11) является общим выражением для определения количества информации на один символ сообщения для случая неравномерных и взаимозависимых символов.

Так как $p(a_i)\, p(b_j/a_i)$ представляет собой вероятность совместного появления двух событий $p(a_i, b_j)$, то формулу (11) можно записать следующим образом:

$$H(B/A) = -\sum_i \sum_j p(a_i, b_j) \log p(b_j/a_i). \qquad (12)$$

Понятие общей и частной условной энтропии широко используется при вычислении информационных потерь в каналах связи с шумами.

В общем случае, если мы передаем m сигналов A и ожидаем получить m сигналов B, влияние помех в канале связи полностью описывается *канальной матрицей*, которую мы приводим ниже:

A \ B	b_1	b_2	...	b_j	...	b_m
a_1	$p(b_1/a_1)$,	$p(b_2/a_1)$,	...,	$p(b_j/a_1)$,	...,	$p(b_m/a_1)$
a_2	$p(b_1/a_2)$,	$p(b_2/a_2)$,	...,	$p(b_j/a_2)$,	...,	$p(b_m/a_2)$
...	...	...	...	...	...	...
a_i	$p(b_1/a_i)$,	$p(b_2/a_i)$,	...,	$p(b_j/a_i)$,	...,	$p(b_m/a_i)$
...	...	...	...	...	...	...
a_m	$p(b_1/a_m)$,	$p(b_2/a_m)$,	...,	$p(b_j/a_m)$,	...,	$p(b_m/a_m)$

Вероятности, которые расположены по диагонали (выделенные полужирным шрифтом), определяют правильный прием, остальные — ложный. Значения цифр, заполняющих колонки канальной матрицы, обычно уменьшаются по мере удаления от главной диагонали и при полном отсутствии помех все, кроме цифр, расположенных на главной диагонали, равны нулю.

Если описывать канал связи со стороны источника сообщений, то прохождение данного вида сигнала в данном канале связи описывается распределением условных вероятностей вида $p(b_j / a_i)$, так для сигнала a_1 распределением вида

$$p(b_1 / a_1) + p(b_2 / a_1) + \ldots + p(b_j / a_1) + \ldots + p(b_m / a_1). \quad (13)$$

Сумма вероятностей распределения (13) всегда должна равняться 1. Потери информации, которые приходятся на долю сигнала a_1, описываются при помощи *частной условной энтропии* вида

$$H(b_j / a_1) = -\sum_{j=1} p(b_j / a_1) \log p(b_j / a_1). \quad (14)$$

Суммирование производится по j, так как i-е состояние (в данном случае первое) остается постоянным.

Чтобы учесть потери при передаче всех сигналов по данному каналу связи, следует просуммировать все частные условные энтропии, т. е. произвести двойное суммирование по i и по j. При этом в случае равновероятных появлений сигналов на выходе источника сообщений

$$H(B / A) = -\frac{1}{m} \sum_j p(b_j / a_i) \log p(b_j / a_i) \quad (15)$$

(на m делим, так как энтропия есть неопределенность на один символ).

В случае неравновероятного появления символов источника сообщений следует учесть вероятность появления каждого символа, умножив на нее соответствующую частную условную энтропию. При этом общая условная энтропия

$$H(B/A) = -\sum_i p(a_i) \sum_j p(b_j / a_i) \log p(b_j / a_i). \quad (16)$$

Если мы исследуем канал связи со стороны приемника сообщений, то с получением сигнала b_j предполагаем, что был послан какой-то из сигналов $a_1, a_2, \ldots, a_i, \ldots, a_m$. При этом канальная матрица будет иметь вид

A \ B	b_1	b_2	$\ldots$	b_j	$\ldots$	b_m
a_1	$p(a_1/b_1)$,	$p(a_1/b_2)$,	$\ldots$,	$p(a_1/b_j)$,	$\ldots$,	$p(a_1/b_m)$
a_2	$p(a_2/b_1)$,	$p(a_2/b_2)$,	$\ldots$,	$p(a_2/b_j)$,	$\ldots$,	$p(a_2/b_m)$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
a_i	$p(a_i/b_1)$,	$p(a_i/b_2)$,	$\ldots$,	$p(a_i/b_j)$,	$\ldots$,	$p(a_i/b_m)$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
a_m	$p(a_m/b_1)$,	$p(a_m/b_2)$,	$\ldots$,	$p(a_m/b_j)$,	$\ldots$,	$p(a_m/b_m)$

Задача 2.1. В результате статистических испытаний установлено, что при передаче каждых 100 сообщений длиной по 5 символов в сообщении символ K встречается 50 раз, а символ T — 30 раз. Вместе с символом K символ T встречается 10 раз. Определить условные энтропии $H(K/T)$ и $H(T/K)$.

Решение. Общее количество переданных символов

$$n = 100 \cdot 5 = 500.$$

Вероятность появления символа K

$$p(K) = \frac{50}{500} = 0{,}1.$$

Вероятность появления символа T

$$p(T) = \frac{30}{500} = 0{,}06.$$

Вероятность совместного появления символа K и T

$$p(KT) = \frac{10}{500} = 0{,}02.$$

Так как $p(KT) = p(T)\,p(K/T) = p(K)\,p(T/K)$, то условная вероятность появления символа K относительно символа T

$$p(K/T) = \frac{p(KT)}{p(T)} = \frac{0{,}02}{0{,}06} = 0{,}33.$$

Условная вероятность появления символа T относительно символа K

$$p(T/K) = \frac{p(KT)}{p(K)} = \frac{0{,}02}{0{,}1} = 0{,}2.$$

Условная энтропия символа K относительно T

$$H(K/T) = -\sum_i p(b_j/a_i)\log_2 p(b_j/a_i) = -\{p(K/T)\log_2 p(K/T) +$$

$$+ [1 - p(K/T)]\log_2[1 - p(K/T)]\} = -(0{,}33\log_2 0{,}33 +$$

$$+ 0{,}67\log_2 0{,}67) = 0{,}9149 \ \textit{бит/символ}.$$

Условная энтропия появления символа T относительно K

$$H(T/K) = -(0{,}2\log_2 0{,}2 + 0{,}8\log_2 0{,}8) = 0{,}7219 \ \textit{бит/символ}.$$

Задача 2.2. При передаче текстовых сообщений статистические наблюдения показали, что для слов со средней длиной в 6 букв на каждые 100 сообщений буква A встречается 80 раз, буква B встречается 50 раз, буквы A и B вместе встречаются 10 раз. Определить условную энтропию появления A, если в слове присутствует B, и условную энтропию B, если в слове присутствует A.

2.5 Практическое занятие №9-10 (4 часа).

Тема: «Словарно-ориентированные алгоритмы сжатия. Метод Лемпела-Зива LZ77. Метод LZSS. Метод LZ78. Метод LZW. LZ-алгоритмы распаковки данных.»

2.5.1 Задание для работы:

1. Семантическая мера информации

2. Семантическая информация

2.5.2 Краткое описание проводимого занятия:

Контрольные задачи

1. В результате статистических испытаний канала связи № 1 со стороны источника сообщений были получены следующие условные вероятности: $p(b_1/a_1) = 0{,}9$; $p(b_2/a_1) = 0{,}1$; $p(b_3/a_1) = 0$; $p(b_1/a_2) = 0{,}1$; $p(b_2/a_2) = 0{,}8$; $p(b_3/a_2) = 0{,}1$; $p(b_1/a_3) = 0$; $p(b_2/a_3) = 0{,}1$; $p(b_3/a_3) = 0{,}9$.

При испытании канала связи № 2 со стороны приемника сообщений получены условные вероятности $p(a_1/b_1) = 0{,}9$; $p(a_1/b_2) = 0{,}08$; $p(a_1/b_3) = 0$; $p(a_2/b_1) = 0{,}1$; $p(a_2/b_2) = 0{,}8$; $b(a_2/b_3) = 0{,}08$; $p(a_3/b_1) = 0$; $p(a_3/b_2) = 0{,}12$; $p(a_3/b_3) = 0{,}92$.

Построить соответствующие канальные матрицы и определить частные условные энтропии относительно сигнала a_2 (со стороны источника сообщений) и сигнала b_3 (со стороны приемника).

2. Определить все возможные информационные характеристики канала связи, в котором взаимосвязь источника с приемником может быть описана матрицей вида

$$p(A, B) = \begin{vmatrix} 0{,}2 & 0{,}1 & 0 \\ 0{,}2 & 0 & 0{,}2 \\ 0{,}1 & 0{,}1 & 0{,}1 \end{vmatrix}.$$

3. Вероятности появления сигналов на входе приемника сообщений равны соответственно: $p(b_1) = 0{,}2$; $p(b_2) = 0{,}3$; $p(b_3) = 0{,}5$. Канал связи описан следующей канальной матрицей:

$$p(a/b) = \begin{vmatrix} 0{,}97 & 0 & 0{,}01 \\ 0{,}02 & 0{,}98 & 0{,}01 \\ 0{,}01 & 0{,}02 & 0{,}98 \end{vmatrix}.$$

Определить энтропию источника сообщений.

4. Определить частную условную энтропию относительно каждого символа источника сообщений при передаче по каналу связи, описанному следующей канальной матрицей:

$$p(a, b) = \begin{vmatrix} 0{,}2 & 0 & 0 \\ 0{,}1 & 0{,}2 & 0 \\ 0 & 0{,}1 & 0{,}4 \end{vmatrix}.$$

5. В результате статистических испытаний канала связи были получены следующие условные вероятности перехода одного сигнала в другой: $p(b_1/a_1) = 0{,}85$; $p(b_2/a_1) = 0{,}1$; $p(b_3/a_1) = 0{,}05$; $p(b_1/a_2) = 0{,}09$; $p(b_2/a_2) = 0{,}91$; $p(b_3/a_2) = 0$; $p(b_1/a_3) = 0$; $p(b_2/a_3) = 0{,}08$; $p(b_3/a_3) = 0{,}92$. Построить канальную матрицу и определить общую условную энтропию сообщений, передаваемых по данному каналу связи.

6. Построить произвольные канальные матрицы, описывающие канал связи как со стороны источника сообщений, так и со стороны приемника. В чем разница таких матриц? Как определить частные условные энтропии по одной и по другой матрице?

7. Построить произвольную матрицу некоторой объединенной системы. Какие замечательные свойства такой матрицы?

8. Показать процесс перехода от матрицы с вероятностями вида $p(a, b)$ к матрице с вероятностями вида $p(b/a)$.

9. Определить полные условные энтропии двух систем A и B, если матрица вероятностей системы, полученной в результате объединения систем A и B, имеет вид

$$p(A, B) = \begin{vmatrix} 0{,}1 & 0{,}1 & 0 \\ 0 & 0{,}2 & 0{,}1 \\ 0 & 0{,}2 & 0{,}3 \end{vmatrix}.$$

2.6 Практическое занятие №11-12 (4 часа).

Тема: «Программы сжатия. Особенности программ-архиваторов. Сжатие информации с потерями.»

2.6.1 Задание для работы:

1. Необратимое сжатие

2. Обратимое сжатие

2.6.2 Краткое описание проводимого занятия:

Сжатие информации представляет собой операцию, в результате которой данному коду или сообщению ставится в соответствие более короткий код или сообщение[1].

Сжатие информации имеет целью — ускорение и удешевление процессов механизированной обработки, хранения и поиска информации, экономия памяти ЭВМ. При сжатии следует стремиться к минимальной неоднозначности сжатых кодов при максимальной простоте алгоритма сжатия. Рассмотрим наиболее характерные методы сжатия информации.

Сжатие информации делением кода на части, меньшие некоторой наперед заданной величины A, заключается в том, что исходный код делится на части, меньшие A, после чего полученные части кода складываются между собой либо по правилам двоичной арифметики, либо по модулю 2. Например, исходный код 101011010110; $A = 4$

$$\begin{array}{r} 1\,0\,1\,0 \\ +\,1\,1\,0\,1 \\ \hline 0\,1\,1\,0 \\ \hline 1\,1\,1\,0\,1 \end{array}$$ — сжатый код,

$$\begin{array}{r} 1\,0\,1\,0 \\ \oplus\,1\,1\,0\,1 \\ \hline 0\,1\,1\,0 \\ \hline 0\,0\,0\,1 \end{array}$$ — сжатый код.

Сжатие информации с побуквенным сдвигом в каждом разряде [5], как и предыдущий способ, не предусматривает восстановления сжимаемых кодов, а применяется лишь для сокращения адреса либо самого кода сжимаемого слова в памяти ЭВМ.

Предположим, исходное слово «газета» кодируется кодом, в котором длина кодовой комбинации буквы $l = 8$:
г — 01000111; а — 11110000; з — 01100011; е — 00010111; т — 11011000.

Полный код слова «Газета»

0100011111110000011000110001011111011000111100000.

Сжатие осуществляется сложением по модулю 2 двоичных кодов букв сжимаемого слова с побуквенным сдвигом в каждом разряде.

```
  1 1 1 1 0 0 0 0          а
    1 1 0 1 1 0 0 0        т
⊕     0 0 0 1 0 1 1 1      е
        0 1 1 0 0 0 1 1    з
          1 1 1 1 0 0 0 0  а
            0 1 0 0 0 1 1 1 г
-----------------------------
1 0 0 1 1 0 0 0 1 0 0 1 1 — сжатый код.
```

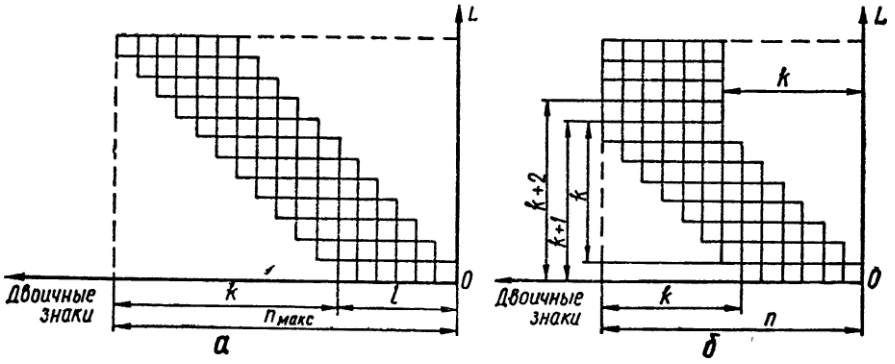


Рис. 8

Допустимое количество разрядов сжатого кода является вполне определенной величиной, зависящей от способа кодирования и от емкости ЗУ. Количество адресов, а соответственно максимальное количество слов в выделенном участке памяти машины определяется из следующего соотношения

$$2^{n_{макс}} \leqslant N, \qquad (88)$$

где $n_{макс}$ — максимально допустимая длина (количество двоичных разрядов) сжатого кода; N — возможное количество адресов в ЗУ.

Если представить процесс побуквенного сдвига в общем виде, как показано на рис. 8, *а*, то длина сжатого кода

$$n = k + l,$$

где k — число побуквенных сдвигов; l — длина кодовой комбинации буквы.

Так как сдвигаются все буквы, кроме первой, то и число сдвигов $k = L - 1$, где L — число букв в слове. Тогда

$$n = l + (L - 1).$$

В русском языке наиболее длинные слова имеют 23—25 букв. Если принять $L_{макс} = 23$, с условием осуществления побуквенного сдвига с каждым шагом ровно на один разряд, для n и l могут быть получены следующие соотношения

$$l = \begin{cases} 6 \text{ дв. разрядов} \\ 7 \text{ —»—} \\ 8 \text{ —»—} \\ 9 \text{ —»—} \end{cases} \qquad n_{макс} = \begin{cases} 28 \text{ дв. разрядов} \\ 29 \text{ —»—} \\ 30 \text{ —»—} \\ 31 \text{ —»—} \end{cases}$$

Если значение $n_{макс}$ не удовлетворяет неравенству (88), можно конечные буквы слова складывать по модулю 2 без сдвига относительно предыдущей буквы, как это показано на рис 8, *б*.

Например, если для предыдущего примера со словом «Газета» $n_{макс} = 11$, сжатый код будет иметь вид:

$$\begin{array}{r} 1\,1\,1\,1\,0\,0\,0\,0 \\ 1\,1\,0\,1\,1\,0\,0\,0 \\ 0\,0\,0\,1\,0\,1\,1\,1 \\ \oplus\quad 0\,1\,1\,0\,0\,0\,1\,1 \\ \quad\;\; 1\,1\,1\,1\,0\,0\,0\,0 \\ \quad\quad\;\; 0\,1\,0\,0\,0\,1\,1\,1 \\ \hline 0\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1 \end{array}$$

Метод сжатия информации на основе исключения повторения в старших разрядах последующих строк массивов одинаковых элементов старших разрядов предыдущих строк массивов основан на том, что в сжатых массивах повторяющиеся элементы старших разрядов заменяются некоторым условным символом.

Очень часто обрабатываемая информация бывает представлена в виде набора однородных массивов, в которых элементы столбцов или строк массивов расположены в нарастающем порядке. Если считать старшими разряды, расположенные левее данного элемента, а младшими — расположенные правее, то можно заметить, что во многих случаях строки матриц отличаются друг от друга в младших разрядах. Если при записи каждого последующего элемента массива отбрасывать все повторяющиеся в предыдущем элементы, например в строке стоящие подряд элементы старших разрядов, то массивы могут быть сокращены от 2 до 10 и более разрядов [2].

Для учета выброшенных разрядов вводится знак раздела ρ, который позволяет отделить элементы в свернутом массиве. В случае полного повторения строк записывается соответствующе

количество ρ. При развертывании вместо знака ρ восстанавливаются все пропущенные разряды, которые были до элемента, стоящего непосредственно за ρ в сжатом тексте.

Для примера рассмотрим следующий массив:

```
9 5 7 0 1 2 4
9 5 7 0 1 2 5
9 5 7 0 3 8 6
9 5 7 0 3 9 0
1 2 3 4 5 6 7
1 2 3 4 5 9 1
1 2 3 4 5 9 3
```

Свернутый массив будет иметь вид:

```
9 5 7 0 1 2 4
ρ 5 ρ 3 8 6 ρ
9 0 1 2 3 4 5
6 7 ρ 9 1 ρ 3
```

Расшифровка (развертывание) происходит с конца массива. Переход на следующую строку происходит по двум условиям: либо по заполнению строки, либо при встрече ρ.

```
9 5 7 0 1 2 4
. . . . . .5
. . . .3 8 6
. . . . .9 0
1 2 3 4 5 6 7
. . . . .9 1
. . . . . .3
```

Пропущенные цифры заполняются автоматически по аналогичным разрядам предыдущей строки. Заполнение производится с начала массива. Этот метод можно развить и для свертывания массивов, в которых повторяющиеся разряды встречаются не только с начала строки. Если в строке один повторяющийся участок, то кроме ρ добавляется еще один дополнительный символ *K*, означающий конец строки. Расшифровка ведется от *K* до *K*. Длина строки известна. Нужно, чтобы оставшиеся между *K* цифры вместе с пропущенными разрядами составляли полную строку. При этом нам все равно, в каком месте строки выбрасываются повторяющиеся

разряды, лишь бы в строке было не более одного участка с повторяющимися разрядами. Например:

Исходный массив	Свернутый массив
1 2 3 4 5 6 7	1 2 3 4 5 6 7
1 2 3 4 5 8 6	*K* ρ 8 6 *K* 2 1
2 1 3 4 5 2 4	ρ 2 4 *K* ρ 9 *K*
2 1 3 4 5 2 9	4 2 9 ρ *K* ρ *K*
4 2 9 4 5 2 9	ρ *K* 5 ρ 1 *K*
4 2 9 4 5 2 9	
4 2 9 4 5 2 9	
5 2 9 4 5 2 1	

Если в строке есть два повторяющихся участка, то, используя этот метод, выбрасываем бо́льший.

Процесс развертывания массива осуществляется следующим образом: переход на следующую строку происходит при встрече *K*

1 2 3 4 5 6 7
. . . . . 8 6
2 1 . . . 2 4
. 9
4 2 9 . . . .
. . . . . .
. . . . . .
5 1

Пропущенные цифры заполняются по аналогичным разрядам предыдущей строки начиная с конца массива.

Если в строке массива несколько повторяющихся участков, то можно вместо ρ вставлять специальные символы, указывающие на необходимое число пропусков.

Например, если обозначить количество пропусков, соответственно, $X-2$; $Y-3$; $Z-5$, то исходный и свернутый массивы будут иметь вид:

Исходный массив	Свернутый массив
1 9 7 1 1 3 7 4 3 0	1 9 7 1 1 3 7 4 3 0
1 9 7 1 1 3 7 4 3 1	*Z XX* 1 *ZXX* 2 *Y* 2
1 9 7 1 1 3 7 4 3 2	*Y X* 0 *ZXX* 1 *ZXX*
1 9 7 2 1 3 7 4 3 0	2
1 9 7 2 1 3 7 4 3 1	
1 9 7 2 1 3 7 4 3 2	

Задача 8.1. Сложить по правилам двоичной арифметики и по модулю два такие числа 01111011, 1110110011, 1101.

Задача 8.2. Исходный код 10001110101110111011. Чему равен сжатый код, если его длина не должна превышать величину $A = 9$? Сжатие осуществить двумя способами: по правилам двоичной арифметики и по модулю 2.

Задача 8.4. Какой вид имеет сжатый код слова «привет», если сжатие осуществляется методом побуквенного сдвига в каждом разряде, а буквы слова кодируются стандартным телеграфным кодом № 3 (см. приложение 4).

Задача 8.5. Определить максимальное количество разрядов сжатого кода, если сжатие осуществляется методом побуквенного сдвига в каждом разряде, а допустимое количество адресов ЗУ—1024.

2.7 Практическое занятие №13-14 (4 часа).

Тема: «Алгоритм арифметического кодирования. Адаптивное арифметическое кодирование.»

2.7.1 Задание для работы:

1. Кодирование Хаффмана

2. Упорядоченное дерево Хаффмана

2.7.2 Краткое описание проводимого занятия:

Построение кода Хаффмана сводится к построению соответствующего бинарного дерева по следующему алгоритму:

-Составим список кодируемых символов, при этом будем рассматривать один символ как дерево, состоящее из одного элемента с весом, равным частоте появления символа в строке.

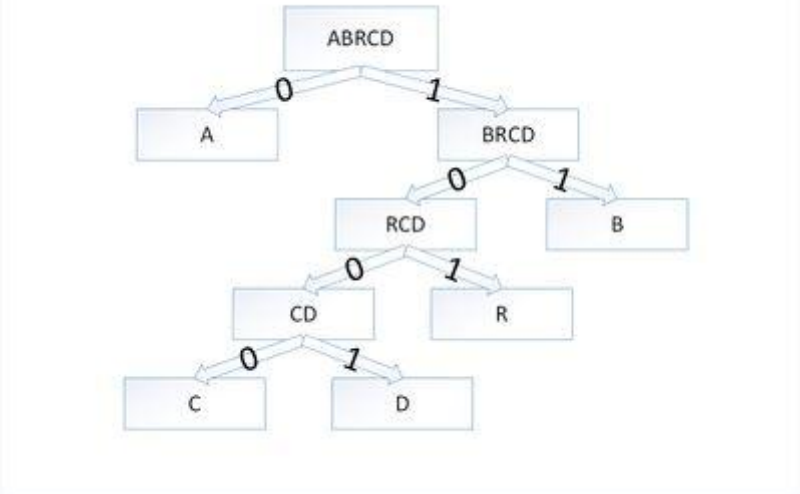
-Из списка выберем два узла с наименьшим весом.

-Сформируем новый узел с весом, равным сумме весов выбранных узлов, и присоединим к нему два выбранных узла в качестве детей.

-Добавим к списку только что сформированный узел вместо двух объединенных узлов.

-Если в списке больше одного узла, то повторим пункты со второго по пятый.

Пример



Дерево Хаффмана для слова $abracadabra$

Закодируем слово $abracadabra$. Тогда алфавит будет $A = \{a, b, r, c, d\}$, а набор весов (частота появления символов алфавита в кодируемом слове) $W = \{5, 2, 2, 1, 1\}$:

В дереве Хаффмана будет 5 узлов:

Узел	a	b	r	c	d
Вес	5	2	2	1	1

По алгоритму возьмем два символа с наименьшей частотой — это c и d. Сформируем из них новый узел cd весом 2 и добавим его к списку узлов:

Узел	a	b	r	cd
Вес	5	2	2	2

Затем опять объединим в один узел два минимальных по весу узла — r и cd:

Узел	a	rcd	b
Вес	5	4	2

Еще раз повторим эту же операцию, но для узлов rcd и b:

Узел	brcd	a
Вес	6	5

На последнем шаге объединим два узла — $brcd$ и a:

Узел	abrcd
Вес	11

Остался один узел, значит, мы пришли к корню дерева Хаффмана (смотри рисунок). Теперь для каждого символа выберем кодовое слово (бинарная последовательность, обозначающая путь по дереву к этому символу от корня):

Символ	a	b	r	c	d
Код	0	11	101	1000	1001

Таким образом, закодированное слово $abracadabra$ будет выглядеть как 01110101000010010111010. Длина закодированного слова — 23 бита. Стоит заметить, что если бы мы использовали алгоритм кодирования с одинаковой длиной всех кодовых слов, то закодированное слово заняло бы 33 бита, что существенно больше.

1. Проведите кодирование по методу Хаффмана трехбуквенных слов из предыдущей задачи.

2. Проведите кодирование 7 букв из задачи 302 по методу Хаффмана.

2.8 Практическое занятие №15-16 (4 часа).

Тема: «Помехозащитное кодирование. Матричное кодирование и групповые коды. Информационный канал. Помехозащитное кодирование в двоичном симметричном канале. Матричное кодирование. Групповые коды. Совершенные и квазисовершенные коды.»

2.8.1 Задание для работы:

1. Избыточность сообщений

2. Код Фано

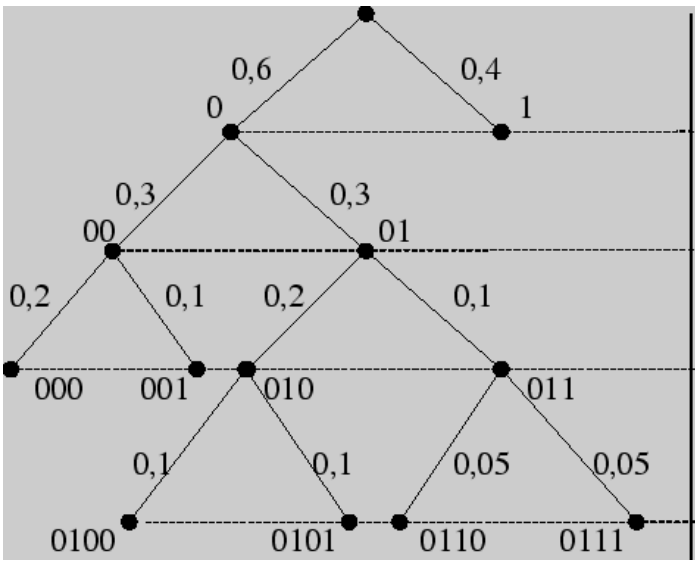
2.8.2 Краткое описание проводимого занятия:

При кодировании по Фано все сообщения записываются в таблицу по степени убывания вероятности и разбиваются на две группы примерно (насколько это возможно) равной вероятности. Соответственно этой процедуре из корня кодового дерева исходят два ребра, которым в качестве весов присваиваются полученные вероятности. Двум образовавшимся вершинам приписывают кодовые символы 0 и 1. Затем каждая из групп вероятностей вновь делится на две подгруппы примерно равной вероятности. В соответствии с этим из каждой вершины 0 и 1 исходят по два ребра с весами, равными вероятностям подгрупп, а вновь образованным вершинам приписывают символы 00 и 01, 10 и 11. В результате многократного повторения процедуры разделения вероятностей и образования вершин приходим к ситуации, когда в качестве веса, приписанного ребру бинарного дерева, выступает вероятность одного из данных сообщений. В этом случае вновь образованная вершина оказывается листом дерева, т.к. процесс деления вероятностей для нее завершен. Задача кодирования считается решенной, когда на всех ветвях кодового бинарного дерева образуются листья.

Пример 151. Закодировать по Фано сообщения, имеющие следующие вероятности:

сообщение	1	2	3	4	5	6	7
вероятность	0,4	0,2	0,1	0,1	0,1	0,05	0,05

Решение 1 (с использованием кодового дерева)

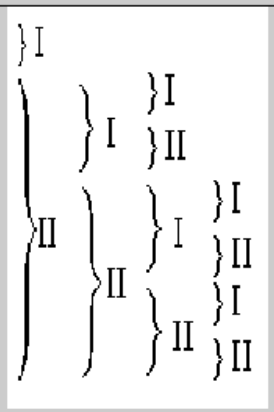


Коды	Сообщения
1	1
000	2
001	3
0100	4
0101	5
0110	6
0111	7

Рис. 91

Листья кодового дерева представляют собой кодируемые сообщения с присвоенными им кодовыми словами.

Решение 2 (табличный способ)

Сообщение	P	Разбиение на подгруппы	Кодовое обозначение
1	0,4		1
2	0,2		000
3	0,1		001
4	0,1		0100
5	0,1		0101
6	0,05		0110
7	0,05		0111

Цена кодирования (средняя длина кодового слова $l)$ является критерием степени оптимальности кодирования. Вычислим ее в нашем случае.

$$l = \sum_{i=1}^{7} l_i \cdot p_i = 1 \cdot 0,4 + 3 \cdot 0,2 + 3 \cdot 0,1 + 4 \cdot (0,1 \cdot 2 + 0,05 \cdot 2) = 2,5.$$

Для того, чтобы закодировать сообщения по Хаффману, предварительно преобразуется таблица, задающая вероятности сообщений. Исходные данные записываются в столбец, две последние (наименьшие) вероятности в котором

складываются, а полученная сумма становится новым элементом таблицы, занимающим соответствующее место в списке убывающих по величине вероятностей. Эта процедура продолжается до тех пор, пока в столбце не останутся всего два элемента.

1. Проведите кодирование по методу Фано алфавита из четырех букв, вероятности которых равны 0,4; 0,3; 0,2 и 0,1.

2. Алфавит содержит 7 букв, которые встречаются с вероятностями 0,4; 0,2; 0,1; 0,1; 0,1; 0,05; 0,05. Осуществите кодирование по методу Фано.

3. Алфавит состоит из двух букв, $А$ и $Б$, встречающихся с вероятностями $P(А)$ = 0,8 и $P(Б)$ = 0,2. Примените метод Фано к кодированию всевозможных двухбуквенных и трехбуквенных комбинаций.

2.9 Практическое занятие №17-18 (4 часа).

Тема: «Полиномиальные коды. Принцип построения полиномиальных кодов. Линейные коды. Циклические коды. Исправление ошибок при построении кодов.**»**

2.9.1 Задание для работы:

1. Алгоритмы арифметического кодирования

2.9.2 Краткое описание проводимого занятия:

Кодирование

Алгоритму передаются текст для кодирования и список частот встречаемости символов.

1. Рассмотрим отрезок $[0; 1)$ на координатной прямой.
2. Поставим каждому символу текста в соответствие отрезок, длина которого равна частоте его появления.
3. Считаем символ из входного потока и рассмотрим отрезок, соответствующий этому символу. Этот отрезок разделим на части, пропорциональные частотам встречаемости символов.
4. Повторим пункт (3) до конца входного потока.
5. Выберем любое число из получившегося отрезка. Это и будет результат арифметического кодирования.

```
left = 0
```

```
right = 1
while !eof
   read(symb)
   newRight = left + (right - left) * segment[symb].right  //segment[symb] — подотрезок отрезка [0; 1),
                                                           //соответствующий символу symb
   newLeft = left + (right - left) * segment[symb].left
   left = newLeft
   right = newRight
ans = (left + right) / 2
```

Декодирование

Алгоритм по вещественному числу восстанавливает исходный текст.

1. Выберем на отрезке $[0; 1)$, разделенном на части, длины которых равны вероятностям появления символов в тексте, подотрезок, содержащий входное вещественное число. Символ, соответствующий этому подотрезку, дописываем в ответ.
2. Нормируем подотрезок и вещественное число.
3. Повторим п. (1-2) до тех пор, пока не получим ответ (до конца файла).

```
do
for i = 1 to n
   if code >= segment[i].left && code < segment[i].right
      write(segment[i].character)
      code = (code – segment[i].left) / (segment[i].right – segment[i].left)
      break
while (segment[i].character != eof)
```

Замечание

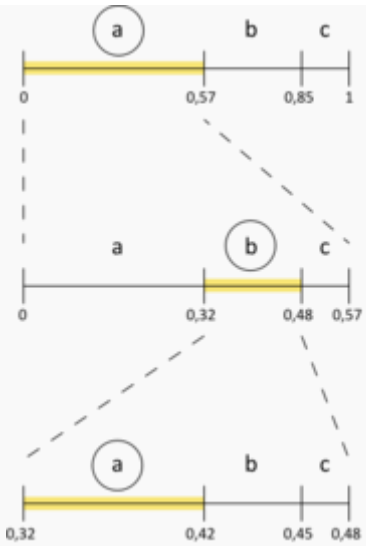
К кодировщику и декодировщику должно быть известно, когда завершать работу. Для этого можно передавать в качестве аргумента длину текста или символ конца файла, после которого процесс должен быть остановлен.

Для оптимизации размера кода необходимо выбрать из окончательного диапазона число, содержащее наименьшее количество знаков в двоичной записи.

Рассмотрим в качестве примера строку $abacaba$

Кодирование

Символ	Частота появления
a	0.571429
b	0.285714
c	0.142857



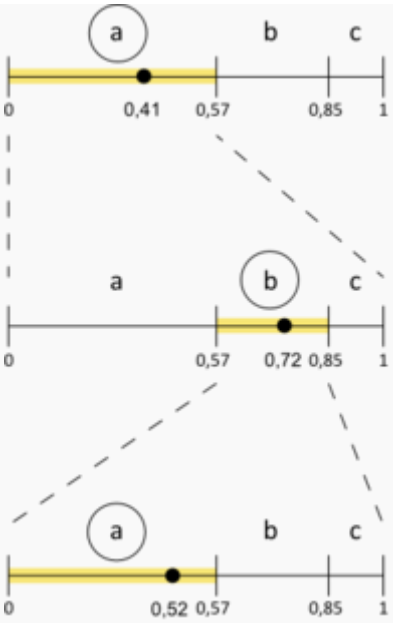
Пример работы кодировщика

Считанный символ	Левая граница отрезка	Правая граница отрезка
	0	1
a	0	0.571429
b	0.326531	0.489796
a	0.326531	0.419825
c	0.406497	0.419825
a	0.406497	0.414113
b	0.410849	0.413025
a	0.410849	0.412093

Код: 0.411471

Декодирование

Код: 0.411471



Пример работы декодировщика

Декодируемый символ	Код
a	0.411471
b	0.720074
a	0.520259
c	0.910454
a	0.373178
b	0.653061
a	0.285714

Замечание

При декодировании текста можно не только нормализовывать рабочий отрезок и текущий код, но и уменьшать рабочий отрезок (аналогично кодированию), не изменяя значение кода.

Задание:

1. Объясните алгоритм построения арифметического кода.

2. Постройте арифметический код сообщения S = «2+2!» для символов первичного алфавита A={«+», «2», «!»} с известными вероятностями появления символов P={0,4; 0,5; 0,1} (символ «!» не несет смысловой нагрузки и является признаком конца сообщения).

3. Декодируйте сообщение S, составленное из символов 53 0 0,3 0,46 0,9 1 0,3 0,9 0,46 0,48 0,84 алфавита A={«1», «2», «3», «!»} с вероятностями P={0,4; 0,3; 0,2; 0,1}, если известен его арифметический код, записываемый в двоичной системе счисления как 0,10001012

2.10 Практическое занятие №19-21 (6 часа).

Тема: «Криптография. Криптосистема без передачи ключей. Криптосистема с открытым ключом. Электронная подпись. Стандарт шифрования данных. Кодировка текстовой информации.»

2.10.1 Задание для работы:

1. Алгоритмы арифметического кодирования

2.10.2 Краткое описание проводимого занятия:

Кодирование

Алгоритму передаются текст для кодирования и список частот встречаемости символов.

6. Рассмотрим отрезок $[0; 1)$ на координатной прямой.
7. Поставим каждому символу текста в соответствие отрезок, длина которого равна частоте его появления.
8. Считаем символ из входного потока и рассмотрим отрезок, соответствующий этому символу. Этот отрезок разделим на части, пропорциональные частотам встречаемости символов.
9. Повторим пункт (3) до конца входного потока.
10. Выберем любое число из получившегося отрезка. Это и будет результат арифметического кодирования.

```
left = 0
right = 1
while !eof
   read(symb)
   newRight = left + (right - left) * segment[symb].right  //segment[symb] — подотрезок отрезка [0; 1),
                                                           //соответствующий символу symb
   newLeft = left + (right - left) * segment[symb].left
   left = newLeft
   right = newRight
ans = (left + right) / 2
```

Декодирование

Алгоритм по вещественному числу восстанавливает исходный текст.

4. Выберем на отрезке $[0; 1)$, разделенном на части, длины которых равны вероятностям появления символов в тексте, подотрезок, содержащий входное вещественное число. Символ, соответствующий этому подотрезку, дописываем в ответ.
5. Нормируем подотрезок и вещественное число.
6. Повторим п. (1-2) до тех пор, пока не получим ответ (до конца файла).

```
do
for i = 1 to n
   if code >= segment[i].left && code < segment[i].right
      write(segment[i].character)
      code = (code – segment[i].left) / (segment[i].right – segment[i].left)
      break
while (segment[i].character != eof)
```

Замечание

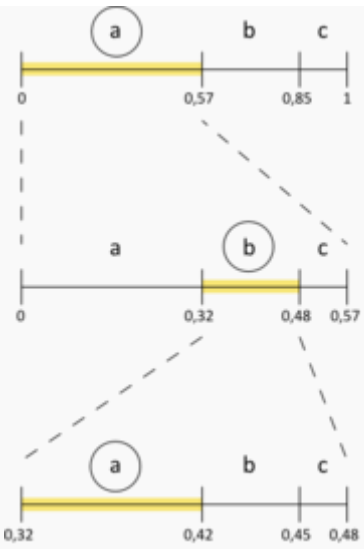
К кодировщику и декодировщику должно быть известно, когда завершать работу. Для этого можно передавать в качестве аргумента длину текста или символ конца файла, после которого процесс должен быть остановлен.

Для оптимизации размера кода необходимо выбрать из окончательного диапазона число, содержащее наименьшее количество знаков в двоичной записи.

Рассмотрим в качестве примера строку $abacaba$

Кодирование

Символ	Частота появления
a	0.571429
b	0.285714
c	0.142857



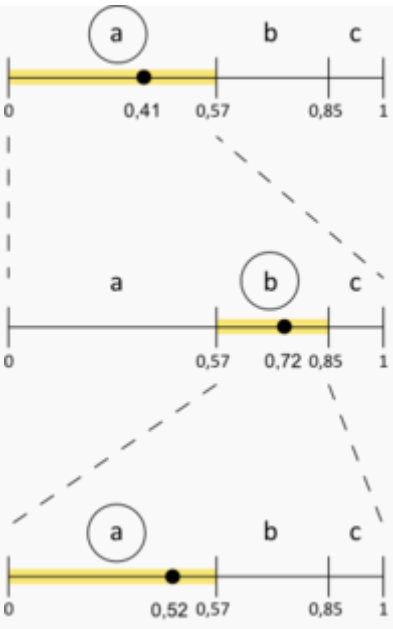
Пример работы кодировщика

Считанный символ	Левая граница отрезка	Правая граница отрезка
	0	1
a	0	0.571429
b	0.326531	0.489796
a	0.326531	0.419825
c	0.406497	0.419825
a	0.406497	0.414113
b	0.410849	0.413025
a	0.410849	0.412093

Код: 0.411471

Декодирование

Код: 0.411471



Пример работы декодировщика

Декодируемый символ	Код
a	0.411471
b	0.720074
a	0.520259
c	0.910454
a	0.373178
b	0.653061
a	0.285714

Задание:

1. Объясните алгоритм получения кода целого положительного числа.

59 2 36 37 1 2 18 18 0 2 8 9 1 2 4 4 0 2 2 2 0 1

2. Получите шестнадцатиразрядный дополнительный код числа 6510.

3. Определите число по его дополнительному коду: 0001110011100001