Словарные методы сжатия

Дискретный анализ 2012/13

Андрей Калинин, Татьяна Романова

25 мая 2013 г.

Литература

- Witten, Moffat, Bell, Managing gigabytes: compressing and indexing documents and images.
- M. J. Atallah, Algorithms and Theory of Computation Handbook, 12.5 LZW Coding.
- Ватолин Д., Ратушняк А., Смирнов М., Юкин В., Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео.

Раздел

Словарные методы

- LZ77
- LZW

Основная идея

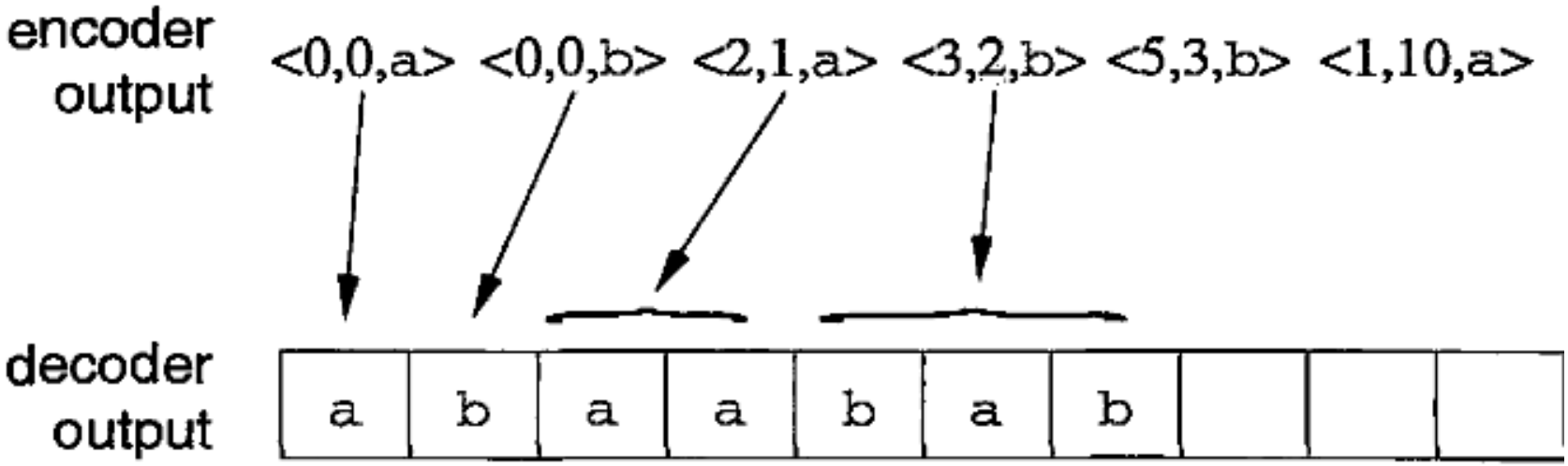
- Некоторые последовательности символов кодируются специальными кодами, хранящимся в словарях.
- Словарь может быть статическим или динамическим.
- Динамический словарь составляется из уже обработанных последовательностей символов (возможно, не всех).
- В качестве кода может выбираться смещение и длина последовательности в обработанном тексте.
- Непосредственное кодирование обычно выполняется простыми методами (коды фиксированной длины или коды Хаффмана).
- Основные алгоритмы разработны Зивом и Лемпелем в 70-х годах.

Раздел

Словарные методы

- LZ77
- LZW

Пример работы для $abaabab$



Алгоритм кодирования, длина окна W

1 $p \leftarrow 1$
2 **while** Ещё есть текст
3 Найти самое длинное вхождение $S[p \ldots]$ в $S[p - W \ldots p - 1]$
Допустим, есть вхождение в позиции m, длиной l
4 Вывести тройку $\langle p - m, l, S[p + l] \rangle$
5 $p \leftarrow p + l + 1$

Алгоритм декодирования, длина окна W

1 $p \leftarrow 1$
2 **while** $\langle f, l, c \rangle \leftarrow$ NEXT-TRIPLE()
3 $\quad S[p \ldots p + l - 1] \leftarrow S[p - f \ldots p - f + l - 1]$
4 $\quad S[p + l] \leftarrow c$
5 $\quad p \leftarrow p + l + 1$

Особенности

- Чрезвычайно простой алгоритм декодирования — позволяет включать вместе с архивом и программу по его распаковки (не сильно увеличивает размер архива).
- Предполагается что всегда выполняется поиск самого длинного вхождения на каждом шаге — может быть неэффективно по скорости и по степени сжатия.

Реализация с использованием суффиксных деревьев

- Откажемся от хранения троек — будем хранить пары (смещение, длина) или незакодированные символы.
- До начала сжатия строки S для неё строится суффиксное дерево $\mathbb{T}$.
- Каждая вершина $v \in \mathbb{T}$ нумеруется c_v — наименьший из позиций листьев, соответствующи v.
- Когда нужно вычислить $\langle f, l \rangle$, проходим по дереву $\mathbb{T}$ до точки p, где i равно строковой глубине точки p плюс c_v, где v — первая вершина в p или ниже.
- Получаем самый длинный преффикс $S[i \dots m]$, который встречается так же в $S[1 \dots i]$.
- Время на поиск $\langle s_i, l_i \rangle$ равно $O(l_i)$ и весь алгоритм сжатия работает за $O(m)$.

gzip

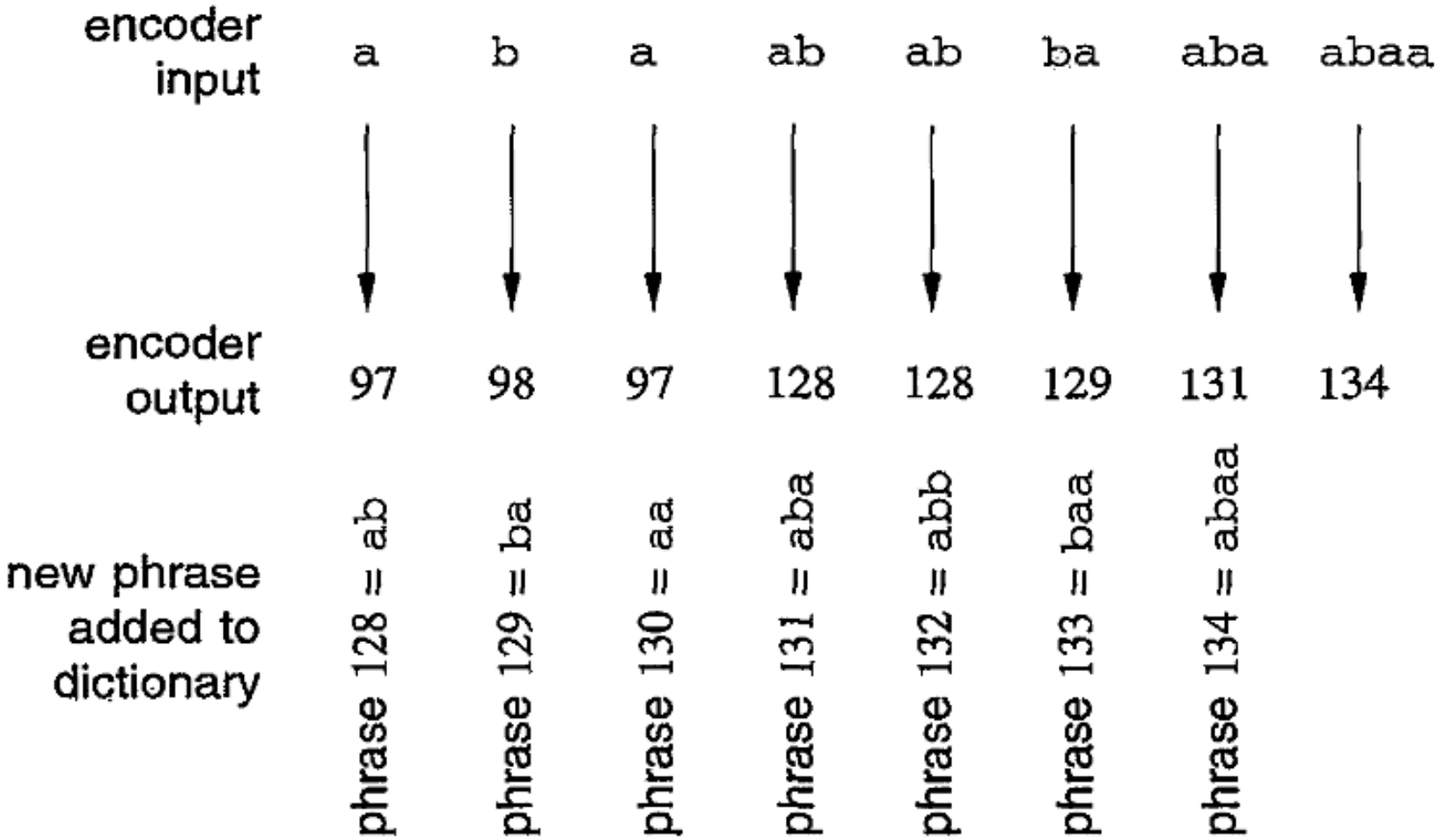
- Используется хеш-таблица для обнаружения предыдущих строк: хеш строится по первым трём символам, коллизии решаются списками. Для увелечения скорости списки ограничиваются в длине.
- Если есть вхождение, в выходной файл записывается его смещение и длина. В противном случае передаётся код символа.
- Смещение кодируется одним кодом Хаффмана, длина и символы — другим. Это позволяет уменьшить расходы на передачу длины и символа за ней.
- Коды Хаффмана строятся полустатично: входной файл делится на блоки по 64КБ, которые сжимаются и для которых рассчитываются коды Хаффмана.
- Есть возможность проверки того, что лучше: сжать всё, начиная с данного символа, или пропустить и сжимать следующую последовательность.

Раздел

Словарные методы

- LZ77
- LZW

Пример работы для *abaababababbaabaabaa*



Алгоритм сжатия

1 $p \leftarrow 1$
2 **for** $d \leftarrow 0$ **to** $q - 1$
3 $\quad D[d] \leftarrow \textsc{chr}(d)$
4 $d \leftarrow q - 1$
5 **while** есть ещё текст
6 $\quad \langle c, l \rangle \leftarrow \textsc{Longest-Match-For}(S[p \ldots], D)$
7 $\quad \textsc{Output}(c)$
8 $\quad d \leftarrow d + 1,\ p \leftarrow p + l$
9 $\quad D[d] \leftarrow "D[c]S[p]"$

Декомпрессия

```
1   p ← 1
2   for d ← 0 to q − 1
3         D[d] ← CHR(d)
4   d ← q − 1
5   while c ← NEXT-PHRASE-NUM()
6         if d ≠ q − 1
7               D[d][|D[d]|] ← D[c][0]
8         OUTPUT(D[c])
9         d ← d + 1
10        D[d] ← "D[c]?"
```