Московский авиационный институт (национальный исследовательский университет)

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

Лабораторная работа по предмету "Дискретный анализ" №4

Студент: Кострюков Е.С.

Преподаватель: Макаров Н.К.

Группа: М8О-207Б-22

Дата:

Оценка:

Подпись:

Москва 2024 г.

Оглавление

Цель работы

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

Постановка задачи

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.

Общий алгоритм решения

1. Чтение входных данных: программа считывает текстовый фрагмент text и шаблон поиска pattern.
2. Объединение образца и текста: она объединяет образец и текст с разделителем $ для создания объединенной строки combined.
3. Вычисление Z-функции: программа вычисляет Z-функцию для объединенной строки combined, которая представляет собой массив Z, где каждый элемент Z[i] указывает на длину наибольшего префикса подстроки, начиная с символа с индексом i, который также является суффиксом объединенной строки.
4. Поиск вхождений образца в тексте: для каждого символа в тексте программа проверяет, равно ли соответствующее значение Z длине образца. Если это так, значит, в тексте найдено вхождение образца. Позиция вхождения выводится на экран.

Реализация

lab4.cpp

```
#include <iostream>

#include <vector>

#include <string>


// Функция для вычисления Z-функции

std::vector<int> computeZ(const std::string& str) {
```

```
    int n = str.length();
    std::vector<int> Z(n);
    int L = 0, R = 0;

    for (int i = 1; i < n; ++i) {
        if (i <= R) {
            Z[i] = std::min(R - i + 1, Z[i - L]);
        }
        while (i + Z[i] < n && str[Z[i]] == str[i + Z[i]]) {
            ++Z[i];
        }
        if (i + Z[i] - 1 > R) {
            L = i;
            R = i + Z[i] - 1;
        }
    }
    return Z;
}

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

    // Объединяем образец и текст с разделителем
    std::string combined = pattern + '$' + text;
    std::vector<int> Z = computeZ(combined);

    int patternLength = pattern.length();
    int textLength = text.length();

    // Ищем позиции вхождения образца в текст
    for (int count = 0; count < textLength; ++count) {
```

```
        if (Z[count + patternLength + 1] == patternLength) {
            std::cout << count << std::endl;
        }
    }

    return 0;
}
```

Пример работы

Input	Output
konikoniko ko	0 4 8
abacaba ab	0 4
aaabaaa a	0 1 2 4 5 6

Вывод

В ходе лабораторной работы была успешно реализована программа на языке C++, которая эффективно ищет вхождения образца в тексте. Для решения этой задачи был использован алгоритм Z-блоков, гарантирующий линейное время выполнения (O(n)). Программа успешно прошла тестирование на различных примерах, подтвердив свою корректность. Полученные результаты демонстрируют, что данный алгоритм является перспективным инструментом для поиска подстрок в больших текстах в различных практических приложениях.