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: Р. Р. Гаптулхаков
Преподаватель: Н. С. Капралов
Группа: М8О-308Б

Дата: 12.09.21
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №5

Задача:

Вариант №5: Поиск наибольшей общей подстроки.

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).

Найти самую длинную общую подстроку двух строк.
Формат входных данных: Две строки.

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

1 Описание

Алгоритм Укконена строит неявное суффиксное дерево T_i для каждого префикса $S[1..i]$ строки S, начиная с T_1 и увеличивая i на единицу пока не будет постороено T_m. m - длина строки. Настоящее суффиксное дерево получается из T_m и строится за время $O(m)$.[1]

Обычно при описании алгоритма за линейное время рассказывают алгоритм построения за $O(m^3)$, но при использовании некоторых улучшений, мы сможем улучшить время работы до линейного.
Также при разработке алгоритма Мак-Крейга были введены трюки, которые позволяли существенно улучшить протранственную сложность алгоритма.

Построение суффиксного дерева основано на трёх правилах:

1. В текущем дереве пусть β кончается в листе. Это значит, что путь от корня с меткой β доходит до конца «листовой» дуги. При изменение дерева нужно добавить к концу этой листовой дуги символ $S(i+1)$.

2. Ни один путь из конца строки β не начитаенся символом $S(i+1)$, но по крайней мере один начинающийся от туда путь имеется.
   В этом случае должна быть создана новая листовая дуга начинающаяся в конце β и помеченная сисволом $S(i+1)$. При этом, если β кончается внутри дуги, должна быть создана новая вершина. Листу в конце листовой дуги сопоставляется номер j.

3. Некоторый путь из конца строки β начинается символом $S(i+1)$. В этом случае сторока β $S(i+1)$ уже имеется в текущем дереве, так что ничего не надо делать.

2 Исходный код

Таблица методов и функций

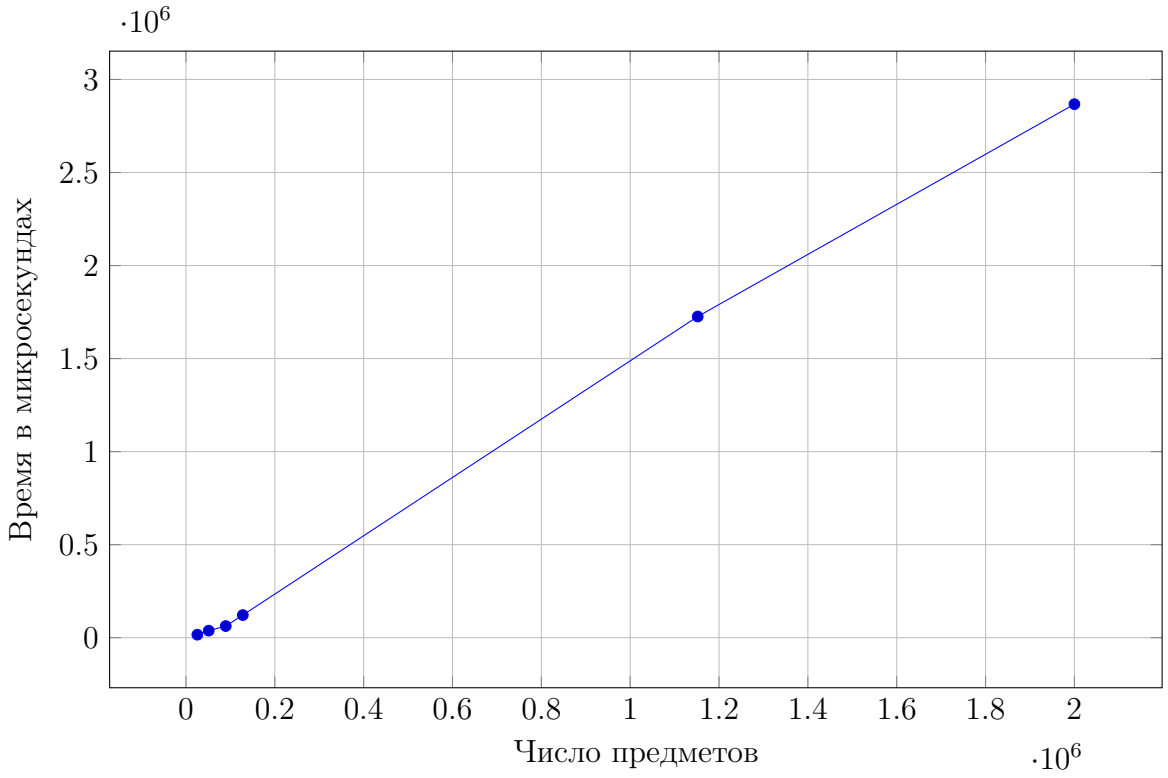
main.cpp	
Функция	Описание
int main()	Главная функция, начальная точка работы программы.
suffixTree.hpp	
Тип данных	Описание
class TSuffixNode	Узел дерева, который хранит левый и правый индексы в строке, номер строки, суффиксную ссылку, ссылки на детей
struct LCS	Хранит необходимую информацию для поиска максимальной подстроки
class TSuffixTree	Дерево
void Build()	Функция построения
void StartPhase(long long i)	начало новой фазы
std::set<int> MarkUp(TSuffixNode* node)	Расставляет номера строк в узлах

```
1  #include <map>
2  #include <vector>
3  #include <iostream>
4  #include <sstream>
5  #include "suffixTree.hpp"
6  #include <chrono>
7  using duration_t = std::chrono::microseconds;
8  int main() {
9      //std::chrono::time_point<std::chrono::system_clock> start, end;
10     //int res_time = 0;
11     //start = std::chrono::system_clock::now();
12     //std::ios::sync_with_stdio(false);
13     std::cin.tie(nullptr);
14     std::cout.tie(nullptr);
15     std::string pattern1, pattern2;
16     std::cin >> pattern1 >> pattern2;
17     TSuffixTree st(pattern1, pattern2);
18     //end = std::chrono::system_clock::now();
19     //res_time += std::chrono::duration_cast<duration_t>( end - start ).count();
20     //std::cout << "Time: " << res_time << "*10^-6 seconds" <<'\n'; //st.DisplayTree();
21     return 0;
22 }
```

3 Тест производительности

1 Графики

Синий: **line**



2 Консольный вывод

rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/lab5DA ./solution < test25k.txt
Time: 16088mcseconds
rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/lab5DA ./solution < test50k.txt
Time: 37860mcseconds
rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/lab5DA ./solution < test90k.txt
Time: 121550mcseconds
rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/lab5DA ./solution < test128k.txt
Time: 121550mcseconds
rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/lab5DA ./solution < test1m.txt
Time: 1725984mcseconds
rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/lab5DA ./solution < test2m.txt
Time: 2867144mcseconds

4 Выводы

Реализация алгоритма Укконена была одна из самых интерестных задач курса. Суффиксное дерево является хорошим алгоритмов для выполнения поисковых задач, однако стоит отметить грамоскость и сложность реализации.
Алгоритм предложенный Укконеном был самым простым из всех, но не самым эффективный. Поэтому были разработаны некоторые улучшения, которые позволили достигнуть лучшие алгоритмы по времени в наихудших случаях.

Список литературы

[1] Ден Гасфилд. *Строки деревья и подпоследовательности в алгоритмах: информатика и вычислительная биология.* — Издательский дом «Невский диалект», 2003. Перевод с английского: И. В. Романовский. — 654 с. (ISBN 5-7940-0103-8 (рус.))

[2] 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.))

[3] Дональд Э. Кнут. *Искусство программирования. Том 3. Сортировка и поиск, 2-е издание.* — Правообладатель «Диалектика-Вильямс», 2018. Перевод с английского: И. В. Красиков, В. Т. Тертышный. — 834 с. (ISBN 978-5-8459-0082-1, 0-201-89685-0)

[4] *Ukkonen's algorithm*
URL: `https://neerc.ifmo.ru/wiki/index.php?title=Ukkonen's_algorithm` (дата обращения: 04.09.21)

[5] *EMAX*
URL: `https://e-maxx.ru/algo/ukkonen` (дата обращения: 04.09.21)