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

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Построение и Анализ Алгоритмов»
Тема: «Алгоритм Ахо-Корасик»

Студент гр. 7304 _____ Петруненко Д.А

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург
2019

Цель работы

Освоить алгоритм Ахо – Карасик, который реализует поиск множества подстрок из словаря в данной строке.

Постановка задачи

1. Разработайте программу, решающую задачу точного поиска набора образцов.
    1.1 Входные данные: текст T, затем число подстрок N, далее N строк, которые нужно найти в T;
    1.2 Выходные данные: 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;

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.
    2.1 Входные данные: текст (T, $1 \leq |T| \leq 100000$), далее шаблон (P, $1 \leq |P| \leq 40$), после чего символ джокера; ν
    2.2 Выходные данные: Строки с номерами позиций вхождений шаблона (каждая строка содержит только один номер). Номера должны выводиться в порядке возрастания;

Ход работы

1. Алгоритм Ахо-Корасик

1.1. На основе набора паттернов строим префиксное дерево.

1.2. Рассматриваем префиксное дерево как конечный детерминированный автомат. Стартовая позиция в корне.

1.3. Считываем первый символ текста.

1.4. Переходим в следующее состояние по ребру, обозначающему этот символ. Если такого ребра нет, то идем по суффиксной ссылке. Если суффиксной ссылки нет, то запоминаем символ, по которому пришли в данный узел, берем суффиксную ссылку родителя и пытаемся перейти по этому символу. Данный шаг выполняется рекурсивно, пока не найден такой переход или не достигнут корень.

1.5. Выполняем шаг 4 до тех пор, пока не найдем валидный переход по текущему символу текста или пока не достигнем корня.

1.6. Проверяем является ли текущее состояние каким-либо паттерном. Для этого переходим по суффиксным ссылкам, проверяя соответствующий флаг. Если это паттерн, выводим его номер и позицию в тексте в консоль.

1.7. Если не конец текста переходим к шагу 3.

2. Алгоритм Ахо-Корасик + паттерн с джокером

2.1. Разбиваем паттерн на части, разделенные джокерами. Запоминаем их позицию в паттерне (индекс).

2.2. Используя алгоритм Ахо-Корасик ищем позицию этих паттернов в тексте.

2.3. Создаем массив нулей, размер которого совпадает с длиной текста.

2.4. Для каждого вхождения паттерна инкрементируем $i - j + 1$ позицию в массиве, где i – индекс вхождения, j – позиция данного паттерна в исходном паттерне.

2.5. Таким образом, в тех позициях массива, где значение совпадает с количеством паттернов, присутствует совпадение с исходным паттерном.

Результат работы алгоритма Ахо-Корасик

1. Входные данные:

ATATAT

1

AT

Выходные данные:

1 1

3 1

5 1

2. Входные данные:

ATTATNA

2

AT

TN

Выходные данные:

1 1

1 4

2 5

Результат работы алгоритма Ахо-Корасик с поиска шаблонов с масками

1. Входные данные:

ACAGA

A?A

?

Выходные данные:

1

3

2. Входные данные:

ATATNATNT

T

$

Выходные данные:

1

3

6

Вывод

В ходе выполнения данной лабораторной работы были изучен и реализован на языке программирования c++ алгоритм Ахо–Корасик, результатом работы которого являются индексы вхождений подстроки в тексте и номер этой подстроки. Для работы этого алгоритма понадобилась реализовать бор, содержащий все подстроки, которые необходимо найти. Бор был реализован в качестве конечного детерминированного автомата, для прохода по которому используются суффиксальные ссылки. Также, этот алгоритм был использован для поиска в строке паттерна, содержащего символ джокер.