Лабораторная работа № 5 по курсу дискретного анализа: суффиксные деревья

Выполнил студент группы 08-208 МАИ *Сизонов Артём.*

Условие

Кратко описывается задача:

1. Общая постановка задачи: 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 образца с использованием статистики совпадений (вариант 3).

Метод решения

Первым делом строится суффиксное дерево. Оно представляет из себя вектор вершин. После этого поиском в ширину определяются глубины вершин. Далее осуществляется заполнение вектора статистики совпадений:
1. В первую очередь ищется ветка, по которой можно выйти из корня.
2. Далее происходит проход по веткам дерева вглубь со сравнением с текстом. Сравнение происходит до тех пор, пока не придём в лист, или пока не закончится текст, или пока не найдём первое несовпадение символов.
3. В случае прихода в лист или в вершину, из которой некуда выйти, происходит добавление в конец вектора статистики совпадений числа, равного глубине вершины, на которой мы находимся. Далее происходит проход до суффиксным ссылкам. В худшем случае приходим в корень.
4. В случае несовпадения символов происходит добавление в конец вектора статистики совпадений числа, равного глубине вершины минус единица. Далее происходит прыжок по счётчику и последующее сравнение.

Описание программы

Программа написана одним файлом:
struct TNode — тип вершины суффиксного дерева.
Построение массива статистики совпадений происходит с использованием функции *Jump*.
Основная часть этого алгоритма написана в *main*.
Построение суффиксного дерева происходит с использованием функций *GetState, Split, GetLink, ContinueSuffTree, GetDepths, CreateSuffTree* и *struct TCurrentState*.

Бенчмарк:
При увеличение общего количества символов во входных данных в 10 раз время работы также увеличивается примерно в 10 раз.

```
bsb@dell:~/Рабочий стол/Информатика/Дискретный анализ/Лабораторные/LabN5$ time cat test1 | ./stf.out

real    0m0.051s
user    0m0.004s
sys     0m0.000s
bsb@dell:~/Рабочий стол/Информатика/Дискретный анализ/Лабораторные/LabN5$ time cat test2 | ./stf.out

real    0m0.029s
user    0m0.020s
sys     0m0.004s
bsb@dell:~/Рабочий стол/Информатика/Дискретный анализ/Лабораторные/LabN5$ time cat test3 | ./stf.out

real    0m0.143s
user    0m0.136s
sys     0m0.000s
bsb@dell:~/Рабочий стол/Информатика/Дискретный анализ/Лабораторные/LabN5$ time cat test4 | ./stf.out

real    0m0.911s
user    0m0.888s
sys     0m0.012s
bsb@dell:~/Рабочий стол/Информатика/Дискретный анализ/Лабораторные/LabN5$ time cat test5 | ./stf.out

real    0m8.885s
user    0m8.824s
sys     0m0.068s
```

Дневник отладки

Посылка 1: *Status:* Compilation error. Решение: Исправлены ошики компиляции: неиспользуещиеся переменные.
Посылка 2: *Status:* Wrong answer at test 02.t. Проблема: Была допущена ошибка в алгоритме поиска. Неполностью реализовывался прыжок по счётчику.
Посылка 3: *Status:* Wrong answer at test 05.t. Проблема: Часть проблемы решена. Но всё ещё допущена ошибка в алгоритме поиска. Неполностью реализовывался прыжок по счётчику. Доработки не исправили ошибку.
Посылка 4: *Status:* Wrong answer at test 05.t. Проблема: Допущена ошибка в алгоритме поиска. Неполностью реализовывался прыжок по счётчику. Доработки не исправили ошибку.
Посылка 5: *Status:* Wrong answer at test 05.t. Проблема: Допущена ошибка в алгоритме поиска. Неполностью реализовывался прыжок по счётчику. Доработки не исправили ошибку.
Посылка 6: *Status:* Wrong answer at test 05.t. Проблема: Допущена ошибка в алгоритме поиска. Неполностью реализовывался прыжок по счётчику. Доработки не исправили ошибку.
Посылка 7: *Status:* Time limit exceeded at test 10.t. Проблема: Программа выполняет слишком много излишних копирований, неэффективно вычисляется глубины вершин.
Посылка 8: *Status:* Time limit exceeded at test 10.t. Проблема: Программа выполняет слишком много излишних копирований.
Посылка 9: *Status:* OK.

Недочёты

Выявленных недочётов нет.

Выводы

Суффиксное дерево используется в таких задачах, как точное совпадение строк, поиск подстроки в строке, наибольная общая подстрока двух строк, построение суффиксного массива и прочих задачах. Построение суффиксного дерево происходит за $O(m)$, где m — длина текста. Поиск вхождения образца в текст происходит за $O(n)$, где n — длина образца. В случае статистики совпадений всё наоборот, так как в суффиксное дерево заносится образец, а не текст. Из этого следует, что статистика совпадений приводит к уменьшению требуемой памяти, так как образец, обычно, намного меньше, чем текст. Также существуют несколько алгоритмов построения суффиксного дерева: Укконена, Вайнера, Мак-Крейга.