Лабораторная работа № 5 по курсу дискретного анализа: Суффиксные деревья

Выполнил студент группы М8О-308Б-20 МАИ *Попов Матвей*.

Условие

1. Необходимо реализовать алгоритм Укконена построения суффиксного дерева за линейное время. Построив такое дерево для некоторых из входных строк, необходимо воспользоваться полученным суффиксным деревом для решения своего варианта задания.

2. Вариант задания: поиск в известном тексте неизвестных заранее образцов

Метод решения

Реализован класс суффиксного дерева, в качестве конструктора использовался алгоритм Укконена, для которого специально было введено 5 полей в классе. Для реализации алгоритма поиска был реализован метод *Search*, возвращающий множество вхождений аргумента в текст, по которому было построено дерево. Отличительной особенностью программы является то, что в ней нет рекурсии.

Описание программы

Программа состоит из трёх файлов: *TSuffixTree.hpp* с реализацией суффиксного дерева и необходимых алгоритмов, *main.cpp* и *Makefile* для сборки программы.

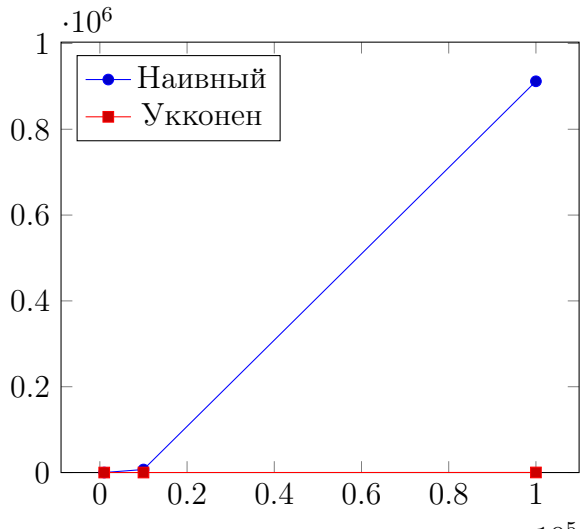
Дневник отладки

1. После нескольких неудачных попыток реализации алгоритма Укконена было принято решение выделить некоторые необходимые переменные в поля класса суффиксного дерева, чтобы запоминать необходимые значения даже после окончания работы функции вставки одного суффикса.

2. Было принято решение заменть тип возвращаемого значения метода *Search* с вектора на множество, чтобы не беспокоиться о сортировке возвращаемых результатов, а также для более эффективного расхода памяти.

Тест производительности

Ниже приведено сравнение времени работы наивного алгоритма построения суффиксного дерева и алгоритма Укконена. По оси X — количество символов в тексте, по оси Y — время построения суффиксного дерева в милисекундах (меньше — лучше).



Кол-во символов	Наивный	Укконен
1000	72	1
10000	7028	16
100000	911808	185

Из тестов ясно видно, что наивный алгоритм работает за $O(n^2)$, а алгоритм Укконена — за линейное время.

Недочёты

Эта лабораторная работа стала первой, выполненной в нескольких файлах и задействующей систему сборки *Make*, однако всё ещё есть недостатки, например заголовочный файл содержит и определение, и реализацию методов, а их можно было бы разбить на два файла.

Выводы

Проделав лабораторную работу, познакомился с практическим применением суффиксного дерева, а также с алгоритмом Укконена, который делает эту структуру данных пожалуй наиболее удобным решением задач поиска множества неизвестных образцов в заранее известном тексте.