Лабораторная работа № 5 по курсу дискретного анализа: суффиксное дерево, поиск множества образцов

Выполнил студент группы 08-207 МАИ *Хренов Геннадий.*

Условие

1. 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.

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

3. Вариант: Найти в заранее известном тексте поступающие на вход образцы.

Метод решения

Суффиксное дерево предполагает, что дуги, соединяющие его узлы, помечены подстроками заданной строки так, чтобы при конкатенации меток дуг с корня до листа получался один из суффиксов заданной строки. Эти метки я буду хранить в узле, в который входит заданная дуга. Для экономии памяти эти метки будут храниться как два индекса, обозначающих граничные слева и справа номера символов заданной строки. Структура узла содержит: метку входящей дуги, суффиксную ссылку, список всех детей с первыми символами дуг, которые соединяют детей с текущим узлом, индекс. Для листов индекс показывает, с какого места начинается данный суффикс, а для внутренних узлов он равен -1. Алгоритм Укконена предполагает N(длинна строки) фаз, каждая из которых делится i(текущий номер фазы) продолжений. Продолжение совершается по одному из 3 правил:
1) Добавление в конец дуги листа. Нужно обратить внимание на то, что если узел является листом, то для него выполняется одинаковая операция каждую фазу - прибавление единицы к правой границе метки. Чтобы не бегать по дереву и не инкрементировать каждый лист, правая граница каждого листа связывается с переменной(глобальный конец), которая инкрементируется в начале каждой фазы.
2)Если мы, находясь в текущем узле, видим, что нет выходящих дуг с добавляемым символом, то мы создаем и присоединяем новый узел с новой меткой к текущему узлу. Если это произошло в внутри дуги, то сначала создаем узел split, который разделит эту дугу на две, а потом к нему присоединяем новый узел с новой меткой.
3)Если наш добавляемый символ уже находится в дереве после текущей позиции, то мы завершаем фазу и выполнем это явное продолжение в следующей фазе.
Для поиска подстроки мы спускаемся по суффиксому дереву, посимвольно сравнивая образец с дугой дерева. Если удалось спуститься на весь образец, то мы нашли вхожде-

ние, причем с позиции индекса текущего узла(если это лист) или всех листьев, которые можно достичь из текущего узла.

Описание программы

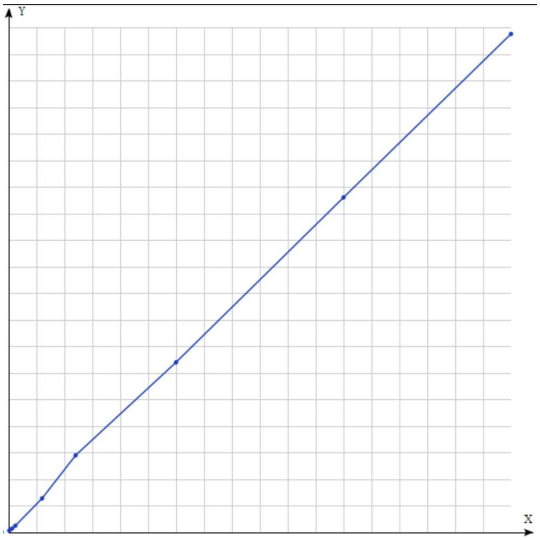
Программа состоит из файла suftr.cpp Основные функции:
IsLeaf - проверяет, является ли узел листом
SkipDown - прыжок по счетчику
Extensions - полностью описывает фазу алгоритма
GetIndex - раздает индексы всем листям
Build - построение суффиксного дерева
Search - поиск подстроки

Дневник отладки

1 - ошибка выполнения - при сравнении map с нулем в неё добавляеся нулевой элемент, учел это замечание.
2-5 - неправильный ответ - ошибка в алгоритме поиска, исправил

Тест производительности

длина текста; время(с)
(1000; 0,013)
(5000; 0,029)
(10000; 0,05)
(50000; 0,256)
(100000; 0,58)
(250000; 1,28)
(500000; 2,52)
(750000; 3,75)



На графике просматривается линейная сложность.

Недочёты

Для хранения детей и первых символов их дуг можно было использовать массив, это бы ускорило доступ к элементам, однако отразилось бы на памяти. В моем случае, так как алфавит конечен, асимптотика с использованием map все равно остается линейной.

Выводы

Суффиксное дерево - полезная структура данных, которая имеет множество приложений. Одно их них - множественное точное совпадение. Существует алгоритм, решающий эту задачу с похожей асимптотикой(n+m+k) - это Ахо-Корасик, однако он обрабатывает образцы, в то время как суффиксное дерево - текст. Поэтому выбор более предпочтительного алгоритма зависит от отношения длин текста и набора образцов, а также от того, что приоритетнее экономить: память или время.