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-Морриаса-Пратта

Студент гр. 9382 _____ Рыжих Р.В.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить алгоритм Кнута-Морриаса-Пратта. Написать программы, которые реализуют данный алгоритм, используя полученные знания.

Задание.

Step 1.

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

Step 2.

Заданы две строки A ($|A| \leq 5000000$) 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

Описание алгоритма:

Step 1. Найти вхождение первой строки во вторую.

- В начале вычисляется префикс-функция для первой строки P.
- Префикс-функция для какого-либо символа показывает максимальную длину совпадающего префикса и суффикса подстроки, которая заканчивается этим символом.
- Далее начинается посимвольное сравнение двух строк. Если при сравнении все символы первой строки совпадают со второй, значит нашлось вхождение. В массив записывается индекс начала вхождения.
- Если при сравнении первый символ строки P не совпал с первым символом строки T, то сравнивается первый символ строки P со вторым символом строки T и так далее.
- Если при сравнении не первый символ строки P не совпадает с символом строки T, то следующим символом, с которым будет сравниваться строки P и T, будет символ строки P под индексом префикс-функции предыдущего символа.
- Алгоритм закончится, когда все символы строки будут сравнены.

Step 2. Является ли строка циклическим сдвигом второй строки.

- Используется тот же самый алгоритм, что и в Step 2, однако поиск ведётся в удвоенной первой строке, потому что удвоенная первая строка будет содержать в себе вторую строку, если первая строка является циклическим сдвигом второй строки.
- Алгоритм заканчивается:
  - когда находится первое вхождение.
  - когда просматриваются все символы удвоенной первой строки.
  - когда строки изначально не равны по длине.

Сложность алгоритма:

В обеих программах сложность по памяти равна O(m + n), где m – длина первой строки, n – длина второй строки.

Сложность по времени равна O(m + n), где m – длина первой строки, n – длина второй строки.

Функции и структуры данных:

Структуры данных:

class KMP – класс для реализации алгоритма Кнута-Морриаса-Пратта.

std::vector<int> arrPrefix — массив префик-функций

Функции:

void KMP::Read() — функция для считывания данных.

void KMP::PrintAnswer() — функция, которая печатает ответ на экран

void KMP::Prefix() — фукнция, которая формирует массив значений префикс-функций символов строки P.

void KMP::AlgorithmKMP() — функция, которая реализует алгоритм Кнута-Морриса-Пратта.

void KMP::CyclicShift() — функция, которая проверят, является ли строка A циклическим сдвигом строки B.

Тестирование.

Step 1

Входные данные	Выходные данные
ab abab	0,2
qw qwerty	0
asd qwerty	-1
to rtohtoto	1,4,6

Step 2

Входные данные	Выходные данные
defabc abcdef	3
qwerty asdfgh	-1
qwerty rtyqwe	3
zxcvb vbzxc	3

Выводы.

В результате выполнения работы был изучен алгоритм Кнута-Морриаса-Пратта. Основываясь на полученных знаниях, были написаны программы, которые реализуют данный алгоритм.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл lab4_1.cpp

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

#define INFO

class KMP {
private:
	std::string P, T;
	std::vector<int> arrPrefix;
	std::vector<int> answer;
public:
	KMP() {};
	void Read();
	void PrintAnswer();
	void Prefix();
	void AlgorithmKMP();
};


void KMP::Read() {
#ifdef INFO
    std::cout << "Введите первую строку P:\n";
#endif
    std::cin >> P;
#ifdef INFO
    std::cout << "Введите вторую строку T:\n";
#endif
    std::cin >> T;
}

void KMP::PrintAnswer() {
#ifdef INFO
    std::cout << "\nОтвет:\n";
#endif
    if (!answer.size()) {
#ifdef INFO
        std::cout << "P не входит в T\n";
#endif
        std::cout << -1;
    }
    else {
#ifdef INFO
        std::cout << "Индексы начал вхождений P в T:\n";
#endif
        for (auto a : answer) {
            if (a == answer.front())
                std::cout << a;
            else
```

```
                std::cout << "," << a;
            }
        }
        std::cout << '\n';
}

void KMP::Prefix() {   //префикс-функция
#ifdef INFO
        std::cout << "\nПодсчет префикс-функции:\n\n";
#endif
        int pLength = P.length();
        arrPrefix = std::vector<int>(pLength);
        arrPrefix[0] = 0;
#ifdef INFO
        std::cout << "Значение префикс-функции для символа " << P[0] << " под номером " << 0
<< " равно " << 0 << "\n\n";
#endif
        for (int i = 1; i < pLength; i++) {
            int curLength = arrPrefix[i - 1];
#ifdef INFO
            std::cout << "Вычисление значения префикс-функции для символа " << P[i] << " под
номером " << i << "\n";
#endif
            while (curLength > 0 && (P[curLength] != P[i])) {   // если предыдущий суффикс
нельзя расширить, то рассматриваем суф-фикс меньшего размера
#ifdef INFO
                std::cout << "Предыдущий суффикс размера " << curLength << " нельзя расши-
рить.\n";
#endif
                curLength = arrPrefix[curLength - 1];
#ifdef INFO
                std::cout << "Рассмотрим новый суффикс меньшего раз-мера: " << curLength <<
'\n';
#endif
            }

            if (P[curLength] == P[i]) {    // проверка на расширение
#ifdef INFO
                std::cout << "Суффикс длинны " << curLength << " можно расширить.\n";
#endif
                curLength++;
#ifdef INFO
                std::cout << "Новый размер суффикса: " << curLength << '\n';
#endif
            }
#ifdef INFO
            std::cout << "Значения префикс-функции равно " << curLength << "\n\n";
#endif
            arrPrefix[i] = curLength;
        }
#ifdef INFO
        std::cout << "\nЗначения префикс-функции:\n";
        for (auto j : P) {
            std::cout << j << ' ';
        }
        std::cout << '\n';
        for (auto i : arrPrefix) {
            std::cout << i << ' ';
        }
        std::cout << '\n';
#endif
}


void KMP::AlgorithmKMP() {
```

```
    Prefix();
#ifdef INFO
    std::cout << "\n\nАлгоритм Кнута-Морриса-Пратта:\n\n";
#endif
    int tSize = T.size();
    int pSize = P.size();
    int pIndex = 0;
    int tIndex = 0;
    while (tIndex < tSize) {
        if (P[pIndex] == T[tIndex]) {  //если символы равны
#ifdef INFO
            std::cout << "Найдено совпадение " << pIndex << " символа образца " << P[pIn-
dex] << " и " << tIndex << " символа текста " << T[tIndex] << '\n';
#endif
            pIndex++;
            tIndex++;
            if (pIndex == pSize) {  //если вхождение найдено
#ifdef INFO
                std::cout << "Вхождение найдено, индекс равен " << tIndex - pIndex <<
"\n\n";
#endif
                answer.push_back(tIndex - pIndex);
                pIndex = arrPrefix[pIndex - 1]; //переход на по-зицию равную предпослед-
нему значению префикс-функции
            }
        }
        else if (pIndex == 0) // если сравнение с первым символом
            tIndex++;
        else  //если же по образцу продвинулись
            pIndex = arrPrefix[pIndex - 1];
    }
}


int main() {
    setlocale(LC_ALL, "Russian");
    KMP k;
    k = KMP();
    k.Read();
    k.AlgorithmKMP();
    k.PrintAnswer();
    return 0;
}
```

Файл lab4_2.cpp

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

#define INFO

class KMP {
private:
       std::string A, B;
       std::vector<int> arrPrefix;
       int answer;
public:
       KMP() {};
       void Read();
       void PrintAnswer();
       void Prefix();
       void CyclicShift();
};

void KMP::Read() {
```

```
#ifdef INFO
    std::cout << "Введите первую строку A:\n";
#endif
    std::cin >> A;
#ifdef INFO
    std::cout << "Введите вторую строку B:\n";
#endif
    std::cin >> B;
}


void KMP::PrintAnswer() {
#ifdef INFO
    std::cout << "\n\nОтвет:\n";
#endif
    if (answer == -1) {
#ifdef INFO
        std::cout << "A Не является циклическим сдвигом B.\n";
#endif
        std::cout << answer << '\n';
    }
    else {
#ifdef INFO
        std::cout << "A является циклическим сдвигом B.\n";
        std::cout << "Индекс начала строки B в A: ";
#endif
        std::cout << answer << '\n';
    }
}


void KMP::Prefix() {   //префикс-функция
#ifdef INFO
    std::cout << "\nПодсчет префикс-функции:\n\n";
#endif
    int bLength = B.length();
    arrPrefix = std::vector<int>(bLength);
    arrPrefix[0] = 0;
#ifdef INFO
    std::cout << "Значение префикс-функции для символа " << B[0] << " под номером " << 0
<< " равна " << 0 << '\n';
#endif
    for (int i = 1; i < bLength; i++) {
        int curLength = arrPrefix[i - 1];
#ifdef INFO
        std::cout << "Вычисление значения префикс-функции для символа " << B[i] << " под
номером " << i << "\n";
#endif

        while (curLength > 0 && (B[curLength] != B[i])) {   // если предыдущий суффикс
нельзя расширить, то рассматриваем суф-фикс меньшего размера
#ifdef INFO
            std::cout << "Предыдущий суффикс размера " << curLength << " нельзя расши-
рить.\n";
#endif
            curLength = arrPrefix[curLength - 1];
#ifdef INFO
            std::cout << "Рассмотрим новый суффикс меньшего раз-мера: " << curLength <<
'\n';
#endif
        }

        if (B[curLength] == B[i]) {    // проверка на расширение
#ifdef INFO
            std::cout << "Суффикс длинны " << curLength << " можно расширить.\n";
```

```
#endif
            curLength++;
#ifdef INFO
            std::cout << "Новый размер суффикса: " << curLength << '\n';
#endif
        }
#ifdef INFO
        std::cout << "Значение префикс-функции равно " << curLength << "\n\n";
#endif
        arrPrefix[i] = curLength;
    }
#ifdef INFO
    std::cout << "\nЗначения префикс-функции:\n";
    for (auto j : B) {
        std::cout << j << ' ';
    }
    std::cout << '\n';
    for (auto i : arrPrefix) {
        std::cout << i << ' ';
    }
    std::cout << '\n';
#endif
}


void KMP::CyclicShift() {
    if (A.length() != B.length()) {
#ifdef INFO
        std::cout << "Длинны строк не равны, значит это не цикли-ческий сдвиг.\n";
#endif
        answer = -1;
        return;
    }

    Prefix();
#ifdef INFO
    std::cout << "\n\nОпределение, является ли A циклическим сдвигом B:\n\n";
#endif
    int aLength = A.length();
    int curBLength = 0;
    for (int i = 0; i < aLength * 2; i++) { //поиск по удвоенной первой строке
        int j = i % aLength;
        if (B[curBLength] != A[j]) //если символы не равны
#ifdef INFO
            std::cout << "Несовпадение " << j << " символа строки A " << '(' << A[j] <<
')' << " и " << curBLength << " символа строки B " << '(' << B[curBLength] << ")";
#endif
        while (curBLength > 0 && B[curBLength] != A[j]) {
            curBLength = arrPrefix[curBLength - 1];
        }
        std::cout << '\n';
        if (B[curBLength] == A[j]) { //если символы равны
#ifdef INFO
            std::cout << "Найдено совпадение " << j << " символа строки A " << '(' <<
A[j] << ')' << " и " << curBLength << " символа строки B " << '(' << B[curBLength] <<
")";
#endif
            curBLength++;
        }

        if (curBLength == aLength) {  //если нашлось вхождение
            answer = i - curBLength + 1;
#ifdef INFO
            std::cout << "\n\nВхождение нашлось. Индекс равен " << answer << '\n';
#endif
```

```
            return;
        }
    }

    answer = -1;
    return;
}


int main() {
    setlocale(LC_ALL, "Russian");
    KMP k;
    k = KMP();
    k.Read();
    k.CyclicShift();
    k.PrintAnswer();
    return 0;
}
```