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: В. П. Будникова
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 суффиксным деревом для решения своего варианта задания.

Вариант алгоритма: Линеаризовать циклическую строку, то есть найти минимальный в лексикографическом смысле разрез циклической строки.

Вариант алфавита: `строчные буквы латинского алфавита (т.е. от a до z).`

Формат входных данных: Некий разрез циклической строки.

Формат результата: Минимальный в лексикографическом смысле разрез.

1 Описание

Как сказално в [1]: « 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» В данной лабораторной работе требуется построить суффиксное дерево с помощью алгоритма Укконена. Для хранения узлов создана структура TNode, в которой хранятся: левая, правая граница подстроки, которая лежит на ребре этого узла, номер этого узла(этой вершины), суффиксная ссылка и структура map, в которой будут хранится следующие вершины с ребрами, ключ это та буква, с которой начинается ребро, значение это ее номер. Также создана структура Iter, которая нужна для передвижения по дереву во время вставки. В ней хранятся: текущая вершина(вершина, в которой мы находимся в данный момент, ключ(буква), которую нужно найти(возможно) в одном из следующих ребер, переменная длины, которая обозначает подстроку, которую мы вставляем на данном шаге, и счетчик количество вставок на данном шаге(нужен для определения суффиксных ссылок). В самом классе суффиксного дерева содержится строка, из которой нужно построить суффиксное дерево, счетчик конца(нужен для определения длины листов), счетчик вставок ребер и указатель на TNode, который нужен для хранения суффиксного дерева. В конструктор подается строка, из которой нужно построить суффиксное дерево. Выделяется память, для хранения вершин. Из [1]: «Суффиксное дерево содержит не более одного листа на каждый суффикс (в точности один с символом-стражем). Каждый внутренний узел должен быть узлом ветвления, следовательно, внутренний узел имеет по меньшей мере двух потомков. Каждое ветвление увеличивает число листьев по меньшей мере на единицу, поэтому мы имеем не более n внутренних узлов и не более n листьев. Каждый узел имеет не более одного предка.» Следовательно для хранения нам потребуется 2n ячеек, где n - длина строки. Рассмотрим работу вставки. При вставке мы читаем сроку с начала, находимся в корне, если с прочитанного символа не начинается ни одно ребро, выходящее из корня, то мы вставляем новое ребро(т. к. суффикс, который заканчивается на эту букву на данном шаге не существует в дереве). При каждом считывании буквы со строки мы увеличиваем счетчик end. При вставке листа, мы присваиваем правой границе длину строки, но при подсчете длины данного ребра(у листа) берем минимально из длины строки и счетчика end. После любого действия(вставки или деления), если было деление до действия, то присваивается суффиксная ссылка предыдущему элементу. Также после каждого действия, проверяется наличие суффиксной ссылки у данного узла, если она существует, то осуществляется переход по ней, если мы находимся в корне, то происходит вставка нового суффикса, начинающегося со следующей буквы. Если буква, прочитанная в начале существует в дереве, то осуществляется поиск для вствки или деления. При делении запоминается номер вершины, чтобы на следующем шаге(возможно) присвоить суффиксную ссылку этому узлу. В деструкторе очищается выделенная память.

Алгоритм поиска минимального в лексикографическом смысле разреза циклической строки. На вроде строка дублируется, что означает, что каждый суффикс, длины n, будет каким-либо разрезом строки. Далее из этой строки строится дерево. После построения осуществляется проход по суффиксному дереву, каждый раз проходясь по ребру, начинающемуся с минимальной, в лексикографическом смысле буквы. Проход осуществляется до тех пор, пока количество считанных букв не будет равно длине изначальной строки.

2 Исходный код

Структуры	
struct TNode	Структура для хранения узлов
struct TIter	Структура для итератора
class SuffTree	
SuffTree(std::string &s)	Конструктор (построение дерева)
∼ SuffTree()	Деструктор
void Print()	Печать дерева
void Task(long length)	Поиск минимального в лексикографическом смысле разреза циклической строки

```
1  void Task(long length) {
2    long i = 0;
3    long len;
4    while (length > 0) {
5      char min = 'z' + 1;
6      long j = -1;
7      for (std::unordered_map<char, long>::iterator it = vertexes[i].next.begin(); it !=
           vertexes[i].next.end(); ++it) {
8        if (std::get<0>(*it) < min && std::get<0>(*it) != '$') {
9          min = std::get<0>(*it);
10         j = std::get<1>(*it);
11       }
12     }
13     i = j;
14     len = vertexes[i].r - vertexes[i].l;
15     if (len < length) {
16       std::cout << str.substr(vertexes[i].l, len);
17       length -= len;
18     } else {
19       std::cout << str.substr(vertexes[i].l, length);
```

```
20      return;
21    }
22  }
23 }
24
25 int main() {
26    std::ios::sync_with_stdio(false);
27
28    std::string str;
29    std::cin >> str;
30    if (str == "") {
31        return 0;
32    }
33    long len = str.length();
34    str = str + str;
35    str.push_back('$');
36
37    SuffTree Tree(str);
38    Tree.Task(len);
39    std::cout << std::endl;
40
41    return 0;
```

3 Консоль

```
Lera:SuffTr valeriabudnikova$ ./lab5
aaba
 $
   aaba$
   $
 ba
$
        aaba$
        $
     ba
     aba$
   a
     aaba$
     $
   ba
  a

aaab
```

4 Тест производительности

Сравним время работы данного алгоритма с наивным алгоритмом, который переставляет буквы по одной и сравнивает строки. Сравнение происходит на стоках, содержащих повторение букв и подстрок, длинной 18500 символов

```
Lera:SuffTr valeriabudnikova$ make t
./time <test.txt
lab5: 9e-06 s
naive: 0.001913 s
```

Сравним на 185000 символах:

```
Lera:SuffTr valeriabudnikova$ make t
./time <test.txt
lab5: 7.6e-05 s
naive: 0.152269 s
```

Алгоритм выиграл у наивного, так как сложность наивного алгоритма $O(n^2)$, а сложность алгоритма поиска строки по суффиксному дереву $O(n)$

5 Выводы

Выполнив четвертую лабораторную работу по курсу «Дискретный анализ», я реализовала суффиксное дерево, научилась работать с данной структурой, а также реализовала алгоритм поиска минимального в лексикографическом смысле разреза циклической строки.

Список литературы

[1] *Суффиксное дерево — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Суффиксное_дерево` (дата обращения: 10.05.2021).