Лабораторная работа № 4 по курсу дискретного анализа: Поиск образца в строке

Выполнил студент группы 08-208 МАИ *Зинин Владислав*.

Условие

Кратко описывается задача:

1. Необходимо реализовать поиск одного образца в тексте с использованием алгоритма Z-блоков. Алфавит — строчные латинские буквы.

Метод решения

Решение представляет собой реализацию эффективной Z-функции, которая имеет O(n), поскольку каждое значение проходится не более двух раз. Данный алгоритм называется эффективным, потому что он значительно быстрее наивного алгоритма, который заключается в обычном подсчете схожих элементов и имеет сложность O(n*n). Эффективный способ основан на использовании информации об уже посчитанных значениях Z-функции для предыдущих значений, а именно использование "границ"и поиск аналогов элементов, находящихся в этих границах. В программе представлены функции ZFunction, представляющая собой эффективный алгоритм Z-функции, NaiveZF, соответственно, наивный и Result для получения результата, оперируя информацией, полученной с помощью Z-функции. Мы ищем в массиве, начиная с позиции текста, значения, равные длине нашего паттерна и выводим их вхождение.

Описание программы

Программа написана в 1 файле: main.cpp.
std::vector<int> ZFunction (std::string zF) - эффективный алгоритм Z-функции
std::vector<int> NaiveZF(std::string text) - наивный алгоритм Z-функции
void Result(std::string text, std::stringpattern) - функция для вывода результата

Исходный код

main.cpp

:

```
#include <iostream>
#include <vector>

std::vector<int> ZFunction (std::string& zF){
    int n = zF.size();
```

```
    int left = 0;
    int right = 0;
    std::vector<int>Res(n);
    Res[0] = zF.size();
    for(int i = 1; i < n; i++){
        if(i > right){
            int j = 0;
            while(zF[i + j] == zF[j] && j < n){
                ++j;
            }
            Res[i] = j;
            if(i + j > right){
                right = i + j - 1;
                left = i;
            }
        }
        else{
            if(Res[i - left] + i - 1 < right){
                Res[i] = Res[i - left];
            }
            else{
                int j = right - i + 1;
                while(zF[j] == zF[j + i] && j + i < n){
                    j++;
                    Res[i]++;
                }
                Res[i] += right - i + 1;
                if(Res[i] + i - 1 > right){
                    right = Res[i] + i - 1;
                    left = i;
                }
            }
        }
    }
    return Res;
}

std::vector<int> NaiveZF(std::string& text){
    int n = text.size();
    std::vector<int> Res(n);
    Res[0] = n;
    for(int i = 1; i < n; i++){
```

```
            int j = 0;
            while(text[0 + j] == text[i + j] && i + j < n){
                ++j;
            }
            Res[i] = j;
        }
        return Res;
    }

    void Result(std::string& text, std::string&pattern){
        std::string zF = pattern + "$" + text;
        std::vector<int> Res = ZFunction(zF);
        for(int i = 0; i < text.size(); i++){
            if(Res[pattern.size() + 1 + i] == pattern.size()){
                std::cout << i << "\n";
            }
        }
    }

    int main(){
        std::cin.tie(nullptr);
        std::cout.tie(nullptr);
        std::ios_base::sync_with_stdio(false);
        std::string text;
        std::string pattern;
        std::cin >> text >> pattern;
        Result(text, pattern);
    }
```

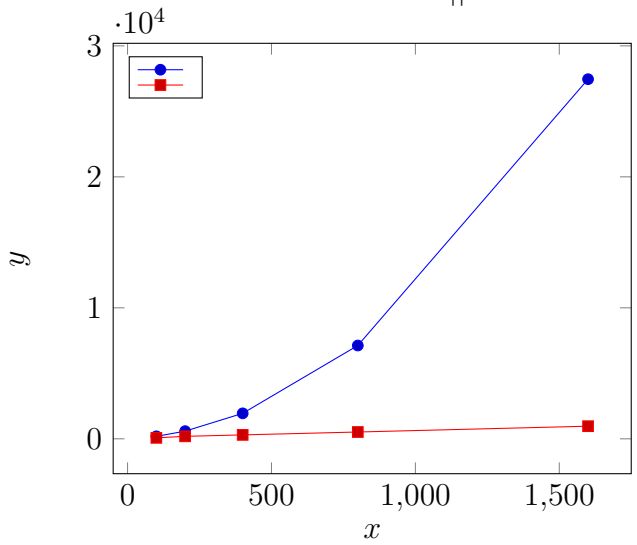
Дневник отладки

Возникли трудности с написанием эффективной Z-функции для случая, когда длина оставшейся строки, схожей с той, что находится в границах, меньше либо равна Z-функции, равной в данном аналоге (случай Res[i - left] + i - 1 >= right). После продолжительных раздумий я смог определить, что позиция следующего элемента за данным определяется по формуле right - i + 1, что так же является значением Z-функции в позиции, являющейся аналогом искомой.

Тест производительности

Для теста производительности я написал генератор, который генерировал текст + паттерн, текст и паттерн состояли из одинаковых букв. Всего тестов - 10000. Паттерн

меньше текста в 10 раз. Каждый новый тест я увеличивал длину текста и паттерна, результаты получились следующими:
(Naive || Z-func)
MAX LENGTH = 100 - 190 ms || 75 ms
MAX LENGTH = 200 - 585 ms || 190 ms
MAX LENGTH = 400 - 1945 ms || 300 ms
MAX LENGTH = 800 - 7124 ms || 520 ms
MAX LENGTH = 1600 - 27456 ms || 964 ms



Синий - наивный алгоритм, красный - эффективный.

Недочёты

Недочетов не обнаружено.

Выводы

В данной лабораторной мною была реализована программа, производящая поиск паттерна в строке. Я познакомился с наивным и эффективным алгоритмами построения Z-функции и использованием её для нахождения паттерна в тексте. Также я сравнил эффективный алгоритм с наивным и убедился в том, что эффективный алгоритм работает за линейное время.