1. 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 1. 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 2. *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 3. *Максимальный повтор называется* супермаксимальным повтором*, если он не является подсловом никакого другого максимального повтора.*

7.1. Поиск всех максимальных повторов

Лемма 1. *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 1. *В каждом слове длины n может быть не более n максимальных повторов.*

Определение 4. *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 2. *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 2 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 3. *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 5. *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 2. *Если w — внутренняя вершина T, то никакое из вхождений α в S, задаваемых $L(w)$ не является свидетельством почти супермаксимальности α.*

Лемма 3. *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 4. *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 $S^i = S(i)\ldots S(n)S(1)\ldots S(i-1)$ 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 6. *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 5. *Алгоритм сжатия 1 можно реализовать в линейное время как алгоритм, последовательно обрабатывающий символы строки S за один проход.*