Лабораторная работа № 7. Алгоритмы словарного сжатия информации

**Цель работы:** изучить основные алгоритмы словарного сжатия информации и приобрести практические навыки работы с ними.

1. Краткие теоретические сведения
   1. **Основные определения**

**Сжатие данных** – это процесс, обеспечивающий уменьшение объема данных путем сокращения их избыточности. Сжатие данных можно разделить на два основных типа:

* *Сжатие без потерь (полностью обратимое)* – это метод сжатия данных, при котором ранее закодированная порция данных восстанавливается после их распаковки полностью без внесения изменений. Для каждого типа данных, как правило, существуют свои оптимальные алгоритмы сжатия без потерь.
* *Сжатие с потерями* – это метод сжатия данных, при котором для обеспечения *максимальной степени* сжатия исходного массива данных часть содержащихся в нем данных отбрасывается.

Для текстовых, числовых и табличных данных использование программ, реализующих сжатие с потерями является неприемлемым. В основном такие алгоритмы применяются для сжатия аудио- и видеоданных, статических изображений.

**Алгоритм сжатия данных (алгоритм архивации)** – это *алгоритм*, который устраняет *избыточность* данных.

**Алфавит кода** – множество всех символов входного потока. При сжатии текстов обычно используют множество из 256 *ASCII* кодов. Это и есть алфавит кода.

**Кодовый символ** – наименьшая *единица* данных, подлежащая сжатию. В ASCII кодировке один символ – это 1 *байт*.

**Кодовое слово** – это последовательность кодовых символов из алфавита кода. Если все слова имеют одинаковую длину (число символов), то такой код называется *равномерным (фиксированной длины)*, а если же допускаются слова разной длины, то – *неравномерным (переменной длины)*.

**Код** – полное множество слов.

**Токен** – *единица* данных, записываемая в сжатый *поток* некоторым алгоритмом сжатия. *Токен* состоит из нескольких полей фиксированной или переменной длины.

**Фраза** – фрагмент данных, помещаемый в словарь для дальнейшего использования в сжатии.

**Кодирование** – процесс сжатия данных.

**Декодирование** – *обратный* кодированию процесс, при котором осуществляется восстановление данных.

**Коэффициент сжатия** – величина, характеризующая степень сжатия информации.

***Коэффициент сжатия = Объем исходной информации/Объем сжатой информации***

Значения коэффициента сжатия больше 1 обозначают сжатие, а значения меньше 1 – расширение.

Существуют две основных группы алгоритмов сжатия информации.

***Словарные алгоритмы сжатия*–** это алгоритмы сжатия, хранящие фрагменты данных в "словаре" (некоторая *структура данных*). Если строка новых данных, поступающих на вход, идентична какому-либо фрагменту, уже находящемуся в словаре, в выходной *поток*  помещается *указатель* на этот фрагмент.

***Статистические алгоритмы сжатия*** – алгоритмы сжатия, присваивающие коды переменной длины символам входного потока, причем более короткие коды присваиваются символам или группам символам, имеющим большую *вероятность* появления во входном потоке.

# **Алгоритм RLE (*run-length encoding)***

Алгоритм RLE является, наверное, самым простейшим из всех: суть его заключается в кодировании повторов. Если информация представляет собой последовательности повторяющихся кодов, то вместо такой последовательности кодов можно записывать пары «количество × значение». Например, строка вида «AAAAAAAABCCCC» может быть преобразована в запись вроде «8×A, 1×B, 4×C».

**Пример реализации**

Допустим, имеется набор неких целочисленных коэффициентов, которые могут принимать значения от 0 до 255. Логичным образом мы пришли к выводу, что разумно хранить этот набор в виде массива байт:

unsigned char data[] = {

0, 0, 0, 0, 0, 0, 4, 2, 0, 4, 4, 4, 4, 4, 4, 4,

80, 80, 80, 80, 0, 2, 2, 2, 2, 255, 255, 255, 255, 255, 0, 0

};

Всего 32 символа, которые будут занимать 32 байта

Для экономии места желательно как-то сжимать такие наборы. В наборе часто попадаются подпоследовательности, состоящие из одинаковых элементов.   
Эти данные могут быть закодированы следующим образом: 6×0, 1×4, 1×2, 1×0, 7×4, 4×80, 1×0, 4×2, 5×255, 2×0.

Для однообразия можно структурировать закодированные данные, указывая количество не только для повторяемых, но и последующих далее одиночных элементов.

В этом случае данные примут вид 6×0, 1x(4, 2, 0), 7×4, 4×80, 1x 0, 4×2, 5×255, 2×0.

В результате получены два вида последовательностей: цепочки одиночных элементов (вроде «4, 2, 0») и цепочки одинаковых элементов (вроде «0, 0, 0, 0, 0, 0»). Можно в «служебных» байтах выделить один бит под тип последовательности: 0 — одиночные элементы, 1 — одинаковые.   
В оставшихся 7 битах байта можно хранить длины последовательностей, т.е. максимальная длина кодируемой последовательности — 127 байт (127 символов).

В связи с тем, что последовательностей с нулевой длиной не существует, можно увеличить максимальную длину кодируемой последовательности до 128 байт, отнимая от длины единицу при кодировании и прибавляя при декодировании. Следовательно можно кодировать длины от 1 до 128 вместо длин от 0 до 127.

Кроме того, последовательностей одинаковых элементов единичной длины не бывает. Поэтому, от значения длины таких последовательностей при кодировании можно будет отнять ещё единицу, увеличив тем самым их максимальную длину до 129 (максимальная длина цепочки одиночных элементов по-прежнему равна 128). Т.е. цепочки одинаковых элементов могут иметь длину от 2 до 129.  
Закодируем наши данные снова, но теперь уже и в понятном для компьютера виде. Будем записывать служебные байты как [T|L], где T — тип последовательности, а L — длина. Будем сразу учитывать, что длины мы записываем в изменённом виде: при T=0 отнимаем от L единицу, при T=1 — двойку.

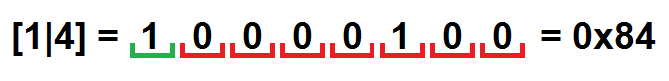
[1|4], 0, [0|2], 4, 2, 0, [1|5], 4, [1|2], 80, [0|0], 0, [1|2], 2, [1|3], 255, [1|0], 0

В результате получили 18 байт, значит коэффициент сжатия составит

**К=32/18=1,78**

Попробуем декодировать наш результат:

* [1|4]. T=1, значит следующий байт будет повторяться L+2 (4+2) раз: 0, 0, 0, 0, 0, 0.
* [0|2]. T=0, значит просто читаем L+1 (2+1) байт: 4, 2, 0.
* [1|5]. T=1, повторяем следующий байт 5+2 раз: 4, 4, 4, 4, 4, 4, 4.
* [1|2]. T=1, повторяем следующий байт 2+2 раз: 80, 80, 80, 80.
* [0|0]. T=0, читаем 0+1 байт: 0.
* [1|2]. T=1, повторяем байт 2+2 раз: 2, 2, 2, 2.
* [1|3]. T=1, повторяем байт 3+2 раз: 255, 255, 255, 255, 255.
* [1|0]. T=1, повторяем байт 0+2 раз: 0, 0.

А теперь последний шаг: сохраним полученный результат как массив байт. Например, пара [1|4], упакованная в байт, будет выглядеть вот так:  


В итоге получаем следующее:

0000: 84 00 02 04 02 00 85 04 82 80 00 00 82 02 83 FF 80 00

Таким вот нехитрым образом на данном примере входных данных мы из 32 байт получили 18. Неплохой результат, особенно если учесть, что на более длинных цепочках он может оказаться гораздо лучше.

#### **Возможные улучшения**

Эффективность алгоритма зависит не только от собственно алгоритма, но и от способа его реализации. Поэтому, для разных данных можно разрабатывать разные вариации кодирования и представления закодированных данных. Например, при кодировании изображений можно сделать цепочки переменной длины: выделить один бит под индикацию длинной цепочки, и если он выставлен в единицу, то хранить длину и в следующем байте тоже. Так мы жертвуем длиной коротких цепочек (65 элементов вместо 129), но зато даём возможность всего тремя байтами закодировать цепочки длиною до 16385 элементов (214 + 2)!

Дополнительная эффективность может быть достигнута путём использования эвристических методов кодирования. Например, закодируем нашим способом следующую цепочку: «ABBA». Мы получим «[0|0], A, [1|0], B, [0|0], A» — т.е. 4 байта мы превратили в 6, раздули исходные данные аж в полтора раза! И чем больше таких коротких чередующихся разнотипных последовательностей, тем больше избыточных данных. Если это учесть, то можно было бы закодировать результат как «[0|3], A, B, B, A» — мы бы потратили всего один лишний байт.

* 1. **Словарные алгоритмы сжатия**

Основным видом информации, которой обмениваются люди между собой является текстовая информация. Поэтому сжатие текста является актуальной задачей. Словарные методы сжатия предполагают использование словаря комбинаций символов. При сжатии данных вместо комбинаций символов в исходном тексте можно вставлять номер этой комбинации из словаря. При распаковке текста можно считывать номера комбинаций из сжатого текста, а затем вместо каждого номера вставлять комбинацию символов из словаря. Основным недостатком словарных методов сжатия является необходимость хранения и постоянного пополнения словаря, объем которого может существенно превышать размер исходного текста.

Вторым недостатком словарных методов является слишком большой размер номера комбинации символов в словаре. Если в качестве комбинаций символов использовать слова русского (или любого другого) языка, то максимальное значение номера слова в словаре будет равно числу слов в языке. По различным оценка число слов в русском языке приблизительно равно 30 000. С учетом различных изменений слов (склонения, спряжения, рода, времени) количество слов будет не менее 150 000. Значит количество бит, необходимое для хранения номера слова составит как минимум N=log2 150 000 ≈ 20. Значит каждое слово, независимо от количества букв в нем необходимо кодировать 20 битным (3 байтным) кодом. При таком кодировании коэффициент сжатия будет близок к единице.

Другим подходом к словарному сжатию является запись в словарь не всех слов языка, а комбинаций букв языка, не обязательно представляющих собой полное слово. В этом случае можно рассматривать ASCII таблицу, содержащую 256 символов как алфавит, а всевозможные комбинации символов этого алфавита будут определяться как размещения, количество которых определяется формулой

где m – количество символов в комбинации,

n – алфавит.

В таблице 1 приведено количество комбинаций символов различной длины для алфавита из 256 символов. 1012

|  |  |
| --- | --- |
| m | A |
| 1 | 256 |
| 2 | 256\*255=A1\*(A1-m+1) =65280 |
| 3 | 256\*255\*254=A2\*(A2-m+1)=16581120=1.7\*107 |
| 4 | 256\*255\*254\*253=A3\*(A3-m+1)=41950233=4.2\*109 |
| 5 | A4\*(A4-m+1)=1.06\*1012 |

Из таблицы 1 видно, что количество комбинаций символов огромно и для номера комбинации необходимо выделять количество байт, которое может быть значительно больше, чем количество байт для хранения этой последовательности в ASCII кодировке. Необходимо отметить, что бОльшую часть этих комбинаций представляют собой комбинации, никогда не встречающиеся в реальном языке (типа ьъ, жш, оь, еь и тп. в русском языке).

**Вывод.** Для использования словарного метода сжатия необходимо решить следующие проблемы.

1. Найти способ внесения в словарь только тех комбинаций символов, которые реально встречаются в сжимаемом тексте, что приведет к существенному уменьшению размера словаря.
2. Избавиться от необходимости хранения, передачи и постоянного пополнения словаря, а формировать словарь только для сжимаемого текста.

Приведенные проблемы решает группа алгоритмов Лемпеля-Зива (LZ)

* 1. **Алгоритм LZ77**

### Идея алгоритма

В тексте часто содержатся совпадающие длинные подстроки (комбинации символов). Идея, лежащая в основе LZ77, заключается в замене повторений на ссылки на позиции в тексте, где такие комбинации символов уже встречались.

LZ77 использует словарь, содержащий тройки данных – смещение, длина серии и символ расхождения. Смещение – как далеко от начала текста находится комбинация символов. Длина комбинации – сколько символов, считая от смещения, принадлежат комбинации. Символ расхождения показывает, что найдена новая фраза, похожая на ту, что обозначена смещением и длиной, за исключением этого символа. Словарь меняется по мере просмотра текста при помощи скользящего окна. К примеру, размер окна может быть 64Мб, тогда словарь будет содержать данные из последних 64 мегабайт входных данных.

**Создание словаря комбинаций символов**

* 1. Создается таблица, содержащая 3 строки: строка 1 – Комбинации символов, строка 2 – позиция комбинации, строка 3 - Код комбинации.
* 2. Из сжимаемого текста последовательно читаются символы и каждый считанный символ считается комбинацией и проверяется на предмет повторения комбинаций
* 3. Если комбинация не встречалась, и она содержит 1 символ, то она вносится в таблицу с кодом 0,0, «символ».
* 4. Если комбинация встречалась, то определяется ее смещение в количестве символов (байт) относительно текущего символа и к ней справа приписывается следующий символ входной последовательности и переход к п.4 иначе переход к п.5
* 5. Если комбинация не встречалась и она содержит более 1 символа, то она записывается в таблицу в виде «число1, число2, символ» , где число 1 – количество символов (байт) от текущего символа до символа в начале ранее встречавшейся комбинации, число 2 – количество символов в ранее встречавшейся комбинации, символ – последний символ в текущей комбинации.
* 6. Если следующего символа в сжимаемом тексте нет, то КОНЕЦ (словарь сформирован)

К примеру, для входных данных «расколотый\_колокол\_около\_колокольни\_расколот» (44 символа – 44 байта)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сим-вол | р | а | с | к | о | л | от | ы | й | \_ | коло | кол\_ |
| Код | 0,0, р | 0,0,а | 0,0,с | 0,0,к | 0,0,о | 0,0,л | 2,1,т | 0,0,ы | 0,0,й | 0,0,\_ | 7,4,о | 8,3,л |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сим-вол | около | \_колоколь | н | и | \_р | асколот |
| Код | 5,4,о | 14,7,ь | 0,0,н | 0,0,и | 11,1,р | 36,6, |

**Формирование сжатого текста**

Последовательная запись кодов (без запятых) из таблицы образует сжатый текст

Сжатый текст

**00р00а00с00к00о00л21т00ы00й00\_74о83л540147ь00н00и111р366**

Сжатый текст содержит 18 комбинаций по три символа (байта) – итого 54 байт.

Комбинации 00 можно заменить одним нулем – таких комбинаций 11

Размер сжатого текста составит 43 байта. Значит Ксж>1.

При увеличении размера текста появится больше комбинаций из3, 4 и более символов и коэффициент сжатия увеличится.

**Распаковка текста**

Последовательно читается сжатый текст. Если символу предшествует нулевой байт, то в распакованный текст вносится сам символ. Если символу предшествуют два числа M и N больше нуля, то в восстановленном тексте читается комбинация символов, начинающаяся на М символов левее текущего и содержащая N последовательных символов. Эта комбинация вставляется в текст.

* 1. **Алгоритм LZ78**

Суть алгоритма заключается в следующем

**Сжатие текста**

1. Создается словарь комбинаций символов для сжимаемого текста.
2. Комбинации символов в сжимаемом тексте заменяются их кодом (не номером) из словаря символов.

**Создание словаря комбинаций символов**

1. Создается таблица, содержащая 3 строки: строка 1 – Комбинации символов, строка 2 – Номер комбинации по порядку, строка 3 – Код комбинации в виде «число» «символ»

2. В таблицу вносится пустая комбинация символов и ей присваивается номер 0.

3. Из сжимаемого текста читается первый символ (пусть это буква А) и считается, что он является комбинацией, ему присваивается номер 1 и код 0A (для символа «A»)

4. Из сжимаемого текста читается следующий символ и проверяется существует ли такая комбинация в таблице.

5. Если комбинация не существует, то она записывается в таблицу, ей присваивается очередной номер и код аналогично п.3

6. Если комбинация существует, то код комбинации начинается с номера существующей комбинации, а из сжимаемого текста читается следующий символ и дописывается справа к прочитанной комбинации и переход к пункту 6.

7. Если следующего символа в сжимаемом тексте нет, то КОНЕЦ (словарь сформирован)

**Пример.** Пусть сжимаемым является текст «расколотый колокол около колокольни расколот» (44 символа). Значит объем составляет 44\*8=352 бит.

Таблица формирования словаря до внесения в нее данных имеет вид

|  |  |  |
| --- | --- | --- |
| Комбинация |  |  |
| Номер | 0 |  |
| Код |  |  |

После прочтения первого символа таблица формирования словаря примет вид

|  |  |  |
| --- | --- | --- |
| Комбинация |  | р |
| Номер | 0 | 1 |
| Код |  | 0р |

При прочтении второго символа комбинация имеет вид «а». Такой комбинации нет в таблице, поэтому в соответствии с п.5 алгоритма вносим эту комбинацию в таблицу с номером 1 и кодом 0а

Аналогично вносятся в таблицу комбинации «с», «к», «о», «л».

Таблица формирования словаря принимает следующий вид

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Комбинация |  | р | а | с | к | о | л |
| Номер | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Код |  | 0р | 0а | 0с | 0к | 0о | 0л |

При чтении символа «о» после буквы «л» определяем, что такая комбинация в таблице есть и она имеет номер 5. В соответствии с п.6 алгоритма читаем следующий символ из текста и получаем комбинацию «от» - такой комбинации нет, вносим ее в таблицу с кодом 5т. Цифра 5 – это предыдущая комбинация (в данном случае буква «о»). То есть код комбинации – это номер существующей в словаре комбинации и символ, который добавлен к ней.

Поступая аналогичным образом можно сформировать всю таблицу, которая будет иметь следующий вид (для наглядности пробел заменен символом подчеркивания \_)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Комбинация |  | р | а | с | к | о | л | от | ы | й | \_ | ко |
| Номер | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Код |  | 0р | 0а | 0с | 0к | 0о | 0л | 5т | 0ы | 0й | 0\_ | 4о |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Комбинация | ло | кол | \_о | коло | \_к | ол | око | ль | н | и | \_р | ас | колот |
| Номер | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| Код | 6о | 11л | 10о | 13о | 10к | 5о | 17о | 6ь | 0н | 0и | 10р | 2с | 15т |

Словарь сформирован.

**Создание сжатого текста**

Для сжатия теста можно просто последовательно записать коды из таблицы. Сжатый текст будет иметь вид

0р0а0с0к0о0л5т0ы0й0\_4о6о11л10о13о10к5о17о6ь0н0и10р2с15т

Сжатый текст содержит 55 символов, а исходный текст содержит 44 символов. Значит при ASCII кодировке сжатого текста коэффициент сжатия будет меньше 1.

Код каждой последовательности состоит из целого числа и следующего за ним символа. Максимальное значение числа 24. Следовательно числа можно закодировать 5-битным кодом (25=32). Значит каждая комбинация может быть закодирована 13 битным кодом.

Сжатый может быть представлен 312 битами. Коэффициент сжатия составляет 352/313=1,125

Из таблицы видно, что при продвижении к концу текста возникают многосимвольные комбинации – сначала двухсимвольные, затем трехсимвольные, а последней является пяти символьная комбинация. Все комбинации кодируются одинаковым количеством бит, значит чем длиннее комбинации и чем их больше тем больше коэффициент сжатия. Чем длиннее текст тем больше многосимвольных комбинаций и тем больше коэффициент сжатия. Но при этом необходимо также учитывать и тот факт, что с ростом числа комбинаций будет расти и число, стоящее в коле комбинации перед символом и для его представления понадобится больше бит.

**Распаковка текста**

Алгоритм распаковки сжатого текста можно представить в виде следующих операций

1. Последовательно читаются символы сжатого текста и по ним восстанавливается словарь
2. Коды комбинаций символов заменятся самими комбинациями из словаря

Пример.

Из сжатого текста

**0р0а0с0к0о0л5т0ы0й0\_4о6о11л10о13о10к5о17о6ь0н0и10р2с15т**

восстановим словарь.

Для этого нужно построить таблицу, содержащую 3 строки и в первый столбец этой таблицы записать пустую комбинацию с номером 0.

|  |  |  |
| --- | --- | --- |
| Комбинация |  |  |
| Номер | 0 |  |
| Код |  |  |

Зная, что закодированный текст содержит пары «Номер комбинации» «Символ», алгоритм восстановления словаря можно записать в следующем виде.

1. Прочитать очередной номер комбинации.
2. Если номер комбинации равен 0, то в строку «Комбинация» нужно записать просто символ из кода, а в строку номер записать очередной номер.
3. Если номер комбинации не равен нулю, то в строку «Комбинация» записать комбинацию символов с прочитанным номером и дописать к ней справа символ из кода, а в строку номер записать очередной номер.
4. Если кодов больше нет, то КОНЕЦ – словарь восстановлен.

Читаем первую пару в закодированном тексте - 0р. В соответствии с п.2 алгоритма в таблицу записываем комбинацию р и в строку Номер записываем 1. Аналогично поступаем для пар

0а, 0с, 0к, 0о, 0л. Таблица формирования словаря принимает сдедующий вид

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Комбинация |  | р | а | с | к | о | л |
| Номер | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Код |  | 0р | 0а | 0с | 0к | 0о | 0л |

Следующая комбинация 5т. В соответствии с п.4 алгоритма в строку «Комбинация» записываем комбинация с номером 5 – это символ «о» и добавляем к ней справа символ «т». Получаем комбинацию «от» и присваиваем ей номер 7. Действую аналогично можно восстановить всю таблицу формирования словаря.

**Восстановления исходного текста**

При последовательной записи всех комбинаций символов из строки «Комбинация» получаем исходный текст.

### Алгоритм LZW

**Алгори́тм Ле́мпеля — Зи́ва — Уэлча** (**Lempel-Ziv-Welch**, **LZW**) — это универсальный алгоритм сжатия данных без потерь, созданный Авраамом Лемпелем (англ. *Abraham Lempel*), Яаковом Зивом (англ. *Jacob Ziv*) и Терри Велчем (англ. *Terry Welch*). Он был опубликован Велчем в 1984 году в качестве улучшенной реализации алгоритма LZ78, опубликованного Лемпелем и Зивом в 1978 году.

**Сжатие текста**

1. Создается словарь комбинаций символов для сжимаемого текста.
2. Комбинации символов в сжимаемом тексте заменяются их **номером** из словаря символов.

**Создание словаря комбинаций символов**

1. Создается таблица, содержащая 4 строки: строка 1 – Номер комбинации символов, строка 2 – Сжимаемый текст, строка 3 – Код комбинации, строка 4 – Комбинация в словаре

2. В таблицу вносится пустая комбинация символов и ей присваивается номер 0.

3. В таблицу вносятся все символы алфавита текста и им присваиваются последовательные номера. (Могут быть внесены все символы таблицы ASCII с номерами от 1 до 255)

4. Из сжимаемого текста читается очередной символ и он считается текущей комбинацией.

5. Если текущая комбинация есть в словаре, то запоминается ее код (номер) и читается следующий символ из текста и приписывается справа к текущей комбинации. Если такой комбинации нет, то она записывается в словарь с очередным номером (в строке «Номер»), а в сжатый текст записывается запомненный код (номер) текущей комбинации. Если такая комбинация есть в словаре, то она становится текущей и переход к п.5.

Если больше нет символов, то номер последней комбинации помещается в сжатый текст и кодирование завершается.

**Пример**. Сжимается текст «расколотый\_колокол\_около\_колокольни\_расколот». Размер исходного текста 44 символа – 44 байта (352 бита). Вместо пробела используется символ подчеркивания \_.

Алфавит текста

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер символа | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Символ | а | и | й | к | л | н | о | р | с | т | ы | ь | \_ |

Алфавит текста содержит 12 символов, для его кодирования достаточно 4 бита (24=16)

Таблица формирования словаря до внесения в нее данных будет иметь вид

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Номер** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| **Текст** |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **р** | **а** | **с** | **к** | **о** | **л** | **о** |
| **Код** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 8 | 1 | 9 | 4 | 7 | 5 | 7 |
| **Словарь** |  | **а** | **и** | **й** | **к** | **л** | **н** | **о** | **р** | **с** | **т** | **ы** | **ь** | **\_** | **ра** | **ас** | **ск** | **ко** | **ол** | **ло** | **от** |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Номер** | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| **Текст** | **т** | **ы** | **й** | **\_** | **ко** | **ло** | **кол** | **\_** | **о** | **кол** | **о** |
| **Код** | 10 | 11 | 3 | 13 | 17 | 19 | 25 | 13 | 7 | 25 | 7 |
| **Словарь** | **ты** | **ый** | **й\_** | **\_к** | **кол** | **лок** | **кол\_** | **\_о** | **ок** | **коло** | **о\_** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Шаги** | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 |
| **Текст** | **\_к** | **ол** | **ок** | **ол** | **ь** | **н** | **и** | **\_** | **ра** | **ск** | **оло** | **т** |
| **Код** | 24 | 18 | 29 | 18 | 12 | 6 | 2 | 13 | 14 | 16 | 33 | 10 |
| **Словарь** | **коло** | **оло** | **око** | **оль** | **ьн** | **ни** | **и\_** | **\_р** | **рас** | **ско** | **олот** |  |

**Создание сжатого текста**

Сжатый текст может быть представлен в виде последовательности чисел из строки «Код»

8 1 9 4 7 5 7 10 11 3 13 17 19 25 13 7 25 7 24 18 29 18 12 6 2 13 14 16 33 10

Максимальное число в этом коде 30, значит при использовании 5-битового кода можно записать все числа. Всего 30 чисел, значит объем сжатого текста составляет 30\*5=150 бит. Коэффициент сжатия составляет Ксж=352/150=2,35. Однако, если используется собственный словарь символов (не ASCII), то необходимо его передавать вместе со сжатым текстом, что приведет к снижению коэффициента сжатия. При использование ASCII кодирования символы будут закодированы числами от 1 до 255, значит для кодов комбинаций потребуется более 8 бит на каждое число в сжатом тексте.

**Распаковка текста**

Алгоритм распаковки сжатого текста можно представить в виде следующих операций

1. Формируется таблица из 4 строк, аналогичная таблице кодирования
2. В строку код последовательно записываются коды из сжатого текста
3. Коды сжатого текста преобразуются в символы по словарю символов.
4. Как только прочитана первая пара символов формируется комбинация (ра) и записывается в словарь после символов алфавита (под номером 14) и определяется код этой комбинации.
5. При прочтении следующей буквы формируется следующая комбинация путем присоединения полученной буквы к предыдущей декодированной (ас) и определяется ее код.
6. Если очередное число в сжатом тексте не входит в перечень кодов алфавита (число 17), но комбинация с таким кодом уже есть в словаре, она может быть определена по коду (ко).
7. В словарь вносится комбинация, состоящая из предыдущей комбинации (ко) дополненной первой буквой следующей комбинации (ло). (Вносится комбинация «кол»).
8. Если символ является последним – словарь сформирован. Последовательность символов в

строке «Текст» является распакованным текстом.

**Пример.** Сжатый текст

8 1 9 4 7 5 7 10 11 3 13 17 19 25 13 7 25 7 24 18 29 18 12 6 2 14 16 33 10

Таблица кодов алфавита символов

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Номер** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| **Текст** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Код** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| **Словарь** |  | **а** | **и** | **й** | **к** | **л** | **н** | **о** | **р** | **с** | **т** | **ы** | **ь** | **\_** |

После внесения кодов алфавита в таблицу в строку «Код» последовательно вносятся коды комбинаций из сжатого текста. По этому коду определяется комбинация из предыдущей части таблицы и ей присваивается очередной номер, который записывается в строку «Номер». В строку «Словарь» записывается текущая комбинация, дополненная первым символом следующей комбинации.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Номер** | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| **Текст** | **р** | **а** | **с** | **к** | **о** | **л** | **о** | **т** | **ы** | **й** | **\_** | **ко** | **ло** | **кол** |
| **Код** | 8 | 1 | 9 | 4 | 7 | 5 | 7 | 10 | 11 | 3 | 13 | 17 | 19 | 25 |
| **Словарь** | **ра** | **ас** | **ск** | **ко** | **ол** | **ло** | **от** | **ты** | **ый** | **й\_** | **\_к** | **кол** | **лок** | **коло\_** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Номер** | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| **Текст** | **\_** | **о** | **кол** | **о** | **\_к** | **ол** | **ок** | **ол** | **ь** | **н** | **и** | **\_** | **ра** |
| **Код** | 13 | 7 | 25 | 7 | 24 | 18 | 29 | 18 | 12 | 6 | 2 | 13 | 14 |
| **Словарь** | **\_к** | **ок** | **коло** | **о\_** | **\_ко** | **оло** | **око** | **оль** | **ьн** | **ни** | **и\_** | **\_р** | **рас** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер** | 41 | 42 | 43 |
| **Текст** | **ск** | **оло** | **т** |
| **Код** | 16 | 33 | 10 |
| **Словарь** | **ско** | **олот** |  |

В строке «Текст» таблицы формируется распакованный текст.

### Выводы

1. Сжатие данных является процессом, обеспечивающим уменьшение объема данных путем сокращения их избыточности.
2. Сжатие данных может происходить с потерями и без потерь.
3. Коэффициент сжатия характеризует степень сжатия данных.
4. Существуют два основных группы алгоритмов сжатия: статистические алгоритмы и словарные алгоритмы.
5. Алгоритмы LZ77, LZ78 и LZW относится к словарным алгоритмам.
6. Порядок выполнения работы
7. Создать текстовый файл и записать в него свою фамилию, имя и отчество.
8. В соответствии с алгоритмами LZ77, LZ78 и LZW создать словарь комбинаций символов и закодировать текст. (Таблицы для сжатия и распаковки текста)
9. Оценить коэффициент сжатия текста для каждого алгоритма.

Содержание отчета о лабораторной работе

1. Титульный лист
2. Цель работы и задание для выполнения
3. Содержание текстового файла.
4. Таблицы для сжатия и распаковки текста для трех алгоритмов (LZ77, LZ78, LZW).
5. Текст, закодированный по каждому алгоритму и оценка коэффициента сжатия информации.
6. Выводы по лабораторной работе.

Контрольные вопросы

1.На чем основаны алгоритмы сжатия информации?

2. Как определяется коэффициент сжатия информации?

3. Как производится сжатие и распаковка текста в алгоритмах LZ77, LZ78, LZW?

4. Как производится сжатие информации в алгоритме RLE?

5. Можно ли присвоить всем словам номера и сжать информация, заменяя слова их номерами?