Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий

**КУРСОВАЯ РАБОТА**

Реализация структуры данных «Суффиксное дерево»

по дисциплине «Алгоритмы и структуры данных»

Выполнил

студент гр. 23531/2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лесничая М.Д.

*(подпись)*

Принял:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алексюк А.О.

(*подпись*)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2018 г.

Санкт-Петербург

2018

Санкт-Петербургский государственный политехнический университет

**ЗАДАНИЕ**

**НА ВЫПолнение курсовой работы**

студенту группы 23531/2 Лесничей Марии Дмитриевне

*(номер группы)* *(ФИО)*

***1. Тема проекта (работы):*** Реализация структуры данных «Суффиксное дерево» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***2. Срок сдачи законченной работы*** 18 декабря 2018

***3. Исходные данные к работе***: IDE: IntelliJ Idea

***4. Содержание пояснительной записки***: введение, основная часть (текст программы, описание программы, испытания программы), заключение, список используемых источников.

***Дата получения задания***: «18»  сентября      2018 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алексюк А.О.

*(подпись)*

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лесничая М.Д.

*(подпись)*

\_\_\_\_\_\_\_\_\_\_\_\_

*(дата)*

**Содержание**

Введение 4

Реализация суффиксного дерева

1.1 Текст программы7

1.2 Описание программы7

1.3 Тестирование программы8

Заключение9

Список используемых источников10

**ВВЕДЕНИЕ**

Cуффиксное дерево - это trie-подобная структура данных, представляющая все суффиксы строки. Такие поисковые деревья играют центральную роль во многих поисковых алгоритмах для строк.

Входные данные для алгоритма суффиксного дерева – это строка *s*, состоящая из n символов *s[0], s[1], …, s[n-1].* Строки вида *s[i..j] = s[i]s[i+1]…s[j],* где *i* и *j* – некоторые целые числа, называются подстроками строки *s*. Подстроки вида *s[i..n-1],* где *i* – некоторое целое, называются суффиксами. Символ *s[n-1]* принято считать специальным символом.

*Суффиксное дерево для строки s* – это минимальное по числу вершин дерево, каждое ребро которого помечено непустой подстрокой s таким образом, что каждый суффикс s[i..n-1] может быть прочитан на пути из корня до какого-нибудь листа и, наоборот, каждая строка, прочитанная на пути из корня до какого-нибудь листа, является суффиксом s.

В суффиксном дереве не более *2n+1* вершин. В этом можно убедиться, если конструировать дерево последовательной вставкой суффиксов: при вставке очередного суффикса может появиться новый лист и вершина, к которой этот лист крепится. Поскольку *s[n-1]* — это особый символ, в суффиксном дереве ровно *n* листьев.

Визуализация суффиксного дерева для строки *forester* будет выглядеть следующим образом:

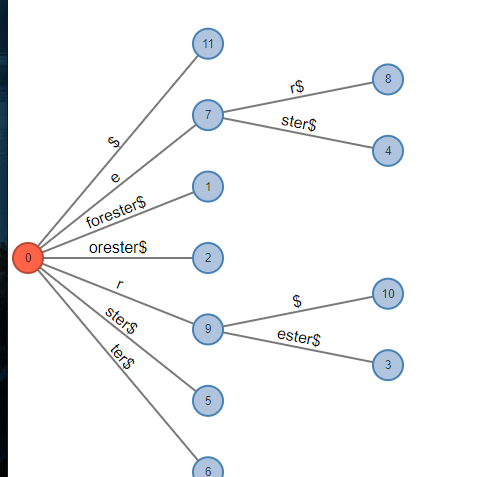


Рис.1

Существуют причины присоединения к оригинальной строке одного или нескольких специальных символов (обычно используется знак доллара «*$*»), когда используются определенные алгоритмы таких структур данных, как суффиксные деревья и суффиксные массивы. Основная причина в случае суффиксных деревьев – это комбинация двух особенностей такой структуры хранения данных:

1) Суффиксные деревья – это нагруженные деревья с видом оптимизации *Patricia*, что означает, что значения ветвей являются строками, состоящими из одного или нескольких символов.

2) Внутренние узлы существуют только в точках ветвления.

Это означает, что можно было бы теоретически попасть в ситуацию, когда метка ветви является префиксом другой ветви:

enter image description here

Рис.2

Идея заключается в том, что черный узел справа является листом, то есть, суффикс здесь заканчивается. Но если в тексте содержится суффикс *аа*, тогда единственный символ *а* также может являться суффиксом. Но нельзя хранить информация, что суффикс заканчивается после первого символа *а*, поскольку *аа* формирует одну непрерывную ветвь дерева (в соответствии со свойством 1). Мы должны были бы ввести промежуточный узел, в котором мы могли бы хранить информацию, например:

enter image description here

Рис.3

Но это было бы недопустимо из-за свойства 2: внутренний узел не должен существовать, если нет точки ветвления.

Проблема решается, если мы можем гарантировать, что последний символ текста является символом, который больше нигде не встречается во всей строке. Знак доллара обычно используется в качестве символа для этого.

Ясно, что, если последний символ больше нигде не встречается, не может быть никакого повторения (например, *aa* или даже более сложного, такого как *abcabc*) в конце строки, поэтому проблема неразветвленных внутренних узлов не появляется. Приведенный выше пример будет выглядеть следующим образом после помещения *$* в конец строки:

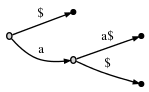


Рис.4

Теперь существует три суффикса: *aa$, a$* и *$,* но ни один не является префиксом другого. После введения внутреннего узла, имеем три листа. Так что, безусловно, преимущество специального символа не в экономии места или повышения эффективности, это - способ гарантировать два свойства выше. Они, когда мы доказываем некоторые полезные характеристики деревьев суффиксов, в том числе тот факт, что его число внутренних узлов является линейным относительно длины строки (это было бы невозможно доказать, если неразветвляющиеся внутренние узлы были бы разрешены).

Самый распространенный способ использования суффиксного дерева – поиск подстрок в заданной строке. В данной курсовой работе выполнена реализация суффиксного дерева с возможностью поиска подстрок в заданной строке, а также поиска в ней суффиксов.

**РЕАЛИЗАЦИЯ СУФФИКСНОГО ДЕРЕВА**

* 1. **Текст программы**

<https://github.com/MariaForester/ProgrammingLabAutumn2018>

* 1. **Описание программы**

**• Класс Node**

Представляет собой узел суффиксного дерева. Включает в себя следующие поля: *edge* (метка ветви дерева, строковая величина), *children* (список детей данного узла, лист целочисленных значений), *index* (индекс узла (его номер), целочисленная переменная).   
 В классе реализованы методы *set* и *get* для каждого из полей, *toString, hashcode*  и *equals* (используемые для строкового представления узла и для их корректного сравнения соответственно), а также дополнительные методы, такие как: *addChild*(позволяет добавить ребенка на последнее место в списке или на определенную позицию)*, addMultipleChildren* (позволяет добавить список детей к существующему)*, hasChild* (проверка наличия ребенка в списке детей) и *hasChildren* (проверка наличия множества детей в списке).

**• Класс SuffixTree**

Представляет собой саму реализацию дерева с такими полями, как *nodes* (список узлов дерева) и *treeName* (имя дерева, задается входной строкой).

Класс содержит следующие методы:

• *get*-методы для извлечения одного конкретного или всех узлов дерева;

• Методы для построения дерева: *addSuffix* (непосредственно добавление суффикса в дерево; выполняется для каждого суффикса входной строки), *split* («разрезание» ветви в определенном месте на несколько ветвей; образуется внутренний узел со своим ветвлением), *createNodeForSubstring*(создание нового узла, когда подстрока оригинальна относительно предыдущих, уже содержащихся в дереве); методы *split* и *createNodeForSubstring* являются вспомогательными для метода *addSuffix*;

• Метод *itHasSuchBranch*, позволяющий проверить, присутствует ли в созданном дереве определенная ветвь;

• Основные функциональные методы суффиксного дерева: *containsSuffix* (определяет, содержит ли входная строка определенных суффикс; используемый вспомогательный метод – *deepSearchSuffix,* осуществляющий поиск вглубь дерева) и *containsSubstring* (определяет, содержит ли входная строка определенную подстроку; используемый вспомогательный метод – *deepSearchSubstring,* осуществляющий поиск вглубь дерева).

**• Класс Drawer**

Изображает в консоли реализованное суффиксное дерево. Если введена пустая строка, вывод сообщение, что дерево пустое и нет никаких данных, которые могут быть выведены.

**• Класс Main**

**2.3. Тестирование программы**

Для программы были написаны два набора тестов. В первом наборе, «черном ящике», идет проверка на корректную работу суффиксного дерева как программы, которая выполняет поиск подстрок и суффиксом во входных заданных строках, то есть, выполняются тесты для рядового пользователя, который не знает, как реализовано дерево и как именно оно работает. Второй набор тестов, «белый ящик», включается в себя такие тесты, как наличие определенных ветвей и количество узлов в дереве, то есть тех данных, которые могут быть проверены лишь при наличии знаний о структуре данного дерева. Также в ручном режиме было протестировано построение дерева для различных строк, используя класс *Drawer*.

**ЗАКЛЮЧЕНИЕ**

Было разработано суффиксное дерево, позволяющее находить суффиксы заданной строки или подстроки в тексте; получены знания и навыки по написанию структуры данных, приобретены детальные знания о строении и работе суффиксного дерева; получен опыт графического вывода дерева на консоль.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

<https://stackoverflow.com/>

<http://brenden.github.io>

<https://www.cs.helsinki.fi/>

<https://habr.com/>

<http://kspt.icc.spbstu.ru/course/algorithms/>

<https://en.wikipedia.org>