Практическая работа № 7. Сжатие данных с помощь метода RLE.

Цель работы: научиться сжимать информацию с помощью метода RLE.

Задание также выставлено в Гугл классе

Срок сдачи до 21.00

ТЕОРИТИЧЕСКАЯ ЧАСТЬ

Метод RLE.

Наиболее известный простой подход и алгоритм сжатия информации обратимым путем - это кодирование серий последовательностей (Run Length Encoding - RLE). Суть методов данного подхода состоит в замене цепочек или серий повторяющихся байтов или их последовательностей на один кодирующий байт и счетчик числа их повторений. Проблема всех аналогичных методов заключается лишь в определении способа, при помощи которого распаковывающий алгоритм мог бы отличить в результирующем потоке байтов кодированную серию от других - некодированных последовательностей байтов. Решение проблемы достигается обычно простановкой меток в начале кодированных цепочек. Такими метками могут быть, например, характерные значения битов в первом байте кодированной серии, значения первого байта кодированной серии и т.п. Данные методы, как правило, достаточно эффективны для сжатия растровых графических изображений (BMP, PCX, TIF, GIF), т.к. последние содержат достаточно много длинных серий повторяющихся последовательностей байтов. Недостатком метода RLE является достаточно низкая степень сжатия или стоимость кодирования файлов с малым числом серий и, что еще хуже - с малым числом повторяющихся байтов в сериях.

✓ Пример решения: Закодировать последовательность: **BBBBBBACCCABBBBBB**.

1) ищем цепочки (упакованная последовательность содержит управляющие байты, за каждым управляющим байтом следует один или несколько байтов данных)

6B 1A 3C 1A 6B

2) кодируем управляющие байты

• если следующий за управляющим байт данных при распаковке нужно повторить, то в старший бит управляющего байта записывается 1, а в оставшихся 7 битах управляющего байта сколько раз нужно повторить байт данных

• если следующий за управляющим один или несколько байтов данных данных при распаковке повторять не нужно, то в старший бит управляющего байта записывается 0, а в оставшихся 7 битах управляющего байта сколько следующих байтов данных надо взять без изменения

6B

6 - количество повторов символа (**B** - символ)

$6_{10} = 110_2$

дополняем до байта 0 слева

00000110

заменяем старший бит на 1, т.к. надо повторять

10000110

разбиваем на тетрады (для перевода в 16 систему счисления)

86 B

1A

1 - количество повторов символа, т.е. символ не повторяется, а встречается 1 раз (**A** - символ)

$1_{10} = 1_2$

дополняем до байта 0 слева

00000001
оставляем старший бит 0, т.к. символ не повторяется, а после него идёт повтор (3C)
00000001
разбиваем на тетрады (для перевода в 16 систему счисления)
01 A

3C
3 - количество повторов символа (**C** - символ)
$3_{10} = 11_2$
дополняем до байта 0 слева
00000011
заменяем старший бит на 1, т.к. надо повторять
10000011
разбиваем на тетрады (для перевода в 16 систему счисления)

83 B
потом повторяется 2 и 1 цепочки:
1A → 01 A
6B → 86 B
3) кодируем байты данных
• для определения шестнадцатеричных кодов символов используем таблицу ASCII (скрин 1)
• в первом столбце записана первая цифра шестнадцатеричного кода символа
• в первой строке записана вторая цифра шестнадцатеричного кода символа
код большой (прописной) латинской буквы «**A**» с помощью шестнадцатеричных цифр можно записать как **41**
код большой (прописной) латинской буквы «**B**» с помощью шестнадцатеричных цифр можно записать как **42**
код большой (прописной) латинской буквы «**C**» с помощью шестнадцатеричных цифр можно записать как **43**
4) собираем всё вместе: 86 42 01 41 83 43 01 41 86 42

✓ Пример: Раскодировать: 85 49 82 4E 01 46

1) Выделенное желтым = кол-во повторений (можно узнать воспользовавшись таблицой тетрад)

Красным = закодированный символ (декодируется с помощью таблицы ASCII)

2) 85

Переводим с помощью таблицы в двоичный вид : 1000 0101 – т.к. 1-й бит указывает на то, есть повторения символа или нет, наличие еденицы говорит о повторе, оставшиеся 7 бит закодировано кол-во повторений. $101_2 = 5$

3) 49

По таблице ASCII можно узнать что это символ I.

4) 82

Вновь воспользуемся таблицой тетрад и узнаем что 82- 10000010, что в свою очередь означает **повторить 2 раза**

5) 4E

По таблице ASCII можно узнать что это символ N.

6) 01

01=00000001, 1й бит занимает ноль, это значит что ко-ло повторений отсутствует, следоварельно кол-во сиволов 1.

7) 46

По таблице ASCII можно узнать что это символ F.

8) Соединяем : IIIIINNF

ПРАКТИЧЕСКАЯ ЧАСТЬ

Алгоритм RLE

Для определения символов по их шестнадцатеричным кодом используйте приведённую ниже таблицу ASCII. В первом столбце записана первая цифра шестнадцатеричного кода символа, а в первой строке – вторая. Например, символ «&» имеет шестнадцатеричный код 26_{16}.

	.0	**.1**	**.2**	**.3**	**.4**	**.5**	**.6**	**.7**	**.8**	**.9**	**.A**	**.B**	**.C**	**.D**	**.E**	**.F**
0.	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1.	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2.		!	"	#	$	%	&	'	(	)	*	+	,	—	.	/
3.	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4.	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5.	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6.	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7.	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

1. Используя алгоритм RLE, закодируйте последовательность символов

   ABACBBBBAAAAAAAAAAA

   Запишите результат в виде шестнадцатеричных кодов (каждый символ кодируется в виде байта, который представлен двумя шестнадцатеричными цифрами).

   Ответ:

2. Раскодируйте последовательность, упакованную с помощью алгоритма RLE (приводятся шестнадцатеричные коды): **84 5A 83 49 83 4D 83 41**.

   Ответ:

3. Определите количество байтов в исходной и распакованной последовательности (см. предыдущее задание) и вычислите коэффициент сжатия:

   Формула $K = \frac{\text{Ис.п}}{\text{Сж.п}}$

Сжатая последовательность	**Несжатая последовательность**	**Коэффициент сжатия**