**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе № 4**

**по дисциплине «Построение и Анализ Алгоритмов»**

Тема: «Поиск подстроки в строке»

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Коршков А.А. |
| Преподаватель |  | Жангиров Т. Р. |

Санкт-Петербург

2025

# Цель работы

Изучить принцип работы алгоритма Кнута-Морриса-Пратта (КМП). Написать программу, осуществляющую поиск вхождений подстроки в строку и определяющую, являются ли строки циклическим сдвигом друг друга, найти индекс начала вхождения второй строки в первую.

# Задание

Реализуйте алгоритм КМП и с его помощью для заданных шаблона P (∣P∣ ≤ 15000) и текста T (∣T∣ ≤ 5000000) найдите все вхождения P в T.

Вход:

Первая строка - P

Вторая строка - T

Выход:

индексы начал вхождений P в T, разделенных запятой, если P не входит в T, то вывести −1

**Sample Input:**

ab

abab

**Sample Output:**

0,2

Заданы две строки A (∣A∣ ≤ 5000000) и B (∣B∣ ≤ 5000000).

Определить, является ли А циклическим сдвигом В (это значит, что А и В имеют одинаковую длину и А состоит из суффикса В, склеенного с префиксом В). Например, defabc является циклическим сдвигом abcdef.

Вход:

Первая строка - A

Вторая строка - B

Выход:

Если A вляется циклическим сдвигом B, индекс начала строки B в A, иначе вывести −1. Если возможно несколько сдвигов вывести первый индекс.

**Sample Input:**

defabc

abcdef

**Sample Output:**

3

## Основные теоретические положения

**Описание алгоритма Префикс-функции**

1) Создаем последовательность (в дальнейшем pi) длиной равной длине строки. Все элементы этой последовательности равны 0. Создаем две переменные для хранения индексов (і, j в дальнейшем). Присваиваем ј индекс первого элемента, а і индекс следующего за ним элемента. Переходим к пункту 2.

2) Сравниваем символы на индексах і и ј.

• Символы по индексам равны. Записываем pi[i] = j + 1. Увеличиваем на единицу і и ј.

• Символы не равны. Если ј равно индексу первого элемента, то записываем pi[i] = 0 и увеличиваем і на единицу. Если ј не равно индексу первого элемента, то устанавливаем j = pi[ј-1].

3) Проверяем если і меньше или равен индексу последнего элемента переходим в пункт 2, в противном случае заканчиваем алгоритм.

**Описание алгоритма Кнута-Морриса-Пратта**

1) Вычислить префикс-функцию для образца (в дальнейшем последовательность рі). Объявить две дополнительные переменные для хранения индексов позиции в строке и подстроке соответственно (в дальнейшем і и ј). Присвоить і значение индекса первого элемента. Создаём массив, где будут храниться . Перейти к пункту 2.

2) Проверяем совпадение символов на і позиции в строке и на ј позиции в подстроке.

Символы совпадают. Перейти к пункту 3.

Символы не совпадают. Перейти к пункту 4.

3) Если ј равен индексу последнего символа подстроки заканчиваем алгоритм (поиск успешен). В противном случае увеличить і и ј на единицу и перейти к пункту 2.

4) Выполняем проверку индексов і и ј.

Индекс і равен или больше индекса последнего символа строки закончить алгоритм (поиск неудачен).

Индекс ј равен индексу первого символа в подстроке, то увеличить значение і на единицу и перейти к пункту 2.

Индекс ј не равен индексу первого символа в подстроке. В таком случае установить значение ј = pi[j-1] перейти к пункту 2.

**Описание алгоритма нахождения циклического сдвига.**

Алгоритм нахождения циклического сдвига во многом является модифицированной версией КМП.

1) Проверка длин строк:

Если длины строк text и sub\_text не совпадают, завершить алгоритм (циклический сдвиг невозможен). В противном случае перейти к пункту 2.

2) Вычисление префикс-функции:

Вычислить префикс-функцию для строки sub\_text (массив pi, где pi[i] — длина наибольшего префикса, совпадающего с суффиксом для подстроки sub\_text[0..i]).

3) Инициализация переменных:

Объявить переменную j = 0 (индекс текущего символа в sub\_text). Запустить цикл по переменной i от 0 до 2 \* size(text) - 1.

4) Определение текущего символа в строке:

Вычислить mod\_idx = i % size(text) (эмулирует циклический сдвиг строки text).Это нужно, чтобы при увеличении i можно было вернуться на начало строки.

5) Обработка несовпадения символов:

Пока j > 0 и text[mod\_idx] ≠ sub\_text[j], обновить j = pi[j - 1].

6) Проверка совпадения символов:

Если text[mod\_idx] == sub\_text[j], увеличить j на единицу.

Иначе оставить j без изменений.

7) Проверка завершения поиска:

Если j == size(sub\_text), завершить алгоритм. Циклический сдвиг найден, его величина равна (i - j + 1) % size(text).

Иначе перейти к следующему i (пункт 4), если цикл не исчерпан.

**Оценка сложности по памяти и операциям**

Сложность по времени для алгоритма Кнута-Морриса-Пратта в наихудшем случае O (n + m), где n - длина строки, m - длина подстроки. За O(m) осуществляется построение префикс-функции, а за O(n) – проход по всей строке.

Сложность по памяти O(n), потому что необходимо хранить результат префиксной формы в векторной форме и сравнивать со строкой.

Сложность по времени для алгоритма нахождения индекса циклического сдвига O (2n + m), т.к. необходимо пройтись по строке дважды и построить префикс-функцию.

Сложность по памяти O(n), т.к. необходимо хранить массив префиксов подстроки.

## Выполнение работы

**Описание работы**

Для решения заданиях были написаны три функции.

std::vector<*int*> prefix\_func(*const* std::string &text) – префикс-функция, принимает на входу строку, и вычисляет значения максимальных длин префиксов для каждого элемента (векторная форма). Записывает в вектор значения и возвращает его.

std::vector<*int*> kmp(*const* std::string &text, *const* std::string &sub\_text)

Функция, принимающая на вход строку text и подстроку sub\_text. Возвращается вектор индексов начала вхождений подстроки в строку.

*int* find\_cyclic\_shift(*const* std::string &text, *const* std::string &sub\_text)

Функция, принимающая на вход две строки, первым аргументом принимается та строка, в которой будет осуществляться поиск сдвига, а вторым та, которую будем искать. Возвращает индекс начала вхождения второй строки в первую.

# Тестирование

Таблица 1 – Тестирование алгоритмов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Комментарии |
| 1. | adadadafffaaa | 0, 0, 1, 2, 3, 4, 5, 0, 0, 0, 1, 1, 1 | Префикс-функция. |
| 2. | ab  abab | 0,2 | КМП. |
| 3. | iron  spider | -1 | КМП. В тексте нет вхождений подстроки. |
| 4. | defabc  abcdef | 3 | Цикл. сдвиг. |
| 5. | spider  menace | -1 | Цикл. сдвиг. Нет никакого вхождения подстроки в строку |

# Выводы

Изучен принцип работы алгоритма Кнута-Морриса-Пратта. Написаны программы, корректно решающие поставленные задачи с помощью функции вычисления максимальной длины префикса для каждого символа.

ПРИЛОЖЕНИЕ А  
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

/\*\*

\* @file main.cpp

\* @author Korzik

\* @brief Главный файл программы

\*/

#ifdef \_WIN32

#include <windows.h>

#endif

#include "kmp.hpp"

/\*\*

\* @brief Главная функция программы, содержит два задания (KMP и циклический сдвиг)

\* @return 0

\*/

int main() {

#ifdef \_WIN32

SetConsoleCP(CP\_UTF8);

SetConsoleOutputCP(CP\_UTF8);

#endif

std::cout << "Задание №1 (KMP, найти все индексы)" << std::endl;

std::string p, t; // Подстрока и строка

std::cin >> p >> t; // Вводим подстроку и строку

if (const std::vector<int> result = kmp(t, p); !result.empty()) {

std::cout << "Кол-во найденных вхождений: " << result.size() << std::endl; // Выводим кол-во вхождений

std::cout << "Индексы: ";

for (size\_t i = 0; i < result.size(); ++i) {

// Проходим по всем индексам

std::cout << result[i]; // Выводим результат

if (i < result.size() - 1) std::cout << ","; // Если не последний элемент, то выводим запятую

}

}

else std::cout << "Нет вхождений " << -1; // Если нет совпадений, то выводим -1

std::cout << std::endl << std::endl;

std::cout << "Задание №2 (Циклический сдвиг, найти первый индекс сдвига)" << std::endl;

std::string a, b; // Строки

std::cin >> a >> b; // Вводим строки

std::cout << "Индекс сдвига строки: " << find\_cyclic\_shift(a, b) << std::endl; // Выводим результат

}

Название файла: kmp.cpp

/\*\*

\* @file kmp.cpp

\* @author Korzik

\* @brief Реализация алгоритма Кнута-Морриса-Пратта и поиска циклического сдвига

\*/

#include "kmp.hpp"

std::vector<int> prefix\_func(const std::string &text) {

std::vector pi(text.size(), 0);

int j = 0;

std::cout << "Для первого символа text[0] = '" << text[0] << "' pi[0] = 0" << std::endl;

for (int i = 1; i < text.size(); ++i) {

std::cout << "\nОбработка символа text[" << i << "] = '" << text[i] << "'" << std::endl;

j = pi[i - 1];

std::cout << "\tИзначально j = pi[" << (i - 1) << "] = " << j << std::endl;

while (j > 0 && text[i] != text[j]) {

std::cout << "\ttext[" << i << "] != text[" << j << "] (" << text[i] << " != " << text[j] << "), j = pi["

<< (j - 1) << "] = " << pi[j - 1] << std::endl;

j = pi[j - 1];

}

if (text[i] == text[j]) {

std::cout << "\tСовпадение: text[" << i << "] == text[" << j << "] (" << text[i] << "), увеличиваем j до "

<< (j + 1) << std::endl;

j++;

} else std::cout << "\tСовпадений нет, j остается 0" << std::endl;

pi[i] = j;

std::cout << "\tУстановлен pi[" << i << "] = " << j << std::endl;

}

std::cout << "\nИтоговая префикс-функция: ";

for (size\_t idx = 0; idx < pi.size(); ++idx) {

std::cout << pi[idx];

if (idx != pi.size() - 1) std::cout << ", ";

}

std::cout << std::endl << std::endl;

return pi;

}

std::vector<int> kmp(const std::string &text, const std::string &sub\_text) {

std::vector<int> res\_indexes;

std::cout << "Запуск KMP для поиска \"" << sub\_text << "\" в \"" << text << "\"" << std::endl << std::endl;

int j = 0;

std::cout << "Этап 1: Вычисление префикс-функции для подстроки" << std::endl;

const std::vector<int> pi = prefix\_func(sub\_text);

std::cout << "Этап 2: Поиск подстроки в тексте" << std::endl;

for (int i = 0; i < text.size(); ++i) {

std::cout << "\nТекущий символ текста: text[" << i << "] = '" << text[i] << "'" << std::endl;

while (j > 0 && text[i] != sub\_text[j]) {

std::cout << "\tНесовпадение: text[" << i << "] != sub\_text[" << j << "] (" << text[i] << " != "

<< sub\_text[j] << "), j = pi[" << (j - 1) << "] = " << pi[j - 1] << std::endl;

j = pi[j - 1];

}

if (text[i] == sub\_text[j]) {

std::cout << "\tСовпадение: text[" << i << "] == sub\_text[" << j << "] ("

<< text[i] << "), увеличиваем j до " << (j + 1) << std::endl;

j++;

} else std::cout << "\tСовпадений нет, j остается " << j << std::endl;

if (j >= sub\_text.size()) {

std::cout << "!!! Найдено полное вхождение на позиции " << i - j + 1 << " !!!" << std::endl;

res\_indexes.push\_back(i - j + 1);

j = pi[j - 1];

std::cout << "\tСброс j = pi[" << (sub\_text.size() - 1) << "] = " << j << std::endl;

}

}

return res\_indexes;

}

int find\_cyclic\_shift(const std::string &text, const std::string &sub\_text) {

std::cout << "Поиск циклического сдвига между \"" << text << "\" и \"" << sub\_text << "\"" << std::endl;

if (text.size() != sub\_text.size()) {

std::cerr << "Ошибка: длины строк отличаются (" << text.size() << " vs " << sub\_text.size() << ")" << std::endl;

return -1;

}

int j = 0;

std::cout << "\nЭтап 1: Вычисление префикс-функции для подстроки" << std::endl;

const std::vector<int> pi = prefix\_func(sub\_text);

std::cout << "Этап 2: Поиск циклического сдвига" << std::endl;

for (int i = 0; i < text.size() \* 2; ++i) {

const int mod\_idx = i % text.size();

std::cout << "\nШаг " << i << ": text[" << mod\_idx << "] = '" << text[mod\_idx]

<< "', sub\_text[" << j << "] = '" << sub\_text[j] << "'" << std::endl;

while (j > 0 && text[mod\_idx] != sub\_text[j]) {

std::cout << "\tНесовпадение: text[" << mod\_idx << "] != sub\_text[" << j << "] (" << text[mod\_idx] << " != "

<< sub\_text[j] << "), j = pi[" << (j - 1) << "] = " << pi[j - 1] << std::endl;

j = pi[j - 1];

}

if (text[mod\_idx] == sub\_text[j]) {

std::cout << "\tСовпадение: text[" << mod\_idx << "] == sub\_text[" << j << "] ("

<< text[mod\_idx] << "), увеличиваем j до " << (j + 1) << std::endl;

j++;

} else std::cout << "\tСовпадений нет, j остается " << j << std::endl;

if (j == sub\_text.size()) {

std::cout << "!!! Найден циклический сдвиг: " << i - j + 1 << " !!!" << std::endl;

return i - j + 1;

}

}

std::cout << "\nЦиклический сдвиг не найден" << std::endl;

return -1;

}

Название файла: kmp.hpp

/\*\*

\* @file kmp.hpp

\* @author Korzik

\* @brief Заголовочный файл для kmp.cpp

\*/

#pragma once

#include <iostream>

#include <vector>

/\*\*

\* @brief Функция для вычисления префикс-функции

\* @param text Строка для вычисления префикс-функции

\* @return Значения префикс-функции (векторная форма)

\*/

std::vector<int> prefix\_func(const std::string &text);

/\*\*

\* @brief Функция для поиска подстроки в строке

\* @param text Строка для поиска подстроки

\* @param sub\_text Подстрока для поиска

\* @return Вектор индексов, где найдена подстрока

\*/

std::vector<int> kmp(const std::string &text, const std::string &sub\_text);

/\*\*

\* @brief Функция для поиска циклического сдвига

\* @param text Строка для поиска циклического сдвига

\* @param sub\_text Подстрока для поиска

\* @return Индекс сдвига

\*/

int find\_cyclic\_shift(const std::string &text, const std::string &sub\_text);

Название файла: CMakeLists.txt

cmake\_minimum\_required(VERSION 3.20) # проверка версии CMake

project(kmp) # название проекта

set(CMAKE\_CXX\_STANDARD 20) # стандарт C++

add\_executable(kmp sources/main.cpp sources/kmp.cpp) # исполняемый файл