

Юрий Лифшиц

ПОМИ РАН - СПбГУ ИТМО

Осень 2006

1 / 28

Во всяком хаосе есть космос, в каждом беспорядке скрыт тайный порядок

Карл Юнг

2 / 28

История BWT

Придумали: Michael Burrows и David Wheeler в 1994, работая в Digital Equipment Corporation (Теперь это Hewlett-Packard).

Реализации: bzip2, gzip

David Wheeler:



3 / 28

План лекции

- 1 Вычисление BWT
 - Определение преобразования
 - Вычисление в прямую сторону
- 2 Вычисление обратного преобразования
 - Магический вектор T
 - Восстановление текста с помощью вектора T
- 3 Применение к архивированию
 - Кодирование move-to-front
 - Почему BWT помогает сжатию текстов

4 / 28

Часть I

Как определяется преобразование Берроуза-Вилера?

Как вычислить BWT за линейное время?

5 / 28

Определение

Пусть дан текст $S = s_1 \dots s_n$. Преобразование Берроуза-Вилера получает из него новый текст следующим образом:

- Приписываем в конец знак \$ (последняя буква алфавита)
- Выписываем все $n + 1$ циклических сдвига текста
- Сортируем этот $n + 1$ текст в алфавитном порядке
- Выдаем последний столбец L

Зная только L , как догадаться, на каком месте в отсортированном списке шел исходный текст S ?

6 / 28

Пример

Пусть $S = racaa$, дописываем \$

Циклические сдвиги:

$racaa\$$
 $acaa\$r$
 $caaa\$r$
 $aa\$rac$
 $a\$raca$
 $\$racaa$

Сортируем:

$aa\$rac$
 $acaa\$r$
 $a\$raca$
 $caaa\$r$
 $racaa\$$
 $\$racaa$

Результат: $craa\$a$

7 / 28

BWT за линейное время

С какой фразы начинается алгоритм?

Конечно, "Построим суффиксное дерево..."

Напомним: чтобы построить **суффиксное дерево** (ST), нужно приписать специальный символ \$ к тексту, взять все $n + 1$ суффикс, подвесить их за начала и склеить все ветки, идущие по одинаковым буквам. В каждом листе записывается номер суффикса, который в нем заканчивается.

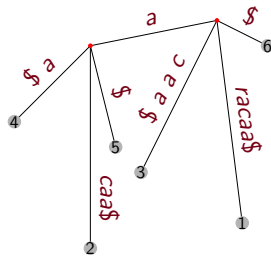
8 / 28

- Допишем \$
- Построим суффиксное дерево
- В каждой внутренней вершине отсортируем детей в алфавитном порядке
- Прочтем числа во всех листьях слева на право
- Читая i , пишем в ответ S_{i-1}

Наблюдения:

- Алфавитный порядок суффиксов совпадает с алфавитным порядком сдвигов
- В листьях мы читаем порядок стартов суффиксов
- Буква S_{i-1} - это последняя буква сдвига, начинающегося с S_i

Пример: racaa\$



Порядок суффиксов: aa\$, acaa\$, a\$, caa\$, racaa\$, \$
 Порядок сдвигов: aa\$rac, acaa\$r, a\$rac, caa\$ra, racaa\$, \$racaa
 Числа в листьях: 4, 2, 5, 3, 1, 6 Ответ: $S_3, S_1, S_4, S_2, S_6, S_5 = craaa

Часть II

Как вычислить обратное преобразование за линейное время?

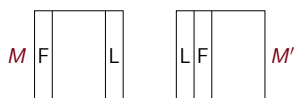
Вспомогательная конструкция: магический вектор T

Как вычислить T , и как с его помощью восстановить текст?

Определение вектора T

Пусть матрица букв M — это все отсортированные сдвиги текста S . Мы знаем только ее последний столбец L .

Определение: вспомогательная матрица M' получается перестановкой столбца L в начало.



Определение: для каждого k определим $T[k]$ так, чтобы k -ая строка M' совпадала с $T[k]$ -ой строкой M .

Переформулировка: пусть M_k — Это k -ый по алфавиту сдвиг текста, тогда **следующий** сдвиг в отсортированном списке сдвигов — это $M_{T[k]}$.

Пример магического вектора

Матрица M :

$$\begin{pmatrix} a & a & \$ & r & a & c \\ a & c & a & a & \$ & r \\ a & \$ & r & a & c & a \\ c & a & a & \$ & r & a \\ r & a & c & a & a & \$ \\ \$ & r & a & c & a & a \end{pmatrix}$$

Матрица M' :

$$\begin{pmatrix} c & a & a & \$ & r & a \\ r & a & c & a & a & \$ \\ a & a & \$ & r & a & c \\ a & c & a & a & \$ & r \\ \$ & r & a & c & a & a \\ a & \$ & r & a & c & a \end{pmatrix}$$

$T[1] = 4, T[2] = 5, T[3] = 1, T[4] = 2, T[5] = 6, T[6] = 3$

Вычисление магического вектора T

Два пробега по L , один по алфавиту:

- Для каждой буквы алфавита α вычислить количество раз $C(\alpha)$, которые она встречается
- Для каждой буквы алфавита α вычислить количество раз $D(\alpha)$, которые встречаются все меньшие ее буквы
- Для каждого номера $1 \leq i \leq n+1$ вычислить количество раз $P(i)$, которое буква L_i уже нам встретила выше по столбцу

Таинственная формула: $T[i] = D(L_i) + P(i) + 1$

Домашняя задача: как сэкономить один пробег по тексту?

Формула для T

Таинственная формула: $T[i] = D(L_i) + P(i) + 1$

Доказательство:

- Нам нужно узнать, на каком месте в отсортированном списке находится сдвиг, предыдущий к M_i
- Он начинается с буквы L_i
- Это значит он стоит после $D(L_i)$ сдвигов, которые начинаются на меньшие буквы
- (!) Среди сдвигов, которые начинаются на L_i он находится в точности на $P(i) + 1$ месте

Формула восстановления

Зная столбец L как узнать последнюю букву текста?

Глупый вопрос! Последняя буква — это всегда $\$$

Зная столбец L как узнать предпоследнюю букву текста?

Пусть $\$$ находится на месте l в столбце L . Тогда предпоследняя буква — это $L_{T[l]}$.

Доказательство: как мы знаем, M_l — это исходный текст, тогда $M_{T[l]}$ — это текст, в котором $\$$ перенесли в начало, т.е. на последнем месте $L_{T[l]}$ оказалась как раз предпоследняя буква.

17 / 28

От буквы к следующей

Пусть мы уже знаем, что $M_k = s_{j+1} \dots s_n \$ s_1 \dots s_j$, и соответственно $s_j = L_k$, тогда мы можем узнать предыдущую букву:

- По определению вектора T мы знаем, что $M_{T[k]} = s_j \dots s_n \$ s_1 \dots s_{j-1}$
- Таким образом, $s_{j-1} = L_{T[k]}$
- Формулы для всех букв текста от начала к концу:

$$\$ = s_{n+1} = L_l; \quad s_n = L_{T[l]}, \quad \dots$$

$$\dots \quad s_{n-k} = L_{T^{k+1}[l]} \quad \dots \quad s_1 = L_{T^n[l]}$$

18 / 28

Общий алгоритм обратного BWT

- Для каждой буквы алфавита α вычислить количество раз $C(\alpha)$, которые она встречается
- Для каждой буквы алфавита α вычислить количество раз $D(\alpha)$, которые встречаются все меньшие ее буквы
- Для каждого номера $1 \leq i \leq n+1$ вычислить количество раз $P(i)$, которое буква L_i уже нам встретилась выше по столбцу
- По формуле $T[i] = D(L_i) + P(i) + 1$ вычислить вектор T
- По формуле $s_{n-k} = L_{T^{k+1}[l]}$ вычислить все буквы текста от конца к началу

19 / 28

Пример вычисления: *racaa* $\$$

Столбец L : *craa* $\$$ *a*

Вектор T : [4,5,1,2,6,3]

Стартовый индекс l : 5

$$s_6 = L_l = L_5 = \$,$$

$$s_5 = L_{T[5]} = L_6 = a,$$

$$s_4 = L_{T[6]} = L_3 = a,$$

$$s_3 = L_{T[3]} = L_1 = c,$$

$$s_2 = L_{T[1]} = L_4 = a,$$

$$s_1 = L_{T[4]} = L_2 = r$$

20 / 28

Часть III

Что делать с текстом после преобразования Берроуза-Вилера?

Почему BWT так хорошо работает?

21 / 28

Определение MTF

Move-to-front — это перезапись текста в “адаптирующей” кодировке:

- Начинаем с кодировки $Enc(a) = 0, \dots, Enc(z) = 25$
- Читаем текст слева направо
- Читаем очередную букву α
- Пишем в выходной поток $Enc(\alpha)$
- Меняем кодировку: $Enc(\alpha) = 0$, для всех букв, которые стояли выше α применяем $Enc := Enc + 1$
- Результат: последовательность чисел и финальная кодировка

Как обратно восстановить текст из кодировки move-to-front?

22 / 28

Пример: *racaa* $\$$

Текст: *racaa* $\$$ Начальная кодировка:

$$Enc(a) = 0, Enc(c) = 1, Enc(r) = 2, Enc(\$) = 3$$

Читаем r — Пишем 2, новая кодировка *rac* $\$$

Читаем a — Пишем 1, новая кодировка *arc* $\$$

Читаем c — Пишем 2, новая кодировка *cra* $\$$

Читаем a — Пишем 2, новая кодировка *acr* $\$$

Читаем a — Пишем 0, новая кодировка *acr* $\$$

Читаем $\$$ — Пишем 3, финальная кодировка *acr* $\$$

Результат: 212203, кодировка *acr* $\$$

23 / 28

Почему BWT работает?

Схема архивирования:

- Применить к тексту прямое преобразование Берроуза-Вилера
- Закодировать полученный последний столбец с помощью Move-to-Front
- Полученную числовую последовательность заархивировать классическим архиватором (например, Хаффманом)

Почему текст полученный по BWT+MTF лучше архивируется, чем исходный?

Ответ: После BWT текст становится “локально-однородным”, после MTF последовательность содержит много маленьких чисел и мало больших.

24 / 28

Иллюстрация из [BW94]:

final char (L)	sorted rotations
a	n to decompress. It achieves compression
o	n to perform only comparisons to a depth
o	n transformation} This section describes
o	n transformation} We use the example and
o	n treats the right-hand side as the most
a	n tree for each 16 kbyte input block, enc
a	n tree in the output stream, then encodes
i	n turn, set $L[i]$ to be the
i	n turn, set $R[i]$ to be the
o	n unusual data. Like the algorithm of Man
a	n use a single set of probabilities table
e	n using the positions of the suffixes in
i	n value at a given point in the vector R
e	n we present modifications that improve t
e	n when the block size is quite large. Ho
i	n which codes that have not been seen in
i	n with $\$ch\$$ appear in the $\{\em same order$
i	n with $\$ch\$$. In our exam
o	n with Huffman or arithmetic coding. Bri
o	n with figures given by Bell" cite{bell} .

25 / 28

Задача

Статус задач: решать их необязательно, но они будут выдаваться на экзамен (на пятерку и в спорных случаях).

Как по столбцу L вычислить вектор T за два пробега по тексту и один пробег по алфавиту?

Можно ли реализовать Move-to-Front за время $O(n \log |\Sigma|)$, где Σ — используемый алфавит, а n — длина текста?

26 / 28

Главные моменты

Сегодня мы узнали:

- Преобразование Берроуза-Вилера: взять все циклические сдвиги, отсортировать и выдать последний столбец
- Прямое преобразование считается с помощью суффиксного дерева
- Обратное преобразование считается с помощью магического вектора T
- После BWT нужно применить move-to-front, затем Хаффмана.

Вопросы?

27 / 28

Источники

Страница курса <http://logic.pdmi.ras.ru/~yura/internet.html>

Использованные материалы:

-  [M. Burrows and D. Wheeler.](#)
A block sorting lossless data compression algorithm.
<http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-124.pdf>
-  [Mark Nelson](#)
Data Compression with the Burrows-Wheeler Transform
<http://marknelson.us/1996/09/01/bwt/>
-  [Giovanni Manzini](#)
The Burrows-Wheeler Transform: Theory and Practice
<http://www.mfn.unipmn.it/~manzini/papers/mfcs99x.pdf>
-  [Юрий Лифшиц](#)
Преобразование Берроуза-Вилера. Аудиолекция.
http://rpod.ru/personal/storage/00/00/00/34/23/Yury_Lifshits_Strings_03_BWT.MP3

28 / 28