Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Ярославский государственный университет им. П. Г. Демидова
Кафедра компьютерных сетей

Методы сжатия

Методические указания

Рекомендовано
Научно-методическим советом университета
для студентов, обучающихся по специальности
Прикладная математика и информатика

Ярославль 2009

ŋqdqyeY bɆ dedɆbYзe d caabcy Ɉqddyqyè

УДК 519.2
ББК В 182я73
М 54

Рекомендовано
Редакционно-издательским советом университета
в качестве учебного издания. План 2009 года

Рецензент
кафедра компьютерных сетей
Ярославского государственного университета им. П. Г. Демидова

Составитель М. В. Краснов

Методы сжатия: метод. указания / сост. М. В. Краснов; Яросл. гос. ун-т им. П. Г. Демидова. – Ярославль : ЯрГУ, 2009. – 44 с.

М 54

Основное использование вычислительной техники связано с хранением и передачей информации, вследствие чего возникает задача об экономном методе ее записи. Методические указания содержат описание некоторых математических понятий и приемов, используемых при решении этой задачи.

Предназначены для студентов, обучающихся по специальности 010501 Прикладная математика и информатика (дисциплина «Теория информации и кодирования», блок СД), очной формы обучения.

УДК 519.2
ББК В 182я73

ЭБ ЯрГУ им. П.Г. Демидова

Введение

В настоящее время электронная вычислительная техника применяется во многих сферах человеческой деятельности. Основное использование вычислительной техники связано с хранением и организацией доступа к информации. При этом возникает задача о сжатии данных. Цель сжатия данных – обеспечить компактное представление данных, вырабатываемых источником, для их более экономного хранения и передачи по каналам связи.

Все способы сжатия можно разделить на две категории: обратимое и необратимое сжатие.

Обратимое сжатие, или сжатие без потерь, приводит к снижению объема выходного потока информации без изменения его информативности, т. е. без потери информационной структуры. Из выходного потока, полученного после выполнения алгоритма обратимого сжатия, при помощи декодирования получается поток, в точности совпадающий с исходным.

Под необратимым сжатием, или сжатием с потерями, подразумевают такое преобразование входного потока данных, при котором выходной поток представляет из себя объект, с некоторой точки зрения достаточно похожий по внешним характеристикам на входной поток, однако отличается от него объемом.

Таким образом, сжатие с потерями состоит из двух этапов:

а) выделение сохраняемой части информации с помощью модели, зависящей от цели сжатия;

б) сжатие без потерь.

Такие подходы и алгоритмы используются для сжатия, например данных растровых графических файлов, где на выходе нужно представить графическую картинку, очень похожую на оригинал, но не обязательно являющуюся его точной копией.

Заметим, что сжатию могут подвергаться не только сами исходные данные, но и какие-либо преобразования над ними.

Методы сжатия без потерь

Рассмотрим методы сжатия без потерь на основе статистических алгоритмов (кодирование по Хаффману и арифметическое кодирование), а также на основе алгоритма RLE.

В основе этих методов сжатия лежит идея: «Если представлять часто используемые элементы короткими кодами, а редко используемые – длинными кодами, то для хранения блока данных требуется меньший объем памяти, чем если бы все элементы представлять кодами одинаковой длины».

Однако хочется заметить, что ни один компрессор не может сжать любой файл. После обработки любым компрессором размер части файлов уменьшится, а оставшейся части – увеличится или останется неизменным.

Утверждение. Элемент s_i, вероятность появления которого равняется $p(s_i)$, выгоднее всего представлять $-\log_2 p(s_i)$ битами.

Если распределение вероятностей F неизменно и вероятности появления элементов независимы, то среднюю длину кодов в битах можно найти как

$$H = -\sum p(s_i)\log_2 p(s_i),$$

это значение также называют энтропией источника в заданный момент времени.

Обычно вероятность появления элемента является условной. Тогда при кодировании очередного элемента s_i распределение вероятностей F принимает одно из возможных значений F_k и соответственно $H = H_k$. Среднюю длину кодов в битах можно вычислить по формуле

$$H = -\sum_k P_k H_k = -\sum_{k,i} P_k p_k(s_i)\log p_k(s_i),$$

где P_k – вероятность того, что F принимает k-е значение, которому соответствует набор вероятностей $p_k(s_i)$ генерации всех возможных элементов s_i.

Статистические алгоритмы нуждаются в знании вероятностей появления символов, оценкой, которой может являться частота появления символов во входных данных.

С учетом этого статистические алгоритмы можно разделить на три класса:

• Неадаптивные. Они используют фиксированные, заблаговременно заданные вероятности символов. Таблица вероятностей символов не передается вместе с файлом, поскольку она известна заблаговременно.

• Полуадаптивные. Для каждого файла строится таблица частот символов, и на её основе сжимают файл. Вместе со сжатым файлом передается таблица символов. Кодирование происходит за два этапа (на первом происходит подсчет частот, а на втором – кодирование).

• Адаптивные. Они начинают работать с фиксированной начальной таблицей частот символов (обычно все символы сначала равновероятны), и в процессе работы эта таблица изменяется в зависимости от символов, которые встречаются в файле. Кодирование происходит за один проход.

Алгоритм Хаффмана

Исходными данными алгоритма являются:

• алфавит $A = \{a_1, \ldots, a_n\}$, состоящий из n символов;

• $p(a_i)$ – вероятность появления каждого символа алфавита в рассматриваемом сообщении.

Алгоритм Хаффмана состоит из нескольких шагов.

Шаг 1. Выстраиваем все символы текущего алфавита в порядке убывания вероятностей.

Шаг 2. Строим новый алфавит A_j, который получается из предыдущего заменой двух символов $a_{i_{r-1}}$ a_{i_r} (с наименьшими вероятностями) на псевдосимвол a', причем $p(a') = p(a_{i_{r-1}}) + p(a_{i_r})$.

Продолжая эту процедуру (шаг 1 – шаг 2), будем приходить ко все более коротким алфавитам; после $n-2$ – кратного сжатия придем к алфавиту A_{n-2}, содержавшему уже всего две буквы.

Шаг 3. Символам последнего алфавита поставим в соответствие кодовые обозначения 0 и 1.