Московский авиационный институт (национальный исследовательский университет)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа по курсу "Дискретный анализ" №4

Студент: Клименко В. М.

Преподаватель: Макаров Н. К.

Группа: М8О-203Б-22

Дата: _____

Оценка: _____

Подпись: _____

Оглавление

Цель работы

Научиться писать алгоритмы поиска образца в тексте, работающие быстрее чем за $O(n^2)$.

Постановка задачи

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

Формат ввода

На первой строке входного файла текст, на следующей – образец. Образец и текст помещаются в оперативной памяти.

Формат вывода

В выходной файл нужно вывести информацию о всех позициях текста, начиная с которых встретились вхождения образца. Выводить следует по одной позиции на строчке, нумерация позиций в тексте начинается с 0.

Общие сведения о программе

Программа состоит из трех функций:

`main` - точка входа программы, в ней происходит считывание образца, текста и вывод результата из функции нахождения образца.

`findPattern` - производит поиска образца в тексте: конструирует новую строку из образца и текста для построения Z-функции, получает Z-функцию из `calculateZArray` и проверяет на наличие полностью совпадающего образца.

`calculateZArray` - построение Z-функции при помощи Z-блоков.

Реализация

main.cpp:

```
#include <iostream>
```

```cpp
#include <vector>
#include <string>


std::vector<size_t> calculateZArray(const std::string& text) {
    size_t textLength = text.length();
    std::vector<size_t> zArray(textLength, 0);

    // z box indecies
    size_t left = 0, right = 0;

    for (size_t index = 1; index < textLength; ++index) {
        // in the z box => copy from it if in bound
        if (index <= right) {
            zArray[index] = std::min(right - index + 1, zArray[index
- left]);
        }

        // boundary check and check prefix with current text position
        while (index + zArray[index] < textLength &&
text[zArray[index]] == text[index + zArray[index]]) {
            ++zArray[index];
        }

        // set up a new z box
        if (index + zArray[index] - 1 > right) {
            left = index;
            right = index + zArray[index] - 1;
        }
    }

    return zArray;
}


std::vector<size_t> findPattern(const std::string& text, const
std::string& pattern) {
    std::string concatenated = pattern + '$' + text;
```

```
    std::vector<size_t> zArray = calculateZArray(concatenated);

    size_t patternLength = pattern.length();

    std::vector<size_t> positions;

    for (size_t index = patternLength + 1; index <
concatenated.length(); ++index) {
        if (zArray[index] == patternLength) { // match
            positions.push_back(index - patternLength - 1);
        }
    }

    return positions;
}


int main() {
    std::string text, pattern;

    std::getline(std::cin, text);
    std::getline(std::cin, pattern);

    std::vector<size_t> positions = findPattern(text, pattern);

    for (size_t position : positions) {
        std::cout << position << '\n';
    }

    return 0;
}
```

Пример работы

Input:

abobaboba

bob

Output:

1

5

Вывод

В ходе лабораторной работы я научился писать поиск подстроки в строке за $O(n)$.