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

Александра Игоревна Кононова

МИЭТ

26 января 2021 г. — актуальную версию можно найти на `https://gitlab.com/illinc/otik`

Код Зива–Лемпеля, LZ78/LZ2 (концепция)

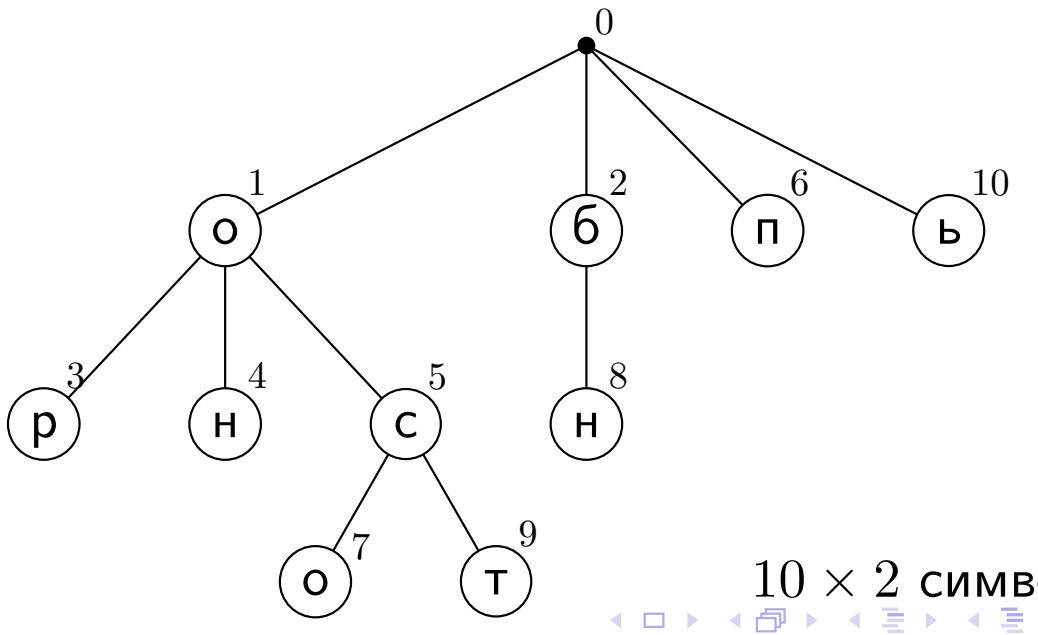
1978 г., Якоб Зив (Jacob Ziv) и Абрахам Лемпель (Abraham Lempel):

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), $n = 0$, $i = 0$.
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



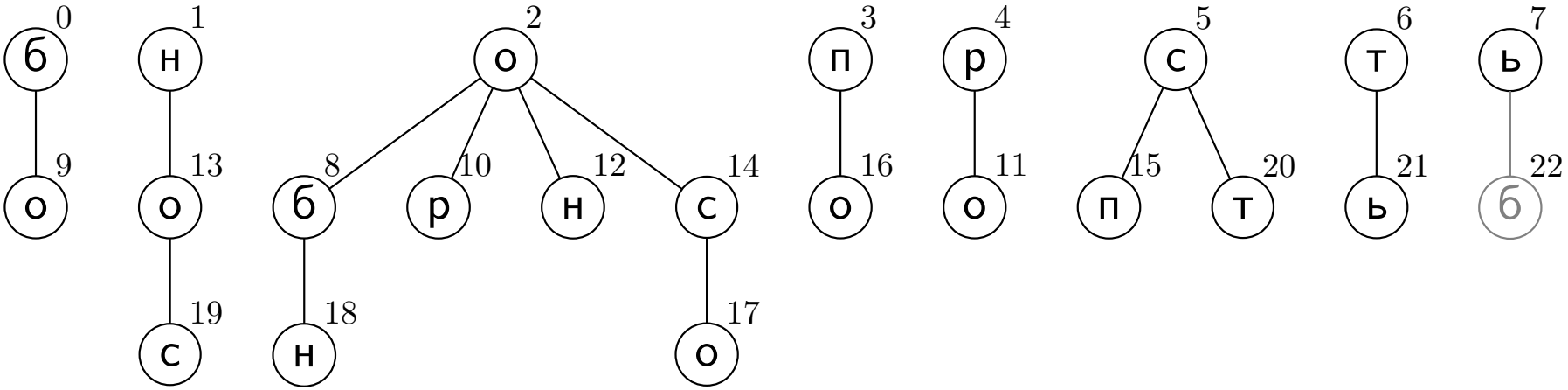
10×2 символов

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	символов		

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 $i+1$),

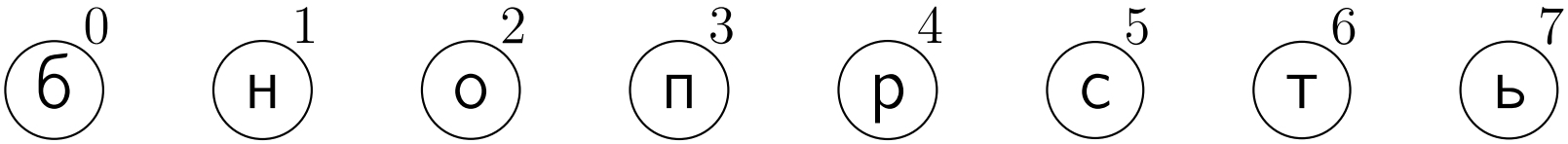
   поэтому на практике узел i добавляется в словарь на шаге $i+1$.

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



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
http://miet.ru/

Александра Игоревна Кононова
illinc@mail.ru