Лабораторная работа № 5 по курсу дискретного анализа: суффиксные деревья

Выполнил студент группы М8О-208Б-18 МАИ *Коростелев Дмитрий Васильевич.*

Задание

Необходимо реализовать алгоритм Укконена построения суффиксного дерева за линейное время. Построив такое дерево для некоторых из выходных строк, необходимо воспользоваться полученным суффисным деревом для решения своего варианта задания.

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

Вариант №1

Найти в заранее известном тексте поступающие на вход образцы.

Формат входных данных

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

Формат результата

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

Метод решения

Алгоритм Укконена

Для начала требуется построить суффиксное дерево заданной строки. Для этого используется алгоритм укконена. Алгоритм Укконена строит суффиксное дерево за линейное в отличие от наивного метода за счет ряда оптимизаций.

1-я оптимизация – если, добавили лист, то этот узел всегда остается листом. Данной оптимизацией мы избавляем себя от работы постоянного наращивания листов, так как очевидно, что с добавлением нового символа в строке, все суффиксы в этой строке удлиняться на добавленный символ, такая операция происходит всегда и чтобы не совершать это действие листовые ребра всегда будут указывать на конец строки.

2-я оптимизация – если прошли по ребру, то и в меньших суффиксах также пройдем по ребру.

Если мы добавляем новую вершину к определённому суффиксу, то и в меньших суффиксах мы также должны добавить новый символ, для этого, если находимся в листе, переходим к вершине-предку, далее от него по суффиксной ссылке, и спускаемся на тоже ребру по которому поднялись к родителю на первом шаге и добавляем уже к найденному меньшему суффиксу новую вершину (можно было сразу перейти по суффиксной ссылке, не переходя к предку, однако, возможен такой вариант, что суффиксная ссылка от листа ведет в мнимую вершину, которая лежит на ребре, выше описанным механизмом появление неявных суффиксных ссылок исключается). Далее, вершина, возможно и мнимая, в которой мы остановимся, и будет суффиксной ссылкой для изначальной.

3-я оптимизация. Далее, для строки aaabbbccc будем делать следующее – active_point – вершина, от которой на текущий момент планируется вставлять новый узел. Пусть суффиксное дерево для строки aaab уже сформировано, далее считываем следующие две буквы b (если будет задана другая строка на этом этапе можем попасть в новую вершину, тогда переходим в эту вершину и меняем указатель active_point на ту, в которую пришли), когда встретим c, остаток – то есть, то насколько глубоко мы прошли по дереву будет показывать, где и сколько нужно вставить новых вершин. Таким образом, пока этот остаток не обнулится – добавляем новые вершины, гуляя по дереву (по суффиксным сслыкам).

Для **поиска** подстрок в строке используется суффиксный массив – строится за линейное время обходом в глубину по суффиксному дереву. Сам поиск подстрок в строке имеет логарифмическую сложность от длины строки на длину строки $O(n * log(n))$. Суть в следующем, в суффиксном массиве записаны индексы, с которых начинается определенный суффикс, т.к. все суффиксы в строке отсортированы в лексикографическом порядке можем использовать бинарный поиск для нахождения диапазона чисел, обозначающих подходящие подстроки.

Отладка и проверка программы.

№	Название ошибки	Причина возникновения ошибки
1-4	Ошибка компиляции	Неисправленные предупреждения
5-9	Ошибка выполнения	Утечки памяти
10-14	Неправильный ответ	Ошибки в алгоритме поиска
15	Превышено реальное время работы	Слишком сложный алгоритм поиска
16-20	Неправильный ответ	Улучшения алгоритма поиска

Вывод

При реализации такой структуры данных, как суффиксное дерево возникает ряд трудностей – первоначально с пониманием самого алгоритма, а затем с его написанием. Однако по итогу получили мощную структуру, позволяющую решать целый кластер всевозможных задач связанных со строками.