Лабораторная работа №4 по курсу дискретного анализа: суффиксные деревья.

Выполнил студент группы М80-308Б-20 Зубко Дмитрий Валерьевич.

Условие

Линеаризовать циклическую строку, то есть найти минимальный в лексикографическом смысле разрез циклической строки.

Метод решения

Суффиксное дерево - это компакт трай, который построен по всем суффиксам строки (нумеруем от 1), кроме пустого. Должен обладать следующими свойствами:

1) Такое же количество листьев как суффиксов в строке
2) Должен быть сжатым

С помощью суффиксного дерева мы решаем задачу, когда мы берем текст, обрабатываем его за O(n), где n – длина текста, а потом запускаем функцию поиска для паттерна за O(m), где m – длина паттерна. В итоге решаем задачу за O(n + m).

Суффиксное дерево можно построить наивным алгоритмом за O(n^2), где n – длина текста. Берем все суффиксы строки, добавляем их в наш трай.

Алгоритм Укконена – алгоритм построения суффиксного дерева за O(n). Берем все префиксы строки и добавляем их префиксы. Применяем правила:

1) Добавляем в лист – дописать букву на ребро
2) Нет пути – создать новый лист
3) Есть строка – ничего не делаем

Также используем некоторые эвристики. На ребрах храним два числа, обозначающие начало и конец текста на ребре. Для ребер, ведущих в листья, второе число – глобальная переменная end, которую мы инкрементируем на каждой итерации. Вводим понятие суффиксной ссылки. Она перемещает из строки вида xALPHA к ALPHA, где x и ALPHA – какие-то строки. Если мы хотим вставить суффикс, который есть в дереве, то мы пропускаем эту вставку, а также вставку всех его суффиксов, потому что они уже есть в дереве.

Описание программы

Программа состоит из трёх файлов.

В файле main.cpp содержится ввод и вывод информации, а также функция нахождения минимального лексикографического разреза циклической строки. Чтобы линеаризовать циклическую строку нужно конкатенировать строку с самой собой и добавить в ее конец синтинель – символ, который не встречается в нашем алфавите. В лабораторной работе я

использовал символ $. Чтобы получить ответ, я создаю переменную, которая отвечает за длину полученной строки. Пока она не станет равна половине длины удвоенной исходной строки, то мы движемся по минимальным лексикографическим символам текста. Синтинель имеет максимальный лексикографический приоритет.

В node.cpp реализована структура ноды: позиции start, end; map детей ноды.

В tree.cpp – суффиксное дерево, которое строится с помощью алгортма Укконена. Структура дерева содержит корень; конкретная нода на итерации; длина, которую стоит обработать; ребро, на котором находимся, глобальный end.

Дневник отладки

Программа прошла тесты с первого раза.

Тест производительности

Сравнение алгоритма Укконена и наивного алгоритма.

1 тест. Длина строки 1000 символов.
Наивный алгоритм: 50 ms
Алгоритм Укконена: 20 ms.

2 тест. Длина строки 5000 символов.
Наивный алгоритм: 800 ms
Алгоритм Укконена: 100 ms.

3 тест. Длина строки 10000 символов.
Наивный алгоритм: 5700 ms
Алгоритм Укконена: 254 ms.

4 тест. Длина строки 30000 символов.
Наивный алгоритм: 56713 ms
Алгоритм Укконена: 873 ms.

Из тестов видно, что линейный алгоритм во много раз превосходит квадратичный алгоритм.

Недочёты

Программа работает только при условии корректного ввода. Неправильный ввод может убить работоспособность программы.

Выводы

Данная лабораторная работа помогла мне лучше осознать структуру данных суффиксное дерево. Для проверки производительности я реализовал суффиксное дерево за O(n^2), через добавление всех суффиксов строки в трай. Затем реализовал суффиксное дерево

через алгоритм Укконена за O(n). Применил суффиксное дерево для нахождения разреза циклической строки. Существует большое множество задач поиска чего-либо в большом тексте, поэтому суффиксное дерево имеет большую область применения.