Лабораторная работа №5 по курсу дискретного анализа: суффиксные деревья

Выполнил студент группы М8О-307Б-21 МАИ *Кажекин Денис*.

Условие

По заданию было необходимо найти самые длинные общие подстроки двух строк с использованием суффиксного дерева. (Вариант 5)

Метод решения

У меня приведена стандартная реализация суффиксного дерева по алгоритму Укконена с использованием приема «Активной точки» и переменной остатка, которые хранили место, с которого необходимо продолжать вставку в дерево и количество оставшихся суффиксов для вставки соответственно. Реализованы все допущения, которые позволяют строить суффиксное дерево за O(n) времени и памяти. В целом, для решения задачи было необходимо сконкатенировать две строки, введенные пользователем, соединенные двумя символами синтелл, и подать эту строку в функцию <build>. Та, в свою очередь, каждый символ отдельно добавляет в дерево. Добавление происходит с помощью добавления суффиксов всех префиксов, используя неявное суффиксное дерево, т.е. некоторые листья содержатся в дереве неявно. Для опитмизации этого процесса используется глобальная переменная <end>, которая автоматически каждому ребру, хранящемуся в формате двух указателей для оптимизации памяти, добавляет символ. Если текущий добавляем символ содержится на ребре, то в свою роль вступает активная точка, которая будет хранить позицию на ребре, с которой на следующей итерации мы начнем проверку. Суффиксные ссылки, подсоединямые к ребрам после создания внутренних вершин в дереве, позволяли производить добавление суффикса практически за одну операцию.

После построения дерева необходимо было реализовать функцию DFS для решения задачи поиска максимальной общей подстроки. Эту задачу я решил таким путем: сначала пронумеровал все ребра дерева (определил к какой строке они принадлежат к 1 или 2). Затем, основываясь на этой информации, пометил внутренние ребра цифрой 3, если те, в свою очередь, принадлежат как первой, так и второй строке. После этого необходимо было, опускаясь по дереву в глубину, накапливать высоту в символах, которую я прохожу и каждый раз обновлять максимум, когда встречаю ребро с меткой 3, и добавлять соответствующие строки в ответ. В случае если накопилась большая длина и встретилось ребро с меткой 3, нужно было обновить вектор ответа и максимум, начиная добавлять строки заново.

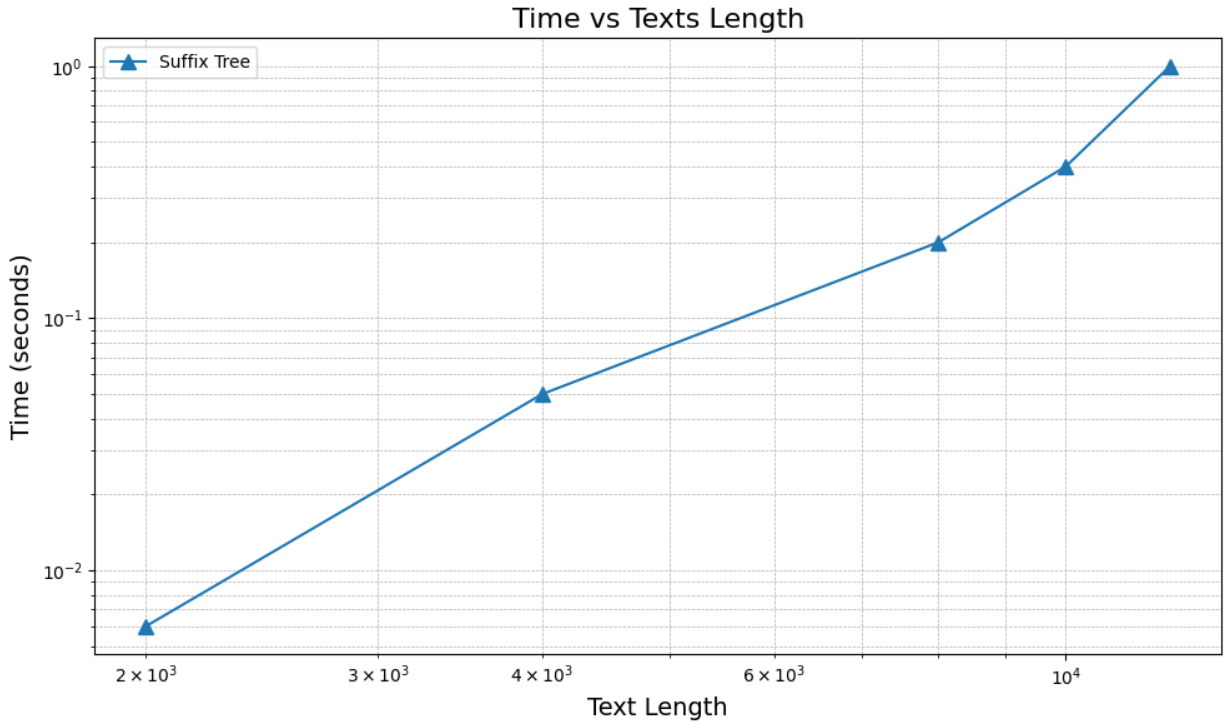
Описание программы

Файл с кодом всего один – suffix_tree.cpp

Дневник отладки

В принципе, трудностей особо не возникло, единственное что - была ошибка TL. Благодаря небольшому анализу понял, что она возникала из-за использования мной встроенной функции <find>, с помощью которой я искал символ “#” для идентификации ребра к какому типу оно принадлежит 1 или 2. Но так как эта функция работает за O(n) по памяти, то при больших входных данных возникали проблемы. Удалось устранить эту проблему путем вычисления индекса символа “#” заранее и использования этой информации за O(1) на каждом шагу алгоритма.

Тест производительности



Наблюдаем относительно линейный график

Недочёты

Недочеты не выявлены

Выводы

В данной лабораторной работе я реализовал суффиксное дерево и алгоритм поиска наибольших общих подстрок в двух текстах.