Лабораторная работа № 5 по курсу дискретного анализа:
« Суффиксные деревья»

Выполнил студент группы 08-307 МАИ *Дубровин Дмитрий*

Условие:

Реализовать поиск подстрок в тексте с использование суффиксного дерева. Суффиксное дерево можно построить за O(n^2) наивным методом.

Метод решения

Программа представляет собой реализацию поиска подстрок в тексте с использованием модифицированного суффиксного дерева. Сначала происходит построение суффиксного дерева, затем выполняется поиск вхождений заданного шаблона в текст. Давайте рассмотрим основные аспекты метода.

Построение суффиксного дерева:

Суффиксное дерево строится путем добавления каждого суффикса строки в дерево. Основная функция `AddNode` отвечает за этот процесс. В ней происходит обход дерева, проверка на совпадение символов и, при необходимости, разделение узлов для вставки нового суффикса.

Структура TrieNode:

Каждый узел суффиксного дерева представлен структурой `TrieNode`, содержащей информацию о границах подстроки (left и right), индексе начала вхождения в оригинальном тексте (index) и коллекцию дочерних узлов (`children`).

Поиск подстрок в тексте:

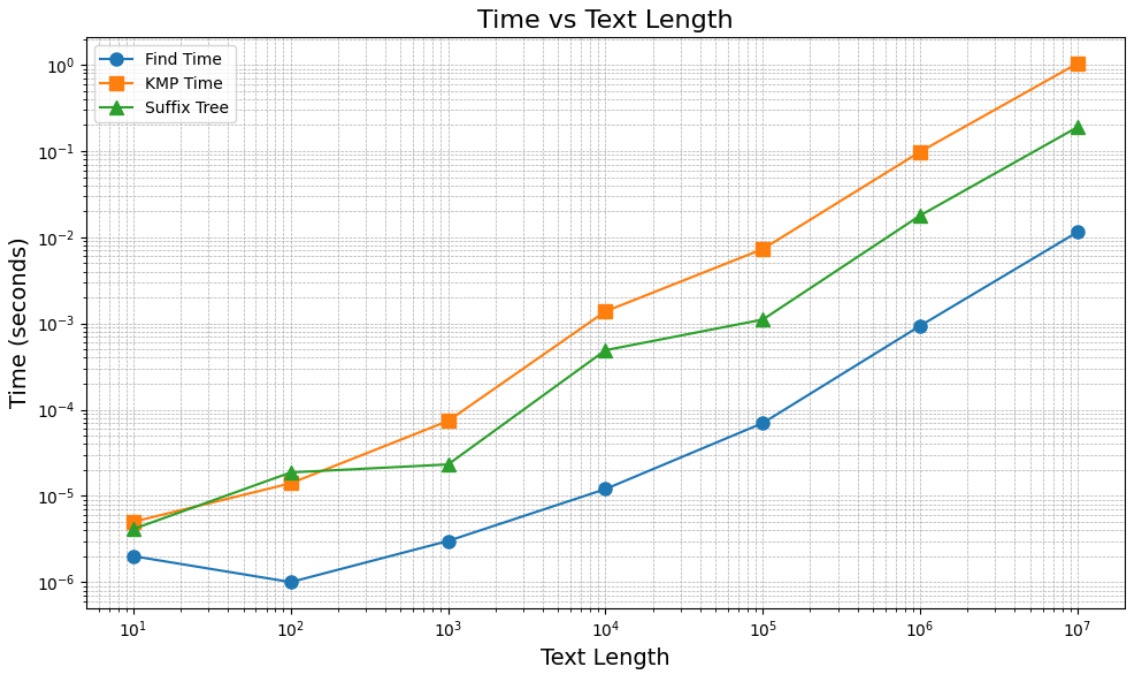
Для поиска вхождений заданного шаблона используется функция `FindOccurrences`. Она производит обход суффиксного дерева, сравнивает

символы шаблона с символами текста в соответствующих узлах. При совпадении продолжает обход, а при несовпадении завершает поиск в текущем узле. Результаты вхождений сохраняются в вектор.

Вывод результатов:

Найденные вхождения выводятся с учетом номеров строк. Весь этот процесс обработки текста и поиска подстрок реализован в методе `ExecuteAlgorithm`.

Тест производительности



Тест производительности из себя представляет собой следующее: я сравниваю время работы разработанного мною алгоритма с КМП и базовым методом find на Python. Код для получения данных для графика, а также тесты представлены в одноименной папке. Также стоит заметить, что тесты получены с помощью библиотеки random на Python.

Как видно из графика, алгоритм быстрее базовой реализации КМП, однако уступает в скорости методу find. Алгоритм создания суффиксного дерева методом наивного построения обладает квадратичной сложностью

O(n^2), где n представляет собой длину текста. Это делает его менее предпочтительным для обработки больших объемов данных. Сложность операции поиска подстроки, включая построение суффиксного дерева, может достигать O(n^2+m+k), где m - длина искомой подстроки, а k - количество вхождений этой подстроки в текст. Такой подход существенно снижает производительность с увеличением размера входных данных.

Выводы

В данной лабораторной работе я реализовал поиск всех вхождений образца в тексте с использованием суффиксного дерева. Суффиксное дерево является эффективным инструментом для решения различных задач на строках, включая поиск подстроки в тексте.