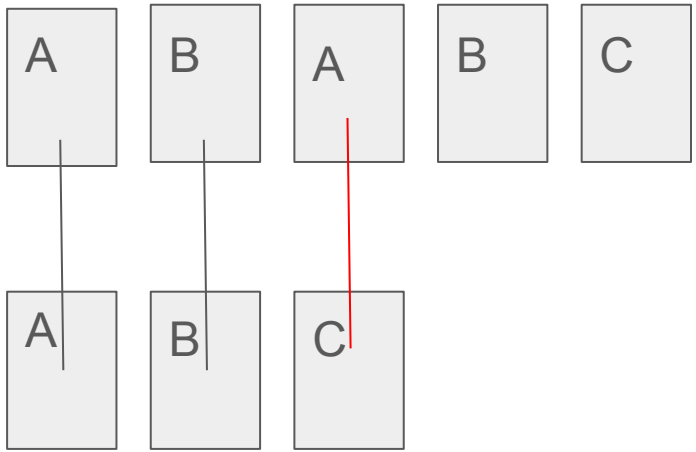
Алгоритмы на строках

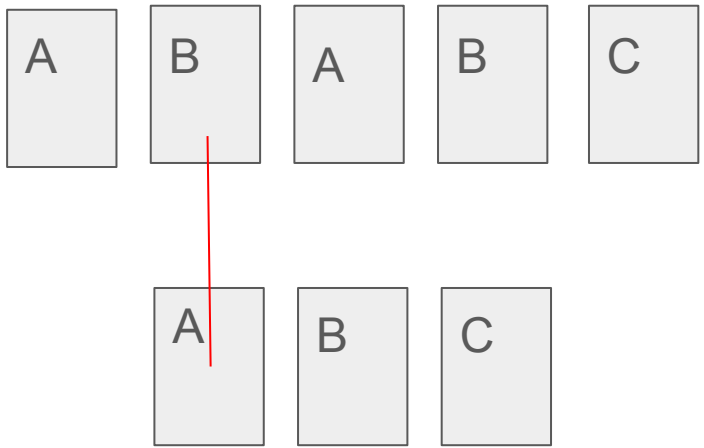
Формулировка задачи

СТРОКА

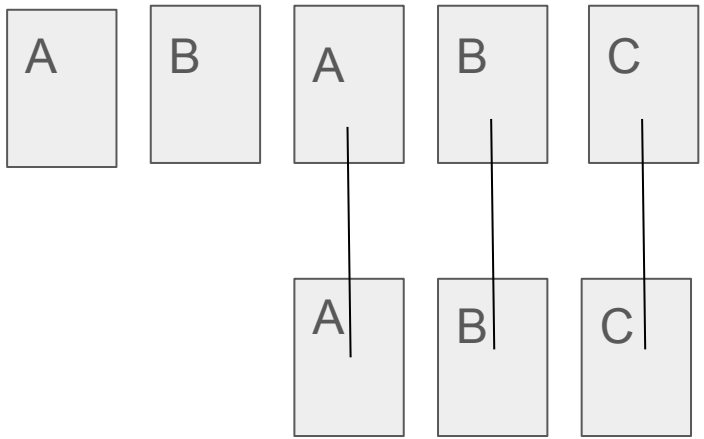
ПАТТЕРН



Формулировка задачи



Формулировка задачи



Проблемы

ДОЛГО

АААААААААААААААААААААААААААА

АААААААХ

Алгоритм Рабина - Карпа

- Можно сократить время на переводе строк в числа - не нужно сравнивать посимвольно за O(N)

Алгоритм КМП

Вспоминаем эту ситуацию

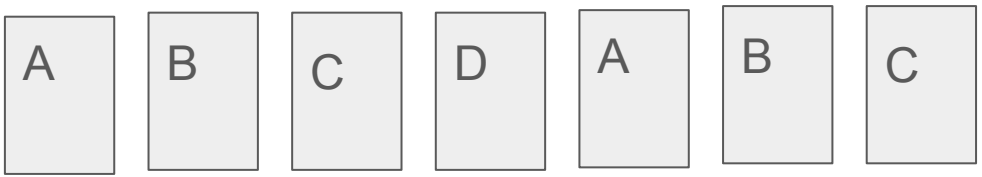
AAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAX

Что если можно было бы перепрыгивать повторяющиеся элементы?

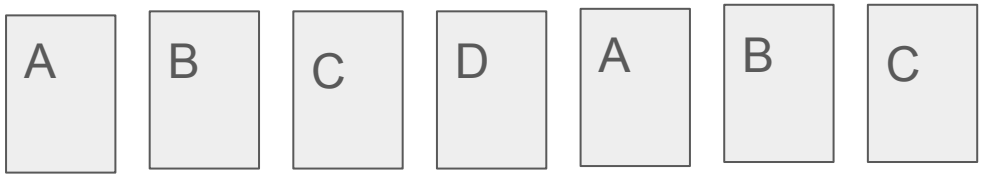
Суффикс - Префикс функция

АБСТРАГИРУЕМСЯ ОТ СТРОКИ, СЕЙЧАС РАБОТАЕМ ТОЛЬКО С ПАТТЕРНОМ.



Суффикс - Префикс функция

АБСТРАГИРУЕМСЯ ОТ СТРОКИ, СЕЙЧАС РАБОТАЕМ ТОЛЬКО С ПАТТЕРНОМ.



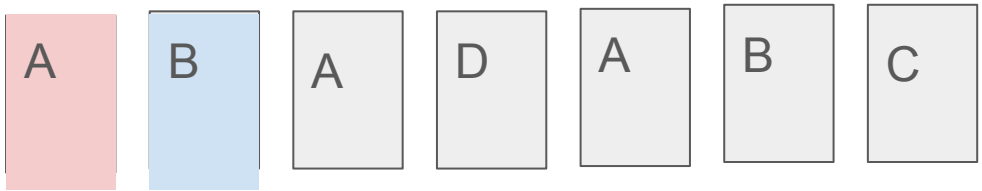
SP(0) = 0

Суффикс - Префикс функция

АБСТРАГИРУЕМСЯ ОТ СТРОКИ, СЕЙЧАС РАБОТАЕМ ТОЛЬКО С ПАТТЕРНОМ.

SP(0) = 0

SP(1) = 0



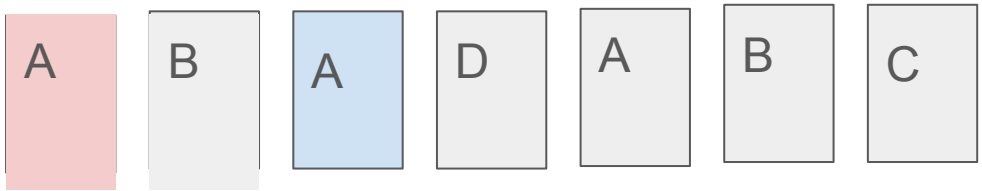
Суффикс - Префикс функция

АБСТРАГИРУЕМСЯ ОТ СТРОКИ, СЕЙЧАС РАБОТАЕМ ТОЛЬКО С ПАТТЕРНОМ.

SP(0) = 0

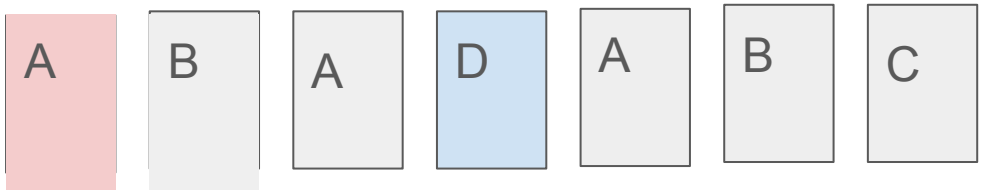
SP(1) = 0

SP(2) = 1



Суффикс - Префикс функция

АБСТРАГИРУЕМСЯ ОТ СТРОКИ, СЕЙЧАС РАБОТАЕМ ТОЛЬКО С ПАТТЕРНОМ.



SP(0) = 0

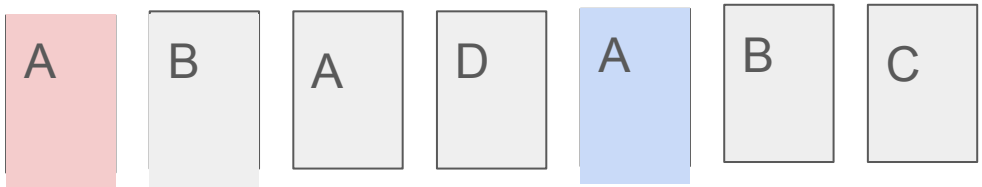
SP(1) = 0

SP(2) = 1

SP(3) = 0

Суффикс - Префикс функция

АБСТРАГИРУЕМСЯ ОТ СТРОКИ, СЕЙЧАС РАБОТАЕМ ТОЛЬКО С ПАТТЕРНОМ.



SP(0) = 0

SP(1) = 0

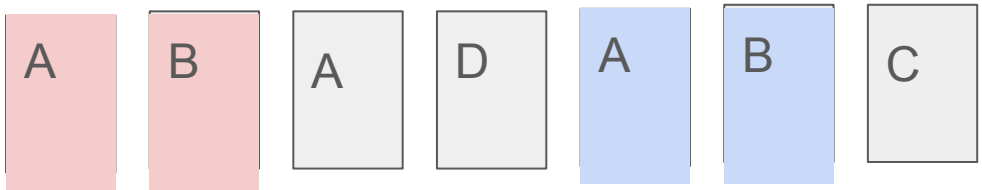
SP(2) = 1

SP(3) = 0

SP(4) = 1

Суффикс - Префикс функция

АБСТРАГИРУЕМСЯ ОТ СТРОКИ, СЕЙЧАС РАБОТАЕМ ТОЛЬКО С ПАТТЕРНОМ.



SP(0) = 0

SP(1) = 0

SP(2) = 1

SP(3) = 0

SP(4) = 1

SP(5) = 2

Суффикс - Префикс функция

АБСТРАГИРУЕМСЯ ОТ СТРОКИ, СЕЙЧАС РАБОТАЕМ ТОЛЬКО С ПАТТЕРНОМ.

SP(0) = 0

SP(1) = 0

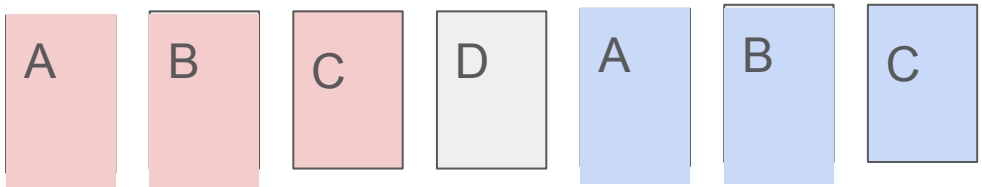
SP(2) = 1

SP(3) = 0

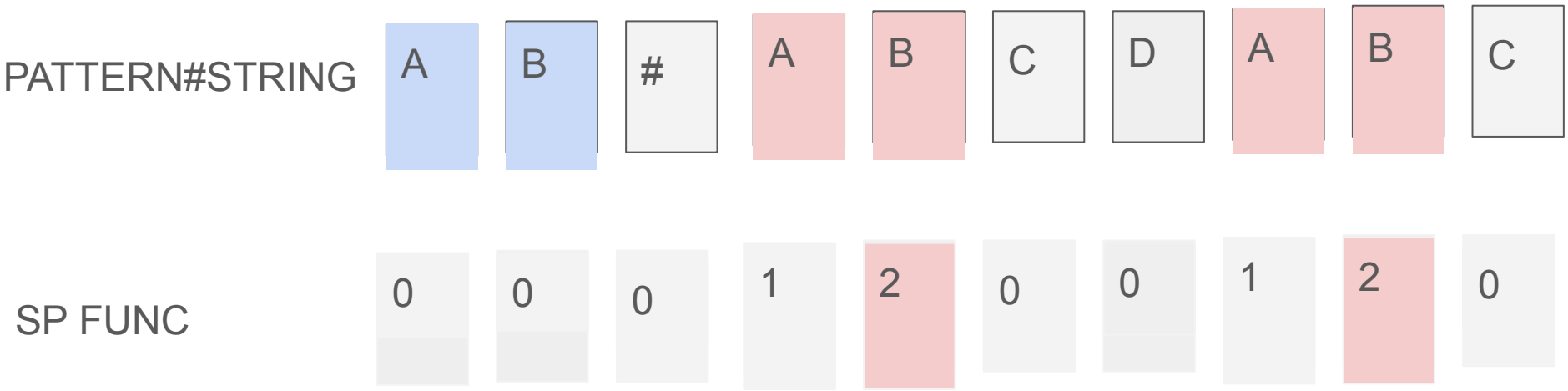
SP(4) = 1

SP(5) = 2

SP(6) = 3



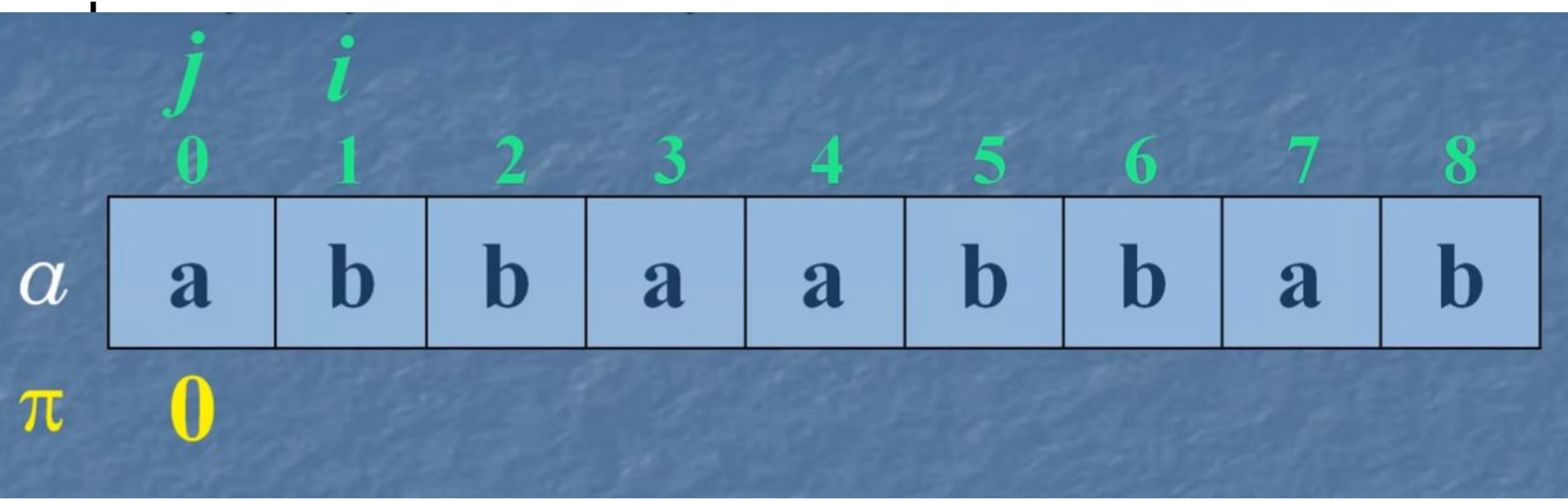
Кнут-Моррис-Пратт (наивная реализация)



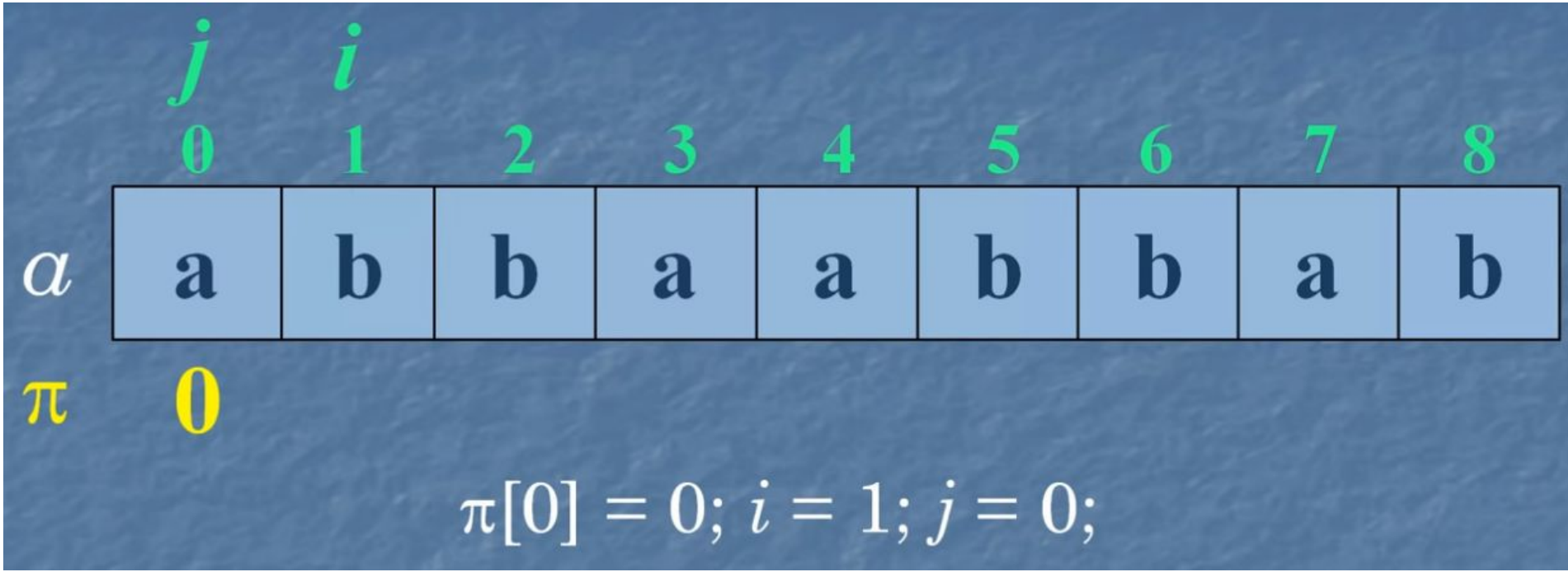
Таким образом, мы молодцы и, подсчетом суффикс-префикс функции, нашли все вхождения паттерна в строку.

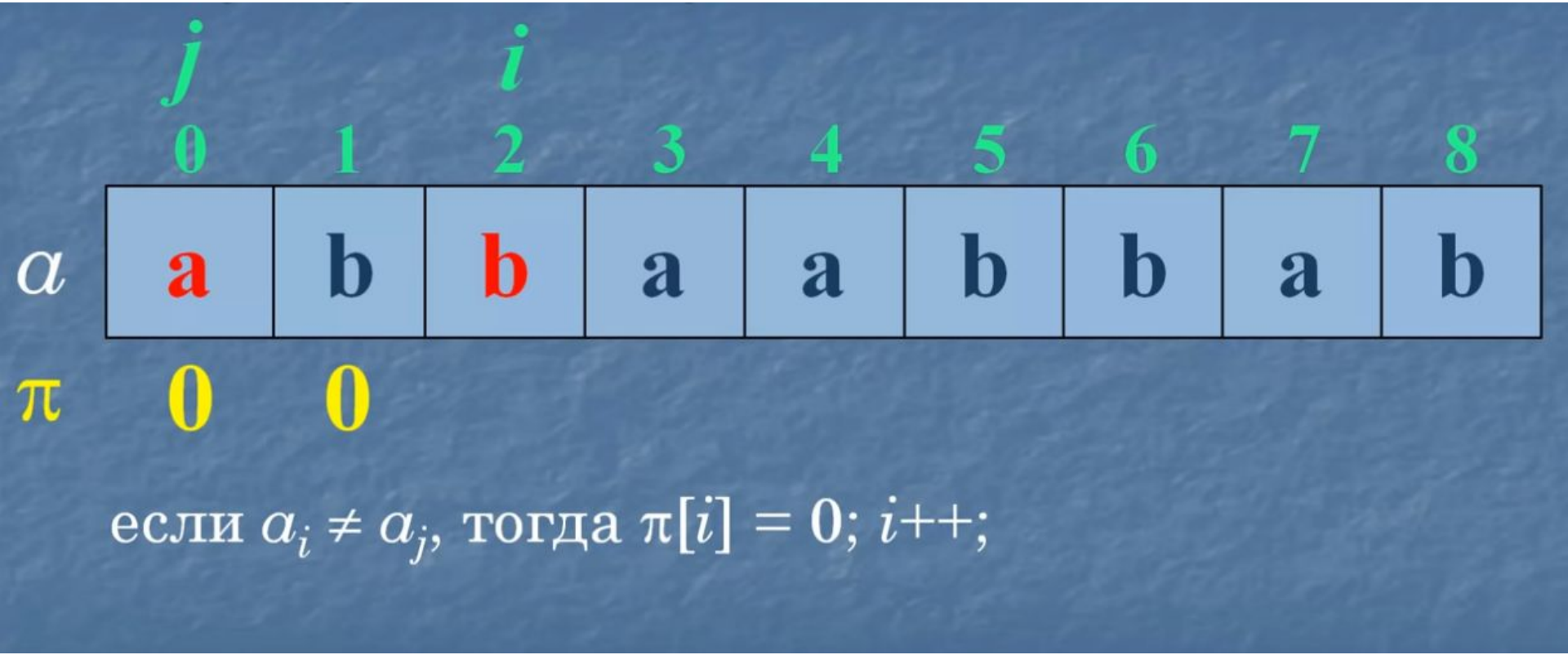
Но пока время работы занимает O((N + M + 1) ^ 3)

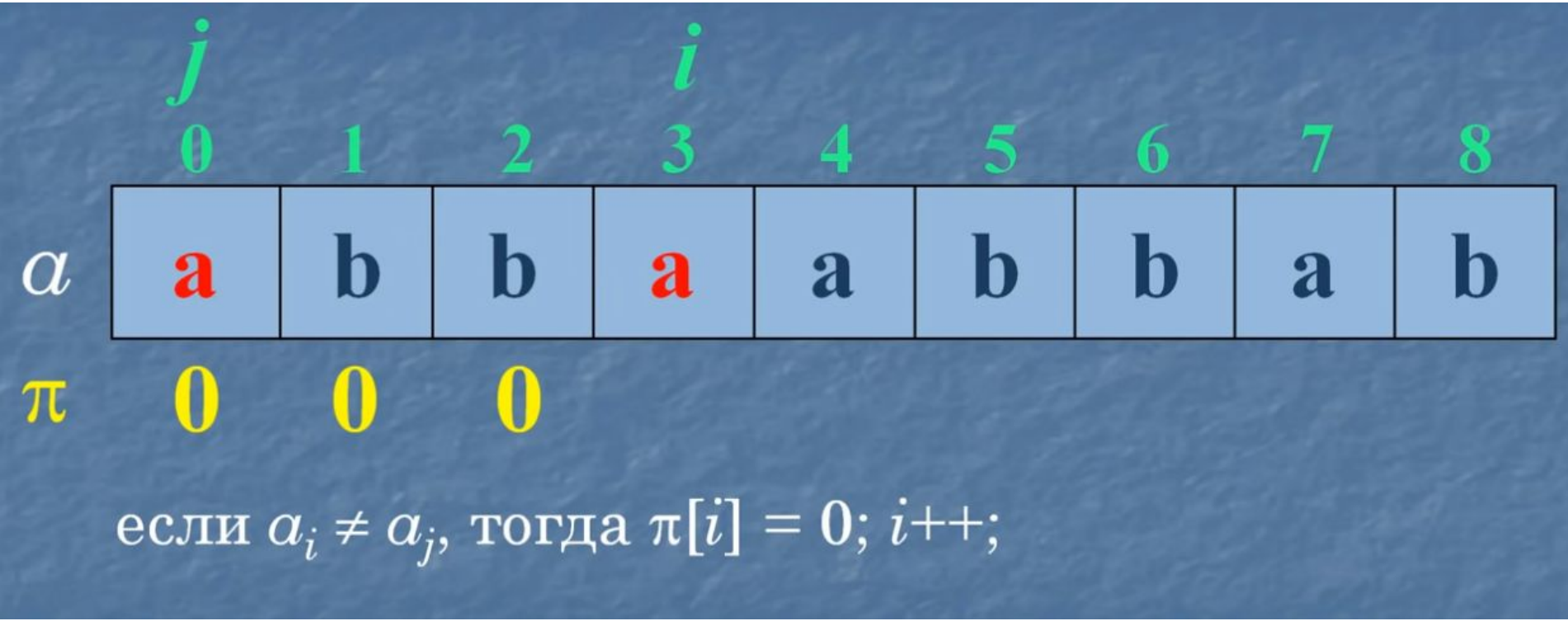
Быстрый вариант подсчета суффикс-префикс



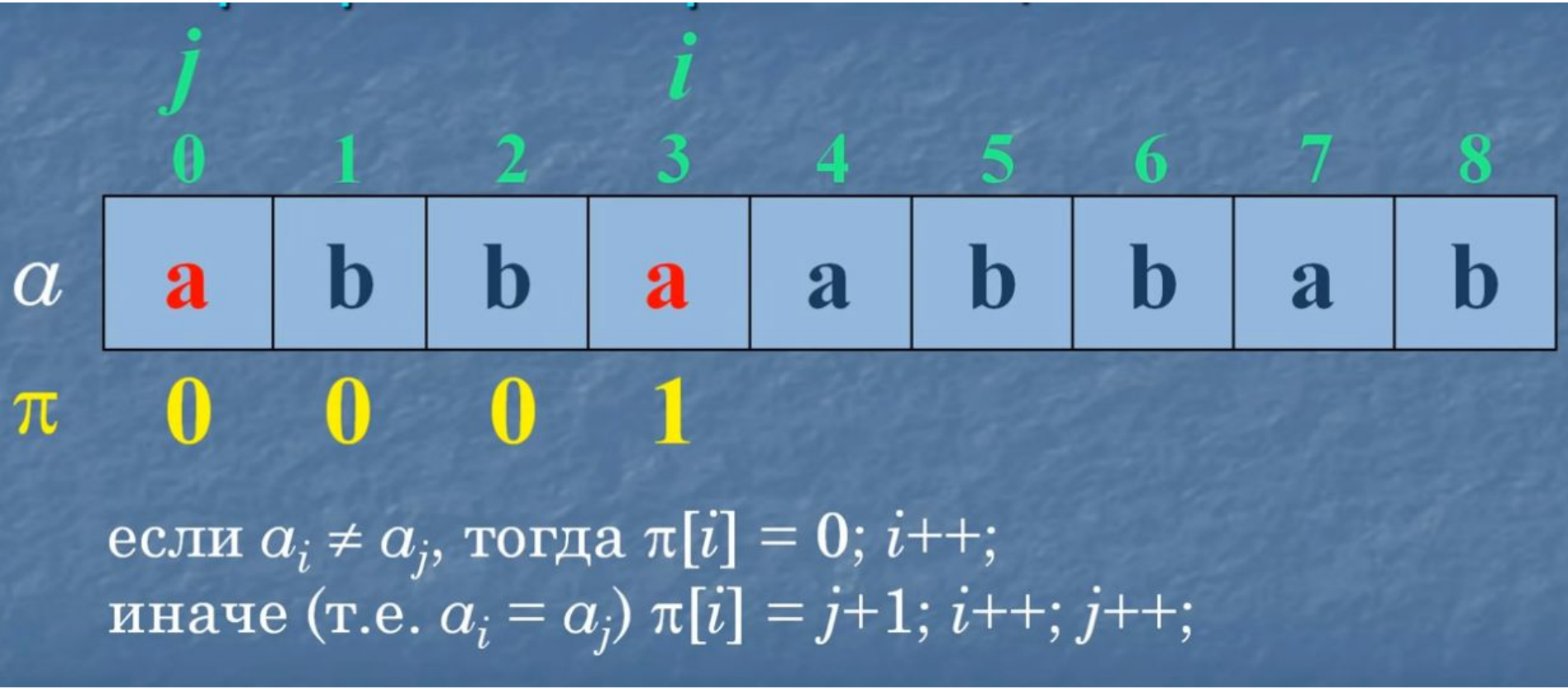
Источник: https://www.youtube.com/watch?time_continue=645&v=7g-WEBj3igk


j
i
0 1 2 3 4 5 6 7 8
α
a b b a a b b a b
π
0
π[0] = 0; i = 1; j = 0;

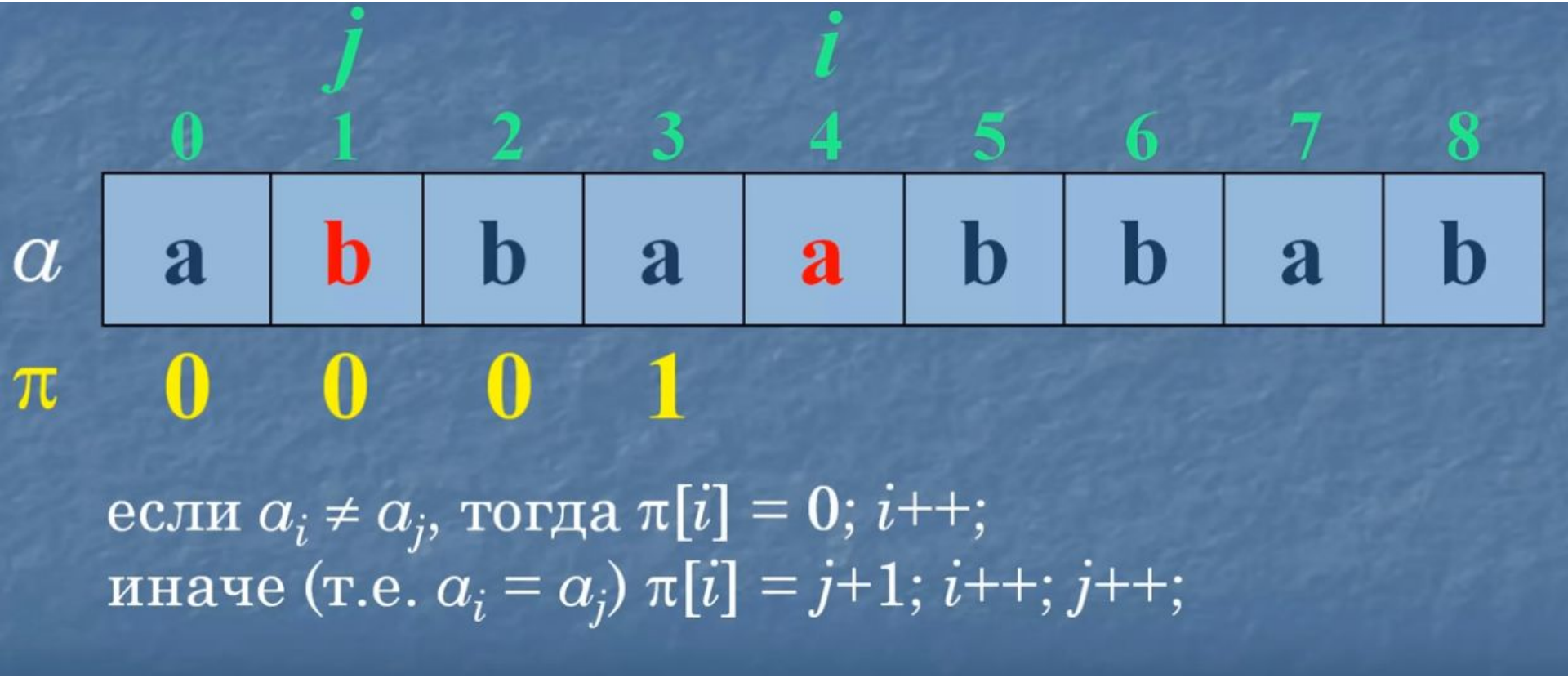

j
i
0 1 2 3 4 5 6 7 8
a
a b b a a b b a b
π
0 0
если $a_i \neq a_j$, тогда $\pi[i] = 0$; i++;

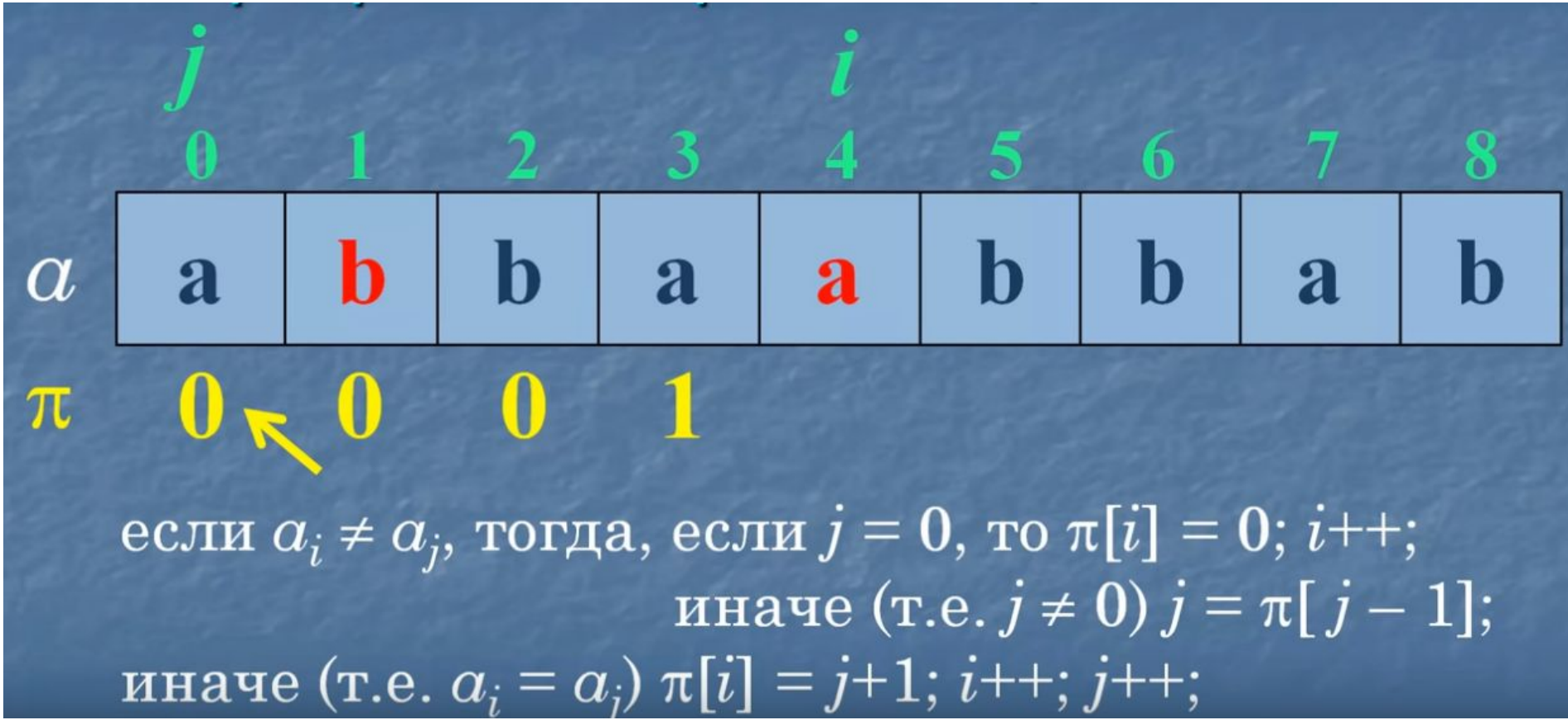


если $a_i \neq a_j$, тогда $\pi[i] = 0$; i++;



если $a_i \neq a_j$, тогда $\pi[i] = 0$; i++;
иначе (т.е. $a_i = a_j$) $\pi[i] = j+1$; i++; j++;

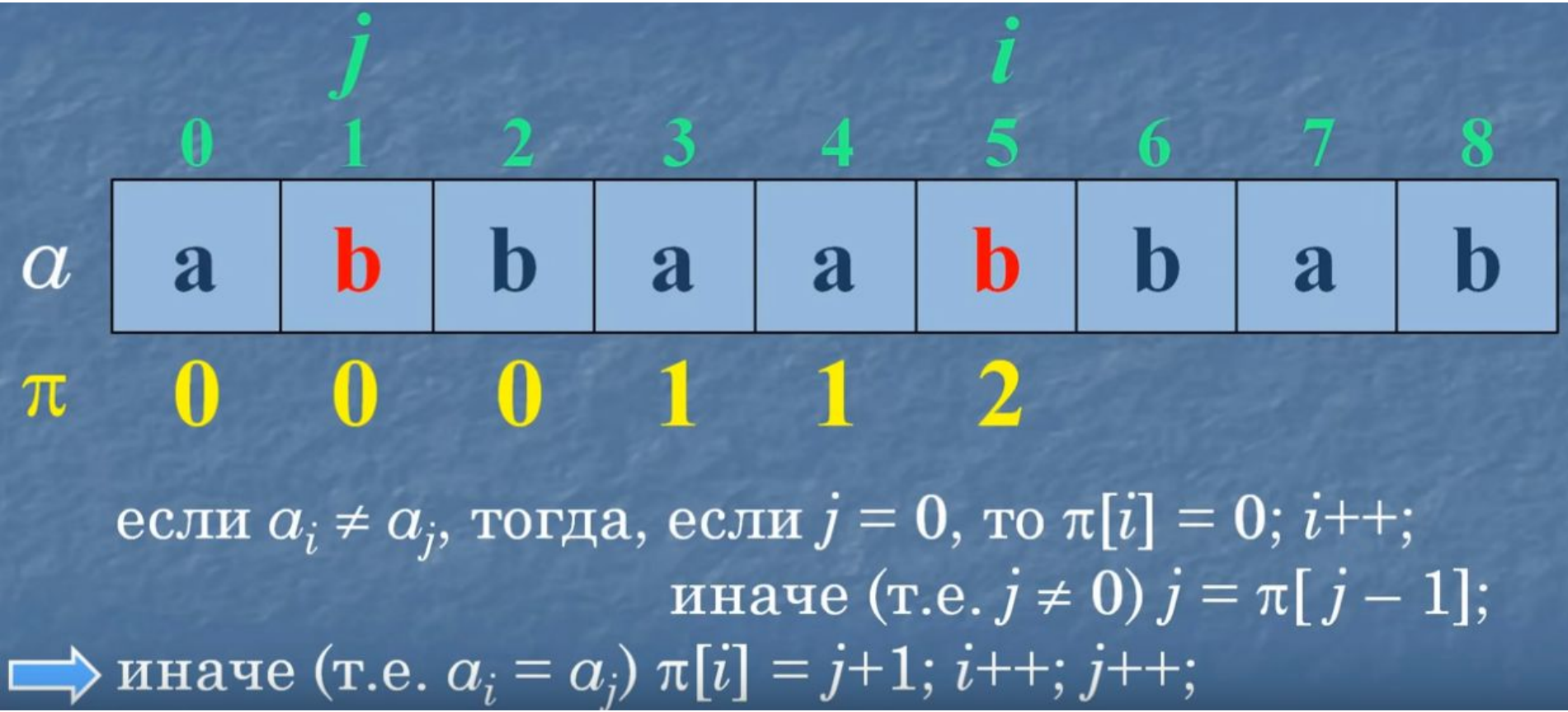

j
i
0 1 2 3 4 5 6 7 8
a
a b b a a b b a b
π
0 0 0 1
если $a_i \neq a_j$, тогда $\pi[i] = 0$; i++;
иначе (т.е. $a_i = a_j$) $\pi[i] = j+1$; i++; j++;



если $a_i \neq a_j$, тогда, если $j = 0$, то $\pi[i] = 0$; i++;

иначе (т.е. $j \neq 0$) $j = \pi[j - 1]$;

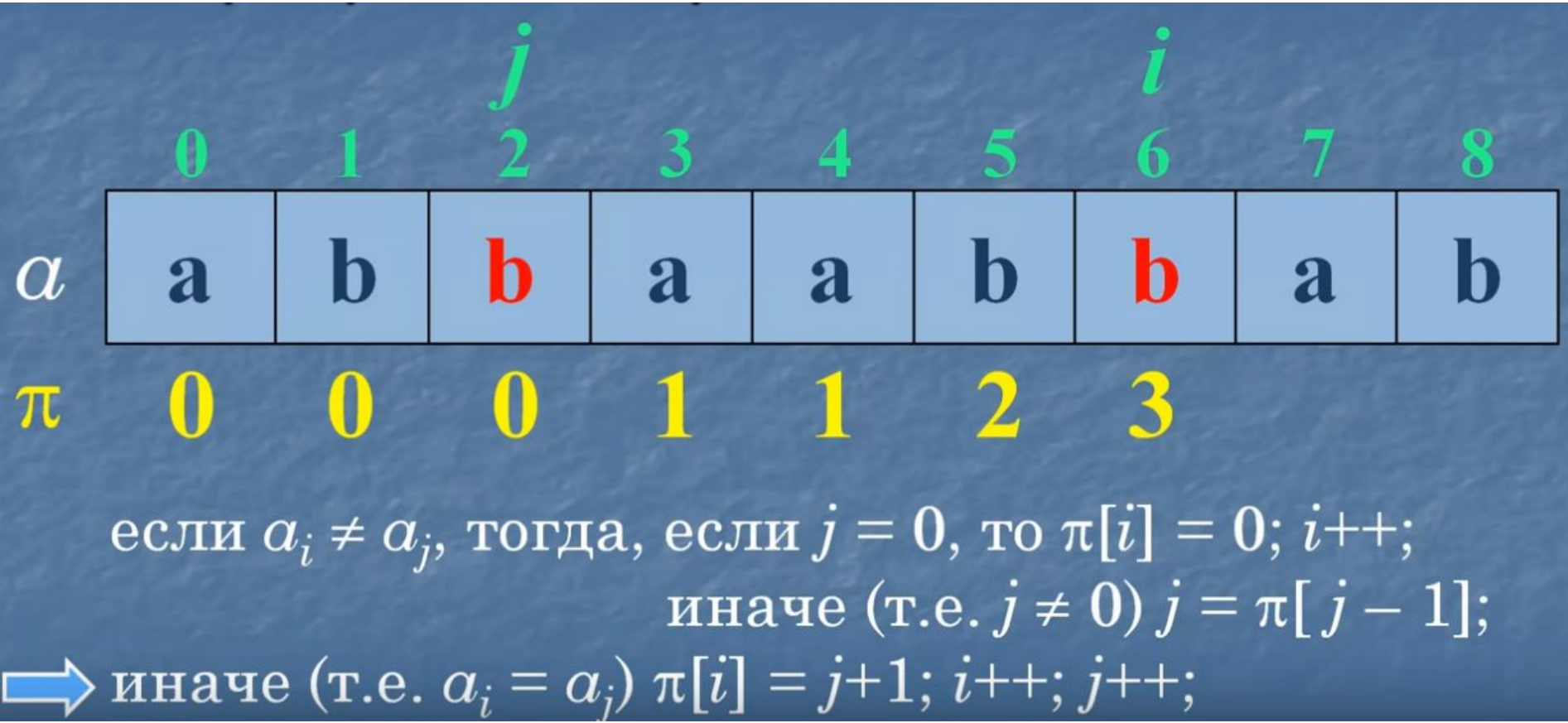
иначе (т.е. $a_i = a_j$) $\pi[i] = j+1$; i++; j++;



если $a_i \neq a_j$, тогда, если $j = 0$, то $\pi[i] = 0$; i++;

иначе (т.е. $j \neq 0$) $j = \pi[j - 1]$;

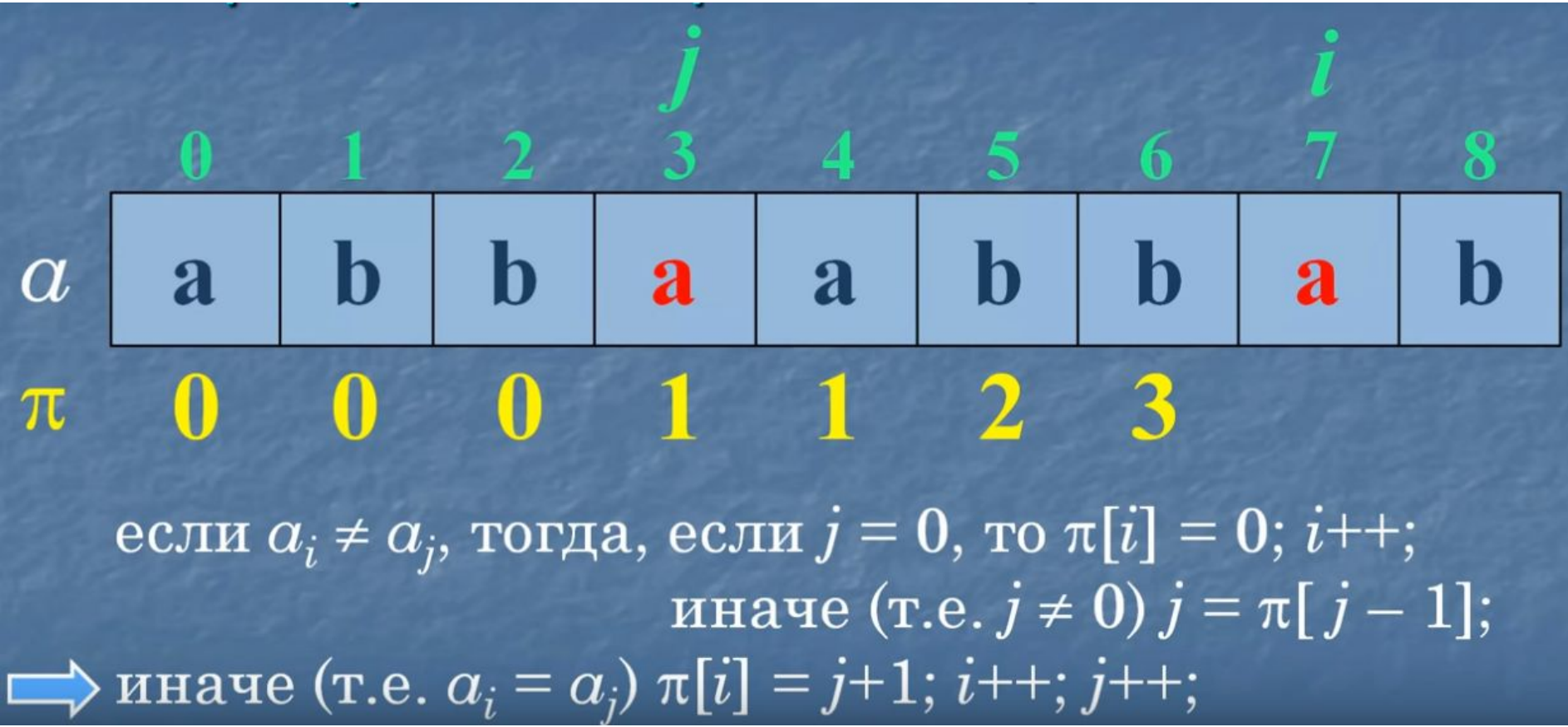
⇨ иначе (т.е. $a_i = a_j$) $\pi[i] = j+1$; i++; j++;



если $a_i \neq a_j$, тогда, если $j = 0$, то $\pi[i] = 0$; i++;

иначе (т.е. $j \neq 0$) $j = \pi[j - 1]$;

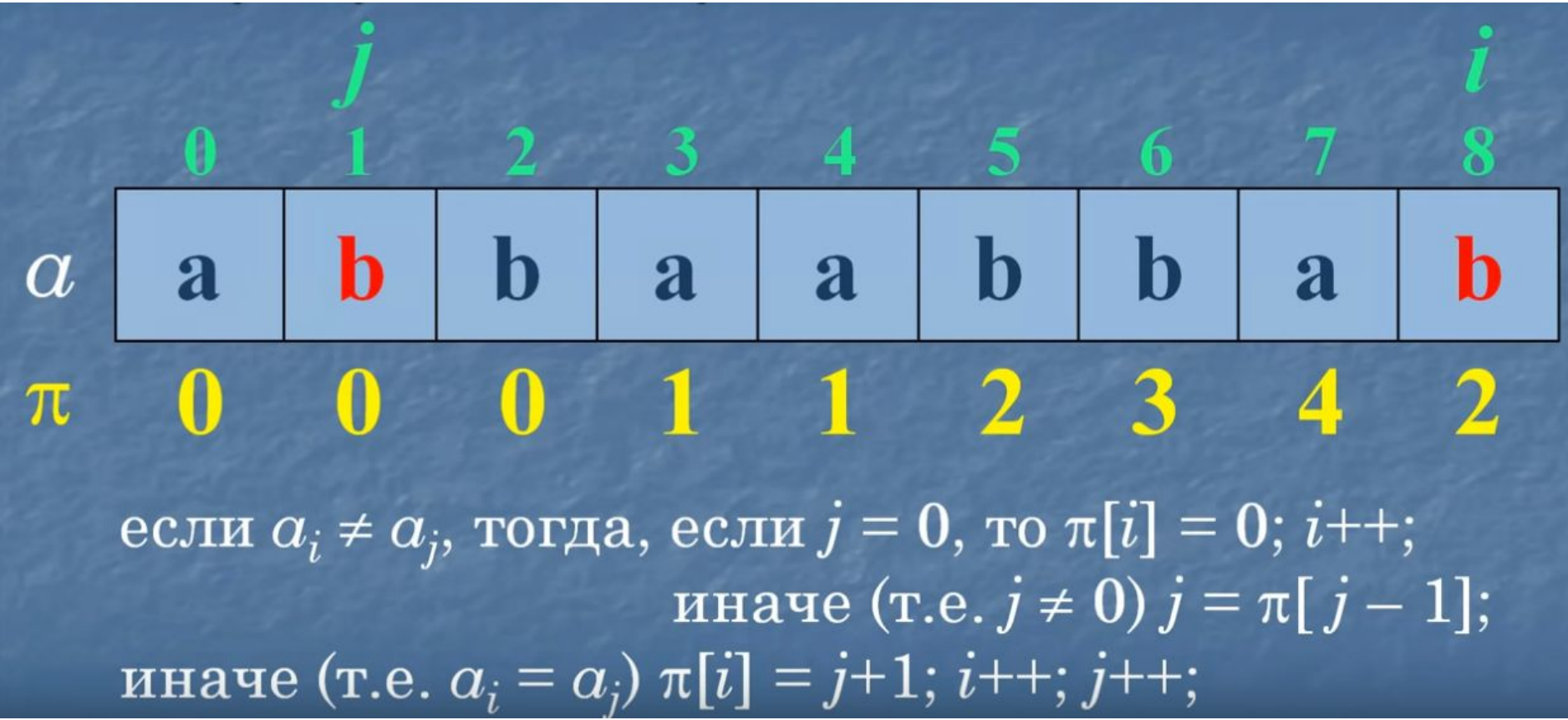
иначе (т.е. $a_i = a_j$) $\pi[i] = j+1$; i++; j++;



если $a_i \neq a_j$, тогда, если $j = 0$, то $\pi[i] = 0$; i++;

иначе (т.е. $j \neq 0$) $j = \pi[j - 1]$;

⇨ иначе (т.е. $a_i = a_j$) $\pi[i] = j+1$; i++; j++;



если $a_i \neq a_j$, тогда, если $j = 0$, то $\pi[i] = 0$; i++;

иначе (т.е. $j \neq 0$) $j = \pi[j - 1]$;

иначе (т.е. $a_i = a_j$) $\pi[i] = j+1$; i++; j++;

$\pi[0]=0;$

$j=0;$

$i=1;$

пока не закончился образ проверяем одно единственное условие:

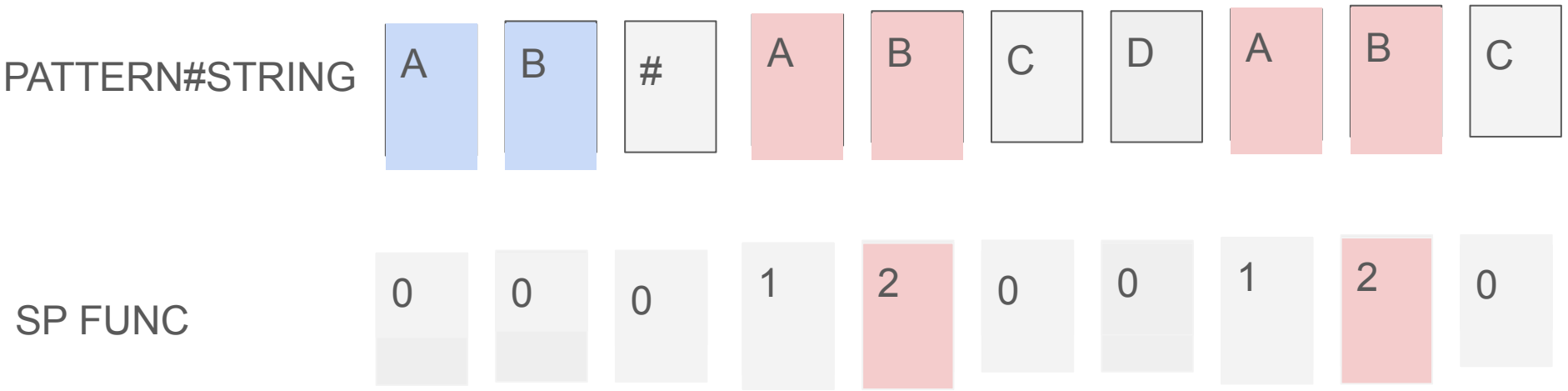
if $a_i == a_j$ then $\pi[i] = j+1;$ $i{+}{+};$ $j{+}{+};$ // $a_i == a_j$

else if $j == 0$ then $\pi[i]=0;$ $i{+}{+};$ // $a_i != a_j$ и $j == 0$

else $j = \pi[j-1];$ // $a_i != a_j$ и $j != 0$

Кнут-Моррис-Пратт (семинарский вариант)

[ссылка на источник этого варианта - КМП]



Таким образом, мы молодцы и, подсчетом суффикс-префикс функции, нашли все вхождения паттерна в строку.

Теперь все отлично, и время работы составляет O(N + M + 1)= O(N + M)

Кнут-Моррис-Пратт

На лекции был разобран другой вариант этого алгоритма, не подразумевавший склеивание строк, но работающий за тоже время.

Важно, что ОБА варианта позволяют найти ВСЕ вхождения паттерна длины M в текст длины N за O(N + M)