Лабораторная работа № 5 по курсу дискретного анализа: Суффиксные деревья

Выполнил студент группы 80-308 МАИ *Захаров Игорь*.

Условие

Кратко описывается задача:

1. Необходимо реализовать алгоритм Укконена построения суффиксного дерева за линейное время. Построив такое дерево для некоторых из входных строк, необходимо воспользоваться полученным суффиксным деревом для решения своего варианта задания. Алфавит строк: строчные буквы латинского алфавита (т.е., от a до z).

2. Найти в заранее известном тексте поступающие на вход образцы с использованием суффиксного массива. Формат входных и выходных данных, а так же примеры аналогичны указанным во первом задании.

Метод решения

Реализован алгоритм Укконена, с помощью которого было построено суффиксное дерево за линейное время. После этого из суффиксного дерева поиском в глубину строится суффиксный массив. Для нахождения паттерна бинарным поиском будем итерироваться по элементам массива. Построение суффиксного дерева для заданной строки делится на m фаз. Каждая i-тая фаза делится в свою очередь на i1 продолжений. Каждое из продолжений имеет один из 3-х видов:

1. Путь для текущего продолжения кончается в листе. Тогда просто продлеваем метку для этого листа на один символ. В данной реализации все продолжения такого типа будут совершаться за O(1) действий простым увеличением значения переменной, хранящей индекс конца каждого из существующих листов.

2. Путь для текущего продолжения оканчивается во внутренней вершине либо посередине ребра. В данном случае явной вставки не избежать. Если путь закончился во внутренней вершине, то просто добавляем новое исходящеее ребро в лист дерева, помеченное меткой [i, end], где i – номер текущей фазы, end – та самая глобальная переменная, храянящая индекс конца. Если же путь закончился посередине ребра, то нужно вставить в этом месте новую внутреннюю вершину и добавить исходящее ребро в новый лист, как сказано выше.

3. Если при попытке продолжения пути оказалось, что следующий символ уже в дереве представлен далее, то не делаем ничего. В данной реализации просто завершаем фазу.

Описание программы

Алгоритм Укконена реализован при помощи итераторов и ActiveNode. Каждая вершина содержит указатель на начало и конец данной подстроки в тексте, суффиксную ссылку, построенную по заданному алгоритму, а также словарь с ребрами, обозначающими какие вершины выходят из текущей. ActiveNode нужен для того чтобы указывать на вершину, где мы сейчас находимся, а также для того чтобы понимать сколько символов нам необходимо пропустить, чтобы прийти в нужную вершину, если на данном этапе алгоритма мы в ней не находимся. В главном файле определен весь функционал дерева, заявленный в заголовочном. Так- же в main.cpp находятся все необходимые функции для суффиксного массива, а именно поиск в глубину и бинарный поиск образца.

Выводы

Главное преимущество суффиксных массивов над суффиксными деревьями - более эффективный расход памяти. Зачастую данная деталь очень важна при инференсе на малопроизводительных устройствах. Как мы можем увидеть по приведенному графику, скорость работы алгоритма поиска в суффиксном массиве не уступает скорости работы аналогичного алгоритма в суффиксном дереве в данной релизации. Само же по себе суффиксное дерево представляет очень полезную структуру дан- ных, которая позволяет реализовать поиск подстроки в строке, сравнение двух под- строк, поиск наибольшего префикса и нахождение количества разных подстрок.