Лабораторная работа № 5 по курсу дискретного анализа: Суффиксные деревья

Выполнил студент группы 08-207 МАИ *Днепров Иван.*

Условие

1. Необходимо реализовать алгоритм Укконена построения суффиксного дерева за линеиное время. Построив такое дерево для некоторых из входных строк, необходимо воспользоваться полученным суффиксным деревом для решения своего варианта задания.

2. Алфавит строк: строчные буквы латинского алфавита (т.е., от a до z).

3. Вариант 1: Поиск в известном тексте неизвестных заранее образцов

4. Наити в заранее известном тексте поступающие на вход образцы. Входные данные: текст располагается на первои строке, затем, до конца фаила, следуют строки с образцами. Выходные данные: для каждого образца, наиденного в тексте, нужно распечатать строчку, начинающуюся с последовательного номера этого образца и двоеточия, за которым, через запятую, нужно перечислить номера позиции, где встречается образец в порядке возрастания.

Метод решения

В начале строим суффиксное дерево, в вершинах с символом окончания строки которого храним позицию начала добавляемой подстроки.

Построение суффиксного дерева заключается в последовательном добавлении подстрок таким образом, что первая добавленная полстрока равна текущей строке, а каждая последующая подстрока на один символ короче текущей.

Дерево строится за линейное время благодаря использованию суффиксных ссылок и лемме о том, что лист всегда будет оставаться листом.

Для линейности в дереве хранятся не подстроки, а начальный и конечнаый итераторы подстроки.

Далее остается только найти вхождения заданных подсрок.

Для этого мы углубляемся в дерево, пока не закончится совпадающая часть заданной подстроки. Если вся строка совпала, углубляемся с текущей ноды в дерево и выводим номера вхождений этой подстроки после сортировки. Если же подстрока не полностью пройдена, значит даной подстроки в дереве нет и выводить нечего.

Поиск тоже работает за линию.

Описание программы

Структура суффиксного дерева состоит из текста, корня, ноды, для которой нужно добавить суффиксную ссылку, текущей ноды, напоминателя, текущей длинны и текущего барьера.

Для класса дерева предусмотрены следующие функии: конструктор, деструктор, функция добавления суффиксных ссылок, функция углубления в дерева и функция поиска всех вхождений подстроки.

Основыне идеи работы этих функций описаны выше.

Выводы

Така как дерево сторится за линейное время и поиск в нем осуществляется тоже за линию, итоговая сложность алгоритма так же линейна, а благодаря идеи хранить указатели начала и конца подстроки, соответствующей ноде, вместо самой подстроки, используемая память тоже линейна.

Хоть алгоритм и работает за линейное время, у него есть очевидные недостатки. Алгоритм работает медленнее, чем алгоритм поиска в суффиксном масиве и хранить дерево значительно накладнее, чем массив. По этому для больших объемов данных этот алгоритм не является лучшим вариантом. В то же время, для небольших объемов гораздо проще и эффективнее использовать преффикс функцию.

В заключение хочу сказать, что хотя алгоритм поиска в суффиксном дереве можно скорее рассматривать как переходный алгоритм к алгоритму поиска в суффиксном массиве, данный алгоритм всё равно работает достаточно быстро и интересен в образовательных целях.

Вообще, суфифксное дерево посзоляет решать целую кучу разнообразных задач за линейное время и если научиться его строить за линейное время можно достаточно эффективно решать большое количество задач связанных с работой со строками, таких как линеоризация строк и поиск наибольшей общей подстроки.