Лабораторная работа № 5 по курсу дискретного анализа: суффиксные деревья

Выполнил студент группы М8О-307Б-21 Меркулов Фёдор.

Условие

Найти в заранее известном тексте поступающие на вход образцы.
Вариант алгоритма: В. 1 Поиск в известном тексте неизвестных заранее образцов

Формат ввода

Текст располагается на первой строке, затем, до конца файла, следуют строки с образцами.

Формат вывода

Для каждого образца, найденного в тексте, нужно распечатать строчку, начинающуюся с последовательного номера этого образца и двоеточия, за которым, через запятую, нужно перечислить номера позиций, где встречается образец в порядке возрастания.

Метод решения

Требуется написать реализацию алгоритма Укконена для построения суффиксного дерева и при помощи него решить поставленную задачу. Мы будем строить явное суффиксное дерево, терминирующим символом будет $. Опишем так называемые **правила продолжения суффиксов**:

Укажем точные правила для *продолжения суффикса*. Пусть $S[j..i] = \beta$ - суффикс $S[1..i]$. В продолжении j, когда алгоритм находит конец β в текущем дереве, он продолжает β, чтобы обеспечить присутствие суффикса $\beta S(i+1)$ в дереве. Алгоритм действует по одному из следующих трёх правил:

Правило 1. В текущем дереве путь β кончается в листе. Это значит, что путь от корня с меткой β доходит до конца некоторой "листовой" дуги (дуги, входящей в лист). При изменении дерева нужно добавить к концу метки этой листовой дуги $S(i+1)$.

Правило 2. Ни один путь из конца строки β не начинается символом $S(i+1)$, но по крайней мере один начинающийся оттуда путь имеется. В этом случае должна быть создана новая листовая дуга, начинающаяся в конце β и помеченная символом $S(i+1)$. При этом, если β кончается внутри дуги, должна быть создана новая вершина. Листу в конце новой листовой дуги сопоставляется номер j.

Правило 3. Некоторый путь из конца строки β начинается символом $S(i+1)$. В этом случае строка $\beta S(i+1)$ уже имеется в текущем дереве, так что ничего не надо делать.

Для ускорения алгоритма используем суффиксные связи. Пусть $x\alpha$ обозначает произвольную строку, где x - её первый символ, а α - оставшаяся подстрока (возможно, пустая). Если для внутренней вершины v с путевой меткой $x\alpha$ существует другая вершина $s(v)$ с путевой отметкой α, то указатель из v в $s(v)$ называется суффиксной связью. Суффиксную связь будем обозначать парой $(v, s(v))$. Если строка α пуста, связь идёт в корневую вершину. Но это лишь уменьшает количество передвижений от корня в каждом продолжении.

Для оптимизации алгоритма используем 3 приёма:

Приём 1.

"Перескок". Суть этого приёма заключается в том, что если мы хотим найти место для вставки вхождения строки в дерево, то мы не идём в дерево посимвольно, если длина строки γ больше длины текущей дуги. В таком случае мы её просто пропускаем, а идти начинаем от $|\gamma|$ - длина дуги.

Приём 2.

Заканчивать каждую фразу $i + 1$ после первого же использования правила 3. Если это случится в продолжении j, то уже не требуется явно находить концы строк $S[k..i]$ с $k > j$. Также стоит учесть, что дуги мы "сжимаем" до дуговых меток, т. е. это у нас 2 не массив суффиксов, а пара чисел (p, e), где e - "текущий конец".

Приём 3.

Если дуга стала листовой - она таковой и останется. Из этого следует, что e для листов можно ввести как глобальную переменную, и работать с ней за константу

Описание программы

Считывается исходный текст, добавляется терминирующий символ “$” в конец текста и строится суффиксное дерево. Далее считываются образцы и передаются в метод поиска образца в суффиксном дереве, вместе с их порядковым номером $order$. Далее из корня мы пытаемся найти поочерёдно каждый из символов образца, если в какой-то момент не смогли найти соответствующий символ, то образца нет в исходном тексте, если нашли все символы и остановились на дуге или внутреннем узле, то из этого листа находим всевозможные листья в которые мы можем прийти и их номера будут отвечать за позицию вхождения образца в текст, то есть просто выводим “$order$: [все номера листов в которые мы смогли дойти через запятую]$\backslash n$” и так для каждого образца.

Дневник отладки

Неправильное написание метода поиска образца в суффиксном дереве приводило к $WA4$ и $TL4$ (бесконечный цикл)

После первого OK попытался хранить в каждом узле все листья к которым можно прийти от этого узла, чтобы не использовать рекурсию каждый раз, а просто обратиться к уже заранее найденной информации, но словил $TL9$

Тест производительности

Время работы алгоритма я решил проверить с помощью утилиты time. Входные данные: случайно сгенерированные строки: исходный текст имеет размер – 10^n, а образцы имеют случайный размер от 1 до 10001. Цифра в названии тестовых файлов иллюстрирует количество образцов, а также и размер исходного текста – 10^n, если название тестового файла testn.txt.

Также был оставлен вывод самой проверяемой программы, чтобы можно было отследить был ли найден хотя бы один образец; при работе с test5.txt появляются троеточия – символ того, что было выведено намного больше информации, но для проверки времени работы эта информация является излишней.

papik@papik-VirtualBox:~/DA5$ time ./a.out < test1.txt

real 0m0,016s

user 0m0,005s

sys 0m0,011s

papik@papik-VirtualBox:~/DA5$ time ./a.out < test2.txt

real 0m0,024s

user 0m0,013s

sys 0m0,011s

papik@papik-VirtualBox:~/DA5$ time ./a.out < test3.txt

real 0m0,065s

user 0m0,040s

sys 0m0,012s

papik@papik-VirtualBox:~/DA5$ time ./a.out < test4.txt

5087: 1116, 1161, 1495, 1507, 1627, 1752, 2279, 2807, 3564, 3801, 5523, 6179, 7116, 7135, 8951, 9095, 9466, 9512

real 0m0,470s

user 0m0,365s

sys 0m0,093s

papik@papik-VirtualBox:~/DA5$ time ./a.out < test5.txt

9798: 18, 43, 57…

10261: 6, 77, 101…

13086: 66508, 81110, 96291

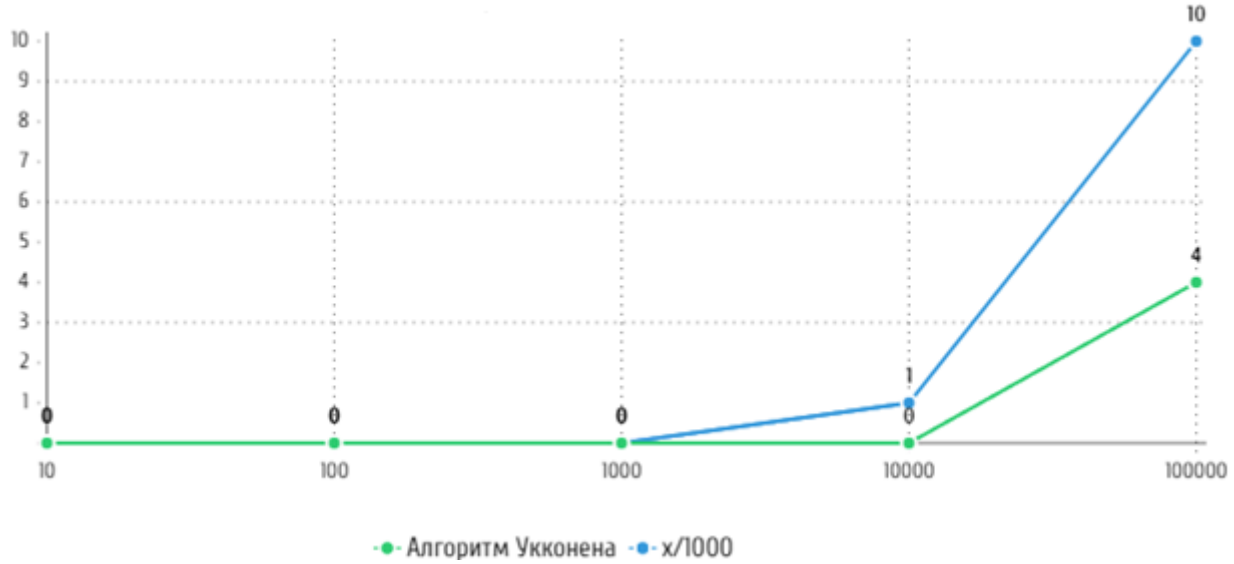
15564: 866, 2282, 2310…

…

real 0m4,917s

user 0m3,691s

sys 0m0,898s



	10	100	1000	10000	100000
x/1000	0,001 с	0,01 с	0,1 с	1 с	10 с
Алгоритм Укконена	0,016 с	0,024 с	0,065 с	0,470 с	4,917 с

Согласно данным измерениям, можно сказать, что с ростом количества входных данных время работы согласуется с заявленной сложностью, то есть растёт линейно.

Выводы

Выполнив пятую лабораторную работу по курсу «Дискретный анализ», я познакомился с алгоритмом Укконена построения суффиксного дерева за линейное время, а также с самими суфиксными деревьями, узнал, какими ускорениями можно топорный алгорим построения суффиксного дерева, работающий за кубическое время, преобразить в квадратное и даже в линейное.