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.В. Семенов
Преподаватель: А.А. Кухтичев
Группа: М8О-206Б
Дата:
Оценка:
Подпись:

Москва, 2019

Лабораторная работа №5

Задача: 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.

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

Вариант здания: 5: Поиск наибольшей общей подстроки.

Найти самую длинную общую подстроку двух строк.

1 Описание

Программа должна обрабатывать две строки входного файла, строить по ним обобщенное суффиксное дерево и находить в нем наибольшую общую подстроку двух строк.

Входные данные: две строки.

Выходные данные: на первой строке нужно распечатать длину максимальной общей подстроки, затем перечислить все возможные варианты общих подстрок этой длины в порядке лексикографического возрастания без повторов.

2 Исходный код

Основные этапы написания кода:

1. Анализ поставленной задачи
2. Сбор необходимой информации для реализации
3. Написание кода:
   (a) Определение классов и функций
   (b) Написание конструкторов
   (c) Написание процедур добавления суффикса в дерево
   (d) Реализация обхода дерева в глубину
4. Тестирование кода, отладка
5. Составление отчета

Для хранения подстрок мы создадим класс TSTNode, каждый элемент которого будет содержать $map < char, TSTNode* >$ содержащий следующие переходы и два числа: *begin* и *end* для экономии места.

Основная процедура построения - это PushBack. Я использовал алгоритм Укконена. Чтобы построить дерево мы должны поочередно вызвать процедуру, передавая ей число от 0 до кол-ва литер -1. Процедура сама найдет нужные строки в веденном тексте.

Вторая важная функция - это поиск наибольшей общей строки в дереве. По мере своего продвижения она помещает пройденные ноды в deque, чтобы затем можно было легко восстановить строки, скрытые за абстракцией дерева.

```
1
2  void TSTree::PushBack1(int pos_ch) {
3      how_much_left++;
4      char new_ch = text1[pos_ch]; // char to be pushed
5         suffix_link_change = root;
6
7      while(how_much_left) {
8          if (active_length == 0) {
9              active_char_pos = pos_ch;
10         }
11         std::map<char, TSTNode* >::iterator pos = active_node->next_nodes.find(text1[
               active_char_pos]);
12
13         TSTNode *next = nullptr;
14         if (pos == active_node->next_nodes.end()) {
```

```
15            TSTNode *leaf = new TSTNode(pos_ch, textsize1 - pos_ch, true,false);
16            active_node->next_nodes[text1[active_char_pos]] = leaf;
17            add_link(active_node);
18        // we found link, 2 ways
19        } else {
20            next = pos->second;
21            // if we can go dipper
22            if (active_length >= next->length) {
23                active_char_pos += next->length;
24                active_length -= next->length;
25                active_node = next;
26                continue;
27            }
28            // 3 rule, char (prolonged) nalready in node, have to know nex, false,
                  truet
29
30            if( text1[next->begin + active_length] == new_ch) {
31                ++active_length;
32                add_link(active_node);
33                break;
34            }
35
36            TSTNode *split = new TSTNode(next->begin, active_length, true, false);
37            TSTNode *leaf = new TSTNode(pos_ch, textsize1-pos_ch, true, false);
38
39            active_node->next_nodes[text1[active_char_pos]] = split;
40
41            split->next_nodes[new_ch] = leaf;
42            next->begin += active_length;
43            next->length -= active_length;
44
45            split->next_nodes[text1[next->begin]] = next;
46
47
48            add_link(split);
49        }
50
51        how_much_left--;
52        if(active_node == root && active_length != 0) {
53            // going to previous suffix
54            active_length--;
55            active_char_pos = pos_ch - how_much_left + 1;
56        } else {
57            active_node = (active_node->slink) ? active_node->slink : root;
58        }
59    }
60 }
61
62 int TSTree::FindLCS_Node(TSTNode *node, size_t depth) { // ??? &str
```

```
63
64      deq.push_back(node);
65
66      int got = 5;
67      int to_ret = 5;
68      depth = depth + node->length;
69
70      for (std::map<char, TSTNode *>::iterator it = node->next_nodes.begin(); it != node
            ->next_nodes.end(); ++it) {
71          got = FindLCS_Node(it->second, depth); // never got 5
72
73          if (got == 0) to_ret = 0;
74          if (got == 1) {
75              if (to_ret == 5) to_ret = 1;
76              if (to_ret == 2) to_ret = 0;
77          }
78          if (got == 2){
79              if (to_ret == 1) to_ret = 0;
80              if (to_ret == 5) to_ret = 2;
81          }
82      }
83
84      if (to_ret == 5){
85          if (node->is_s1) to_ret = 1;
86          if (node->is_s2) to_ret = 2;
87      }
88
89      if (to_ret == 0) {
90          if (depth > out.first) {
91              out.first = depth;
92              out.second.clear();
93              out.second.insert(GetStr());
94          }
95          if (depth == out.first) {
96              out.second.insert(GetStr());
97          }
98      }
99
100     deq.pop_back();
101
102     return to_ret;
103
104 }
```

3 Консоль

```
denis@denis-Extensa-2510G:~/labs_3sem/labs_da/5lab_release/5lab$ ./5lab
hellohohlandi
hohhel
3
hel
hoh

denis@denis-Extensa-2510G:~/labs_3sem/labs_da/5lab_release/5lab$ ./5lab <text1.txt
998
thtprytrtvuttqtrvtquthtpvtturtuhtrrhtqyxrprthurrpytxurtrxtqyxrprthurrpytrpyrxvhtpw
prtrhpvxvpvtrthtwrqturttrhtrthtrpyrxvhtpwuryurrutryqturtwpxvrrtxtrxqutxvthxrpythtr
prytrtvuttqtrvtqthxhtptrhuqtthtuxrtthxvttwhtvxtwxvthxprytrtvuttqtrvuxtptpttrtvtxtt
rhpvtrtxtththtptrwxthutwrxtttptrxxptptrtpthttvppprxtpthtrxurptxpqutthttqpprytrtvut
tqtrvptthtqttuthxuxtxrutxxprtptxurptxpqutyurptrxuxrrxvhtprttrxrtxxhwthtuxtpyqtutyu
rppuxutpquthwtptpptxtprytrtvuttqtrvphwtvttxvvtwtrttthtwruurttppxvpxpttrtrxrttxttqr
tppqtqyqthhuppqyrputtrxrtthtttqwtrtprtpprtqythupuvutttrtxttthtpvtturtuhtrrhtputhrx
rprthurrpytrttptptpttprrhtqtrtrttyuppttvtqtrtxtxpvupvttvxhrhprprttrtttrxvprxxtprtuth
tprytrtvuttqtrvtqthtpvtturtuhtrrhtpxtxptxtxvqyyttttthtpvtturtuhtrrhtqyxrprthurrpyt
rtttxprppxqhtxpxxthtrthtpttpvutxxxprptuxttxtyxvthtqrqtvptyyttryvthtuxvtrpvtpxpvxth
tpwxththttwxttxpvxxthtpvtturtuthtqutrprqurtvxxxthtpvtturtuthtptrtqpxxthtpvtturtuth
ttvxttrthuqxthtpvtturtuthtqtqprhtrxxthtpvtturtuthtqtryrrttxxxthtpvtturtuthtrpptrqt
trhtpvtturtxpr
```

4 Тест производительности

```
1  std::string s1, s2;
2    s1 = "In the meantime, Tom goes on a picnic to McDougal's Cave with Becky and their
          classmates. However, Tom and Becky get lost and end up wandering in the
         extensive cave complex for the several days, facing starvation and dehydration.
          Becky becomes extremely dehydrated and weak, and Tom's search for a way out
         grows more desperate. He accidentally encounters Injun Joe in the caves one day
         , but is not seen by his nemesis. Eventually, Tom finds a way out, and they are
          joyfully welcomed back by their community. As a preventive measure, Judge
         Thatcher, Becky's father, has McDougal's Cave sealed off with an iron door.
         When Tom hears of the sealing two weeks later, he is horror-stricken, knowing
         that Injun Joe is still inside. He directs a posse to the cave, where they find
          Injun Joe's corpse just inside the sealed entrance, starved to death after
         having desperately consumed raw bats and candle stubs as a last resort. The
         place of his death, and specifically the in situ cup he used to collect water
         from a dripping stalactite, becomes a local tourist attraction. Tom and others
         in the town feel pity at the horribly cruel death, despite Injun Joe's
         wickedness, and a petition is started to the governor to posthumously pardon
         him.";
3    s2 = "A week later, having deduced from Injun Joe's presence at McDougal's Cave
         that the villain must have hidden the stolen gold inside, Tom takes Huck to the
          cave and they find the box of gold, the proceeds of which are invested for
         them. The Widow Douglas adopts Huck, but he finds the restrictions of a
         civilized home life painful, attempting to escape back to his vagrant life. Tom
          tricks him into thinking that he can later join Tom's new scheme of starting a
          robber band if he returns to the widow. Reluctantly, Huck agrees and goes back
          to the widow. ";
4    time_t b1 = clock();
5    TSTree tree = TSTree(s1, s2);
6    tree.FindLCS();
7    time_t end = clock();
8
9    std::cout << "Time is: " << end - b1 << std::endl;
```

длина текста	LCS	нативный	дерево суффиксов
128	4	174	864
624	624	6902	5697
1521	152	8617	7372

Как мы видим, с увеличением количества слов превосходство алгоритма с использованием суффиксного дерева очевидно.

5 Выводы

Выполнив вторую лабораторную работу по курсу «Дискретный анализ», я наконец понял, что такое суффиксные деревья, как их строить и как ими пользоваться. Оказалось, что у этой структуры данных существует множество различных применений, от обычного поиска подстроки в строке до распознания загрязнения ДНК. Однако, эта структура очень сложна для разбора и у меня ушло много времени, чтобы понять и грамотно реализовать ее.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))

[2] Дэн Гасфилд *Строки, деревья и последовательности в алгоритмах. Информатика и вычислительная биология.* — Издательство Невский Диалект БЧВ-Петербург. Перевод с английского И.В.Романовского.

[3] *Статья на Stackoverflow.com* https://stackoverflow.com/questions/9452701/ukkonens-suffix-tree-algorithm-in-plain-english