

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №4
по дисциплине «Построение и анализ алгоритмов»
Тема: Алгоритм Кнута-Морриса-Пратта

Студент гр. 8383

Бессуднов Г. И.

Преподаватель

Фирсов М. А.

Санкт-Петербург

2020

Цель работы

Изучить принцип работы алгоритма Кнутта-Морриса-Пратта для поиска подстроки в строке.

Постановка задачи

Реализуйте алгоритм КМП и с его помощью для заданных шаблона P ($|P| \leq 15000$) и текста T ($|T| \leq 5000000$) 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 AA ($|A| \leq 5000000$) и B ($|B| \leq 5000000$).

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

Вход:

Первая строка – A

Вторая строка – B

Выход:

Если A является 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

Вар. 2. Оптимизация по памяти: программа должна требовать $O(m)$ памяти, где m - длина образца.

Описание алгоритма КМП

На вход алгоритма передается строка-образец, вхождения которой нужно найти, и строка-текст, в которой нужно найти вхождения. Сначала вычисляется префикс-функция строки-образца.

Переменная-счетчик k приравнивается к 0. Переменная-позиция в образце изначально $i = 0$. При каждом совпадении i -го символа образца и k -го символа текста, переменные увеличиваются на 1. Если i равен размеру образца, значит вхождение найдено. Если очередной символ текста не совпал с k -ым символом образца, то обращаемся к массиву с префикс-функциями, смотрим значение префикс-функции для символа, предшествующего не совпавшему, приравниваем индекс символа в образце этому значению.

Описание алгоритма проверки циклического сдвига

Для поиска циклического сдвига используется алгоритм КМП. Его использование обусловлено тем, что если строки циклически, то при склеивании двух копий, в новой строке будет содержаться искомая строка. Важно, чтобы строки были одинаковой длины, это условие проверяется перед вызовом КМП алгоритма. Перед вызовом КМП алгоритма необходимо склеить исходный текст сам с собой и запустить алгоритм, передав ему искомую строку.

Описание структур данных и функций

```
class StringSearcher {
```

```

private:
    std::string text; //строка-текст
    std::vector<int> answers; //вектор для записи найденных
вхождений
    std::vector<int> prefixies; //вектор для хранения префиксов

    void prefix(const std::string &str); //функция для вычисления
префикс-функции

    void printAnswerKMP(); //функция печати результатов КМП
    void printAnswerShift(); //функция печати проверки строк на
цикличность
    void printPrefixies(); //функция печати префиксов
public:
    StringSearcher(); //конструктор

    void KMP(const std::string &pattern); //функция КМП-алгоритма
    void shiftCheck(const std::string &checkString); //функция
проверки строк на цикличность
};

```

Класс, необходимый для поиска подстроки в строке и выполнения проверки на цикличность строк. В нем содержатся функции:

void KMP(const std::string &pattern) - функция поиска вхождений подстроки pattern в строку text. При помощи алгоритма КМП.

void shiftCheck(const std::string &checkString) - проверка на цикличность строк checkString и text.

void prefix(const std::string &str) - функция для нахождения префикс-функции в строке str.

Оценка сложности алгоритмов

Функция вычисления префикс-функции проходится по строке-образцу 1 раз, поэтому требует $O(m)$ времени, где m – длина строки-образца. Процесс поиска вхождений строки-образца в строке-текста выполняется, пока не закончится строка-текст т.е. требует $O(n)$ времени, где n - длина строки-текста. Итого общая оценка сложности по времени алгоритма Кнута-Морриса-Пратта составляет $O(m + n)$.

Для работы, алгоритм вычисляет значение префикс функции для каждого символа строки-образца и хранит эти значения в массиве, следовательно сложность алгоритма по памяти составляет $O(m)$, где m - длина строки-образца.

Сложность алгоритма проверки циклического сдвига по времени составляет $O(3n)$, где n - длина строки, т.к. алгоритм сначала найдет префикс-функцию для одной строки длины n , а затем пройдет по склеенной строке длины $2n$.

Сложность алгоритма по памяти составляет $O(n)$, т.к. высчитывается и хранится префикс-функция для исходной строки размера n .

Тестирование

Пример вывода результата для алгоритма КМР представлены на рис. 1.

Пример вывода результата для алгоритма проверки циклического сдвига представлен на рис. 2.

```

Enter text: abcdcdhjdkababab
For KMP search enter 1
For shift check enter 2
1
Enter pattern for searching: ab

Calculating prefix function for string:
ab
Prefixies:
0 0

KMP algorithm start
Text at index 0 and pattern at index 0 are equal
Text at index 1 and pattern at index 1 are equal
FOUND PATTERN
Text at index 10 and pattern at index 0 are equal
Text at index 11 and pattern at index 1 are equal
FOUND PATTERN
Text at index 12 and pattern at index 0 are equal
Text at index 13 and pattern at index 1 are equal
FOUND PATTERN
Text at index 14 and pattern at index 0 are equal
Text at index 15 and pattern at index 1 are equal
FOUND PATTERN

Patterns are starting from: 0 10 12 14

```

Рисунок 1 - Результат работы алгоритма КМР.

```

Enter text: ghjkl
For KMP search enter 1
For shift check enter 2
2
Enter string for checking: jklgh
Shift check start

Calculating prefix function for string:
jklgh
Prefixies:
0 0 0 0 0

KMP algorithm start
Text at index 2 and pattern at index 0 are equal
Text at index 3 and pattern at index 1 are equal
Text at index 4 and pattern at index 2 are equal
Text at index 5 and pattern at index 3 are equal
Text at index 6 and pattern at index 4 are equal
FOUND PATTERN
Text at index 7 and pattern at index 0 are equal
Text at index 8 and pattern at index 1 are equal
Text at index 9 and pattern at index 2 are equal

First shifted index: 2

```

Рисунок 2 - Результат работы алгоритма проверки циклического сдвига.

Тестирование алгоритма КМР.

№	Input	Output
1	aaaaa a	0 1 2 3 4
2	aaaaaaaaaaaaaaaaaaaaaaaaaaaaa b	No patterns were found
3	abcdegh c	2
4	bfgjhukbfgjkhbfgbfg bfg	0 7 13 16

Тестирование алгоритма поиска циклического сдвига.

№	Input	Output
1	abcd abcd	0
2	abcdef defabc	3
3	sss sss	0
4	f f	0

Код программ приведен в приложении А.

Вывод

В ходе лабораторной работы был реализован на языке C++ алгоритм Кнутта-Морриса-Пратта для поиска подстроки в строке, а также алгоритм для проверки циклического сдвига.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include "pch.h"
#include <iostream>
#include <string>
#include <vector>

class StringSearcher {
private:
    std::string text; //строка-текст
    std::vector<int> answers; //вектор для записи найденных вхождений
    std::vector<int> prefixies; //вектор для хранения префиксов

    void prefix(const std::string &str); //функция для вычисления префикс-функции

    void printPrefixies(); //функция печати префиксов
public:
    StringSearcher(); //конструкторы
    StringSearcher(std::string text);

    void printAnswerKMP(); //функция печати результатов КМП
    void printAnswerShift(); //функция печати проверки строк на цикличность

    void KMP(const std::string &pattern); //функция КМП-алгоритма
    void shiftCheck(const std::string &checkString); //функция проверки строк на
цикличность
};

StringSearcher::StringSearcher(std::string text) : text(text) {

}

StringSearcher::StringSearcher() {
    std::cin >> text;
}

void StringSearcher::KMP(const std::string &pattern) {
    answers.clear();
    prefix(pattern);
    std::cout << "KMP algorithm start" << std::endl;
    for (int k = 0, i = 0; k != text.length(); k++) {
        if (text.at(k) == pattern.at(i)) {
            std::cout << "Text at index " << k << " and pattern at index " << i
<< " are equal" << std::endl;
            k++;
            i++;
            if (i == pattern.length()) {
                std::cout << "FOUND PATTERN" << std::endl;
                i = 0;
                answers.push_back(k - pattern.length());
            }
        } else {
            if (i == 0) {
                k++;
            } else {
                std::cout << "Reset i to " << prefixies.at(i - 1) <<
std::endl;
                i = prefixies.at(i - 1);
            }
        }
    }
}
```



```

    }
}

}

void StringSearcher::shiftCheck(const std::string &checkString) {
    answers.clear();
    std::cout << "Shift check start" << std::endl;
    if (checkString.length() != text.length()) {
        std::cout << "Strings have different length" << std::endl;
        return;
    }
    text += text;
    KMP(checkString);
}

void StringSearcher::prefix(const std::string &str) {
    std::cout << "\n\nCalculating prefix function for string:" << std::endl;
    std::cout << str << std::endl;
    prefixies.resize(str.size());
    int k;
    for (int i = 1; i < prefixies.size(); i++) {
        k = prefixies.at(i - 1);
        while (k > 0 && str.at(i) != str.at(k)) {
            k = prefixies.at(k - 1);
        }
        if (str.at(k) == str.at(i)) {
            ++k;
        }
        prefixies.at(i) = k;
    }

    printPrefixies();
}

void StringSearcher::printAnswerKMP() {
    std::cout << "\n" << std::endl;
    if (answers.empty()) {
        std::cout << "No patterns were found" << std::endl;
    } else {
        std::cout << "Patterns are starting from: ";
        for (auto &index : answers) {
            std::cout << index << " ";
        }

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

void StringSearcher::printAnswerShift() {
    std::cout << "\n" << std::endl;
    if (answers.empty()) {
        std::cout << "Strings are not shifted";
    } else {
        std::cout << "First shifted index: " << answers[0] << std::endl;
    }
}

void StringSearcher::printPrefixies() {
    std::cout << "Prefixies:" << std::endl;
    for (auto &pref : prefixies) {

```

```

        std::cout << pref << " ";
    }
    std::cout << "\n" << std::endl;
}

int main() {
    std::string temp;
    std::cin >> temp;
    StringSearcher stringSearcher(temp);
    std::cin >> temp;

    stringSearcher.KMP(temp);
    stringSearcher.printAnswerKMP();

    stringSearcher.shiftCheck(temp);
    stringSearcher.printAnswerShift();

    return 0;
}

```