Суффиксные деревья (Питер Вайнер, 1973)

T = bananas

T_1 = bananas$

T_2 = ananas$

T_3 = nanas$

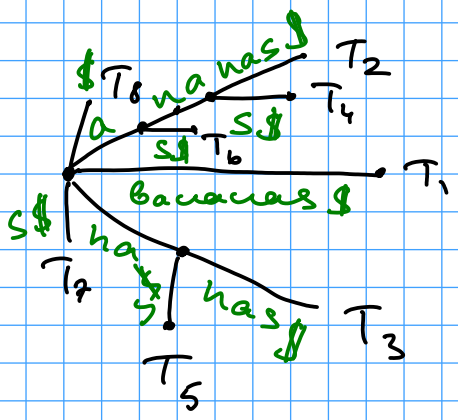
T_4 = anas$

T_5 = nas$

T_6 = as$

T_7 = s$

T_8 = $



NB: можно хранить в $O(n)$ памети ("сжатое представление")

Поиск:

P - образец.

Поиск: $O(|P| + \#\text{ вхождений})$

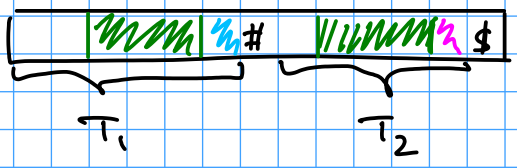
D. Knuth (1970)

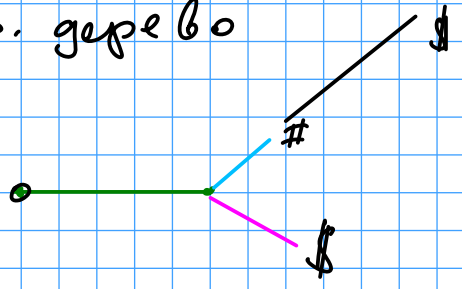
Conj: Задача о макс общ. подстроке решается за $\Omega(n \log n)$.

Вход: T_1, T_2

$T = T_1 \# T_2 \$$

Построим для T суфф. дерево





$\Rightarrow$ решается за время постр. с.д. $+ O(n) = O(n)$

Минусы:

- сложный алгоритм построения

- $\begin{cases} \text{памети предуства} & O(|\Sigma| \cdot n) \\ \text{поиск за} & O(P \cdot \log|\Sigma|) \end{cases}$

Суффиксный массив

T_1 = banana$
T_2 = anana$
T_3 = nana$
T_4 = ana$
T_5 = na$
T_6 = a$
T_7 = s$
T_8 = $

T_8 = $
T_2 = ananas$
T_4 = ana$
T_6 = a$
T_1 = banana$
T_3 = nana$
T_5 = na$
T_7 = s$

SA | 8 2 4 6 1 3 5 7 |

Тот же порядок, что и в с.г. ⇒ можно построить за $O(n)$

Построим за $O(n \log n)$.

1	banana$
2	anana$
3	nana$
4	ana$
5	na$
6	a$
7	s$

→

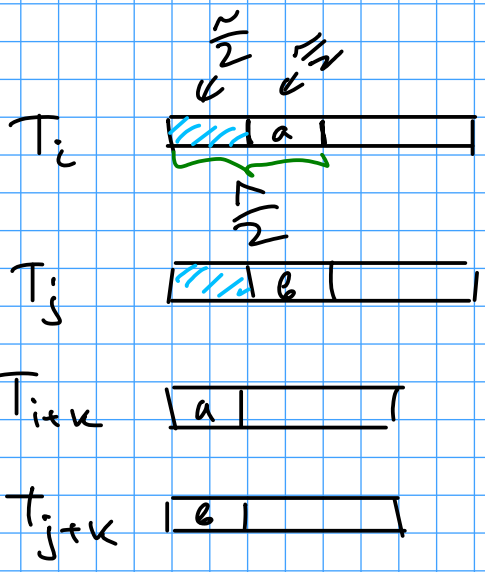
	anana$
	ana$
	a$
0 →	banana$
0 →	nana$
→	na$
	s$

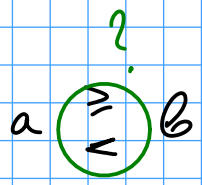
$\overline{2} \to \overline{2}' \subseteq \overline{2} \times \overline{2}$

Слова в алфавите $\tilde{\overline{2}}$

Переходим к алфавиту $\hat{\overline{2}} \subseteq \tilde{\overline{2}} \times \hat{\overline{2}}$

Мы хотим отсортировать по первой букве. Порядок на $\hat{\overline{2}}$ должен соответствовать порядку от $\overline{2}$.





решается сравнением номеров T_{i+k} и T_{j+k}

$\Rightarrow$ $\log n$ фаз по n операций (подсчёт)

$\Rightarrow$ $O(n \log n)$

Naive search

Вход: SA для T

P - образец

$O(|P| \cdot \log |T|)$ - бинарный поиск

Точно за $O(|P| + \log |T|)$

SA

8
2
4
6
1
3
5
7

ISA

5
2
6
3
7
4
8
1

$SA[ISA[i]] = i$

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

Утв. 1

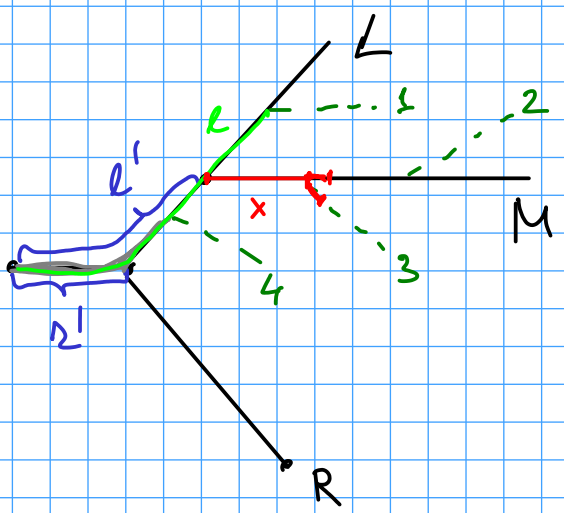
$$|lcp(T_{SA[i]}, T_{SA[j]})| = \min_{i \le k < j} |lcp(T_{SA[k]}, T_{SA[k+1]})|$$

Будем искать бинарным поиском

Find (P, L, R, ℓ, r)

M = строка посередине м/у L и R

$\ell = |lcp(L, P)|$
$r = |lcp(P, R)|$
$\ell' = |lcp(L, M)|$
$r' = |lcp(M, R)|$
) из рекурсии
) ? ⑤



• $\ell \geq r$

- $\ell' > \ell \Rightarrow$ ④ go right
- $\ell' < \ell \Rightarrow$ ① go left
- $\ell = \ell' \Rightarrow$ ② v ③

читаем $X+1$ символ и в зависимости от него решаем куда идти.

• $\ell < r$

симметрично

$\Rightarrow$ Поиск за $O(\log|T| + |P|)$

⑤ Как вычислить LCP

1. Предподсчёт $LCP(T_i, T_j)$ $O(T^2)$ памяти и времени
2. Предподсчёт только для пар суффиксов, которые могут встретиться. $O(T^2)$ времени и $O(T)$ памяти.

3. Давайте вычислим lcp для посл. суфф.

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

Утв 2.

По утв 1.

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

RMQ за $(O(n), O(1)) \Rightarrow$ можем вычислять за $O(1)$ при посте.

Как вычислить $\overline{LCP}$

Алгоритм (Arimura, Arikawa, Lee, Kosaki, Park)

$\ell = |lcp(T_1, T_{SA[ISA[1]+1]})|$

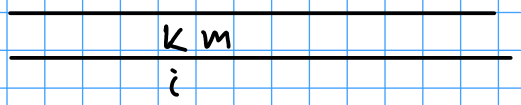
for $i = 2$ to n:

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

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

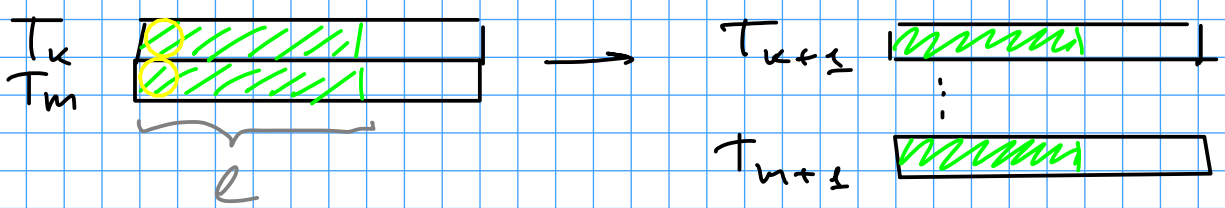
$L = L + 1$

$LCP[i] = \ell$



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

$\Rightarrow |lcp(T_{k+1}, T_{m+1})| \geq \ell - 1$



$O(n)$