Словарные коды класса LZ

Словарные коды класса LZ широко используются в практических задачах. На их основе реализовано множество программ-архиваторов.

Словарные методы сжатия данных позволяют эффективно кодировать источники **с неизвестной или меняющейся статистикой.**

Важными свойствами этих методов являются высокая скорость кодирования и декодирования, а также относительно небольшая сложность реализации.

Кроме того, LZ-методы обладают способностью быстро адаптироваться к изменению статистической структуры сообщений.

Общая схема кодирования в LZ-методах

- При кодировании сообщение разбивается на слова переменной длины.
- При обработке очередного слова ведется поиск ему подобного в ранее закодированной части сообщения.

    + Если слово найдено, то передается соответствующий ему код.

    + Если слово не найдено, то передается специальный символ, обозначающий его отсутствие, и новое обозначение этого слова.

- Каждое новое слово, не встречавшееся ранее, запоминается, и ему присваивается индивидуальный код.

- При декодировании по принятому коду определяется закодированное слово.
- В случае получения специального символа, сигнализирующего о передаче нового слова, принятое слово запоминается, и ему присваивается такой же, как и при кодировании, код.

Таким образом, декодирование является **однозначным**, т.к. каждому слову соответствует свой собственный код.

По способу организации хранения и поиска слов словарные методы можно разделить на две большие группы:

- алгоритмы, осуществляющие поиск слов в какой-либо части ранее закодированного текста, называемой **окном**;
- алгоритмы, использующие адаптивный словарь, который включает ранее встретившиеся слова. Если словарь заполняется до окончания процесса кодирования, то в некоторых методах он обновляется (на место ранее встретившихся слов записываются новые), а в некоторых кодирование продолжается без обновления словаря.

Алгоритмы класса LZ отличаются:

- размерами окна;
- способами кодирования слов;
- алгоритмами обновления словаря и т.п.

Все указанные факторы влияют и на характеристики этих методов: скорость кодирования, объем требуемой памяти и степень сжатия данных, различные для разных алгоритмов.

Однако в целом методы из класса LZ представляют значительный практический интерес и позволяют достаточно эффективно сжимать данные с неизвестной статистикой.

Кодирование с использованием скользящего окна

Рассмотрим основные этапы кодирования сообщения $X=x_1x_2x_3x_4\ldots$, которое порождается некоторым источником информа

$$\ldots \underbrace{x_{i-W}\ldots x_{i-2}x_{i-1}}_{\text{окно}} x_i x_{i+1} x_{i+2} \ldots$$

Сначала осуществляется поиск в окне символа x_1. Если символ не найден, то в качестве кода передается 0 как признак того, что этого символа нет в окне и двоичное представление x_1.

Если символ x_1 найден, то осуществляется поиск в окне слова x_1x_2, *начинающегося с этого символа*.

Если слово x_1x_2 есть в окне, то производится поиск слова $x_1x_2x_3$, затем $x_1x_2x_3x_4$ и так далее, пока не будет найдено слово, состоящее из *наибольшего количества входных символов в порядке их поступления*.

В этом случае в качестве кода передается 1 и пара чисел (*i, j*), указывающая положение найденного слова в окне (*i* – номер позиции окна, с которой начинается это слово, *j* – длина этого слова, позиции в окне нумеруются справа налево).

Затем окно сдвигается на *j* символов вправо по тексту и кодирование продолжается.

Для кодирования чисел (*i, j*) могут быть использованы рассмотренные ранее коды целых чисел.

Пример. Пусть алфавит источника *A*={*a, b, c*}, длина окна *W*=6. Необходимо закодировать исходное сообщение *bababaabacabac.*

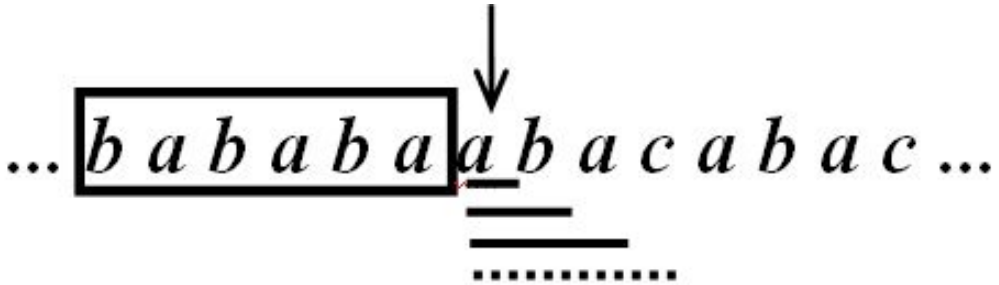
После кодирования первых шести букв окно будет иметь вид *bababa*.

- Далее проверяем наличие в окне буквы *a.* Она найдена, добавляем к ней *b*, ищем в окне *ab.* Эта пара есть в окне, добавляем букву *a*, ищем *aba.* Это слово есть в окне, добавляем букву *c*, ищем *abac*. Этого слова нет в окне, тогда кодируем *aba* кодовой комбинацией (1,3,3), где 1– признак того, что слово есть в «окне», 3– номер позиции в окне, с которой начинается это слово, 3– длина этого слова.

- Далее окно сдвигаем на 3 символа вправо и ищем в окне букву *c*. Ее нет в окне, поэтому кодируем комбинацией (0, «*c*»), где 0– признак того, что буквы нет в окне, «*c*»– двоичное представление буквы. Окно сдвигаем на 1 символ вправо.

- Ищем в окне букву *a*, она найдена, добавляем к ней *b*, ищем в окне *ab*. Эта пара есть в окне, добавляем букву *a*, ищем *aba*. Это слово есть в окне, добавляем букву *c*, ищем *abac*. Это слово есть в окне, тогда кодируем *abac* кодовой комбинацией (1,4,4), где 1 – признак того, что слово есть в окне, 4 –номер позиции в окне, с которой начинается это слово, 4 – длина этого слова.

Кодирование последовательности *bababaabacabac*



код (1,3,3)

код (0, «*c*»)

код (1,4,4)

Кодирование с использованием адаптивного словаря

В этих словарных методах для хранения слов используется адаптивный словарь, заполняющийся в процессе кодирования. Перед началом кодирования в словарь включаются все символы алфавита источника.

При кодировании сообщения $X=x_1x_2x_3x_4\ldots$ сначала читаются два символа x_1x_2, поскольку это слово отсутствует в словаре, то передается код символа x_1 (обычно это номер строки, содержащей найденный символ или слово).

В свободную строку таблицы записывается слово x_1x_2, далее читается символ x_3 и осуществляется поиск в словаре слова x_2x_3. Если это слово есть в словаре, то проверяется наличие слова $x_2x_3x_4$ и так далее. Таким образом, слово из наибольшего количества входных символов, найденное в словаре, кодируется как номер

Пример. Пусть алфавит источника $A=\{a, b, c\}$, размер словаря $V=8$. Необходимо закодировать исходное сообщение *abababaabacabac*.

1. Запишем символы алфавита A в словарь, каждому символу припишем кодовое слово длины $L = \lceil \log_2 V \rceil = \lceil \log_2 8 \rceil = 3$.

Номер строки	Слово	Код
0	*a*	000
1	*b*	001
2	*c*	010
3	–	
4	–	
5	–	
6	–	
7	–	

2. Читаем первые две буквы *ab*, ищем слово *ab* в словаре. Этого слова нет, поэтому поместим слово *ab* в свободную 3-ю строку словаря, а букву *a* закодируем кодом 000.

Номер строки	Слово	Код
0	*a*	000
1	*b*	001
2	*c*	010
3	*ab*	011
4	–	
5	–	
6	–	
7	–	

3. Далее читаем букву *a* и ищем в словаре слово *ba.* Этого слова нет, поэтому запишем в 4-ю строку словаря слово *ba,* букву *b* закодируем кодом 001.

Номер строки	Слово	Код
0	*a*	000
1	*b*	001
2	*c*	010
3	*ab*	011
4	*ba*	100
5	–	
6	–	
7	–	

4. Читаем букву *b*, ищем в словаре слово *ab*. Это слово есть в словаре в строке 3. Читаем следующую букву *a*, получим слово *aba*, его нет в словаре. Запишем слово *aba* в 5-ю строку словаря, и за

Номер строки	Слово	Код
0	*a*	000
1	*b*	001
2	*c*	010
3	*ab*	011
4	*ba*	100
5	*aba*	011
6	–	
7	–	

5. Читаем букву *b,* ищем в словаре слово *ab.* Это слово есть в словаре в строке 3. Читаем следующую букву *a*, получим слово *aba.* Это слово есть в словаре в строке 5. Читаем букву *a*, получим слово *abaa,* его нет в словаре. Запишем слово *abaa* в 6-ю строку словаря, и закодируем *aba* кодом 101.

Номер строки	Слово	Код
0	*a*	000
1	*b*	001
2	*c*	010
3	*ab*	011
4	*ba*	100
5	*aba*	011
6	*abaa*	101
7	–	

6. Читаем букву *b,* ищем в словаре слово *ab.* Это слово есть в словаре в строке 3. Читаем следующую букву *a*, получим слово *aba.* Это слово есть в словаре в строке 5. Читаем букву *c*, получим слово *abac,* его нет в словаре. Запишем слово *abac* в 7-ю строку словаря, и закодируем *aba* кодом 101.

Номер строки	Слово	Код
0	*a*	000
1	*b*	001
2	*c*	010
3	*ab*	011
4	*ba*	100
5	*aba*	101
6	*abaa*	110
7	*abac*	111

Если словарь заполняется до окончания кодирования, то можно записывать новые слова в словарь, начиная со строки с наибольшим номером, удаляя ранее записанные там слова.

7. Читаем букву *a,* ищем в словаре слово *ca.* Этого слова нет в словаре, поэтому запишем слово *ca* в 7-ю строку словаря, удалив слово *abac,* и закодируем букву *c* кодом 0

Номер строки	Слово	Код
0	*a*	000
1	*b*	001
2	*c*	010
3	*ab*	011
4	*ba*	100
5	*aba*	101
6	*abaa*	110
7	~~*abac*~~ *ca*	111

8. Читаем букву *b,* ищем в словаре слово *ab.* Это слово есть в словаре в строке 3. Читаем следующую букву *a*, получим слово *aba.* Это слово есть в словаре в строке 5. Читаем букву *c*, получим слово *abac,* его нет в словаре. Запишем слово *abac* в 6-ю строку словаря, и закодируем *aba* кодом 101.

Номер строки	Слово	Код
0	a	000
1	b	001
2	c	010
3	ab	011
4	ba	100
5	aba	101
6	~~abaa~~ abac	110
7	ca	111

9. Закодируем букву *c* кодом 010. Конец входной последовательности.

Таким образом, входное сообщение будет закодировано так:

Вход кодера:	*a*	*b*	*ab*	*aba*	*aba*	*c*	*aba*	*c*
Выход кодера:	000	001	011	101	101	010	101	010

Алгоритм на псевдокоде

Кодирование с адаптивным словарем

О**бозначим:**

CurCode – текущий код

PrevCode – предыдущий код

M – массив, содержащий текущую последовательность

L – длина текущей последовательности

C – словарь (массив строк)

S – текущая длина кода

DicPos – количество последовательностей в словаре

```
<Инициализация словаря символами исходного алфавита>
S:=9; L:=0; DicPos:=257                    (256+конец сжатия)
DO (not EOF)
    CurCode:=Read( )                    (читаем следующий байт из файла)
    M[L]:=CurCode;   L:=L+1
    IF (текущая последовательность найдена в словаре)
        CurCode:=номер найденной последовательности
    ELSE
        C[DicPos]:=M;  DicPos:=DicPos+1
        IF (log(DicPos)+1)>S  S:=S+1 FI (использовать соотношение п.6.1)
        Write(PrevCode,S)        (пишем в выходной файл S бит PrevCode)
        M[0]:=CurCode;  L:=1
    FI
    PrevCode:=CurCode
OD
Write(PrevCode,S)                          (сохраняем оставшийся код)
Write(256,S)                               (конец сжатия)
```

Рассмотрим теперь на примере ранее закодированного сообщения *abababaabacabac* (алфавит источника $A=\{a, b, c\}$, размер словаря $V=8$) процесс декодирования. Закодированная последовательность имела такой вид

000001011101101010101010

1. Запишем символы алфавита A в словарь, каждому символу припишем кодовое слово длины $L = \lceil \log_2 V \rceil = \lceil \log_2 8 \rceil = 3$. (Процесс заполнения словаря будет таким же, как и при кодировании.)

2. Читаем первые три бита кодовой последовательности (код 000), по коду найдем в словаре букву *a*.

3. Читаем следующий код 001, по коду найдем в словаре букву *b*. Получим новое слово *ab*, которого нет в словаре, поместим слово *ab* в свободную 3-ю строку словаря. На выход декодера передаем букву *a*, букву *b* запоминаем.

4. Читаем код 011, по коду находим в словаре слово *ab*. Добавляем первую букву *a* к предыдущему декодированному слову *b*, получим слово *ba*, его нет в словаре. Поместим слово *ba* в свободную 4-ю строку словаря. На выход декодера передаем букву *b*, слово *ab* запоминаем.

5. Читаем код 101, такого кода нет в словаре. Тогда добавляем к слову *ab* первую букву этой же последовательности – *a*, получим слово *aba*, его нет в словаре. Поместим слово *aba* в свободную 5-ю строку словаря. На выход декодера передаем слово *ab*, слово *aba* запоминаем.

6. Читаем код 101, по коду находим в словаре слово *aba*, добавляем первую букву *a* к предыдущему декодированному слову *aba*, получим *abaa*. Добавим слово *abaa* в словарь в свободную 6-ю строку. На выход декодера передаем слово *aba* , и слово *aba* запоминаем.

7. Читаем код 010, по коду находим в словаре букву *c*, добавляем букву *c* к предыдущему декодированному слову *aba*, получим *abac*. Добавим слово *abac* в словарь в свободную 7-ю строку. На выход декодера передаем слово *aba* , букву *c* запоминаем.

8. Читаем код 101, по коду находим в словаре слово *aba*, добавляем первую букву *a* к предыдущей декодированной букве *c*, получим слово *ca.*

Так как словарь заполнился до окончания декодирования, то будем записывать новые слова в словарь, начиная со строки с наибольшим номером, удаляя ранее записанные там слова. Добавим слово *ca* в 7-ю строку словаря вместо слова *abac*. На выход декодера передаем букву *c*, слово *aba* запоминаем.

9. Читаем код 010, по коду находим в словаре букву *c*, добавляем букву *c* к предыдущему декодированному слову *aba*, получим *abac*. Добавим слово *abac* в 6-ю строку словаря вместо слова *abaa*. На выход декодера передаем слово *aba,* букву *c* запоминаем.

10. На выход декодера передаем букву *c*.

В результате декодирования получим исходное сообщение

Вход декодера:	000	001	011	101	101	010	101	010
Выход декодера:	*a*	*b*	*ab*	*aba*	*aba*	*c*	*aba*	*c*

Алгоритм на псевдокоде

Декодирование с адаптивным словарем

Обозначим:

CurCode – текущий код

PrevCode – предыдущий код

M – массив, содержащий текущую последовательность

L – длина текущей последовательности

C – словарь (массив строк)

S – текущая длина кода

DicPos – количество последовательностей в словаре

<Инициализация словаря символами исходного алфавита>

S:=9; L:=0; DicPos:=257

DO

CurCode:=Read(S) (читаем из файла S бит)

IF (CurCode=256) break FI

IF (C[CurCode]<>0) (в словаре найдена послед-ть с номером CurCode)

M[L]:=C[CurCode][0] (в конец текущей последовательности приписываем первый символ найденной последовательности)

L:=L+1

ELSE M[L]:=M[0]; L:=L+1

FI

```
    IF (текущая последовательность M не найдена в словаре C)
        Write(C[PrevCode])
        C[DicPos]:=M    (добавляем текущую послед-ть в словарь)
        DicPos:=DicPos+1
        IF  (log DicPos+1)>S  S:=S+1 FI
        M:=C[CurCode]        (в текущую послед-ть заносим
        L=длина слова    слово сномером CurCode)
    FI
    PrevCode:=CurCode
OD
Write(C[PrevCode])
```