Информация.

Информация - нематериальная сущность, при помощи которой с любой точностью можно описывать реальные (материальные), виртуальные (возможные) и понятийные сущности. Информация - противоположность неопределенности.

Канал связи - это среда передачи информации, которая характеризуется в первую очередь максимально возможной для нее скоростью передачи данных (емкостью канала связи).

Шум - это помехи в канале связи при передаче информации.

Кодирование - преобразование дискретной информации одним из следующих способов: шифрование, сжатие, защита от шума.

Сжатие информации. Оптимальное сжатие.

Сжатие информации – важнейший аспект передачи данных, что дает возможность более оперативно передавать данные. Цель сжатия - уменьшение количества бит, необходимых для хранения или передачи заданной информации, что дает возможность передавать сообщения более быстро и хранить более экономно и оперативно (последнее означает, что операция извлечения данной информации с устройства ее хранения будет проходить быстрее, что возможно, если скорость распаковки данных выше скорости считывания данных с носителя информации).

Сжатие данных не может быть большим некоторого теоретические предела. Для формального определения этого предела рассматриваем любое информационное сообщение длины *n* как последовательность независимых, одинаково распределенных д.с.в. X_i или как выборки длины *n* значений одной д.с.в. *X*.

Среднее количество бит, приходящихся на одно кодируемое значение д.с.в., не может быть меньшим, чем энтропия этой д.с.в., т.е. $ML(X) \geq HX$ для любой д.с.в. и любого ее кода.

Кроме того, существует такое кодирование (Шеннона-Фэно, Fano), что $HX \geq ML(X) - 1$.

Рассмотрим д.с.в. X_1 и X_2, независимые и одинаково распределенные. $HX_1 = HX_2$ и $I(X_1, X_1) = 0$, следовательно, $H(X_1, X_2) = HX_1 + HX_2 - I(X_1, X_2) = 2HX_1.$

Вместо X_1 и X_2 можно говорить о двумерной д.с.в. $\vec{X} = (X_1, X_2)$. Аналогичным образом для *n*-мерной д.с.в. $\vec{X} = (X_1, X_2, \ldots, X_n)$ можно получить, что $H\vec{X} = nHX_1$.

Пусть $L_1(\vec{X}) = L(\vec{X})/n$, где $\vec{X} = (X_1, X_2, \ldots, X_n)$, т.е. $L_1(\vec{X})$ - это количество бит кода на единицу сообщения $\vec{X}$. Тогда $ML_1(\vec{X})$ - это среднее количество бит кода на единицу сообщения при передаче бесконечного множества сообщений $\vec{X}$. Из $ML(\vec{X}) - 1 \leqslant H\vec{X} \leqslant ML(\vec{X})$ для кода Шеннона-Фэно для $\vec{X}$ следует $ML_1(\vec{X}) - 1/n \leqslant HX_1 \leqslant ML_1(\vec{X})$ для этого же кода.

Таким образом, доказана основная теорема о кодировании при отсутствии помех, а именно то, что с ростом длины сообщения, при кодировании методом Шеннона-Фэно всего сообщения целиком среднее количество бит на единицу сообщения будет сколь угодно мало отличаться от энтропии единицы сообщения. Подобное кодирование практически не реализуемо из-за того, что с ростом длины сообщения трудоемкость построения этого кода становится недопустимо большой. Кроме того, такое кодирование делает невозможным отправку сообщения по частям, что необходимо для непрерывных процессов передачи данных. Дополнительным недостатком этого способа кодирования является необходимость отправки или хранения собственно полученного кода вместе с его исходной длиной, что снижает эффект от сжатия.

Алгоритмы Хаффмана.

Метод Шеннона-Фэно состоит в следующем, значения д.с.в. располагают в порядке убывания их вероятностей, а затем последовательно делят на две части с приблизительно равными вероятностями, к коду первой части добавляют 0, а к коду второй - 1.

Для предшествующего примера получим:

$\vec{X}$	p	$code(\vec{X})$
00	9/16	0
01	3/16	10
10	3/16	110
11	1/16	111,

$$ML_1(\vec{X}) = 27/32 = 0.84375 \text{ бит/сим.}$$

Еще один пример. Код составляется после сортировки, т.е. после перестановки значений B и C.

X	p	$code(X)$
A	0.4	0
B	0.2	11
C	0.4	10,

$$ML(X) = ML_1(X) = 1.6 \text{ бит/сим},$$
$$HX = \log_2 5 - 0.8 \approx 1.523 \text{ бит/сим.}$$

Метод Хаффмена (**Huffman**) разработан в 1952 г. Он более практичен и никогда по степени сжатия не уступает методу Шеннона-Фэно, более того, он сжимает максимально плотно. Код строится при помощи двоичного (бинарного) дерева. Вероятности значений д.с.в. приписываются его листьям; все дерево строится, опираясь на листья. Величина, приписанная к узлу дерева, называется весом узла. Два листа с наименьшими весами создают родительский узел с весом, равным сумме их весов; в дальнейшем этот узел учитывается наравне с оставшимися листьями, а образовавшие его узлы от такого рассмотрения устраняются. После постройки корня нужно приписать каждой из ветвей, исходящих из родительских узлов, значения 0 или 1. Код каждого значения д.с.в. - это число, получаемое при обходе ветвей от корня к листу, соответствующему данному значению.

Для методов Хаффмена и Шеннона-Фэно каждый раз вместе с собственно сообщением нужно передавать и таблицу кодов. Например, для случая из примера 2 нужно сообщить, что коду 10 соответствует символ C, коду 0 - A и т.д.

Построим коды Хаффмена для значений д.с.в. из двух предыдущих примеров.

$\vec{X}$	00	01	10	11
p	$^{9}/_{16}$	$^{3}/_{16}$	$^{3}/_{16}$	$^{1}/_{16}$
			0 ↘	↙ 1
				$\frac{4}{16}$
		0 ↘	↙ 1	
		$\frac{7}{16}$		
	0 ↘	↙ 1		
	1			
$code(\vec{X})$	0	10	110	111

$ML_1(\vec{X}) = ML(\vec{X})/2 = 27/32 = 0.84375$ бит/сим.

X	A	B	C
p	0.4	0.2	0.4
		0 ↘	↙ 1
			0.6
	0 ↘	↙ 1	
		1	
$code(X)$	0	10	11

$ML_1(X) = ML(X) = 1.6$ бит/сим.

LZ, LZW.

Методы Шеннона-Фэно, Хаффмена и арифметическое кодирование обобщающе называются статистическими методами. Словарные алгоритмы носят более практичный характер. Их частое преимущество перед статистическими теоретически объясняется тем, что они позволяют кодировать последовательности символов разной длины. Неадаптивные статистические алгоритмы тоже можно использовать для таких последовательностей, но в этом случае их реализация становится весьма ресурсоемкой.

Алгоритм LZ77 был опубликован в 1977 г. Разработан израильскими математиками Якобом Зивом (Ziv) и Авраамом Лемпелом (Lempel). Многие программы сжатия информации используют ту или иную модификацию LZ77. Одной из причин популярности алгоритмов LZ является их исключительная простота при высокой эффективности сжатия.

Основная идея LZ77 состоит в том, что второе и последующие вхождения некоторой строки символов в сообщении заменяются ссылками на ее первое вхождение.

LZ77 использует уже просмотренную часть сообщения как словарь. Чтобы добиться сжатия, он пытается заменить очередной фрагмент сообщения на указатель в содержимое словаря.

LZ77 использует "скользящее" по сообщению окно, разделенное на две неравные части. Первая, большая по размеру, включает уже просмотренную часть сообщения. Вторая, намного меньшая, является буфером, содержащим еще незакодированные символы входного потока. Обычно размер окна составляет несколько килобайт, а размер буфера - не более ста байт. Алгоритм пытается найти в словаре (большей части окна) фрагмент, совпадающий с содержимым буфера.

Алгоритм LZ77 выдает коды, состоящие из трех элементов:

- смещение в словаре относительно его начала подстроки, совпадающей с началом содержимого буфера;
- длина этой подстроки;
- первый символ буфера, следующий за подстрокой.

Декодирование кодов LZ77 проще их получения, т.к. не нужно осуществлять поиск в словаре.

Недостатки LZ77:

- с ростом размеров словаря скорость работы алгоритма-кодера пропорционально замедляется;
- кодирование одиночных символов очень неэффективно;
- невозможность кодирования подстрок, отстоящих друг от друга на расстоянии, большем длины словаря;
- длина подстроки, которую можно закодировать, ограничена размером буфера.

Кодирование одиночных символов можно сделать эффективным, отказавшись от ненужной ссылки на словарь для них. Кроме того, в некоторые модификации LZ77 для повышения степени сжатия добавляется возможность для кодирования идущих подряд одинаковых символов.

Пример. «Красная краска».

СЛОВАРЬ(8)	БУФЕР(5)	КОД
"........"	"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'>

В 1978 г. авторами LZ77 был разработан **алгоритм LZ78**, лишенный названных недостатков.

LZ78 не использует "скользящее" окно, он хранит словарь из уже просмотренных фраз. При старте алгоритма этот словарь содержит только одну пустую строку (строку длины нуль). 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.

Ключевым для размера получаемых кодов является размер словаря во фразах, потому что каждый код при кодировании по методу LZ78 содержит номер фразы в словаре. Из последнего следует, что эти коды имеют постоянную длину, равную округленному в большую сторону двоичному логарифму размера словаря (это количество бит в байт-коде расширенного ASCII).

Пример. «Красная краска». Словарь длиной 16 фраз.

ВХОДНАЯ ФРАЗА (В СЛОВАРЬ)	КОД	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

В 1984 г. Уэлчем (Welch) был путем модификации LZ78 создан алгоритм LZW.

Пошаговое описание алгоритма-кодера.

Шаг 1. Инициализация словаря всеми возможными односимвольными фразами (обычно 256 символами расширенного ASCII). Инициализация входной фразы ***w*** первым символом сообщения.

Шаг 2. Считать очередной символ ***K*** из кодируемого сообщения.

Шаг 3. Если ***КОНЕЦ_СООБЩЕНИЯ***, Выдать код для ***w***, Конец.

Если фраза ***wK*** уже есть в словаре, Присвоить входной фразе значение ***wK***, Перейти к Шагу 2.

Иначе, Выдать код ***w***, Добавить ***wK*** в словарь, Присвоить входной фразе значение ***K***, Перейти к Шагу 2.

Пример. «Красная краска». Размер словаря - 500 фраз.

ВХОДНАЯ ФРАЗА, wK (В СЛОВАРЬ)	КОД 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>	264
"АСК"	<258>	265
"КА"	0'К'	266
"А"	0'А'	

Арифметическое сжатие.

Хотелось бы иметь такую схему кодирования, которая позволяла бы кодировать некоторые символы менее чем одним битом. Одной из лучших среди таких схем является арифметическое кодирование, разработанное в 70-х годах XX века.

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

Принципиальное отличие этого кодирования от рассмотренных ранее методов в его непрерывности, т.е. в ненужности блокирования. Код здесь строится не для отдельных значений д.с.в. или их групп фиксированного размера, а для всего предшествующего сообщения в целом. Эффективность арифметического кодирования растет с ростом длины сжимаемого сообщения (для кодирования Хаффмена или Шеннона-Фэно этого не происходит). Хотя арифметическое кодирование дает обычно лучшее сжатие, чем кодирование Хаффмена, оно пока используется на практике сравнительно редко, т.к. оно появилось гораздо позже и требует больших вычислительных ресурсов.

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

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

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

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