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: Э. Л. Носов
Преподаватель: А. А. Кухтичев
Группа: М8О-307Б
Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №1

Задача: Необходимо реализовать один из стандартных алгоритмов поиска образцов для указанного алфавита.

Вариант алгоритма: Поиск большого количества образцов при помощи алгоритма Ахо-Корасик.

Вариант алфавита: Числа в диапазоне от 0 до $2^{32} - 1$

Запрещается реализовывать алгоритмы на алфавитах меньшей размерности, чем указано в задании.

1 Описание

Идея алгоритма заключается в построении трая, имеющего связи выхода, которые ведут его к паттернам, которые встретились внутри данного паттерна, и связи неудач, которые ведут в вершину другого патерна, являющуюся окончанием наибольшего суффикса пройденного в исходном паттерне пути, являющегося префиксом другого паттерна.

2 Исходный код

В начале подается некоторое колличество паттернов: по одному на строчку вплоть до пустой строки, в это время производится построение трая. После того как трай построен для него строятся связи выхода и неудач: связи неудач всех вершин первого уровня ведут в корень, для всех остальных они строятся следующим образом: происходит откат к родителю, после чего повторяется последовательность «переход по связи неудач - попытка перехода по своему ситмволу», в случае успешного перехода связь неудач устанавливается с этой вершиной, если успешный переход так и не был осуществлен, то связь неудач будет указывать в корень. При построении связей неудач связи выхода либо создаются при установлении связи выхода с терминальной вершиной, либо наследуются, либо у вершины связи неудач уже имеется связь выхода.

```
1  #include<iostream>
2  #include<unordered_map>
3  #include<cstdlib>
4  #include<string>
5  #include<sstream>
6  #include<vector>
7  #include<iterator>
8
9  class TTrie{
10     private:
11
12     struct TTrieNode{
13         std::unordered_map<unsigned int, TTrieNode *> links;
14         TTrieNode * parent;
15         TTrieNode * failure;
16         TTrieNode * exit;
17         unsigned int length;
18         unsigned int number;
19         unsigned int symb;
20
21         TTrieNode(){
22             symb = 0;
23             parent = nullptr;
24             failure = nullptr;
25             exit = nullptr;
26             length = 0;
27             number = 0;
28         }
29     };
30
31     TTrieNode* root;
32
33     TTrieNode* iter;
34
35     unsigned int depth;
36     unsigned int maxDepth;
37     unsigned int quantity;
38
39     public:
40
41     TTrie(){
42         root = new TTrieNode;
43         iter = root;
44         depth = 0;
45         maxDepth = 0;
46         quantity = 0;
47     }
48
```

```
49      void SaveWord(unsigned int& number){
50          iter->number = number;
51          iter->length = depth;
52          iter = root;
53          if(depth > maxDepth) maxDepth = depth;
54          depth = 0;
55      }
56
57      void Add(unsigned int& key){
58          if(iter->links.find(key) == iter->links.end()){
59              iter->links[key] = new TTrieNode;
60              iter->links[key]->parent = iter;
61          }
62          iter = iter->links[key];
63          if(iter->parent == root){
64              iter->failure = root;
65          }
66          iter->symb = key;
67          depth++;
68      }
69
70      void EstConnectionsForLevel(const int& maxLevel, TTrieNode * curNode, int curLevel){
71          if(curLevel != maxLevel ){
72              for(const auto& someVertexPair : curNode->links) {
73                  EstConnectionsForLevel(maxLevel, someVertexPair.second, curLevel+1);
74              }
75          }
76          if(curLevel == maxLevel){
77              curNode->failure = curNode->parent;
78              while(curNode->failure->failure != nullptr){
79                  curNode->failure = curNode->failure->failure;
80                  if(curNode->failure->links.find(curNode->symb) != curNode->failure->links.end()){
81                      curNode->failure = curNode->failure->links[curNode->symb];
82                      break;
83                  }
84              }
85              if(curNode->failure == nullptr) curNode->failure = root;
86              if(curNode->failure->number != 0){
87                  curNode->exit = curNode->failure;
88              }
89              else if(curNode->failure->exit != nullptr) curNode->exit = curNode->failure->exit;
90          }
91      }
92
93      void EstConnections() {
94          for(unsigned int i = 2; i <= maxDepth; i++){
95              EstConnectionsForLevel(i, root, 0);
96          }
97      }
98
99
100     void Find(const std::vector<unsigned int>& text, std::vector<long long>& occur) {
101         TTrieNode* exitPointer;
102         for(long unsigned int i = 0; i < text.size(); i++){
103             if(iter->links.find(text[i]) != iter->links.end()){
104                 iter = iter->links[text[i]];
105             }
106             else{
107                 while(iter != root){
108                     iter = iter->failure;
109                     if(iter->links.find(text[i]) != iter->links.end()){
110                         iter = iter->links[text[i]];
```

```
                    break;
                }
            }
        }
        exitPointer = iter->exit;
        while(exitPointer != nullptr){
            occur.push_back(i+1);
            occur.push_back(exitPointer->length);
            occur.push_back(exitPointer->number);
            exitPointer = exitPointer->exit;
        }
        if(iter->length != 0){
            occur.push_back(i+1);
            occur.push_back(iter->length);
            occur.push_back(iter->number);
        }
    }


}
void Match(const std::vector<unsigned int>& text, std::vector<long long>& occur) {
    iter = root;
    Find(text, occur);
}

void clearall(TTrieNode * curNode){
    for(auto& someVertexPair : curNode->links){
        clearall(someVertexPair.second);
        delete someVertexPair.second;
    }
}

~TTrie(){
    clearall(root);
    delete root;
}

};

int main(){
    std::ios::sync_with_stdio(false);
    TTrie trie;
    unsigned int number, wordNumber = 0, lineNumber = 1;
    std::string str;
    std::string rts;
    std::stringstream strstr;

    while(getline(std::cin, str)){

        strstr.str(str);
        while(strstr >> number){
            wordNumber++;
            trie.Add(number);
        }
        if(wordNumber == 0) break;
        wordNumber = 0;
        strstr.clear();
        trie.SaveWord(lineNumber);
        lineNumber++;

    }
    trie.EstConnections();
```

```
    std::vector<unsigned int> text;
    std::vector <unsigned int> lineEndings;
    lineEndings.push_back(0);
    while(getline(std::cin, str)) {
        char* p = (char*)str.c_str();
        char* end = p;
        while (true) {
            int cur = strtol(p, &end, 10);
            if (end == p) {
                break;
            }
            p = end;
            text.push_back(cur);
        }
        lineEndings.push_back(text.size());
    }
    std::vector<long long> occur;
    trie.Match(text, occur);
    for(unsigned int i = 0; i < occur.size(); i+=3){
        auto lower = std::lower_bound(lineEndings.begin(), lineEndings.end(), occur[i]-occur[i+1]+1);
        if(lower != lineEndings.end()) lineNumber = std::distance(lineEndings.begin(), lower);
        std::cout << lineNumber << ", " << occur[i]-occur[i+1]+1-lineEndings[lineNumber-1] << ", " << occur
             [i+2] << '\n';
    }
}
```

3 Консоль

```
maloletniydebil@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/DA/lab4$ ./a.out
1 2 3 4 3 2
3 4 3 3 2 2
3 5 6 4 3 6 7
2 3 5 6 7
4 3
3

1 2 3 4 3 2 3 5 6 7 6 7 4 3 4
1, 3, 6
1, 4, 5
1, 5, 6
1, 1, 1
1, 7, 6
1, 6, 4
1, 14, 6
1, 13, 5
```

4 Тест производительности

Тест производительности будет проведен для одного паттерна на тексте размера 1000000 чисел, с длинной паттерна 1000 чисел и колличеством паттернов около 100. В качестве противника будет выступать наивный алгоритм поиска подстроки в строке.

```
maloletniydebil@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/DA/lab4$ g++ test4.cpp -o testpr
maloletniydebil@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/DA/lab4$ time ./a.out < test

real    0m0.299s
user    0m0.103s
sys     0m0.034s
maloletniydebil@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/DA/lab4$ time ./testpr < test

real    0m0.644s
user    0m0.249s
sys     0m0.078s
```

Можно наблюдать, что в такой ситуации алгоритм Ахо-Корасик оказался почти в два раза быстрее наивного алгоритма.

5 Выводы

Выполнив четвертую лабораторную работу по курсу «Дискретный анализ», я научился реализовывать алгоритм Ахо-Корасик, при этом ознакомился со многими средствами STL как для хранения данных так и для манипуляций с ними, кроме того в ходе данной работы мне довелось использовать крайне удобное расширение для визуализации отладочных данных, которое сильно ускорило процесс выполнения работы.

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] *Сортировка подсчётом — Википедия.*
URL: `http://ru.wikipedia.org/wiki/Сортировка_подсчётом` (дата обращения: 16.12.2013).

[3] Список использованных источников оформлять нужно по ГОСТ Р 7.05-2008