Лабораторная работа № 5 по курсу дискретного анализа: суффиксные дерефья

Выполнил студент группы М8О-207Б МАИ *Цапков Александр.*

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.

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

2. Вариант: Найти образец в тексте используя статистику совпадений.

Метод решения

Для построения суффиксного дерева за линейное время я воспользовался алгоритмом Укконена (как и было предложено в варианте). Этот алгритм работает по средствам построения неявных суффиксных деревьев из префиксов нашей строки. В конце строки добавляется сантинел и суффиксное дерево становится явным. Если мы уже построили суффиксное дерево для i префикса, то $i + 1$ продолжение состоит в продление всех суффиксов построенного на предыдущем продолжении. Также, для достижения временной асимптотики используются несколько "ускорений": суффиксные ссылки, прыжки по счетчику, листом был листом и останешься и др.

После построения суф. дерева нужно найти статистику совпадений для каждой позиции текста, и если статистика совпадений на этой позиции равна длинне паттерна, то паттерн входит в текст с этой позиции. При поиске статистики совпадений используются суффиксные ссылки, которые были построены при построении суф. дерева. При самом вычислении статистики совпадений мы ищем часть строки текста в суф дереве до момента несовподения, тогда мы записываем стат. совпадений для позиции, переходим по суффиксной ссылке и продолжаем проход по дереву, приняв за начальную точку отсчета статистики совпадений – ее же значения у предыдущей позиции.

Описание программы

Моя программа состоит из 2-х файлов: SuffixTree.hpp с реализацией суффиксного дерева и поиска статистики совпадений и основного файла программы main.cpp. Все суффиксное дерево и поиск статистики совпадений реализованны в классе TSuffTree. Его конструктор принимает строку std::string и строит из нее суффиксное дерево по алгоритму укконена. Имеется метод matchStatistics(), который возвращяет вектор со статистикой совпадений. В main я считываю строку с помощью getline, строю из нее

суффиксное дерево, вызываю matchStatistics по этому дереву от текста и получаю обратно вектор. Затем я прохожу по нему и вывожу все номера индексов на которых статистика совпадений равна длинне паттерна.

Дневник отладки

1-е посылки: неправильный ответ, ошибка в построениии суффиксного дерава, а в частности в добавлении внутреннего узела на ребро

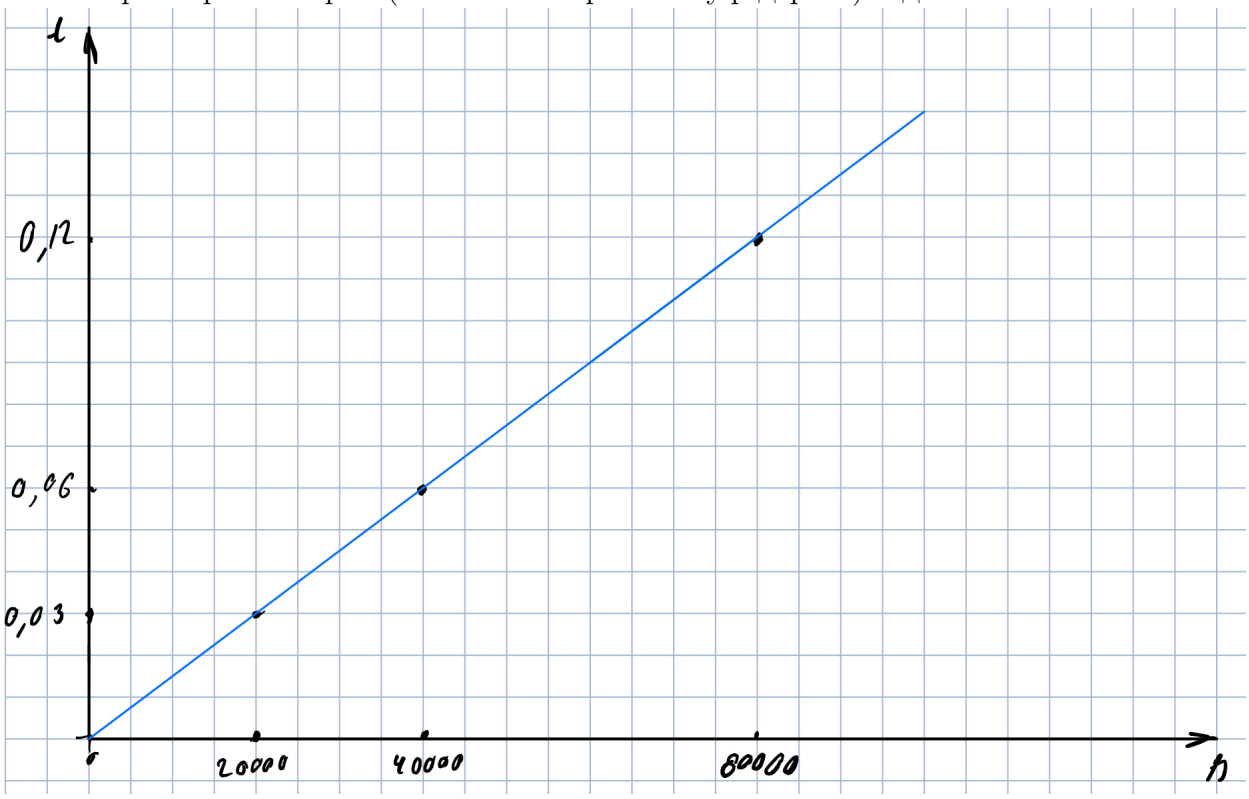
3 посылка: превышено время работы. Для каждого узла выделяется статический массив 256 сиволов (алфавит), когда размер нашего алфавита 26.

Почти все остальные попытки: ошибка выполнения. В поиске статистики совпадений был баг, из-за которого в самом конце происходил поиск по нул терминатору, из-за чего было обращение к массиву детей по нему. Из-за того что я пытался сократить размер алфавита в программе, я убирал нуль терминатор и происходило неправильное чтение. Баг был исправлен и алфавит укорочен.

последняя попытка: ожидает подтверждения.

Тест производительности

Для проверки скорости программы я воспользовался встренной утилитой time. количестве и сам строит график. По графику можно убедится функуия зависимости времени от размера паттерна (то из чего строится суф дерево) – действительно линейна.



Выводы

В данной ЛР я встретился с одним из самых сложных для меня алгоритмов в курсе Дискретного анализа. Это морально подготовило меня к следующим ЛР. Суффиксное дерево окалазось достаточно интересным и очень мощным (суды по приложениям в Гасфилде), но мне пока не пришлось столкнуться с задачей, в которой я бы не смог обойтись бкз фуфиксных деревьев