Суффиксные деревья: новые идеи и открытые проблемы

Татьяна Стариковская
Computer Science Center, 2014

LINEAR PATTERN MATCHING ALGORITHMS

Peter Weiner

The Rand Corporation, Santa Monica, California*

Abstract

In 1970, Knuth, Pratt, and Morris [1] showed how to do basic pattern matching in linear time. Related problems, such as those discussed in [4], have previously been solved by efficient but sub-optimal algorithms. In this paper, we introduce an interesting data structure called a bi-tree. A linear time algorithm for obtaining a compacted version of a bi-tree associated with a given string is presented. With this construction as the basic tool, we indicate how to solve several pattern matching problems, including some from [4], in linear time.

"Algorithm of the year 1973" -Donald Knuth

I. Introduction

In 1970, Knuth, Morris, and Pratt [1-2] showed how to match a given pattern into another given string in time proportional to the sum of the lengths of the pattern and string. Their algorithm was derived from a result of Cook [3] that the 2-way deterministic pushdown languages are recognizable on a random access machine in time O(n). Since 1970, attention has been given to several related problems in pattern matching [4-6], but the algorithms developed in these investigations usually run in time which is slightly worse than linear, for example O(n log n). It is of considerable interest to either establish that there exists a non-linear lower bound on the run time of all algorithms which solve a given pattern matching problem, or to exhibit an algorithm whose run time is of O(n).

In the following sections, we introduce an interesting data structure, called a bi-tree, and show how an efficient calculation of a bi-tree can be applied to

[illegible] giving a formal definition of a bi-tree, we review [illegible] definitions and terminology concerning t-ary trees [illegible] see Knuth [7] for further details.)

A [illegible] tree T over $\Sigma = \{\sigma_1,\ldots,\sigma_t\}$ is a set of nodes N which is [illegible] empty or consists of a *root*, $n_0 \in N$, and t ordered, disjoint t-ary trees.

Clearly, every node $n_i \in N$ is the root of some t-ary tree T^i which itself consists of n_i and t ordered, disjoint t-ary trees, $T^i_1, T^i_2, \ldots, T^i_t$. We call the tree T^i_j a *sub-tree* of T^i, and all sub-trees of T^i_j are considered to be sub-trees of T^i. It is natural to associate with a tree T a *successor function*

$$S: N\times\Sigma \rightarrow (N-\{n_0\}) \cup \{NIL\}$$

defined for all $n_i \in N$ and $\sigma_j \in \Sigma$ by

n^i_j the root of T^i_j if T^i_j is non-empty

suffix tree

Scholar

About 92,200 results (**0.04** sec)

Articles

Case law

My library **New!**

Any time

Since 2014

Since 2013

Since 2010

Custom range...

Sort by relevance

Sort by date

A space-economical **suffix tree** construction algorithm

EM McCreight - Journal of the ACM (JACM), 1976 - dl.acm.org

Abstract A new algorithm is presented for constructing auxiliary digital search trees to aid in exact-match substring searching. This algorithm has the same asymptotic running time bound as previously published algorithms, but is more economical in space. Some **...**

Cited by 1712 Related articles All 15 versions Cite Save

Optimal **suffix tree** construction with large alphabets

M Farach - ... of Computer Science, 1997. Proceedings., 38th ..., 1997 - ieeexplore.ieee.org

Abstract The sufix **tree** of a string is the fundamental data structure of combinatorial pattern matching. Weiner [Wei73], who introduced the data structure, gave an O (n)-time algorithm for building the mfiz **tree** of an n-character string drawn from a constant size alphabet. In **...**

Cited by 367 Related articles All 22 versions Cite Save

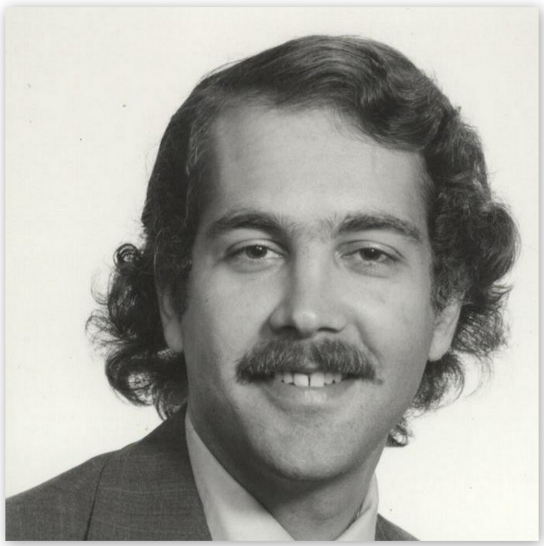
Spelling approximate repeated or common motifs using a **suffix tree**

MF Sagot - LATIN'98: Theoretical Informatics, 1998 - Springer

Abstract We present in this paper two algorithms. The first one extracts repeated motifs from a sequence defined over an alphabet σ. For instance, σ may be equal to {A, C, G, T} and the

1973+40=2013

Серия докладов на Combinatorial Pattern Matching 2013 (suffixtree.org)



?=

- Лекция 1: Определение и приложения.
- Лекция 2: Построение.
- Лекция 3: Когда дерево является суффиксным?

Лекция 1: Определение и приложения.

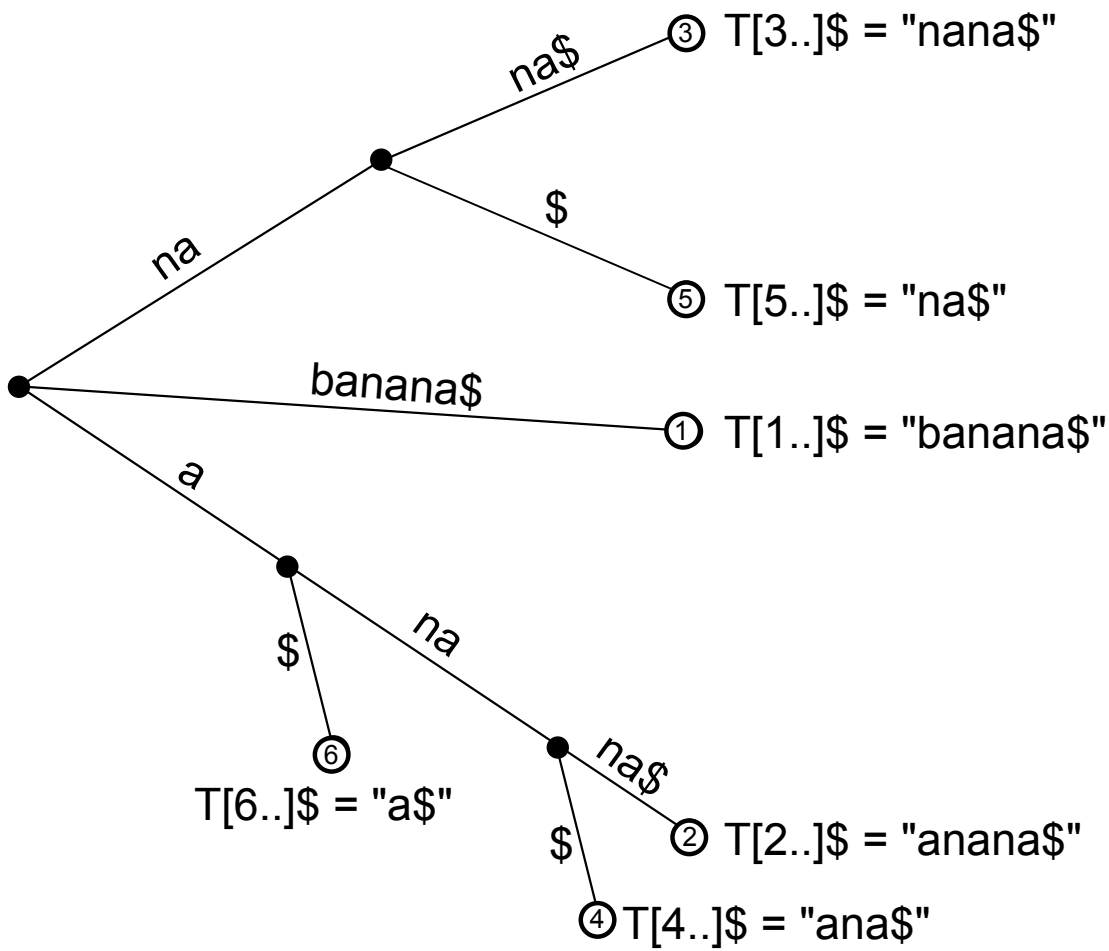
Суффиксы слова $T =$ “$banana''$:

$T[1..] =$ “$banana''$
$T[2..] =$ “$anana''$
$T[3..] =$ “$nana''$
$T[4..] =$ “ana'',
$T[5..] =$ “na''
$T[6..] =$ “a''

Суффиксное дерево: определение

- СД слова T длины n — ордерево, у которого n листьев
- Каждая внутренняя вершина имеет хотя бы двух сыновей, на каждом ребре написано подслово T
- Метки ребер, выходящих из одной вершины, начинаются с разных букв
- Вдоль пути от корня до листа i написано слово $T[i..]\$$

Суффиксное дерево слова $T = \text{“}banana''$



Память $= \mathcal{O}(n)$**:**

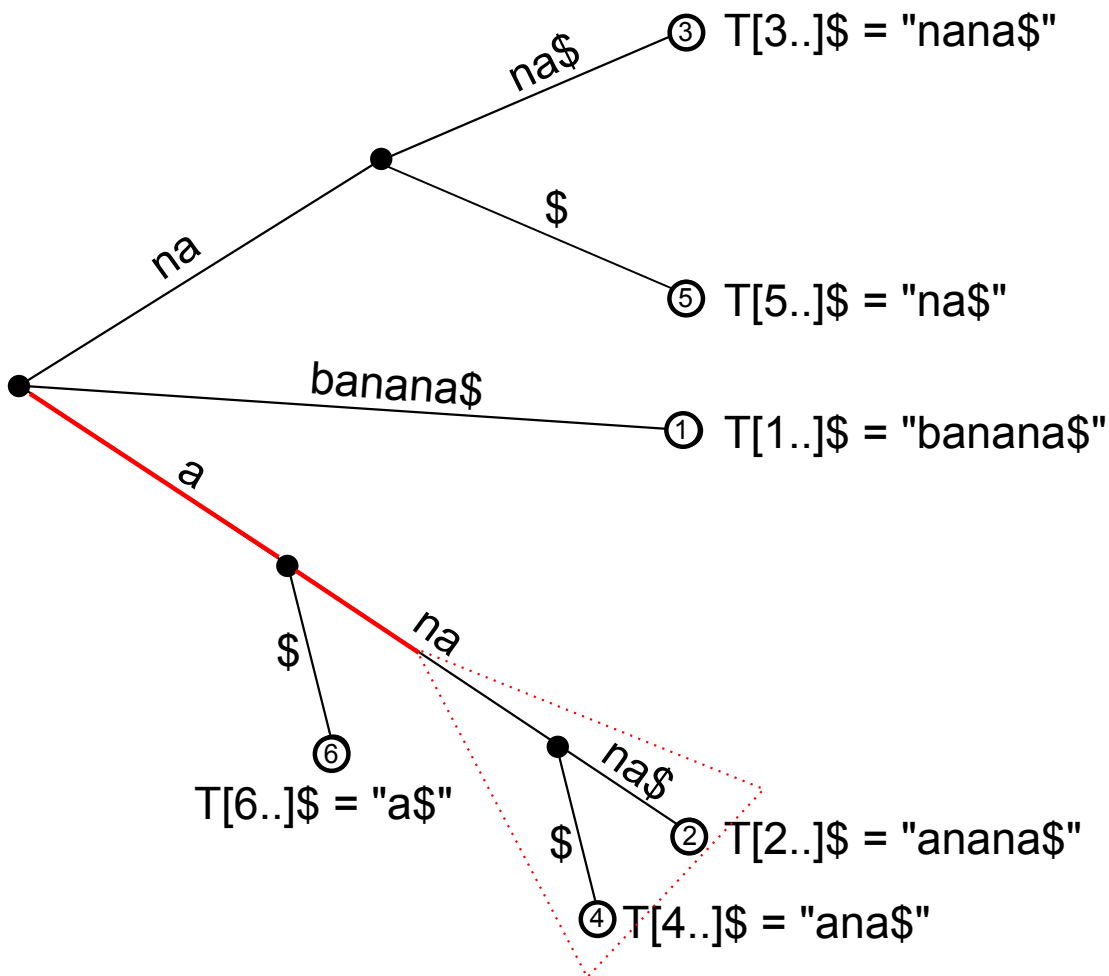
- $\leq 2n - 1$ вершин
- $\leq 2n - 2$ ребер
- В каждой вершине массив: буква $\to$ сын

Время построения $= \mathcal{O}(n)$

Поиск вхождений слова P в слово T

- i — позиция вхождения слова P в $T \Leftrightarrow T[i..]$ начинается с P
- Пусть v — конец пути из корня с меткой P
- i — позиция вхождения слова P в $T \Leftrightarrow$ лист i находится в поддереве v
- **Алгоритм:** найти v, перечислить листы

Найти вхождения $P =$ “an” в $T =$ “$banana$”



Время: $\mathcal{O}(|P| + occ)$

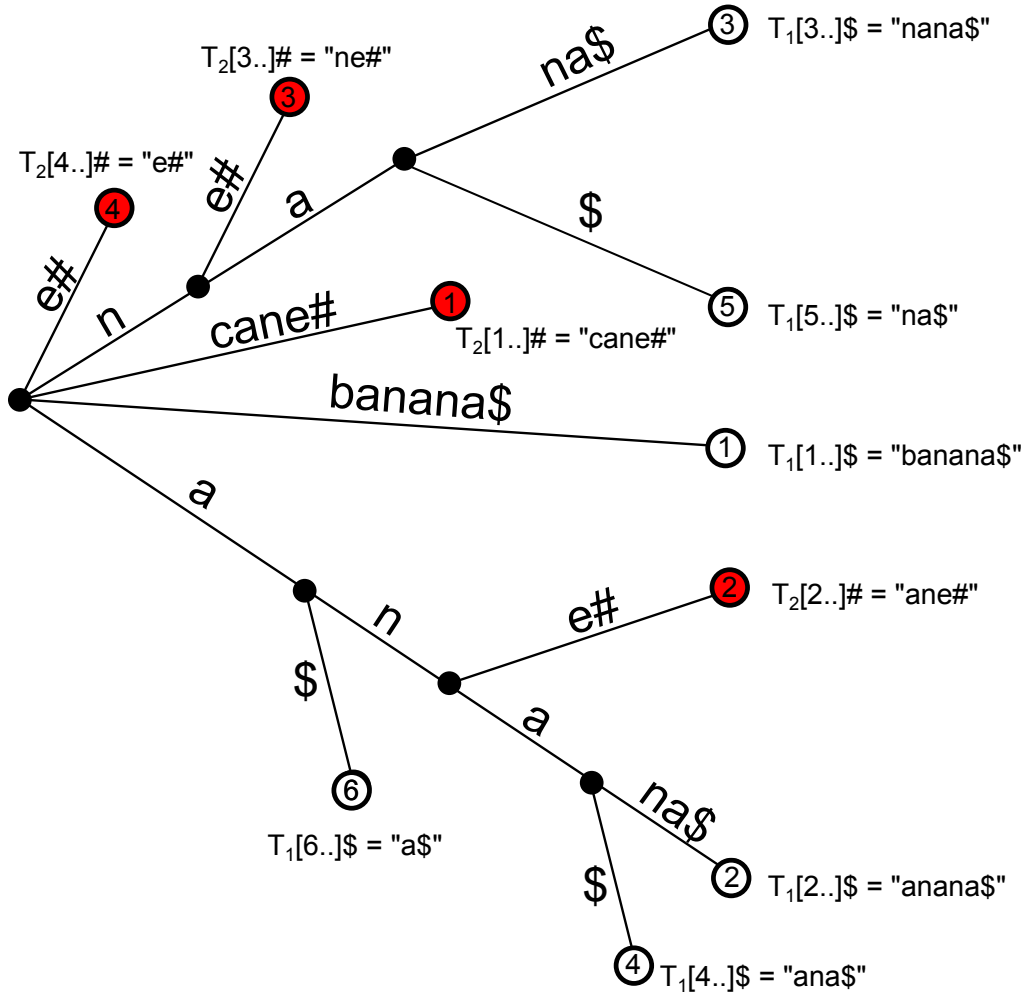
Поиск наибольшего общего подслова

- Даны слова T и S суммарной длины n. Найти максимальное по длине слово, входящее и в T, и в S.
- Дональд Кнут, 1970: для решения требуется $\Omega(n \log n)$ времени
- Суффиксные деревья — $\mathcal{O}(n)$ времени

Поиск наибольшего общего подслова

- Обобщенное СД содержит суффиксы слов T_1 и T_2
- Листы покрашены в два цвета
- Метка вершины входит в T_1 и в $T_2 \Leftrightarrow$ в ее поддереве есть листы обоих цветов

Найти наибольшее общее подслово $T_1 =$ “$banana$” и $T_2 =$ “$cane$”



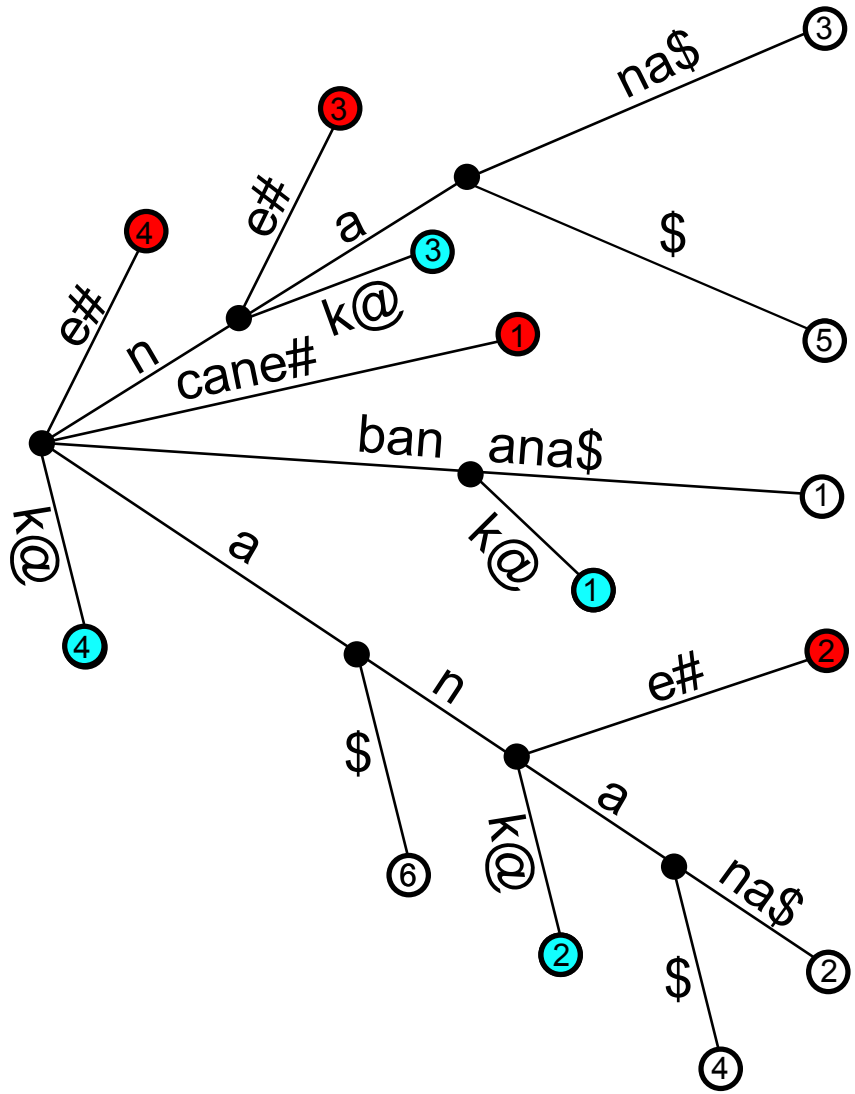
Время: $\mathcal{O}(n)$

Поиск наибольшего общего подслова

- Даны слова $T_1, T_2, \ldots, T_m$ суммарной длины n. Найти максимальное по длине слово, входящее в $\geq d$ из этих слов.

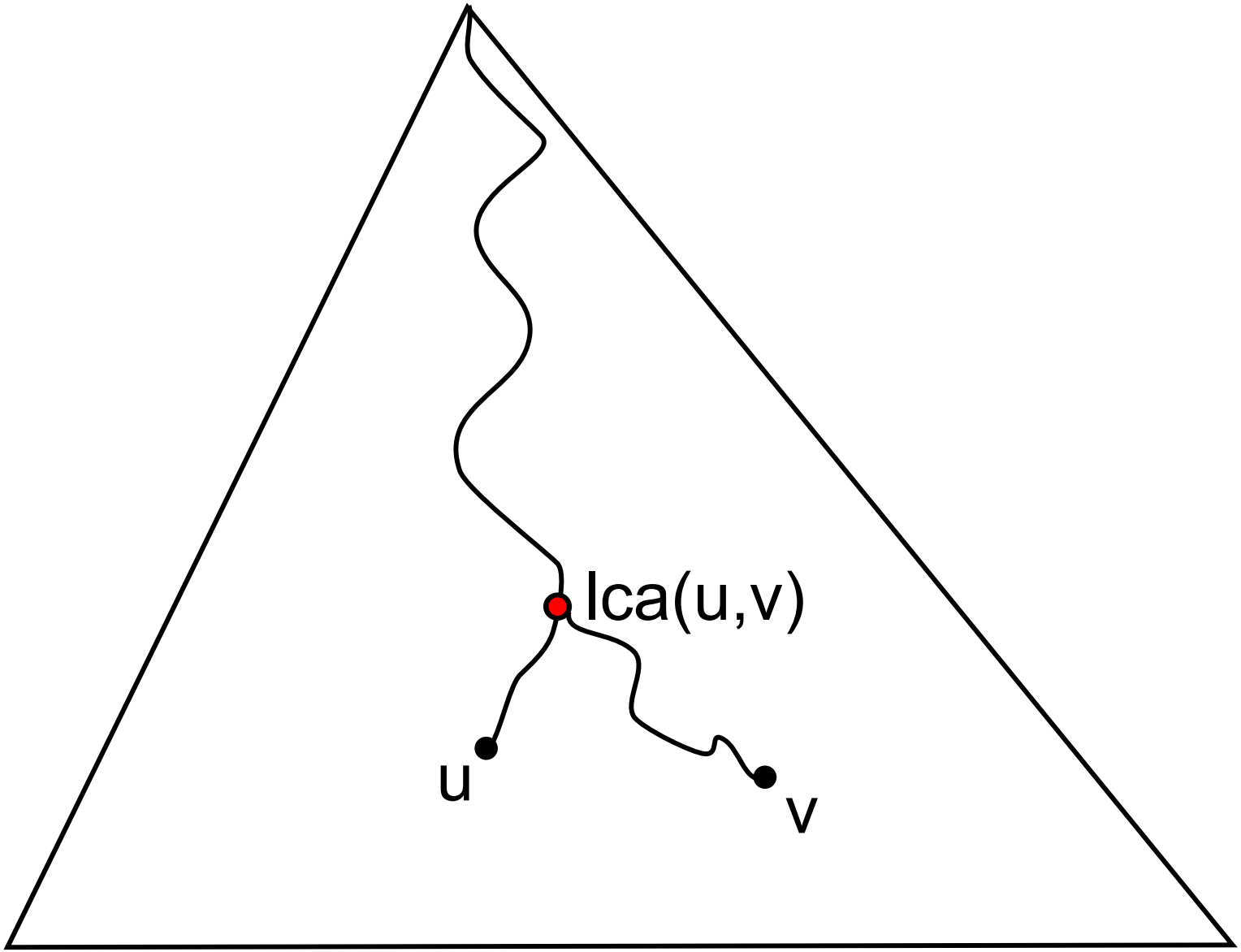
- Обобщенное СД содержит суффиксы слов $T_1, T_2, \ldots, T_m$
- Листы покрашены в m цветов
- Метка вершины входит в $\geq d$ слов $\Leftrightarrow$ в ее поддереве есть листы d цветов

Найти макс. по длине слово, входящее в хотя бы два слова из $T_1 =$ “$banana$”, $T_2 =$ “$cane$” и $T_3 =$ “$bank$”

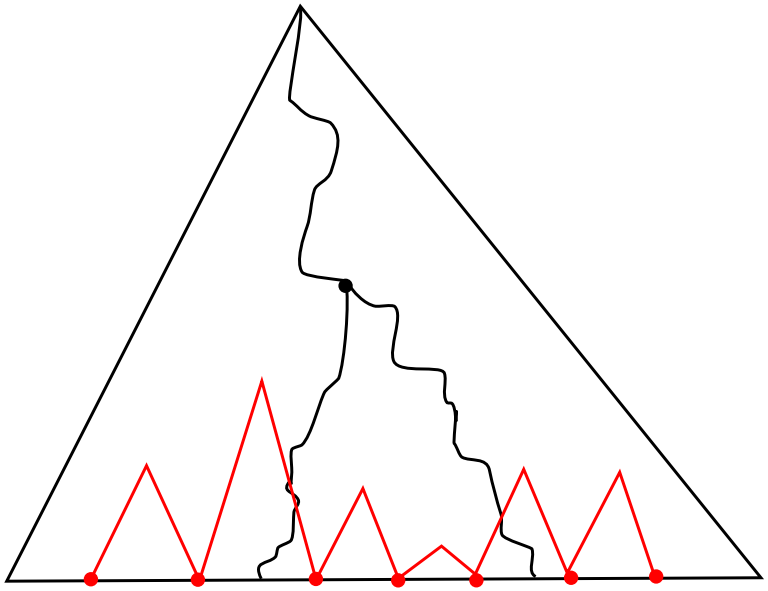


Подзадача: найти количество цветов в каждом поддереве

Задача LCA (самый глубокий общий предок)



$\mathcal{O}(n)$ памяти, $\mathcal{O}(1)$ времени на поиск $lca(u, v)$



- $LCA_c(x)$ — # вершин в T_x, являющихся LCA двух последовательных листьев цвета c
- # листьев цвета c в поддереве x - $LCA_c(x) = 1$
- $\sum_c$ # листьев цвета c в T_x - $LCA_c(x)$ = # цветов в поддереве x!
- $\sum_c$ # листьев цвета c в T_x = # листьев в поддереве x
- $\sum_c LCA_c(x)$?

- Лукас Чи Квонг Хьюи, 1992: $\mathcal{O}(n)$ времени, $\mathcal{O}(n)$ памяти
- $\Rightarrow$ Задача о наибольшем общем подслове может быть решена за $\mathcal{O}(n)$ времени, $\mathcal{O}(n)$ памяти

Класс А	Класс Б
a342cb214d0001acd24a3a12dadbcb4a0000000	31422bd131b4413cd422a1acda332342d3ab4c4
41cacbddad3142a2344a2ac23421c00adb4b3cb	a11acb2d3dbc1ca22c23242c3a142b3adb243c1
4d4ac42d23b141acd24a3a12dadbcb4a2134141	2d2a4b1d32b21ca2312a3411d00000000000000
3dcacb1dadbc42ac2cc31012dadbcb4adb40000	4344c32d21b1123cdc000000000000000000000
3d43232d32323c213c22d2c23234c332db4b300	ad12cbdd3d4c1ca112cad2ccd00000000000000
ad1acbdda212b1acd24a3a12dadbcb400000000	431a2b2d2d44b2acd2cad2c2223b40000000000
2124cbddadbcb1a42cca3412dadbcb423134bc1	2d2a1bd2431141342c13d212d233c34a3b3b000
4d4a2b1dadbc3ca22c00000000000000000000	4d4a1bdd23b242a22c2a1a1cda2b1baa33a0000
a24acb1d32b412acd24a3a12dadbcb422143bc0	23c4cbddadb23c322c2a222223232b443b24bc3
ad1ac3d2a23431223c000012dadbcb400000000	4313c31d42b14c421c42332cd2242b3433a3343
3dcacbd32d313c21142323cc300000000000000	ad122b1da2b11242dc1a3a12100000000000000
4d1ac3dd43421240d24a3a12dadbcb400000000	ad1a13d23d3cb2a21ccada24d2131b440000000
a24acb11a3b24cacd12a241cdadbcb4adb4b300	33c4cbd142141ca424cad34c122413223ba4b40
adcacb1dad3141ac212a3a1c3a144ba2db41b43	adcacbddadbc42ac2c2ada2cda341baa3b24321
40c2cbddadb4b1acd24a3a12dadbcb43d133bc4	4dc2cb2dadb24c412c1ada2c3a341ba20000000
4dc4cbdd31b1b2213c4ad412dadbcb4adb00000	431acbddad3c4c213412da22d3d1132a1344b1b
4d4a23d24131413234123a243a2413a21441343	a21a1b2dadb24ca22c1ada2cd32413200000000
4d14c3d2ad4cbcac1c003a12dadbcb4adb40000	3d2a2bddadbcbca11c2a2accda1b2ba20000000
a21ac3d2ad3c4c4cd40a3a12dadbcb400000000	
a2cacbd1a13211a2d02a2412d0dbcb4adb4b3c0	
adc4cbddadbcbc2c2cc43a12dadbcb4211ab343	
a3cacbddadbcbca42c2a3212dadbcb42344b3cb	

Jacob B.A., Levitt S.D. Rotten apples: an investigation of the prevalence and predictors of teacher cheating.

Класс А	Класс Б
a342cb214d0001acd24a3a12dadbcb4a0000000	31422bd131b4413cd422a1acda332342d3ab4c4
41cacbddad3142a2344a2ac23421c00adb4b3cb	a11acb2d3dbc1ca22c23242c3a142b3adb243c1
4d4ac42d23b141acd24a3a12dadbcb4a2134141	2d2a4b1d32b21ca2312a3411d00000000000000
3dcacb1dadbc42ac2cc31012dadbcb4adb40000	4344c32d21b1123cdc000000000000000000000
3d43232d32323c213c22d2c23234c332db4b300	ad12cbdd3d4c1ca112cad2ccd00000000000000
ad1acbdda212b1acd24a3a12dadbcb400000000	431a2b2d2d44b2acd2cad2c2223b40000000000
2124cbddadbcb1a42cca3412dadbcb423134bc1	2d2a1bd2431141342c13d212d233c34a3b3b000
4d4a2b1dadbc3ca22c00000000000000000000	4d4a1bdd23b242a22c2a1a1cda2b1baa33a0000
a24acb1d32b412acd24a3a12dadbcb422143bc0	23c4cbddadb23c322c2a222223232b443b24bc3
ad1ac3d2a23431223c00012dadbcb400000000	4313c31d42b14c421c42332cd2242b3433a3343
3dcacbd32d313c21142323cc300000000000000	ad122b1da2b11242dc1a3a12100000000000000
4d1ac3dd43421240d24a3a12dadbcb400000000	ad1a13d23d3cb2a21ccada24d2131b440000000
a24acb11a3b24cacd12a241cdadbcb4adb4b300	33c4cbd142141ca424cad34c122413223ba4b40
adcacb1dad3141ac212a3a1c3a144ba2db41b43	adcacbddadbc42ac2c2ada2cda341baa3b24321
40c2cbddadb4b1acd24a3a12dadbcb43d133bc4	4dc2cb2dadb24c412c1ada2c3a341ba20000000
4dc4cbdd31b1b2213c4ad412dadbcb4adb00000	431acbddad3c4c213412da22d3d1132a1344b1b
4d4a23d24131413234123a243a2413a21441343	a21a1b2dadb24ca22c1ada2cd32413200000000
4d14c3d2ad4cbcac1c003a12dadbcb4adb40000	3d2a2bddadbcbca11c2a2accda1b2ba20000000
a21ac3d2ad3c4c4cd40a3a12dadbcb400000000	
a2cacbd1a13211a2d02a2412d0dbcb4adb4b3c0	
adc4cbddadbcbc2c2cc43a12dadbcb4211ab343	
a3cacbddadbcbca42c2a3212dadbcb42344b3cb	

Jacob B.A., Levitt S.D. Rotten apples: an investigation of the prevalence and predictors of teacher cheating.

Поиск документов, содержащих слово P

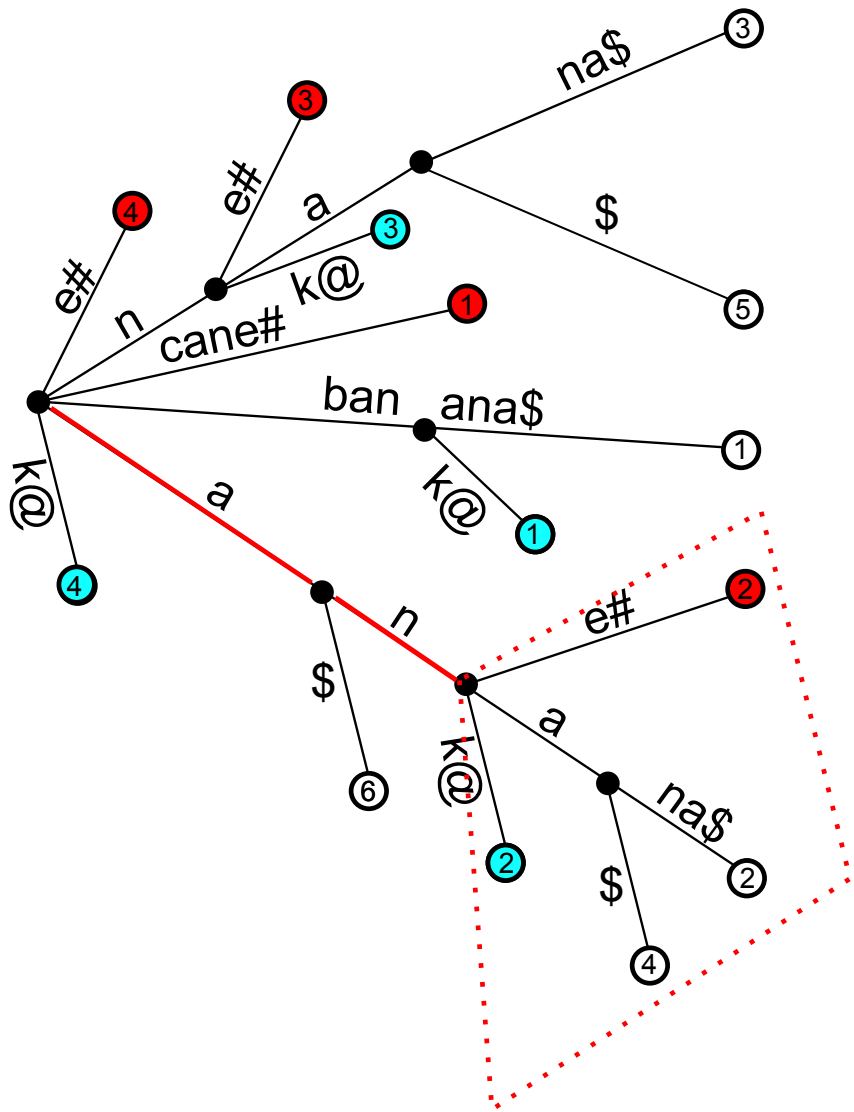
- Даны слова (документы) $T_1, T_2, \ldots, T_m$ суммарной длины n. Найти документы, содержащие слово P.

- Обобщенное СД содержит суффиксы слов $T_1, T_2, \ldots, T_m$. Листы покрашены в m цветов.

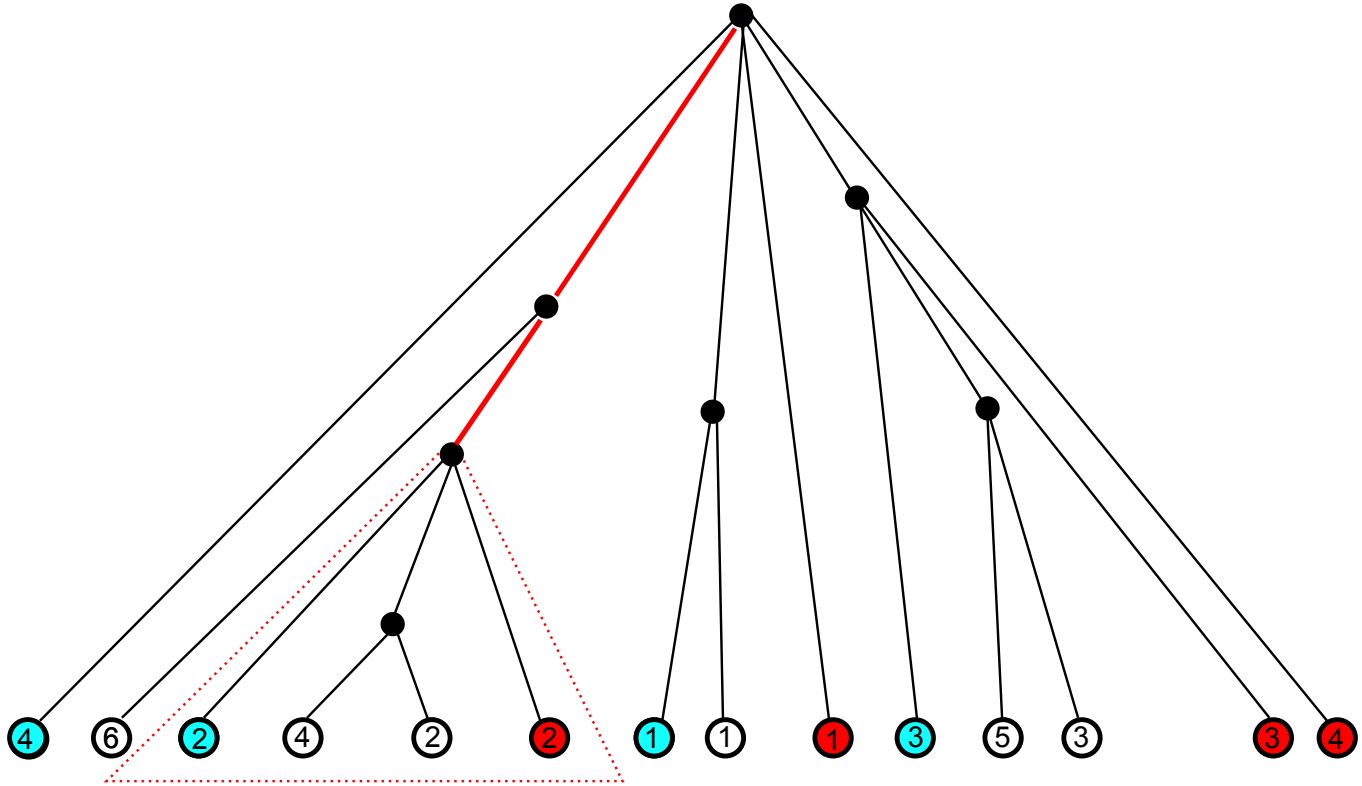
- Пусть v — конец пути из корня с меткой P.

- Документ T_i содержит $P \Leftrightarrow$ в поддереве v есть цвет i

Документы $T_1 = banana$, $T_2 = cane$, $T_3 = bank$, слово $P = an$.



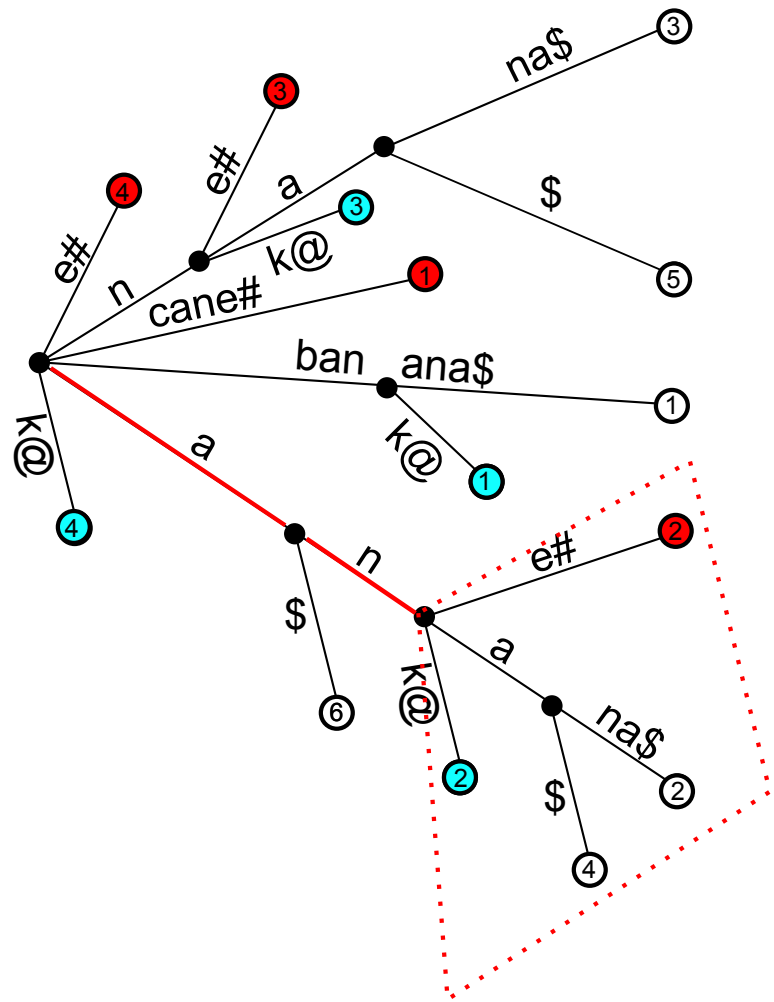
Документы $T_1 = banana$, $T_2 = cane$, $T_3 = bank$, слово $P = an$.



- Дан массив цветов C длины n
- Выдать все различные значения в $C[\ell, r]$
- $Prev[k]$ — предыдущая ячейка массива C цвета $C[k]$
- $k \in [\ell, r]$ — первая ячейка, содержащая цвет $C[k] \Leftrightarrow Prev[k] < \ell$
- Как находить $k \in [\ell, r]$ т.ч. $Prev[k] < \ell$?

- Найти ячейку $Prev[k] \in Prev[\ell, r]$ содержащую минимальное значение ($\mathcal{O}(1)$ времени)
- Если $Prev[k] < \ell$, выдать $C[k]$ и запустить алгоритм для $Prev[\ell, k-1]$ и $Prev[k+1, r]$
- Время работы $= \mathcal{O}(\text{кол-во цветов})$

Документы $T_1 = banana$, $T_2 = cane$, $T_3 = bank$, слово $P = an$.



На поиск документов, содержащих P, требуется $\mathcal{O}(|P| + output)$ времени и $\mathcal{O}(n)$ памяти
[Муту Мутукришнан, 2002]

Максимальные
общие
несовпадениях
о
строки
Сжатие
методу
повторы
по
данных
Лемпеля-Зива
выражения
задача
образца
Суффиксно-префиксные
Наибольшие
Регулярные
совпадения
циклической
Линеаризация
подслова
палиндромы
Вхождения
k

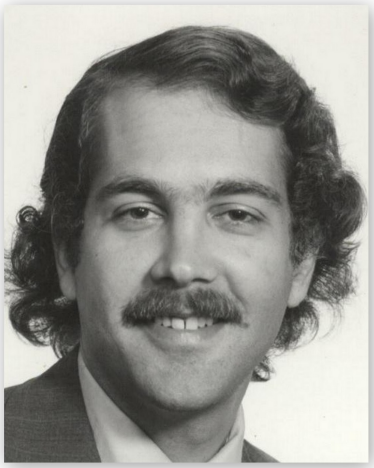
Лекция 1

- Суффиксное дерево слова длины n занимает $\mathcal{O}(n)$ памяти
- Время поиска вхождений P составляет $\mathcal{O}(|P| + \text{кол-во вхождений})$
- Время поиска документов, содержащих P, составляет $\mathcal{O}(|P| + output)$

- Лекция 1: Определение и приложения.
- Лекция 2: Построение.
  [D. Breslauer, G. Italiano. *Near real-time suffix tree construction via the fringe marked ancestor problem.* Best paper SPIRE 2011!]
- Лекция 3: Когда дерево является суффиксным?

Лекция 2: Построение

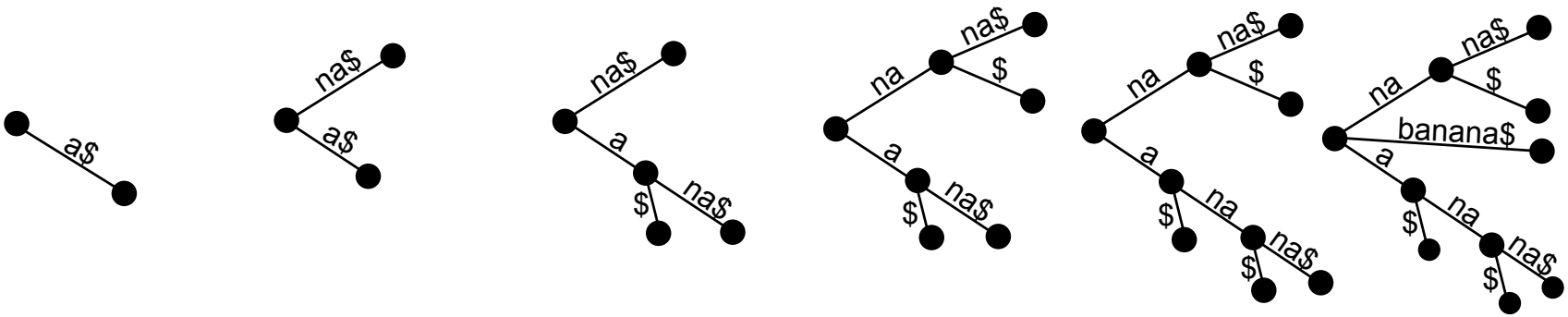
Алгоритмы с линейным временем работы



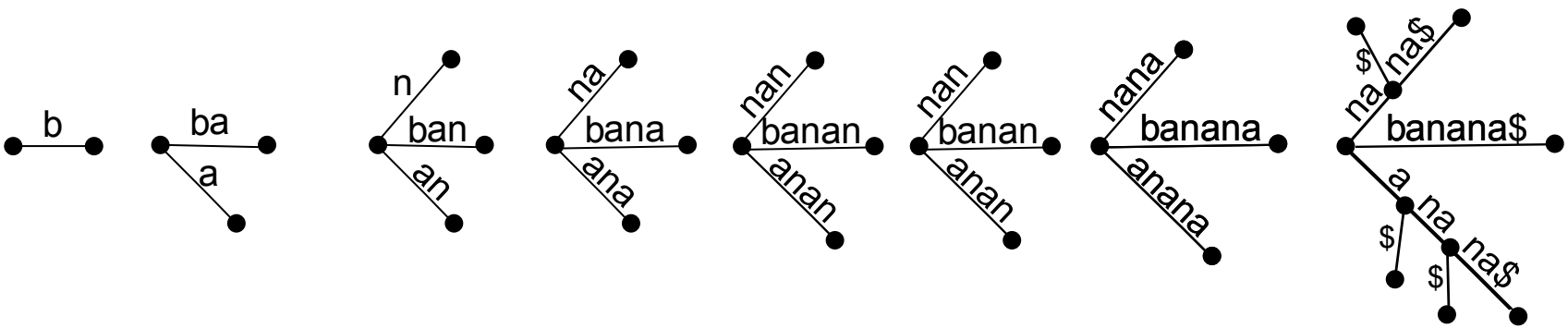

- Алгоритм Питера Вайнера, 1973
- Алгоритм Эварда МакКрейта, 1976
- Алгоритм Эско Укконена, 1996
- Алгоритм Мартина Фарах-Колтона, 1997

$T =$ "*banana*"

- Алгоритм Питера Вайнера, 1973



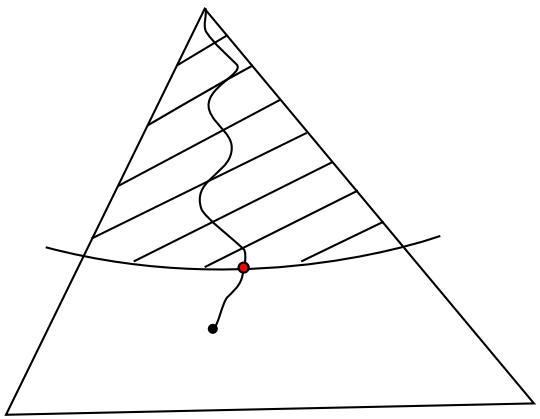
- Алгоритм Эско Укконена, 1996



$\mathcal{O}(1)$ амортизированного времени на букву!

- Цви Копелович и др., 2005 — $\mathcal{O}(\log n)$ времени на букву
- Дэни Бреслауер, Джузеппе Итальяно, 2011 — $\mathcal{O}(\log \log n)$ времени на букву (**сегодня**)
- Открытый вопрос — $\mathcal{O}(1)$ времени на букву

Задача о граничных помеченных предках

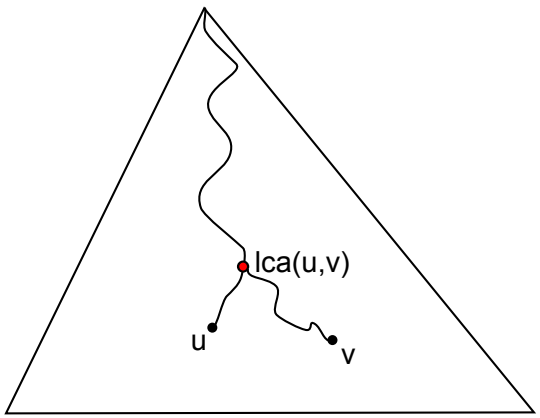


- По вершине x найти ее ближайшего помеченного родственника
- Пометить вершину x (родитель должен быть помечен)
- Добавить вершину x на ребро (u, v)
- Добавить лист x как сына вершины u

Линейная память, $\mathcal{O}(\log logn)$ времени на операцию

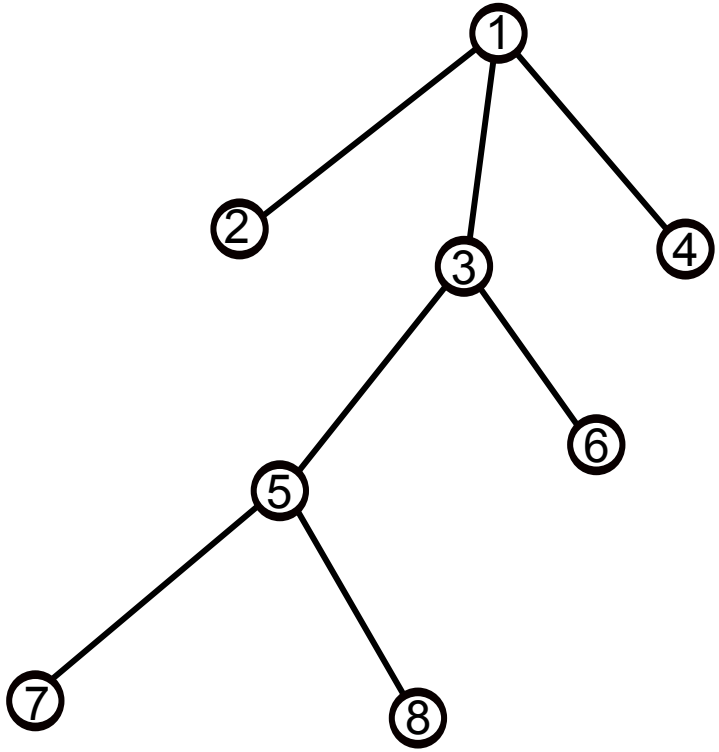
Задача о помеченном предке (общий случай) — $\Omega(\log n / \log \log n)$ времени на операцию!

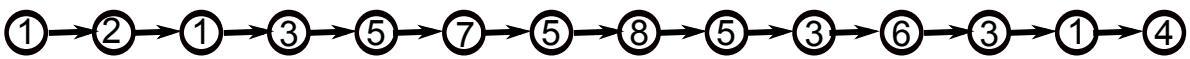
Задача LCA (самый глубокий общий предок)



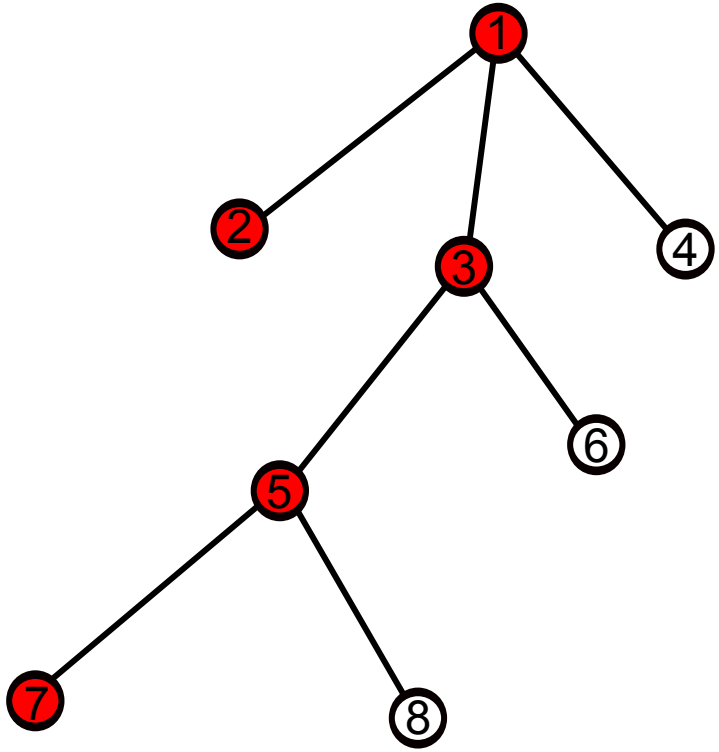
- Найти $lca(u, v)$
- Добавить вершину x на ребро (u, v)
- Добавить лист x как сына вершины u

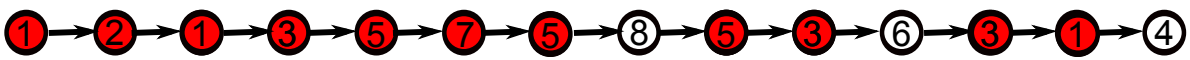
Линейная память, $\mathcal{O}(1)$ времени на операцию
[Коул, Харихаран, 2005]



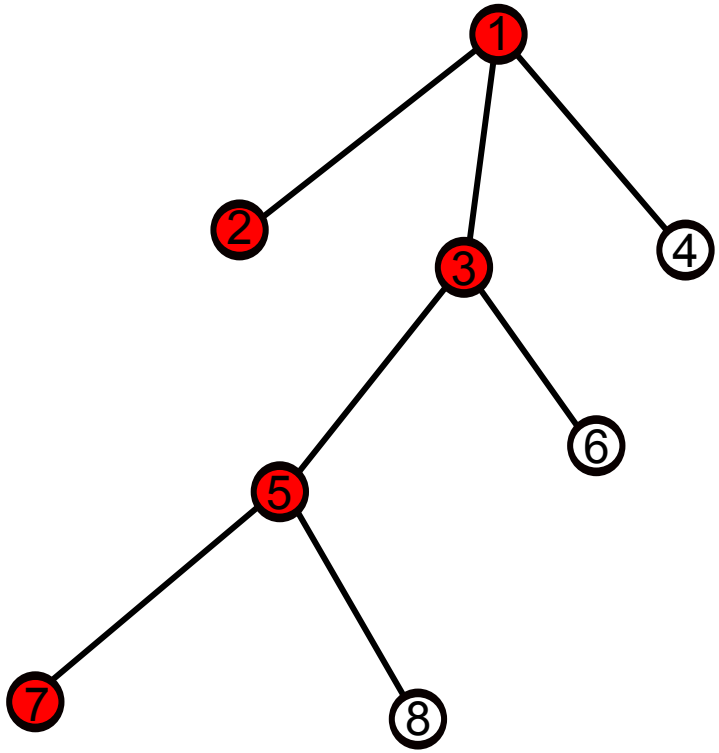


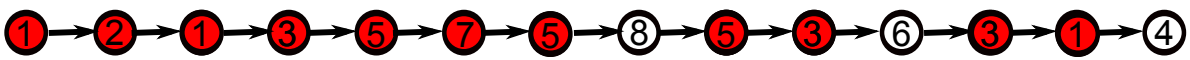
- $T(i)$ — момент первого появления i в обходе





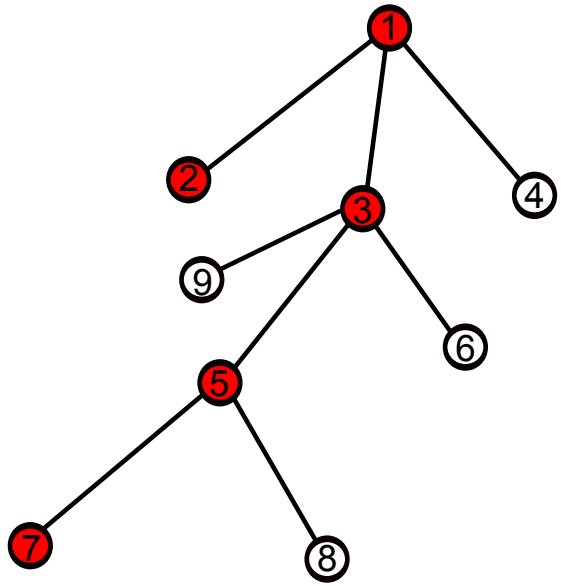
- $T(i)$ — момент первого появления i в обходе
- Задача: по вершине j найти помеченную вершину $i = prev_m(j)$ с $T(i) \le T(j)$, $T(i)$ макс.

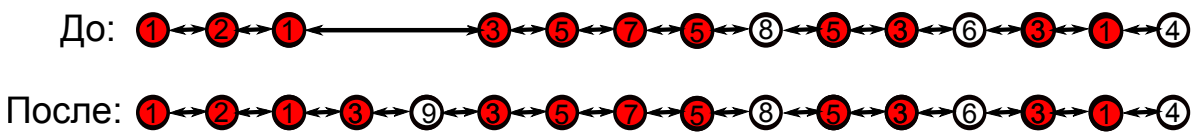




- $T(i)$ — момент первого появления i в обходе
- Задача: по вершине j найти помеченную вершину $i = prev_m(j)$ с $T(i) \le T(j)$, $T(i)$ макс.
- Лемма: $lca(j, prev_m(j))$ — граничный помеченный предок j (на доске)

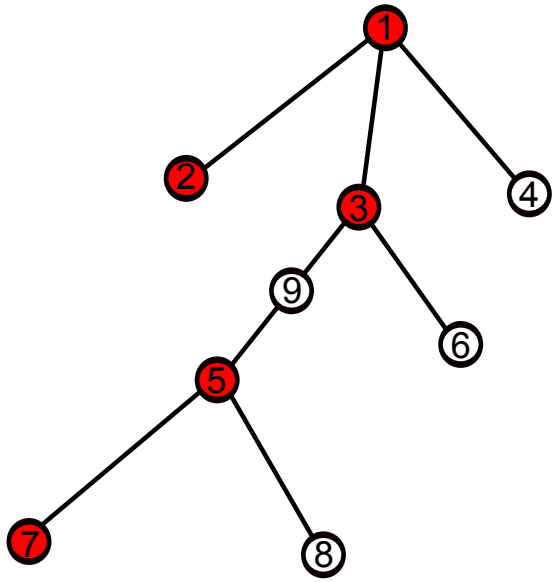
Как меняется список при изменении дерева?

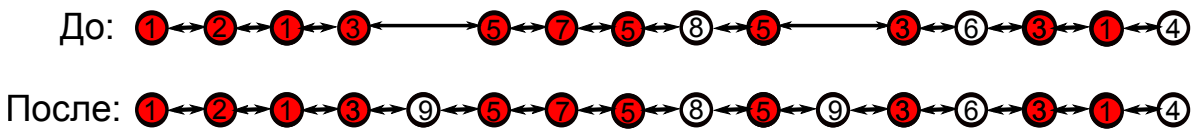




Добавить лист 9 как сына вершины 3

Как меняется список при изменении дерева?





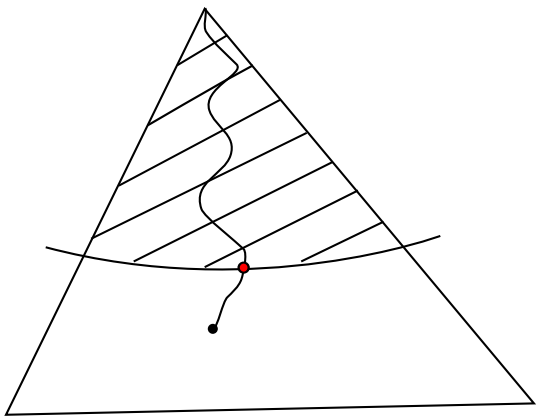
Добавить вершину 9 на ребро $(3, 5)$

Как помечаются вершины и ищется $prev_m(i)$? Для списка поддерживаем структуру Диеца и Рамана (1991) со следующими операциями:

- Добавить элемент x после элемента y
- Пометить элемент x
- Найти помеченного предшественника x

$\mathcal{O}(\log \log n)$ времени на операцию

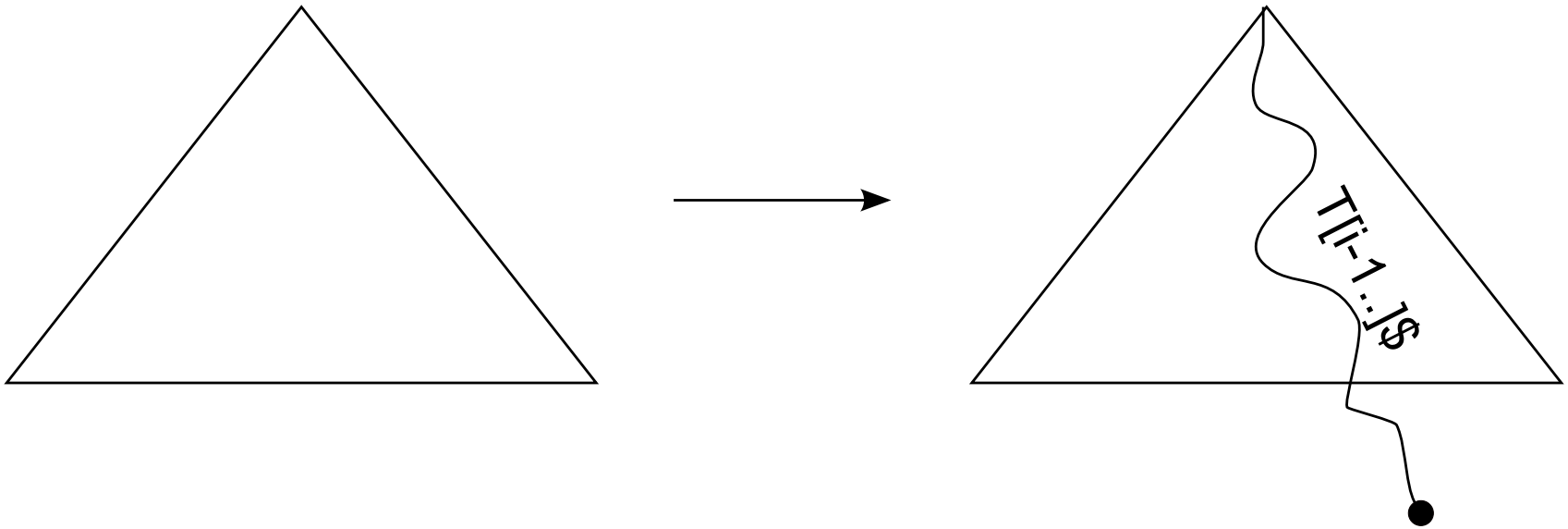
Задача о граничных помеченных предках



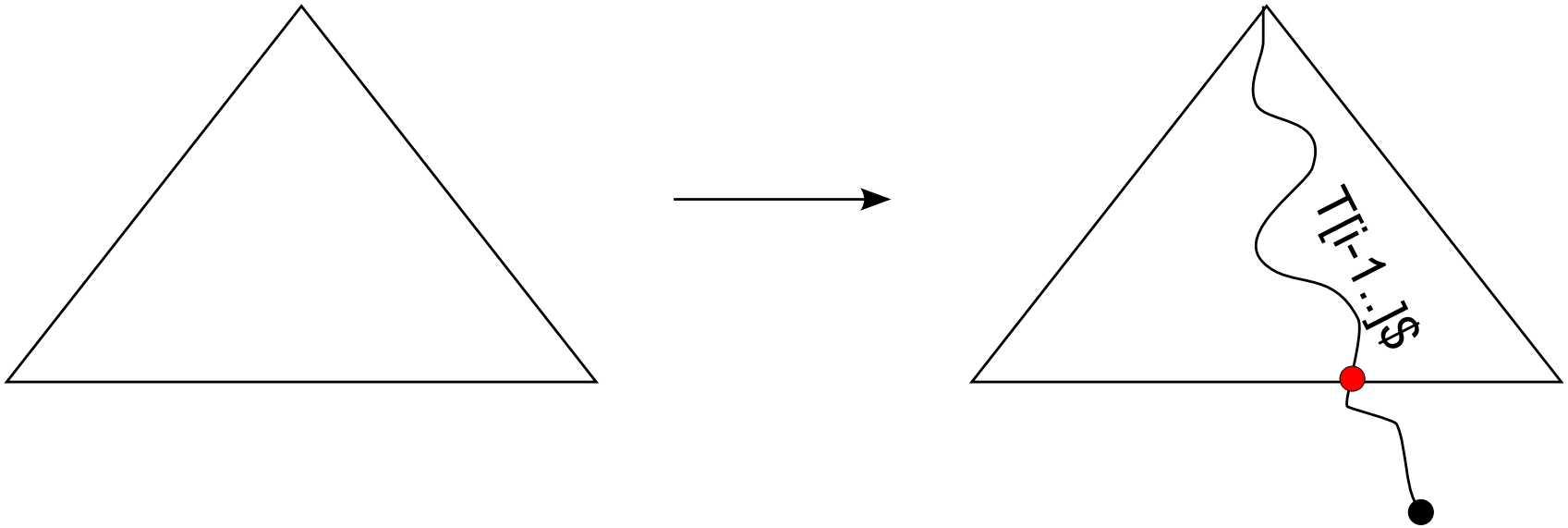
- По вершине x найти ее ближайшего помеченного родственника
- Пометить вершину x (родитель должен быть помечен)
- Добавить вершину x на ребро (u, v)
- Добавить лист x как сына вершины u

Линейная память, $\mathcal{O}(\log logn)$ времени на операцию

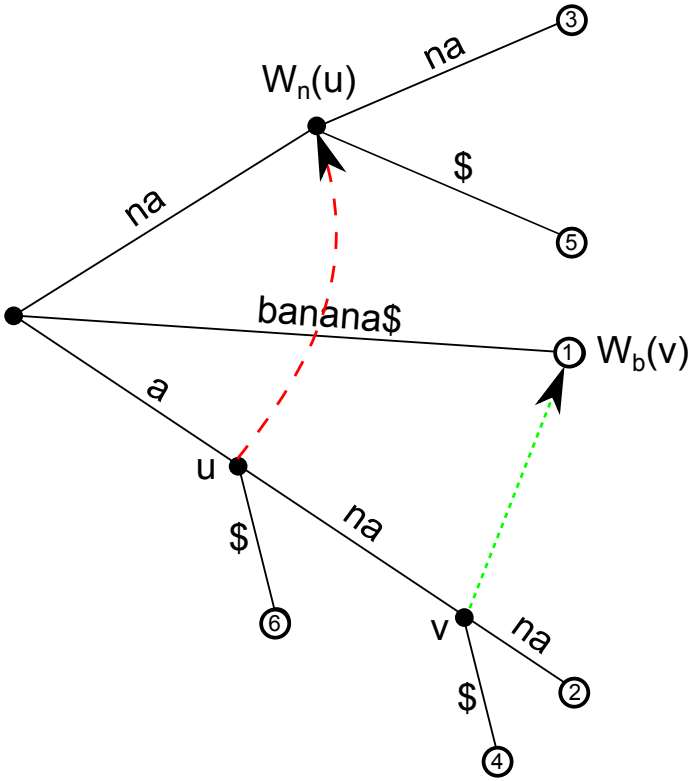
Алгоритм Вайнера. Переход от СД слова $T[i..]\$$ к СД слова $T[i-1..]\$$.



Алгоритм Вайнера. Переход от $T[i..]\$$ к $T[i+1..]\$$.



Суффиксные ссылки Вайнера



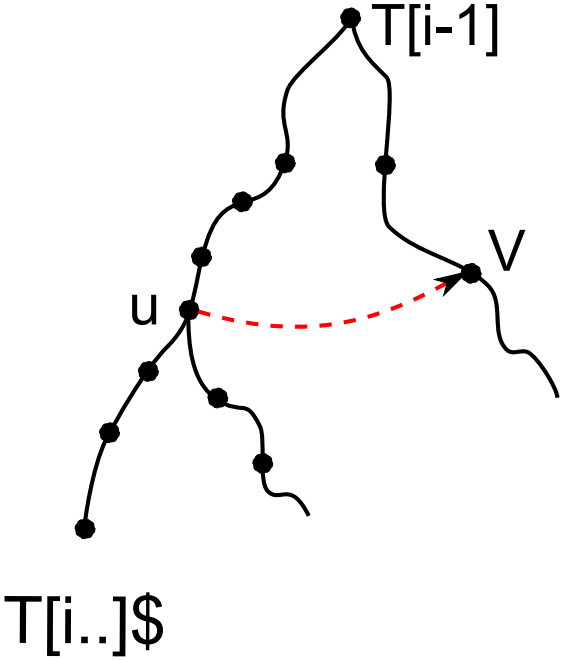
Метка u равна L, метка v равна aL

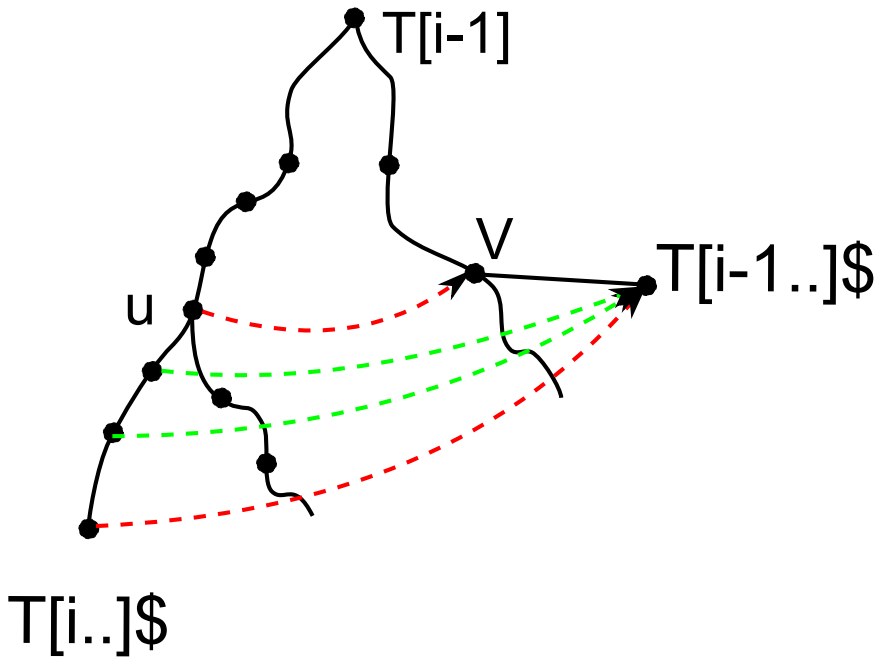
- $W_a(u) = v$, если v — вершина ("зрелые" ссылки)
- $W_a(u)$ — конец ребра содержащего v, если v — позиция на ребре ("непоспевшие" ссылки)

Два свойства суффиксных ссылок

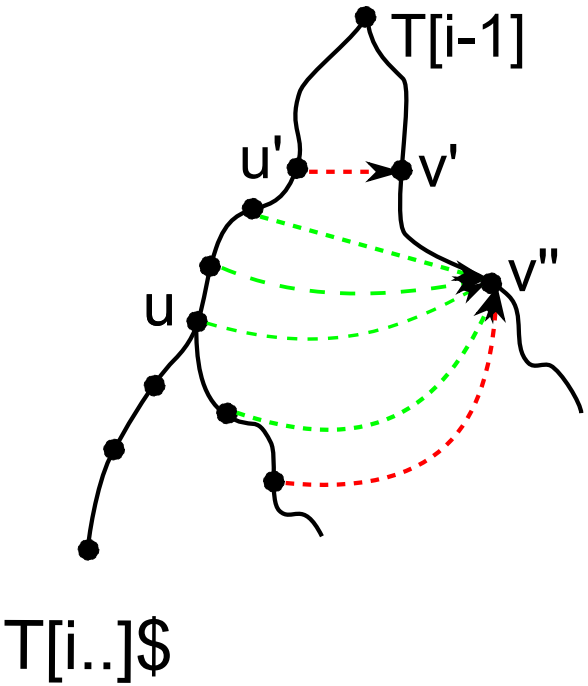
- Если для вершины u определена $W_a(u)$, то и для любого ее предка v определена $W_a(v)$
- Если $v = W_a(u)$, где (u, v) — зрелая ссылка, то $depth(u) \geq depth(v) - 1$

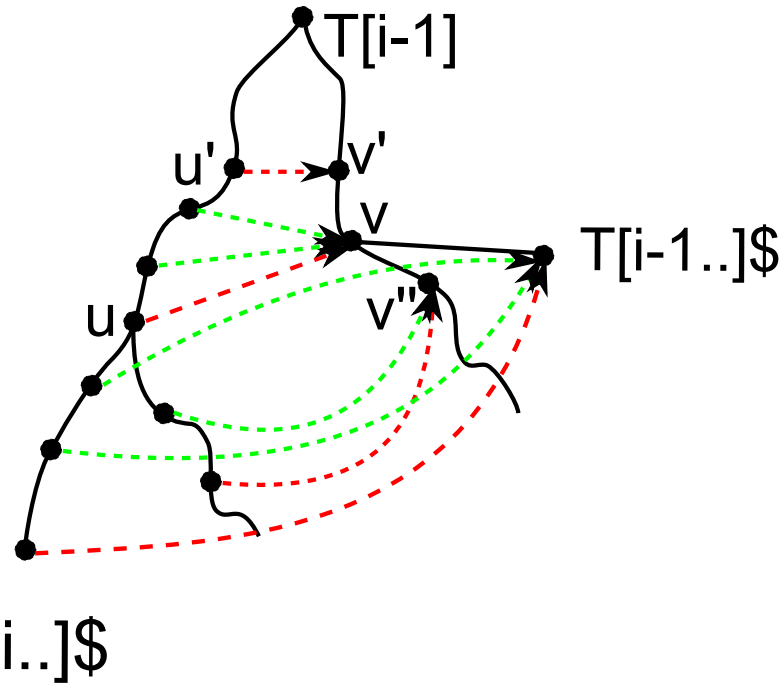
Добавление $T[i-1..]\$$. Случай 1.





Добавление $T[i-1..]\$$. Случай 2.





Количество шагов в случае 1 ограничено

$$depth(T[i..]\$) - depth(u) \leq depth(T[i..]\$) - depth(T[i-1..]\$) + 1$$

Количество шагов в случае 2 ограничено

$$depth(T[i..]\$) - depth(u') \leq depth(T[i..]\$) - depth(T[i-1..]\$)$$

Суммируя, получаем: время работы $= \mathcal{O}(n)$

Количество шагов в случае 1 ограничено

$$depth(T[i..]\$) - depth(u) \leq depth(T[i..]\$) - depth(T[i-1..]\$) + 1$$

Количество шагов в случае 2 ограничено

$$depth(T[i..]\$) - depth(u') \leq depth(T[i..]\$) - depth(T[i-1..]\$)$$

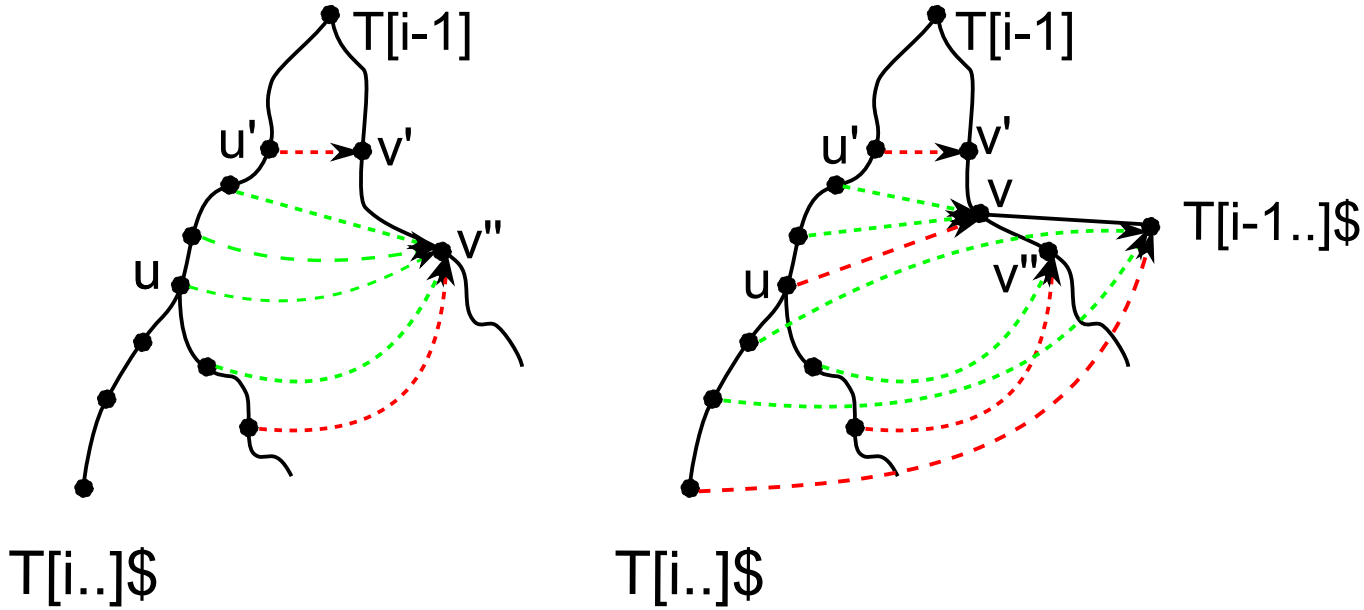
Суммируя, получаем: время работы $= \mathcal{O}(n)$

Небольшая неточность: глубины $T[i..]\$$ в СД $T[i..]$ и СД $T[i-1..]$ могут немного отличаться. К счастью, не более, чем на 1!

$\mathcal{O}(1)$ амортизированного времени на букву. Как получить $\mathcal{O}(\log \log n)$ времени на букву?

- Для каждой буквы a алфавита помечаем вершины u, для которых определены $W_a(u)$
- u находим с помощью задачи о помеченных граничных предках
- Новые вершины и новые зрелые ссылки создаем сразу
- Как поддерживать непоспевшие ссылки?

Непоспевшие ссылки — деамортизация



- $depth(u') \geq depth(v') - 1$
- Обрабатываем две ссылки из стека для каждого i
- Глубины вершин в стеке, для которых надо обновить ссылки, упорядочены по возрастанию
- Суффиксное дерево слова T длины n может быть построено с использованием $\mathcal{O}(\log \log n)$ времени на букву и линейной памяти. [Дэни Бреслауер, Джузеппе Итальяно, 2011]

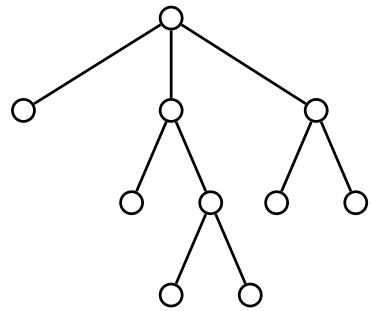
Лекция 2: Построение

- Граничные помеченные предки
- Алгоритм Вайнера, 1973
- Алгоритм Бреслауера-Итальяно, 2011

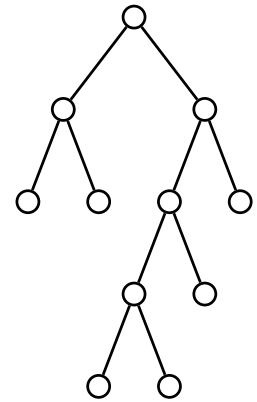
- Лекция 1: Определение и приложения.
- Лекция 2: Построение
- Лекция 3: Когда дерево является суффиксным?
  [Tomohiro I, Shunsuke Inenaga, Hideo Bannai, Masayuki Takeda. *Inferring Strings from Suffix Trees and Links on a Binary Alphabet.* 2011]

Лекция 3: Когда дерево является суффиксным?


$
A
T
$
A
T
$
A
T
$



(a)



(b)

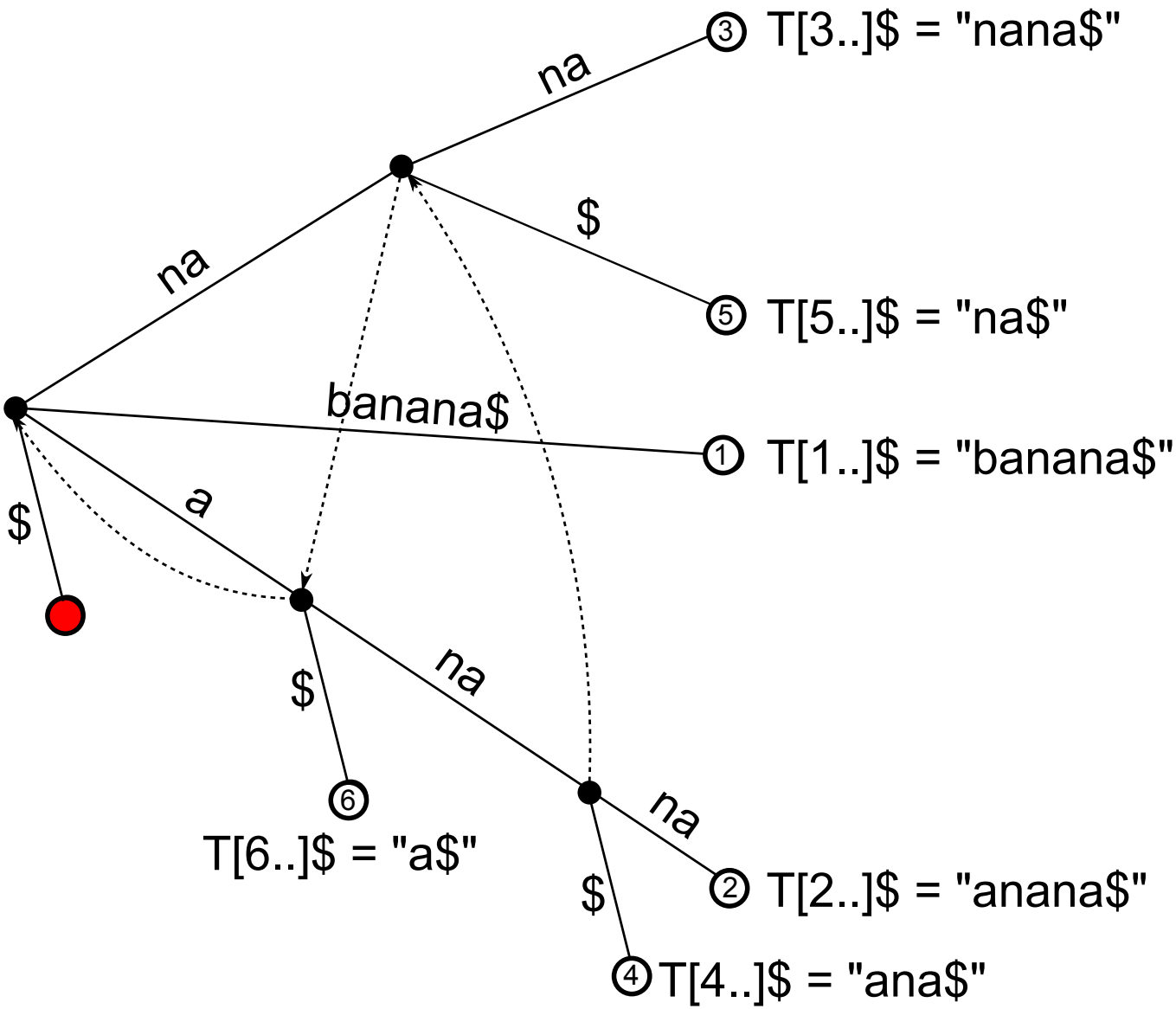
Дерево (a) является суффиксным деревом слова *ababa*, дерево (b) не является суффиксным деревом.

Похожие задачи

- Массив границ
  [Франек и др., 2002]
  [Дюваль и др., 2005]
- Суффиксный массив
  [Дюваль и Лефевр, 2002]
  [Баннаи и др., 2003]
  [Шурман и др., 2005]
  [Кучеров и др., 2013]
- Массив LPF
  [Хе и др., 2011]
- Ориентированный ациклический граф слова
  [Баннаи и др., 2003]
- Функция сдвига в алгоритме КМП
  [Гаврычовски и др., 2010]
- Массив покрытий
  [Крошмор и др., 2010]

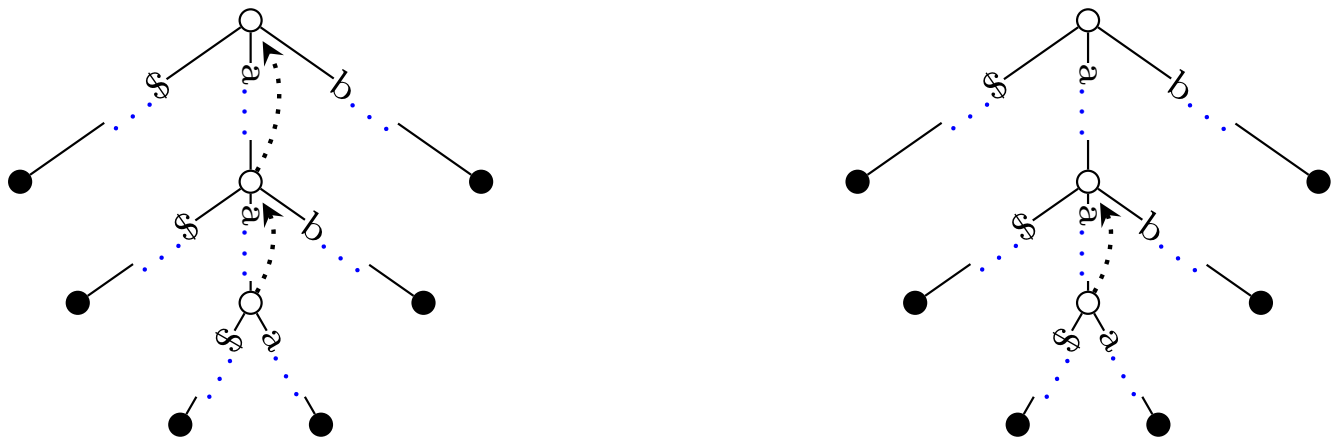
Решение обратных задач приводит к лучшему пониманию структур данных!

Суффиксные ссылки



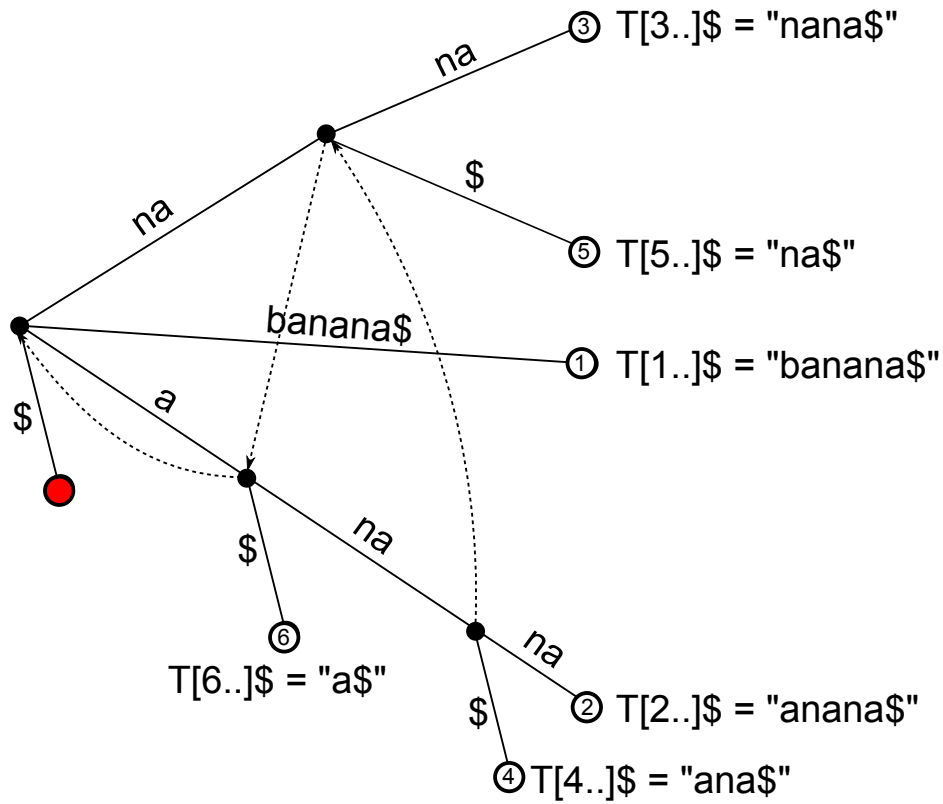
$S(u) = v$, если метка u равна aL, а метка v равна L

Дано дерево, суффиксные ссылки, первые буквы меток.
Существует ли такое суффиксное дерево?



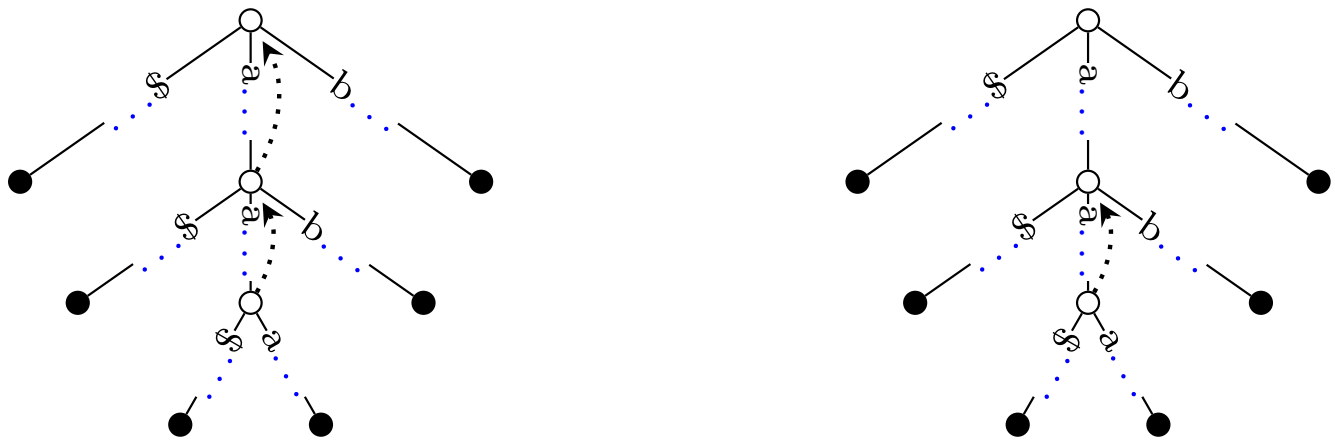
Дерево слева — да (*abaaa*), дерево справа — нет

Необходимые свойства



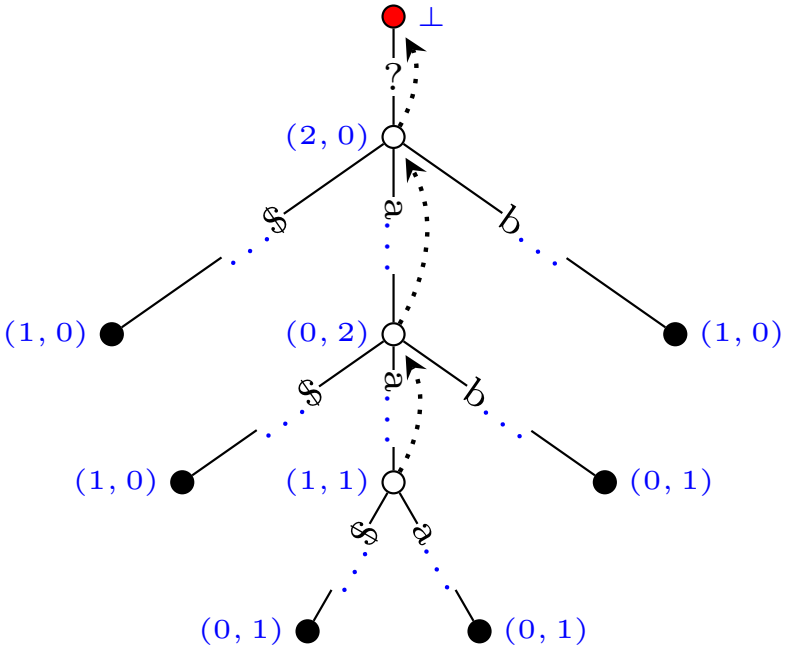
- Из любой внутренней вершины в корень ведет единственный путь, состоящий из суффиксных ссылок
- (u, v) — суффиксная ссылка, u' — сын u, на ребре (u, u') написана буква $a \Rightarrow$ у v есть сын v': на ребре (v, v') написана буква a, а конец суффиксной ссылки, выходящей из u', принадлежит поддереву с корнем в v'

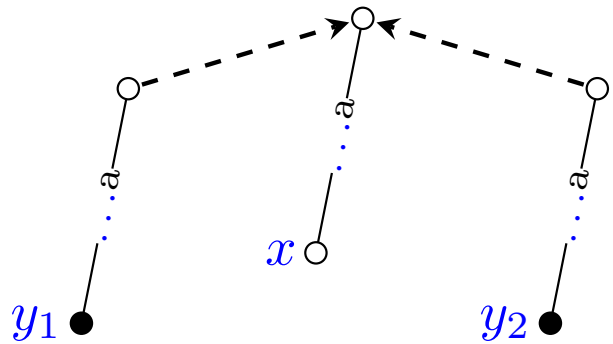
Дано дерево, суффиксные ссылки, первые буквы меток.
Существует ли такое суффиксное дерево?



Дерево слева — да (*abaaa*), дерево справа — нет

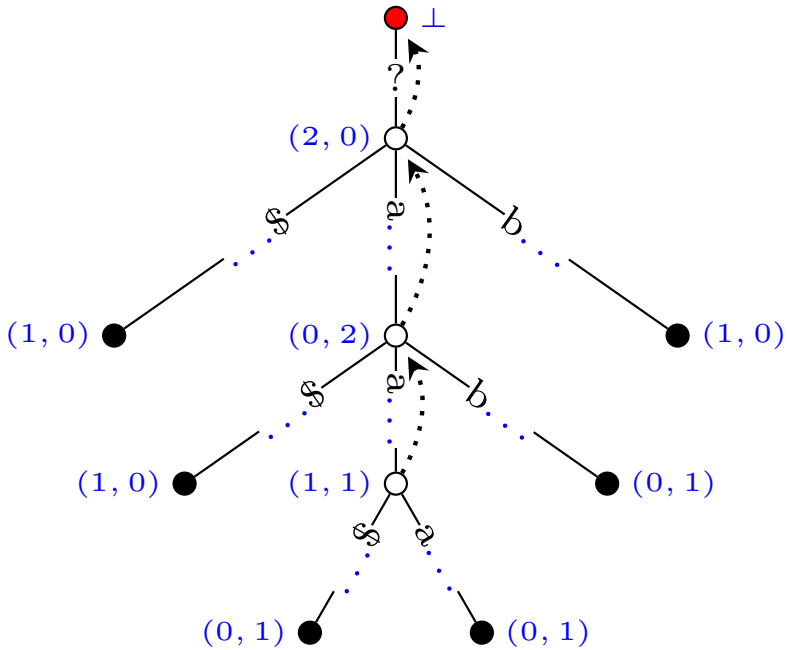
Суффиксный граф

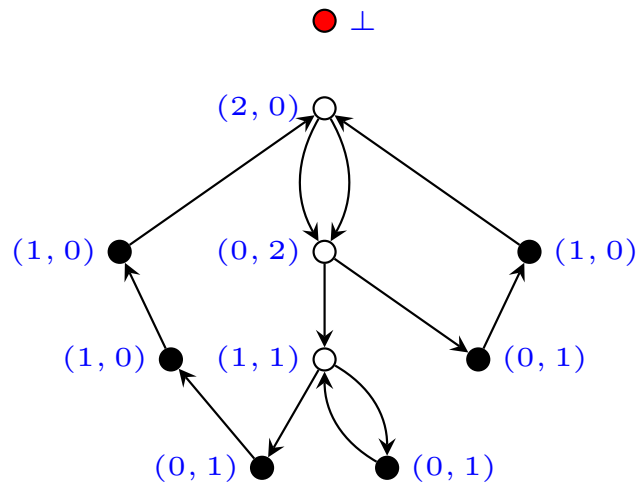




- $\ell(x)$ — количество листьев y: суффиксная ссылка из $parent(y)$ ведет в $parent(x)$ и первые буквы на ребрах $(parent(x), x)$ и $(parent(y), y)$ равны
- $d(x) = |L_x| - \sum_{y \in V_x} \ell(y)$
- $\forall x \;\; d(x) \geq 0$

Суффиксный граф

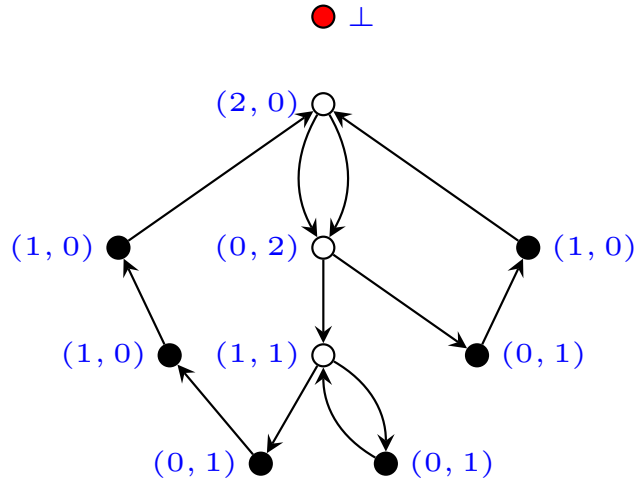




$$E_G = \{(y,x)^k \mid (y,x) \in E, k = d(x)\} \cup \tag{1}$$

$$\{(y,x) \mid y \text{ — лист, вносящий вклад в } \ell(x)\} \tag{2}$$

Свойства



Суффиксный граф —- Эйлеров граф

- x — лист $\Rightarrow \ell(x) = 1 - d(x) \Rightarrow$ в x входит одно ребро, выходит одно ребро.
- x — внутренняя вершина $\Rightarrow$ входит $\ell(x) + d(x)$ ребер, выходит $\sum_{z \in children(x)} d(x)$

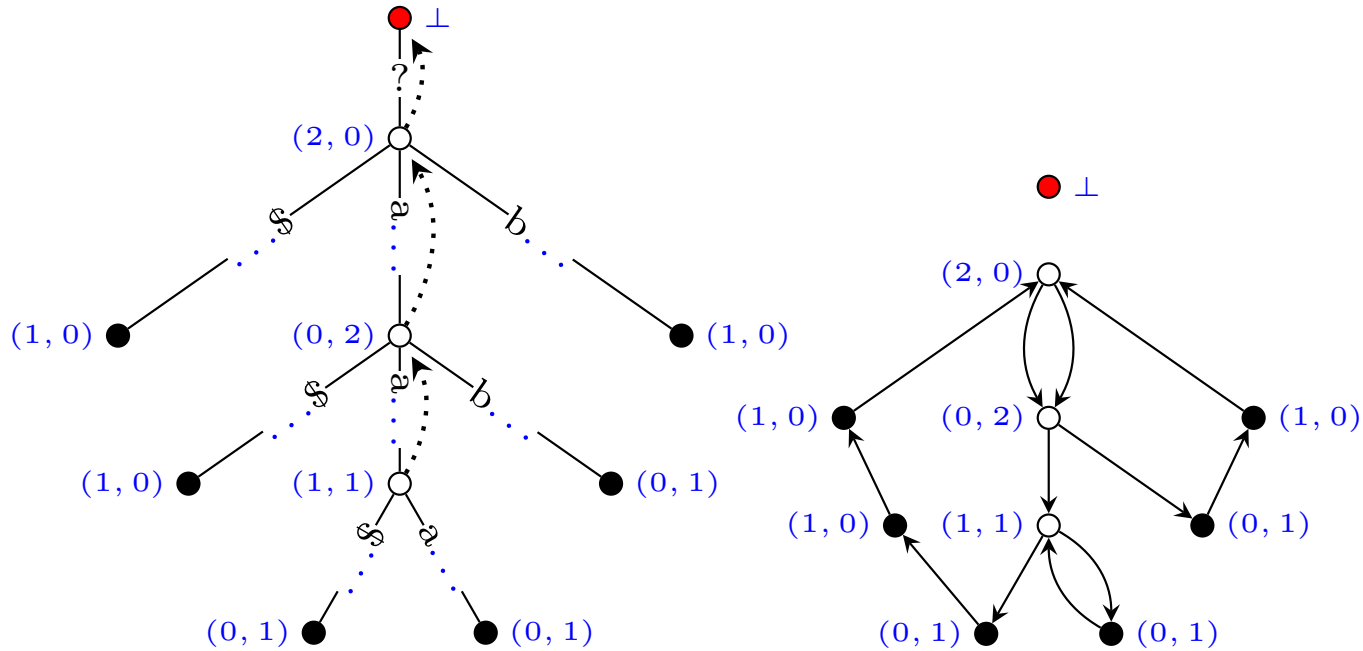
Основная лемма

Пусть выполняются три условия:

- Из любой внутренней вершины в корень ведет единственный путь, состоящий из суффиксных ссылок
- (u, v) — суффиксная ссылка, u' — сын u, на ребре (u, u') написана буква $a \Rightarrow$ у v есть сын v': на ребре (v, v') написана буква a, а конец суффиксной ссылки, выходящей из u', принадлежит поддереву с корнем в v'
- $\forall x\ d(x) \geq 0$

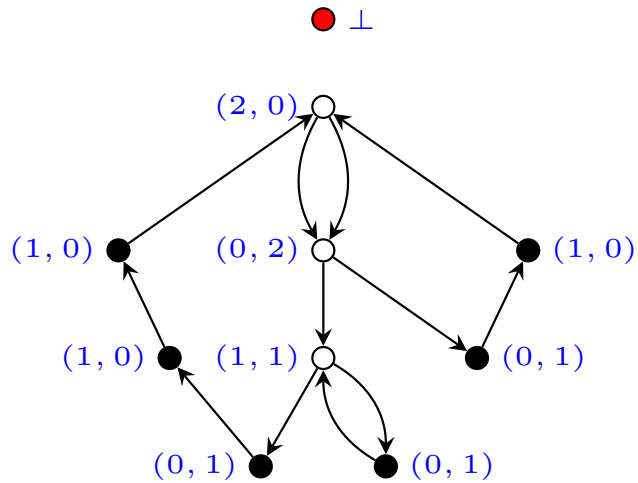
1) Дерево является суффиксным тогда и только тогда, когда суффиксный граф содержит цикл, проходящий через корень и все листы дерева.

2) $T[i]$ — первая буква на пути от корня до листа с номером i



$\mathrm{T}[1] = \mathrm{a},\ \mathrm{T}[2] = \mathrm{b},\ \mathrm{T}[3] = \mathrm{a},\ \mathrm{T}[4] = \mathrm{a},\ \mathrm{T}[5] = \mathrm{a} \Rightarrow \mathrm{T} = \mathrm{abaaa}$

Размер суффиксного графа — $\mathcal{O}(n)$



- $sldepth(x)$ — длина пути по суффиксным ссылкам из x в корень
- y — родитель $x \Rightarrow sldepth(x) \geq sldepth(y) + 1$
- (u, v) образовано из ребра $\Rightarrow sldepth(v) - sldepth(u) < 0$
- (u, v) образовано из суффиксной ссылки $\Rightarrow$ $sldepth(v) - sldepth(u) = 1$

Бинарный алфавит. Восстановление первых букв.

Простые наблюдения:

- Если (u, v) — суфф. ссылка и степени u, v равны, то первые буквы на выходящих из них ребрах совпадают
- Степень любой вершины 2 или 3
- Если у вершины три сына, то на ребре к первому сыну написан \$
- Если у вершины u три сына, то у вершины $S(u)$ три сына

Бинарный алфавит. Восстановление первых букв.

Не такие простые наблюдения:

- Если у вершины u три сына, то у вершины $S(u)$ три сына
- Две вершины степени 3 не могут ссылаться на одну и ту же вершину
- Существует ≤ 1 вершины, на которую не ссылается другая вершина степени 3
- На эту вершину может ссылаться ≤ 2 вершины степени 2
- Если вершина степени 2 ссылается на любую другую вершину степени 3, то для нее первые буквы определены

Бинарный алфавит. Восстановление первых букв.

Вариативность только в первых буквах вершин степени 2, ссылающихся на последнюю вершину степени 3:

- (a, b), (a, b)
- ($, a), (a, b)
- ($, b, a, b)
- (a, b), ($, a)
- (a, b), ($, b)

$\Rightarrow$ 5 возможных разметок

Лекция 3: Когда дерево является суффиксным?

- Когда дерево является суффиксным деревом?
- Бинарный алфавит
- Когда дерево является неявным суффиксным деревом?

Спасибо!

Литература

- Дэн Гасфилд. *Строки, деревья и последовательности в алгоритмах.*
- Peter Weiner. *Linear pattern matching algorithms.*
- Lucas Chi Kwong Hui. *Color set size problem and applications to string matching.*
- S. Muthu Muthukrishnan. *Efficient Algorithms for Document Retrieval Problems.*
- Dany Breslauer, Guiseppe Italiano. *Near real-time suffix tree construction via the fringe marked ancestor problem.*
- Tomohiro I, Shunsuke Inenaga, Hideo Bannai, Masayuki Takeda. *Inferring strings from suffix trees and links on a binary alphabet.*

25th Annual Symposium on Combinatorial Pattern Matching

CPM

JUNE 16-18, 2014 / SUBMISSION DEADLINE: January 10, 2014

Moscow
RUSSIA

KEYNOTE SPEAKERS:

Alberto Apostolico (Georgia Tech and IASI-CNR)
Maxime Crochemore (King's College London)
Zvi Galil (Georgia Tech)
Udi Manber (Google)

ANNIVERSARY LECTURE:

Gene Myers (Max Planck Institute)