Акимушкин Василий Александрович,
Поздняков Сергей Николаевич,
Пухов Алексей Фёдорович

АЛГОРИТМ ХАФФМАНА

Всем известен морской сигнал бедствия SOS. Но задумывались ли вы когда-нибудь, почему сигнал составлен именно из этих трех букв? Многие скажут, что это аббревиатура предложения, например такого «*Save Our Souls*» («спасите наши души»), но истинная причина в том, что этот сигнал стал одним из первых применений радиотелеграфа на судах. Для передачи сигнала служил ключ, на который нажимали рукой, электрическая цепь замыкалась, и генерировался сигнал (рис. 1). Длинный сигнал обозначался как тире, а короткий точкой. С помощью тире и точек кодировались буквы алфавита.

Рис. 1

Увлеченные сборкой радиоприборов школьники знали наизусть азбуку Морзе, которую придумал для передачи слов изобретатель телеграфа Сэмюэл Морзе (рис. 2).

Чтобы быстро выучить буквы придумывали созвучные предложения, в которых точка читалась как короткое «ти», а тире, как длинное «таа». Например, буква «Ф», которая кодировалась так «ти-ти-таа-ти» (· · – ·) заучивалось так «тё-тя Каа-тя». В кодировке Морзе сигнал SOS звучит так «ти-ти-ти-таа-таа-таа-ти-ти-ти» (· · · – – – · · ·), то есть он очень прост как для передачи, так и для приема.

Однако после того как появились более удобные средства передачи информации, азбука Морзе потеряла своё значение. Почему?

Во-первых, легко заметить, что для передачи используются на самом деле не два

Всем известен морской сигнал бедствия SOS...

символа, а три. Третий – пауза между буквами. Если в тексте не писать паузы, то прочитать текст можно по другому: например, фраза «И ТЕ» будет записана тем же набором точек и тире, что и буква «Ф».

Другой проблемой является избыточная длина сообщений. В разных сообщениях разные символы используются с разной частотой, а при кодировке Морзе учесть это для сокращения длины сообщений невозможно.

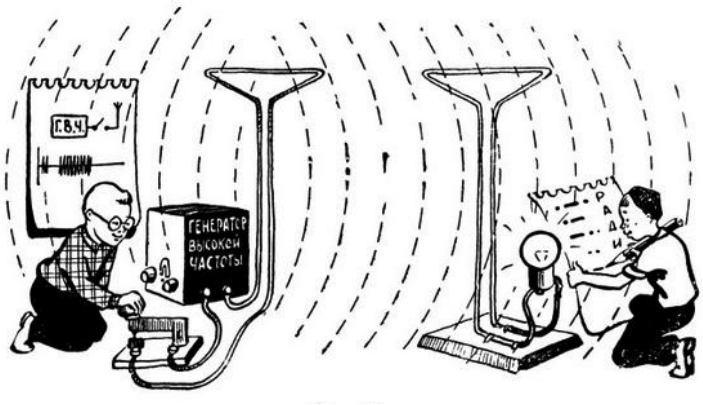

Рис. 2

Сейчас, после появления компьютеров, вопросы сжатия информации стали весьма актуальными. Например, когда фотография кодируется в формате JPEG, она занимает места раз в десять меньше, чем, когда она сохраняется в формате BMP. В первом случае используется алгоритм кодирования, называемый кодом Хаффмана, который позволяет сжимать информацию, которая в формате BMP хранится в более простой кодировке (перечисление параметров всех точек изображения).

Например, картинка с символикой КИО Школы (рис. 3) (в оригинале цветная), размеры которой 137 на 76 пикселей (всего 10412 пикселей), занимает 31366 байт в кодировке BMP и более чем в 10 раз меньше (2766 байт) в кодировке JPG.

На этом занятии мы познакомимся с кодировкой Хаффмана.

Для начала попробуем закодировать какое-нибудь слово, используя только два символа. Вместо точки и тире удобнее использовать 0 и 1.

Задача. *Можно ли для кодирования простых слов из четырёх различных букв А, М, С, Л использовать следующую кодировку?*

А – 0
М – 10
И – 101
Л – 1

Попробуйте расшифровать два сообщения:

0001 и 1010110.

В первом случае вы сразу скажете, что это АААЛ.

Во втором случае, в качестве ответа могут появиться такие слова как МЛАЛЛА, ЛАМЛЛА, МИМ, МИЛА и т. д.

Почему наша кодировка оказалась плоха для второго слова? Потому что коды некоторых символов начинаются с кодов других (кодировка М, начинается с кода Л, кодировка И с кода М). Поскольку с нуля начинался только один код – код буквы А, в первом примере слово удалось расшифровать (декодировать) однозначно.

Определение. *Кодировка называется префиксной, если ни один код символа не начинается с кода другого символа.*

Если Вы захотите придумать свою кодировку, соблюдайте правило ПРЕФИКСНОСТИ кода!

Например, кодировку из предыдущего примера можно подправить так:

А – 01
М – 11
И – 101
Л – 100

Тогда слово МИЛА будет записано как

Рис. 3

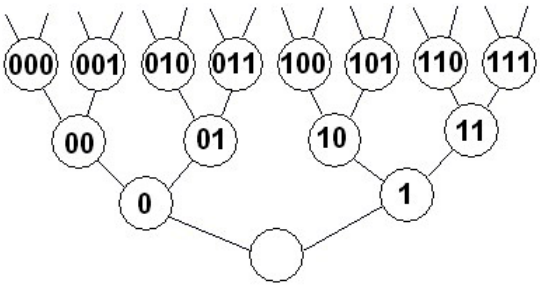

Рис. 4

1110110001, и расшифровать иначе у вас не получится!

Итак, одну проблему мы решили.

Теперь попробуем оптимизировать код так, чтобы длина данного сообщения в двоичной кодировке (кодировке двумя символов) была как можно меньше.

Сначала выясним, как строить префиксные коды в общем случае.

Для алфавита с двумя символами вариант единственный – одну букву обозначить 0, другую 1. Если в алфавите букв три, то придется использовать более длинные кодовые цепочки, например, первую букву обозначить 0, вторую и третью 10 и 11. Второй вариант получится, если 0 и 1 поменять местами (и других вариантов префиксного кодирования не будет). Если в алфавите четыре буквы, то вариантов кодирования больше и они более разнообразны по длине кодовых цепочек. Например, можно рассмотреть кодирование цепочками равной длины: 00, 01, 10, 11, но можно выбрать и неравномерный вариант: 0, 10, 110, 111. Префиксные коды легко строятся с помощью двоичного дерева. Это дерево рисуется так: от корня идут две ветки, левой соответствует 0, правой 1 (или наоборот, это несущественно). Затем на концах ветвей строятся новые вершины, и процесс повторяется. В каждой вершине записывается её код – последовательность 0 и 1, по которой из корня попадаем в эту вершину (рис. 4).

Чтобы построить префиксный код, нужно «обрезать» каким-нибудь образом ветви двоичного дерева и отметить вершины, от которых не растут вверх ветви. Такие вершины называются листьями. Цепочки 0 и 1, записанные в листьях, образуют префиксный код. Доказать это нетрудно: двигаясь от корня к листу, мы проходим через вершины, коды которых являются начальными частями кодов листьев. Поскольку эти вершины в код не включаются, он получается префиксным (рис. 5).

Замечание. Наверное, всем приходилось иметь дело с «тарабарским» языком, который появляется на некоторых страницах в Интернете или в приходящих по почте письмах. «Ошибка кодировки» – так называется эта проблема. Исправляя кодировку страницы, вы наверняка заметили, что разных кодов за время существования компьютеров накопилось много. Почему это произошло? Совсем недавно в Интернете был распространен код ASCII, который позволяет кодировать 128 разных букв и символов. Его вполне хватало для латинского текста (больших и малых букв) и специальных знаков, но при этом каждой стране со своим алфа-

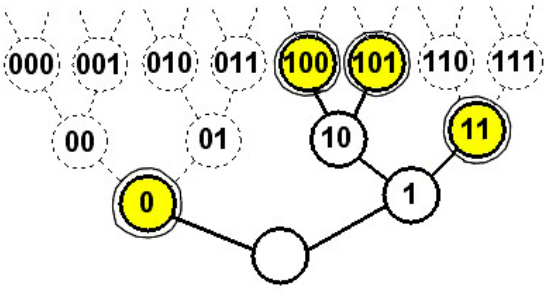

Рис. 5

Чтобы построить префиксный код, нужно «обрезать» каким-нибудь образом ветви двоичного дерева...

витом (кириллицей, иероглифами и пр.) приходилось использовать оставшиеся коды для национального алфавита. Получалось, что в разных странах разные буквы кодировались одинаково. Сейчас в Интернете вы можете видеть страницы на разных языках. Это связано с переходом к новому стандарту кодирования, который называется Юникод, и который реализован в кодировке UTF-8. UTF-8 является префиксным кодом. Для того чтобы согласовать его со старым кодированием ASCII, было решено старые коды начинать с 0, а новые с 1, поэтому даже если программа не распознаёт Юникод, то латинские буквы, арабские цифры и знаки препинания будут отображаться правильно.

Теперь обсудим проблему неравномерности различных символов в сообщении. Как, например, лучше зашифровать буквы для передачи слова АБРАКАДАБРА, используя префиксный код? Всего в этом слове пять различных букв: А, Б, Р, К, Д.

Рассмотрим два способа кодировки:

А – 000, Б – 001, Р – 01, К – 10, Д – 11

А – 0, Б – 100, Р – 101, К – 110, Д – 111

Во втором случае суммарная длина кодов букв (13) больше, чем в первом коде (12) букв, однако слово во второй кодировке существенно короче:

1-ая кодировка АБРАКАДАБРА = 000001010001000011000000101000

2-ая кодировка АБРАКАДАБРА = 010010101100111010010 10

Упражнение. Предложите своим друзьям раскодировать приведенные двоичные последовательности (разумеется, им надо сообщить коды букв). Для решения им понадобится, двигаясь слева направо, сначала выделить код первого символа (в силу префиксности кода имеется только один вариант для этого), затем второго и т. д.

Почему вторая кодировка оказалась лучше? Потому что буква А встречается существенно чаще, чем другие буквы, и выгоднее закодировать её коротким кодом, а для редких букв оставить длинные коды.

Как, зная частоту букв, рассчитать длину кодов оптимальным образом? Эту задачу и решает алгоритм Хаффмана.

... зашифровать буквы для передачи слова АБРАКАДАБРА, используя префиксный код?

На первом шаге строится дерево, «листьями» которого являются буквы. На каждом «листочке» записывается частота этой буквы. Затем находятся две буквы с наименьшими частотами и заменяются одной буквой, частота которой вычисляется сложением частот исходных букв (заметьте, что мы иногда делаем так и с обычными буквами, например, вместо букв Е и Ё используем одну букву Е). В получившемся новом алфавите снова выбираются две самые редкие буквы и заменяются одной буквой с суммарной частотой исходных букв и так далее, пока не останутся две буквы (в приводимом примере для каждой буквы указано число её повторений в тексте; при построении дерева, чтобы не вводить новые символы, имена сливаемых пар составляются из имён исходных букв) (рис. 6).

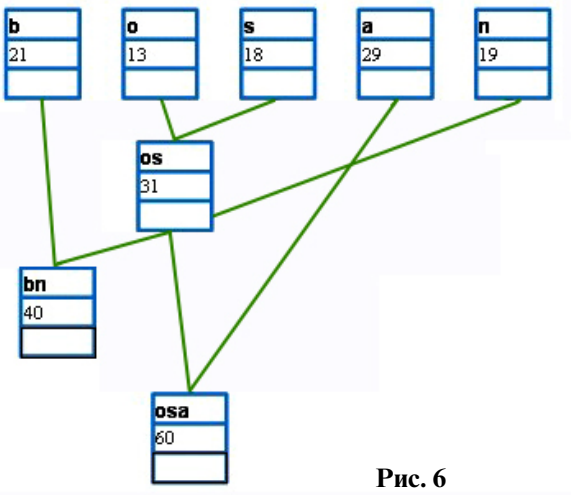

Рис. 6

На следующем этапе последним двум символам даётся кодировка 0 и 1, а далее мы движемся вверх и добавляем к этим кодам 0 или 1, в зависимости от того, по какой ветви поднимаемся (рис. 7).

В результате исходные символы получают оптимальное двоичное кодирование (стоят в верхней строке и являются листьями дерева частот).

Пример. Длина кода, показанного на рисунке, будет

$2 \cdot 21 + 3 \cdot 13 + 3 \cdot 18 + 2 \cdot 29 + 2 \cdot 19 = 231.$

Докажем, что сформулированный алгоритм действительно даёт оптимальный код, то есть, длина получившегося кода не может быть уменьшена.

Сначала докажем вспомогательное утверждение.

Лемма. *Возьмём два символа с наименьшими частотами. Можно построить оптимальный префиксный код, в котором коды этих символов имеют одинаковую максимальную длину и отличаются лишь последним битом.*

Доказательство. Сначала докажем, что символ, который встречается реже, не может иметь более короткий код при оптимальном кодировании. От противного: пусть такой код нашелся. Тогда имеем символ с частотой N и длиной кода K и другой символ с частотой $(N-n)$ и длиной $(K-k)$. Суммарная длина кодов этих символов в сообщении будет $N \cdot K + (N-n) \cdot (K-k)$. Поменяем коды этих символов. Тогда суммарная длина кодов станет равной $N \cdot (K-k) + (N-n) \cdot K$. Какое из выражений короче? Второе, так как разность первого и второго положительна: $N \cdot K + (N-n) \cdot (K-k) - (N \cdot (K-k) + (N-n) \cdot K) = nk$.

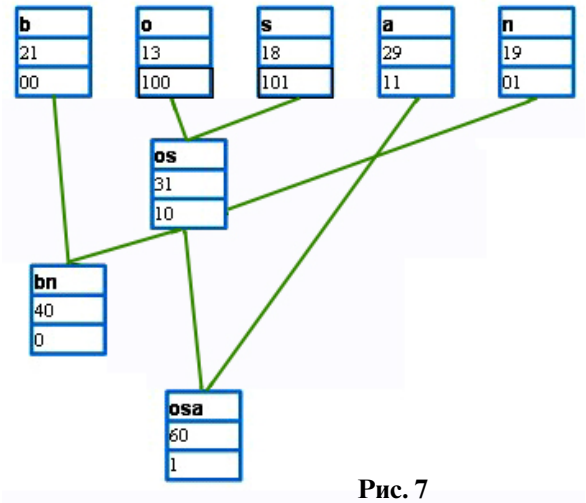

Рис. 7

Заметим также, что максимальную длину кода при оптимальном префиксном кодировании не может иметь только один символ. Действительно, если бы такой символ был, то из его кода можно было бы убрать последний двоичный знак (и этот новый код не может оказаться кодом другого символа вследствие префиксности кода). Полученный код будет короче, следовательно, исходный код не мог быть оптимальным.

Теорема. *Представленный выше алгоритм Хаффмана строит оптимальный префиксный код.*

Доказательство. Докажем по индукции.

База индукции. Если в алфавите всего две буквы, алгоритм, очевидно, работает правильно, так как один символ будет закодирован как 0, второй как 1. Короче закодировать в этом случае нельзя.

Индукционный переход. Предположим для $(n-1)$ символа алгоритм работает правильно.

Докажем, что алгоритм будет оптимальным для кодирования n символов. Для доказательства соединим два символа с наименьшими частотами в один (присвоив ему суммарную частоту этих символов), как это делает алгоритм Хаффмана. Теперь символов станет $(n-1)$. По индукционному предположению оптимальное кодирование для $(n-1)$ символа осуществляется алгоритмом Хаффмана. Выполним это кодирование, после чего «разделим» сдвоенный символ, добавив к концу кода 0 и 1, как делает в алгоритм Хаффмана. Получим кодировку для всех n символов. Будет ли она оптимальной? Пусть это не так и существует лучшая кодировка. Возьмём эту кодировку. По лемме символы с наименьшими частотами будут иметь наибольшие (и одинаковые) длины кодов, которые отличаются в последнем двоичном разряде. Заменим эти два символа одним символом с кодом, получающим-

ся из кодов исходных букв отбрасыванием последнего разряда. Эта кодировка по индукционному предположению должна иметь те же длины кодов, что и в алгоритме Хаффмана, значит, совпадут длины кодов всех n символов.

Замечание. Нетрудно понять, что восхождение по дереву кодов может дать различные оптимальные кодировки (перестановкой 0 и 1 в левой и правой ветвях). Их можно рассматривать как различные вариации алгоритма Хаффмана. Кроме того, можно придумать другие алгоритмы оптимальной кодировки, отличные от алгоритма Хаффмана. Они дадут результаты кодирования, отличающиеся от результатов алгоритма Хаффмана перестановкой кодов символов одинаковой длины.

Например, в предыдущем примере можно закодировать буквы так:
b – 00, o – 100, s –101, a – 01, n – 11. Здесь коды b и n отличаются в двух разрядах, чего не могут дать вариации алгоритма Хаффмана.

От редакции: На сайте Интернет-школы современной информатики и дискретной математики http://kioschool.eltech.ru/, открытой Факультетом компьютерных технологий и информатики Санкт-Петербургского электротехнического университета (ЛЭТИ) совместно с Центром информатизации образования «КИО» – учредителем одноименного конкурса КИО («Конструируй, исследуй, оптимизируй»), можно зарегистрироваться на обучение в Школе и получить доступ к тренажёрам и модулям контроля.

Акимушкин Василий Александрович,
аспирант математико-механического факультета СПбГУ,
программист АНО «КИО»,

Поздняков Сергей Николаевич,
доктор педагогических наук,
профессор кафедры ВМ-2
СПбГЭТУ «ЛЭТИ»,
научный руководитель Интернет-школы современной информатики и дискретной математики,

Пухов Алексей Фёдорович,
аспирант математико-механического факультетаСПбГУ.