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: Д. С. Ляшун
Преподаватель: А. А. Кухтичев
Группа: М8О-207Б

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 суффиксным деревом для решения своего варианта задания.

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

Вариант 2: Найти в заранее известном тексте поступающие на вход образцы с использование суффиксного массива.

1 Описание

Требуется написать реализацию суффиксного дерева и использовать его для построения суффиксного массива. Как сказано в [1], алгоритм Укконена производит построение неявного суффиксного дерева за время $O(N)$, где N – длина входного текста, который затем преобразуется в явное суффиксное дерево. Под неявным суффиксным деревом строки S понимается дерево, полученное из суффиксного дерева $S\$$ удалением всех вхождений символа \$ из рёбер и самих рёбер в случае получения пустого ребра и вершин, имеющих меньше двух детей.

Идея алгоритма Укконена основана на последовательном построении неявных суффиксных деревьев для каждого префикса текста, т.е. тем самым на каждом этапе будет происходить постепенное расширение рабочего суффиксного дерева. В конце работы рассматривается префикс вида $S\$$, тем самым неявное суффиксное дерево будет перестроено в явное.

На каждом этапе построения неявного суффиксного дерева для текущего префикса текста применяется ряд эвристик, которые позволяют достичь константного времени построения дерева для каждого префикса. В частности, это применение суффиксных ссылок, которые позволяют перейти из одной вершины с путевой меткой $x\alpha$ в другую вершину с путевой меткой α, тем самым не делая тривиального обхода по дереву (который необходим только в случае попадания в корень дерева по суффиксной ссылке); использование рёбер вида (индекс начала, длина ребра) для того, чтобы использовать меньше памяти а также для более быстрого прохода по дереву после перехода по суффиксной ссылке, когда рассматриваются только длины рёбер. Также используются такие идеи, что после добавления символа к ребру к листовой вершине, гарантируется, что в следующих итерациях построения будет происходить тоже самое, и при создании новой вершины-листа также будет гарантироваться, что она будет оставаться листовой и дальше. Все это побуждает ввести общее значение длины рёбер к листовым вершинам, которое при построении следующего неявного суффиксного дерева для следующего префикса будет увеличиваться на 1.

Для построения суффиксного массива на основе уже полученного суффиксного дерева нужно произвести полный обход в глубину по этому дереву, при этом от каждой вершины идти по её рёбрам строго в лексикографическом порядке и при переходе в лист добавлять индекс начала суффикса в массив, таким образом получая отсортированные суффиксы (индексы их начал) в массиве. В свою очередь для поиска вхождения паттерна с использованием суффиксного массива необходимо произвести два бинарных поиска по суффиксному массиву - первый для нахождения крайнего левого значения в массиве с индексом начала суффикса вхождения и второй – для крайнего правого значения, таким образом все начала вхождений паттерна будут присутствовать на этом промежутке.

2 Исходный код

Для представления суффиксного дерева был описан класс $TSuffixTree$ со следующими полями и методами:

tree.hpp	
TSuffixTree(std::string&)	Конструктор, принимающий строку с текстом.
void Search(std::string&, std::vector<int>&)	Процедура поиска вхождений паттерна и записи позиций вхождения во входной вектор.
std::vector<int> Len	Вектор, хранящий длины символьных рёбер к вершинам в суффиксном дереве.
std::vector<int> LPos	Вектор, хранящий позиции начала рёбер к вершинам в суффиксном дереве.
std::vector<int> Link	Вектор, хранящий суффиксные ссылки вершин в суффиксном дереве.
std::vector<std::map<int, int> > Edges	Вектор, хранящий карты с рёбрами вершин в суффиксном дереве.
std::string Word	Строка текста, по которому строилось суффиксное дерево.
static const int INF = -1	Константа, обозначающая длину ребра к листовой вершине в суффиксном дереве.

Для представления суффиксного массива был описан класс $TSuffixArray$ со следующими полями и методами:

tree.hpp	
TSuffixArray(TSuffixTree&)	Конструктор, принимающий построенное суффиксное дерево.
void Search(std::string&, std::vector<int>&)	Процедура поиска вхождений паттерна и записи позиций вхождения во входной вектор.
std::vector<int> Array	Вектор, хранящий суффиксный массив текста.
std::string Word	Строка текста, по которому строился суффиксный массив.

В основном коде программы будет производиться чтение входного текста, создание

суффиксного дерева и затем суффиксного массива по нему, чтение входных паттернов и применение метода суффиксного массива для поиска вхождений, вывод результата:

```
#include "tree.hpp"
#include <iostream>
#include <algorithm>
int main() {
    std::string text, pattern;
    std::cin >> text;
    text += "$";
    NMyStd::TSuffixTree tree(text);
    NMyStd::TSuffixArray array(tree);
    int count = 1;
    while (std::cin >> pattern) {
        std::vector<int> results;
        array.Search(pattern, results);
        if (!results.empty()) {
            std::sort(results.begin(), results.end());
            std::cout << count << ": ";
            for (int i = 0; i < results.size(); ++i) {
                std::cout << results[i];
                if (i + 1 != results.size()) {
                    std::cout << ", ";
                }
            }
            std::cout << std::endl;
        }
        ++count;
    }
}
```

3 Консоль

```
dmitry@dmitry-VirtualBox:~/Work_place/DA_labs/lab5$ make
g++ -std=c++11 -O2 -Wextra -Wall -Werror -Wno-sign-compare -Wno-unused-result
-pedantic -c tree.cpp
g++ -std=c++11 -O2 -Wextra -Wall -Werror -Wno-sign-compare -Wno-unused-result
-pedantic tree.o main.cpp -o solution
dmitry@dmitry-VirtualBox:~/Work_place/DA_labs/lab5$ cat test.txt
abcdabc
abcd
bcd
bc
dmitry@dmitry-VirtualBox:~/Work_place/DA_labs/lab5$ ./solution <test.txt
1: 1
2: 2
3: 2,6
```

4 Тест производительности

Тест производительности представляет из себя сравнение времени построения и времени поиска между суффиксным массивом и суффиксным деревом. В каждом тесте число входных паттернов соразмерно размеру входного текста.

Размер текста	Время построения суффиксного дерева	Время построения суффиксного массива	Суммарное время поиска в суффиксном дереве	Суммарное время поиска в суффиксном массиве
10^2	123 мкс	66 мкс	69 мкс	104 мкс
10^3	13188 мкс	708 мкс	4413 мкс	5493 мкс
10^4	29860 мкс	15685 мкс	425274 мкс	453950 мкс

Как видно, построение суффиксного дерева по времени немного уступает времени построения суффиксного массива. Возможно, это связано с тем, что в суффиксном дереве происходит работа с большим объемом памяти и в общей сложности выполняется большее число операций – незначащая константа в $O(N)$ на данных тестах превосходит $log_2 N$, что делает алгоритм построения суффиксного дерева медленнее алгоритма построения суффиксного массива, работающего за $O(Nlog_2 N)$. Однако по времени поиска суффиксное дерево, наоборот, выполняется чуть-быстрее, поскольку асимптотика поиска составляет $O(|P|)$ в отличие от $O(|P|log_2|T|)$ в суффиксном массиве, где $|P|$ - длина входного паттерна, $|T|$ - длина текста.

5 Выводы

Выполнив пятую лабораторную работу по курсу «Дискретный анализ», я научился реализовывать суффиксные деревья с помощью алгоритма Укконена и использовать его для решения задачи поиска набора паттернов в тексте, а также строить суффиксный массив по уже построенному суффиксному дереву и использовать его для решения той же задачи. Основную трудность для меня составила правильная реализация алгоритма Укконена, в частности, корректное добавление новых внутренних вершин и изменение суффиксных ссылок. Также мне пришлось подумать над тем, как лучше всего реализовать алгоритм получения суффиксного массива из суффиксного дерева.

Список литературы

[1] Д. Гасфилд. *Строки, деревья и последовательности в алгоритмах.* — Издательский дом «БХВ-Питербург», 2003. Перевод с английского: И. В. Романовского. — 658 с. (ISBN 0-521-58519-8 (рус.))