**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

**по дисциплине «Построение и анализ алгоритмов»**

**Тема: «Алгоритм Ахо-Корасик»**

Студент гр. 6381 \_\_\_\_\_\_\_\_\_\_\_\_\_ Афийчук И.И.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_ Филатов А.Ю.

Санкт-Петербург

2018

**Задание 1.**

Разработайте программу, решающую задачу точного поиска набора образцов.  
Вход:

Первая стрыока содержит текст (T,1≤|T|≤100000T).   
Вторая - число n (1≤n≤300), каждая следующая из n строк содержит шаблон из набора P={p1,…,pn}1≤|pi|≤75  
Все строки содержат символы из алфавита {A,C,G,T,N}

Выход:

Все вхождения образцов из P в T.

Каждое вхождение образца в текст представить в виде двух чисел - i p

Где i - позиция в тексте (нумерация начинается с 1), с которой начинается

вхождение образца с номером p (нумерация образцов начинается с 1).

Строки выхода должны быть отсортированы по возрастанию, сначала номера позиции, затем номера шаблона.

**Описание алгоритма.**

1. На вход подается текст и шаблоны, которые необходимо найти. На основе введенных шаблонов строится бор;
2. Продвигаясь по тексту, выполняем переход по бору, либо по суффиксной ссылке. Назовем суффиксной ссылкой вершины v указатель на вершину u, такую что строка u — наибольший cобственный суффикс строки v, или, если такой вершины нет в боре, то указатель на корень. В частности, ссылка из корня ведет в него же. Если из текущей вершины есть ребро c текущим символом, то пройдем по нему, иначе пройдем по суффиксной ссылке и запустимся рекурсивно от новой вершины;
3. Для оптимизации введены сжатые суффиксные ссылки, которые позволяют, двигаясь по суф. ссылкам, попадать только в заведомо имеющиеся среди шаблонов;
4. Если flag = true, то есть удалось найти шаблон в тексте, запоминаем позицию начала вхождения, номер шаблона и выводим их на экран;

**Описание функций и структур данных.**

**Описание функций:**

1. **void add\_string (const string& pattern, int num\_pattern, vector <Vertex> &bohr)** – добавление шаблона pattern в бор bohr;
2. **int get\_suff\_link (int v, vector <Vertex> &bohr)** – вычисление суффиксной ссылки;
3. **int get\_link(int v, char s\_char, vector <Vertex> &bohr) –** вычисление перехода;
4. **int get\_compress\_suff\_link( int v, vector <Vertex> &bohr) –** вычисление сжатой суффиксной ссылки;

**Описание структур данных:**

1. Структура вершины бора

struct Vertex {

int next\_vert[K]; //указатель на вершину, в которую ведет ребро, если не ведет -1

bool flag; // последняя посещенная вершина

char value;//символ, по которому мы пришли

int parent; // вершина родитель

int transition[K]; // массив переходов для суффиксных ссылок

int suff\_link; // суффиксная ссылка

int compressed\_suff\_link; //сжатая суффиксная ссылка

int length\_pat; //длина шаблона

vector <int> num\_pat; //номер шаблона

};

1. Структура ответа

struct Answer {

int num; // позиция шаблона

int pat; // номер шаблона

};

**Тестирование.**

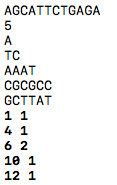
****

рис. 1

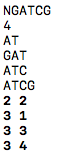
****

рис. 2

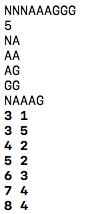


рис. 3

**Задание 2.**

Используя реализацию точного множественного поиска, решите задачу точного поиска для одного образца с джокером.

В шаблоне встречается специальный символ, именуемого джокером (wild card), который "совпадает" с любым символом. По заданному содержащему шаблоны образцу P необходимо найти все вхождения Р в текст Т.

Например, образец аb??с? с джокером ? встречается дважды в тексте xabvccbababcax.

Символ джокер не входит в алфавит, символы которого используются в T. Каждый джокер соответствует одному символу, а не подстроке неопределенной длины. В шаблоне входит хотя бы один символ не джокер, те шаблоны вида ??? недопустимы.

Все строки содержат символы из алфавита {A,C,G,T,N}  
Вход:

Текст (T,1≤|T|≤100000T)  
Шаблон (P,1≤|P|≤40)  
Символ джокера

Выход:

Строки с номерами позиций вхождений шаблона (каждая строка содержит только один номер).

Номера должны выводиться в порядке возрастания.

**Описание алгоритма.**

1. На вход подается текст, шаблон с джокерами и символ джокера. В шаблоне выполняется поиск максимальных подстрок, не содержащих символа джокера, с запоминаем позиции вхождения;
2. Из подстрок, не содержащих символа джокера, строится бор;
3. Алоритмом Ахо-Корасик выполняется поиск всех вхождений подстрок в исходном тексте;
4. Если подстрока найдена, то значение элемента массива С по индексу позиция\_вхождения\_в\_тексте-позиция\_вхождения\_в\_шаблоне+1 увеличивается на 1;
5. Если значение элемента в массиве С равно количеству подстрок без символа джокера в шаблоне, то выводим индекс этого элемента. Это значение будет являться началом вхождения шаблона в тексте;

**Тестирование.**

**../Снимок%20экрана%202018-05-31%20в%200.28.24.png**

рис. 4

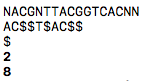


рис. 5

../Снимок%20экрана%202018-05-31%20в%200.29.10.png

рис. 6

**Выводы.**

В ходе лабораторной работы мы подробно ознакомились с алгоритмом Ахо-Корасик и научились решать задачу точного поиска набора образцов. В результате выполнения лабораторной работы были разработаны программы, которые вычисляют начальные позиции вхождения простых шаблонов в тексте, а также шаблонов, содержащих символ джокера.