Приложения суффиксных деревьев

Дискретный анализ 2012/13

Андрей Калинин, Татьяна Романова

26 ноября 2012 г.

Суффиксные деревья

- Некоторые подробности

Применения суффиксных деревьев

- Поиск подстрок
- Общие подстроки
- Статистика совпадений

Литература

- Дэн Гасфилд, «Строки деревья и последовательности в алгоритмах: Информатика и вычислительная биология», 2003. Глава 7, «Первые приложения суффиксных деревьев», стр. 158–213.

Раздел

Суффиксное дерево для набора строк

- Можно обобщить определение суффиксного дерева для набора строк $\{S_1, S_2, \ldots, S_z\}$: такое дерево содержит в себе все суффиксы каждой из строк набора.
- Добавить к каждой строке уникальный терминальный символ, после чего сконкатенировать все строки и построить суффиксное дерево для результата.
- Можно и без конкатенации: последовательно добавляя строки в дерево и продолжая алгоритм Укконена.

Практическая реализация

- Для очень длинных строк и больших алфавитов свойства алгоритма и использования деревьев ухудшаются.
- Как хранить дуги? Варианты: массив размера $|\Sigma|$, линейный список, сбалансированные деревья, хеш.
- В любом случае, время работы зависит от размера алфавита, честная оценка времени построения дерева не линейная: $O(m \log |\Sigma|)$.
- Возможен иной подход к хранению суффиксных деревьев: суффиксные массивы.

Раздел

Поиск подстроки в тексте

- Поистоить суффиксное дерево для текста.
- Пройти по нему от корня по пути с меткой P.
- Если такой путь существует, то все листы в поддереве — вхождения в текст.
- Можно предложить способ с предварительной обработкой образца (позже).

Множественный поиск

- Построить суффиксное дерево.
- Для каждого образца выполнить поиск.
- Время работы $O(m + n + k)$, где k — число вхождения образцов.
- Оценка аналогична алгоритму Ахо-Корасика, но более простое обобщение.

Подстрока для базы образцов

- Есть фиксированный набор строк (база данных). Нужно для строки S найти в базе данных все строки, содержащие S как подстроку.
- Для всех строк из базы данных строится обобщенное суффиксное дерево.
- В этом дереве ищется путь с меткой S: листья в поддереве этого пути соответствуют всем вхождениям строки S в образцы из базы данных.
- Время подготовки $O(m)$, время поиска $O(n + k)$.
- Ни один из рассмотренных до сих пор алгоритмов не обладает такими свойствами.

Линеаризация циклической строки

- Циклическая строка S — за $S(n)$ следует $S(1)$.
- Существует n вариантов разрезов циклической строки.
- Необходимо определить место разреза S таким образом, чтобы полученная линейная строка была лексически наименьшей среди всех n возможных строк, созданных разрезанием S.
- Например, для $baac$ требуется найти разрез $aacb$.

Решение с помощью суффиксных деревьев

- Разрежем строку S в произвольном месте, получив линейную строку L.
- Удвоим L, получив LL.
- Построим суффиксное дерево $\mathbb{T}$ для LL.
- Проходим $\mathbb{T}$ таким образом, чтобы в каждом узле двигаться по дуге с наименьшим первым символом.
- Обход заканчиваем при достижнии строковой глубины n.
- Все листья найденного поддерева могуть быть использованы для искомого разреза.

Раздел

Наибольшая общая подстрока двух строк

- Даны две строки S_1 и S_2, необходимо найти наибольшую их общую подстроку.
- Построим обобщенное суффиксное дерево для $\{S_1, S_2\}$.
- Пометим каждую внутреннюю вершину числом 1, если в ее поддереве существует лист S_1 и 2 — если есть лист S_2.
- Такую разметку дерева можно сделать за один его обход.
- Путевая метка вершины, помеченной одновременно 1 и 2 — общая подстрока.
- Вершина, помеченная 1 и 2 с максимальной строковой глубиной соответствует наибольшей общей подстроке.

Общие подстроки более чем двух строк

- Есть K строк, Общей длиной n. Для $2 \leqslant k \leqslant K$ определим $l(k)$ как длину самой длинной подстроки, общей для $\geqslant k$ строк.
- Нужно рассчитать $C(v)$ — число различных строковых идентификаторов в листьях из поддерева вершины v.
- Можно держать битовый вектор длиной K у каждой вершины: i-и бит соответствует наличию суффикса из i-й строки в поддереве.
- При обходе дерева такой вектор для каждой вершины получается битовым сложение векторов прямых потомков.
- Время работы: $O(Kn)$.

Раздел

Определение

Статистика совпадений — это массив ms, размером с текст, в котором каждый элемент $msr(i)$ является длиной наибольшей подстроки T, начинающейся с позиции i, совпадающей с какой-то подстрокой P. Если есть вхождение P в позиции i, то $msr(i) = |P|$.

Пример

Образец: `wyabcwzqabcdw`

T	a	b	c	x	a	b	c	d	e	x
ms_i	3	2	1	0	4	3	2	1	0	0

Значения $ms(i)$ можно найти, последовательно прикладывая суффиксное дерево для P к каждой позиции T (но это не линейно!)

Вычисление статистики совпадений

- Строится суффиксное дерево $\mathbb{T}$ для образца P, суффиксные связи сохраняются.
- Явно вычисляется $ms(1)$: поиском пути максимальной длины для $T[1..m]$. Находим точку b внутри $\mathbb{T}$, в которой заканчивается строка $T[1..ms(1)]$.
- $ms(i+1)$ строится по $ms(i)$ так:
  - Если b находится во внутренней вершине v, то проходим в $s(v)$.
  - Если b не внутренняя вершина, то поднимаемся до ближайшей вершины v, откуда переходим в $s(v)$ и дальше спускаемся «прыжком по счетчику».
  - Таким образом, находим, где кончается подстрока, соответсвтующая b без первого символа. Оттуда продолжаем прямое стравнение с символом $T(i+ms(i))$ и далее.

Корректность и линейность

Корректен, так как имитирует прямой метод нахождения статистики совпадений.

Линейность обосновывается аналогично линейности алгоритма Укконена: отслеживается вершинная глубина и ее изменение при вычислении $ms(i+1)$ по $ms(i)$.

Небольшое обобщение

Для позиции i в T число $p(i)$ указывает начальную позицию в P, такую, что $P[p(i)..p(i)+ms(i)-1] = T[i..i+ms(i)-1]$.

Расчет $p(i)$:

1. Каждая вершина дерева помечается номером какого-нибудь листа из своего поддерева, $p'(v)$.
2. Если поиск значения $ms(i)$ останавливается в вершине v, то $p(i) \leftarrow p'(v)$.
3. Если же поиск останавливается на дуге $< v, u >$, то $p(i) \leftarrow p'(u)$.

Наибольшая общая подстрока двух строк

1. Построкить суффиксное дерево для меньшей строки ($|S_1| \leqslant |S_2|$).
2. Рассчитать статистику совпадений для S_2.
3. Максиаальное значение $ms(i)$ будет соответствовать длине наибольшей общей подстроки, при этом i — вхождение в S_2, а $p(i)$ — вхождение в S_1.
4. Отличается от предыдущего способа меньшими требованиями к оперативной памяти (меньше размер суффиксного дерева).