Методы сжатия цифровой информации

Файлы больших размеров, особенно звуковые, графические и видео файлы, очень редко хранятся в компьютере или передаются в неупакованном виде. Для уменьшения их размеров используют методы сжатие информации (универсальные или специальные).

Универсальные методы могут использоваться для сжатия любых данных и позволяют полностью восстанавливают исходную информацию. Поэтому универсальные методы еще называют обратимыми. Однако универсальные методы не используют знания о характере обрабатываемой информации и поэтому сжимают файлы достаточно слабо.

Специальные методы сжатия учитывают специфику человеческого восприятия звука или изображений, удаляя при сжатии маловажную информации, за счет чего удается добиться очень большой степени сжатия при некоторой потере качества. Однако данные методы не позволяют полностью восстановить исходную информацию.

Универсальные методы сжатия

Существует достаточно много универсальных (обратимых) методов сжатия, однако в их основе лежит сравнительно небольшое количество теоретических алгоритмов, которые мы рассмотрим на примерах.

Метод упаковки

Напомним, что при кодировании информации каждому объекту (символ, пикселю, измерению звука) ставиться в соответствие двоичный код фиксированной длины i. Тогда объем всей цифровой информации можно определить по формуле $V=i \cdot k$, где k – это количество объектов (символов, пикселей и т.д) в потоке информации.

Идея *метода упаковки* заключается в уменьшении количества бит, отводимых для кодирования каждого объекта, при условии, что в сжимаемом массиве данных присутствует не весь возможный набор объектов, а только его небольшая часть.

Пример 1. Представлен текст в кодировки ASCII, который содержит не все 256 символов, а только 12: цифры от «0» до «9», знак (минус) и пробел. Использование ASCII кодировки будет ставить в соответствие каждому символу код объемом 8 бит. Тогда текст «280 -1296 48 40 365 -159 13 777» в памяти компьютера займет 30сим • 8бит= 240 бит= 30 байт.

Однако для кодирования такого количества символов достаточно всего 4-х бит. Если упаковать коды данных символов в 4 бита (например, так: «0» - 0000, «1» - 0001, ... «9» - 1001, минус - 1110, пробел - 1111), то получим двукратное сжатие данных (15 байт).

Формат записи чисел, при котором число записывается в десятичной системе, а цифры числа кодируются 4-битовыми кодами, называется BCD-форматом (Binary Code Decimal, или двоично-десятичная запись). BCD-формат нередко используется в программировании для хранения целых чисел, например в базах данных.

Пример 2. Сообщение «КОЛ ОКОЛО КОЛОКОЛА» записанное в кодировке ASCII будет весить V_{ascii}=8бит • 18 символов = 144 бита, а кодировке Unicode соответственно $V_{unicode}$ = 16 бит • 18 символов = 288 бит.

Однако данное сообщение содержит всего 5 различных символов, следовательно, каждый символ может быть закодирован тремя битами, например, так: «А» - 000, «К» - 001, «Л» - 010, «О» - 011 и пробел - 111. Тогда объем сообщения будет равен V=18 символов • 3 бит = 54 бита.

В результате мы получаем коэффициент сжатия равный 144/54 = 2,(6) для кодировки ASCII и 288/54=5,(3) для кодировки Unicode.

Одно из преимуществ метода упаковки заключается в том, что любой фрагмент сжатых данных можно распаковать, полностью восстановив исходных данных, совершенно не используя предшествующие данные. Но метод упаковки дает хорошие результаты, только если множество используемых символов, по отношению к полному алфавиту, невелико (см.пример 2). Если же в тексте используются практически все символы алфавита, то коэффициент сжатия окажется незначительным, а возможно, что сжать данные вообще не получиться.

Метод Хаффмана

Недостаток метода упаковки заключается в том, что все символы кодируются битовыми последовательностями одинаковой длины, а изучив раздел «Понятие информации. Измерение количества информации» мы знаем, что оптимального кодирования можно добиться, используя неравномерные коды, например код Хаффмана.

Напомним, что код Хаффмана является неравномерным и префиксным. Неравномерность означает, что те символы, которые встречаются в сообщении чаще, кодируются более короткими кодами, а символы, которые встречаются редко – более длинными. Префиксность говорит о том, что ни один код не является началом другого кода, что позволяет достичь однозначности при декодировании.

Сжатие методом Хаффмана выполняется за два прохода. На первом проходе читаются все входные данные и подсчитываются *частоты встречаемости* всех символов. Затем по этим данным строится *дерево Хаффмана*, по которому вычисляются коды символов. На втором проходе, входные данные читаются еще раз и перекодируются на основе новой кодовой таблицы.

Вычисление частоты встречаемости символов является тривиальной задаче, а построение кодов Хаффмана, а также кодирование и декодирование сообщений мы подробно рассмотрели в разделе «Понятие информация». Поэтому остановимся только на коэффициенте сжатия.

Пример 3. В примере 2 данного раздела мы вычислили объем сообщения «КОЛ ОКОЛО КОЛОКОЛА» при использовании кодировок ASCII (V_{ascii} =144 бита) и Unicode ($V_{unicode}$ = 288 бит). А в *примере 6* раздела «Понятие информации» мы построили код Хаффмана для этой же фразы, на основании которого оценили объем сообщения $V_{хаффмана}$=39 бит.

В результате мы получаем коэффициент сжатия равный 144/39 = 3,7 для кодировки ASCII и 288/39 =7,4 для кодировки Unicode.

Метод RLE

В основу алгоритмов RLE (Run-Length Encoding – кодирование путем учета числа повторений) положен принцип выявления повторяющихся последовательностей данных и замены их простой структурой: повторяющийся фрагмент и коэффициент повторения..

Рассмотрим идею сжатия методом RLE. Во входной последовательности:

1. все повторяющиеся цепочки байтов заменяются на фрагменты {управляющий байт, повторяющийся байт}, причем управляющий байт в десятичной системе счисления должен иметь значение *128 + число повторений байта.* Это означает, что старший бит управляющего байта для цепочки повторяющихся байтов всегда равен 1.
2. все неповторяющиеся цепочки байтов заменяются на фрагменты {управляющий байт, цепочка неповторяющихся байтов}, причем управляющий в десятичной системе счисления должен иметь значение *0+ длина цепочки.* Это означат, что старший бит управляющего байта для цепочки неповторяющихся байтов всегда равен 0.

Замечание. В обоих случаях длина обрабатываем фрагмента не может превосходить 127 байт.

Пример 4. Упакуем методом RLE следующую последовательность из 14-ти байтов:
11111111 11111111 11111111 11111111 11111111 11110000 00001111 11000011 10101010 10101010 10101010 10101010 10101010 10101010

В начале последовательности 5 раз повторяется байт 11111111. Чтобы закодировать эти 5-ть байтов, запишем вначале управляющий байт 10000101, а затем сам повторяющийся байт 11111111. Рассмотрим как получился управляющийся байт: к значению 128 мы прибавили число повторений байта 5, получили $133_{10}=10000101_2$.

Далее идут 3 разных байта. Чтобы их закодировать, мы записываем управляющий байт 00000011 ($11_2=3_{10}$), а затем эти неповторяющиеся байты.

Далее в последовательности 7 раз идет байт 10101010. Чтобы закодировать эти 7 байтов, мы записываем управляющий байт 10000111 ($128+7=135_{10}=1000111_2$), а затем повторяющийся байт 10101010.

В результате применения метода RLE мы получаем последовательность из 8-ми байтов:
10000101 11111111 00000011 11110000 00001111 11000011 10000111 10101010.

Таким образом, коэффициент сжатия 14/8=1,75.

Схема распаковки:

1. Если во входной (сжатой) последовательности встречается управляющий байт с единицей в старшем бите, то следующий за ним байт данный надо записать в выходную последовательность столько раз, сколько указано в оставшихся 7-ми битах управляющего байта.
2. Если во входной (сжатой) последовательности управляющий байт с нулем в старшем бите, то в выходную последовательность нужно поместить столько следующих за управляющим байтов входной последовательности, сколько указано в оставшихся 7-ми битах управляющего байта.

Пример 5. Распакуем сжатую последовательность данных методом RLE:
00000010 00001111 11110000 10000110 11000011 00000011 00001111 00111100 01010101.

Первый байт данной последовательности управляющий 000000010, начинается с 0. Следовательно, следующие за ним $10_2=2_{10}$ байта помещаются в выходную последовательность.

Замечание. Будем зачеркивать обработанные байты.

~~00000010 00001111 11110000~~ 10000110 11000011 00000011 00001111 00111100 01010101

Следующий управляющий байт 10000110 начинается с 1. Следовательно, следующий за ним байт нужно поместить в выходную последовательность $110_2=6_{10}$ раз.

~~00000010 00001111 11110000 10000110 11000011~~ 00000011 00001111 00111100 01010101

Следующий управляющий байт 00000011 начинается с 0. Следовательно, следующие за ним $11_2=3_{10}$ байта нужно поместить в выходную последовательность.

Таким образом, все входные данные мы обработали. А распакованная последовательность байтов выглядит следующим образом:
00001111 11110000 11000011 11000011 11000011 11000011 11000011 11000011 00001111 00111100 01010101

Наилучшими объектами для данного алгоритма являются графические данные, в которых встречаются большие одноцветные участки. Различные реализации данного метода используются в графических форматах PCX, BMP и факсимильной передаче информации. Для текстовых данных данный метод неэффективен.

Специальные методы сжатия

Проведенные в конце XX века исследования психофизиологических характеристик зрения и слуха обнаружили ряд особенностей человеческого восприятия информации, использование которых позволяет существенно увеличивать степень сжатия звуковой, графической и видеоинформации.

Например, было установлено, что глаз человека наиболее чувствителен к зеленому цвету, чувствительность к красному ниже примерно в 4 раза, а к синему — почти в 10 раз! Это означает, что на хранение информации о красной и синей составляющих цвета можно было бы отводить меньше бит, чем на зеленую составляющую.

К середине 90-х годов прошлого века были разработаны высокоэффективные методы сжатия графической, звуковой и видео информации, учитывающей особенности человеческого зрения и слуха. Характерной чертой этих методов является возможность регулируемого удаления маловажной для человеческого восприятия информации, за счет чего удается достичь высоких коэффициентов сжатия. Но т.к. часть данных удаляется (безвозвратно), то полное восстановление исходной информации невозможно.

Наиболее известными методами сжатия с регулируемой потерей информации являются:
- JPEG — метод сжатия графических данных;
- MP3 — метод сжатия звуковых данных;
- MPEG — группа методов сжатия видеоданных.

Эти методы непросты в реализации, в них используется достаточно сложный математический аппарат, поэтому мы рассмотрим лишь обзорное описание данных методов.

Алгоритм JPEG

Алгоритм JPEG используется для сжатия статических изображений.

Сжатие JPEG осуществляется в несколько этапов: сперва цвета пикселей переводятся из RGB-представления в YCbCr-представление (в данной модели цвет представляется компонентами «яркость» Y, «цветоразность зеленый-красный» Cr и «цветоразность зеленый-синий» Cb). Затем в каждой второй строке и каждом втором столбце матрицы пикселей информация о цветовых компонентах Cb и Cr просто удаляется, что мгновенно уменьшает объем данных вдвое. Оставшиеся данные подвергаются спе-

циальной процедуре «сглаживания», при которой объем данных не изменяется, но потенциальная степень их сжимаемости резко увеличивается (на этом этапе учитывается коэффициент сжатия). Затем данные сжимаются алгоритмом Хаффмана.

Алгоритм JPEG способен упаковывать графические изображения в несколько десятков раз, при этом потери качества становятся заметными только при очень высоких коэффициентах сжатия.

Алгоритм MP3

Сжатие MP3 является частью стандарта MPEG и применяется для сжатия аудиоинформации. По мимо сжимаемой информации алгоритму передается желаемый битрейт – количество бит, используемых для кодирования одной секунды звука.Этот параметр регулирует долю информации, которая будет удаляться.

Сжатие MP3 также осуществляется в несколько этапов: звуковой фрагмент разбивается на небольшие участки — *фреймы,* а в каждом фрейме звук разлагается на составляющие звуковые колебания, которые в физике называют гармониками. Затем начинается психоакустическая обработка — удаление маловажной для человеческого восприятия звуковой информации. Желаемый битрейт определяет, какие эффекты будут учитываться при сжатии, а также какое количество информации будет удалятся. На следующем этапе оставшиеся данные сжимаются алгоритмом Хаффмана.

Алгоритм MP3 позволяет сжимать звуковые файлы в несколько раз. Даже при самом «плохом» раскладе обеспечивается четырехкратное сжатие аудиоинформации.

Алгоритмы MPEG

MPEG — это целое семейство методов сжатия видеоданных. В них используется очень большое количество приемов сжатия.

Первый прием - использование «опорного кадра» — заключается в том, чтобы сохранять не целиком кадры, а только изменения кадров. Например, в фильме есть сцена беседы героев в комнате. При этом от кадра к кадру меняются только выражения лиц, а большая часть изображения неподвижна. Закодировав первый кадр сцены и отличия остальных ее кадров от первого, можно получить очень большую степень сжатия.

Следующий прием заключается в том, чтобы быстро сменяемые участки изображения кодировать с качеством, которое намного ниже качества статичных участков, — человеческий глаз не успевает рассмотреть их детально.

Кроме того, формат MPEG позволяет сохранять в одном файле несколько так называемых потоков данных. Так, в основном потоке можно сохранить фильм, в другом — логотип (храниться один раз, а не в каждом кадре), в третьем — субтитры (как текст), и т. д. Потоки данных накладываются друг на друга только при воспроизведении.

Разновидности формата MPEG отличаются друг от друга по возможностям, качеству воспроизводимого изображения и максимальной степени сжатия:

- MPEG-1 — использовался в первых Video CD (VCD-I);
- MPEG-2 — используется в DVD и Super Video CD (SVCD, VCD-II);
- MJPEG — формат сжатия видео, в котором каждый кадр сжимается по методу JPEG;
- MPEG-4 — усовершенствованный формат сжатия видео;
- DivX, XviD — улучшенные модификации формата MPEG-4.