Лекция 11: Преобразование Бэрроуза–Уилера (BWT)

А. М. Шур

Кафедра алгебры и фундаментальной информатики УрФУ

2 мая 2020 г.

Бэрроуз и Уилер

В 1994 году два ученых из Кембриджа — Дэвид Уилер и его бывший аспирант Майкл Бэрроуз — опубликовали статью

- M. Burrows, D.J. Wheeler. A block sorting lossless data compression algorithm. Tech. Report 124, Digital Equipment Corporation, 1994

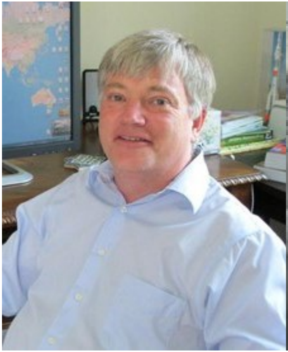
о новом подходе к сжатию данных, основанном на несжимающем комбинаторном преобразовании, получившем позже название BWT (Burrows–Wheeler Transform)

Бэрроуз и Уилер

В 1994 году два ученых из Кембриджа — Дэвид Уилер и его бывший аспирант Майкл Бэрроуз — опубликовали статью

- M. Burrows, D.J. Wheeler. A block sorting lossless data compression algorithm. Tech. Report 124, Digital Equipment Corporation, 1994

о новом подходе к сжатию данных, основанном на несжимающем комбинаторном преобразовании, получившем позже название BWT (Burrows–Wheeler Transform)



Слева — Майкл Бэрроуз, один из авторов первого интернет-поисковика AltaVista

Справа — Дэвид Уилер, один из пионеров программирования, изобретатель ассемблера и процедур

Перестановка и действие перестановки

Перестановка множества $\{1, \ldots, n\}$ есть биекция этого множества на себя

- Перестановку записывают в виде строки/массива $\theta[1..n]$ так, что $\theta[i] = \theta(i)$
  - $\theta = 45321$ переводит 1 в 4, 2 в 5, 3 в 3, 4 в 2 и 5 в 1

Перестановка и действие перестановки

Перестановка множества $\{1, \ldots, n\}$ есть биекция этого множества на себя

- Перестановку записывают в виде строки/массива $\theta[1..n]$ так, что $\theta[i] = \theta(i)$
  - $\theta = 45321$ переводит 1 в 4, 2 в 5, 3 в 3, 4 в 2 и 5 в 1

Перестановкой n-элементного множества можно подействовать на любое n-элементное множество, если его элементы естественным образом линейно упорядочены

- Например, можно действовать перестановкой
  - на строку (через номера позиций)
  - на граф (через номера вершин)
  - на матрицу (через номера строк или столбцов)

★ Формально, для последовательности объектов $(y_1, \ldots, y_n)$ результат $\theta(y_1, \ldots, y_n)$ действия перестановки θ есть $(y_{\theta(1)}, \ldots, y_{\theta(n)})$

Перестановка множества $\{1, \ldots, n\}$ есть биекция этого множества на себя

- Перестановку записывают в виде строки/массива $\theta[1..n]$ так, что $\theta[i] = \theta(i)$
  - $\theta = 45321$ переводит 1 в 4, 2 в 5, 3 в 3, 4 в 2 и 5 в 1

Перестановкой n-элементного множества можно подействовать на любое n-элементное множество, если его элементы естественным образом линейно упорядочены

- Например, можно действовать перестановкой
  - на строку (через номера позиций)
  - на граф (через номера вершин)
  - на матрицу (через номера строк или столбцов)

★ Формально, для последовательности объектов $(y_1, \ldots, y_n)$ результат $\theta(y_1, \ldots, y_n)$ действия перестановки θ есть $(y_{\theta(1)}, \ldots, y_{\theta(n)})$

Пример

Если $\theta = 769345218$, то $\theta(\text{ПОДСТРОКА}) = \text{КОСТРОПАД}$
Тот же результат получится при $\theta = 729345618$

★ Любая перестановка превращает слово в его анаграмму (и обратно: любая анаграмма слова получается из него действием некоторой перестановки)

Преобразование BWT определяется следующим образом:

- ⋆ Алфавит упорядочен, строки можно сортировать лексикографически
- Взять текст T и выписать все $|T|$ его циклических сдвигов в столбик, получив квадратную матрицу
- Отсортировать строки матрицы по возрастанию, получив матрицу $M(T)$
- Вернуть последний столбец матрицы ($\mathrm{bwt}(T)$)

Преобразование BWT определяется следующим образом:

- ⋆ Алфавит упорядочен, строки можно сортировать лексикографически
- Взять текст T и выписать все $|T|$ его циклических сдвигов в столбик, получив квадратную матрицу
- Отсортировать строки матрицы по возрастанию, получив матрицу $M(T)$
- Вернуть последний столбец матрицы ($\mathrm{bwt}(T)$)

★ Каждая буква в T является последней ровно для одного циклического сдвига

$\Rightarrow$ $\mathrm{bwt}(T)$ состоит из тех же символов с теми же частотами, что и T

$\Rightarrow$ $\mathrm{bwt}(T)$ является анаграммой T

$\Rightarrow$ bwt как функция является действием некоторой перестановки (зависящей от T)

★ bwt — не биекция, так как текстам, являющимся циклическими сдвигами друг друга, соответствует одна и та же матрица, а значит, один и тот же bwt-образ

Преобразование BWT определяется следующим образом:

- ⋆ Алфавит упорядочен, строки можно сортировать лексикографически
- Взять текст T и выписать все $|T|$ его циклических сдвигов в столбик, получив квадратную матрицу
- Отсортировать строки матрицы по возрастанию, получив матрицу $M(T)$
- Вернуть последний столбец матрицы ($\mathrm{bwt}(T)$)

★ Каждая буква в T является последней ровно для одного циклического сдвига

$\Rightarrow$ $\mathrm{bwt}(T)$ состоит из тех же символов с теми же частотами, что и T

$\Rightarrow$ $\mathrm{bwt}(T)$ является анаграммой T

$\Rightarrow$ bwt как функция является действием некоторой перестановки (зависящей от T)

★ bwt — не биекция, так как текстам, являющимся циклическими сдвигами друг друга, соответствует одна и та же матрица, а значит, один и тот же bwt-образ

$\cdots$ говорят, что Бэрроуз придумал это преобразование еще студентом, но не нашел, куда его приложить

Преобразование Бэрроуза–Уилера – пример

Рассмотрим наш стандартный пример $T = \text{БанБананана\#}$

- В utf-8 или cp1251 имеем $\# < \text{Б} < \text{а} < \text{н}$; тогда

$$M(\text{БанБананана\#}) = \begin{array}{cccccccccccc}
\# & Б & а & н & Б & а & н & а & н & а & н & а \\
Б & а & н & Б & а & н & а & н & а & н & а & \# \\
\hline
Б & а & н & а & н & а & н & а & \# & Б & а & н \\
а & \# & Б & а & н & Б & а & н & а & н & а & н \\
а & н & Б & а & н & а & н & а & н & а & \# & Б \\
а & н & а & \# & Б & а & н & Б & а & н & а & н \\
а & н & а & н & а & \# & Б & а & н & Б & а & н \\
а & н & а & н & а & н & а & \# & Б & а & н & Б \\
н & а & \# & Б & а & н & Б & а & н & а & н & а \\
н & а & н & а & \# & Б & а & н & Б & а & н & а \\
н & а & н & а & н & а & \# & Б & а & н & Б & а \\
н & Б & а & н & а & н & а & н & а & \# & Б & а
\end{array}$$

$\mathrm{bwt}(\text{БанБананана\#}) = \text{а\#ннБннБаааа}$

Преобразование Бэрроуза–Уилера – пример

Рассмотрим наш стандартный пример $T = \text{БанБананана\#}$

- В utf-8 или cp1251 имеем $\# < \text{Б} < \text{а} < \text{н}$; тогда

$$M(\text{БанБананана\#}) = \begin{array}{cccccccccccc}
\# & Б & а & н & Б & а & н & а & н & а & н & \color{red}{а} \\
Б & а & н & Б & а & н & а & н & а & н & а & \color{red}{\#} \\
\hline
Б & а & н & а & н & а & н & а & \# & Б & а & \color{red}{н} \\
а & \# & Б & а & н & Б & а & н & а & н & а & \color{red}{н} \\
а & н & Б & а & н & а & н & а & н & а & \# & \color{red}{Б} \\
а & н & а & \# & Б & а & н & Б & а & н & а & \color{red}{н} \\
а & н & а & н & а & \# & Б & а & н & Б & а & \color{red}{н} \\
а & н & а & н & а & н & а & \# & Б & а & н & \color{red}{Б} \\
н & а & \# & Б & а & н & Б & а & н & а & н & \color{red}{а} \\
н & а & н & а & \# & Б & а & н & Б & а & н & \color{red}{а} \\
н & а & н & а & н & а & \# & Б & а & н & Б & \color{red}{а} \\
н & Б & а & н & а & н & а & н & а & \# & Б & \color{red}{а}
\end{array}$$

$\text{bwt}(\text{БанБананана\#}) = \text{а\#ннБннБаааа}$

★ Начало каждой строки является правым контекстом в T для последнего символа этой строки

★ Символы, у которых совпадают длинные контексты, часто бывают равны

⇒ Если символы в T сильно зависят от контекста, в $\text{bwt}(T)$ будут длинные последовательности одинаковых символов, которые легко сжимать

План

- Чтобы BWT хотя бы теоретически подходило для препроцессинга данных перед сжатием, нужно доказать, что T можно восстановить, зная bwt(T)

Чтобы получить практически осмысленный алгоритм сжатия, нужно

- быстро строить bwt(T)
- быстро восстанавливать T из bwt(T)
- эффективно сжимать bwt(T) без потерь

План

- Чтобы BWT хотя бы теоретически подходило для препроцессинга данных перед сжатием, нужно доказать, что T можно восстановить, зная $\mathrm{bwt}(T)$

Чтобы получить практически осмысленный алгоритм сжатия, нужно

- быстро строить $\mathrm{bwt}(T)$
- быстро восстанавливать T из $\mathrm{bwt}(T)$
- эффективно сжимать $\mathrm{bwt}(T)$ без потерь

Вначале научимся решать задачу восстановления:

- Как уже упоминалось, разные тексты имеют одинаковый результат BWT, если являются циклическими сдвигами друг друга, а значит — строками одной матрицы
- С другой стороны, мы покажем, как по $\mathrm{bwt}(T)$ построить матрицу $M(T)$
- Для однозначности восстановления T можно хранить номер строки в $M(T)$, которая соответствует T, но удобнее дописать к T уникальный символ конца файла, как мы сделали в примере; тогда номер строки T в $M(T)$ равен позиции этого символа в $\mathrm{bwt}(T)$

Восстановление матрицы $M(T)$

★ В любом столбце матрицы $M(T)$ записана анаграмма T

★ В матрице, составленной из первых i столбцов $M(T)$, где $1 \leqslant i \leqslant n$, по строкам записаны, в лексикографическом порядке, все циклические подстроки T

- циклическая подстрока — это либо $T[i..j]$, $i \leqslant j$, либо $T[j..n]T[1..i]$, $i < j$

Восстановление матрицы $M(T)$

- ★ В любом столбце матрицы $M(T)$ записана анаграмма T
- ★ В матрице, составленной из первых i столбцов $M(T)$, где $1 \leqslant i \leqslant n$, по строкам записаны, в лексикографическом порядке, все циклические подстроки T
  - циклическая подстрока — это либо $T[i..j]$, $i \leqslant j$, либо $T[j..n]T[1..i]$, $i < j$
- Отсортировав $\mathrm{bwt}(T)$, получим первый столбец $M(T)$
- Если поставить первый столбец справа от последнего, в полученной матрице с двумя столбцами записаны все циклические подстроки w длины 2
  - Отсортировав эти подстроки, получим первые два столбца $M(T)$
  - Приставив спереди последний столбец, получим циклические подстроки длины 3
- Будем повторять процедуру до заполнения матрицы:

- ★ В любом столбце матрицы $M(T)$ записана анаграмма T
- ★ В матрице, составленной из первых i столбцов $M(T)$, где $1 \leqslant i \leqslant n$, по строкам записаны, в лексикографическом порядке, все циклические подстроки T
  - циклическая подстрока — это либо $T[i..j]$, $i \leqslant j$, либо $T[j..n]T[1..i]$, $i < j$
- Отсортировав bwt(T), получим первый столбец $M(T)$
- Если поставить первый столбец справа от последнего, в полученной матрице с двумя столбцами записаны все циклические подстроки w длины 2
  - Отсортировав эти подстроки, получим первые два столбца $M(T)$
  - Приставив спереди последний столбец, получим циклические подстроки длины 3
- Будем повторять процедуру до заполнения матрицы:

$$M(\text{БанБананана\#}) = \begin{array}{cccccccccccccc} \# & . & . & . & . & . & . & . & . & . & . & . & . & \text{а} \\ \text{Б} & . & . & . & . & . & . & . & . & . & . & . & . & \# \\ \text{Б} & . & . & . & . & . & . & . & . & . & . & . & . & \text{н} \\ \text{а} & . & . & . & . & . & . & . & . & . & . & . & . & \text{н} \\ \text{а} & . & . & . & . & . & . & . & . & . & . & . & . & \text{Б} \\ \text{а} & . & . & . & . & . & . & . & . & . & . & . & . & \text{н} \\ \text{а} & . & . & . & . & . & . & . & . & . & . & . & . & \text{н} \\ \text{а} & . & . & . & . & . & . & . & . & . & . & . & . & \text{Б} \\ \text{н} & . & . & . & . & . & . & . & . & . & . & . & . & \text{а} \\ \text{н} & . & . & . & . & . & . & . & . & . & . & . & . & \text{а} \\ \text{н} & . & . & . & . & . & . & . & . & . & . & . & . & \text{а} \\ \text{н} & . & . & . & . & . & . & . & . & . & . & . & . & \text{а} \end{array} \Longrightarrow \begin{array}{cc} \text{а} & \# \\ \# & \text{Б} \\ \text{н} & \text{Б} \\ \text{н} & \text{а} \\ \text{Б} & \text{а} \\ \text{н} & \text{а} \\ \text{н} & \text{а} \\ \text{Б} & \text{а} \\ \text{а} & \text{н} \\ \text{а} & \text{н} \\ \text{а} & \text{н} \\ \text{а} & \text{н} \end{array} \Longrightarrow \begin{array}{cc} \# & \text{Б} \\ \text{Б} & \text{а} \\ \text{Б} & \text{а} \\ \text{а} & \# \\ \text{а} & \text{н} \\ \text{а} & \text{н} \\ \text{а} & \text{н} \\ \text{а} & \text{н} \\ \text{н} & \text{Б} \\ \text{н} & \text{а} \\ \text{н} & \text{а} \\ \text{н} & \text{а} \end{array}$$

Восстановление матрицы $M(T)$

- ★ В любом столбце матрицы $M(T)$ записана анаграмма T
- ★ В матрице, составленной из первых i столбцов $M(T)$, где $1 \leqslant i \leqslant n$, по строкам записаны, в лексикографическом порядке, все циклические подстроки T
  - циклическая подстрока — это либо $T[i..j]$, $i \leqslant j$, либо $T[j..n]T[1..i]$, $i < j$
- Отсортировав bwt(T), получим первый столбец $M(T)$
- Если поставить первый столбец справа от последнего, в полученной матрице с двумя столбцами записаны все циклические подстроки w длины 2
  - Отсортировав эти подстроки, получим первые два столбца $M(T)$
  - Приставив спереди последний столбец, получим циклические подстроки длины 3
- Будем повторять процедуру до заполнения матрицы:

$$
M(\text{БанБананана\#}) = \begin{array}{cccc}
\# & \text{Б} & \ldots\ldots\ldots & \text{а} \\
\text{Б} & \text{а} & \ldots\ldots\ldots & \# \\
\text{Б} & \text{а} & \ldots\ldots\ldots & \text{н} \\
\text{а} & \# & \ldots\ldots\ldots & \text{н} \\
\text{а} & \text{н} & \ldots\ldots\ldots & \text{Б} \\
\text{а} & \text{н} & \ldots\ldots\ldots & \text{н} \\
\text{а} & \text{н} & \ldots\ldots\ldots & \text{н} \\
\text{а} & \text{н} & \ldots\ldots\ldots & \text{Б} \\
\text{н} & \text{Б} & \ldots\ldots\ldots & \text{а} \\
\text{н} & \text{а} & \ldots\ldots\ldots & \text{а} \\
\text{н} & \text{а} & \ldots\ldots\ldots & \text{а} \\
\text{н} & \text{а} & \ldots\ldots\ldots & \text{а}
\end{array}
\Longrightarrow
\begin{array}{ccc}
\text{а} & \# & \text{Б} \\
\# & \text{Б} & \text{а} \\
\text{н} & \text{Б} & \text{а} \\
\text{н} & \text{а} & \# \\
\text{Б} & \text{а} & \text{н} \\
\text{н} & \text{а} & \text{н} \\
\text{н} & \text{а} & \text{н} \\
\text{Б} & \text{а} & \text{н} \\
\text{а} & \text{н} & \text{Б} \\
\text{а} & \text{н} & \text{а} \\
\text{а} & \text{н} & \text{а} \\
\text{а} & \text{н} & \text{а}
\end{array}
\Longrightarrow
\begin{array}{ccc}
\# & \text{Б} & \text{а} \\
\text{Б} & \text{а} & \text{н} \\
\text{Б} & \text{а} & \text{н} \\
\text{а} & \# & \text{Б} \\
\text{а} & \text{н} & \text{Б} \\
\text{а} & \text{н} & \text{а} \\
\text{а} & \text{н} & \text{а} \\
\text{а} & \text{н} & \text{а} \\
\text{н} & \text{Б} & \text{а} \\
\text{н} & \text{а} & \# \\
\text{н} & \text{а} & \text{н} \\
\text{н} & \text{а} & \text{н}
\end{array}
$$

Восстановление матрицы $M(T)$

- ★ В любом столбце матрицы $M(T)$ записана анаграмма T
- ★ В матрице, составленной из первых i столбцов $M(T)$, где $1 \leqslant i \leqslant n$, по строкам записаны, в лексикографическом порядке, все циклические подстроки T
  - циклическая подстрока — это либо $T[i..j]$, $i \leqslant j$, либо $T[j..n]T[1..i]$, $i < j$
- Отсортировав bwt(T), получим первый столбец $M(T)$
- Если поставить первый столбец справа от последнего, в полученной матрице с двумя столбцами записаны все циклические подстроки w длины 2
  - Отсортировав эти подстроки, получим первые два столбца $M(T)$
  - Приставив спереди последний столбец, получим циклические подстроки длины 3
- Будем повторять процедуру до заполнения матрицы:

$$M(\text{БанБананана\#}) = \begin{array}{cccccccccccc} \# & Б & а & . & . & . & . & . & . & . & . & \color{red}{а} \\ Б & а & н & . & . & . & . & . & . & . & . & \color{red}{\#} \\ Б & а & н & . & . & . & . & . & . & . & . & \color{red}{н} \\ а & \# & Б & . & . & . & . & . & . & . & . & \color{red}{н} \\ а & н & Б & . & . & . & . & . & . & . & . & \color{red}{Б} \\ а & н & а & . & . & . & . & . & . & . & . & \color{red}{н} \\ а & н & а & . & . & . & . & . & . & . & . & \color{red}{н} \\ а & н & а & . & . & . & . & . & . & . & . & \color{red}{Б} \\ н & Б & а & . & . & . & . & . & . & . & . & \color{red}{а} \\ н & а & \# & . & . & . & . & . & . & . & . & \color{red}{а} \\ н & а & н & . & . & . & . & . & . & . & . & \color{red}{а} \\ н & а & н & . & . & . & . & . & . & . & . & \color{red}{а} \end{array} \Longrightarrow \begin{array}{cccc} а & \# & Б & а \\ \# & Б & а & н \\ н & Б & а & н \\ н & а & \# & Б \\ Б & а & н & Б \\ н & а & н & а \\ н & а & н & а \\ Б & а & н & а \\ а & н & Б & а \\ а & н & а & \# \\ а & н & а & н \\ а & н & а & н \end{array} \Longrightarrow \begin{array}{cccc} \# & Б & а & н \\ Б & а & н & Б \\ Б & а & н & а \\ а & \# & Б & а \\ а & н & Б & а \\ а & н & а & \# \\ а & н & а & н \\ а & н & а & н \\ н & Б & а & н \\ н & а & \# & Б \\ н & а & н & а \\ н & а & н & а \end{array}$$

Восстановление матрицы $M(T)$

- ★ В любом столбце матрицы $M(T)$ записана анаграмма T
- ★ В матрице, составленной из первых i столбцов $M(T)$, где $1 \leqslant i \leqslant n$, по строкам записаны, в лексикографическом порядке, все циклические подстроки T
  - циклическая подстрока — это либо $T[i..j]$, $i \leqslant j$, либо $T[j..n]T[1..i]$, $i < j$
- Отсортировав bwt(T), получим первый столбец $M(T)$
- Если поставить первый столбец справа от последнего, в полученной матрице с двумя столбцами записаны все циклические подстроки w длины 2
  - Отсортировав эти подстроки, получим первые два столбца $M(T)$
  - Приставив спереди последний столбец, получим циклические подстроки длины 3
- Будем повторять процедуру до заполнения матрицы:

$$M(\text{БанБананана\#}) = \begin{array}{cccccccccccc}
\# & Б & а & н & Б & а & н & а & н & а & н & а \\
Б & а & н & Б & а & н & а & н & а & н & а & \# \\
Б & а & н & а & н & а & н & а & \# & Б & а & н \\
а & \# & Б & а & н & Б & а & н & а & н & а & н \\
а & н & а & \# & Б & а & н & Б & а & н & а & н \\
а & н & а & н & а & \# & Б & а & н & Б & а & н \\
а & н & а & н & а & н & а & \# & Б & а & н & Б \\
а & н & Б & а & н & а & н & а & н & а & \# & Б \\
н & а & \# & Б & а & н & Б & а & н & а & н & а \\
н & а & н & а & \# & Б & а & н & Б & а & н & а \\
н & а & н & а & н & а & \# & Б & а & н & Б & а \\
н & Б & а & н & а & н & а & н & а & \# & Б & а
\end{array}$$

Восстановление текста за линейное время

- Матрица имеет размер n^2, ее нельзя восстановить быстрее, чем за $O(n^2)$
- Нам нужна не матрица, а одна строка; как восстановить ее за $O(n)$?

- Матрица имеет размер n^2, ее нельзя восстановить быстрее, чем за $O(n^2)$
- Нам нужна не матрица, а одна строка; как восстановить ее за $O(n)$?

Сортировка строки $w[1..n]$ над упорядоченным алфавитом — это перестановка σ такая, что строка $\sigma(w)$ отсортирована: $i < j$ влечет $\sigma(w)[i] \leqslant \sigma(w)[j]$

- Если в w есть повторяющиеся буквы, то сортировок у него несколько
- Среди них одна не меняет порядок следования одинаковых букв

★ Стабильная сортировка σ_w слова w — это перестановка, удовлетворяющая условиям
  - $i < j$ влечет $\sigma_w(w)[i] \leqslant \sigma_w(w)[j]$
  - $i < j$ и $w[i] = w[j]$ влечет $\sigma_w(i) < \sigma_w(j)$

Восстановление текста за линейное время

- Матрица имеет размер n^2, ее нельзя восстановить быстрее, чем за $O(n^2)$
- Нам нужна не матрица, а одна строка; как восстановить ее за $O(n)$?

Сортировка строки $w[1..n]$ над упорядоченным алфавитом — это перестановка σ такая, что строка $\sigma(w)$ отсортирована: $i < j$ влечет $\sigma(w)[i] \leqslant \sigma(w)[j]$

- Если в w есть повторяющиеся буквы, то сортировок у него несколько
- Среди них одна не меняет порядок следования одинаковых букв

★ Стабильная сортировка σ_w слова w — это перестановка, удовлетворяющая условиям
  - $i < j$ влечет $\sigma_w(w)[i] \leqslant \sigma_w(w)[j]$
  - $i < j$ и $w[i] = w[j]$ влечет $\sigma_w(i) < \sigma_w(j)$

Пример: $\sigma = 4\,1\,9\,10\,2\,11\,12\,3\,5\,6\,7\,8$ — стабильная сортировка для $w =$ а#ннБннБаааа

Восстановление текста за линейное время

- Матрица имеет размер n^2, ее нельзя восстановить быстрее, чем за $O(n^2)$
- Нам нужна не матрица, а одна строка; как восстановить ее за $O(n)$?

Сортировка строки $w[1..n]$ над упорядоченным алфавитом — это перестановка σ такая, что строка $\sigma(w)$ отсортирована: $i < j$ влечет $\sigma(w)[i] \leqslant \sigma(w)[j]$

- Если в w есть повторяющиеся буквы, то сортировок у него несколько
- Среди них одна не меняет порядок следования одинаковых букв

★ Стабильная сортировка σ_w слова w — это перестановка, удовлетворяющая условиям
  - $i < j$ влечет $\sigma_w(w)[i] \leqslant \sigma_w(w)[j]$
  - $i < j$ и $w[i] = w[j]$ влечет $\sigma_w(i) < \sigma_w(j)$

Пример: $\sigma = 4\,1\,9\,10\,2\,11\,12\,3\,5\,6\,7\,8$ — стабильная сортировка для $w =$ а#ннБннБаааа

★ Для быстрого восстановления текста T по bwt(T) нужна перестановка, обратная к стабильной сортировке bwt(T)
  - напомним, что обратная к σ — перестановка σ^{-1} такая, что $(\forall i)\, \sigma^{-1}(\sigma(i)) = i$

В Примере $\sigma^{-1} = 2\,5\,8\,1\,9\,10\,11\,12\,3\,4\,6\,7$

★ Если σ — стабильная сортировка bwt(T), то $\sigma(i)$ — позиция, которую занимает в первом столбце символ bwt(T)[i]
  - ⋆ Значит, i-ый символ первого столбца — это bwt(T)[$\sigma^{-1}(i)$]

Восстановление текста за линейное время (2)

Восстановим T по $\mathrm{bwt}(T)$ и σ^{-1}:

- Номер i нужной строки — это позиция $\#$ в $\mathrm{bwt}(T)$

Восстановление текста за линейное время (2)

Восстановим T по $\mathrm{bwt}(T)$ и σ^{-1}:

- Номер i нужной строки — это позиция $\#$ в $\mathrm{bwt}(T)$
- Первый символ i-ой строки = i-й символ первого столбца = $\sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$

Восстановление текста за линейное время (2)

Восстановим T по $\mathrm{bwt}(T)$ и σ^{-1}:

- Номер i нужной строки — это позиция $\#$ в $\mathrm{bwt}(T)$
- Первый символ i-ой строки $=$ i-й символ первого столбца $= \sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

Восстановим T по $\mathrm{bwt}(T)$ и σ^{-1}:

- Номер i нужной строки — это позиция $\#$ в $\mathrm{bwt}(T)$
- Первый символ i-ой строки = i-й символ первого столбца = $\sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.
  $\Rightarrow$ $T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Восстановим T по bwt(T) и σ^{-1}:

- Номер i нужной строки — это позиция # в bwt(T)
- Первый символ i-ой строки = i-й символ первого столбца = $\sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.
  $\Rightarrow$ $T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Пример: bwt(T) = а#ннБннБаааа, σ^{-1} = 2 5 8 1 9 10 11 12 3 4 6 7

$$i = 2$$

T= . . . . #

Восстановление текста за линейное время (2)

Восстановим T по $\mathrm{bwt}(T)$ и σ^{-1}:

- Номер i нужной строки — это позиция # в $\mathrm{bwt}(T)$
- Первый символ i-ой строки $=$ i-й символ первого столбца $= \sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow \quad T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow \quad T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.
  $\Rightarrow \quad T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Пример: $\mathrm{bwt}(T) =$ а#ннБннБаааа, $\sigma^{-1} =$ 2 5 8 1 9 10 11 12 3 4 6 7

$$\sigma^{-1}(2) = 5 \quad \mathrm{bwt}(T)[5] = \text{Б}$$

T= Б . . . . #

Восстановление текста за линейное время (2)

Восстановим T по $\mathrm{bwt}(T)$ и σ^{-1}:

- Номер i нужной строки — это позиция # в $\mathrm{bwt}(T)$
- Первый символ i-ой строки = i-й символ первого столбца = $\sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.
  $\Rightarrow$ $T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Пример: $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$, $\sigma^{-1} = 2\ 5\ 8\ 1\ 9\ 10\ 11\ 12\ 3\ 4\ 6\ 7$

$$\sigma^{-1}(5) = 9 \quad \mathrm{bwt}(T)[9] = \text{а}$$

T= Б а . . . . #

Восстановление текста за линейное время (2)

Восстановим T по bwt(T) и σ^{-1}:

- Номер i нужной строки — это позиция $\#$ в bwt(T)
- Первый символ i-ой строки $=$ i-й символ первого столбца $= \sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.
  $\Rightarrow$ $T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Пример: bwt(T) = а#ннБннБаааа, σ^{-1} = 2 5 8 1 9 10 11 12 3 4 6 7

$$\sigma^{-1}(9) = 3 \quad \mathrm{bwt}(T)[3] = \text{н}$$

T= Б а н . . . . #

Восстановление текста за линейное время (2)

Восстановим T по $\mathrm{bwt}(T)$ и σ^{-1}:

- Номер i нужной строки — это позиция # в $\mathrm{bwt}(T)$
- Первый символ i-ой строки = i-й символ первого столбца = $\sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.
  $\Rightarrow$ $T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Пример: $\mathrm{bwt}(T) =$ а#ннБннБаааа, $\sigma^{-1} = 2\ 5\ 8\ 1\ 9\ 10\ 11\ 12\ 3\ 4\ 6\ 7$

$$\sigma^{-1}(3) = 8 \quad \mathrm{bwt}(T)[8] = \text{Б}$$

T= Б а н Б . . . . #

Восстановление текста за линейное время (2)

Восстановим T по $\mathrm{bwt}(T)$ и σ^{-1}:

- Номер i нужной строки — это позиция # в $\mathrm{bwt}(T)$
- Первый символ i-ой строки = i-й символ первого столбца = $\sigma^{-1}(i)$-й символ последнего столбца

  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)

  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$
- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.

  $\Rightarrow$ $T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Пример: $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$, $\sigma^{-1} = 2\ 5\ 8\ 1\ 9\ 10\ 11\ 12\ 3\ 4\ 6\ 7$

$$\sigma^{-1}(8) = 12 \quad \mathrm{bwt}(T)[12] = \text{а}$$

T= Б а н Б а . . . . . . #

Восстановление текста за линейное время (2)

Восстановим T по $\mathrm{bwt}(T)$ и σ^{-1}:

- Номер i нужной строки — это позиция # в $\mathrm{bwt}(T)$
- Первый символ i-ой строки = i-й символ первого столбца = $\sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.
  $\Rightarrow$ $T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Пример: $\mathrm{bwt}(T) =$ а#ннБннБаааа, $\sigma^{-1} = 2\ 5\ 8\ 1\ 9\ 10\ 11\ 12\ 3\ 4\ 6\ 7$

$$\sigma^{-1}(12) = 7 \quad \mathrm{bwt}(T)[7] = \text{н}$$

T= Б а н Б а н . . . . . #

Восстановление текста за линейное время (2)

Восстановим T по bwt(T) и σ^{-1}:

- Номер i нужной строки — это позиция # в bwt(T)
- Первый символ i-ой строки = i-й символ первого столбца = $\sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.
  $\Rightarrow$ $T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Пример: bwt(T) = а#ннБннБаааа, σ^{-1} = 2 5 8 1 9 10 11 12 3 4 6 7

$$\sigma^{-1}(7) = 11 \quad \mathrm{bwt}(T)[11] = \text{а}$$

T= Б а н Б а н а . . . . #

Восстановление текста за линейное время (2)

Восстановим T по $\mathrm{bwt}(T)$ и σ^{-1}:

- Номер i нужной строки — это позиция # в $\mathrm{bwt}(T)$
- Первый символ i-ой строки = i-й символ первого столбца = $\sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.
  $\Rightarrow$ $T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Пример: $\mathrm{bwt}(T) =$ а#ннБннБаааа, $\sigma^{-1} = 2\ 5\ 8\ 1\ 9\ 10\ 11\ 12\ 3\ 4\ 6\ 7$

$$\sigma^{-1}(11) = 6 \quad \mathrm{bwt}(T)[6] = \text{н}$$

T= Б а н Б а н а н . . . #

Восстановление текста за линейное время (2)

Восстановим T по $\mathrm{bwt}(T)$ и σ^{-1}:

- Номер i нужной строки — это позиция # в $\mathrm{bwt}(T)$
- Первый символ i-ой строки = i-й символ первого столбца = $\sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.
  $\Rightarrow$ $T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Пример: $\mathrm{bwt}(T) = \mathrm{а\#ннБннБаааа}$, $\sigma^{-1} = 2\,5\,8\,1\,9\,10\,11\,12\,3\,4\,6\,7$

$$\sigma^{-1}(6) = 10 \quad \mathrm{bwt}(T)[10] = \mathrm{а}$$

T= Б а н Б а н а н а . . #

Восстановление текста за линейное время (2)

Восстановим T по $\mathrm{bwt}(T)$ и σ^{-1}:

- Номер i нужной строки — это позиция # в $\mathrm{bwt}(T)$
- Первый символ i-ой строки = i-й символ первого столбца = $\sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.
  $\Rightarrow$ $T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Пример: $\mathrm{bwt}(T) =$ а#ннБннБаааа, $\sigma^{-1} = 2\ 5\ 8\ 1\ 9\ 10\ 11\ 12\ 3\ 4\ 6\ 7$

$$\sigma^{-1}(10) = 4 \quad \mathrm{bwt}(T)[4] = \text{н}$$

T= Б а н Б а н а н а н . #

Восстановление текста за линейное время (2)

Восстановим T по bwt(T) и σ^{-1}:

- Номер i нужной строки — это позиция # в bwt(T)
- Первый символ i-ой строки = i-й символ первого столбца = $\sigma^{-1}(i)$-й символ последнего столбца
  $\Rightarrow$ $T[1] = \mathrm{bwt}(T)[\sigma^{-1}(i)]$
- Второй символ i-ой строки следует за первым

$\Rightarrow$ он является первым в той строке, в которой первый является последним (т.е. в $\sigma^{-1}(i)$-ой), а последним — в $\sigma^{-1}(\sigma^{-1}(i))$-ой (т.е. в $\sigma^{-2}(i)$-ой)
  $\Rightarrow$ $T[2] = \mathrm{bwt}(T)[\sigma^{-2}(i)]$

- Аналогично, третий символ i-ой строки является первым в строке, где последним является второй, и т.д.
  $\Rightarrow$ $T[k] = \mathrm{bwt}(T)[\sigma^{-k}(i)], k = 1, \ldots, n$

Пример: bwt(T) = а#ннБннБаааа, σ^{-1} = 2 5 8 1 9 10 11 12 3 4 6 7

$$\sigma^{-1}(4) = 1 \quad \mathrm{bwt}(T)[1] = \text{а}$$

T= Б а н Б а н а н а н а #

- Легко посчитать σ^{-1} за время $O(n)$: надо посчитать массив частот символов, для каждого символа x вычислить диапазон в упорядоченном списке и сделать один проход по bwt(T)
  ★ значит, текст восстанавливается по bwt за время $O(n)$

Построение bwt(T)

Для построения bwt(T) нужно отсортировать лексикографически все циклические сдвиги T

- ★ Матрицу $M(T)$ строить не нужно, достаточно получить массив S, в котором любой элемент $S[i]$ равен номеру строки в $M(T)$, занимаемой циклическим сдвигом $T_i = T[i..n]T[1..i{-}1]$
  - тогда $\mathrm{bwt}[S[i]] = T[i{-}1]$ (последний символ строки T_i)

Для построения bwt(T) нужно отсортировать лексикографически все циклические сдвиги T

- ★ Матрицу $M(T)$ строить не нужно, достаточно получить массив S, в котором любой злемент $S[i]$ равен номеру строки в $M(T)$, занимаемой циклическим сдвигом $T_i = T[i..n]T[1..i{-}1]$
  - тогда $\mathrm{bwt}[S[i]] = T[i{-}1]$ (последний символ строки T_i)
- ★ Наличие спецсимвола конца строки упрощает задачу сортировки:
  - считаем, что спецсимвол (мы его обозначаем через #) меньше любого символа
  - результат сравнения T_i и T_j будет таким же, как для $T[i..n]$ и $T[j..n]$
  - ⋆ вместо циклических сдвигов достаточно отсортировать суффиксы текста

Для построения bwt(T) нужно отсортировать лексикографически все циклические сдвиги T

- ★ Матрицу $M(T)$ строить не нужно, достаточно получить массив S, в котором любой элемент $S[i]$ равен номеру строки в $M(T)$, занимаемой циклическим сдвигом $T_i = T[i..n]T[1..i{-}1]$
  - тогда bwt$[S[i]] = T[i{-}1]$ (последний символ строки T_i)
- ★ Наличие спецсимвола конца строки упрощает задачу сортировки:
  - считаем, что спецсимвол (мы его обозначаем через #) меньше любого символа
  - результат сравнения T_i и T_j будет таким же, как для $T[i..n]$ и $T[j..n]$
  - ★ вместо циклических сдвигов достаточно отсортировать суффиксы текста

- Итак, S — массив, в котором $S[i]$ равен номеру суффикса $T[i..n]$ в отсортированном списке всех суффиксов строки T
  - такой массив называется суффиксным массивом строки T
- Суффиксные массивы известны с 1990-х, это часто используемая в разных задачах структура данных, заслуживающая отдельной лекции
- Здесь только упомянем, что суффиксный массив можно построить за время $O(n)$, хотя до сих пор используют и более простые алгоритмы с временем работы $O(n \log n)$
  - строить суффиксный массив умеют многие специализированные библиотеки

Способы сжатия результата BWT

В предположении, что символы в тексте значительно зависят от своих контекстов,

- результат BWT состоит из большого количества кусков:
  - каждый кусок достаточно однороден (часто в нем вообще повторяется единственный символ)
  - между кусками — резкие переходы (смена контекста)
  - куски сжимаются хорошо, переходы ухудшают сжатие

Способы сжатия результата BWT

В предположении, что символы в тексте значительно зависят от своих контекстов,

- результат BWT состоит из большого количества кусков:
  - каждый кусок достаточно однороден (часто в нем вообще повторяется единственный символ)
  - между кусками — резкие переходы (смена контекста)
  - куски сжимаются хорошо, переходы ухудшают сжатие

★ Имеет смысл кастомизировать порядок символов в алфавите, чтобы уменьшить число резких переходов

  - для текста на естественном языке символы разбивают на три группы (гласные, согласные, небуквенные символы), а внутри каждой группы упорядочивают в соответствии с особенностями языка

В предположении, что символы в тексте значительно зависят от своих контекстов,

- результат BWT состоит из большого количества кусков:
  - каждый кусок достаточно однороден (часто в нем вообще повторяется единственный символ)
  - между кусками — резкие переходы (смена контекста)
  - куски сжимаются хорошо, переходы ухудшают сжатие
- ★ Имеет смысл кастомизировать порядок символов в алфавите, чтобы уменьшить число резких переходов
  - для текста на естественном языке символы разбивают на три группы (гласные, согласные, небуквенные символы), а внутри каждой группы упорядочивают в соответствии с особенностями языка

Способы сжатия:

- MTF+ZLE+ARI:
  - Вначале применяется еще одно обратимое несжимающее преобразование, которое преобразует последовательности равных символов в последовательности нулей и "перекашивает" статистику остальных символов;
  - На втором этапе последовательности нулей заменяются их длинами
  - На третьем применяется арифметическое кодирование в алфавите байтов
- DC:
  - Кодируются расстояния между последовательными вхождениями одного символа
  - Сжатие в основном достигается за счет того, что последовательности одинаковых символов... не требуют кодирования вообще
- Про другие способы сжатия (и про названные тоже) можно прочитать, например, в статье S. Deorowicz. Second step algorithms in the Burrows–Wheeler compression algorithm. Software—Practice and Experience, 2002

Метод стопки книг (MTF)

Преобразование Move-To-Front (MTF) читает входную строку слева направо и посимвольно меняет ее по следующим правилам:

- Алфавит упорядочен, все символы адресуются своими номерами в списке (первый элемент имеет номер 0)
- Очередной символ a заменяется на номер a в текущем списке, после чего
  - a перемещается в начало списка (получает номер 0)
  - у каждого символа, номер которого был меньше номера a, номер увеличивается на 1

⋆ Отсюда образ стопки книг: книгу a достали из стопки и положили наверх

Метод стопки книг (MTF)

Преобразование Move-To-Front (MTF) читает входную строку слева направо и посимвольно меняет ее по следующим правилам:

- Алфавит упорядочен, все символы адресуются своими номерами в списке (первый элемент имеет номер 0)
- Очередной символ a заменяется на номер a в текущем списке, после чего
  - a перемещается в начало списка (получает номер 0)
  - у каждого символа, номер которого был меньше номера a, номер увеличивается на 1

⋆ Отсюда образ стопки книг: книгу a достали из стопки и положили наверх

★ Если на вход преобразования MTF подать bwt(T), получим два эффекта, полезных для сжатия:

- основной: последовательности одинаковых символов заменяются на последовательности нулей
- дополнительный: маленькие номера встречаются часто, а большие — очень редко (часто встречающиеся символы будут находиться близко к началу списка)

Метод стопки книг (MTF)

Преобразование Move-To-Front (MTF) читает входную строку слева направо и посимвольно меняет ее по следующим правилам:

- Алфавит упорядочен, все символы адресуются своими номерами в списке (первый элемент имеет номер 0)
- Очередной символ a заменяется на номер a в текущем списке, после чего
  - a перемещается в начало списка (получает номер 0)
  - у каждого символа, номер которого был меньше номера a, номер увеличивается на 1

⋆ Отсюда образ стопки книг: книгу a достали из стопки и положили наверх

★ Если на вход преобразования MTF подать bwt(T), получим два эффекта, полезных для сжатия:

- основной: последовательности одинаковых символов заменяются на последовательности нулей
- дополнительный: маленькие номера встречаются часто, а большие — очень редко (часто встречающиеся символы будут находиться близко к началу списка)

Пример:

$$
\begin{array}{rcccccccccccc}
w = & \text{а} & \# & \text{н} & \text{н} & \text{Б} & \text{н} & \text{н} & \text{Б} & \text{а} & \text{а} & \text{а} & \text{а} \\
\mathrm{mtf}(w) = & 2 & 1 & 3 & 0 & 3 & 1 & 0 & 1 & 3 & 0 & 0 & 0 \\
\text{алфавит}: & \begin{bmatrix} \# \\ \text{Б} \\ \text{а} \\ \text{н} \end{bmatrix} & \begin{bmatrix} \text{а} \\ \# \\ \text{Б} \\ \text{н} \end{bmatrix} & \begin{bmatrix} \# \\ \text{а} \\ \text{Б} \\ \text{н} \end{bmatrix} & \begin{bmatrix} \text{н} \\ \# \\ \text{а} \\ \text{Б} \end{bmatrix} & \begin{bmatrix} \text{н} \\ \# \\ \text{а} \\ \text{Б} \end{bmatrix} & \begin{bmatrix} \text{Б} \\ \text{н} \\ \# \\ \text{а} \end{bmatrix} & \begin{bmatrix} \text{н} \\ \text{Б} \\ \# \\ \text{а} \end{bmatrix} & \begin{bmatrix} \text{н} \\ \text{Б} \\ \# \\ \text{а} \end{bmatrix} & \begin{bmatrix} \text{Б} \\ \text{н} \\ \# \\ \text{а} \end{bmatrix} & \begin{bmatrix} \text{а} \\ \text{Б} \\ \text{н} \\ \# \end{bmatrix} & \begin{bmatrix} \text{а} \\ \text{Б} \\ \text{н} \\ \# \end{bmatrix} & \begin{bmatrix} \text{а} \\ \text{Б} \\ \text{н} \\ \# \end{bmatrix}
\end{array}
$$

Кодирование нулей (ZLE)

- ★ В тексте mtf(bwt(T)) есть длинные повторы нулей, соответствующие контекстным зависимостям в T, в то время как повторы других символов при MTF могут возникнуть только случайно (а значит, они короткие)

На первом этапе сжатия кодируются повторы нулей

- ★ Лучше всего работает метод Zero Length Encoding (ZLE); он хорошо подготавливает почву для арифметического кодирования на втором этапе

Кодирование нулей (ZLE)

★ В тексте mtf(bwt(T)) есть длинные повторы нулей, соответствующие контекстным зависимостям в T, в то время как повторы других символов при MTF могут возникнуть только случайно (а значит, они короткие)

На первом этапе сжатия кодируются повторы нулей

★ Лучше всего работает метод Zero Length Encoding (ZLE); он хорошо подготавливает почву для арифметического кодирования на втором этапе

ZLE:

- сделаем 254 и 255 — два самых редких символа в mtf(bwt(T)) — служебными
- каждую последовательность нулей заменим на ее длину, в качестве цифр 0 и 1 используя 254 и 255
- 254 и 255, если они встречались, заменяем на пару байт 0 254 (0 255)

⋆ дополнительный трюк: для последовательности из N нулей кодируем число N+1 без первой цифры
  - $N+1 \geqslant 2$: двоичная запись начинается с 1 и содержит еще хотя бы одну цифру
  - трюк позволяет в большинстве случаев сэкономить одну цифру по сравнению с двоичным кодом числа N

Кодирование нулей (ZLE)

★ В тексте mtf(bwt(T)) есть длинные повторы нулей, соответствующие контекстным зависимостям в T, в то время как повторы других символов при MTF могут возникнуть только случайно (а значит, они короткие)

На первом этапе сжатия кодируются повторы нулей

★ Лучше всего работает метод Zero Length Encoding (ZLE); он хорошо подготавливает почву для арифметического кодирования на втором этапе

ZLE:

- сделаем 254 и 255 — два самых редких символа в mtf(bwt(T)) — служебными
- каждую последовательность нулей заменим на ее длину, в качестве цифр 0 и 1 используя 254 и 255
- 254 и 255, если они встречались, заменяем на пару байт 0 254 (0 255)

★ дополнительный трюк: для последовательности из N нулей кодируем число $N{+}1$ без первой цифры
  - $N{+}1 \geqslant 2$: двоичная запись начинается с 1 и содержит еще хотя бы одну цифру
  - трюк позволяет в большинстве случаев сэкономить одну цифру по сравнению с двоичным кодом числа N

Пример: $\underbrace{0\cdots0}_{100\text{раз}}$ ZLE закодирует как 255 254 254 255 254 255, так как $101_2 = \underline{1}100101$

Кодирование расстояний (DC)

Альтернативный вариант сжатия после BWT — Distance Coding (DC)

- Текст сканируется слева направо, для каждого символа записывается ссылка на следующий такой же символ

Кодирование расстояний (DC)

Альтернативный вариант сжатия после BWT — Distance Coding (DC)

- Текст сканируется слева направо, для каждого символа записывается ссылка на следующий такой же символ

Более строго:

- Припишем слева к тексту $\mathrm{bwt}(T)$ алфавит A (обычно $|A| = 256$); пусть $X = A \cdot \mathrm{bwt}(T)$
  - декодер в этот момент знает все символы A и ни одного символа $\mathrm{bwt}(T)$
- Для каждого символа $a = X[i]$ слева направо записываем номер: каким по счету слева из неизвестных декодеру символов является следующий символ a
  - если вхождений a больше нет, пишем номер 0
  - символ a, на который мы сослались, стал известен декодеру
- ★ Если кодер передает номер для символа $X[i]$, а на символ $X[i+1]$ еще не было ссылки, то эта ссылка должна быть сделана сейчас, то есть $X[i+1] = X[i]$
  - декодер в состоянии сам определить эту ситуацию (ему неизвестен символ, следующий за текущим) и скопировать текущий символ в следующий, сделав следующий известным
  - ⇒ кодер ничего не кодирует в данной ситуации
  - ⋆ это правило — основной источник сжатия
- Кодирование завершается, когда декодер знает все символы X
- Номер кодируется с переполнением (см. Лекцию 6), например, по схеме 8+16+24 бит или 8+16+16+...
- Пример ⟹

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\mathsf{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 1

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	?															
декодер	#	Б	а	н	.	.	.	.	.	.	.	.	.	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 1

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2															
декодер	#	Б	а	н	.	.	.	.	.	.	.	.	.	.	.	.

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 2

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	?														
декодер	#	Б	а	н	.	#	.	.	.	.	.	.	.	.	.	.

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 2

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4														
декодер	#	Б	а	н	.	#	.	.	.	.	.	.	.	.	.	.

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 3

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	?													
декодер	#	Б	а	н	.	#	.	.	Б	.	.	.	.	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\text{bwt}(T) = $ а#ннБннБаааа
- Алфавит {#, Б, а, н}
- $X =$ #Бана#ннБннБаааа
- Символы, неизвестные декодеру, обозначены точками

Итерация 3

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1													
декодер	#	Б	а	н	.	#	.	.	Б	.	.	.	.	.	.	.

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 4

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	?												
декодер	#	Б	а	н	а	#	.	.	Б	.	.	.	.	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 4

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1												
декодер	#	Б	а	н	а	#	.	.	Б	.	.	.	.	.	.	.

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 5

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	?											
декодер	#	Б	а	н	а	#	н	.	Б	.	.	.	.	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 5

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5											
декодер	#	Б	а	н	а	#	н	.	Б	.	.	.	.	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 6

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0										
декодер	#	Б	а	н	а	#	н	.	Б	.	.	.	а	.	.	.

Рассмотрим наш сквозной пример:

- Текст $\text{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 7

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0										
декодер	#	Б	а	н	а	#	н	.	Б	.	.	.	а	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 7

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0										
декодер	#	Б	а	н	а	#	н	н	Б	.	.	.	а	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\text{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 8

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0		?								
декодер	#	Б	а	н	а	#	н	н	Б	.	.	.	а	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 8

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0		1								
декодер	#	Б	а	н	а	#	н	н	Б	.	.	.	а	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 9

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0		1	?							
декодер	#	Б	а	н	а	#	н	н	Б	н	.	.	а	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 9

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0		1	2							
декодер	#	Б	а	н	а	#	н	н	Б	н	.	.	а	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\text{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 10

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0		1	2							
декодер	#	Б	а	н	а	#	н	н	Б	н	.	Б	а	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\text{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 10

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0		1	2							
декодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	.	.	.

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 11

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0		1	2		0					
декодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 12

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0		1	2		0	0				
декодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	.	.	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 13

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0		1	2		0	0				
декодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	.	.

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 14

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0		1	2		0	0				
декодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	.

DC — пример

Рассмотрим наш сквозной пример:

- Текст $\mathrm{bwt}(T) = \text{а\#ннБннБаааа}$
- Алфавит $\{\#, \text{Б}, \text{а}, \text{н}\}$
- $X = \text{\#Бана\#ннБннБаааа}$
- Символы, неизвестные декодеру, обозначены точками

Итерация 15

кодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а
	2	4	1	1	5	0		1	2		0	0				
декодер	#	Б	а	н	а	#	н	н	Б	н	н	Б	а	а	а	а

$\mathrm{dc}(\mathrm{bwt}(T)) = 2\,4\,1\,1\,5\,0\,1\,2\,0\,0$

★ Результат DC можно еще немного сжать Хаффманом или арифметикой

Общая характеристика алгоритмов BWT-сжатия

- ★ Изначально BWT-алгоритмы позиционировались как “золотая середина” между быстрыми, но слабыми словарными алгоритмами и мощным, но медленным PPM
- ★ На самом деле, они лучше, чем среднее арифметическое LZ и PPM
- Самый известный архиватор общего назначения, основанный на BWT — bzip2 — сжимает многократно упоминавшийся датасет английской википедии в 3.9 раза, а, например, архиватор M03 — в 6.1 раза, обгоняя 7-Zip и WinRar на их максимальных настройках и по коэффициенту сжатия, и по скорости

Общая характеристика алгоритмов BWT-сжатия

- ★ Изначально BWT-алгоритмы позиционировались как “золотая середина” между быстрыми, но слабыми словарными алгоритмами и мощным, но медленным PPM
- ★ На самом деле, они лучше, чем среднее арифметическое LZ и PPM
- Самый известный архиватор общего назначения, основанный на BWT — bzip2 — сжимает многократно упоминавшийся датасет английской википедии в 3.9 раза, а, например, архиватор M03 — в 6.1 раза, обгоняя 7-Zip и WinRar на их максимальных настройках и по коэффициенту сжатия, и по скорости

- BWT имеет и другие приложения; самое известное — индексирование данных (FM-индекс и его производные)
- ... ну и это просто математически красиво!