Суффиксное дерево (Питер Вайнер, 1973)

T = bananas

T_1 = bananas$

T_2 = ananas$

T_3 = nanas$

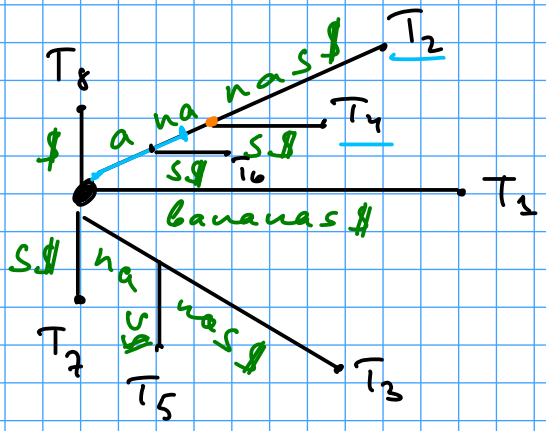
T_4 = anas$

T_5 = nas$

T_6 = as$

T_7 = s$

T_8 = $



NB: если добавлять вершины только при разветвлении, то потребуется $O(n)$ памяти

Поиск: ищем образец в боре.

P = ан и выводим все листья в поддереве.

Время $O(|P| + \#\text{вхождений})$

Факт: Суффиксное дерево можно построить за линейное время.

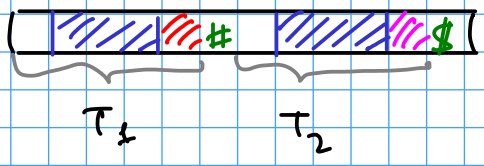
Д. Кнут (1970): // не верно

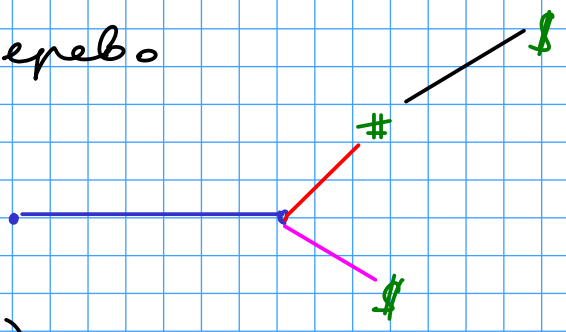
Задача поиска наибольшей общей подстроки двух строк, то она решается за $\Omega(n \log n)$

Вход: T_1 и T_2 Выход: общая подстрока

$T = T_1 \# T_2 \$$

Построим суфф. дерево





⇒ Решение за $O(n)$

Минусы:

- довольно сложная алгоритмика
- [память $O(|T| \cdot |\Sigma|)$ — алфавит
  поиск $O(|P| \log |\Sigma|)$

Суффиксный массив

T = bananas

T_1 = bananas
T_2 = ananas
T_3 = nanas
T_4 = anas
T_5 = nas
T_6 = as
T_7 = s

Отсортируем массив T_i

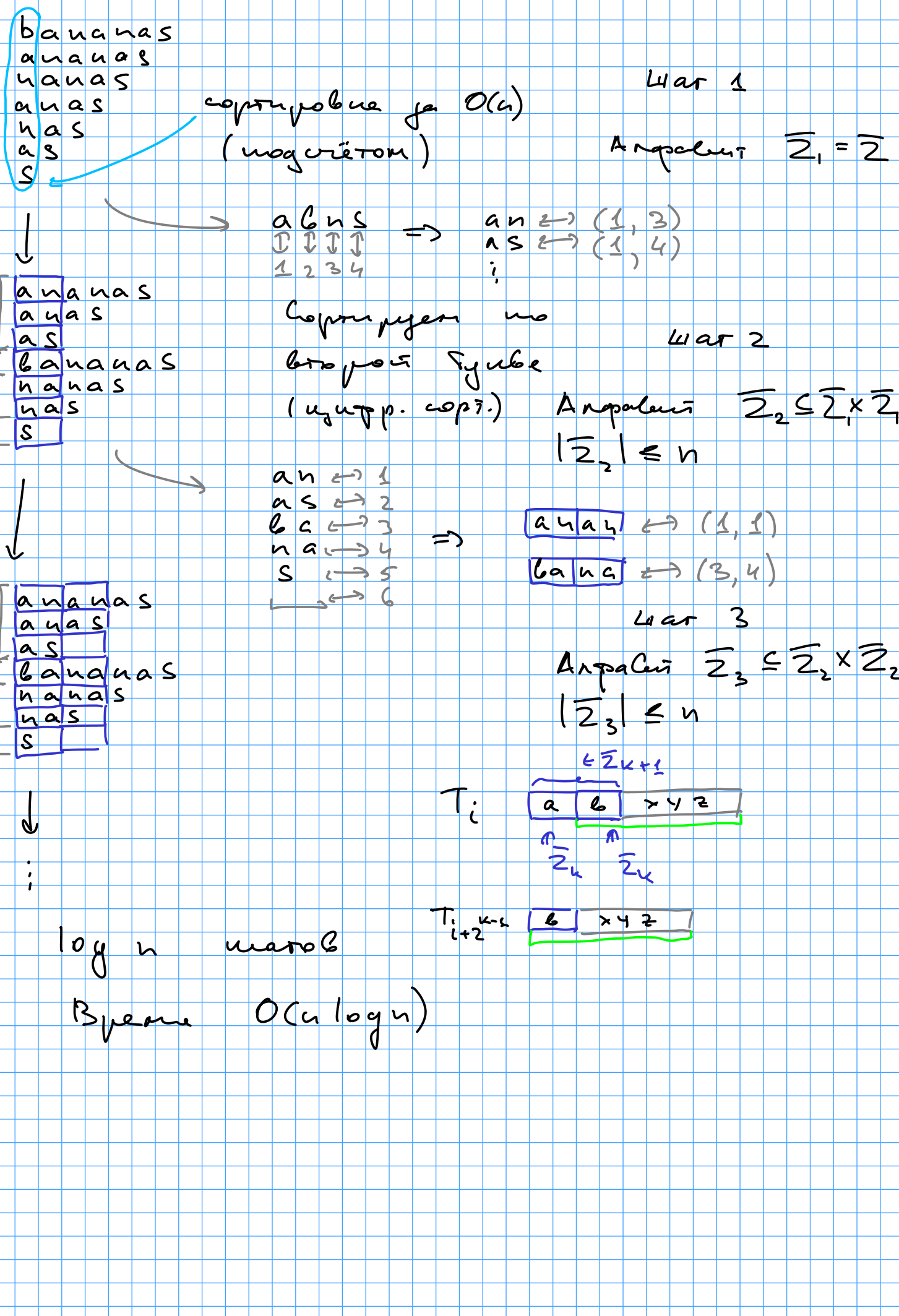
2461357 SA

SA можно построить за $O(n)$.

Например 2/3 С.Д.

Факт ∃ алгоритм построения за $O(n)$ независимо от $|\Sigma|$

Построение за $O(n \log n)$



Поиск в SA

Наивный алгоритм: бин. поиск $\Rightarrow$

$$O(|P| \cdot \log |T|)$$

Можно реализовать за $O(|P| + \log |T|)$

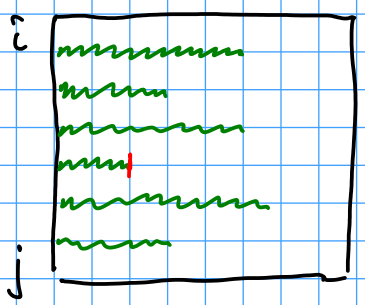
SA	ISA
2	4
4	1
6	5
1	2
3	6
5	3
7	7

$$SA[ISA[i]] = i$$
$$SA[i] = j \iff$$
$$ISA[j] = i$$

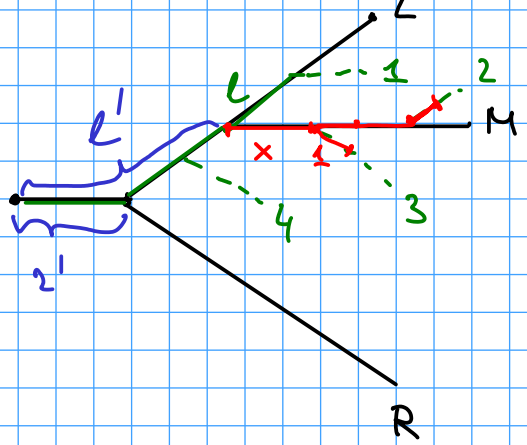
$\equiv$ $lcp(S_1, S_2)$ - наибольший общий префикс

Утв: $|lcp(T_{SA[i]}, T_{SA[j]})| =$

$$= \min_{i \le k < j} |lcp(T_{SA[k]}, T_{SA[k+1]})|$$



Как устроить поиск: Find(P, L, R, ℓ, 2) (строки; вспомог. числа)



$M = \lfloor \frac{L+R}{2} \rfloor$

$\ell = |lcp(L, P)|$
$2 = |lcp(P, R)|$ } за рекурсии

$\ell' = |lcp(L, M)|$
$2' = |lcp(M, R)|$ } *

- $\ell \geq r$
  - $\ell' > \ell$ ④ go right
  - $\ell' < \ell$ ① go left
  - $\ell' = \ell$ ② vs ③

  читаем символы P, пока не отклонимся от M

  (таким образом получим ℓ или r для следующего шага)

- $\ell < r$

  симметрично

$O(|P| + \log |T|)$

* Как вычислять $lcp(T_i, T_j)$

1. Посчитать все пары $lcp(T_i, T_j)$ $O(n^2)$
2. Пересчитать только для тех пар, кот. встречаются в поиске. $O(n^2)$ времени $O(n)$ памяти
3. Давайте вычислим lcp для последоват. суффиксов.

$\overline{LCP}[i] = |lcp(T_{SA[i]}, T_{SA[i+1]})|$

Либо вычислять $|lcp(T_i, T_j)|$ с/з RMQ

Либо запомнить табл. за $O(n)$

Алгоритм (Aimura, Arikawa, Lee, Kosai, Park)

$LCP[1] = \ell = |lcp(T_{SA[1]}, T_{SA[2]})|$

for $i = 2$ to n:

$\ell = \max(0, \ell - 1)$

while $T_i[\ell+1] = T_{SA[ISA[i]+1]}[\ell+1]$

$\ell = \ell + 1$

$LCP[i] = \ell$

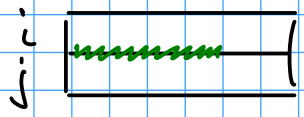
ISA — # суффикса следующего за i в SA

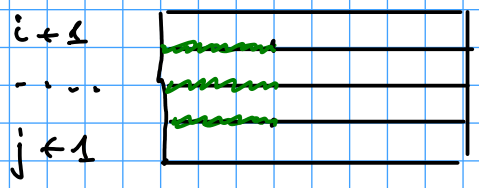
$SA[1] = i$

$SA[2] = j$

$|lcp(T_i, T_j)| = \ell$

$|lcp(T_{i+1}, T_{j+1})| = \ell - 1$





$\overline{LCP}$ за $O(n)$