Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу «Дискретный анализ»

Студент: К. М. Воронов
Преподаватель: А. А. Кухтичев
Группа: М8О-207Б-19

Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №5

Задача: Необходимо реализовать алгоритм Укконена построения суффиксного дерева за линейное время. Построив такое дерево для некоторых из выходных строк, необходимо воспользоваться полученным суффисным деревом для решения своего варианта задания.

Алфавит строк: строчные буквы латинского алфавита (т.е. от a до z).

Вариант:

Найти образец в тексте используя статистику совпадений.

Формат входных данных

На первой строке располагается образец, на второй — текст.

Формат результата

Последовательность строк, содержащих в себе номера позиций, начиная с которых встретился образец. Строки должны быть отсортированы в порядке возрастания номеров.

1 Описание

Как сказано в [2] - "Суффиксное дерево — бор, содержащий все суффиксы некоторой строки (и только их). Позволяет выяснять, входит ли строка w в исходную строку t, за время O(|w|), где |w| — длина строки w."Для потроения суффиксного дерева за линейное время используется алгоритм Укконена. Идея следующая: проходимся по всем суффиксам каждого префикса с использованием нескольких ускорений. Первое ускорение - продление префикса для существующих вершин за константу, путём увеличения значения переменной. Второе - создание и переход по суффиксным ссылкам: если существует суффикс $x\alpha$, то обязательно есть α, который мы можем пропустить. Третье - в некоторых случаях пропуск прохода по ребрам до определённого момента и вместо этого переход по вершинам. Последнее - если хранить на ребрах символы текста, то потребление памяти будет $O(n^2)$, что никогда нам не даст нам линейное время работы, следовательно, будем хранить только два числа: индексы начала и конца данной строки в тексте.

Для поиска образца в тексте можно использовать статистику совпадений. Как сказано в [3] - "Статистика совпадений - это массив ms, размером с текст, в котором каждый элемент ms[i] является длиной наибольшей подстроки T, начинающейся с позиции i , совпадающей с какой-то подстрокой образца."
1. Построение суффиксного дерева для образца.
2. Заполнение массива ms, с использованием ускорений: переход по суффиксным ссылкам и пропуск рёбер.
3. Прохождение по масиву ms и вывод индексов i, при которых ms[i] равна длине образца. Это и будут его вхождения в текст.

2 Исходный код

Для хранения дерева я пользуюсь списками смежности. Для это я создал класс TSuffixTree с полями текст(string DataString), сами списки смежности(vector< vector< TEdge > > Data), вектор суффиксных ссылок(vector<int> SuffixPtr) и вектор, показывающий сколько символов от корня до текущей вершины(vector<int> PathSize). Также в этом классе есть структура ребра TEdge, где присутствуют левая граница текста(int Left), указатель на правую границу(shared_ptr<int> Right), и вершина, куда это ребро ведёт(int IdTo).

```
struct TSuffixTree {
    struct TEdge {
        int Left;
        shared_ptr<int> Right;
        int IdTo;

        TEdge(int l, shared_ptr<int> r, int id) : Left(l), Right(r), IdTo(id) {}
    };

    string DataString;
    vector< vector< TEdge > > Data;
    vector<int> SuffixPtr;
    vector<int> PathSize;
```

main.cpp	
int FindEdgeChar(int id, char c)	Находит ребро, которе начинается с нужного символа
TSuffixTree(const string & s) : DataString(s), Data(s.size()*2), SuffixPtr(s.size()*2), PathSize(s.size()*2)	Конструктор суффиксного дерева.
vector<size_t> Search(const string & s)	Подсчёт статистики совпадений.

3 Консоль

```
kirill@kirill-G3-3779:~/DA/lab5$ cat test.txt
abc
xabcabc
kirill@kirill-G3-3779:~/DA/lab5$ ./s* <test.txt
2
5
```

4 Тест производительности

Сравним суффиксное дерево с Z-функцей на 10^4 символов образец, 10^6 символов текст и 10^6 образец, 10^7 текст.

```
kirill@kirill-G3-3779:~/DA/lab5$ ./s* <tests/1.in
Суффиксное дерево: 17.983ms
Z-функция: 20.194ms

kirill@kirill-G3-3779:~/DA/lab5$ ./s* <tests/1.in
Суффиксное дерево: 359.836ms
Z-функция: 386.886ms
```

Видно, что они работают примерно одинаково, так как оба алгоритма за линейное время.

5 Выводы

Выполнив пятую лабораторную работу по курсу «Дискретный анализ», я познакомился с такой структурой данных, как суффиксное дерево, узнал, как строить и искать в этом дереве за линейное время. Такой подход поиска подстроки в строке хорошо применим в случаях, если у нас есть один текст и много образцов. Один раз построив дерево для текста, мы просто "прикладываем"образец к нему и находим все вхождения.

Список литературы

[1] Лекции Н.К. Макарова, Московский авиационный институт.

[2] *Суффиксное дерево*
URL: `https://ru.wikipedia.org/wiki/Суффиксное_дерево` (дата обращения: 22.03.2021).

[3] *Приложения суффиксных деревьев*
URL: `https://docplayer.ru/30729776-Prilozheniya-suffiksnyh-derevev.html` (дата обращения: 9.05.2021).