Алгоритмы и структуры данных

Преобразование Барроуза-Уилера

CS Center, Новосибирск

Преобразование BW

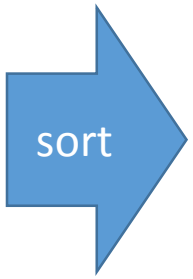
$\alpha \in \Sigma^*$

- Условие:
  - $\alpha[\ |\alpha - 1|\ ] = \$$
- Циклические сдвиги:
  - $\alpha_{shift}[\ i\ ] := \alpha[\ i :)\alpha[\ : i\ )$
- Отсортируем все $\alpha_{shift}[\ i\ ]$
  - Пусть $\alpha_{shift}[\ s[\ j\ ]\ ]$ – это j-ый α_{shift} в порядке сортировки
  - Тогда BWT(α) – это строка β, $|\beta| = |\alpha|$, $\beta[\ j\ ] = \alpha_{shift}[\ s[\ j\ ]\ ][\ |\alpha - 1|\ ]$

Преобразование BW: пример

α=abacaba$

a b a c a b a $
b a c a b a $ a
a c a b a $ a b
c a b a $ a b a
a b a $ a b a c
b a $ a b a c a
a $ a b a c a b
$ a b a c a b a



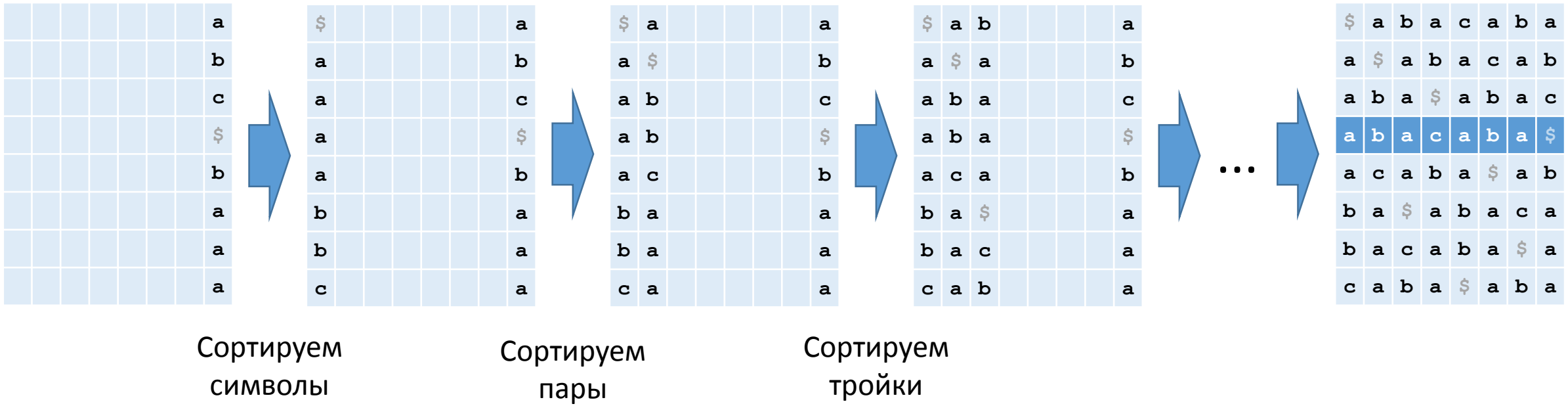
$ a b a c a b a
a $ a b a c a b
a b a $ a b a c
a b a c a b a $
a c a b a $ a b
b a $ a b a c a
b a c a b a $ a
c a b a $ a b a

i	SA[i]	SA[i]-1	α[SA[i]-1]
0	7	6	a
1	6	5	b
2	4	3	c
3	0	7	$
4	2	1	b
5	5	4	a
6	1	0	a
7	3	2	a

BWT(α)=abc$baaa

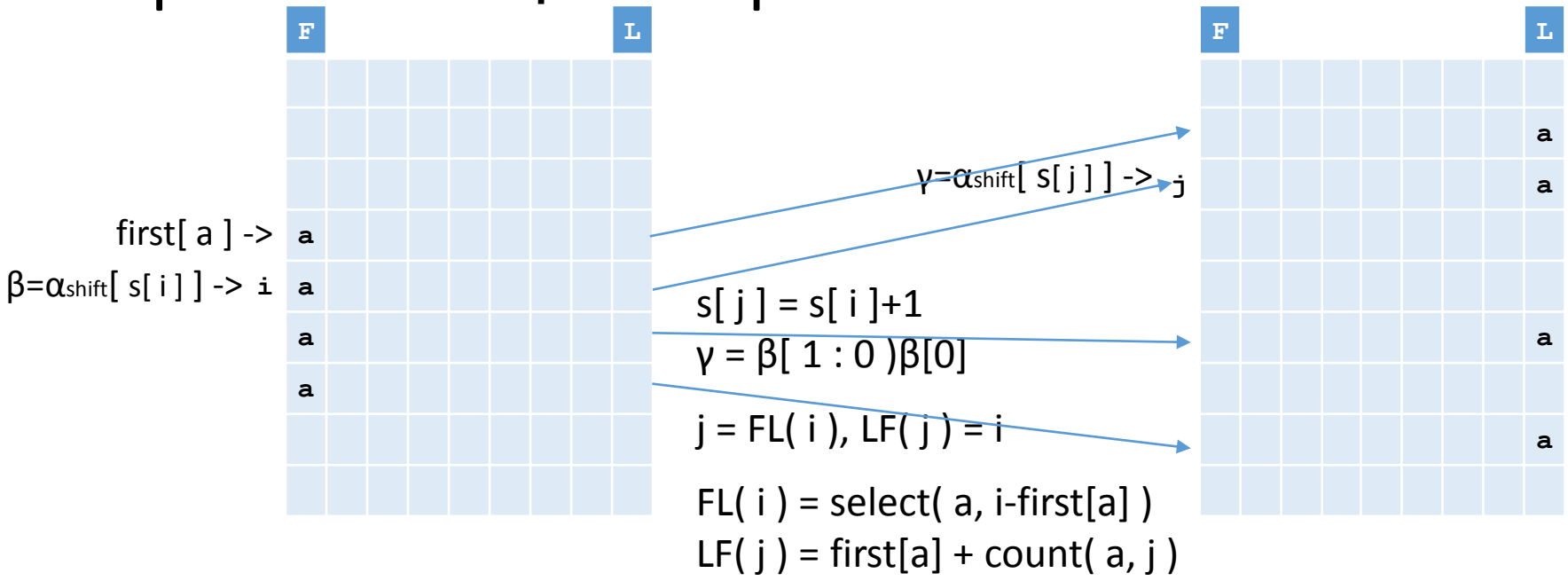
- BWT(α)[i] = α[SA[i] - 1]
  - Построение за O(|α|)

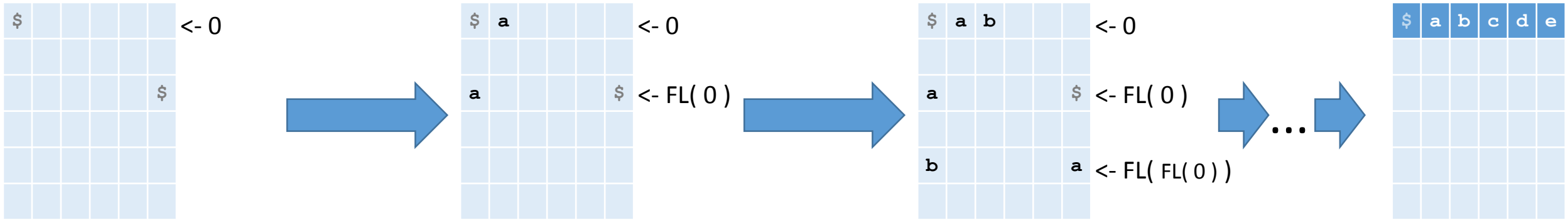
Обратное преобразование BW



- Неэффективный алгоритм

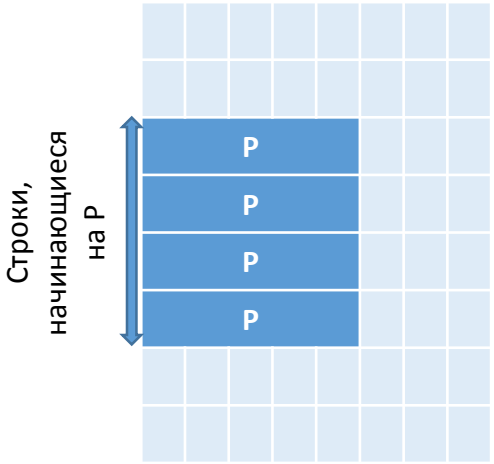
Обратное преобразование BW за O(|α|)



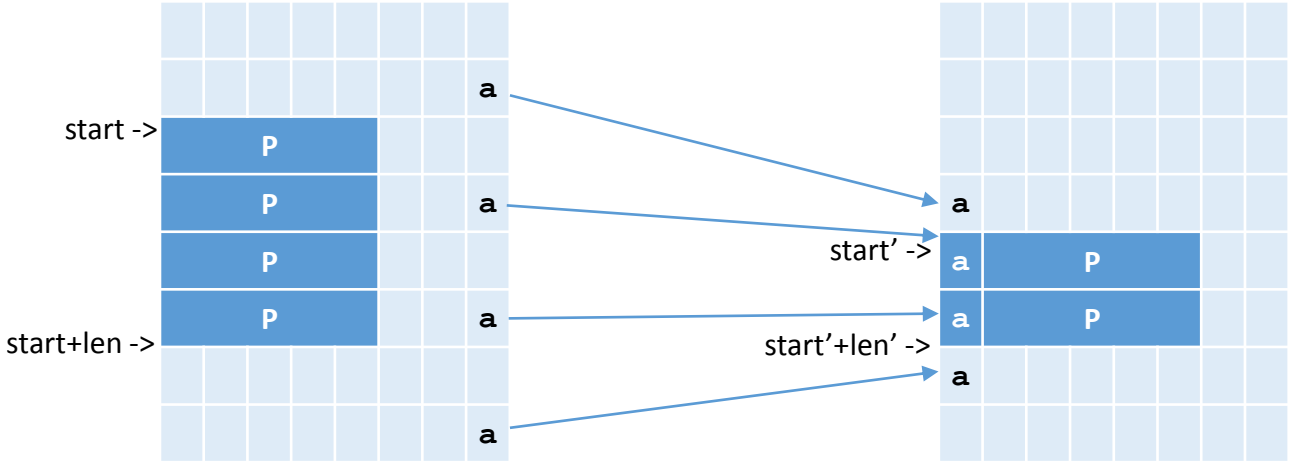


Преобразование BW для поиска шаблона

- Ищем образец P посимвольно, читая его *<u>справа налево</u>*
  - на каждом шаге поддерживаем диапазон строк, начинающихся на P
    - диапазон [start : start + len)



Переход от P к P’=aP



- start’ = first[a] + count(a, start)
- len’ = count(a, start+len) - count(a, start)

Реализация count

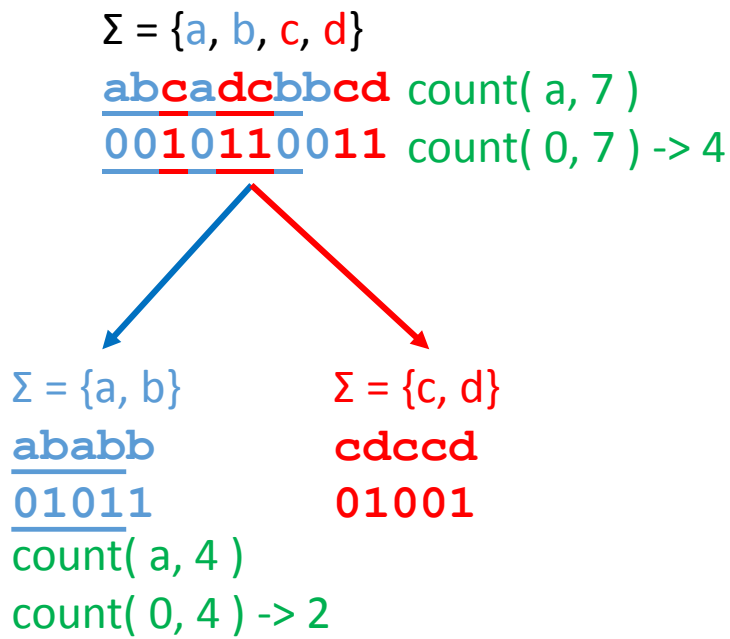
- Дана строка $\alpha \in \Sigma^*$, $|\alpha|=N$
  - Запрос: count(a, i) -> сколько раз «a» встречается в α[: i)
- Случай $\Sigma = \{ 0, 1 \}$
  - Достаточно считать count(1, i) // count(0, i) = i - count(1, i)
    - Можно хранить все значения count(1, i), $0 \le i < N$
      - Неэффективное использование памяти:
        - Строка N бит
        - Массив NlogN бит
    - Режем битовую строку на блоки длины w = размер машинного слова
    - Храним только значения count(1, j*w)
      - Строка и массив по N бит
    - Вычисляем count(1, i) = count(1, (i / w) * w) + «count1 в префиксе последнего блока»
      - Используем popcount для подсчёта числа единиц в машинном слове

Реализация count

- Дана строка $\alpha \in \{ 0, 1 \}^*$, $|\alpha|=N$
  - Запрос: count1(i) -> сколько раз «1» встречается в $\alpha[: i)$
- Режем битовую строку на блоки длины $w = \log N$
- Имеем суперблоки длины $B = w*k$
  - Храним значения count1(j * B)
    - на границах суперблоков
      - $N/B * \log N$ бит
  - Храним значения count1(j * w) – count1((j / k) * B)
    - на границах блоков, отсчитано от начало суперблока
      - $N/\log N * \log B$ бит
  - Суммарно $N\log N/B + N\log B/\log N$ бит
    - $B = \log N*\log(\log N)$:
      - $O(N/\log(\log N) + N\log(\log N)/\log N) = o(N)$ бит

Реализация count: wavelet tree

- Разбиваем алфавит на две части $\Sigma = \Sigma_0 \cup \Sigma_1$
  - $a \in \Sigma_i \Rightarrow m_\Sigma[a] = i$
- Формируем $mask_\alpha \in \{0, 1\}^*$
  - $|mask_\alpha| = |\alpha|$, $mask_\alpha[i] = m_\Sigma[\alpha[i]]$
- Формируем две строки $\alpha_i = filter(\alpha, \Sigma_i)$
- Продолжаем рекурсивно пока $|\Sigma| > 2$
  - Высота дерева $\log|\Sigma|$



- $count_{\alpha\Sigma}(a, i) = count_{\alpha_{m_\Sigma[a]}\Sigma_{m_\Sigma[a]}}(a, count_{mask_\alpha\{0, 1\}}(m_\Sigma[a], i))$
  - $O(\log|\Sigma|)$ времени
  - $O(N\log|\Sigma|)$ памяти