**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 |  | Поляков Н. С. |
| Преподаватель |  | Филатов А. Ю. |

Санкт-Петербург

2018

**Цель работы.**

Изучение алгоритма Ахо-Корасик для поиска всех вхождений нескольких шаблонов в текст.

**Условия задания.**

1. Разработайте программу,  решающую задачу точного поиска набора образцов.
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 P в текст T. 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}.

**Входные данные.**

1. Первая строка – текст (T). Вторая строка – число шаблонов. 3 – n+2 строки – шаблоны.
2. Первая строка – текст (T). Вторая – шаблон (P). Третья – символ джокера.

**Выходные данные.**

1. Все индексы вхождения шаблонов с номерами шаблонов.
2. Все индексы вхождения шаблона.

**Описание структур данных.**

|  |  |  |
| --- | --- | --- |
| Класс | Имя | Назначение |
| Trie | Trie() | Конструктор |
| std::vector<std::pair<int,int> > result | Массив результата поиска |
| item \*root | Корень бора |
| std::vector<std::pair<int,int> > AK(const std::string&) | Функция, реализующая алгоритм Ахо-Корасик |
| void addReferences() | Функция добавляет суффиксные ссылки в бор |
| void addString(int number, const std::string&) | Добавляет строку в бор |
| item | std::vector<item\*> next | Массив указателей на сыновей |
| std::pair<int,int> number | Номер шаблона и его длина |
| char name | Символ |
| item\* has(char) | Возвращает указатель на сына с нужным именем |
| void addRef(item\*,std::deque< std::pair<item\*,item\*> > &) | Добавляет суффиксную ссылку к текущей вершине |
| item\* addItem(char) | Добавляет сына |
| item()  item(char,int)  item(char) | Конструкторы |
| item \*ref | Суффиксная ссылка |

**Описание алгоритма.**

1. Поиск подстрок в строке.

Для всех шаблонов строится бор, добавляются суффиксные ссылки. Изначально текущей вершиной является корень бора. На каждом шаге алгоритма рассматривается символ текста если у текущей вершины есть сын с таким именем, переходим к нему и на следующий шаг алгоритма. Если нет, то переходим по суффиксной ссылке и проверяем там и так далее до тех пор пока текущей вершиной не станет корень. Если в текущей вершине заканчивается какой-либо шаблон, добавляем его с соответствующем индексом в массив с результатом.

1. Поиск подстроки с джокером в строке.

Шаблон разбивается на подшаблоны, не содержащие джокеров и производится их поиск в строке. Далее создается массив целых чисел длинны строки и для каждого найденного подшаблона в индексе, элемент с индексом в котором должно было быть начало строки, инкрементируется. В этом массиве производится поиск числа, равного количеству подшаблонов. Эти индексы – ответ.

**Тестирование.**

1. Поиск подстрок в строке.

|  |  |
| --- | --- |
| Ввод | Вывод |
| NGATCG  4  AT  GAT  ATC  ATCG | 2 2  3 1  3 3  3 4 |
| AGCATTCTGAGA  5  A  TC  AAAT  CGCGCC  GCTTAT | 1 1  4 1  6 2  10 1  12 1 |

1. Поиск подстроки с джокером в строке.

|  |  |
| --- | --- |
| Ввод | Вывод |
| NACGNTTACGGTCACNN  AC$$T$AC$$  $ | 2  8 |
| AACGTGTNNAACGNNN  ACGT  $ | 2 |

**Вывод.**

В ходе выполнения данной лабораторной работы был изучен и реализован алгоритм Ахо-Корасик, решающий задачу точного поиска набора образцов.