Методическая разработка

для проведения лекции

Занятие 15. Кодирование источника сообщений

Учебные вопросы занятия:
1. Сжатие сообщений
2. Коды Шеннона-Фано, Хаффмана, арифметическое кодирование
Заключительная часть

Введение

На передающей стороне телекоммуникационной системы, а также в локальных вычислительных сетях работу кодера источника в обобщенном виде можно представить как отображение некоторых элементов сообщения $\langle s_i \rangle_\mu$, $i = 1(1)N$ длиной μ источника $S_{\{N\}}$ в определенные кодовые комбинации используемого кода $C_{\{N\}}$. На приемной стороне системы осуществляется наблюдение некоторой двоичной последовательности, которая представляет собой совокупность кодовых комбинаций анализируемого кода источника. В свою очередь, декодер кода источника реализует обратное отображение, обеспечивающее взаимно однозначное соответствие между совокупностями двоичных символов и элементами сообщения.

1 Сжатие сообщений

Необходимо отметить, что сжатие сообщений является сложной процедурой и реализуется, как правило, в виде последовательности следующих операций (рисунок 1):

– преобразование модели исходного сообщения, отражающей информационные свойства источника;

– непосредственно кодирование элементов модели.

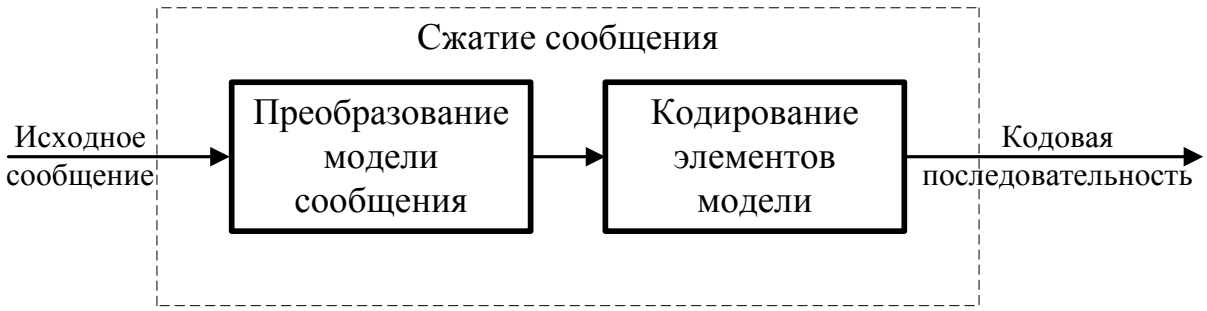

Рисунок 1 – Схематичное представление процедуры сжатия сообщения

Множество различных методов преобразования модели исходного сообщения и их кодирования порождает большое число процедур сжатия, что обуславливает необходимость получения формализованного описания существующих алгоритмов. Создание таких описаний основывается на классификации процедур сжатия по различным признакам.

Вариант классификации процедур такого рода представлен на рисунке 2.

Рисунок 2 – Вариант классификации процедур сжатия

Для оценки качества рассматриваемого процесса используется коэффициент сжатия $K_{сж}$, равный отношению объема $V_{исх}$ исходного сообщения, выраженного в двоичных символах, к объему $V_{сж}$ преобразованного сообщения [5]:

$$K_{сж} = \frac{V_{исх}}{V_{сж}}. \quad (1)$$

1.1 Преобразование модели сообщения, алгоритмы группы *LZ*

Наиболее простая идея при преобразовании модели сообщения положена в основу алгоритма кодирования длин серий (*RLE* кодирование), обеспечивающего кодирование повторяющихся символов сообщения. Любая

последовательность символов, содержащая не менее трех повторяющихся знаков, заменяется одиночной копией символа, за которой следует маркер и счетчик повторений. Если в потоке данных встречается символ со значением маркера, то после него в выходной поток данных вставляется счетчик повторений с нулевым значением длины.

В качестве маркера выбирают символ, имеющий наименьшую вероятность встречаемости (в системах электронного документооборота наиболее часто используется символ с десятичным значением 144). Счетчик, как правило, представляется однобайтовым числом. Такой подход применим к достаточно ограниченному кругу источников информации.

Исторически алгоритмы *RLE* появились первыми. В настоящее время они продолжают использоваться при пересылке электронной почты, в отечественных схемах кодирования черно-белых изображений, при кодировании растровых цветных изображений в формате *PCX*. Относительно невысокие значения достигаемого коэффициента сжатия привели к вытеснению алгоритмов *RLE* другими методами преобразования модели источника сообщений. В современных системах связи алгоритмы указанного типа не применяются.

В большинстве современных процедур сжатия (*RAR*, *ZIP*, *ARJ*), применяемых в отношении сообщений текстовых и смешанных видов, операция преобразования модели исходного сообщения основывается на группе алгоритмов Лемпеля-Зива (*LZ*77, *LZ*78, *LZSS*, *LZW*, *LZSUM* и др.). При этом формирование правил преобразования модели исходного сообщения осуществляется путем накопления статистики источника сообщений в словаре с учетом априорных или эвристических данных о его параметрах. Разработанный метод кодирования данных по словарю предполагает преобразование блоков данных переменного размера в кодовые слова постоянной длины, когда статистика и вид фраз заранее неизвестны.

В основе всех модификаций алгоритмов группы *LZ* лежит один из двух принципов описания источника сообщений.

1. В алгоритме *LZ*77 кодируемые строки заменяются ссылками на строки, расположенные в скользящем окне фиксированной длины (буфере кодирования), хранящем предыдущий текст сообщения. Ссылка содержит информацию о расположении строки в окне и ее длине.

2. В алгоритме *LZ*78 кодируемые строки заменяются ссылками на список уже закодированных строк, хранящихся в словаре кодера. Ссылка содержит информацию о номере строки в словаре кодера.

Базовый вариант алгоритма *LZ*77 включает:

– правило грамматического разбора строк символов из конечного алфавита источника $S_{\{N\}}$ на подстроки, длины которых не превышают предопределенного целого числа (максимальная длина совпадения, которая равна длине буфера кодирования);

– схему кодирования, которая последовательно отображает эти подстроки в единственным образом дешифруемые кодовые слова фиксированной длины над тем же самым алфавитом.

В рамках алгоритма *LZ*77 производится замена очередного фрагмента сообщения указателем в содержимое словаря. В качестве модели сообщения в представленном алгоритме используется «скользящее» по сообщению окно, разделенное на две неравные части. Первая, большая по размеру, включает уже просмотренную часть сообщения – словарь длиной L_c. Вторая, меньшая часть является буфером длиной L_b, содержащим еще не закодированные символы входного потока. В алгоритме выполняется грамматический разбор содержимого буфера кодирования, в ходе которого в словаре отыскивается подстрока наибольшей длины, совпадающая с содержимым буфера. Каждой подстроке присваивается уникальное кодовое слово, состоящее из трех элементов: кода смещения подстроки в «скользящем окне», кода ее длины и кода следующего в буфере символа. Последние два элемента кодового слова определяют значение указателя на местоположение данной подстроки в словаре.

Значения длин L_c и L_b выбираются с учетом требований по задержке кодирования и декодирования. При этом размерность кодовых комбинаций определяется следующим образом. Длина кодируемой строки не может быть больше размера буфера кодирования, а смещение не может превышать величину, равную уменьшенному на единицу размеру словаря. С учетом указанного обстоятельства длина двоичного кода смещения l_c определяется округленным в большую сторону значением $\log_2(L_c)$, а длина двоичного кода для совпадающей подстроки l_b – округленным в большую сторону значением $\log_2(L_b)$.

Пример 1. Пусть существует строка символов «красная_краска».

Необходимо преобразовать данную строку с использованием алгоритма *LZ*77.

Этапы преобразования и формируемые коды для словаря L_c длиной 8 и буфера L_b длиной 5 символов представлены в таблице 1. Длина исходной кодовой последовательности при использовании кодировки ASCII (8 бит для кодирования каждого символа алфавита) равна 14·8=112 бит.

Длина полученной кодовой последовательности соответственно равна $9\cdot(l_c+l_b+8)$=126 бит.

Таблица 1

Словарь	Буфер кодирования	Кодовая комбинация
_____	красн	<0,0,’к’>
_____к	расна	<0,0,’р’>
_____кр	асная	<0,0,’а’>
_____кра	сная_	<0,0,’с’>

____крас	ная_к	<0,0,’н’>
___красн	ая_кр	<5,1,’я’>
_красная	_крас	<0,0,’_’>
красная_	краск	<0,4,’к’>
ая_краск	а____	<0,0,’а’>(ошибка)

Существует множество модификаций основного варианта, отличающихся размером буфера кодирования (ограниченное или неограниченное «скользящее окно»), максимальной и минимальной длиной совпадения, способом кодирования символов и указателей. Они обладают различными коэффициентами сжатия и сложностью.

Одним из таких алгоритмов является алгоритм *LZ*SS, получивший широкое распространение в современных программных и аппаратных средствах сжатия. Кодовая последовательность, формируемая в процессе преобразования сообщения, начинается с префикса длиной 1 бит, значение которого отличает кодовую последовательность от незакодированного символа. Кодовая последовательность состоит из смещения относительно начала словаря и длины совпадения. Скользящее окно сдвигается либо на величину, равную длине совпадающей подстроки, либо на 1, если совпадений в словаре не обнаружено. Длина двоичных кодов смещения l_c и совпадающей подстроки l_b определяется по аналогии с предыдущим алгоритмом.

Пример 2. Пусть существует строка символов «красная_краска».

Необходимо преобразовать данную строку с использованием алгоритма *LZSS*.

Этапы преобразования и формируемые коды для словаря L_c длиной 8 и буфера L_b длиной 5 символов представлены в таблице 2.

Таблица 2.1.2

Словарь	Буфер кодирования	Кодовая комбинация	Длина кода
_____	красн	0,’к’	9
_____к	расна	0,’р’	9
_____кр	асная	0,’а’	9
_____кра	сная_	0,’с’	9
____крас	ная_к	0,’н’	9
___красн	ая_кр	1<5,1>	7
__красна	я_кра	0,’я’	9
_красная	_крас	0,’_’	9
красная_	краск	1<0,4>	7
ная_крас	ка___	1<4,1>	7
ая_краск	а____	1<0,1>	7

Длина полученной кодовой последовательности равна 7·9+4·7=91 бит.

Представленным алгоритмам свойственны следующие недостатки:

– ограниченные размеры словаря не позволяют закодировать строку, расположенную на удалении от совпадающей подстроки, превышающем размеры словаря;

– длина кодируемой строки не может превышать размеры буфера.

Словарный алгоритм *LZ*78 не обладает указанными недостатками, поскольку отличается от *LZ*77 как правилом грамматического разбора, так и способом кодирования сформированных в результате разбора фрагментов. В отличие от *LZ*77 для функционирования алгоритма *LZ*78 словарем является потенциально бесконечный список выделенных ранее подстрок, а не скользящее окно.

До начала работы словари алгоритмов прямого и обратного преобразования содержат только пустые строки. После считывания очередного символа сообщения он добавляется к текущей подстроке. Процесс продолжается до тех пор, пока сохраняется соответствие текущей кодируемой подстроки какой-либо подстроке в словаре. При появлении символа, нарушающего совпадение подстрок, кодер *LZ*78 выдает код, состоящий из двух частей. Первая часть – код индекса последнего совпадения (указателя на подстроку в словаре), вторая – код первого нарушившего совпадение подстрок символа. Новая подстрока, образованная слиянием найденной в словаре подстроки и этим символом, добавляется в словарь и ей присваивается новое кодовое слово. При последующем появлении данной подстроки в сообщении она будет использована для построения очередной, более длинной, подстроки, что повышает коэффициент сжатия. Если словарь уже заполнен, то из него предварительно удаляют наименее часто используемую фразу. Длина l_k двоичного кода является постоянной величиной и определяется округленным в большую сторону и увеличенным на 8 значением $\log_2(L_c)$, где L_c – размер словаря.

Пример 3. Пусть существует строка символов «красная_краска».

Необходимо преобразовать данную строку с использованием алгоритма *LZ*78.

Этапы преобразования и формируемые коды для словаря L_c длиной 16 фраз представлены в таблице 3.

Каждое кодовое слово имеет длину $l_k > \log_2(16) + 8$, а длина полученной кодовой последовательности соответственно равна 10·(4+8)=120 бит.

Алгоритм *LZ*78 также имеет множество модификаций, отличающихся размером и способом обновления словаря, способом кодирования символов и указателей. Самой известной и распространенной разновидностью этого алгоритма является алгоритм Лемпела-Зива-Велча (*LZW*). Основные отличия алгоритма *LZW* от *LZ*78 заключаются в следующем.

Таблица 3

Входная фраза (в словарь)	Кодовая комбинация	Позиция словаря
«_»		0
«к»	<0,’к’>	1
«р»	<0,’р’>	2
«а»	<0,’а’>	3
«с»	<0,’с’>	4
«н»	<0,’н’>	5
«ая»	<3,’я’>	6
«_»	<0,’_’>	7
«кр»	<1,’р’>	8
«ас»	<3,’с’>	9
«ка»	<1,’а’>	10

Первоначально словарь содержит все возможные подстроки длиной в один символ. Кодовое слово состоит только из кода индекса подстроки в словаре. Последний символ каждой новой подстроки одновременно является первым символом следующей подстроки. Как и в предыдущем случае, длина l_k двоичного кода для алгоритма *LZW* является фиксированной величиной и равна округленному в большую сторону значению $\log_2(L_c)$.

Отличительными чертами такого подхода также являются простота реализации и большая скорость преобразования модели по сравнению с любыми другими методами. В связи с этим данный алгоритм часто реализуется аппаратно.

Пример 4. Пусть существует строка символов «красная_краска».

Необходимо преобразовать данную строку с использованием алгоритма *LZW*.

Этапы преобразования и формируемые коды для словаря L_c длиной 500 фраз представлены в таблице 4.

Таблица 4

Входная фраза (wK)	Кодовая комбинация для w	Позиция словаря
ASCII		0-255
«кр»	0,’к’	256
«ра»	0,’р’	257
«ас»	0,’а’	258
«сн»	0,’с’	259
«на»	0,’н’	260
«ая»	0,’а’	261
«я_»	0,’я’	262
«_к»	0,’_’	263

«кра»	<256>,’а’	264
«аск»	<258>,’к’	265
«ка»	0,’к’	266
«а»	0,’а’	

Каждое кодовое слово имеет длину $l_k > \log_2(500)$, а длина полученной кодовой последовательности соответственно равна 12·9=108 бит.

Широкое применение алгоритмов группы *LZ* связано с простотой реализации на современной электронной базе и высоким коэффициентом сжатия исходного представления данных.

В среднем значение коэффициента сжатия для алгоритма *LZ*77 больше, чем для *LZ*78, но первый алгоритм преобразования сообщения реализуется за большее число шагов. На практике скорость работы алгоритма *LZ*77 может быть увеличена за счет ограничения размера окна, а также применения сложных алгоритмов поиска совпадающих фраз.

Процедуры сжатия, основу которых составляет алгоритм *LZ*77 и его модификации (*Zip, Rar, Arj*), асимметричны. Прямая процедура (сжатие) достаточно сложна, и в ней выполняется большое число операций (основные ресурсы тратятся на поиск повторов кодируемой строки в скользящем окне просмотра) при обработке сообщения, но обратная процедура (восстановление) проста и может работать с большой скоростью.

Процедуры, в которых реализуется алгоритм *LZ*78 и его модификации (*Stufflt, Compress, Zoo*), более адаптированы к требованиям по оперативности обработки, предъявляемым пользователями систем связи. По этой причине указанный алгоритм часто используется при передаче данных на канальном уровне (*ITU–T V*.42 *bis, RFC* 1144).

1.2 Кодирование элементов модели сообщения, способы оценивания параметров кодирования

По виду кодирования элементов, модели сообщения все применяемые коды подразделяются на равномерные и неравномерные. Равномерные коды в большинстве случаев являются избыточными в силу того, что при их построении не учитываются вероятности появления символов в сообщении и в соответствие каждому из них ставится одинаковое число двоичных символов.

Для сокращения избыточности при кодировании элементов модели сообщения предназначены неравномерные (статистические) коды.

С их помощью осуществляется выравнивание энергетических характеристик символов алфавита сообщения. Более вероятным символам ставятся в соответствие менее длинные комбинации кода источника. Рассматривая процесс во временной области, это можно интерпретировать таким образом, что для передачи более вероятных символов источника требуется меньший промежуток времени, чем для передачи символов, имеющих меньшую вероятность появления в сообщении. Такой способ

кодирования предполагает оценивание вероятностных характеристик $\langle p(s_i) \rangle$, $i = 1(1)N$ символов модели сообщения. Для оценивания и учета указанных характеристик при формировании рассматриваемых кодов находят применение статический, полудинамический и динамический способы (типы словаря кодера).

Статический способ оценивания предназначен для описания моделей сообщений с априорно известной структурой данных, которая постоянна во времени. Значения вероятностных характеристик в служебной части (словаре) сообщения оцениваются на этапе разработки процедуры сжатия и не зависят от параметров обрабатываемого сообщения. Словарь вероятностных характеристик символов формируется путем предварительного исследования свойств сообщений источника либо целого класса источников и остается постоянным для всех допустимых сообщений, причем его содержание заранее известно на передающей и приемной сторонах системы передачи информации.

В процедурах с *полудинамическим (полуадаптивным)* оцениванием статическая модель создается на основе предварительного исследования вероятностных характеристик $\langle p(s_i) \rangle$, $i = 1(1)N$ конкретного сообщения непосредственно перед кодированием. Параметры модели не меняются в процессе кодирования, но априорно неизвестны на этапе обратного преобразования. В силу этого обстоятельства описание параметров модели сообщения хранится вместе с преобразованным сообщением. Представленный тип процедур используется в большинстве современных и перспективных систем и средств сжатия.

В основу *динамического* способа оценивания (способ с динамическим словарем) положена адаптивная процедура переоценки вероятностных характеристик модели сообщения одновременно с кодированием символов. Применение процедур, реализующих представленный способ, не требует передачи описания модели сообщения.

2. Коды Шеннона-Фано, Хаффмана, арифметическое кодирование

Наиболее широкое распространение на практике для кодирования символов сообщений находят префиксные и арифметические коды. Префиксным называется код, любая комбинация которого не является началом любой другой комбинации этого же кода.

При описании параметров неравномерных префиксных кодов (объем алфавита N, распределение вероятностей $\langle p(s_i) \rangle$, $i = 1(1)N$) наибольшее распространение получили кодовые деревья (рисунок 3).

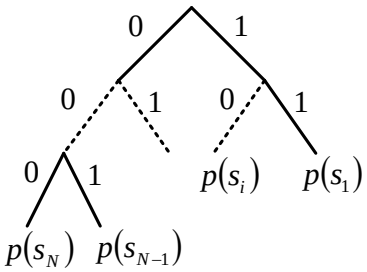

Рисунок 3 – Описание неравномерного префиксного кода с помощью кодового дерева

Формально *кодовое дерево* представляет собой конечное множество *T*, состоящее из одного или более узлов и обладающее следующими свойствами:

– имеется один специально обозначенный узел, называемый корнем данного дерева;

– остальные узлы (исключая корень) содержатся в $q \geq 0$ попарно непересекающихся множествах $T_1,...,T_q$, каждое из которых, в свою очередь, является деревом, при этом деревья $T_1,...,T_q$ называются поддеревьями данного корня.

Число поддеревьев данного узла называется *степенью* этого узла.

Узел с нулевой степенью называется *листом*.

Представленное кодовое дерево служит для описания префиксного кода источника, удовлетворяющего неравенству Крафта

$$\sum_{i=1}^{N} 2^{-l(s_i)} \leq 1, \qquad (2)$$

где $l(s_i)$ – длина элемента s_i в двоичных символах.

На первом этапе разработки процедур сжатия применение нашел *метод Шеннона-Фано* (Zip 1.0). На его основе разработан алгоритм, позволяющий представить каждый символ модели сообщения с помощью $\lceil -\log_b p(s_i) \rceil$ знаков префиксного кода, где *b* – основание вторичного кода.

Алгоритм построения кода отличается простотой, но приемлемого с точки зрения быстродействия алгоритма модификации для кода Шеннона-Фано не предложено, поэтому на практике его используют лишь при определенном способе учета статистических характеристик сообщения.

Код Шеннона-Фано строится следующим образом.

Знаки исходного алфавита и значения вероятности их появления заносятся в список, упорядоченный по убыванию значений вероятностей. Затем этот список разделяется на такие две группы, суммы вероятностей в которых были бы приблизительно равны. Каждый знак алфавита первой группы получает первым символом его кодового слова «0», второй, соответственно, – «1» . Затем каждая из этих групп разбивается по этому же принципу, и так до тех пор, пока в подгруппах не останется по одному знаку.

На практике часто не известны значения вероятностей появления знаков, поэтому их заменяют частотой встречаемости знака в сообщении.

Пример построения кода Шеннона-Фано представлен на рисунке 4.

	1 шаг		2 шаг		3 шаг		итоговая комбинация
	p(i)	символ	p(i)	символ	p(i)	символ	
p(a)=0,5	0,5	1					1
p(b)=0,2	0,5	0	0,2	1			01
p(c)=0,15			0,3	0	0,15	1	001
p(d)=0,15					0,15	0	000

Рисунок 4 – Пример построения кода Шеннона – Фано

Альтернативный алгоритм префиксного кодирования, позволяющий минимизировать длину исходного сообщения, был предложен Хаффманом.

В классическом алгоритме Хаффмана (*Zip* 2.0, *Rar*, *Arj*) в качестве исходных данных используется таблица частот встречаемости элементов в сообщении, на основании которой строится дерево кодирования Хаффмана. Код Хаффмана для каждого знака формируется методом обхода двоичного дерева, полученного следующим образом.

1. Элементы входного алфавита образуют список свободных узлов, причем каждый узел имеет вес, который может быть равен либо вероятности встречаемости символа, либо частоте его появления в сжимаемом сообщении.

2. Два узла с наименьшими весами заменяются новым узлом с весом, равным сумме весов объединяемых узлов.

3. Связь удаленных узлов с новым обозначается для одного знака «0», а для второго – «1».

4. Шаги, начиная со второго, повторяются до тех пор, пока в списке свободных узлов не останется только один узел, который будет считаться корнем дерева.

Использование представленного алгоритма позволяет сформировать конечное множество вариантов кода Хаффмана.

Пример получения данного кода представлен на рисунке 5.

В качестве основных достоинств рассмотренных кодов следует выделить высокую скорость преобразования потока с их помощью, а также их способность фактически устранять избыточность сообщений на

синтаксическом уровне в том случае, если частоты появления символов в сообщении близки к величинам $(1/2)^p$, где p – целое число. Однако чем сильнее отличаются вероятности символов от данных значений, тем больше избыточности остается в сообщении.

	1 шаг		2 шаг		3 шаг		итоговая комбинация
	p(i)	символ	p(i)	символ	p(i)	символ	
p(a)=0,5	0,5		0,5		0,5	1	1
p(b)=0,2	0,2		0,2	0	0,5	0	00
p(c)=0,15	0,15	1	0,3	1			011
p(d)=0,15	0,15	0					010

Рисунок 5 – Пример построения кода Хаффмана

Данным недостатком не обладают арифметические коды (АК). Для их описания обычно используется интервальное представление распределения вероятностей (рисунок 6).

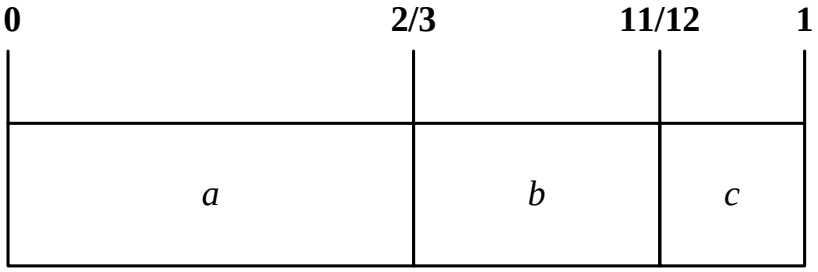

Рисунок 6 – Интервальное представление распределения вероятностей

Исходными данными для базового метода арифметического кодирования являются: список знаков алфавита $\Im_k$ и вероятностей их появления $p(x_j)$ в сообщении. Разбиение интервала описания алфавита $[0;1)$ на подинтервалы $[L_j^{\text{инт}}, H_j^{\text{инт}})$ для каждого знака алфавита осуществляется пропорционально вероятностям появления знака

$$\begin{cases} L_j^{\text{инт}} = H_{j-1}^{\text{инт}} \\ H_j^{\text{инт}} = L_j^{\text{инт}} + p(x_j), \end{cases} \quad j = 1(1)k,\ H_0^{\text{инт}} = 0. \qquad (3)$$

Сообщение описывается интервалом, имеющим начальное состояние $[0;1)$. Каждый новый знак сообщения уменьшает интервал, соответствующий кодируемому сообщению.

Для представления сообщения может быть использовано любое число, принадлежащее описанию интервала кодирования после обработки последнего знака сообщения. Число двоичных символов, необходимое для кодирования сообщения определяется выражением

$$K = -\log_2(R_{\mu}^{\text{инт}}). \qquad (4)$$

В силу того, что $R_{\mu}^{\text{инт}} = \prod_{i=1}^{\mu} p_i$, выражение (2.1.4) можно представить в виде

$$K = -\log_2(R_{\mu}^{\text{инт}}) = -\sum_{i=1}^{\mu} \log_2 p_i = -\sum_{j=1}^{N} p(x_j) \cdot \log_2 p(x_j)\ . \qquad (5)$$

Таким образом, число двоичных символов, получаемое с помощью алгоритма арифметического кодирования, равно энтропии сообщения. Это доказывает, что АК обеспечивает потенциально лучшие значения коэффициента сжатия в сравнении с префиксными кодами. Для большинства реальных источников сообщений выигрыш не превышает единиц процентов.

Пример 5. Пусть существует строка символов вида «авасваааааава», выбранная из сообщения, в котором используются три символа с вероятностями встречаемости: «а» – 2/3, «в» – 1/4, «с» – 1/12.

Необходимо преобразовать данную строку с помощью арифметического кодирования.

Шаги алгоритма при кодировании данного сообщения представлены в таблице 5. Графическая интерпретация процесса уменьшения интервала для первых трех шагов алгоритма АК изображена на рисунке 7.

Таблица 5

Знаки	Вероятность появления знаков в сообщении	Интервал, отводимый знаку при кодировании сообщения на 1 шаге	Вероятность появления знаков в сообщении, пересчитанная в интервал 2 шага	Интервал, отводимый знаку при кодировании сообщения на 2 шаге	Вероятность появления знаков в сообщении, пересчитанная в интервал 3 шага	Интервал, отводимый знаку при кодировании сообщения на 3 шаге

A	2/3	[0,2/3)	4/9	[0,4/9)	2/27	[4/9,14/27)
B	1/4	[2/3, 11/12)	1/6	[4/9,11/18)	1/36	[14/27, 65/108)
C	1/12	[11/12,1)	1/18	[11/18,2/3)	1/108	[65/108, 11/18)
Начальный интервал	Размер интервала на 1 шаге	Интервал на 1 шаге	Размер интервала на 2 шаге	Интервал на 2 шаге	Размер интервала на 3 шаге	Интервал на 3 шаге
[0,1)	2/3	[0,2/3)	1/6	[4/9,11/18)	1/9	[4/9, 14/27)

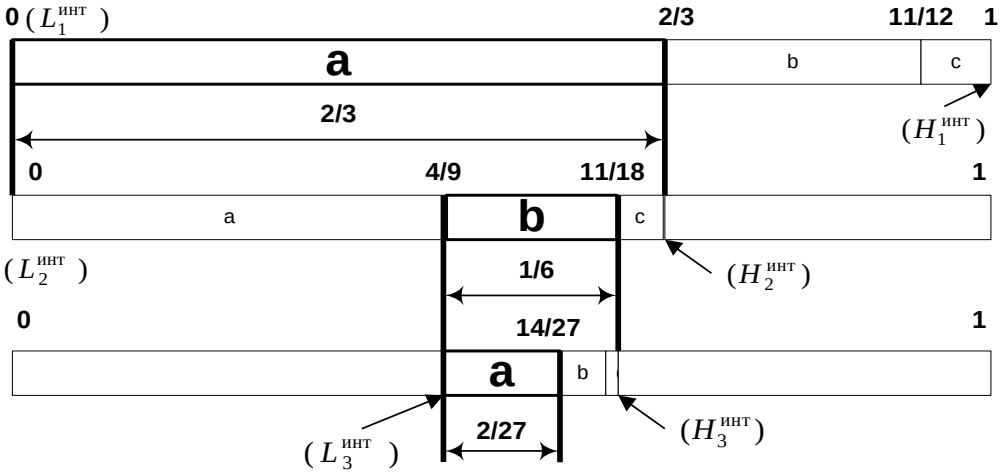

Рисунок 7 – Геометрическое представление алгоритма арифметического кодирования

После кодирования трех первых символов в рассматриваемом примере «ава» интервал, соответствующий этому сообщению, уменьшился до величины [4/9,14/27). Для описания сообщения «ава» можно использовать любое число, принадлежащее итоговому интервалу с точностью представления не хуже 2/27. В двоичном представлении сообщению «ава» соответствует число «1000», причем точность представления равна 1/16.

Заключительная часть.

Подвожу итоги занятия, анализирую степень достижения цели.

Рекомендованная литература:

1. Кричевский Р.Е. Сжатие и поиск информации. – М.: Радио и связь, 1989. – 168 с.