Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу «Дискретный анализ»

Студент: Б. И. Вепринцев
Преподаватель: А. А. Кухтичев
Группа: М8О-207Б

Дата:
Оценка:
Подпись:

Москва, 2019

Лабораторная работа №4

Задача: Требуется разработать программу, осуществляющую ввод образцов и текста, далее нахождение данных образцов в тексте. Поиск образцов в тексте должен осуществляться методом Ахо-Корасик. На выходе мы должны получить три числа для каждого найденного образца 1 - номер строки, 2 - позиция в строке, 3 - номер образца.

Вариант алгоритма поиска: Ахо-Корасик.

Вариант алфавита: `Числа от` 0 `до` $2^{32} - 1$.

1 Описание

В данной лаболаторной работе мы используем алгоритм Ахо-Корасик для нахождения подстрок в строке Данный алгоритм можно использовать не только для этого, но и для ,например, Нахождение лексикографически наименьшей строки данной длины, не содержащей ни один из данных образцов или Нахождение кратчайшей строки, содержащей вхождения одновременно всех образцов.

Для реализации данного алгоритма необходимо реализовать бор и конечный автомат для него. Бор - это дерево с добавление суффиксальных ссылок. Для данного алгоритма его эффективнее будет реализовать на массиве. Для ячейки бора нужно реализовать специальную структуру. Оценочное время работы алгорита для данной задачи: $O(n + m)$ где n - это длина текста, а m - размер ответа. Для успешного выполнения условия времени, необходимо улучшить алгоритм, добавив в него хорошие суффиксальные ссылки.

2 Алгоритм

$add(sample, bor)$
$int\ v = 0$
$bor\ tmp$
for $i \leftarrow 0$ **to** $size$
  if$t[v].child.find(sample[i]) == t[v].child.end()$
    $tmp.link = tmp.link_exit = -1$
    $tmp.p = v$
    $tmp.pch = sample[i]$
    $tmp.leaf = false$
    $t.push_back(tmp)$
    $t[v].child[sample[i]] = sz++$
  $v = t[v].child[sample[i]]$
$[v].leaf = true$
$num++$
$t[v].number_of_word = num$
$t[v].lenght = sample.size()$

$go(t, pos, next)$
if$t[pos].go.find(next) == t[pos].go.end()$
  if$t[pos].child.find(next)! = t[pos].child.end()$
    $t[pos].go[next] = t[pos].child[next]$
  else
    $t[pos].go[next] = (pos == 0?0 : go(t,\ get\_link(t,\ pos),\ next))$
$return\ t[pos].go[next]$

$int\ get\_link(t, v)$
if$t[v].link == -1$
  if$v == 0||t[v].p == 0$
    $t[v].link = 0$
  else
    $t[v].link = go(t,\ get\_link(t,\ t[v].p),\ t[v].pch)$
$return\ t[v].link$

3 Исходный код

В первые строчках вводного файла до пустой строки содержиться образцы, которые необходимо найти. По одному образцу на строку, их необходимо добавить в бор. После через пустую строку вводиться текст, в котором следует найти образцы. Для каждого элемента бора создадим специальную структуру *vertex*. Для текста создадим структуру *text*. Так как хранить весь текст нет необходимости, будем хранить лишь последние несколько слов текста, а именно максимальную длину образца.

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <map>
#include <vector>

struct vertex {
    std::map<unsigned int, size_t> child;
    bool leaf;
    int p;
    int pch;
    int link;
    std::map<unsigned int, size_t> go;
    int number_of_word;
    size_t lenght;
    int link_exit;
};

struct text
{
  int line;
  int begin_in_line;
};

int sz;
int num = 0;

void init(std::vector<vertex>& t)
void add(std::vector<unsigned int>& sample, std::vector<vertex>& t)
int go(std::vector<vertex>& t, int pos, unsigned int next)
int get_link(std::vector<vertex>& t, int v)
int link_exit(std::vector<vertex>& t, int v)
int read_samples(std::vector<vertex>& t)
void read_text_and_answer(std::vector<vertex>& t, int max)
int main()
```

Описание методов и функций:

lab4$_6$.*cpp*	
void init(std :: vector < vertex > & t)	Основная функция работы
void add(std :: vector < unsigned int > & sample, std :: vector < vertex > & t)	Добавление в бор нового образца
int go(std :: vector < vertex > & t, int pos, unsigned int next))	Переход по бору
int get_link(std :: vector < vertex > & t, int v)	Установление суффиксальной ссылки для каждого элемента бора
int link_exit(std :: vector < vertex > & t, int v)	Установление "хорошей "суффиксальной ссылки для каждого элемента бора
int read_samples(std :: vector < vertex > & t)	Считывание образцов
int read_text_and_answer(std :: vector < vertex > & t, int max)	Считывание текста и печать ответа
int main()	Основная функция работы

Помимо этого у нас имеется файл *gen_ test.py*, в котором реализован генератор тестов для программы, принимающий на вход имя тестового файла и количество тестовых строк.

4 Консоль

```
boris@boris-VirtualBox:~/study/lab_da/lab_da4$ g++ -std=c++11 -Wall -o lab
./lab4_6.cpp
boris@boris-VirtualBox:~/study/lab_da/lab_da4$ python3 test_gen_lab4.py
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
boris@boris-VirtualBox:~/study/lab_da/lab_da4$ ./lab <test.txt
1,1,2
251,1,1
501,1,5
751,1,6
1001,1,1
1251,1,5
1501,1,9
1751,1,5
2001,1,2
2251,1,9
2501,1,8
2751,1,2
3001,1,1
3251,1,3
3501,1,6
3751,1,9
4001,1,9
4251,1,10
4501,1,10
4751,1,10
5001,1,3
5251,1,9
5501,1,5
5751,1,2
```

```
6001,1,5
6251,1,1
6501,1,2
6751,1,3
7001,1,3
7251,1,2
7501,1,7
7751,1,10
8001,1,4
8251,1,6
8501,1,1
8751,1,5
9001,1,2
9251,1,5
9501,1,4
9751,1,1
```

5 Вывод

Выполнив данную лабораторную работу, я познакомился с алгоритмами поиска образца в строке, а именно алгоритмом Ахо-Корасик, который позволяет находить несколько образцов за линейное время. Так же во время этой работы я перешел на язык c++ и познал несколько, хоть и базовых, но очень полезных фишек. Так же увеличил свои знания о встроенных библиотеках и вещах, которые содержаться в них.