Преобразование Барроуза-Уилера, массив суффиксов и сжатие словарей

Д.В.Хмелёв

18 ноября 2003

Настоящая статья победила в конкурсе статей о сжатии 01.08.2003-31.10.2003 на сервере `http://www.compression.ru/`.

Аннотация

В статье (а) строго обоснована обратимость преобразования Барроуза-Уилера (BWT), (б) разобрана связь BWT с суффиксными массивами, (в) выявлены условия при которых из BWT можно восстановить заодно и и суффиксный массив, (г) приведены базовые алгоритмы обращения BWT. Кроме того, на основе анализа BWT предложено новое преобразование, которое может быть полезно для сжатия словарей или данных словарного типа.

Содержание

1 Введение

Преобразование Барроуза-Уилера, изобретённое Д.Дж. Уилером в 1983 году и впервые опубликованное в совместной статье с Барроузом [3] применяется как для сжатия текстов без потерь, так и для сжатия дополнительной индексной информации к тексту. Тем не менее, до сих пор отсутствует математически ясное обоснование некоторых фактов о преобразовании, в частности, обратимости и возможности восстановить суффиксный массив. Настоящая статья заполняет этот пробел. Кроме того, здесь предложена новая модификация преобразования, которую можно использовать для улучшения сжатия данных вроде словарных или корпусных.

Преобразование Барроуза-Уилера будет называться BW-преобразованием или просто BWT от Burrows-Wheeler Transformation.

В разделе 2 доказана обратимость BW-преобразования. Раздел 3 освещает тесную связь между суффиксными массивами и BW-преобразованием, включая способ извлечения суффиксного массива из преобразования BW. Раздел 4 содержит базовые алгоритмы связанные с BW-преобразованием. Наконец, раздел 5 включает описание, обоснование и алгоритм нового метода для сжатия данных типа словаря. Предложенный метод можно также использовать для предварительной группировки данных с целью повышения сжатия.

Автор придерживается той точки зрения, что построение алгоритма должно выполняться только после кристально математически ясного понимания ситуации. До того судить о верности алгоритма можно лишь на интуитивном уровне, а интуиция может подвести. Как ни странно, теория BW-преобразования, находится в зачаточном состоянии в то время как багаж наработанных алгоритмов уже достаточно велик. Настоящая работа исправляет этот крен и является чисто теоретической.

2 Преобразование BW и его обратимость

Пусть текст T состоит из $1 + N$ букв, занумерованных с нуля: $T[0..N]$. Буквы $T[i]$ принадлежат некоторому упорядоченному алфавиту $\mathcal{A}$. Лексикографический порядок (строгий) на строках из букв алфавита $\mathcal{A}$ будем обозначать $\preceq$ ($\prec$). Обозначим через $S_k T$ циклический сдвиг текста

T на k символов влево:

$$(S_k T)[j] = T[(j+k) \pmod{n+1}].$$

Существует перестановка σ чисел $\{0, \ldots, N\}$, которая удовлетворяет условию

$$S_{\sigma(i)} T \preceq S_{\sigma(i+1)} T,\ i = 0, \ldots, N-1. \tag{1}$$

Преобразование Барроуза-Уилера текста T есть текст $B[0..N] = BW(T)$, буквы которого заданы соотношением

$$B_i = (S_{\sigma(i)} T)[N]. \tag{2}$$

(другими словами, $B_i = (S_{\sigma(i)-1} T)[0] = T[\sigma(i) - 1 \pmod{N+1}]$).

Теорема 1 (Барроуз-Уилер). *Для восстановления исходного текста T из преобразования B достаточно знать число I, отвечающее условию $\sigma(I) = 0$ (или $S_{\sigma(I)} T = T$).*

Далее изложено формальное доказательство теоремы 1, основанное на изучении перестановки δ чисел $\{0, \ldots, N\}$, удовлетворяющей условию:

$$B_{\delta(i)} \preceq B_{\delta(i+1)} \text{ при } i = 0, \ldots, N-1, \tag{3}$$

и в случае равенства $B_{\delta(i)} = B_{\delta(i+1)}$ выполнено $\delta(i) < \delta(i+1)$. Перестановка δ однозначно определяется текстом B и её можно подсчитать за время $O(N)$ с помощью сортировки подсчётом (см. далее алгоритм 1 в разделе 4).

Перестановку δ можно рассматривать в качестве отображения $\delta : \{0, \ldots, N\} \to \{0, \ldots, N\}$. Итерацию k отображения δ обозначим через $\delta^k = \delta \circ \delta^{k-1}$, причём $\delta^0(i) \equiv i$, $\delta^1(i) = \delta(i)$.

Теорема 2. *При всех $m = 1, \ldots, N+1$ верны утверждения*

$$B_{\delta(i)} \ldots B_{\delta^m(i)} \preceq B_{\delta(i+1)} \ldots B_{\delta^m(i+1)} \text{ при } i = 0, \ldots, N-1, \tag{4}$$

и

$$B_i B_{\delta(i)} \ldots B_{\delta^{m-1}(i)} = (S_{\sigma(i)-1} T)[0..m-1] \text{ при } i = 0, \ldots, N. \tag{5}$$

Обычно обратимость BW-преобразования поясняют при помощи составления "концептуальной" таблицы $\mathcal{M}$ размерности $(N+1)\times(N+1)$, последний столбец которой составлен из букв B_i:

$$\mathcal{M} = \begin{array}{cccccc} * & * & * & \dots & B_0 \\ * & * & * & \dots & B_1 \\ * & * & * & \dots & B_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ * & * & * & \dots & B_N \end{array}$$

Если лексикографически отсортировать буквы последнего столбца и поместить их в первый столбец, то получается таблица

$$\begin{array}{cccccc} B_{\delta(0)} & * & * & \dots & B_0 \\ B_{\delta(1)} & * & * & \dots & B_1 \\ B_{\delta(2)} & * & * & \dots & B_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{\delta(N)} & * & * & \dots & B_N \end{array}$$

в которой буквы первого столбца $B_{\delta(i)}$ совпадают с $T[\sigma_i]$ в силу лексикографического порядка. Теперь, вспомнив, что пары $B_iB_{\delta(i)}$ суть префиксы всех циклических перестановок исходного текста, можно отсортировать эти пары и получить второй столбец матрицы, который даст тройки букв и т.д.

Таким образом, конечно, можно восстановить всю матрицу. Зная номер I строки, отвечающей исходному тексту T, легко найти сам текст T, поскольку $\sigma(I)=0$ и $S_{\sigma(I)}T=T$.

Обычно после такого объяснения, используя аналогии различной степени неточности, без строгого доказательства предлагается метод для восстановления T, который заключается в том, $T[i]=B_{\delta^i(I)}$, $i=0,\dots,N$. Автору не удалось обнаружить *ни одного* строгого доказательства правильности этого метода. Некое подобие обоснования дано в оригинальной работе Барроуза и Уилера [3], но они в действительности пользуются преобразованием δ^{-1} и обосновывают алгоритм только для этого случая.

Теорема 2 призвана внести ясность в обоснование BW-преобразования

и его обращения. Из неё вытекает, что действительно

$$\mathcal{M} = \begin{pmatrix} S_{\sigma_0}T \\ S_{\sigma_1}T \\ S_{\sigma_2}T \\ \vdots \\ S_{\sigma_N}T \end{pmatrix} = \begin{pmatrix} B_{\delta(0)} & B_{\delta^2(0)} & B_{\delta^3(0)} & \dots & B_{\delta^N(0)} & B_0 \\ B_{\delta(1)} & B_{\delta^2(1)} & B_{\delta^3(1)} & \dots & B_{\delta^N(1)} & B_1 \\ B_{\delta(2)} & B_{\delta^2(2)} & B_{\delta^3(2)} & \dots & B_{\delta^N(2)} & B_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ B_{\delta(N)} & B_{\delta^2(N)} & B_{\delta^3(N)} & \dots & B_{\delta^N(N)} & B_N \end{pmatrix}.$$

Доказательство теоремы 2 проводится индукцией по $m = 1, \dots, N+1$. База индукции $m = 1$. В силу определения (3) при всех $i = 0, \dots, N-1$ верно сравнение $B_{\delta(i)} \preceq B_{\delta(i+1)}$. В силу определения (2) выполнено $B_i = (S_{\sigma(i)-1}T)[0]$.

Пусть утверждения теоремы верны при $m-1$. Докажем их для m. В силу предположения индукции

$$B_{\delta(i)} \dots B_{\delta^{m-1}(i)} \preceq B_{\delta(i+1)} \dots B_{\delta^{m-1}(i+1)} \text{ при } i = 0, \dots, N-1 \qquad (6)$$

Кроме того,

$$B_i B_{\delta(i)} \dots B_{\delta^{m-2}(i)} = (S_{\sigma(i)-1}T)[0..m-2] \text{ при } i = 0, \dots, N \qquad (7)$$

Набор строк

$$(B_i B_{\delta(i)} \dots B_{\delta^{m-2}(i)},\ i = 0, \dots, N) \qquad (8)$$

совпадает с точностью до перестановки δ с набором строк

$$(B_{\delta(j)} B_{\delta^2(j)} \dots B_{\delta^{m-1}(j)},\ j = 0, \dots, N). \qquad (9)$$

В силу (6), набор (9) лексикографически упорядочен, а в силу (7) набор (8) есть набор префиксов длины $m-1$ всех циклических сдвигов исходного текста.

Однако, в силу определения (1) перестановки σ набор $(S_{\sigma(i)}T)[0..m-2]$ также является лексикографически упорядоченным набором префиксов длины $m-1$ всех циклических сдвигов текста T.

Поэтому

$$B_{\delta(j)} B_{\delta^2(j)} \dots B_{\delta^{m-1}(j)} = (S_{\sigma(j)}T)[0..m-2] \text{ при } j = 0, \dots, N \qquad (10)$$

С другой стороны, $B_j = (S_{\sigma(j)-1}T)[0]$. Поэтому

$$B_j B_{\delta(j)} B_{\delta^2(j)} \dots B_{\delta^{m-1}(j)} = (S_{\sigma(j)-1}T)[0..m-1] \text{ при } j = 0, \dots, N,$$

как и требуется в (5).

Остаётся доказать (4), то есть, что набор строк

$$(B_{\delta(j)}B_{\delta^2(j)}\dots B_{\delta^m(\delta(j))},\ j=0,\dots,N).$$

лексикографически упорядочен. Если

$$B_{\delta(j)}\dots B_{\delta^{m-1}(j)} \neq B_{\delta(j+1)}\dots B_{\delta^{m-1}(j+1)},$$

то, в силу (6)

$$B_{\delta(j)}\dots B_{\delta^{m-1}(j)} \prec B_{\delta(j+1)}\dots B_{\delta^{m-1}(j+1)},$$

откуда вытекает

$$B_{\delta(j)}\dots B_{\delta^{m-1}(j)}B_{\delta^m(j)} \prec B_{\delta(j+1)}\dots B_{\delta^{m-1}(j+1)}B_{\delta^m(j+1)}.$$

Поэтому предположим, что

$$B_{\delta(j)}\dots B_{\delta^{m-1}(j)} = B_{\delta(j+1)}\dots B_{\delta^{m-1}(j+1)}.$$

Это, в частности, означает, что $B_{\delta(j)} = B_{\delta(j+1)}$, откуда в силу определения (3) вытекает неравенство $\delta(j) < \delta(j+1)$. Обозначим $r = \delta(j)$, $s = \delta(j+1)$. Выражая вышесказанное через r и s, получим $B_r = B_s$ и $r < s$. В силу $B_r = B_s$, порядок между

$$B_rB_{\delta(r)}\dots B_{\delta^{m-1}(r)} = B_{\delta(j)}B_{\delta^2(j)}\dots B_{\delta^m(j)} \text{ и}$$
$$B_sB_{\delta(s)}\dots B_{\delta^{m-1}(s)} = B_{\delta(j+1)}B_{\delta^2(j+1)}\dots B_{\delta^m(j+1)}$$

совпадает с порядком между

$$B_{\delta(r)}\dots B_{\delta^{m-1}(r)} \text{ и } B_{\delta(s)}\dots B_{\delta^{m-1}(s)}$$

Напомним (см. (10)), что

$$B_{\delta(r)}\dots B_{\delta^{m-1}(r)} = (S_{\sigma(r)}T)[0..m-2] \text{ и } B_{\delta(s)}\dots B_{\delta^{m-1}(s)} = (S_{\sigma(s)}T)[0..m-2]$$

Из определения (1) и неравенства $r < s$, получается сравнение

$$(S_{\sigma(r)}T)[0..m-2] \preceq (S_{\sigma(s)}T)[0..m-2],$$

что даёт требуемый в (4) порядок. □

Воспользуемся теоремой 2 при $m = N + 1$. Тогда при всех $i = 0, \ldots, N$ выполнено

$$S_{\sigma(i)}T = B_{\delta(i)} \ldots B_{\delta^{N+1}(i)}$$

Подстановка $i = I$ с учётом $\sigma(I) = 0$ и $S_0T = T$ даёт

$$T = B_{\delta(I)} \ldots B_{\delta^{N+1}(I)}. \tag{11}$$

Доказательство теоремы 1. Поскольку перестановка δ однозначно определяется по B (см. (3)), получаем, что текст T действительно однозначно определяется текстом B и числом I по формуле (11). □

3 Суффиксный массив и BWT

Перестановку σ называют иногда *суффиксным массивом* или *массивом суффиксов*. Это не совсем точно: разные определения суффиксного массива пояснены в конце раздела. Суффиксные массивы играют важную роль в индексации информации. Например, с их помощью легко найти все повторяющиеся подстроки текста и много других важных характеристик текста (см., например, [2]).

Теорема 1 позволяет восстановить исходный текст T из преобразованного текста B и индекса I. Поскольку перестановка σ несёт важную информацию о тексте, возникает естественный вопрос: можно ли попутно восстановить перестановку σ? Оказывается, можно!

Наивный подход заключается в следующем. Известно, что $\sigma(I) = 0$. Естественно положить $\sigma(\delta(I)) = 1, \ldots, \sigma(\delta^N(I)) = N$. Проблема заключается в том, что набор чисел I, $\delta(I)$, $\ldots$, $\delta^N(I)$ может и не оказаться перестановкой чисел $0, \ldots, N$. Пример, когда это не так, был построен ещё Барроузом и Уилером [3].

Действительно, пусть $T[0..5]$ =”канкан”. Тогда матрица из лексикографически отсортированных строк выглядит так:

$$\begin{aligned}
S_1T &= \text{анканк},\\
S_4T &= \text{анканк},\\
S_0T &= \text{канкан},\\
S_3T &= \text{канкан},\\
S_2T &= \text{нканка},\\
S_5T &= \text{нканка},
\end{aligned}$$

то есть $\sigma = (1, 4, 0, 3, 2, 5)$, $B =$"ккннаа" и $I = 2$. Перестановка δ есть

$$\begin{aligned}
\delta(0) &= 4,\\
\delta(1) &= 5,\\
\delta(2) &= 0,\\
\delta(3) &= 1,\\
\delta(4) &= 2,\\
\delta(5) &= 3.
\end{aligned}$$

Очевидно, имеется цикл $0 \overset{\delta}{\mapsto} 4 \overset{\delta}{\mapsto} 2 \overset{\delta}{\mapsto} 0$, и наивный подход проваливается.

Из теоремы 2 вытекает следующая лемма:

Лемма 1. *Предположим, что*

$$\{I, \delta(I), \ldots, \delta^N(I)\} = \{0, \ldots, N\}. \tag{12}$$

Тогда

$$\sigma(\delta^i(I)) = i \text{ при } i = 0, \ldots, N. \tag{13}$$

Что означает условие (12)? Оно означает *неприводимость* перестановки δ.

Перестановка δ называется *неприводимой*, если при некотором j совпадают множества $\{j, \delta(j), \ldots, \delta^N(j)\} = \{0, \ldots, N\}$.

Лемма 2. *Пусть при некотором j выполнено равенство $\{j, \delta(j), \ldots, \delta^N(j)\} = \{0, \ldots, N\}$. Тогда*

1) $\delta^{N+1}(j) = j$;

2) При всех $i \in \{0, \ldots, N\}$ выполнено

$$\{i, \delta(i), \ldots, \delta^N(i)\} = \{0, \ldots, N\}.$$

3) При всех $i \in \{0, \ldots, N\}$ имеем $\delta^{N+1}(i) = i$.

Доказательство. Утверждения 2) и 3) легко вытекают из утверждения 1), которое и будет показано далее. В силу того, что δ — перестановка,

$$\delta\{j, \delta(j), \ldots, \delta^N(j)\} = \delta\{0, \ldots, N\} = \{0, \ldots, N\}.$$

С другой стороны,

$$\delta\{j, \delta(j), \ldots \delta^{N-1}(j)\} = \{\delta(j), \delta^2(j), \ldots \delta^N(j)\} = \{0, \ldots, N\} \setminus \{j\}.$$

Таким образом, единственным возможным значением для $\delta(\delta^N(j))$ является j. □

Не следует путать неприводимую перестановку с *циклической*. Перестановка δ называется *циклической*, если при некотором K выполнено $\delta(i) = i + K \pmod{N+1}$ для всех $i \in \{0, \ldots, N\}$. Циклическая перестановка является неприводимой тогда и только тогда, когда $N+1$ и K взаимно просты: НОД$(N+1, K) = 1$. Подкрепляющий пример: преобразование BW(T) определяется упорядочиванием *циклических* перестановок (сдвигов) текста S_jT. Для одновременного восстановления текста T *и* перестановки σ желательна (в силу леммы 1) *неприводимость* δ.

Лемма 3. *Предположим, что символ $T[N]$ отличен от всех остальных символов текста T: $T[N] \neq T[i]$, $i = 0, \ldots, N-1$. Тогда перестановка δ — неприводима.*

Доказательство. В силу теоремы 2

$$T = B_{\delta(I)} \ldots B_{\delta^{N+1}(I)}.$$

Поскольку символ $T[N]$ отличен от всех остальных символов, $\delta^i(I) \neq I$ при $i = 1, \ldots, N$. Если бы среди чисел $\delta^i(I)$ оказалось два одинаковых, например, $\delta^j(I) = \delta^k(I)$, $1 \leq j < k \leq N$, то $\delta^{j+s}(I) = \delta^{j+(s \pmod{k-j})}(I)$, а потому $\delta^{j+s}(I) \neq I$ при всех $s > 0$, что противоречит условию $\delta^{N+1}(I) = I$. Поэтому все числа $\delta^i(I)$, $i = 1, \ldots, N$ различны, а вместе с I они составляют множество чисел

$$\{I, \ldots, \delta^N(I)\} = \{0, \ldots, N\}.$$

Неприводимость перестановки δ следует из леммы 2. □

Лемма 4. *Предположим, что какой-нибудь символ $T[j]$ отличен от всех остальных символов текста T: $T[j] \neq T[i]$, $i \neq j$. Тогда перестановка δ — неприводимая.*

Доказательство. Рассмотрим циклический сдвиг $S_{j+1}T$, у которого

$$(S_{j+1}T)[N] = T[j].$$

Ясно, что $BW(T) = BW(S_{j+1}T)$. Обозначим $B = BW(T) = BW(S_{j+1}T)$. Значит, перестановка δ, что строится по $BW(T)$ и $BW(S_{j+1}T)$, одна и та же. Применяя лемму 3 к $S_{j+1}T$ получаем неприводимость δ. □

Лирическое отступление. Вышесказанное не означает, конечно же, что не существует алгоритма, восстанавливающего перестановку σ для неприводимых δ. Составление такого алгоритма — интересная задача, которая "закруглила" бы теорию преобразования Барроуза-Уилера. Автор настоящей статьи не относится к математикам, которые любят доказывать результаты в максимальной общности. Поэтому здесь дана лишь правдоподобная гипотеза о строении δ, которая верна для строки "канкан" (см. начало раздела).

Назовём перестановку δ *допустимой*, если она может возникнуть для какого-нибудь текста T: $\delta = \delta(\mathrm{BW}(T))$ при некотором $T[0..N]$.

Гипотеза 1. *Пусть $k > 0$, $n > 0$ — разложение числа $N+1$ на целые множители: $N+1 = kn$. Тогда допустима перестановка δ со свойствами:*

1) $\delta^k(0) = 0$,

2) множество $\{0, \delta(0), \ldots, \delta^{k-1}(0)\}$ содержит k разных чисел,

3) $\delta^j(i) = \delta^j(0) + i$ при $j = 0, \ldots, k-1$, и $i = 0, \ldots, n-1$.

Других допустимых перестановок δ нет.

Если $N+1$ — простое число, то из гипотезы вытекает, что возможны 2 варианта: либо δ — неприводима (случай $k \times n = (N+1) \times 1$), либо $\delta(i) = i$ при всех i (случай $k \times n = 1 \times (N+1)$). Например, первый случай возникает если буква $T[N]$ отличается от всех остальных. Второй — если текст $T[0..N]$ есть повторение одной буквы (например, $T[0..N] = \underbrace{a \ldots a}_{N+1 \text{ раз}}$).

Конец лирического отступления.

Обозначим $\$ = T[N]$ — последний символ T, отличный от всех остальных. Будем называть символ $\$ *разделителем.* Название обуславливается тем, что этот символ отделяет начало текста от конца. В разделе 5 будет использовано несколько разделителей, чтобы представить несколько текстов в одной строке. В принципе, возможны различные порядки $\$

относительно других букв алфавита $\mathcal{A}$. Обычно выбирают \$ либо лексикографически младшим, либо лексикографически старшим в $\mathcal{A}$. В первом случае $\sigma_0 = 0$, а во втором случае $\sigma_N = N$. *Суффиксным массивом* называют перестановку σ, из которой убран индекс, указывающий на \$.

С точки обозначений наиболее простым является соглашение о лексикографически старшем \$. Тогда перестановка $\sigma = (\sigma_0, \ldots, \sigma_{N-1}, N)$, а суффиксный массив — это набор $(\sigma_0, \ldots, \sigma_{N-1})$. Обычно это упрощает алгоритмы обработки строк, см., например, [2]

Таким образом, чтобы найти перестановку σ можно воспользоваться любым из многочисленных методов построения суффиксных массивов для текста T [4, 5, 6], а потом просто добавить суффикс, отвечающий символу $T[N]$. В приведённых статьях [4, 5, 6] обычно делается предположение, что $\$ \preceq \mathcal{A}$.

Для индекса I, отвечающего $\sigma(I) = 0$ выполнено $B_I = T[N] = \$$. Поэтому, чтобы избежать необходимости кодировать символ \$ достаточно передать текст $B' = B[0..I-1]B[I+1..N]$ и номер I. Очевидно, по этой информации легко восстановить

$$B = B'[0..I-1]\,\$\,B'[I..N-1].$$

Обладая этой информацией, по формулам (11) и (13) можно восстановить T и σ.

4 Алгоритмы восстановления T и σ по B и I

После прочтения предыдущих разделов не должно вызывать затруднений построение текста B (или текста B'). Остался незатронутым только вопрос о нахождении δ по тексту B.

Алгоритм 1 *(Нахождение δ).* Пусть $\mathcal{A} = \{a_1, \ldots, a_n\}$, причём $a_i \prec a_{i+1}$.

1 **foreach** $a \in \{a_1, \ldots, a_n\}$ **do** $C[a] := 0$; **od**
2 **foreach** $i \in \{0, \ldots, N\}$ **do** $C[T[i]] := C[T[i]] + 1$; **od**
3 теперь $C[a]$ содержит количество букв a в тексте T
4 $S := 0$;
5 **for** $i := 1$ **to** n **do**
6 $S := S + C[a_i]$;
7 $C[a_i] := S - C[a_i]$;
8 **od**

9 Теперь $C[a_i]$ указывает на первую позицию символа a_i в первом
10 столбце отсортированной матрицы циклических сдвигов T
11 **for** $i := 0$ **to** N **do**
12 $\delta(C[B_i]) := i$;
13 $C[B_i] := C[B_i] + 1$;
14 **od**

Нетрудно видеть, что алгоритм 1 есть всего навсего модификация сортировки подсчётом.

Пользуясь формулой

$$T = B_{\delta(I)} \dots B_{\delta^{N+1}(I)}$$

не представляет труда создать алгоритм, восстанавливающий текст T по BW-преобразованию B и индексу I.

Алгоритм 2 *(Восстановление T).*

1 $j := I$;
2 **for** $i := 0$ **to** N **do**
3 $j := \delta(j)$;
4 $T[i] := B_j$;
5 **od**

Если перестановка δ неприводима, то в силу леммы 1,

$$\sigma(\delta^i(I)) = i \text{ при } i = 0, \dots, N.$$

Поэтому алгоритм попутного восстановления перестановки σ выглядит следующим образом.

Алгоритм 3 *(Одновременное восстановление T и σ).* Предполагается, что перестановка δ неприводима.

1 $j := I$;
2 **for** $i := 0$ **to** N **do**
3 $\sigma(j) = i$;
4 $j := \delta(j)$;
5 $T[i] := B_j$;
6 **od**

Используя рассуждения, приведённые в конце раздела 3 легко модифицировать алгоритмы 1, 2, 3 для восстановления текста непосредственно из массива B' с учётом лексикографического положения разделителя \$ относительно других букв алфавита.

5 Преобразование данных словарного типа

5.1 Теория

Пусть текст $T[0..N]$ состоит из n частей, которые разделены специальными символами $\$_0, \ldots, \$_{n-1}$, нигде более в тексте T не встречающимися, причём $T[N] = \$_{n-1}$.

Будем считать, что позиции $i_0, \ldots, i_{n-1}$ разделителей упорядочены:

$$0 \le i_0 < \ldots < i_{n-1} = N, \text{ причём } T[i_k] = \$_k,\ k = 0, \ldots, n-1.$$

Другими словами,

$$T = T_0\$_0 \ldots T_{n-1}\$_{n-1}.$$

Кроме того, пусть разделители лексикографически предшествуют всем остальным буквам алфавита $\mathcal{A}$:

$$\$_k \prec a \text{ для всех } a \in \mathcal{A} \setminus \{\$_0, \ldots, \$_{n-1}\} \text{ и для всех } k = 0, \ldots, n-1.$$

Каковы свойства у $\mathrm{BW}(T)$? В силу того, что разделители предшествуют всем остальным символам алфавита, они будут стоять (некоторой перестановкой) в первом столбце концептуальной матрицы лексикографически отсортированных сдвигов T:

$$\mathcal{M} = \begin{array}{cccccc} \$_{\omega(0)} & * & * & * & \ldots & B_0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \$_{\omega(n-1)} & * & * & * & \ldots & B_{n-1} \\ B_{\delta(n)} & * & * & * & \ldots & B_n \\ B_{\delta(n+1)} & * & * & * & \ldots & B_{n+1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{\delta(N)} & * & * & * & \ldots & B_N \end{array} \tag{14}$$

где ω — перестановка, удовлетворяющая условию

$$\$_{\omega(0)} \prec \$_{\omega(1)} \prec \ldots \prec \$_{\omega(n-1)}. \tag{15}$$

Обозначим через I_k позицию разделителя $\$_k$ в тексте B:

$$B[I_k] = \$_k.$$

Раз уж $B_{\delta(j)} = \$_{\omega(j)}$,

$$\delta(j) = I_{\omega(j)}, \text{ при } j = 0, \ldots, n-1. \tag{16}$$

Пусть l_k — длина фрагмента T_k:

$$\begin{aligned} &l_0 = i_0, \\ &l_k = i_k - (i_{k-1} + 1), \text{ при } k = 1, \ldots, n-1. \end{aligned}$$

Определим также

$$m(k) = \min\{m \mid \delta^m(I_k) < n\}, \text{ и положим } m(-1) \equiv m(n-1). \tag{17}$$

Лемма 5. *При каждом $k = 0, \ldots, n-1$ верны утверждения*
1) $l_k = m(k-1)$.
2) $T_k = (S_{i_{k-1}}T)[1..l_k] = B_{\delta(I_{k-1})} \ldots B_{\delta^{m(k-1)}(I_{k-1})}$.
(здесь $i_{-1} \equiv i_{n-1}$, $I_{-1} \equiv I_{n-1}$).

Доказательство. В силу того, что разделитель $\$_{k-1}$ непосредственно предшествует строке T_k, из теоремы 2 вытекает, что

$$T_k = B_{\delta(I_{k-1})} \ldots B_{\delta^{l_k}(I_{k-1})}$$

Заметим, что $\delta(\delta^{l_k}(I_{k-1})) = \$_k$, а потому в силу структуры (14) концептуальной матрицы $0 \le \delta^{l_k}(I_{k-1}) < n$. Поскольку T_k не содержит разделителей, для всех $m = 1, \ldots, l_k - 1$ выполнено $\delta^m(I_{k-1}) \ge n$. Таким образом, $m(k-1) = l_k$, и оба заявленных утверждения верны. □

Рассмотрим теперь текст $\widehat{B}$, полученный из текста B с помощью замены всех символов-разделителей $\$_k$ на один символ-разделитель \$, который предшествует всем символам алфавита $\mathcal{A}$ исходного текста T:

$$\widehat{B}_i = \begin{cases} B_i, & \text{если } B_i \ne \$_k, k = 0, \ldots, n-1, \\ \$, & \text{если } B_i = \$_k \text{ при некотором } 0 \le k \le n-1. \end{cases} \tag{18}$$

По $\widehat{B}_i$ можно построить перестановку $\widehat{\delta}$, удовлетворяющую условию:

$$\widehat{B}_{\widehat{\delta}(i)} \preceq \widehat{B}_{\widehat{\delta}(i+1)} \text{ при } i = 0, \ldots, N-1, \tag{19}$$

и в случае равенства $\widehat{B}_{\delta(i)} = \widehat{B}_{\delta(i+1)}$ выполнено $\widehat{\delta}(i) < \widehat{\delta}(i+1)$. Алгоритм построения $\widehat{\delta}$ совпадает с алгоритмом 1 построения δ, только надо на вход подать $\widehat{B}$ вместо B.

Легко видеть, что условия (19) и (3) различаются лишь при $i = 0, \ldots, n-1$, причём для $i = 0, \ldots, n-1$ имеем $B_{\delta(i)} = \$_{\omega(i)}$ (см. также (16)) и $\widehat{B}_{\widehat{\delta}(i)} = \$$.

Поэтому справедлива следующая лемма.

Лемма 6. *$\delta(i) = \widehat{\delta}(i)$ при $i = n, \ldots, N$.*

Обозначим через $\widehat{I}_0 < \ldots < \widehat{I}_{n-1}$ позиции разделителя \$ в $\widehat{B}$:

$$\widehat{B}[\widehat{I}_k] = \$ \text{ при } k = 0, \ldots, n-1.$$

Из определения $\widehat{B}$ вытекает

Лемма 7. $\{\widehat{I}_0, \ldots, \widehat{I}_{n-1}\} = \{I_0, \ldots, I_{n-1}\}$. *Другими словами, набор чисел $(\widehat{I}_0, \ldots, \widehat{I}_{n-1})$ есть перестановка набора чисел $(I_0, \ldots, I_{n-1})$.*

Из определения (19) перестановки $\widehat{\delta}$ вытекает, что

$$\widehat{\delta}(k) = \widehat{I}_k \text{ при } k = 0, \ldots, n-1. \tag{20}$$

Пусть γ — это и есть перестановка чисел $\{0, \ldots, n-1\}$, переводящая набор $(\widehat{I}_0, \ldots, \widehat{I}_{n-1})$ в набор $(I_0, \ldots, I_{n-1})$:

$$\widehat{I}_{\gamma(k)} = I_k. \tag{21}$$

Кстати говоря, в этой перестановке известен последний элемент: $\gamma(n-1) = n-1$. По аналогии с (17) определим

$$\widehat{m}(k) = \min\{m \mid \widehat{\delta}^m(\widehat{I}_k) < n\}, \text{ и положим } \widehat{m}(-1) \equiv \widehat{m}(n-1). \tag{22}$$

Поскольку $\widehat{\delta}(i) = \delta(i)$ при $i = n, \ldots, N$, то условие $\widehat{\delta}^m(\widehat{I}_k) < n$ эквивалентно условию $\delta^m(\widehat{I}_k) < n$. Поэтому справедливо тождество

$$\widehat{m}(\gamma(k)) = m(k). \tag{23}$$

Для упрощения формулировок, положим $\widehat{I}_{-1} \equiv \widehat{I}_{n-1}$, $\gamma(-1) \equiv \gamma(n-1)$.

Теорема 3. *Обозначим*

$$\widehat{T}_j = \widehat{B}_{\widehat{\delta}(\widehat{I}_{j-1})} \ldots \widehat{B}_{\widehat{\delta}^{\widehat{m}(j-1)}(\widehat{I}_{j-1})} \text{ при } j = 0, \ldots, n-1. \tag{24}$$

Для всех $k = 0, \ldots, n-1$,

$$T_k = \widehat{T}_{\gamma(k-1)+1}. \tag{25}$$

Другими словами, наборы строк

$$(\widehat{T}_j \mid j = 0, \ldots, n-1) \text{ и } (\widehat{T}_k \mid k = 0, \ldots, n-1)$$

совпадают с точностью до перестановки, явная формула для которой в терминах γ *дана тождеством* (25).

Доказательство. Пользуясь определением текста $\widehat{B}$ и леммой 6, можно упростить выражение (24):

$$\widehat{T}_j = B_{\delta(\widehat{I}_{j-1})} \ldots B_{\delta^{\widehat{m}(j-1)}(\widehat{I}_{j-1})}.$$

Подставим теперь $j = \gamma(k-1)+1$. Тогда $j-1 = \gamma(k-1)$ и формула приобретает вид

$$\widehat{T}_{\gamma(k-1)+1} = B_{\delta(\widehat{I}_{\gamma(k-1)})} \ldots B_{\delta^{\widehat{m}(\gamma(k-1))}(\widehat{I}_{\gamma(k-1)})}.$$

что с учётом тождеств (23) и (21) даёт упростить выражение справа до

$$B_{\delta(I_{k-1})} \ldots B_{\delta^{m(k-1)}(I_{k-1})} = T_k$$

в силу утверждения 2) леммы 5. □

Теорема 4. *Имеет место тождество*

$$\delta(i) = \begin{cases} \widehat{\delta}(i) & i = n, \ldots, N \\ \widehat{\delta}(\gamma(\omega(i))) & i = 0, \ldots, n-1. \end{cases} \tag{26}$$

Доказательство. При $i \geq n$ тождество (26) доказано в лемме 6. В силу (20) и определения (21) выполнено тождество $\widehat{\delta}(\gamma(k)) = \widehat{I}_{\gamma(k)} = I_k$ при $k = 0, \ldots, n-1$. Если же подставить $k = \omega(i)$, $i = 0, \ldots, n-1$, то из (16) получаем

$$\widehat{\delta}(\gamma(\omega(i))) = I_{\omega(i)} = \delta(i).$$ □

Теорема 5. *Текст* T *(включая разделители с их порядком* ω*) и перестановка* σ *однозначно восстанавливаются по набору* $(\widehat{B}, I, \zeta)$*, где текст* $\widehat{B}$ *определён в* (18)*, целое число* I *определяется из условия* $\sigma(I) = 0$*, и* $\zeta = \gamma \circ \omega$.

Замечание 1. В силу леммы 3 перестановка δ неприводима (последний символ $\$_{n-1}$ отличен от всех остальных). Это согласуется с выводом теоремы 5, который справедлив лишь при неприводимой перестановке δ.

Доказательство. В силу (26) с помощью $\zeta = \gamma \circ \omega$ и $\widehat{\delta}$ однозначно восстанавливается перестановка δ. Обозначим $\widehat{T} = T_0\$ \dots T_{n-1}\$$. В силу (11), используя известное I и найденную δ, можно получить

$$\widehat{T} = \widehat{B}_{\delta(I)} \dots \widehat{B}_{\delta^{N+1}(I)}.$$

Зная $\widehat{T}$ легко восстановить номера разделителей \$. При известных номерах можно восстановить I_k, и перестановка ω восстанавливается из порядка $\delta^{-1}(I_k)$. Можно восстановить ω и с другого конца: зная I_k можно восстановить γ, откуда однозначно находится $\omega = \gamma^{-1} \circ \zeta$.

Наконец, поскольку δ — неприводима, в силу леммы 1 легко восстанавливается и перестановка σ. □

5.2 Применение для сжатия

Теперь покажем, как теорему 3 можно применять на практике. Предположим, что у нас имеестся n последовательностей символов T_0, …, T_{n-1}, порядок которых нам не существеннен. Это может быть словарь, либо набор статей в корпусе текстов (такие статьи обычно содержат все необходимые читателю выходные данные) и т.п. Объединим эти тексты в один текст, разделив их с помощью специальных символов разделителей, которые нигде больше в T_0, …, T_{n-1} не встречаются:

$$T = T_0\$_0 \dots, T_{n-1}\$_{n-1}.$$

Теперь необходимо задать какой-то лексикографический порядок на разделителях. Теорема 3 допускает *произвольный* порядок, лишь бы разделители лексикографически предшествовали всем символам из текстов $T_0, \dots, T_{n-1}$. Практика показывает, что для данных типа словаря является выгодным порядок разделителей, отвечающий лексикографическому порядку текстов T_k: $\$_j \prec \$_k \iff T_j \prec T_k$. Если строки T_0, …, T_{n-1} уже были лексикографически отсортированы, то такой порядок задать ещё проще: $\$_j \prec \$_k \iff j < k$.

После определения порядка символов следует воспользоваться сортировкой суффиксов для построения перестановки σ, удовлетворяющей условию (1).

Замечание 2. Если Вы пользуетесь языком Си, тексты T_i не содержат символа с кодом 0, и порядок на суффиксах задаётся по правилу $\$_j \prec \$_k \iff j < k$, то можно разделять тексты T_i только одним символом — с кодом 0, и воспользоваться следующей функцией для сравнения суффиксов $T[i..N]$ и $T[j..N]$ при сортировке:

```
int N;              /* длина минус 1 */
unsigned char *T;   /* текст T[0..N] */
int suffix_compare(int i, int j){
  int res=strcmp(T+i,T+j);
  if(res==0) res=i-j;
  return res;
}
```

Получив перестановку σ, необходимо построить текст $\widehat{B}$ по правилу:

$$\widehat{B}_i = \begin{cases} S_{\sigma(i)-1}T[0], & \text{если } S_{\sigma(i)-1}T[0] \neq \$_k, k = 0, \ldots, n-1, \\ \$, & \text{если } S_{\sigma(i)-1}T[0] = \$_k, \text{ при некотором } k = 0, \ldots, n-1. \end{cases}$$

Текст $\widehat{B}$ можно подвергнуть сжатию с использованием разнообразных методов: RLE, MTF, DC (Distance Coding), 01BFA (Е.Шелвин), MTH, Huffman, ARI, не считая 0-1 кодирования и прочих хитростей. Реализация этих методов подробно описана другими авторами (см. книгу [1]).

Приведём алгоритм восстановления набора текстов $\widehat{T}_j$, $j = 0, \ldots, n-1$, который является некоторой перестановкой текстов T_k, из текста $\widehat{B}$. Алгоритм пользуется соотношением (24). Отметим, что никакой дополнительной информации, кроме самого текста $\widehat{B}$ не требуется (в отличие от алгоритмов 2 и 3).

Алгоритм 4 *(Восстановление $\widehat{T}_j$).*
На выходе алгоритма определены переменные:
n — количество фрагментов,
$\widehat{l}_j$ — длина фрагмента j при $j = 0, \ldots, n-1$,

$\widehat{T}_j[0..\widehat{l}_j - 1]$ — фрагмент $\widehat{T}_j$, $j = 0, \ldots, n-1$.

1 Определить перестановку $\widehat{\delta}$ из $\widehat{B}$ с помощью алгоритма 1
2 $n :=$ количество символов \$ в $\widehat{B}[0..N]$;
3 $j := 0$; $i := -1$;

```
4  do (* инвариант: i = Î_{j-1} *)
5      if i ≠ −1 then s := δ̂(i); else s := δ̂(N); fi
6      l̂_j := 0;
7      while s ≥ n do
8              T̂_j[l̂_j] := B̂_s;
9              l̂_j := l̂_j + 1;
10             s := δ̂(s);
11     od
12     j := j + 1;
13     do
14         i := i + 1;
15         if i > N then goto 17; fi
16     od while B̂(i) ≠ $;
17 od while j < n;
```

5.3 Применение в качестве фильтра

В силу теоремы 5 для однозначного восстановления исходного текста T из $\widehat{B}$ необходимо ещё передать перестановку $\zeta = \gamma \circ \omega$ и номер I строки T в концептуальной матрице $\mathcal{M}$. Эта информация позволяет полностью восстановить исходный текст. Основной трюк заключается в том, что справедлива формула

$$\delta(i) = \begin{cases} \widehat{\delta}(i) & i = n, \ldots, N \\ \widehat{\delta}(\zeta(i)) & i = 0, \ldots, n-1. \end{cases} \qquad \text{см. (26)}$$

Доказательство теоремы 5 описывает метод восстановления T и σ.

В связи с этим возникает естественное предложение о применении $\widehat{B}$ в качестве предварительного фильтра неоднородных данных. Именно, можно назначить во входных данных разделитель (например, символ конца предложения — точку), и попытаться сгруппировать данные так, чтобы было больше близких контекстов в $\widehat{B}$. При этом можно свободно выбрать порядок на разделителях $\$_k$.

Имеет ли смысл такая группировка может показать только практика. При условии равновероятности всех перестановок для оптимального кодирования ζ можно использовать арифметическое сжатие: первое число — одно из n, следующее — одно из $n-1$ оставшихся и т.д. Таким образом

сжатая информация о ζ будет требовать

$$\log_2(n) + \log_2(n-1) + \ldots + \log_2(1) =$$
$$= \log_2(n!) \approx \frac{n(\log(n)-1)}{\ln 2} \approx 1.44n(\log(n)-1) \text{ бит.}$$

Если брать объём $N < 2^{30}$, $n \approx N/1000$, то количество дополнительной информации не превосходит $0.0023N$ байт, а хорошая группировка может, наверное, улучшить сжатие больше чем на 1%.

Возможны и другие хитрости. Например, можно разбивать разделители на группы и передавать информацию о перестановках внутри этих групп. В общем, этот метод представляется многообещающим в плане улучшения сжатия.

Благодарности

Автор пользуется случаем выразить признательность Е. Шелвину и В. Юкину за обсуждение алгоритма 4.

Список литературы

[1] Д. Ватолин, А. Ратушняк, М. Смирнов, и В. Юкин. *Методы сжатия данных*. Диалог-МИФИ, Москва, 2002. `http://www.compression.ru`.

[2] M. I. Abouelhoda, S. Kurtz, and E. Ohlebusch. The enhanced suffix array and its applications to genome analysis. In *Proceedings of the 2nd Workshop on Algorithms in Bioinformatics*, number 2452 in LNCS, pages 449–463, 2002. `http://theorie.informatik.uni-ulm.de/Personen/eo/PAPERS/WABI02.pdf`.

[3] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, Digital SRC, Palo Alto, 1994.

[4] J. Kärkkäinen and P. Sanders. Simple linear work suffix array construction. In *30th International Colloquium on Automata, Languages and Programming*, number 2719 in LNCS, pages 943–955, 2003. `http://citeseer.nj.nec.com/arkk03simple.html`.

[5] N. J. Larsson and K. Sadakane. Faster suffix sorting. Technical Report LU-CS-TR:99-214, LUNDFD6/(NFCS-3140)/1–20/(1999), Department of Computer Science, Lund University, Sweden, May 1999. `http://www.compression.ru/download/articles/bwt/sada_larsson_1999_pdf.rar`.

[6] U. Manber and G. Myers. Suffix arrays: a new method for on-line string searches. *SIAM J. Comput.*, 22(5):935–948, 1993.