ПРАКТИЧЕСКАЯ РАБОТА №3 АЛГОРИТМЫ СЖАТИЯ

1 АЛГОРИТМЫ СЖАТИЯ ШЕННОНА-ФАНО И ХАФФМЕНА

Кодирование алгоритмом Шеннона-Фано производится следующим образом. Кодируемые знаки выписывают в таблицу в порядке убывания их вероятностей в сообщениях. Затем их разделяют на две группы так, чтобы значения сумм вероятностей в каждой группе были близкими. Все знаки одной из групп в соответствующем разряде кодируются, например, единицей, тогда знаки второй группы кодируются нулем. Каждую полученную в процессе деления группу подвергают вышеописанной операции до тех пор, пока в результате очередного деления в каждой группе не останется по одному знаку.

При использовании алгоритма Хаффмена, кодируемые знаки также располагают в порядке убывания их вероятностей. Далее на каждом этапе две последние позиции списка заменяются одной и ей приписывают вероятность, равную сумме вероятностей заменяемых позиций. После этого производится пересортировка списка по убыванию вероятностей, с сохранением информации о том, какие именно знаки объединялись на каждом этапе. Процесс продолжается до тех пор, пока не останется единственная позиция с вероятностью, равной 1.

После этого строится кодовое дерево. Корню дерева ставится в соответствие узел с вероятностью, равной 1. Далее каждому узлу приписываются два потомка с вероятностями, которые участвовали в формировании значения вероятности обрабатываемого узла. Так продолжают до достижения узлов, соответствующих вероятностям исходных знаков.

Процесс кодирования по кодовому дереву осуществляется следующим образом. Одной из ветвей, выходящей из каждого узла, например, с более высокой вероятностью, ставится в соответствие символ 1, а с меньшей – 0. Спуск от корня к нужному знаку дает код этого знака. Правило кодирования в случае равных вероятностей оговаривается особо.

Задача

Используя алгоритмы Шеннона-Фано и Хаффмена проверить эффективное кодирование ансамбля из восьми знаков с заданными вероятностями.

Для построенного эффективного кода определить среднюю длину кодовой комбинации.

Решение

$$Z = \begin{bmatrix} z_1 & z_2 & z_3 & z_4 & z_5 & z_6 & z_7 & z_8 \\ 0{,}26 & 0{,}24 & 0{,}15 & 0{,}12 & 0{,}09 & 0{,}07 & 0{,}05 & 0{,}02 \end{bmatrix}.$$

Выполним кодирование этого ансамбля по алгоритму Шеннона-Фано.

Продемонстрируем первые три шага алгоритма после сортировки ансамбля по убыванию вероятностей.

a	0,26	
b	0,24	
c	0,15	
d	0,12	
e	0,09	
f	0,07	
j	0,05	
h	0,02	

a	0,26	0
b	0,24	0
c	0,15	1
d	0,12	1
e	0,09	1
f	0,07	1
j	0,05	1
h	0,02	1

a	0,26	00
b	0,24	01
c	0,15	10
d	0,12	10
e	0,09	11
f	0,07	11
j	0,05	11
h	0,02	11

a	0,26	00
b	0,24	01
c	0,15	10
d	0,12	10
e	0,09	11
f	0,07	11
j	0,05	11
h	0,02	11

Выполняем эту процедуру до тех пор, пока в результате очередного деления в каждой группе не останется по одному знаку.

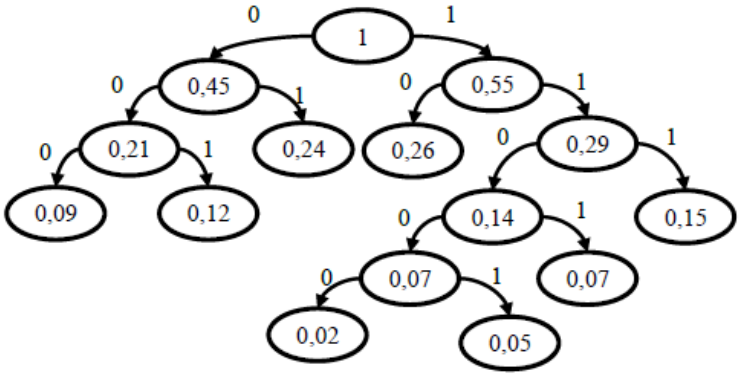
a	0,26	00
b	0,24	01
c	0,15	100
d	0,12	101
e	0,09	110
f	0,07	1110
j	0,05	11110
h	0,02	11111

Средняя длина кодовой комбинации получившегося кода:
n = 0,26 * 2 + 0,24 * 2 + 0,15 * 3 + 0,12 * 3 + 0,09 * 3 + 0,07 * 4 + 0,05 * 5 + 0,02 * 5 = 2,21

Выполним кодирование этого ансамбля по алгоритму Хаффмена. Заполним таблицу после сортировки ансамбля по убыванию вероятностей. Будем отмечать новое получившиеся значение вероятностей(сумму) на каждом шаге подчеркиванием и полужирным шрифтом.

a	0,26	0,26	0,26	**0,26**	**0,29**	**0,45**	**0,55**	**1**
b	0,24	0,24	0,24	0,24	0,26	0,29	0,45	
c	0,15	0,15	0,15	**0,21**	0,24	0,26		
d	0,12	0,12	**0,14**	0,15	0,21			
e	0,09	0,09	0,12	0,14				
f	0,07	0,07	0,09					
j	0,05	**0,07**						
h	0,02							

Построим дерево кодирования на основе полученной таблицы:



a	0,26	10
b	0,24	01
c	0,15	111
d	0,12	001
e	0,09	000
f	0,07	1101
j	0,05	11001
h	0,02	11000

Средняя длина кодовой комбинации получившегося кода:
n = 0,26 * 2 + 0,24 * 2 + 0,15 * 3 + 0,12 * 3 + 0,09 * 3 + 0,07 * 4 + 0,05 * 5 + 0,02 * 5 = 2,21.

Варианты заданий:

Вариант	a	b	c	d	e	f	j	h
1	0,26	0,24	0,15	0,12	0,09	0,07	0,05	0,02
2	0,26	0,19	0,15	0,13	0,12	0,09	0,06	0,05
3	0,19	0,21	0,13	0,12	0,15	0,06	0,05	0,09
4	0,17	0,23	0,15	0,14	0,1	0,09	0,07	0,05
5	0,26	0,19	0,14	0,11	0,1	0,08	0,07	0,05
6	0,22	0,26	0,16	0,12	0,09	0,07	0,05	0,03
7	0,02	0,18	0,14	0,12	0,12	0,1	0,08	0,06
8	0,28	0,22	0,15	0,11	0,11	0,07	0,04	0,02
9	0,22	0,18	0,15	0,13	0,13	0,09	0,05	0,05
10	0,25	0,19	0,15	0,11	0,09	0,09	0,07	0,05
11	0,17	0,24	0,14	0,16	0,09	0,08	0,07	0,05
12	0,24	0,23	0,16	0,14	0,06	0,08	0,06	0,03
13	0,21	0,17	0,14	0,13	0,12	0,1	0,07	0,06
14	0,21	0,18	0,16	0,12	0,12	0,12	0,05	0,05

2 АЛГОРИТМЫ СЖАТИЯ СЛОВАРНО ОРИЕНТИРОВАННЫЕ МЕТОДЫ КОДИРОВАНИЯ. МЕТОДЫ ЛЕМПЕЛЯ-ЗИВА.

Преимущество словарных алгоритмов состоит в том, что они позволяют кодировать последовательности символов разной длины. Первым был опубликован алгоритм LZ77(1977 год). После он был многократно модифицирован. Многие модификации получали название LZX, где X – первая буква имени автора модификации. Популярность алгоритмов LZ объясняется их простотой при высокой эффективности сжатия.

Ниже рассмотрим 2 алгоритма: LZ77, LZ78.

Алгоритм LZ77

Алгоритм LZ77 использует «скользящее» по сообщению окно, разделенное на две неравные части. Первая большая по размеру, включает уже просмотренную часть сообщения. Вторая, намного меньшая, является буфером, содержащим еще незакодированные символы входного потока. Алгоритм пытается найти в словаре(большей части окна) фрагмент, совпадающий с содержимым буфера.

Алгоритм LZ77 выдает коды, состоящие из трех элементов: (1)смещение в словаре относительно его начала строки, совпадающего с началом содержимого буфера; (2)длина этой строки; (3)первый символ буфера, следующий за совпадением.

Пример. Закодировать по алгоритму LZ77 строку «ЗЕЛЕНАЯ_ЗЕЛЕНЬ_ЗЕЛЕНЕЕТ». Размер буфера 7 символов, а словаря – 9 символов.

Решение.

Словарь (9)									Буфер (7)							Код
1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	
									З	Е	Л	Е	Н	А	Я	<0,0,З>
								З	Е	Л	Е	Н	А	Я	_	<0,0,Е>
							З	Е	Л	Е	Н	А	Я	_	З	<0,0,Л>
						З	**Е**	Л	**Е**	Н	А	Я	_	З	Е	<7,1,Н>
				З	Е	Л	Е	Н	А	Я	_	З	Е	Л	Е	<0,0,А>
			З	Е	Л	Е	Н	А	Я	_	З	Е	Л	Е	Н	<0,0,Я>
		З	Е	Л	Е	Н	А	Я	_	З	Е	Л	Е	Н	Ь	<0,0,_>
	З	**Е**	**Л**	**Е**	**Н**	А	Я	_	**З**	**Е**	**Л**	**Е**	**Н**	Ь	_	<1,5,Ь>
А	Я	_	З	**Е**	**Л**	**Е**	**Н**	Ь	_	З	**Е**	**Л**	**Е**	**Н**	Е	<2,6,Е>
Н	Ь	_	З	**Е**	Л	Е	Н	Е	**Е**	Т						<4,1,Т>

Длина кода вычисляется следующим образом: длина строки не может быть большего размера буфера + 1, а смещение не может быть больше размера словаря. Следовательно, длина двоичного кода смещения будет округленным в большую сторону log (размер словаря), а длина двоичного кода для длины строки будет округленным в большую сторону log (размер буфера + 1). А символ кодируется 8 битами (например,ASCII++).

Длина полученного кода: 10 · (log 9 + log (7+1) + 8) ≈ 10 · (4 +3 + 8),т.е.150 бит.

Длина исходной строки 23 символа, т.е. 23 ·8 = 184 бит.

Пример. Длина словаря 9 символов. Распаковать сообщение, сжатое LZ77:

<0,0,З>, <0,0,Е>, <0,0,Л>, <7,1,Н>, <0,0,А>, <0,0,Я>, <0,0,_>, <1,5,Ь>, <2,6,Е>, <4,1,Т>.

Решение.

Входной код	Печать	Словарь (9)								
		1	2	3	4	5	6	7	8	9
<0,0,З>	З									З
<0,0,Е>	Е								З	Е
<0,0,Л>	Л							З	**Е**	Л
<7,1,Н>	ЕН					З	Е	Л	Е	Н
<0,0,А>	А				З	Е	Л	Е	Н	А
<0,0,Я>	Я			З	Е	Л	Е	Н	А	Я
<0,0,_>	_		З	**Е**	**Л**	**Е**	**Н**	А	Я	_
<1,5,Ь>	ЗЕЛЕНЬ	А	Я	_	З	**Е**	**Л**	**Е**	**Н**	Ь
<2,6,Е>	_ЗЕЛЕНЕ	Н	Ь	_	З	**Е**	Л	Е	Н	Е
<4,1,Т>	ЕТ	_	З	Е	Л	Е	Н	Е	Е	Т

Алгоритм LZSS

В 1982 г. Сторером (Storer) и Шиманским (Szimanski) на базе LZ77 был разработан алгоритм LZSS, который отличается от LZ77 производимыми кодами.

Код, выдаваемый LZSS, начинается с однобитного префикса, различающего собственно код от незакодированного символа. Код состоит из пары: смещение и длина, такими же, как и для LZ77. В LZSS окно сдвигается ровно на длину найденной подстроки или на 1, если не найдено вхождение подстроки из буфера в словарь. Длина подстроки в LZSS всегда больше нуля, поэтому длина двоичного кода для длины подстроки – это округленный до большего целого двоичный логарифм от длины буфера.

Пример. Закодировать по алгоритму LZSS строку “ ЗЕЛЕНАЯ_ЗЕЛЕНЬ_ЗЕЛЕНЕЕТ ”.

СЛОВАРЬ(8)	БУФЕР(5)	КОД	ДЛИНА КОДА
“. . . . . .”	“КРАСН”	0’К’	9
“. . . . . К”	“РАСНА”	0’Р’	9
“. . . . .КР”	“АСНАЯ”	0’А’	9
“. . . .КРА”	“СНАЯ “	0’С’	9
“. . .КРАС”	“НАЯ К”	0’Н’	9

“. . .КРАСН”	“АЯ КР”	1<5,1>	7
“.КРАСНАЯ”	“ Я КРА”	0’Я’	9
“КРАСНАЯ ”	“ КРАС”	0’ ’	9
“АЯ КРАСК”	“КРАСК”	1<0,4>	7
“АЯ КРАСК”	“КА. . . ”	1<4,1>	7
“АЯ КРАСК”	“А. . . .”	1<0,1>	7

Здесь длина полученного кода равна $7 \cdot 9 + 4 \cdot 7 = 91$ бит.

LZ77 и LZSS обладают следующими очевидными недостатками:

1) невозможность кодирования подстрок отстоящих друг от друга на расстоянии, большем длины словаря;

2) Длина подстроки, которую можно закодировать, ограничена размером буфера.

Декодирование

3) LZSS, длина словаря — 8 байт (символов). Коды сжатого сообщения — 0,’К’ 0’Р’ 0’А’ 0’С’ 0’Н’ 1<5,1> 0’Я’ 0’ ’ 1<0,4> 1<4,1> 1<0,1>.

ВХОДНОЙ КОД	ПЕЧАТЬ	СЛОВАРЬ
0‘К’	"К"	".....К"
0‘Р’	"Р"	".....КР"
0‘А’	"А"	"....КРА"
0‘С’	"С"	"...КРАС"
0‘Н’	"Н"	"..КРАСН"
1 <5,1>	"А"	".КРАСНА"
0’Я’	"Я"	"КРАСНАЯ"
0’ ’	" "	"КРАСНАЯ "
1 <0,4>	"КРАС"	"НАЯ КРАС"
1 <4,1>	"К"	"АЯ КРАСК"
1 <0,1>	"А"	"Я КРАСКА"

Алгоритм LZ78

Алгоритм LZ77 не позволяет кодировать строки, отстоящие друг от друга на расстояние больше длины словаря, а длина строки, которую можно закодировать ограничена размером буфера. Если увеличивать размеры словаря и буфера, то это снизит эффективность кодирования.

LZ78 не использует «скользящее» окно, он хранит словарь из уже просмотренных фраз. При старте алгоритма этот словарь содержит одну пустую строку длины ноль. Алгоритм считывает символы сообщения до тех пор, пока накапливаемая строка входит целиком в одну из фраз словаря. Как только эта строка перестанет соответствовать хотя бы одной фразе словаря, которая до последнего введенного символа содержала входную строку, и символа, нарушившего совпадение. Затем в словарь добавляется введенная строка. Если словарь уже заполнен, то из него предварительно удаляют менее всех используемую в сравнении фразу.

Длина полученного двоичного кода будет округленным в большую сторону log(размер словаря) + 8(8 битами кодируются символы, например, ASCII+).

Пример. Закодировать по алгоритму LZ78 строку «ЗЕЛЕНАЯ_ЗЕЛЕНЬ_ЗЕЛЕНЕЕТ», используя словарь длинной 16 фраз.

Решение.

Входная фраза(в словарь)	Код	Позиция в словаре
«»		0
З	<0,З>	1
Е	<0,Е>	2
Л	<0,Л>	3
ЕН	<2,Н>	4
А	<0,А>	5
Я	<0,Я>	6
_	<0,_>	7
ЗЕ	<1,Е>	8
ЛЕ	<3,Е>	9
Н	<0,Н>	10
Ь	<0,Ь>	11
_З	<7,З>	12
ЕЛ	<2,Л>	13
ЕНЕ	<4,Е>	14
ЕТ	<2,Т>	15

Указать на любую фразу такого словаря это число от 0 до 15, для его кодирования достаточно log 16 = 4 бит.

Длина полученного кода равна: 15 * (4 + 8) = 180 бит.

Пример. Длина словаря 16 фраз. Распаковать сообщение, сжатое LZ78: <0,З>, <0,Е>, <0,Л>, <2,Н>, <0,А>, <0,Я>, <0,_>, <1,Е>, <3,Е>, <0,Н>, <0,Ь>, <7,З>, <2,Л>, <4,Е>, <2,Т>.

Решение.

Входной код	Печать(словарь)	Позиция словаря
«»		0
<0,З>	З	1
<0,Е>	Е	2
<0,Л>	Л	3
<2,Н>	ЕН	4
<0,А>	А	5
<0,Я>	Я	6
<0,_>	_	7
<1,Е>	ЗЕ	8
<3,Е>	ЛЕ	9

<0,Н>	Н	10
<0,Ь>	Т	11
<7,З>	_З	12
<2,Л>	ЕЛ	13
<4,Е>	ЕНЕ	14
<2,Т>	ЕТ	15

Алгоритм LZW

1. Инициализация словаря всеми возможными односимвольными фразами. Инициализация входной фразы ω первым символом сообщения.

2. Считать очередной символ К из кодируемого сообщения.

3. Если КОНЕЦ_СООБЩЕНИЯ, то выдать код для ω, закончить кодирование, иначе переход к 4.

4. Если фраза ωK уже есть в словаре, присвоить входной фразе значение ωК и перейти к шагу 2, иначе выдать код, соответствующий ω, добавить ωК в словарь, присвоить входной фразе значение К и перейти к шагу 2.

Задача.

Закодировать строку(см.варианты), составленную из символов заданного алфавита, при помощи алгоритм LZW.

Решение задачи.

Кодируется последовательность“abcdabceab”.

Таблица 6

Текущая строка(ω)	Новый символ(К)	Входной символ	Словарь
			0:a
			1:b
			2:c
			3:d
			4:e
a	b	0	5:ab
b	c	1	6:bc
c	d	2	7:cd
d	a	3	8:da
a	b	-	-
ab	c	5	9:abc
c	e	2	10:ce
e	a	4	11:ea
e	b	-	-
ab	-	5	-

Итоговый код: 0,1,2,3,5,2,4,5.

Задача.

Декодировать строку, составленную из символов заданного алфавита из кодовой последовательности полученной при помощи алгоритма LZW.

Код «0,1,2,3,5,2,4,5». Алфавит {“a”, “b”, “c”, “d”, “e”}.

Решение.

Декодируется сообщение «0,1,2,3,5,2,4,5».

Алфавит {“a”, “b”, “c”, “d”, “e”}.

Таблица 7

Входной символ	На выходе	Словарь	
		Полная запись	Частичная запись
		0:a	
		1:b	
		2:c	
		3:d	
		4:e	
0	a	-	5:a?
1	b	5:ab	6:b?
2	c	6:bc	7:c?
3	d	7:cd	8:d?
5	ab	8:da	9:ab?
2	c	9:abc	10:c?
4	e	10:ce	11:e?
5	ab	11:ea	-

Декодируемая строка: “abcdabceab”.

Варианты заданий для самостоятельного решения:

1) Строка “ababcdabc”. Алфавит {“a”, “b”, “c”, “d”}.
2) Строка “abcabdabc”. Алфавит {“a”, “b”, “c”, “d”}.