БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра РЭС

Реферат на тему:

**«Словарные методы кодирования. Метод Зива-Лемпела. Дифференциальное кодирование»**

МИНСК, 2009

**Словарные методы кодирования. Метод Зива-Лемпела**

Практически все словарные методы кодирования пpинадлежат семье алгоритмов из работы двух израильских ученых - Зива и Лемпела, опубликованной в 1977 году. *Сущность их состоит в том, что фразы в сжимаемом тексте заменяются указателем на то место, где они в этом тексте уже pанее появлялись*.

Это семейство алгоритмов называется методом Зива-Лемпела и обозначается как *LZ-сжатие*. Этот метод быстpо пpиспосабливается к стpуктуpе текста и может кодировать короткие функциональные слова, так как они очень часто в нем появляются. Новые слова и фразы могут также формироваться из частей ранее встреченных слов.

Декодирование сжатого текста осуществляется напрямую - происходит простая замена указателя готовой фразой из словаря, на которую тот указывает. На практике LZ-метод добивается хорошего сжатия, его важным свойством является очень быстрая работа декодера. (Когда мы говорим о тексте, то предполагаем, что кодированию подвергается некоторый вектор данных с конечным дискретным алфавитом, и это не обязательно текст в буквальном смысле этого слова.)

Большинство словарных методов кодирования носят имя авторов идеи метода Зива и Лемпела, и часто считают, что все они используют один и тот же алгоритм кодирования. На самом деле разные представители этого семейства алгоритмов очень сильно различаются в деталях своей работы.

*Все словарные методы кодирования можно разбить на две группы.*

Методы, принадлежащие к *первой группе,* находя в кодируемой последовательности цепочки символов, которые ранее уже встречались, вместо того, чтобы повторять эти цепочки, заменяют их указателями на предыдущие повторения.

Словарь в этой группе алгоритмов в неявном виде содержится в обрабатываемых данных, сохраняются лишь указатели на встречающиеся цепочки повторяющихся символов.

Все методы этой группы базируются на алгоритме, разработанном и опубликованном, как уже отмечалось, сравнительно недавно - в 1977 году Абрахамом Лемпелем и Якобом Зивом, - LZ77. Наиболее совершенным представителем этой группы, включившим в себя все достижения, полученные в данном направлении, является алгоритм LZSS, опубликованный в 1982 году Сторером и Шимански.

Процедура кодирования в соответствии с алгоритмами этой группы иллюстрируется рис. 1.

.

.

.

**P**

.

.

.

**Выходные данные**

**Входные данные**

Рис. 1

Алгоритмы *второй группы* в дополнение к исходному словарю источника создают словарь фраз, представляющих собой повторяющиеся комбинации символов исходного словаря, встречающиеся во входных данных.

При этом размер словаря источника возрастает, и для его кодирования потребуется большее число бит, но значительная часть этого словаря будет представлять собой уже не отдельные буквы, а буквосочетания или целые слова.

Когда кодер обнаруживает фразу, которая ранее уже встречалась, он заменяет ее индексом словаря, содержащим эту фразу. При этом длина кода индекса получается меньше или намного меньше длины кода фразы.

Все методы этой группы базируются на алгоритме, разработанном и опубликованном Лемпелем и Зивом в 1978 году, – LZ78. Наиболее совершенным на данный момент представителем этой группы словарных методов является алгоритм LZW, разработанный в 1984 году Терри Вэлчем.

Идею этой группы алгоритмов можно также пояснить с помощью рис. 2.

.

.

.

.

.

.

**Выходные данные**

**Входные данные**

**1.**

**4**

**Словарь**

**1.**

**2.**

**3.**

**4.**

**5.**

Рис. 2

Алгоритмы второй группы несколько проще в объяснении их работы, поэтому начнем рассмотрение принципа действия LZ-кодеров с алгоритма LZW.

Рассмотрим в самом общем виде работу LZW-кодера и декодера.

### **Процедура кодирование**

Процесс сжатия выглядит достаточно просто. Мы последовательно считываем символы входного потока (строку) и проверяем, есть ли в уже созданной нами таблице такая строка.

Если строка есть, то считываем следующий символ, а если такой строки нет, - заносим в выходной поток код для предыдущей найденной строки, заносим ее в таблицу и начинаем поиск снова.

Пусть на вход кодера поступает последовательность символов вида **/ WED / WE / WEE / WEB**, при этом размер алфавита входных символов ***dim A = 255.***

Схема сжатия выглядит следующим образом:

Входные символы Выходной код Новые символы словаря

 ***/W /*** 256 = ***/W***

 ***E W*** 257 = ***WE***

 ***D E*** 258 = ***ED***

 ***/ D*** 259 = ***D/***

 ***WE 256*** 260 = ***/WE***

 ***/ E*** 261 = ***E/***

 ***WEE 260*** 262 = ***/WEE***

 ***/W 261*** 263 = ***E/W***

 ***EB 257*** 264 = ***WEB***

 ***<END> B***

В результате получим выходной код

***/WED<256>E<260><261><257>B***.

 Как при этом изменилась длина выходного кода в сравнении с входным ?

Если для двоичного кодирования строки ***/ WED / WE / WEE / WEB*** длиной в 15 букв и размером алфавита ***dim A = 255*** нам понадобилось бы ***15 • log2 255 = 15х8 = 120 бит***, то для двоичного кодирования выходной строки кодера ***/ WED <256> E <260> <261> <257> B*** длиной в 10 новых символов с алфавитом в 264 буквы – ***10 • 9 = 90 бит.***

###

### **Поцедура декодирование**

LZW-декодер, обрабатывая входной поток закодированных данных, восстанавливает из него исходные данные. Так же, как и алгоритм сжатия, декодер добавляет новые строки в словарь всякий раз, когда находит во входном потоке новый код. Все, что ему остается сделать, – это преобразовать входной код в выходную строку символов и отдать ее на выход кодера.

Схема работы LZW-декодера выглядит следующим образом:

строка на входе кодера - ***/WED<256>E<260><261><257>B.***

Входные символы Выходная строка Новые символы словаря

 ***/ /***

 ***W W*** 256 ***= /W***

 ***E E*** 257 ***= WE***

 ***D D*** 258 ***= ED***

 ***256 /W*** 259 ***= D/***

 ***E E*** 260 ***= /WE***

 ***260 /WE 261 = E/***

 ***261 E/*** 262 ***= /WEE***

 ***257 WE 263 = E/W***

 ***B B*** 264 ***= WEB***

Самым замечательным качеством этого способа сжатия является то, что *весь словарь новых символов передается декодеру без собственно передачи*. В конце процесса декодирования декодер имеет точно такой же словарь новых символов, какой в процессе кодирования был накоплен кодером, при этом его создание было частью процесса декодирования.

Работа кодера/декодера семейства LZ77 - первой опубликованной версии LZ-метода - выглядит несколько иначе.

В алгоритме LZ77 указатели обозначают фразы в окне постоянного pазмеpа, пpедшествующие позиции кода. Максимальная длина заменяемых указателями подстрок определяется параметром ***F*** (обычно это от 10 до 20). Эти ограничения позволяют LZ77 использовать "скользящее окно" из ***N*** символов. Из них первые ***N-F*** были уже закодированы, а последние ***F*** составляют упреждающий буфер.

При кодировании символа в первых ***N-F*** символах окна ищут самую длинную, совпадающую с этим буфером строку. Она может частично перекрывать буфер, но не может быть самим буфером.

Найденное наибольшее соответствие затем кодируется триадой [***i, j, a***] где ***i*** есть его смещение от начала буфера, ***j*** - длина соответствия, ***a*** - первый символ, не соответствующий подстроке окна.

Затем окно сдвигается вправо на ***j +1*** символ и готово к новому шагу алгоритма.

Привязка определенного символа к каждому указателю гарантирует, что кодирование будет выполняться даже в том случае, если для первого символа упpеждающего буфера не будет найдено соответствие.

Объем памяти, требуемый кодеру и декодеру, ограничивается размером окна. Количество бит, необходимое для представления смещения ( ***i*** ) в триаде, составляет [***log(N-F)***]. Количество символов ( ***j*** ), заменяемых триадой, может быть закодировано [***logF***] битами.

Декодирование осуществляется очень просто и быстро. При этом поддерживается тот же порядок работы с окном, что и при кодировании, но в отличие от поиска одинаковых строк он, наоборот, копирует их из окна в соответствии с очередной триадой.

## Дифференциальное кодирование

Работа дифференциального кодера основана на том факте, что для многих типов данных разница между соседними отсчетами относительно невелика, даже если сами данные имеют большие значения. Например, нельзя ожидать большой разницы между соседними пикселами цифрового изображения.

Следующий простой пример показывает, какое преимущество может дать дифференциальное кодирование (кодирование разности между соседними отсчетами) в сравнении с простым кодированием без памяти (кодированием отсчетов независимо друг от друга).

Просканируем 8-битовое (256-уровневое) цифровое изображение, при этом десять последовательных пикселов имеют уровни:

144, 147, 150, 146, 141, 142, 138, 143, 145, 142.

Если закодировать эти уровни пиксел за пикселом каким-либо кодом без памяти, использующим 8 бит на пиксел изображения, получим кодовое слово, содержащее 80 бит.

Предположим теперь, что прежде чем подвергать отсчеты изображения кодированию, мы вычислим разности между соседними пикселами. Эта процедура даст нам последовательность следующего вида:

144, 147, 150, 146, 141, 142, 138, 143, 145, 142.

 ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓

144, 3, 3, - 4, - 5, 1, - 4, 5, 2, -3.

Исходная последовательность может быть легко восстановлена из разностной простым суммированием (дискретным интегрированием):

144, 144+3, 147+3, 150–4, 146–5, 141+1, 142–4, 138+5, 143+2, 145-3

 ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓

144, 147, 150, 146, 141, 142, 138, 143, 145, 142.

Для кодирования первого числа из полученной последовательности разностей отсчетов, как и ранее, понадобится 8 бит, все остальные числа можно закодировать 4-битовыми словами (один знаковый бит и 3 бита на кодирование модуля числа ).

Таким образом, в результате кодирования получим кодовое слово длиной ***8 + 9\*4 = 44*** бита или почти вдвое более короткое, нежели при индивидуальном кодировании отсчетов.

Метод дифференциального кодирования очень широко используется в тех случаях, когда природа данных такова, что их соседние значения незначительно отличаются друг от друга, при том, что сами значения могут быть сколь угодно большими.

Это относится к звуковым сигналам, особенно к речи, изображениям, соседние пиксели которых имеют практически одинаковые яркости и цвет и т.п. В то же время этот метод совершенно не подходит для кодирования текстов, чертежей или каких-либо цифровых данных с независимыми соседними значениями.

**ЛИТЕРАТУРА**

1. Лидовский В.И. Теория информации. - М., «Высшая школа», 2002г. – 120с.
2. Метрология и радиоизмерения в телекоммуникационных системах. Учебник для ВУЗов. / В.И. Нефедов, В.И. Халкин, Е.В. Федоров и др. – М.: Высшая школа, 2001 г. – 383с.
3. Цапенко М.П. Измерительные информационные системы. – М.: Энергоатом издат, 2005. - 440с.
4. Зюко А.Г., Кловский Д.Д., Назаров М.В., Финк Л.М. Теория передачи сигналов. М: Радио и связь, 2001 г. –368 с.
5. Б. Скляр. Цифровая связь. Теоретические основы и практическое применение. Изд. 2-е, испр.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003 г. – 1104 с.