1. СУФФИКСНЫЕ ДЕРЕВЬЯ

Пусть $S = S[1..n]$ — слово в алфавите Σ и символ \$ не входит в Σ. Тогда в слове $S\$$ никакой суффикс не является префиксом другого суффикса.

Определение 1. *Суффиксным деревом для $S\$$ называется такое дерево T, у которого имеется $n+1$ лист, занумерованный числами от 1 до $(n+1)$, все дуги T помечены подсловами слова $S\$$, из каждой внутренней вершины выходят хотя бы две дуги, метки всех выходящих из одной вершины дуг, начинаются с разных букв и в каждый лист j из корня ведет путь, на котором "написан"суффикс $S[j..n]\$$.*

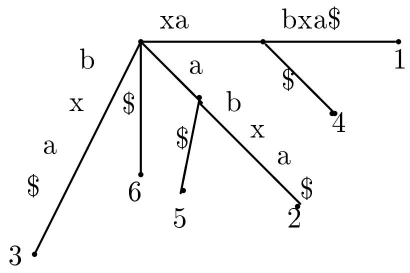
Из этого определения непосредственно следует, что число вершин в T не превосходит $2n$. Но общая длина всех меток T может быть квадратична, т.е. $O(n^2)$. Линейное представление дерева получится, если для каждого ребра его метку-подслово $S[j..k]$ заменить на пару чисел (j, k). Описанный ниже алгоритм будет работать с метками-словами, но его можно легко преобразовать в алгоритм, работающий с метками-парами индексов (как?).

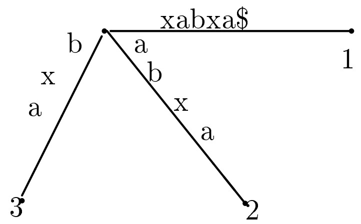
1.1. **Алгоритм Укконена для построения суффиксного дерева (Ukkonen - 1995).**

Определение 2. *Неявное суффиксное дерево для слова S — это дерево, которое получается из суффиксного дерева для $S\$$ следующим путем:*
1) из меток всех дуг удаляется символ \$;
2) удаляются все ребра, оставшиеся без меток;
3) удаляются вершины, у которых остался 1 сын.

Неявное суффиксное дерево для префикса $S[1..i]$ слова S получается таким же образом из суффиксного дерева для слова $S[1..i]\$$. Обозначим это дерево через I_i.

На рисунке ниже показаны суффиксное дерево T для слова $S\$ = xabxa\$$ и неявное суффиксное дерево T' для слова $S = xabxa$.





Вначале опишем алгоритм, который будет строить последовательность неявных суффиксных деревьев $I_1, I_2, \ldots, I_n$ по слову $S[1..n]$. Алгоритм работает фазами – на фазе $i+1$ по дереву I_i строится дерево I_{i+1}.

Общая схема алгоритма

1. Построить I_1.
2. **FOR** $i = 1$ **TO** $n-1$ **DO**
   /* Фаза $i+1$:
3. **FOR** $j = 1$ **TO** $i+1$ **DO**
   /* Расширение j:

4. Найти в текущем дереве конец пути, помеченного $S[j..i]$
и, если требуется, расширить этот путь, добавив $S[i+1]$.

Пусть $\beta = S[j..i]$. Уточним алгоритм расширения суффикса.

Расширение суффикса

Случай 1: *Путь β в текущем дереве заканчивается в листе:*

Добавить символ $S(i+1)$ к метке дуги, ведущей в этот лист;

Случай 2: *Никакой путь, продолжающий β не начинается с $S(i+1)$, но хоть одно продолжение у пути с β имеется:*

Если β заканчивается внутри дуги (u, w), то создать новую вершину v "между"u и w, в которую из корня ведет путь, помеченный β.
Иначе пусть v — это вершина, в которой заканчивается β.
Создать новый лист с меткой позиции j и дугу из v в него, с меткой $S(i+1)$;

Случай 3: *Имеется путь, продолжающий β символом $S(i+1)$:*

Ничего не делать.

Ясно, что этими тремя случаями исчерпываются все возможности. При "наивной"реализации алгоритма на поиск конца β в текущем дереве требуется $|\beta|$ шагов, после этого реализация расширения выполняется за $O(1)$ шагов. Тогда i-я фаза может быть выполнена за время $O(n^2)$, а весь алгоритм за время $O(n^3)$.

Определение 3. *Суффиксной ссылкой для внутренней вершины дерева v, в которую ведет путь, помеченный $x\alpha$ ($x \in \Sigma, \alpha \in \Sigma^*$) называется пара вершин ("дуга") $(s, s(v))$ такая, что к вершине $w = s(v)$ ведет путь, помеченный α. В частности, если $\alpha = \varepsilon$, то в качестве $s(v)$ выступает корень дерева.*

Лемма 1. *Если внутренняя вершина v с путем-меткой $x\alpha$ создается при расширении j на фазе $i+1$, то либо путь, помеченный α, уже заканчивается в некоторой вершине дерева, либо вершина, в которую ведет путь α, будет создана при следующем расширении $j+1$ на той же фазе $i+1$.*

Следствие. *Во всяком неявном суффиксном дереве I_i для любой внутренней вершины v, в которую ведет путь с меткой $x\alpha$, существует вершина $s(v)$, в которую ведет путь α.*

Алгоритм SEA для расширения j

*1. Зная конец p (p не обязательно вершина!) предыдущего расширения $S[j-1..i]$, найти первую вершину v на пути из p в корень, для которой существует суффиксная ссылка $s(v) = w$. Если такой v нет, то положить $v := w :=$*корень дерева. *Пусть γ — это метка пути из v в p.*

2. "Спуститься"из w по пути, помеченному γ.

3. Применить к найденной точке алгоритм расширения суффикса $S[j..i]S(i+1)$.

4. Если в предыдущем расширении $j-1$ была создана новая вершина и с путем-меткой $x\alpha$, , то по лемме 1 путь α теперь должен вести в некоторую вершину $s(u)$. Тогда создать суффиксную ссылку $(u, s(u))$.

Трюк 1. *Подсчитать - пропустить*

Пусть $|\gamma| = g$ и известны длины слов-меток для всех ребер текущего дерева. Тогда при поиске пути из $s(v)$ по γ можно найти следующую вершину за $O(1)$ шагов:
a) $h := 1;\ w := s(v)$;
b) Найти выходящее из w ребро (w, u) с первым символом = h-му символу γ;
c) Пусть g' — длина метки этого ребра. Если $g' < g$, то $w := u; g := g - g'; h := h + g'$; снова перейти к пункту (b);
d) Если $g \leq< g'$, то выдать $(w, u), g$ (это означает, что γ заканчивается на g-ом символе ребра (w, u)).

Лемма 2. *Пусть $(v, s(v))$ — суффиксная ссылка в текущем дереве, по которой в алгоритме осуществляется переход. Тогда глубина(v) $\leq$ глубина$(s(v)) + 1$.*

Теорема 1. *С использованием трюка 1 каждую фазу алгоритма можно выполнить за время $O(n)$.*

Трюк 2. *Заканчивать фазу $i+1$ как только при некотором расширении будет иметь место Случай 3.*
Действительно, легко понять, что если для расширения j имеет место случай 3, то для всех последующих расширений $j+1, j+2, ...$ той же фазы тоже будет выполнен случай 3.

Заметим, что после создания листа с меткой позиции j, эта вершина останется листом и для нее всегда будет выполняться случай 1. Обозначим через j_i номер последнего расширения фазы $(i+1)$, в последовательности расширений со случаями 1 или 2. Этот номер не убывает: $j_i \leq j_{i+1}$ (почему?).

Трюк 3. *На фазе $i+1$ при создании ребра с меткой $S[j..(i+1)]$, ведущего в лист, вместо пары индексов $(j, i+1)$ приписывать этому ребру пару (j, e), где e — символ переменной, принимающей на каждой фазе $i+1$ значение $i+1$.*

Уточним теперь алгоритм выполнения одной фазы алгоритма.

Фаза $i+1$.
1. $e := i+1$;
2. Последовательно применять единичные расширения SEA, начиная с $j = j_i + 1$ до первого расширения j^*, для которого выполнится случай 3, а если случай 3 не наступит, то до конца фазы, т.е. до $j^* = i+1$.
3. $j_{i+1} := j^* - 1$.

Теорема 2. *С использованием суффиксных связей и трюков 1-3 алгоритм Укконена строит последовательность неяных суффиксных деревьев $I_1, \ldots, I_n$ за время $O(n)$.*

Как получить явное суффиксное дерево? Для этого достаточно применить алгоритм Укконена к слову $S\$$, а затем заменить в метках ребер символ переменной e на число $n+1$ - номер последнего символа \$. Таким образом, справедлива

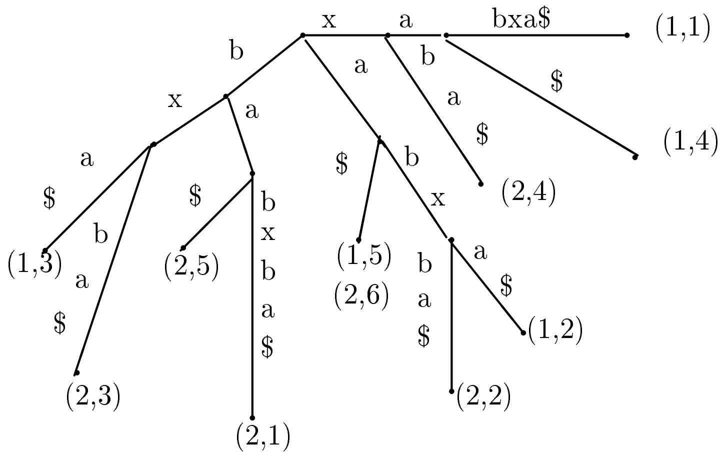
Теорема 3. *С помощью алгоритма Укконена можно построить суффиксное дерево (со всеми суффиксными ссылками) для слова $S\$$ за время $O(|S|)$.*

Объединенное суффиксное дерево для множества слов

Пусть задано множество слов $\{S_1, \ldots, S_M\}$ общей длиной n, $\sum_{i=1}^{M} |S_i| = n$. Обобщим понятие суффиксного дерева для множества слов так, чтобы в объединенном дереве были представлены суффиксы всех слов из заданного множества.

Определение 4. *Объединенным суффиксным деревом для $\{S_1, \ldots, S_M\}$ называется такое дерево T, у которого имеется $\leq n$ листьев, каждому из которых приписана одна или несколько меток вида $(i, j), 1 \leq M, 1 \leq |S_i|$, причем каждая метка такого вида приписана какому-либо листу, все дуги T помечены подсловами слов $S_i\$$, из каждой внутренней вершины выходят хотя бы две дуги, метки всех выходящих из одной вершины дуг, начинаются с разных букв и, если листу приписана метка (i, j), то из корня в него ведет путь, на котором "написан"суффикс слова $S_i[j..n]\$$.*

На следующем рисунке представлено объединенное суффиксное дерево для двух слов: $S_1 = xabxa$ и $S_2 = babxba$.



Задача 1. Модифицируйте алгоритм Укконена так, чтобы он строил объединенное суффиксное дерево для множества слов общей длины n за время $O(n)$.

Замечание. Метки дуг в объединенном дереве имеют вид $(i; p, q)$, где i — номер слова во множестве, а p и q начальная и заключительная позиция подслова-метки в слове S_i.

Задача 2. Предположим, что к заданному множеству слов добавляется новое слово или из него удаляется некоторое слово. Предложите алгоритмы для эффективного выполнения этих операций на объединенном суффиксном дереве.

1.2. **Применения суффиксных деревьев.**

1. Поиск всех вхождений подслова

Чтобы найти все вхождения образца-слова P длины m в слово S длины n, построим суффиксное дерево T для S, а затем пройдем в T по единственному пути, помеченному символами P. Если на этом пути лист T встретится раньше, чем закончится P, то P не входит в S. Если же P в вершине v или на дуге, ведущей в эту вершину, то все вхождения P в S — это метки листьев, из поддерева с корнем в вершине v (почему?). Время работы этого алгоритма $O(n+m)$.

2. Поиск всех вхождений слов-образцов

Пусть задано множество слов-образцов $\mathcal{P} = \{U_1, \ldots, U_k\}$, суммарная длина которых равна n. Требуется найти все вхождения этих образцов в текст S длины m. Докажите, что это можно сделать с использованием суффиксного дерева за время $O(n+m+r)$, где r — общее число искомых вхождений.

3. Поиск вхождений слов в базу данных

Пусть база данных содержит слова общей суммарной (большой!) длиной n. Требуется так организовать базу данных, чтобы эффективно по слову-запросу P выяснить входит ли оно в базу данных, а если не входит, то определить, се слова БД, для которых оно является префиксом, а если и таких слов нет, то найти самый длинный префикс P является префиксом какого-либо слова в базе данных.

Для решения этой проблемы можно организовать суффиксное дерево для конкатенации слов БД, разделенных новым символом за время $O(n)$, а затем отвечать на любой запрос длины m за время $O(n+k)$, где k — число слов БД, в которые входит P.

4. Наибольшее общее подслово для двух слов

Задача: по двум словам S_1 и S_2 найти самое длинное слово P, которое входит как подслово и в S_1, и в S_2.

Докажите, что с использованием суффиксных деревьев эту задачу можно решить за время, линейное от $(|S_1| + |S_2|)$.

5. Проблема загрязненной ДНК

Задача: для заданного слова S_1 (новая цепочка ДНК) и известного слова S_2 (в нем объединены различные источники загрязнений) найти все подслова S_2 длины не меньше l, входящие в S_1. Эти подслова являются возможными нежелательными частями S_2, которые могут загрязнять ДНК.

Докажите, что с использованием суффиксных деревьев и эту задачу можно решить за время, линейное от $(|S_1| + |S_2|)$.

6. Общие подстроки для множества строк

Пусть дано множество W, состоящее из K различных строк (слов) суммарной длины n. Для каждого i от 2 до K определим $l(i)$ как длину самой длинной подстроки, входящей в не менее чем в i строк W. Задача состоит в том, чтобы построить таблицу из $(K-1)$-й строки, в которой для каждого i указано число $l(i)$ и какая-нибудь строка длины $l(i)$, входящая в $\geq i$ строк из W.

Построим объединенное суффиксное дерево T для множества W, в котором каждый лист имеет метку слова, суффикс которого в этом листе заканчивается. Для каждой внутренней вершины v обозначим через $C(v)$ число различных слов из W, суффиксы которых заканчиваются в поддереве с корнем v.

6.1. Покажите, как, зная значения $C(v)$ для всех внутренних вершин, построить требуемую таблицу за время $O(n)$.

6.2. Предложите алгоритм вычисления значений $C(v)$ для всех внутренних вершин T за время $O(Kn)$. (Имеется и более эффективный алгоритм, решающий эту задачу за время $O(n)$.)

7. Повторяющиеся структуры в слове

Определение 5. Максимальная пара *в слове S — это пара одинаковых подслов α и β, у которых окаймляющие их слева и справа символы различны, т.е. расширение этих слов в любом направлении нарушит их равенство.*

Максимальная пара представляется тройкой (p_1, p_2, n'), где p_1 и p_2 — начальные позиции этих подслов, а n' — их длина. Через $\mathcal{R}(S)$ обозначим множество всех троек, задающих максимальные пары в S.

Например, в строке $S = xabcyiiizabcqabcyrxar$ имеется три вхождения подслова abc. Первое и второе образуют максимальную пару $(2, 10, 3)$, второе и третье — максимальную пару $(10, 14, 3)$, а первое и третье не образуют максимальную пару.

Определение 6. *Подслово α слова S называется* максимальным повтором *, если оно входит в некоторую максимальную пару. Через $\mathcal{R}^{'}(S)$ обозначим множество всех максимальных повторов S.*

В нашем примере и abc, и $abcy$ являются максимальными повторами.

Определение 7. *Максимальный повтор называется* супермаксимальным повтором*, если он не является подсловом никакого другого максимального повтора.*

7.1. Поиск всех максимальных повторов

Лемма 3. *Пусть T — суффиксное дерево для слова S. Если подслово α является максимальным повтором, то путь в T, помеченный α, ведет в некоторую вершину v.*

Из этой леммы непосредственно следует

Теорема 4. *В каждом слове длины n может быть не более n максимальных повторов.*

Определение 8. *Для каждой позиции i слова S назовем символ $S(i-1)$ левым символом для i. Левый символ для листа дерева T — это левый символ позиции-метки этого листа.*

Внутренняя вершина v дерева T называется левой вилкой *(left diverse), если хотя бы два листа в поддереве с корнем v имеют различные левые символы.*

Заметим, что, если v — левая вилка, то и все ее предки в T также являются левыми вилками.

Теорема 5. *Слово α, ведущее в дереве T в вершину v, является максимальным повторм тогда и только тогда, когда v является левой вилкой.*

Задача. Постройте алгоритм сложности $O(n)$, который для каждой вершины T определит, является ли она левой вилкой.

Из теоремы 5 следует, что все максимальные повторы однозначно представляются некоторым срезом дерева T, включающим только левые вилки. Это поддерево имеет размер $O(n)$, хотя общая длина всех максимальных повторов может достигать $O(n^2)$.

Теорема 6. *Все максимальные повторы в слове S можно найти за время $O(n)$, а дерево, представляющее их, можно получить из суффиксного дерева T за такое же время.*

7.1. Поиск всех супермаксимальных повторов

Определение 9. *Максимальный повтор α называется* почти супермаксимальным повтором, *если он хотя бы один раз входит в S в такой позиции, в которой не является подсловом другого максимального повтора. Такое вхождение α назовем* свидетельством *этого факта.*

Например, в строке $a\alpha bx\alpha ya\alpha bx\alpha b$ подслово α является максимальным повтором, но не будет ни супермаксимальным, ни даже почти супермаксимальным повтором. А в строке $a\alpha bx\alpha ya\alpha b$ это же подслово α снова не супермаксимально, но почти супермаксимально. Свидетельством этого будет второе вхождение α.

Пусть вершина v суффиксного дерева T для слова S соответствует максимальному повтору α и пусть w — один из сыновей v. Обозначим через $L(w)$ множество вхождений α в S, определяемых листьями поддерева с корнем w.

Лемма 4. *Если w — внутренняя вершина T, то никакое из вхождений α в S, задаваемых $L(w)$ не является свидетельством почти супермаксимальности α.*

Лемма 5. *Пусть w является листом T, к которому ведет путь $\beta = \alpha\gamma$. Пусть x — это левый символ для w. Тогда вхождение α в S, задаваемое w является свидетельством почти супермаксимальности α тогда и только тогда, когда x не является левым символом ни у какого другого листа под вершиной v.*

Теорема 7. *Внутренняя вершина v, являющаяся левой вилкой, представляет почти супермаксимальный повтор α тогда и только тогда, когда один из сыновей v является листом (задающим некоторую позицию i), левый символ которого $S(i-1)$ не является левым символом никакого другого листа под v. Левая вилка v представляет супермаксимальный повтор тогда и только тогда, когда все сыновья v являются листьями с различными левыми символами.*

Отсюда следует, что все супермаксимальные и почти супермаксимальные повторы в слове S можно найти за линейное время $O(n)$.

8. Линеаризация циклического слова

Пусть задано циклическое слово $S[1..n]$, в котором за каждым символом $S(i)$ следует символ $S(i+1 \mod n)$. Задача состоит в том, чтобы выбрать место разрезания этого слова i так, чтобы получившееся линейное слово $S^i = S(i) \dots S(n)S(1) \dots S(i-1)$ было лексикографически минимальным среди всех таких линейных слов.

Эта задача возникает в химических базах данных для кольцевых молекул.

Задача. Предложите алгоритм линейной сложности для линеаризации циклических слов.

Указание: используйте суффиксное дерево для слова LL, где L — произвольная линеаризация S.

9. Сжатие данных по Зиву-Лемпелю (Ziv-Lempel)

Определение 10. *Для позиции i слова S длины n обозначим через $Prior_i$ самый длинный префикс $S[i..n]$, который является подсловом слова $S[1..i-1]$.*

Через l_i обозначим длину $Prior_i$. При $l_i > 0$ через s_i обозначим начальную позицию самого левого вхождения $Prior_i$.

Например, при $S = abaxcabaxabz$ мы имеем $Prior_7 = bax,\ l_7 = 3$ и $s_7 = 2$.

Алгоритм сжатия 1

begin
i:=1;
Repeat
  вычислить l_i и s_i;
  if $l_i > 0$ *then*
    { выдать (s_i, l_i);
    $i := i + 1$}
  else
    { выдать $S(i)$;
    $i := i + 1$}
Until $i > n$
end

Для вышеприведенного слова S результатом алгоритма будет представление $ab(1,1)c(1,3)x$ $(1,2)z$. По нему можно легко восстановить S.

Пусть T — суффиксное дерево для S. Для вершины v обозначим через c_v минимальную позицию суффикса S среди листьев поддерева с корнем v, т.е. c_v — это позиция первого вхождения метки-пути из корня в v.

Алгоритм Укконена можно использовать для построения сжатого представления S следующим образом. Предположим, что уже построено сжатое представление для $S[1..i-1]$ и неявное суффиксное дерево I_{i-1} для строки $S[1..i-1]$. Предположим также, что для каждой вершины v определено c_v. Тогда пару (s_i, l_i) можно получить, пойдя в I_{i-1} по пути, помеченному $S[i..m]$ до такой точки p (не обязательно вершины), в которой либо выполнено условие $i =$(длина слова, ведущего в p) $+c_v$, где v — первая вершина на этом пути не выше p, либо после p нет подходящего продолжения $S[i..m]$. Во всех случаях $s_i = c_v$ и $l_i =$(длина слова, ведущего в p). Время такого вычисления пары (s_i, l_i) ограничено $O(l_i)$.

Далее выполняется $(i+1)$-я фаза алгоритма Укконена, на которой I_{i-1} преобразуется в I_i. При этом в момент создания новой вершины v, разбивающей ребро (u, w) на две части, положим $c_v = c_w$, а если создается новый лист v с меткой j, то полагаем $c_v = j$.

Теорема 8. *Алгоритм сжатия 1 можно реализовать в линейное время как алгоритм, последовательно обрабатывающий символы строки S за один проход.*