Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет «Компьютерные науки и прикладная математика»

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу

«Дискретный анализ»

Тема работы

“Строковые алгоритмы”

Студент : А.А. Дудовцев

Группа : М8О-208Б-22

Оценка : _____

Дата : _____

Подпись : _____

Москва, 2024

1. Постановка задачи

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

Вариант алгоритма: Поиск одного образца основанный на построении Z-блоков.

Вариант алфавита: Числа в диапазоне от 0 до 232 − 1..

2. Описание

Единственное отличие между Z-блоками и Z-функцией является в наличии у первой отрезков совпадения - подстрок паттерна (требуемой для поиска подстроки), совпадающих с префиксом строки (в отрезках совпадения минимум 2 элемента). Далее вместо "Z-блок"будет использоваться "Z-функция".

Z-функция от строки S - массив Z1; : : : ; Zn такой что Zi равен длине наибольшего общего префикса начинающегося с позиции i суффикса строки и самой строки S.

Рассмотрим алгоритм вычисления Z-функции за линейное время:

1) Будем хранить левый и правый индексы (границы) l и r самого правого отрезка совпадения. Пусть i - текущий индекс, для которого мы хотим вычислить Zi(S).

2) Если i <= r - попали в отрезок совпадения, так как строки совпадают, то и Z-блоки для них по отдельности совпадают) Zi(S) = Zi-l. Так как i + Zi(S) может быть за пределами отрезка совпадения, то нужно ограничить значение величиной r - i + 1. (Zi(S) = min(Zi-l, r - i + 1)).

3) i > r - тривиальный алгоритм - паттерн прикладывается к тексту и каждый раз сдвигается (паттерн) на один символ.

4) В конце обновляем отрезок совпадения, если i + Zi(S) - 1 > r (тривиальный алгортим вышел за отрезок совпадения): левая граница l = i, правая граница r = i + Zi(S) - 1.

Рассмотрим алгоритм поиска подстроки в строке с помощью Z-фукнкции:

1) Сканируется строка - паттерн. Числа, входящие в него вносятся в созданный вектор типа пара (значение, строка с которой считалось - это будет нужно для итогового вывода).
2) В конец этого вектора ставится пара <-1, -1 >, чтобы разделить паттерн от текста (в алфавите значение чисел >= 0).

3) Вектор дополняется числами из текста аналогично 1 пункту, но уже сканируются все строки.

4) Считается Z-функция по алгоритму выше.

5) Если подстрока полностью есть в строке, то выводятся номер строки и номер слова в строке, с которого начинается найденный образец. Так происходит для всех строк, где есть полностью паттерн.

Сложность равна **O(p+t)**, где p - длина паттерна, t - длина текста.

3. Исходный код

```
#include <iostream>
#include <vector>
#include <string>
#include <sstream>

using namespace std;

void imedy(vector<pair<long long, long long>>& s, long long& beforeT){
      long long n = (long long) s.size();
      vector<long long> z(n);
      for (long long i=1, l=0, r=0; i<n; ++i){

            if (i <= r){
                long long minh = min(r-i+1, z[i-l]);
                z[i] = minh;
            }
      while (i+z[i] < n && s[z[i]].first == s[i+z[i]].first){
                  z[i]++;
      }
            if (i+z[i] - 1 > r){
                  l = i;
                  r = i+z[i] - 1;
            }
      }
            long long linenow;
          long long lineprev = 0;
          long long find = 1;
      for(long long i = beforeT; i < n; ++i){
          linenow = s[i].second;
        if(linenow > lineprev && lineprev != 0){
            find = 1;
        }
          if (z[i] == beforeT - 1){
              cout << s[i].second << ", " << find << "\n";
          }
          lineprev = linenow;
```

```
            ++find;
        }

}

int main(){
    ios::sync_with_stdio(false); cin.tie(0);
    long long line = 0; string s;
    getline(cin, s);
    vector <pair<long long, long long>> prov;
    stringstream ss(s);
    long long num;
    while (ss >> num){
        prov.push_back(make_pair(num, line));
    }
    ++line;

    prov.push_back(make_pair(-1, -1));

    long long beforeT = prov.size();
    while(getline(cin, s)){
    stringstream ss(s);

        while (ss >> num){
        prov.push_back(make_pair(num, line));
        }
        ++line;
    }
    imedy(prov, beforeT);

    return 0;
}
```

4. Тесты

Тестировать программу буду ручным способом.

qwz@qwz-VirtualBox:~/DA4$ ./main

11 45 11 45 90

0011 45 011 0045 11 45 90 11

45 11 45 90

1, 3

1, 8

qwz@qwz-VirtualBox:~/DA4$ ./main

11 45 11 45 90

0011 45 011 0045 11 45 90 11

45 11 45 90

11 45 11 45 90

1, 3

1, 8

3, 1

Как можно заметить, все работает верно.

5. Выводы

При выполнении четвертой лабораторной работы по курсу «Дискретный анализ» я познокомился с одним из стандартных алгоритмов поиска подстроки в строке, основанном на потроении Z-блоков (функции), смог его реализовать. Надеюсь мне понадобятся знания, полученные при выполнении этой лабораторной работы.