Сжатие с учётом контекста. Словарные методы с отдельным словарём (дерево/таблица) — семейство LZ78

Александра Игоревна Кононова

МИЭТ

4 марта 2023 г. — актуальную версию можно найти на `https://gitlab.com/illinc/otik`

Алфавит и сообщение

В норме для кодов семейства LZ78, как и для любого кода, алфавит — набор байтов, исходный текст — последовательность байтов.

Рассмотрим сообщение «Обороноспособность» (18 символов всего, 8 разных):

1. в 8-символьном алфавите из 3-битных байтов (в сообщении встречаются все 8 возможных символов);
2. в 16-символьном алфавите из 4-битных байтов (в сообщении встречается только часть из 16 возможных символов).

Код Зива–Лемпеля, LZ78/LZ2 (концепция)

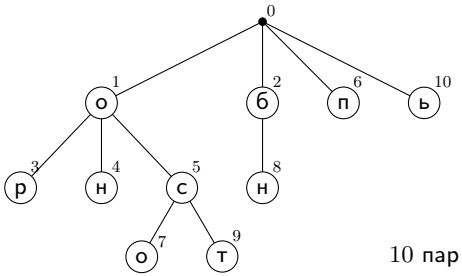
1978 г., Якоб Зив (Jacob Ziv) и Абрахам Лемпель (Abraham Lempel):

1. Скользящее окно не используем — кодируем в один проход вперёд $\Longrightarrow$ высокая скорость кодирования-декодирования.
2. Словарь = дерево, узел — номер и символ (n, c), корень — $(0,$ пустая строка$)$, слово читается от корня.
3. Вначале словарь пуст (только корень).
4. На каждом шаге
   - к словарю добавляется узел (лист);
   - в выходной поток — номер родителя и символ нового листа (P, c).
5. Когда кончается ёмкость номера листа, дерево:
   - либо уничтожается и растится заново;
   - либо ветви уничтожаются выборочно (сложно);
   - либо фиксируется и не растёт (нет прикорневого узла $\Longrightarrow$ сбои);
   - либо **увеличивается разрядность номера.**
6. При необходимости вх-й поток дополняется (либо конец обр-ся особо).

«Обороноспособность» (18, 8 разных)

1. Вначале словарь = корень (пустая строка),
   $n = 1$ (№ добавляемого узла, с 1), $i = 0$ (№ символа во вх. потоке, с 0).
2. $P = 0$ (текущий узел — корень), c_i (текущий символ входного потока);
3. Если c_i — дочерний P, $P = c_i$ и читаем c_{i+1} $(++i)$
4. Если c_i нет в дочерних узлах P:
   - добавляем P дочерний узел (n, c_i), $++n$ и читаем c_{i+1} $(++i)$;
   - в выходной поток пишем (P, c_i).

1	(0,о)	о
2	(0,б)	б
3	(1,р)	ор
4	(1,н)	он
5	(1,с)	ос
6	(0,п)	п
7	(5,о)	осо
8	(2,н)	бн
9	(5,т)	ост
10	(0,ь)	ь



10 пар

Код Зива–Лемпеля, LZ78/LZ2 (концепция) — Минимально возможная длина кода

В коде сообщения $n_{\max} = 10$ пар (P, c):

- разрядность $|c|$ символа постоянна и равна разрядности символа в исходном тексте;
- разрядность $|P|$ номера узла-родителя не равна $|c|$:
  - может быть постоянной: $|P| \geqslant \log_2(n_{\max} - 1)$, здесь $|P| \geqslant 4$ бита, обычно $|P| >> |c|$;
  - может быть разной для разных пар: **минимальная длина** кода достигается при побитовом увеличении $|P|$ (тогда поток пар (P, c) — битовый, а не байтовый).

Рассчитаем эту **минимальную длину** $|code|$.

№	Возможные P	$\min(\|P\|)$, бит
1	только 0	0 (не сохр.)
2	0 или 1	1
3	0, 1, 2	2
4	0, 1, 2, 3	2
5	0, 1, ... 4	3
6	0, 1, ... 5	3
7	0, 1, ... 6	3
8	0, 1, ... 7	3
9	0, 1, ... 8	4
10	0, 1, ... 9	4

Суммарная длина (в символах) полей c во всех 10 парах постоянна и равна 10 символов (байтов).
Минимальная суммарная длина (в битах) полей P во всех 10 парах: $|P|_{\sum} = 1 + 2 \cdot 2 + 3 \cdot 4 + 4 \cdot 2 = 25$ бит.

Минимальная общая длина $|code|$ (в символах=байтах) кода:
а) $|c| = 3$ (3-битный байт): $|code| = 10 + \frac{25}{3} = 10 + 8 + \frac{1}{3} \cong 19$;
б) $|c| = 4$ (4-битный байт): $|code| = 10 + \frac{25}{4} = 10 + 6 + \frac{1}{4} \cong 17$;
(для записи «лишнего» бита необходим целый байт).

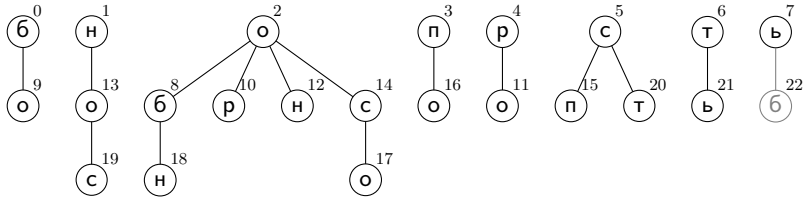
Длина исходного текста — 18 символов=байтов.

Код Зива–Лемпеля–Велча, LZW

1984 г., Терри Велч (Terry Welch) по концепции LZ78:

1. Вначале словарь = первый уровень (все одиночные символы, N штук). Тогда корень можно не нумеровать (прикорневые нумеруем с нуля).
2. При добавлении P дочернего узла (n, c_i):
   - оставляем c_i во входном потоке;
   - в выходной поток пишем (P).
3. При декодировании узла n:
   - в выходной поток пишем всю ветвь;
   - в дерево добавляем только прикорневой узел.
4. $|n| >> |c|$, во многих реализациях увеличивается по битам.
5. Дерево часто разворачивается в таблицу.
6. Вх-й поток всегда дополняется как минимум одним незначащим символом.

«Обороноспособность» (18, алфавит из 8)



8	(2, б)	об	2
9	(0, о)	бо	0
10	(2, р)	ор	2
11	(4, о)	ро	4
12	(2, н)	он	2
13	(1, о)	но	1
14	(2, с)	ос	2
15	(5, п)	сп	5

16	(3, о)	по	3
17	(14, о)	осо	14
18	(8, н)	обн	8
19	(13, с)	нос	13
20	(5, т)	ст	5
21	(6, ь)	ть	6
22	(7, б)	ьб	7
15 значений			

Код Зива–Лемпеля–Велча, LZW — Минимально возможная длина кода

В коде сообщения $n_{\max} - n_{\min} + 1 = 22 - 8 + 1 = 15$ значений P;
рассчитаем **минимально возможную** их длину:

- 1 размером 3 бита ($n = 8$: $P \in \{0, 1, ...7\}$);
- 8 размером 4 бита (от $n = 9$: $P \in \{0, 1, ...7, 8\}$ до $n = 16$: $P \in \{0, 1, ...15\}$);
- 6 размером 5 бит (от $n = 17$: $P \in \{0, 1, ...16\}$ до $n = 22$: $P \in \{0, 1, ...22\}$);
  потенциально 5 бит хватило бы на 16 значений (до $n = 32$: $P \in \{0, 1, ...31\}$),
  но сообщение закончилось раньше.

Суммарная длина (в битах) $3 + 4 \cdot 8 + 5 \cdot 6 = 65$ бит.

Символ=байт при таком дереве может занимать только 3 бита (алфавит из 8 символов).

Суммарная длина (в символах) $\frac{65}{3} = 21\frac{2}{3} \cong 22$ байта.

Декодирование (замечания)

1. При декодировании на i-м шаге (входной номер P_i) в выходной поток добавляется строка C_i, соответствующая узлу P_i (то есть на один символ короче, чем была на i-м шаге при кодировании),
   а в дереве словаря от самого P_i должен отрасти дочерний узел с номером i и неизвестным символом c_i.
2. Символ c_i узла i становится известным только на шаге $i+1$ (c_i — это первый символ подстроки C_{i+1} шага $i+1$),
   поэтому на практике узел i добавляется в словарь на шаге $i+1$.
3. Если на $i+1$ шаге получаем ссылку на ещё не добавленный узел $P_{i+1}=i$, то всё равно c_i — это первый символ подстроки C_{i+1},
   а первый символ C_{i+1} мы знаем даже при $P_{i+1}=i$ (как и все до предпоследнего включительно)!
   При $P_{i+1}=i$ последний символ строки C_{i+1} совпадает с первым: $C_{i+1}=axx\ldots xa$.

Декодирование

0	1	2	3	4	5	6	7
б	н	о	п	р	с	т	ь

2	о		
0	б	8	(2, б)
2	о	9	(0, о)
4	р	10	(2, р)
2	о	11	(4, о)
1	н	12	(2, н)
2	о	13	(0, о)
5	с	14	(2, с)

3	п	15	(5, с)
14	ос	16	(3, о)
8	об	17	(14, о)
13	но	18	(8, н)
5	с	19	(13, с)
6	т	20	(5, т)
7	ь	21	(6, ь)

Спасибо за внимание!

МИЭТ
www.miet.ru

Александра Игоревна Кононова
illinc@mail.ru
gitlab.com/illinc/raspisanie