Задание 7

Тема. Кодирование и сжатие данных методами без потерь

Оглавление

Требуется выполнить два задания

Задание 1 Исследование алгоритмов сжатия на примерах

1) Выполнить каждую задачу варианта, представив алгоритм решения в виде таблицы и указав результат сжатия. Примеры оформления решения представлены в Приложении1 этого документа.
2) Описать процесс восстановления сжатого текста.
3) Сформировать отчет, включив задание, вариант задания, результаты выполнения задания варианта.

Задание 2 Разработать программы сжатия и восстановления текста методами Хаффмана и Шеннона – Фано.

1) Реализовать и отладить программы.
2) Сформировать отчет по разработке каждой программы в соответствии с требованиями.

   - По методу Шеннона-Фано привести: постановку задачи, описать алгоритм формирования префиксного дерева и алгоритм кодирования, декодирования, код и результаты тестирования. Рассчитать коэффициент сжатия. Сравнить с результат сжатия вашим алгоритмом с результатом людого архиватора.

   - по методу Хаффмана выполнить и отобразить результаты выполнения всех требований, предъявленных в задании и оформить разработку программы: постановка, подход к решению, код, результаты тестирования.

Варианты задания 1

Вариант	Закодировать фразу методами Шеннона–Фано	Сжатие данных по методу Лемпеля–Зива LZ77 Используя двухсимвольный алфавит (0, 1) закодировать следующую фразу:	Закодировать следующую фразу, используя код LZ78
1	Ана, дэус, рики, паки, Дормы кормы констунтаки, Дэус дэус канадэус – бац!	0001010010101001101	кукуркукурекурекун
2	One, two, Freddy's coming for you Three, four, better lock your door Five, six, grab a crucifix Seven, eight, gonna stay up late.	0100100010010000101	упупапекапекаупуп
3	Эне-бене, рики-таки, Буль-буль-буль, Караки-шмаки Эус-деус-краснодеус бац	0100101010010000101	лорлоралоранранлоран
4	Кони-кони, кони-кони, Мы сидели на балконе, Чай пили, воду пили, По-турецки говорили.	0100001000000100001	пропронепронепрнепронас
5	Прибавь к ослиной голове Еще одну, получишь две. Но сколько б ни было ослов, Они и двух не свяжут слов.	10100010010101000101 1	какатанекатанекатата

6	По-турецки говорили. Чяби, чяряби Чяряби, чяби-чяби. Мы набрали в рот воды.	000101110110100111	менменаменаменатеп
7	Тише, мыши, кот на крыше, А котята ещё выше. Кот пошёл за молоком, А котята кувырком.	11010101100110000100 1	долделдолдилделдил
8	Мой котёнок очень странный, Он не хочет есть сметану, К молоку не прикасался И от рыбки отказался.	01011011011010001000 1	sarsalsarsanlasanl 33
9	Эни-бени рити-Фати. Дорба, дорба сентибрати. Дэл. Дэл. Кошка. Дэл. Фати!	00010010110010001000 1	kloklonkolonklonkl
10	Самолёт-вертолёт! Посади меня в полёт! А в полёте пусто – Выросла капуста.	11101001101100011101	tertrektekertektrek
11	Кот пошёл за молоком, А котята кувырком. Кот пришёл без молока, А котята ха-ха-ха.	10101001101100111010	bigbonebigborebigbo
12	Цветом мой зайчишка – белый, А ещё, он очень смелый! Не боится он лисицы,	00010010101010011101	commercommecommerce

	Льва он тоже не боится.		
13	Эне, бене, лики, паки, Цуль, буль-буль, Калики-цваки, Эус-беус, клик-мадеус, бокс...	01011011001010101011	webwerbweberweberweb
14	Ана-дэус-рики-паки, Дормы-кормы-консту-таки, Энус-дэус-кана-дэус-БАЦ!	00101001100100000001	porpoterpoterporter
15	Раз, два – упала гора; три, четыре – прицепило; пять, шесть – бьют шерсть; семь, восемь – сено косим.	10110111100110001101	mantopmentopomantomen
16	Зуба зуба, зуба зуба, Зуба дони дони мэ, А шарли буба раз два три, А ми раз два три замри.	01001010100100000101	roporopoterropoterter
17	Плыл по морю чемодан, В чемодане был диван, На диване ехал слон. Кто не верит – выйди вон!	00010000101010011101	webwerbweberweberweb
18	Дрынцы-брынцы-бубен-цы, Раз-звонились-удальцы, Диги-диги-диги-дон,	11101001101110011101	sionsinossionsinos

	Выхо-ди-скорее-вон!		
19	Перводан, другодан, На колоде барабан; Свистель, коростель, Пятерка, шестерка, утюг.	0001000010101001101	comconcomconacom
20	Эни бэни рики паки Турбаурбасентибряки . Может – выйдет, может – нет, В общем – полный Интернет	0100101010010000101	mantopmentopomantomen

Требования к выполнению задания 2

1. Разработать алгоритм и реализовать программу сжатия текста алгоритмом Шеннона – Фано. Разработать алгоритм и программу восстановления сжатого текста. Выполнить тестирование программы на текстовом файле. Определить процент сжатия.
2. Провести кодирование(сжатие) исходной строки символов «Фамилия Имя Отчество» с использованием алгоритма Хаффмана. Исходная строка символов, таким образом, определяет индивидуальный вариант задания для каждого студента.

Для выполнения работы необходимо выполнить следующие действия:

2.1 Построить таблицу частот встречаемости символов в исходной строке символов для чего сформировать алфавит исходной строки и посчитать количество вхождений (частот) символов и их вероятности появления, например, для строки **пупкин василий кириллович** такая таблица будет иметь вид:

Таблица частот

Алфавит	п	у	к	и	н	« »	в
Кол. вх.	2	1	2	6	1	2	2
Вероятн.	0.08	0.04	0.08	0.24	0.04	0.08	0.08

Алфавит	а	с	л	й	р	о	ч
Кол. вх.	1	1	3	1	1	1	1
Вероятн.	0.04	0.04	0.12	0.04	0.04	0.04	0.04

(скобки < > обозначают пробел в исходной строке)

2.2 Отсортировать алфавит в порядке убывания частот появления символов по аналогии как показано ниже

Таблица отсортированных частот

Алфавит	и	л	п	к	« »	в	у
Кол. вх.	6	3	2	2	2	2	1
Вероятн.	0.24	0.12	0.08	0.08	0.08	0.08	0.04

Алфавит	н	а	с	й	р	о	ч
Кол. вх.	1	1	1	1	1	1	1
Вероятн.	0.04	0.04	0.04	0.04	0.04	0.04	0.04

2.3 Построить дерево кодирования Хаффмана, в данном примере оно имеет вид:

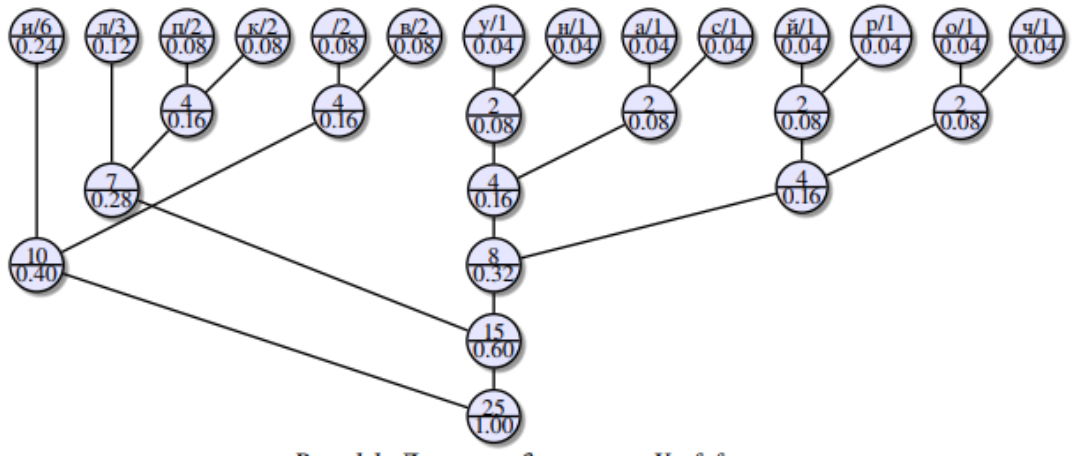
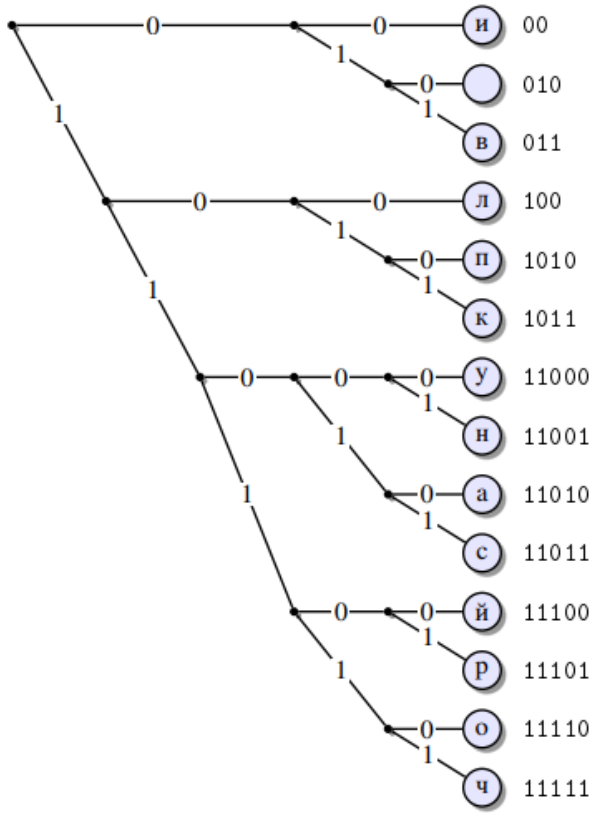


Рисунок Дерево кодирования Хаффмана

2.4 Упорядочить построенное дерево слева-направо (при необходимости).

2.5 Присвоить ветвям коды.

2.6 Определить коды символов:



2.7 Рисунок Упорядоченное дерево кодирования Хаффмана

2.8 Провести кодирование исходной строки по аналогии с примером:

```
 п     у     п     к     и     н    « »    в     а     с     и     л     и     й
1010 11000 1010 1011  00 11001  010  011 11010 11011  00  100  00  11100
« »    к     и     р     и     л     л     о     в     и     ч
010 1011  00 11101  00  100  100 11110  011  00 11111
```

2.9 Рассчитать коэффициенты сжатия относительно кодировки ASCII и относительно равномерного кода.

2.10 Рассчитать среднюю длину полученного кода и его дисперсию.

2.11 По результатам выполненной работы сделать выводы и сформировать отчет.

3. Применить алгоритм Хаффмана для архивации данных текстового файла. Выполнить практическую оценку сложности алгоритма Хаффмана. Провести архивацию этого же файла любым архиватором. Сравнить коэффициенты сжатия разработанного алгоритма и архиватора.

Приложение 1 Примеры оформления задач задания 1 и описание алгоритмов

1. Для метода Шеннон – Фано

Пример оформления таблицы. Закодировать фразу «Тише, мыши, тише, кот на крыше», используя метод Шеннона–Фено.

Таблица 1

Символ	Кол-во	1-я цифра	2-я цифра	3-я цифра	4-я цифра	5-я цифра	Код	Кол-во бит
пробел	5	0	0	0			000	15
ш	4	0	0	1			001	12
е	3	0	1	0			010	9
,	3	0	1	1			011	9
и	3	1	0	0			100	9
т	3	1	0	1	0		1010	12
ы	2	1	0	1	1		1011	8
к	2	1	1	1	0		1110	8
н	1	1	1	1	1		1111	4
о	1	1	1	0	0	0	11000	5
а	1	1	1	0	0	1	11001	5
м	1	1	1	0	1	0	11010	5
р	1	1	1	0	1	1	11011	5
								106

Незакодированная фраза – 30*8 бит = 240 бит.

Закодированная фраза –106 бит.

2. Для метода Лемпеля –Зива LZ77
   1) для сжатия двоичного кода

Исходный текст	00000001111111111110 00000000011011110
LZ-код	0.00.100.001.011.1011.1101. 1010.0110.10010.10001.10110.
R	2 3 4
Вводимые коды	– 10 11 100 101 110 111 1000 1001 1010 1011 1100

Где LZ – сжатый текст (в данном примере в связи с небольшим размером исходного текста размер текста не уменьшился)

R отмечает шаги кодирования, после которых происходит переход на представление кодов A увеличенным числом разрядов R. Так, на первом шаге вводится код 10 для комбинации 00, и поэтому на следующих двух шагах R =

2, после третьего шага R = 3, после седьмого шага R = 4, т.е. в общем случае R = K после шага 2^{K-1} –1.

1. Для метода Лемпеля –Зива LZ7

LZ77 использует скользящее по сообщению окно. Не использует словарь. Допустим, на текущей итерации окно зафиксировано. С правой стороны окна наращиваем подстроку, пока она есть в строке <скользящее окно + наращиваемая строка> и начинается в скользящем окне. Назовем наращиваемую строку буфером. После наращивания алгоритм выдает код <offset,len,char> состоящий из трех элементов:

- смещение в окне - *offset*;
- длина буфера - *len*;
- последний символ буфера - *char*.

В конце итерации алгоритм сдвигает окно на длину равную длине буфера+1.

Пример 1 cababababababm

Содержимое окна (сжимаемый текст)	Содержимое буфера	Код
cababababababm	*c*	<0,0,c>
cababababababm	*a*	<0,0,a>
cababababababm	*b*	<0,0,b>
cababababababm	*aba*	<2,2,a>
cababababababm	*bababz*	<2,5,m>

В рамочке представлено смещающееся окно, которое определяет буфер.

Результат сжатия

c(0,0)a(0,0)(2,2)b(0,0)m(2,5)

Описание алгоритма LZ78

В отличие от LZ77, работающего с уже полученными данными, LZ78 ориентируется на данные, которые только будут получены (LZ78 не использует скользящее окно, он хранит словарь из уже просмотренных фраз). Алгоритм считывает символы сообщения до тех пор, пока накапливаемая подстрока входит целиком в одну из фраз словаря. Как только эта строка перестанет соответствовать хотя бы одной фразе словаря, алгоритм генерирует код, состоящий из индекса строки в словаре, которая до последнего введенного символа содержала входную строку, и символа, нарушившего совпадение. Затем в словарь добавляется введенная подстрока. Если словарь уже заполнен, то из него предварительно удаляют менее всех

используемую в сравнениях фразу. Если в конце алгоритма мы не находим символ, нарушивший совпадения, то тогда мы выдаем код в виде (индекс строки в словаре без последнего символа, последний символ).

Пример abababaabb

Содержимое словаря	Содержимое считанной строки	Код
a	a	<0,a>
a, b	b	<0,b>
a, b, ab	ab	<1,b>
a, b, ab ,aa	aa	<1,a>
a, b, ab ,aa, abb	abb	<3,b>

Код 0a0b1b1a3b

Пример 2. cababababababm

Содержимое словаря	Содержимое считанной строки	Код
	c	<0,c>
c,	*a*	<0,a>
c,a	*b*	<0,b>
c, a, b	*ab*	<2,b>
c, a, b, ab	*aba*	<4,a>
c, a, b, ab, aba	*ba*	<3,a>
c, a, b, ab, aba, ba	*abab*	<5,b>
c, a, b, ab, aba, ba, abab	*m*	<0,m>

Результат сжатия: 0c0a0b2b4a3a5b0m

Описание алгоритма LZ78

Словарь хранится в префиксном дереве, что позволяет легко находить самое длинное продолжение входной строки, уже присутствующее в словаре. При декодировании строится в точности этот же словарь. В сжатом представлении строки словарь не хранится. В начале работы в словаре содержится

единственный элемент под номером ноль: T0 = ε; иными словами, префиксное дерево состоит из корня, помеченного номером 0. На каждом шаге читается самая длинная строка Tj = v, уже имеющаяся в словаре, и выводится её код j; также читается и выводится следующий символ a. При этом в словарь добавляется новая строка va — конкатенация только что прочитанной со следующим входным символом. На префиксном дереве это выглядит так: после вывода очередной пары (j, a) алгоритм переходит в корень префиксного дерева, и дальше читает столько входных символов, сколько возможно. Когда очередной символ прочитать нельзя, создаётся новый лист, при этом выводится номер предыдущей вершины и прочитанный символ. Пример 2. Строка w = abababaaabb, в таблице T0 = ε.

Кодирование: Читается a, выводится 0a, добавляется T1 = a.

Читается b, выводится 0b, добавляется T2 = b.

Читается ab, выводится 1b, добавляется T3 = ab.

Читается aa, выводится 1a, добавляется T4 = aa.

Читается abb, выводится 3b, добавляется T5 = abb.

Код — 0a0b1b1a3b.